

# TURBO C音乐编程指南



# TURBO C 音乐编程指南

李爰      赵华    编  
张明      王旭    审校

北京希望电脑公司

版权所有  
不许翻印  
违者必究

■北京市新闻出版局

准印证号: 3568—91568

■订购单位: 北京 8721 信箱资料部

■电 话: 2562329

■电 传: 01—2561057

■电 挂: 0755

■地 址: 海淀影剧院北侧

■乘 车: 320、332、302路海淀黄庄下车

■办公地点: 公司大楼 2 层

---

内部成本价: 17.00 元

## 内 容 提 要

本书是一本讲述广泛应用于合成乐及切分乐作曲领域的程序和函数的编程指导书，书中运用了大量的程序实例，来反映实验性音乐领域作曲家们在算法的研究成果。本书的程序分为六组，在每一组程序之前都有一段程序说明，以便读者能够熟悉该段程序的算法特性及适用范围。本书中论及的软件全部在 IBM-AT 环境下用 Boland's Turbo C 编成，因而为有志于用 C 语言开发音乐软件的才士创造了良好的条件。

## TURBO C 音 乐 编 程 指 南

李 雯	赵 华	编
张 明	王 旭	审校

## 目 录

前言 .....	1
简介 .....	2
第一章 音乐智能软件与 MIDI .....	9
概论 .....	9
MIDI 的应用 .....	14
MIDI 通信界面 .....	15
实际应用方面 .....	18
MIDI 实际音乐谱曲 .....	20
MIDI 基础 .....	21
信息传输 .....	25
MIDI 中的时间测量 .....	27
ASCII 文件格式 .....	28
从 MIDI 数据列到标准谱曲 .....	32
乐谱印刷软件 .....	32
计算机—MIDI 接口软件简介 .....	32
第二章 工具库 .....	34
函数组:Tunings.c .....	34
函数:Pitchtab( ) .....	36
函数组:Vectors.c .....	37
函数组:Curves.c .....	39
函数:Fractab( ) .....	42
函数:Durrod( ) .....	44
函数:Euereduc( ) .....	45
函数组:Fractsum.c .....	47
函数:Decfrac( ) .....	49
函数:Fracdec( ) .....	52
函数组:Decbidcc.c .....	53
函数:Stirling .....	57
函数组:Permutot.c .....	58
函数组:Combntot.c .....	61
函数组:Normtabl.c .....	63

函数组:Scaler.c .....	65
<b>第三章 数列与动态运算 .....</b>	<b>71</b>
函数组:Motforms.c .....	71
函数组:Displace.c .....	73
函数组:Altersp.c .....	75
函数组:Setflag.c .....	78
函数:Conshuff( ) .....	79
函数:Addshuff( ) .....	81
函数组:Rowforms.c .....	82
函数组:Rowsquar.c .....	85
函数组:Modops.c .....	87
函数组:Pstnperm.c .....	89
函数:Samplset( ) .....	92
函数组:Rotate.c .....	94
函数组:Timpoint.c .....	98
函数组:Alintseq.c .....	100
函数组:Alintset.c .....	103
函数组:Intlink.c .....	106
函数:Valratio( ) .....	109
函数组:Markov.c .....	110
函数组:Tropes.c .....	114
函数组:Mirror.c .....	117
函数组:Sets.c .....	124
函数组:Constel.c .....	147
<b>第四章 概率分布函数 .....</b>	<b>157</b>
函数组:Beta.c .....	157
函数:Bilexp( ) .....	160
函数:Cauchy( ) .....	162
函数:Expen( ) .....	163
函数:Gamma .....	165
函数:Gauss( ) .....	167
函数:Hypcos( ) .....	169
函数:Linear( ) .....	170
函数:Logistic( ) .....	172
函数:Poisson( ) .....	173

函数:Rnd-rnd( ) .....	175
函数组:Weibull.c .....	176
函数:Triangle( ) .....	178
<b>第五章 排序与查找 .....</b>	<b>181</b>
函数:Shellsrt( ) .....	181
函数:Shaksort( ) .....	183
函数:Quiksort( ) .....	186
函数:Insertst( ) .....	190
函数:Selecsrt( ) .....	193
函数:Tsearch1( ) .....	195
函数组:Tsearch2.c .....	197
函数组:Tsearch3.c .....	200
<b>第六章 歌词 / 配乐设计 .....</b>	<b>205</b>
函数组:Poem.c .....	205
函数组:Phone.c .....	209
函数组:Ttxparse.c .....	213
函数组:Drivel.c .....	224
<b>第七章 作曲总论 .....</b>	<b>229</b>
函数:Loopgen1( ) .....	229
函数:Loopgen2( ) .....	231
函数组:Primops.c .....	233
函数:Trantabl( ) .....	236
函数组:Valuprob.c .....	238
函数:Randwalk( ) .....	241
函数:Matwalk( ) .....	242
函数:Voss( ) .....	245
函数组:Rdintchd.c .....	246
函数:Octscale( ) .....	249
函数:Intgam( ) .....	252
函数组:Rhyprops.c .....	254
函数组:Polyrhy.c .....	258
函数:Meline( ) .....	262
函数:Partspan( ) .....	265

函数组:Ornament.c .....	269
函数组:Seqstore.c .....	271
函数组:Scroform.c.....	273
函数:Midifile( ).....	279
 附录 A 程序头文件(.h) .....	 283
附录 B 子程序库(.c) .....	284
附录 C 函数源文件(.pro) .....	315



## 前 言

编著本书的目的是为了展示当前日新月异，并广泛应用于合成乐及切分乐作曲领域的程序和函数。而且本书的内容将会日渐扩充，以反映实验性音乐领域的作曲家们在算法研究上的最新成果。

作为本书的初版，我们将选编入集的诸多计算机辅助作曲程序分为六组。在对每一组中的子程序块深入研究之前，我们希望读者先浏览一遍每一段程序的说明，以便熟悉该段程序的算法特性及适用范围。

因为本书是一本参考手册，所以具体应用及乐曲范例从略。我们的目标是在有限篇幅内展示诸多用途甚广的算法。至于有关具体的乐段的详情，将在即将出版的另一本计算方法作曲学的书中讨论。

本书论及的软件全部在 IBM PC-AT 环境下用 Boland's Turbo C 语言编成。函数编写尽量从简以使其结构清楚明了，并且引入了驱动程序，这样可以省去输入数据测试语句，从而大大精简程序。(但请注意，在人机对话的交互式应用中，需要在函数程序段中插入附加语句以验证用户。)

本书在出版过程中，得到了多方面人士的关心和支持，在此表示感谢，尤其得到了希望公司秦人华老师的大力支持，在此表示最诚挚的谢意。

编 者

## 简 介

本书中的 C 程序及函数分为六大类：1. 工具库；2. 数列与动态运算；3. 概率分布函数；4. 排序与查找；5. 歌词 / 配乐设计；6. 作曲总汇。注意，将某个算法归入某一类中仅仅是考虑到其表面用途。例如，在弦乐修改函数中的一段过程，可能同样可用于整数运算，只是由到采用了编码专用化，才使其当前效用仅在对相应的 ASCII 码对应的音符字母进行处理。

每一函数，或者说每一个函数组前都有一页前言说明以解释程序用途，提出应用建议以及编程方法。在此说明之后是程序清单，并附有运算结果。

本书编排顺序是为了适应其实用目的，即建立一个应用函数与过程的范例库。因此，虽然逐页新阅读本书效果很好，但并非必须如此。本书前页上的程序目录提供全书概况方便浏览本书的读者按图索骥。

大部分主程序和函数为“硬件实现”以实现其带状显示。因此，若希望将程序修改或交互式软件，则须加入函数接受用户输入数据及编码以测试终端输入数据，并为用户提供指导。数组也必须改变结构以适应变数据类型输出的要求。

## TURBO C

本书中所有程序都用 Borland's Turbo C 语言编成，且绝大多数程序稍加修改即可在其它 C 语言环境中运行。

## C 语言基础

此简介目的并非 C 语言入门指导，而是在于使读者能够看懂本书中的算法。当前市面上有成千上万的 C 语言编程指导书，我建议读者参阅其中一二加深了解 C 语言，实际上编译系统参考手册通常是最好的此类参考书。

本书中的函数可在几种不同环境下的编译系统中运行，如 UNIX 及 VMS 系统下的 NTSU's Vaxen，IBM PCs 下的 Turbo C 1.5 版本及 Microsoft C 5.0 版本，我们尽量提高编码的可移植性，并添加了一些新的函数的编码，这些编码甚至超出了许多编译系统的范围(如 fpower())，如果读者在编译系统库函数中找到这些新的函数，请自行增补。

## 函数

C 语言是一种高度结构化的程序设计语言，每种功能都由一独立的函数实现。函数就是实现某种功能的几行代码。有一特殊的函数称作 main 函数，包括所有其他函数。下面是一个 C 语言的简单例子：

```
main()
{
```

```
printf("Good morining, world");
}
```

"main"后面的括号通知编译系统这是一个函数。开始和末尾的大括号表明函数的开头与结尾。在此例中，main 函数调用另一 printf()函数。在其括号中，自变量传递至函数。main 函数不带自变量，而 printf 函数的自变量是一行字符。

用户可以从其程序段很明了地看到 main 函数的功能。但怎样才能看出 printf 函数的功能。通常程序编制者会在程序中定义所有函数。比如 printf 函数这种通用函数已在用户程序的外部文件中定义。编程者只须告诉编译系统其所在位置。有很多方法实现引入外部文件，比如在程序开头加入下列语句：

```
#include <stdio.h>
```

该语句告诉机器去查找名为 stdio.h 的文件并将其插入到当前程序的当前行。另一种方法是将函数压缩成目标代码并连接。这样可以使用户程序的编译更快，但连接时间加长。通常的作法是将某个大程序中的每个函数放在各自独立的文件中。某一个程序可被命名为 Main.c，另一个可为 Pitchconvert.c，以此类推。在不同的程序运行环境中将会有不同的程序以实现维护功能。例如有一名为 Make 的程序，它实现查找程序所用到的函数，并在需要改动函数时修改文件。

函数定义的几个要素如下：

- 1.函数说明
- 2.参数说明
- 3.变量说明
- 4.语句或函数调用
- 5.注释

## 函数说明

函数说明表明函数名称以及其返回值类型。例如：

```
int test()
```

这是一个名为 test 的函数的说明，其返回值为整型。在大多数编译系统中，返回值类型若不加说明，则自动定为整型。

## 参数说明

参数是由一个函数传递至另一函数的变量。其类型必须先作说明。

```
int test2(junk,array)
int junk;
float array[];
```

此处 fnunction test2()中有两个参数。"junk"和"array"，这两个参数的名称在其类型定义后的括号中给出。一个是整型变量，一个是实型变量。这种格式最普遍，可移植性出最强。更新的编译系统可支持一种类似于 pascal 语言中的格式，即参数类型说明包含在函数的

括号中，格式如下：

```
int test (int junk, float array[])
```

## 变量说明

函数中局部变量的数据类型必须加以说明。局部变量只在本函数中有效，在函数外失掉其原有值。

```
int test()
{
    int j;
    char c;
} /* end of function */
```

局部变量在起始大括号后加以定义。

## 语句

一个语句应包括变量、常量、关键字及运算符。关键字是 C 语言自定义的一部分。语言中总共只有 28 个关键字(不同版本的编译系统有不同的关键字数)。关键字和运算符将在后面列出。语句末尾必须有一分号。

```
j=128;
```

这是一条赋值语句。变量 j 通过运算符“=”被赋以值 128。

```
for (j = 0; j < 10; j++)
    puts("Vaffanculo!");
```

上例是一循环语句将某一字符串打印十遍。该语句包括一个关键字“for”，一个赋循环初值语句(j = 0)，循环集中(j < 10)以及循环步进语句(j++)。该“for”语句应被理解为：

```
    初始状态 j 置为 0;
    若 j 小于 10,
    j 加一
    并且执行下一条语句。
```

库函数 puts() 被调用了十次，每次参数相同。

## 注释行

注释行两头有“/\*”号和“/”号，如下：

```
/* this is a comment */
```

编译系统将跳过注释行。它们只用来记录源代码。注释行在调试程序时也很有用。注释行可以指出需要调试的语句段，以在编译系统中进行调试。

C 关键字如下：

auto	if
break	int
case	long
char	register
continue	return
default	short
do	sizeof
double	static
else	struct
entry	switch
extern	typedef
float	union
for	unsigned
goto	while

### 变量和函数类型

变量在被使用前必须先说明其数据类型。其类型表明它在内存中的存储情况。

C 数据类型定义如下:

int	整型
float	实型
double	双精度型
char	字符型
FILE *	文件指针
int *	指向整型量的指针
char *	指向字符型量的指针
float	指向实型量的指针
double *	指向双精度型量的指针
struct	结构型

void

volatile

auto

数据类型修饰符:

extern

far

near

huge

short

long

register  
signed  
unsigned  
static

下列程序用到一个整型变量:

```
main ()
{
    int j;
    for (j = 0; j < 10; j++)
        printf("What's up, Jack?");
}
```

在此例中, 我们用一条循环语句来将一个句子打印 10 遍. "for" 语句被读作: 置 j 为 0, 然后, 若 j < 10 执行下一语句并使 j 加 1. 在 for 语句中用到几个运算符. = 号和 < 号的用法跟在算术运算中相同. ++ 号使变量加 1. 必须注意赋给变量的值一定要跟该变量类型相配. (比如说, 一个整型变量应赋以整数值.)

运算符:

++	加 1
--	减 1
~	二进制反码
^	XOR 异或
	OR 位操作或
<<	左移
>>	右移
!	NOT 非
&&	AND 与
	OR 或
&	AND 位操作与
/	除法
+	加法
-	减法
==	等于
!=	不等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于
?:	if 如果
=	赋值
operator=	复合赋值

逗号运算符  
% 百分号  
点运算符  
-> 向量运算符

当一个整数有返回值时，它必须说明返回值类型，若不加说明则自动视为整型。

```
main()
{
    float value;
    float calculate();
    value = calculate();
} /* end of main function */
/*=====*/
float calculate()
{
    float a = 2.2;
    int b = 5;
    return (a * b);
} /* end of function */
/*=====*/
```

此处，我们说明 calculate() 函数的返回值为整型(这与其实际返回的值的类型一致)。

## 指针

指针即值为另一变量地址的变量，指针在用到的大的数据结构的文件中很重要，如下列：

```
main()
{
    int j;
    int *jp;

    j = 2;      /* j is assigned the value 2 */
    jp = &j;    /* jp is assigned the address of j */

    printf("The value of j is %d which is stored at location %u",
           j, jp);
    *jp = 10;    /* 10 is stored at the address contained in jp */
                /* this is the same as: j = 10; */
    printf("The value of j is %d which is stored at location %u",
           j, jp);
} /* end of main function */
```

在此例中，“j”被说明为整型，而“jp”为指向整型量的指针，“j”被赋以整型值 2，而“jp”被赋以变量 j 的存储地址。另一指针的用途是用在字符串中，一个字符串可以两种形式说明：一是以字符数组形式，另一形式是以指向字符量的指针说明。

```
char string[10];
char * string2;
```

此处“string”被说明为一个含有 10 个字符的数组，“string2”为一个指针。通常，若要以指针形式说明字符串，则须用下列语句为指针置内存空间：

```
string = (char *) malloc(80)
```

## 程序范例

```

/* comments are enclosed by star-slash combinations */
#include <stdio.h> /* preprocessor statements; stdio.h is
                  not in the current directory */
#include "array.c" /* array.c is in the current directory */
#define MAXDATA 500
int junk;          /* global variable declaration */

/* function prototype */
void TestFunction(int datarray[], int size);

main()              /* start of main function */
{
    register int j; /* local variable declarations */
    int *jp;
    int datarray[MAXDATA], num;
    char string[10];
    char ch;

    for(j = 0; j < 10; j++) /* loop with several statements */
    {
        printf("Enter a number "); /* function call */
        scanf("%d", &num);          /* the address of num is passed
                                     to scanf() */
        datarray[j] = num;          /* assign the value of num to
                                     datarray */
    }                               /* end loop */
    TestFunction(datarray, 10);     /* function call */
} /* end of main function */
/*=====*/
void TestFunction(datarray, size) /* definition of function */
int datarray[], size; /* types of parameters passed to
                       function */

{ /* body of function */
    register int j; /* local variable definition */

    for(j = 0; j < size; j++) /* loop through array */
        datarray[j] += 2;    /* add 2 to each element */
} /* end of function */
/*=====*/

```



## 第一章 音乐智能软件与 MIDI

什么是“音乐智能软件”？它将怎样服务于音乐家呢？这一定是音乐家们及爱好者首先希望弄清楚的问题。尤其是当他们被告知计算机在音乐中的作用跟在其它领域中一样大。这个问题的答案是：计算机适用于实现多样化的功能，它能满足科学家们的需要，同样也能使音乐家们获益。在运行时，它能表现出人类智能的特征。

精确定义音乐智能较为困难，如同精确定义人工智能一样。因为本身构成人类智能的技巧和才能就并未完全为人所理解。通过思考及仔细观察，表象的、逻辑的、认识的功能可以被辨识出来，譬如记忆、回想、比较、模仿、举例，以及分析数据类型的能力，但是包含更多灵感成份的创造性活动，潜意识仍然令人难以提供和明确定义。人类还无法更进一步理解其脑力活动的韵律节奏及其复杂过程的动态特性，因此，一种暂时性定义就足够了。

音乐智能就是研究如何使计算机更加有效地应用于音乐素材、结构、系统的组合与整理。

### 音乐智能的应用

虽然，计算机应用在艺术方面，无法象在科技领域那样收到立竿见影的效果。但是，其数据处理的过程与方法，不仅能对试图攻克癌症的生化学家或是研究微芯片净化的材料学家大有裨益，同样也能使艺术家感到得心应手。

计算机究竟能为音乐艺术做些什么？这门艺术向来都被视为人类灵泉的结晶，它果真能够被看作一种量化、系统编码化的模式来进行调试吗？就好象在研究一架宇宙飞船的计算机仿真模型？

虽然，艺术家的作品怎样才能感人至深还如谜一般难以琢磨(实际上，美学这门艺术价值系统的哲学也未能充分解释作品怎样才会被视为美，美在何处)，但值得注意的是不要把这种观念推广到音乐艺术领域的所有方面。

且不说作曲，几乎所有音乐的其它方面都须求助于计算。问题并不在于计算机能否应用于音乐，而在于怎样使其应用在音乐领域更有意义，以及怎样构筑一个最佳模式来实现应用。

### 概论

计算机辅助设计音乐可分为两大类：一类是音乐合成法造出合成乐(纯商业用途)；另一类是作曲法创作出的作品。在计算机音乐起步伊始，音乐合成和作曲法实际上是在同一工作环境下进行的——即主机，但是，随着时光推移，这两种方法逐渐分道扬镳，原因是

实时控制，微机控制演奏器的出现。今天，用的计算机就相当于作曲者(或演奏者)，而音乐合成器就相当于乐器，跟过去传统演奏方式并无多大差别。将来，随着科学技术日新月异的发展，可能会在同一机器上实现这些各不相同的功能，实际上这已经以机器内固化时序功能在一些音乐合成器内实现。这些装置可以使演奏者将所奏乐段存入内存，而延迟一段时间后再调出重播。这种内时序器相当于一个演奏控制中心，发出指令控制何时奏何种音符，音量大小及音长。

## 在音乐研究领域中的应用

研究音乐系统各种原则、方法的工作者们正在寻求一种途径来列出计算机如何解决音乐问题的方法。一些主要的研究课题如下：

1. 算法及自动作曲
2. 音乐理论专家系统
3. 计算机辅助设计
4. 乐理
5. 演奏
6. 心理声学(认识论)
7. 声学
8. 建筑声学
9. 演奏环境仿真
10. 仿生声学(声学环境)
11. 美学原理
12. 系统论、信息论
13. 多媒体开发(多学科交汇，包含多种信息传播媒介，譬如印刷品、录像、电影、激光音像、光雕、音乐)。

下面将讨论以上各领域详情。

## 作曲

作曲在音乐研究领域占首要地位，因为先将音乐结构构筑起来(通过模拟，设计方法)可以解决许多其它音乐研究方面的问题。这些方面都是很大程度上根据构筑好的音乐结构来决定的。在计算机工作之前，必须先设计出一个作曲目标，范例(感性结构)。本来这项工作是由作曲者来完成的。

下列是几种主要研究课题：

1. 计算方法作曲：数据转换过程应用于声学系统。
2. 自动生成乐曲：研究自定义，自动控制作曲系统，借助相应于某种乐曲格式的算法。
3. 实验性音乐系统与结构：系统概论，随机音乐及信息论。
4. 乐声内部结构研究：合成乐技术辅助研究音乐曲调。

**计算方法作曲** 作曲可定义为从众多音乐元素中挑选出一部分构成一件音乐作品的过程(这些音乐元素包括音高、节拍、曲调等)。

在作曲中,算法即关于一系列特定操作的指令。举一个简单的例子:创作一行乐句就是对计算机编程,使其按某种标准从同一音高的音符中选取出一个组合构成乐句。此算法写作伪指令如下:

1. 从C大调音程的音符中任选一个音符。
2. 查找程序中的择音原则部分以决定选出的音是否合适。
3. 以此类推,按择音标准继续选音。
4. 若选出的音符合标准,则加在已选出的音列之中,音列计数器增值,返回至1。
5. 若选出的音符不符合标准,返回至1。

不管是有意还是无意的,作曲者总是要借助于算法来作曲。但在从前他们必须完全以自己的脑力来计算,所以结果是许多丰富多彩的音乐结构和组合仅仅在作曲者脑中,因为将它们完完全全地整理出来,需要大量的时间和精力。

**共生** 早在十九世纪五十年代,在作曲家们构思音乐结构时,他们就开始把计算机用作精力无限的助手。在这些先驱者中,美国的 Lejaren A.Hiller 和法国的 Iannis Xenakis 为计算机辅助作曲作出巨大贡献,他们被当今音乐界视为计算机算法作曲之父。在他们的辛勤工作之下,有一点很快日渐引人注目,即计算机不仅能准确无误地构筑复杂的音乐结构,而且也是前人所无法预料得到的一种灵感源泉,通过推理性实验以及构筑新作品的模拟结构,作曲者将会发现一条能无限扩大其听觉想象力的途径。人和机器之间也由此而很快发展成为一种共生关系。

现在,最重要的作曲功能尚不能为微机控制的音乐合成器所支持。也许在将来,含有算法作曲功能的软硬件会上市,但现在的个人计算机,只是作为一种兼容各种计算机语言(BASIC,FORTRAN,C或PASCAL)的机器,还仅被现代作曲家们用为创作乐曲的工作环境。

## 音乐基础理论

当今音乐理论着重描述由不同历史时期的音乐家的作品构成的音乐系统。乐曲的不同格式被反复分析,整合以验证音乐系统理论。

乐理的实用性表现在它在专业音乐课程中的核心地位。它的主要目的是通过引导学生仿照古典音乐格式作曲从而加深对音乐的理解。自1975年以来,这种专业性很强的课程广受欢迎,所以在此领域中的计算机辅助设计也得到迅速发展。

下面是一些乐理研究课题:

1. 说明性理论系统的公式化
2. 开发验证理论性假设的分析工具
3. 设计专家系统以模拟古典音乐不同历史时期的主要音乐格式,如Palestrina,巴赫等。
4. 分析/综合整理专家系统

## 计算机辅助指导

研究音乐领域各个主要方面的工作者们都致力于计算机在教学方面的应用。随着指令系统的发展,教师将会有更多的时间花在更有意义的方面,而不是象从前那样耗费时间和精力去准备供训练和测验用的练习曲目。目前,软件化教学材料主要应用于乐理方面,但在不久的将来,它的应用将会扩大到演奏古典乐器的教学,甚至还可被用于查出演奏者可能会碰到的一些有关发音和音质方面的问题。

## 音乐学

音乐学可被看作音乐上的考古学。它范围最广,包含有音乐这门艺术的各个分支、各个方面。它主要研究以现代的观念、方式去理解古典音乐形式、演奏技巧,以及它们与当时文化背景的联系。

一些当代研究课题如下:

1. 比较学的应用(与课题中的乐理研究专家系统相关联)
2. 研究主要音乐格式的显著特征的相互参照索引
3. 比较法研究语音/音乐结构化模式

## 声音心理现象学

作曲者和理论研究者关心人类感官特性的各个方面。人类不仅由于感觉器官能力的限制无法感知到一些信号,而且外界信号与人脑对这些信号的反应之间的关系仍需进一步研究。例如,作曲者和理论研究者很关心长期记忆和短期记忆对于人们对一件艺术作品的感性认识所起的作用。目前,艺术家们都凭自己的直觉来判断人们对一件作品的看法,这当然是一种并不牢靠的办法。毫无疑问,在此领域的新发现将会大大增强作曲者的判断力,使其能够预测乐曲在听众之中引起的反响,从而去掉作曲过程中许多自己主观臆测的东西。

## 演奏环境模拟

直到最近,封闭式的奏曲环境声学还只是建筑声学家的天地。他们研究不同形状大小的厅堂和不同建筑材料的声学特性,以使音乐厅的设计能够产生音色、纯度及其它特性的最佳效果。而音乐家仅仅充当这些建筑声学家的顾问,用他们敏锐的听觉去测试厅堂的音乐特性,譬如声音弥散度、衰减特性、回声、声响特性、回声衰减时间等等。

几十年来,随着越来越先进的演奏空间类型模拟仿真电子设备的出现,音乐师在决定演奏环境的声学特性工作中起着日益重要的作用。由于在声学环境中应用了增音、拟音、听觉定位等理论进行研究,演奏环境本身的形状特性变得并不重要。我们可以在任何空间内模拟出各种不同形状、大小、声学特性的演奏环境。

许多公司都在推广这项新技术,先驱首推加尼福尼亚州的 Good Sound Foundation

ofodate. 该公司着重研究在音乐模块结构中建立数字或声音可变系统(其总经理为一作曲家)。Good Sound 公司完全由音乐家组成,并且坚信只有音乐家自己才能决定如何在不同传播媒介上演奏不同音乐格式的乐曲并力求最佳效果。

在过去,音乐厅为所有演奏者包括独奏者、室内乐队以及管弦乐队提供演奏场所。为了实现此目标,其设计不得不采用折衷方案,因此要使每一种演奏形式都达到最佳效果是不可能的,最多不过能使各种形式的演奏效果不至于太差。而现在,音乐家可通过调整演奏环境以适应不同演奏形式的需要——从钢琴独奏会所要求的温暖、亲近的气氛到一些管弦乐演奏会所需的较为干燥的声学环境。实际上,甚至可以实现在乐章与乐章之间改变演奏环境以更好表现出作曲家在乐章之中的创作意图。

### 声音生态学(声学环境中)

乍一看,这个研究领域仿佛与音乐家所关心的方面相去甚远,因为它主要研究噪声污染的分贝数、类型以及严重程度。但实际上因为音乐工作者比外行人有着更敏锐的听觉也更易为噪声污染所扰,所以他们在此问题上更有发言权。

我们可以对声音生态学进行逆向思维,譬如说周围噪声能够对人产生有害影响,那么,什么样的声音环境才能使人觉得舒适,无压抑感、工作效率高?这项课题也许对音乐家们更有吸引力。他们已经花费大量时间来提高自身对声音特性以及特定声学环境的敏感程度。

### 信息论和美学概念

这个课题的研究始于1955年,当时出版了 Abraham Moles 博士的“信息论和美学概念”(1966,伊利诺斯大学出版)。这本书为美学领域的一种新研究方法奠定了基础。这种方法以通常用于数学随机理论和微积分的分析手段来研究艺术作品。因此,它可以被称不为研究时间变换过程的量化与分类以及音乐体验的层次的自相关的信息理论。理论上,从某一感官接受到的可以被量化的信息与此信息所带有的内在含义之间有一条明显的界限。下面举一例以加深理解这一观点,试想国际标准摩尔斯代码的打点传送法,对于一个对解译电码一窍不通的人来说,一行电码仅仅是代表着一些重复的点,然而对于一个受过此类专门训练的人而言,这行电码就代表着特定的含义,解译它只须借助于发电者与接受者共同承认的一种事先标准化了的规则。

### 研究成果的实际应用

纯理论研究会开发出一些实用产品和装置以完成日常生活中形形色色的工作。在过去的几十年里,音像市场就大大受益于此。

在十九世纪七十年代,John Chowning 博士,斯坦福大学计算机音乐研究者,为提高音乐制品生产力作出巨大贡献。他的有关 FM 声音合成技术方面的专业论著导致了高效数字式发声设备产生。主要营销这种产品的是许多 MIDI 音乐合成器制造商,比如

Yamaha 公司的 Tx81-Z 和 New England Digital Corp 的 Synclavier II。

复音(多音对位)、复调(多声调)音乐合成器给商业化音乐制品的各个方面带来了一场革命。因为它们能相当逼真地模仿出传统乐器的声音,比如小提琴、单簧管和小号。实际上,在聆听广播或电视中的音乐时,甚至专业音乐工作者也很难区分究竟是发自乐器的自然声音还是音乐合成器产生的合成乐。

乐曲的谱曲与改编工作在当今科技发展之下大大受益。在尚无 MIDI 技术的时代,乐曲改编者和谱曲者通常要等很长一段时间才能为将为音像电影制品谱写的曲子灌制成品。由于改编者能否继续被雇用常常取决于制造商对他的一组音乐制品是否满意,所以很显然,乐曲改编者会尽量去做到因循过去成功的作品的技巧,以求稳妥。因而导致了充斥市场的写满陈辞滥调的音乐手册讨论、“通用乐曲改编”之类问题。

MIDI 一交互式音乐合成器的发明消除了乐曲从谱写到制成成品之间的时间间隔,从而使当今谱曲者和改编者可以大胆改革,推陈出新,而不必担心会丢掉饭碗。所有乐曲都可以先在谱曲人自己的工作室里预演和评估,而不必马上交由制造商审定。

**教学环境** 自 1982 年以来(MIDI 技术面市后),飞速发展的数字式音乐处理软件、硬件带来了一项重要成果,即音乐制品和练习材料可以很方便地在教学过程中运用。以前,诸如和声、对位、配器之类的课程很大程度上决定于教师自身在钢琴上演奏曲目的能力。他们通过演奏一支又一支曲子来指导学生理解不同乐器奏出的乐曲或是合唱曲。而在今天,诸如此类的范例曲子可以很轻松地由教学用数字式硬件、软件奏出。这项技术影响之广遍及音乐课程的各门课目。

## 将来的研究开发工作

虽然现在还很预测在未来的日子里计算机音乐研究将会走向何方,下列领域的产品开发是毫无疑问的:

1. 乐谱扫描仪,以及应用于解释不同的乐谱结构并自动在音乐合成器上奏乐的软件。
2. 软、硬件应用于对各种乐器组合奏出的乐音进行数字式采样并转变或乐曲段落。
3. 复合音乐合成器,固化于个人计算机硬盘中。
4. 与娱乐业音像制品配套的自动作曲软件(即可实现同时谱写几种不同音乐格式的曲子的软件)。

## MIDI 的应用

MIDI,其奏乐装置是数字式接口,原本是设计用来实现音乐合成器之间的数据传输。在十九世纪七十年代后期,流行音乐师们求助于乐器制造商,希望他们能精简电子乐器键盘上不同形状的键。因为他们为了力求产生各种音响效果不得不把笨重复杂的乐器从一场音乐会的地点搬到另一场音乐会的地点。通常一种音乐合成器产生一种乐器的音响效果。而摇滚乐师为了造出丰富多彩的音响效果就不得不在舞台上摆上一长列各种各样的乐器,然后演出时在舞台上跑来跑去地演奏。

在 1982 年, 流行音乐师们的难题得到了解决, 这是缘于产生了 MIDI 试用型——即电子乐器制造者共同遵循的一个接口通信标准。按此标准制造乐器, 制造商可以肯定其产品能够与其它制造商生产的电子兼容。

演奏者们于是可以在一个键盘上演奏一段乐句而听到它在几个 MIDI 合成器上同时奏出, 它们是通过数据接口联接在一起, 当在主键盘上按下一键时, 其数字信息(包括是哪一键, 按键时间)被传送到终端音乐合成器, 从而使它发声。

MIDI 一经出现, 很快就发展到引入个人计算机作为各音乐合成器连接的中心控制部件, 并且很快开发出软件应用于录制、存储、编辑, 以及重放长的音乐片段, 由此便为演奏者们开辟了一片新天地, 即用一个键盘就可以达到一个乐队的效果。和声和对位可以事先存储在磁盘文件中, 演奏中再在恰当时刻播放出, 从而免得乐师们在演奏时总要费力去记忆何时该去按哪一个乐器上的键。

大约在 1985 年, 软件设计者们就开始编制程序来将 ASCII 数据文件(根据计算机作曲的简法用 C 或 Pascal 语言编制的程序)转换成为音乐合成器可演奏的时序文件。虽然开始只是研究者和大学里的作曲家对此感兴趣, 但随着这种软件的普及越来越多的流行音乐师也加入到研究它的行列中来, 而且, 它将会加速自动作曲软件占据流行音乐市场。

## MIDI 通信界面:

### 计算机内的音乐表现手法

现在的乐谱是一系列逻辑性的、代表的、音乐元素的组合(称作象素)——是一种经过千百年的演变而成的很难以掌握的复杂组合。虽然, 通过演奏出乐谱上的乐曲可以使听众领会到作曲家的创作思想, 但这一点在计算机上却很难行得通, 因为人与计算机之间的唯一沟通方法是通过计算机语言。

因为计算机实际上是一个由许多与非门组成的复杂网络(这些网络上的网点只有两种状态, 低电平、高电平或称 0 状态、1 状态), 所以首要问题是先解决如何将音乐信息转变为二进制代码。这种转换并不容易。我们可以先从两方面着手: 理论方面和实际应用方面。

### 理论方面

计算机的操作过程跟人脑大致相同。其实, 此过程与人的逻辑推理过程和接受音乐信息的方式毫不相干。人们先从各个角度来对待一种音乐格式、结构, 然后依据自身生理、心理、智力上的偏好来分类、整理、归纳各种音乐作品。因此, 人类构筑音乐结构以及理解此结构的过程是经过“预计算”的。尽管将计算机应用于作曲的计算之中至关重要, 它实际上也只是人与这一与非门网络之间相互利用的一个独立步骤。而且, 音乐概念化的方式方法又通常取决于计算机用户的意向。

人们在为音符研制出一种有效的概念化结构模式方面作了大量工作。许多新观念的出现有利于更精确地在计算机上模拟各种音乐格式、结构的乐曲。这些模拟法分为两大类:

象征性符号方法和程序化方法。

### 象征性符号(语法的)方法

这种理论将音乐结构看作是一种音乐语言的代表,或者说是音符和一些乐符按某种方法规则组成的结构。这种音乐语言的语法由其修辞学用所规定——其修辞学即层次分明的、相互作用的一组原则来控制在作曲中选取合适的韵律以及和声。通常,一个庞大的数据库即可容纳一种音乐格式的所有修辞原则以便作曲时参考,这种(语言法)研究方法是音乐理论、音乐学领域人工智能研究者的课题,而且至少会令设计、分析和充实其作曲风格的专家系统的作曲家们颇感兴趣。

**乐理与作曲** 乐理学家和音乐学家对音乐结构的认识同作曲家并不相同。作曲家主要致力于创作,而分析者则旨在寻求一件作品的内在原则。例如,这两种不同认识的分歧表现在所谓的普通应用时期(大约 1650—1900)的“超越时代”音乐结构观点。实用主义的观点占据了音乐结构理论研究以至于任何其他形式都无从立足,在此阶段,标志着形式与作品结构之间关系的分析性、陈述性方法表明了音符在音乐结构中的标识、传递信息作用。当人们作曲时,参考音乐语法、句法、以及修辞时,总是在很大程度上依赖于弦结构的音符代号,以及和弦内各音与本音程内的音,下一音程内的音,以及整个音域各音程之间的相互关系。下面是最常用的几种和弦音表示法:

I-IV-ii-V-iii-vi-IV-V-I

这些简单的符号在专业音乐工作者脑中就是一系列优雅的和声,此分级方法亦可应用于设计分析、改编普通应用时期的音乐形式的专家系统。这种分级系统提供了一种宏观上的“句柄”以便对主要音乐格式的基本特征进行编码建立数据库,从而大大优于以流式文件形式存储的方法(因为音乐本身结构并不符合通常的统计规律,因此将音乐结构用概率统计方法存储在流式文件中毫无意义)。简而言之,所有的分析方法都必须着重研究音乐的层次结构、各部分之间的关系,以及按自然语言结构建立起来的音乐结构。

### 程式化(构词的)音乐符号

与象征性(语法的)的音乐符号形成鲜明的对照,程式化的乐符将计算机作为作曲家脑力的延续。计算机可以根据作曲家的指令来处理复杂的音乐过程,并且可以大大提高他们谱曲的速度。而在以前,作曲家们通常要耗费毕生精力努力把构思好的曲子用笔写下来。当然这种应用要求计算机能够或多或少识别作曲家的意图,机器只需按作曲者指令完成各项操作,有时也会包含一些分析任务,但决非要求机器本身有意识,或是希望它成为能够谱写多种音乐格式的作曲家。

尽管象征性音乐符号对乐理学家及音乐学家编制程序至关重要,作曲家却常常应用计算机把象征性的、概念性的音乐符号跟其所代表的乐音象素有意识地分开。因为作曲家们不希望拘泥于一种唯一目标,格式固定的程序来作曲,而是努力寻求一种独立于计算机之外的音乐系统作为指导思想,同时作曲家还希望将作曲过程细分为几个不同阶段。

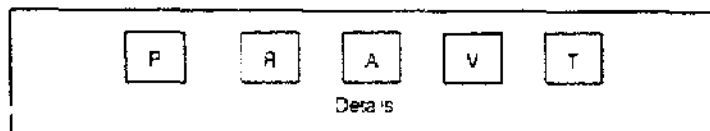
由此可见,作曲家的直觉力就决定了他们会以一种完全不同于其他音乐工作者的方式



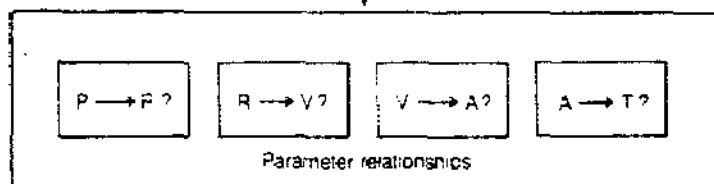
运用计算机。

以上所述的作曲家运用的程式化音乐符号的方法的一个必然结果就是：作曲家本人将能够控制也不能不亲自完成作曲的收尾工作。计算机所做的仅是分析处理数据，然后再输出大量数据代表一些特定的音乐含义。所以，机器只充当了一个“黑匣子”，它根据程序开头输入的数据以及控制参数(通常是相互关联的)来完成一项项操作。下图 1-1 所示了这一过程。

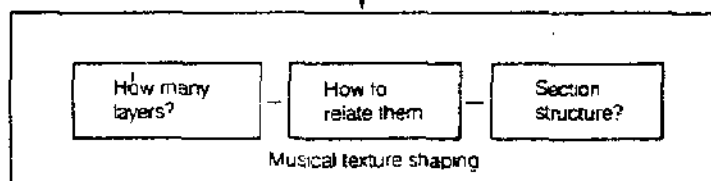
I. Composer provides low-level input data governing musical parameter ranges for pitch, rhythm, volume, timbre (instrument choice).



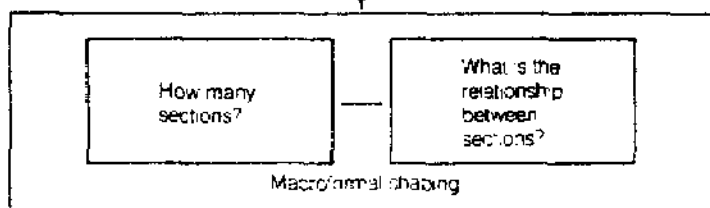
II. Composer provides middle-level input data governing relationships between constituent musical parameters.



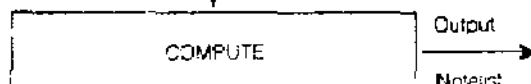
III. Composer provides high-level input data governing relationships between layers of the musical texture.



IV. Composer provides input data governing format of output data (music score).



V. Computer processes input data, generates a computer program, and outputs score.



注:

- I. 作曲者输入初级数据作为参数控制音高、节拍、音量、音质(选择乐器)。
- II. 作曲者输入中级数据控制各参数间关系。
- III. 作曲者输入高级数据控制音乐结构各层次间关系。
- IV. 作曲者输入数据说明输出格式(以音符输出)。
- V. 计算机处理输入数据、计算、生成最终乐谱。

图 1-1 计算机音乐作曲流程图

## 实际应用方面

用具体化语言及范例来阐述音乐系统跟用乐谱来记录音乐大相径庭。它们必须直观实用,而且要将主干音乐要素隔离、分类(这些音乐要素常被称为参数)。

## 音乐数据编码

**音乐参数** 作曲者用参数这一名称指代所有音乐结构中能被分离出并用一系列固定值来规定级别的量化因素,初级参数相当于音乐中的“基本元素”,诸如:音高、节拍、发音方式、力度强弱、音调等。通常这些初级参数以某种形式相互制约,然后作为一组数据存入内存,称作“向量”。一个向量就可以完全定义一个音的各个方面。例如,一个含有 5 个参数的向量如下所示:

音符项目 #1

音高	音长	音量	发音方式	音质
C#3	8 拍	fff	间断音	长笛

作为计算机术语,向量就是一个多维数组,用以存贮某一音项音乐变量的值。数组元素的地址以及各音项用正整数 1 到  $n$  来代表,从而构成了一弦音项表,如图 1-2 所示。一个二维数组中存入了一个度(长度)为 10 的音项表的所有音高、节拍参数。

音项										
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
音高	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
节拍	C	D	E	F	G	G#	A	A#	B	C
	Q	H	E	E	E	E	S	S	H	W

节拍说明: W = 全音 H = 半音 Q = 4/1 音 E = 1/8 音 S = 1/16 音

图 1-2 一个二维数组存入一个长度

同样,比音项初级参数级别更高的参数也可被定为一列标准量值。例如,西方现代音乐中的一个高级参数为“结构密度”,它实质上是按作曲家用于规定在演奏中所使用的乐器种类的数目。

在 MIDI 系统的应用过程中, 其软件输出的计算机算法作曲音乐数据通常是单个的象素形式而不是一个个的象征性符号, 而且还包含一系列根据音乐控制参数值标定的数据。

最后的“乐谱”——即输出表, 将用一种有利于在某种乐器上演奏的格式。一种常用的格式(有时称作音项向量表)是音项表形式: 即从开头从未尾, 按奏响时间先后次序排好的一系列音符。用此方法, 每个音项都可看作在某个特定时刻由各项音乐参数综合起来构成的坐标点。在前面已经提到, 实际上音项向量就是存储一个音项各种参数值的内存地址。

在实际应用中, MIDI 音乐合成器的数据输出格式可以只是代表上述各种阿拉伯数字字母的一系列数字。例如, 上述音项 NOTE-EVENT #1, 可能会显示如下:

61 8 127 .50 18

尽管音乐家可能将一张音项向量表称作乐谱感到很不习惯, 但事实上音项表能体现古典乐谱的一切特征。而且它的格式更适合于在电子乐器上演奏, 当然, 从阅读人的角度看, 这种格式不易看懂而且使分析各种数据变得很困难。这仅仅是由于它用一行显示一个音的各项特征。如图 1-3 所示。

为了解译这种音项向量表, 首要问题是先将音乐的各个部分加以分离。例如, 两种不同的乐音编码在一起, 第一种乐音用 1 声道, 第二种用 2 声道。因为这两种乐音只是交替发声, 我们很快可以明白就是一种音乐上的混音效果, 即不同声道的乐音交替发声, 奏出同一段旋律。但在古典乐谱中, 这一点是显而易见的, 无须推想。还有, 和声和终止在乐谱中是同一时刻, 所以在音项向量表中很难将二者分离开来。

解决这个问题的方法之一, 即设计出可以将音项向量表格式转换成古典乐谱形式的软件。现在, 我们可以用作曲程序来谱写复杂的音项组合, 而且同时并行运用数字式向量表格式和音乐符号格式。将这一原理反过来应用, 即可设计出乐谱扫描仪, 将铅印的古典乐谱自动转换为数字代码并在音乐合成器上演奏出来。

目前, 这种新乐谱形式的重要性还未完全显示出来。人们已经习惯于那些由于受人体演奏能力限制而缺乏复合韵律和丰富的和声结构的曲谱。与此相反, 计算机运用数字系统谱出的曲子中会有许多变化多样的韵律结构, 而这些是人很难用手演奏出来的。对于计算机而言, 一位钢琴家用手能够奏出的东西就少得可怜, 不仅是由于人手指数目限制了能够同时奏响的音数, 而且还因为人的手指都紧挨在一起, 从而不能想象它们能奏出相距很远的键的音构成的复和声。在一架 88 键的电子乐器上, 计算机相当于一个长着 88 根手指的演奏者, 可以在键盘上游刃有余。尽管能同时奏多个音并不能说明奏出的曲子就一定美妙, 但至少作曲者可以利用这一优点来放开思路构思更绚丽多彩的和声结构。

音项向量表说明: Filed1—乐谱起始时间; 192 个时钟信号 = 1/4 音

Filed2—MIDI 音高表(0-127); 中调 C = 60

Filed3—音量(声音大小)(0-127); mf 音量 = 64

Filed4—音长(以时钟信号次数为单位)

Filed5—乐器

(以下数字及题头仅供参考)

Event Number	S T	P	V	D	T
1	200	60	68	40	1
2	248	65	120	40	1
3	296	62	120	40	2
4	344	68	68	40	2
5	392	64	120	40	1
6	440	71	120	40	1
7	488	69	120	40	2
8	536	70	68	40	2
9	584	61	120	40	1
10	632	63	120	40	1
11	680	67	120	40	2
12	728	66	68	40	2
13	776	66	120	40	1
14	824	60	68	40	1
15	872	71	68	40	2
16	920	65	68	40	2
17	968	68	120	40	1
18	1016	64	120	40	1
19	1064	62	120	40	2
20	1112	63	120	40	2
21	1160	70	120	40	1
22	1208	67	68	40	1
23	1256	69	120	40	2
24	1304	61	120	40	2
25	1352	66	120	40	1
26	1400	66	0	40	1
27	1448	60	68	40	2
28	1496	67	120	40	2
29	1544	65	120	40	1
30	1592	65	120	40	1
31	1640	65	120	40	1
32	1688	65	120	40	1
33	1736	65	120	40	1
34	1784	65	120	40	1
35	1832	65	120	40	1

图 1-3 音项向量表

人们也许会问一个问题，既然音符是用来将作曲者的他们意图传送给演奏者，那为什么还要将计算机输出的音符项向量表转译成传统的乐谱形式？二者功能毫无二致，答案在于这样有利于人们阅读，并分析曲谱。长期以来，人们已经习惯了传统的乐谱形式，如果换成其它形式，譬如完全的图表式符号，即便能替代过去的乐谱形式的大部分功能，习惯上人们还是难以接受的。

## MIDI 实现音乐谱曲

当前市场上有很多有关用于标准录音、重放的 MIDI 时序软件的详细介绍，或此类杂志，而本章主要重点放在计算机算法作曲，因此在此不对 MIDI 通信技术作详细讨论。有关详情可参阅这方面的参考书目。

下面的阐述旨在为读者提供一些有关运用 MIDI 音乐合成器以及由制造商配套提供的 ASCII-MIDI 音项表转换软件进行算法作曲的背景知识(此类软件有 TOP-M.EXE 转换软件，出自美国的 SDAD 公司，配套用于其 PSS(Promidi Studio System) MIDI 接口卡)。

本章的目的在于阐述当作曲的所有预备工作完成后对传入音乐合成器中的音乐数据进行运算、谱曲。如果用户希望了解在 MIDI 界面实时控制下实现位操作或字节运算的详情，则需要更深入研究你的计算机中 MIDI 界面单元的寻址方式的细节问题。我们建

议运用 C 语言来实现此功能, 因为 C 程序很容易与汇编语言程序链接。关于这方面的深入研究, 请参看:

C Programming for MIDI by Jim Conger, 1988

(ISBN 0-934375-86-0)

M & T Publishing Company, Redwood City, California.

## MIDI 基础

MIDI——即电子数字式接口——产生于 1982 年, 提供了一种各乐器间通信连接的途径。目前, 数字技术飞速发展, 大多数音乐合成器由微处理机(计算机)实施控制, 而且为适应流行音乐师的各种不同需要, 制造商们已开发出许多种不同类型的通用、专用音乐合成器。因此, 寻求一种在各电子乐器之间通信连接的途径就显得至关重要。

在一些初步的、各自单干的尝试后, 人们一致要求合成器制造商建立一种标准的计算机音乐接口通信协议(操作方法), 以便在设计各种配件时遵循此标准。这样, 各种不同型号、类别的电子乐器在演奏中就可以做到协调统一, 而不至出现通信接口不匹配的问题。

1983 年, 日本制造商将第一代 MIDI 键盘式音乐合成器投放市场。自此, 这种技术逐渐更新换代, 日益完善。1988 年, MIDI 制造者协会通过了一项完整的通信电子乐器间 MIDI 通信接口的软硬件标准通信协议, 并将此标准扩展到用户计算机上。

在作曲过程中引入微计算机是一项创举。计算机有极强功能处理和应用外部软件, 从而大大增加了谱写一系列音符结构的能力。在初期的数字式音乐合成器中, 大部分控制软件是固化在合成器中, 用来调整音高或存储数据块。总之, 软件仅仅服务于它被固化于其中的音乐合成器, 而无法公用。随着个人计算机的出现, 电子乐器在作曲、演奏方面起了天翻地覆的变化。现在的计算机音乐系统由两个子系统组成: 作曲/录制/控制终端(微机)和演奏终端(音乐合成器)。

## MIDI 硬件配置

**实时系统** 图 1-4 所示是一种连接个人计算机与配有 MIDI 接口的音乐合成器的方法: 菊花链 MIDI 音乐合成器有 3 个主通信端口。(1)端口用作源端口发送 MIDI 数据; (2)端口接收 MIDI 数据; (3)端口传输 MIDI 数据至其他合成器。这种硬件配置方式便于灵活选用各音乐合成器的数据文件的一部分或全部。例如, 图 1-4 中的合成器#1, 由程序控制只接收从连接计算机与键盘之间的 MIDI 通道传来的信息; 合成器#2, 则可以接收其它 MIDI 信号, 这一点是可以实现的, 因为输入合成器#1 的数据可以完全不变地传送至合成器#2, 并不因为经过了一个合成器而有任何改变。

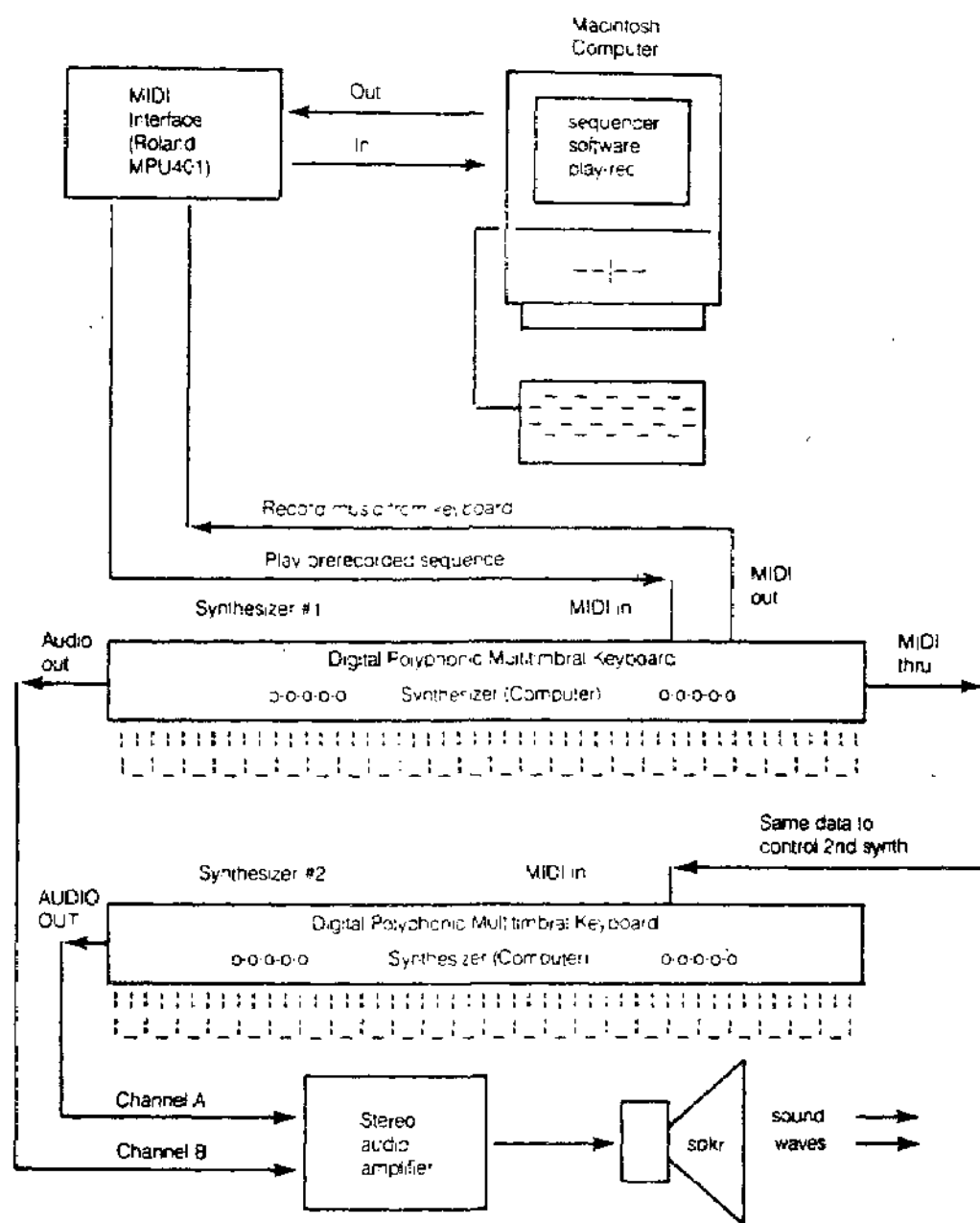


图 1-4 PC 机和 MIDI 音乐合成器的接口范例

演奏软件个人计算机的一个主要功能就是为各种软件包提供运行环境，而这些软件大多是用数字信息记录器、音符(音符段)库，以及乐谱编辑文件。微计算机的大量内存空间使得许多操作可同时运行，包括在音乐合成器上演奏乐段，以及将其存入内存或磁盘以便下次演奏时重新调用。在商业化流行音乐制造工业中，计算机主要应用在上述方面，下

图 1-5 显示了作曲人、商用软件包和键盘式音乐合成器之间的关系。

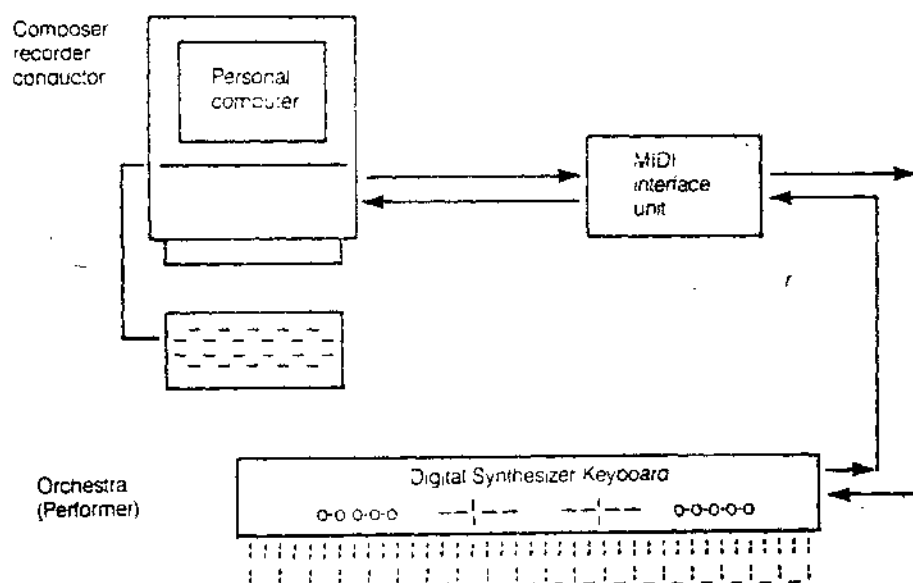


图 1-5 作曲者-软件-音乐合成器相互作用关系

**作曲软件** 自从八十年代 MIDI 问世以来，算法作曲已为学院派和流行音乐家所接受。虽然市场上有许多作曲软件包，想要有效运用它们还需要经过人的设计规划和增删（这些软件包通常用 BASIC, PASCAL, C 或 FORTRAN 语言编成）。

虽然算法在实时控制中可用于在计算机程序与音乐合成器之间传送、整理数据，它却无法实现复杂的处理、修改、增删音等操作，而这些都是需要作曲过程与演奏分离开来进行。图 1-6 所示是作曲者应用计算机辅助作曲的通常过程：

人作曲	计算机辅助作曲
1.构思	1.指定算法
2.速写	2.编程
3.在钢琴上试奏	3.生成样本文件
4.修改	4.修改
5.誊写终稿	5.生成最终乐谱

图 1-6 传统人工作曲过程与计算机辅助作曲过程的对比

在过去，作曲者在把终稿交给演奏者演奏之前，总是要先经过考虑、修改、试奏、以及最后精修点缀，事实上作曲人一直是在借助算术来作曲，不同的只是过去是用脑子作算术，而现在是用计算机。因此，大家可以很容易地看出上述人工作曲与计算机辅助作曲的对比中每一并列步骤实质上是相同的。

图 1-7 所示是一幅作曲者与软件及乐器之间的关系图。注意，图中的乐器已由键盘

或音乐合成器换成了无键电子发音器。因为此处的重点不是在于怎样通过实时数据控制在键盘上奏出音乐，而是着重于怎样运用算法谱写。

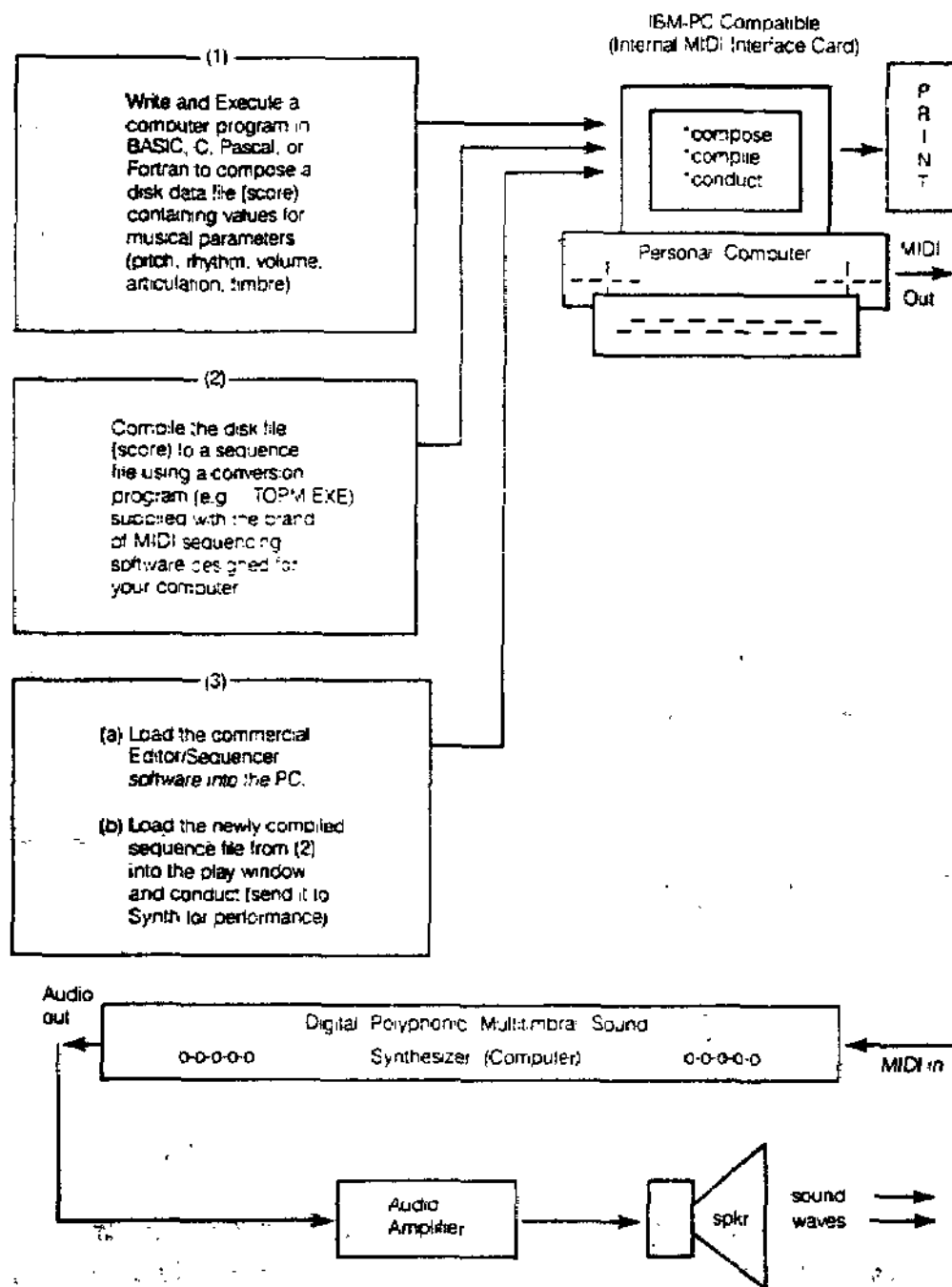


图 1-7 应用作曲算法，为电子乐器谱曲示意图



201-10

**通信接口硬件** MIDI 通信接口软件的核心部件是计算机—MIDI 接口单元。这种单元可有两种型式：一种是接口硬件被固化于计算机或电子乐器之外的黑盒中，另一种是可以插入用户计算机接口槽的卡。在两种型式的连接中，计算机、接口、乐器之间是通过一系列的传输线连接，这些电缆线从计算机通信端口上引出，连接到接口和电子乐器上。

MIDI 单元可实现不同程度的智能化功能，即，简单一些的单元可能仅仅是一条传送信息的通道，复杂一些的单元还可能包含一些逻辑电路，以实现更复杂的处理、时序功能。当然，在任何时候，单独的 MIDI 硬件是毫无用处的，它必须跟一个控制软件包配套使用才能实现自动组合、整理、传输信息。

## 信息传输

**软件** 通信接口实质上就是连接一系列计算机通信端口和音乐合成器的 MIDI 端口的中心控制以及数码解码器模块。计算机—MIDI 端口分为两大类：智能型和非智能型，其名称旨在说明各类型的端口的处理过程不同，并不表明其实用性强弱。

有一点值得引起注意，即软件与接口是否在很大程度上占用 RAM 空间存储音项向量表——音项向量表是一系列音项数据的组合，包含：音高、节拍、发音方式、音量、音质等等。这一点很重要，因为计算机内存中大量的时序/编辑文件占用了许多有限的 RAM 内存空间。如果这些文件也处理输入的 MIDI 数据并存入 RAM 中，那么，在将音项向量表用编辑软件作最后连接之前，RAM 中能存储的向量就很有限。

因此，建议使用 PSS(Promidi Studio System)系统来解决这一问题，此系统用于接口和软件中的编辑、建档、数据排序。它应用直接存储时序文件方式存储数据，再辅以计算机硬盘内存空间，使得音项向量表的长度可以无限加长。而且，PSS 系统的另一优点是可实现高精度节拍衰减，以及高级编辑手段。

**MIDI“语言”** 即使用于传送数据的 MIDI 通信协议可被称作语言，它也只可能是一种所有计算机上都应用的低级语言——机器语言。只有位模式(称做字节)通过 MIDI 电缆线从计算机传到音乐合成器。虽然，被传送的位模式必须有一定的次序以代表特定的音乐数据，这种次序可以以很简单编码来实现，即在初始字节(或称状态字节)设一标志位以通知音乐合成器输入数据流的意义。只有当读者希望深入研讨上述实时控制应用时，这种低级编码才会有所帮助。否则不必作太多考虑。读者只需学会几种处理不同类型的 MIDI 音符或非音符信息的浮点型或模板。其他细节则由负责转译、整理音项向量表数据文件的转换程序处理。

如前所述，MIDI 编码全部为二进制形式，即只有 0、1 两种状态编码。数据传送方式为串行输入输出——即在传输线中每次只传输一个字节(一列 8 位数字)至音乐合成器。因为传输速度很快所以运行顺畅，每秒钟可传送的字节数为 31250(单位是波特率)。

MIDI 数据分为两大类：状态字节和数据字节。状态字节告诉计算机或音乐合成器传输数据的类型，表明该信息代表按键；还是变调盘；或是音调带等等。数据字节紧跟在状态字节之后，它们给出状态字节所代表的某种特定项目的具体值(如图 1-8 所示)。

0 1 1 0 1 1 0

| <— indicates status byte

0 1 1 0 0 0 1 1

| <— indicates data byte

图 1-8 状态字节与数据字节范例

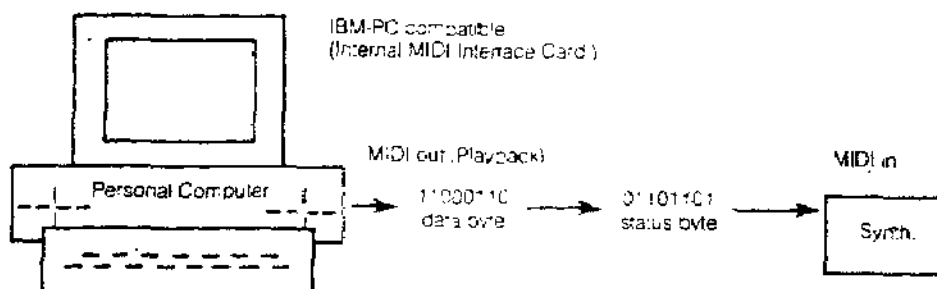


图 1-9 MIDI 数据传输简化图

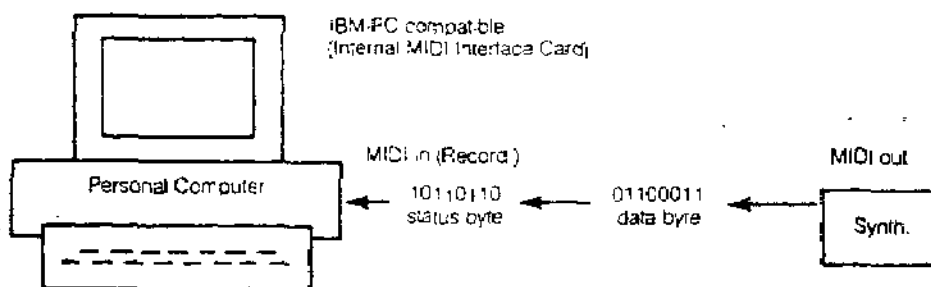


图 1-10 从音乐合成器录制 MIDI 数据至计算机

状态字节的首位为 1，数据字节的首位为 0。

图 1-9 是一幅 MIDI 传输简化图表，可以帮助用户理解此过程。

此过程的反过程即将音乐合成器键盘的 MIDI 数据录制回计算机。

**MIDI 通道** MIDI 通道看起来类似一个多通道录音机。目前的 MIDI 工具最多可提供 16 条并行声道，每一声道都可独立发声；那么只用一条串行数据线传送数据怎样实现 16 个声道同时工作呢？

事实上，每一状态字节中包含了该随字节输入的数据信息应该传送至哪一声道。这样，16 个 MIDI 声道的各种信息就可以综合交汇只用一条传输线传送。虽然在如何用软件实现在恰当的时间以规定次序传送每一数据方面还存在一些问题，详情我们将在第七章 Midifile 功能中跟算法作曲一起讨论。现在，请大家先接受这一点，即一台配有 MIDI 接口的计算机加上 2 个以菊花链或连接的 8 声道复音音乐合成器就相当于一个由 16 名乐师组成的管弦乐队，并可以用复杂的对位法谱曲。

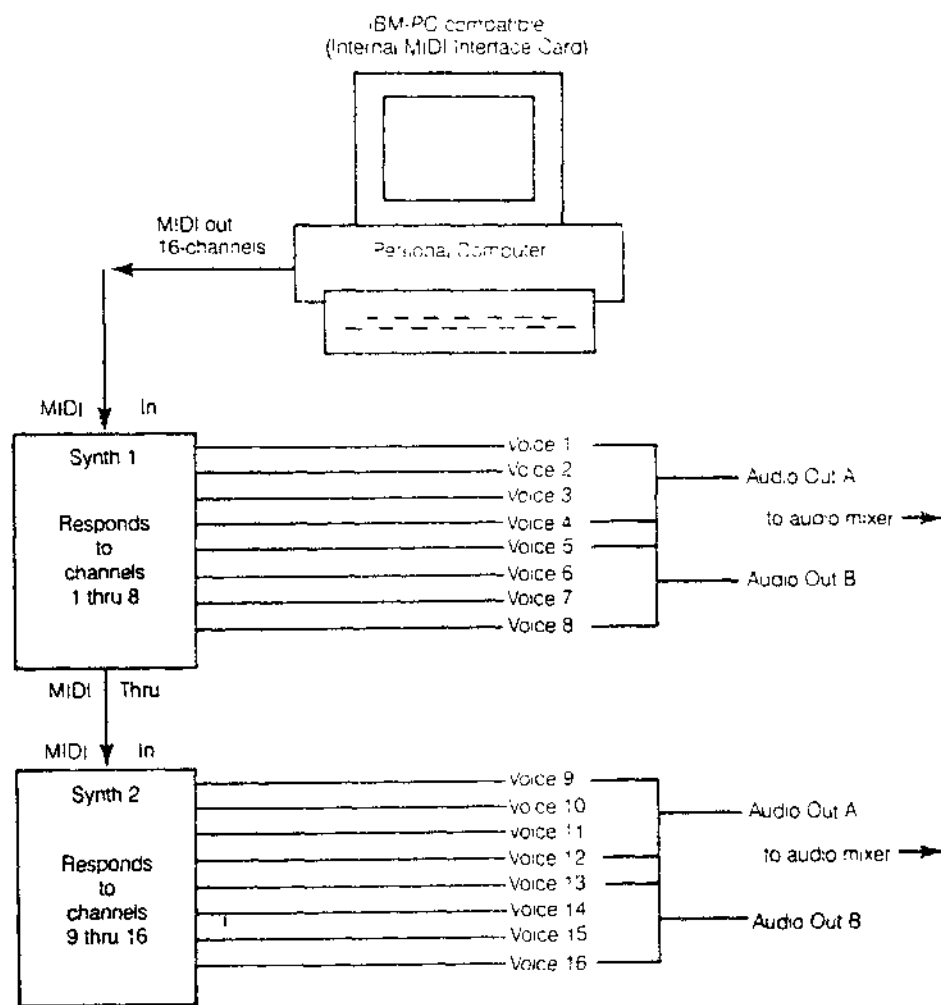


图 1-11 MIDI 时序器的时间测量

## MIDI 中的时间测量

商用时序软件包用根据计算机运行速度而定义的时钟来测量时间。虽然不同的软件测量时间的具体方法各不相同，但它们之间有一个共同点：即测量单位，称为 1 个时钟信号用一段时间相当于多个时钟信号来代表不同长短的时间。

每一段参考时间的时钟信号值(比如与一个  $1/4$  音相当的 T 值)决定了某一音乐系统的节拍结构。同一节拍，在一个将  $1/4$  音分为 192T(时钟信号)的时序软件中，其节拍值就较低，而在一个将  $1/4$  音分为 96T 的时序软件中其节拍值就较高，如图 1-12 所示。

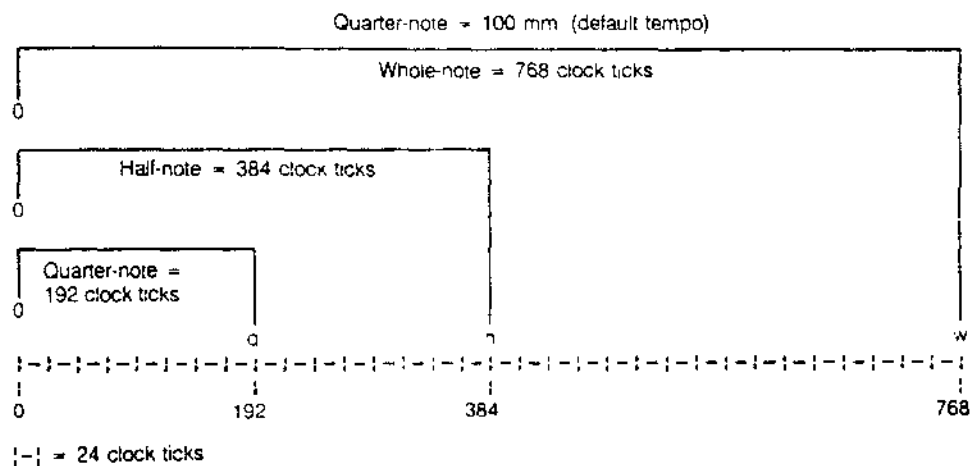


图 1-12 MIDI 时序器的时间测量

因为用整数在代表节拍值比较直观，若选参考值 1 来代表音乐中的全音符，以此类推  $1/1$  代表全音； $1/2$  代表半音； $1/4$  代表四分之一音，等等。在复音旋律中，我们将每一音的节拍值看作是一段以时钟信号(T)量度的时间，即从一个强音的奏响时刻到下一个强音的奏响时刻之间的间隔。(其它音可被视为无振幅，但实际上它们占用时间)这一时间间隔包含的时钟信号数称作“节拍衰减值”。如果用户将一个全音符分成无限多个时钟信号，就可以实现极为复杂的音乐节拍形式。然而，人的感官感知时间间隔的能力是有限的，这一点导致了古典音乐中的节拍形式很有限。同样，由于这一点，无限分音以增加节拍形式就显得意义不大。下图 1-13 就图示了 MIDI 参数与音乐元素之间的关系。

## ASCII 文件格式

图 1-14 是一个计算机用来定义音乐项目以及将其应用于恰当的乐器之上的变量表。

图 1-15 显示了每一音项的外壳结构(ASCII 文件记录)。

变量 DURATION(时钟信号数)加上变量 Noteon(音起始奏响时刻)可以计算出下一个音的奏响时刻。比如，假设一拍为 4 个  $1/4$  音，那么变量 Noteon 就必须每次增加 192 以代表下一个音的奏响时刻，例如：

音项#1     NOTEON = 0  
 音项#2     NOTEON = 192  
 音项#3     NOTEON = 384  
 音项#4     NOTEON = 576

音项 ARTDUR 的值通常比变量 DURATION 值小，因为休止音在变量 DURATION 中计入时间，而在变量 ARTDUR 中不计入。

PITCH			RHYTHM	VELOCITY		TIMBRE	
G	10	127	1 = Whole	768	127	gong	127
F#	10	126	2 = Half	384		bells	126
F	10	125	4 = Quarter	192		star	125
E	10	124	8 = Eighth	96		harpsi	124
D#	10	123	16 = Sixteenth	48		harmon	123
D	10	122	32 = 32nd	24		string	122
C#	10	121	64 = 64th	12		drum	121
C	10	120				sax4	120
						sax3	119
						sax2	118
						sax1	108
C	9	108					
				ff	100	tymp	96
C	8	96				cymbal	84
C	7	84		f	80		
						drum2	72
C	6	72		mf			
					60	drum1	60
C	5	60		mp			
					40	organ	48
C	4	48		p			
						glock	36
C	3	36		pp			
					20	snare	24
C	2	24		ppp			
						piano	12
B	0	11				tuba	11
A#	0	10				trhn	10
A	0	9				trmb	9
G#	0	8				trump	8
F#	0	6				clar	6
G	0	7				bassn	7
F	0	5				oboe	5
E	0	4				flute	4
D#	0	3				bass	3
D	0	2				cello	2
C#	0	1				via	1
C	0	0			0	vin	0

注意：变量 Timbre(音质调整)值为任意指定值。

图 1-13 MIDI 音符各参数值表

(适用于系统设计协会 SDA(System Design Associates)的 TOPM.EXE 文件转换软件)

Protocol: TOPM infile.dat outfile.v01

#### NOTE MESSAGES

Variables: (expressed in decimal form)

NOTEON: MIDI note event start time, measured in ticks  
PITCH: MIDI note numbers, 0 - 127  
VELOCITY: speed of key depression, 0 - 127  
DURATION: time-space (in ticks) allotted for a note  
ARTDUR: articulated duration (regulates the ratio of sound/silence for a given note (time-space)  
CHANNEL: MIDI channel number 0 - 15  
PORT: communication port number (remains constant, Promidi uses port 0)

图 1-14 计算机用于定义音项的各变量表

(当音乐合成器被设置为每次奏一音的单音演奏形式时，每一音符的音长都必须至少

缩短 85%，以提供乐器选通时间。也即，一个音结束到另一音奏响之间有一段衰减时间，这样可防止由于乐器选通所需时间引起的切音以及提高声音清晰度。）

Note Message Format.

Fields.	1	2	3	4	5	6
	NOTEON	PITCH	VELOCITY	AFT'DUR	CHANNEL	PORT
	integer	integer	integer	integer	integer	integer
	(ticks)	(0-127)	(0-127)	(time on)	(0-15)	(0)

图 1-15 各变量的 ASCII 文件记录

下面是 TOPM.EXE ASCII 格式的例子：

```
100 60 127 163 0 0 gated 1/4 note,max velocity, ch 1, middle C
192 61 90 163 00 gated 1/4 note,med velocity, ch 1, middle C#
384 62 50 163 10 gated 1/4 note,low velocity, ch 1, middle D
```

**非音符信号** 各种不同的音乐合成器的控制器及程序调整信息可以被写入 ASCII 文件中，同样也是应用 6 字段外壳，只是信息量减少。

图 1-16 是一个非音符信号的类型表。控制器调整可通过在音奏响前后发出一个信息来实现。图 1-17 所示的 MIDI 控制器为多种音乐合成器所支持。

Message	No. of Bytes	Code Number (in decimal)
Polyphonic Aftertouch	3	160-175 (channels 0-15)
Controller Change	3	176-191 (channels 0-15)
Program Change	2	192-207 (channels 0-15)
Channel Aftertouch	2	208-223 (channels 0-15)
Pitch Bender	3	224-239 (channels 0-15)

图 1-16 非音符记录类型表

Controller	Number (decimal)	Value range (decimal)
Modulation Wheel	1	0-127
Breath Controller	2	0-127
Foot Controller	4	0-127
Portamento Time	5	0-127
Data Entry Slider	6	0-127
Main Volume	7	0-127
Sustain	64	OFF-0, ON-127
Portamento	65	OFF-0, ON-127
Sostenuto	66	OFF-0, ON-127
Soft	67	OFF-0, ON-127
Data Increment	36	127
Data Decrement	37	127
Local	122	OFF-0, ON-127
All Note Off	123	OFF-0
Omni Off	124	OFF-0
Omni On	125	OFF-0
Mono On	126	0-15 (# channels)
Poly On	127	0

图 1-17 上列为合各电子乐器生产厂家生产的控制器主要类型

图 1-18 是一显示 MIDI 主音量控制器的控制器调整调整信息。

图 1-19 是一原始格式表，填表方式如图所示。

图 1-20 是主音量控制器的输出文件。

Variables (decimal form)

STARTIME start time (in ticks)  
 CONTROLLER refer. number of controller (Main vol = 7)  
 CSTATUSCHANNEL status byte, type of message (176 + ch. no.)  
 VALUE data value, range 0 - 127  
 PORT communication port number (remains constant, Promidi uses port 0)

图 1-18 MIDI 主音量控制器的调整信息

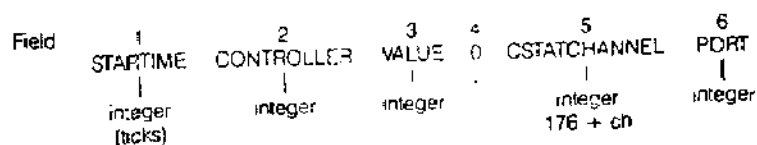


图 1-19 ASCII 文件范例

ASCII File Example

```

090 7 0 0 179 0 set main volume on channel 4 to 0
100 60 127 163 3 0 gated 1/4 note, max velocity, ch 4, middle C
104 7 10 0 179 0 increment main volume for crescendo
106 7 40 0 179 0 increment main volume for crescendo
108 7 60 0 179 0 increment main volume for crescendo
110 7 80 0 179 0 increment main volume for crescendo
115 7 127 0 179 0 return main vol to max for next note
192 61 90 163 3 0 gated 1/4 note, med velocity, ch 4, middle C#
384 62 20 163 3 0 gated 1/4 note, low velocity, ch 4, middle D
  
```

图 1-20 主音量控制器输出文件

音质调整可通过在一音结束另一音奏响之前发出一音质调整指令来实现，如图 1-21 所示。

图 1-22 所示是一个音质调整或控制器调整程序的输出文件，为了防止在输出时出现“信号突变”，可以在输出非音符信号和音符信号之间设置至少 10 个时钟信号来避免这种现象的产生。

Variables: STARTIME start time (in ticks)  
 VOICENUM Timbre (patch) number 0 - n  
 PSTATUSCHANNEL status byte (192 + channel)

Program Change Message Format

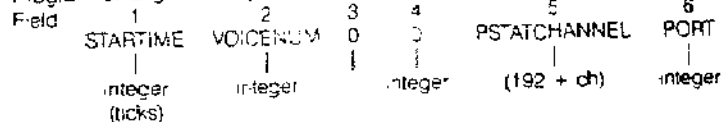


图 1-21 音质调整

#### ASCII File Example

```
090 12 0 0 192 0 select Program (timbre) 12 for channel 1
100 60 127 163 0 0 gated 1/4 note, max velocity, ch 1, middle C
180 12 0 0 192 0 select Program (timbre) 12 for channel 1
192 61 90 163 0 0 gated 1/4 note, med velocity, ch 1, middle C#
372 26 0 0 192 0 select Program (timbre) 26 for channel 1
384 62 50 163 1 0 gated 1/4 note, low velocity, ch 2, middle D
```

图 1-22 程序修改信息输出文件

## 从 MIDI 数据到标准乐谱

**转换程序** 在完成将算法谱曲谱出的音乐数据转换成铅印的乐谱这项工作中，最后环节即是把 MIDI(时序)数据转换成一种可被印谱软件识别的形式。现有几种软件包实现此功能，但应用范围和实效各异，希望读者自己广泛应用，从中选出一最软件。

我们应用兼容 PSS 系统的 IBM PC 机将音项数据转换成时序文件格式，这种软件系统可以产生两种不同的音乐印刷制品：即个人作曲谱与曲谱复制版本。

如果读者不仅有 IBM PC 机，而且有 Macintosh 机，那么就有许多乐谱印刷软件可供选用，只须将 IBM 机的 MIDI OUT 端口连至 Macintosh 机的 MIDI IN 端口即可实现，音乐可以通过在 IBM 源时序器上演奏而录制到 Macintosh 机的接收器(例如，演奏器)上实现传输，录制好的文件就可以存入 Finale 或 Professional Composer 等音乐印谱软件中，并编辑、打印出校样。

图 1-23 图示了该转换过程。

## 乐谱印刷软件

目前市面上有各种乐谱软件，但最简单实用的是 Personal Composer / System 2(用于 IBM PC 机)，该软件在全球范围销售，MIDI 顺序放音文件和乐谱印刷软件必须跟一个文件转换软件接口才能将用计算方法谱出的曲子印制成谱，而系统设计协会推出的 PSS 系统可为 Personal Composer / System 2 和 Copyist Music Notation 软件提供这种转换程序。

## 计算机—MIDI 接口软件简介

撰写这本书之际，还只有 2 个厂家生产软件应用于计算机音乐作曲，并能直接将计算机谱出的音乐数据文件转换成乐谱形式。当然，随着这一领域的发展，越来越多的人会加入到开发音乐软件的行列。

PROMIDI STUDIO SYSTEM 系统含有 MIDI 接口、排序软件；Roland MPU401 仿真扩展器；ASCII-Promidi 系统转换软件；Promidi-PCN(Personal Composer Notation)转换软件；Promidi-CMPS(Copyist Music Printing SoftWare)转换软件。



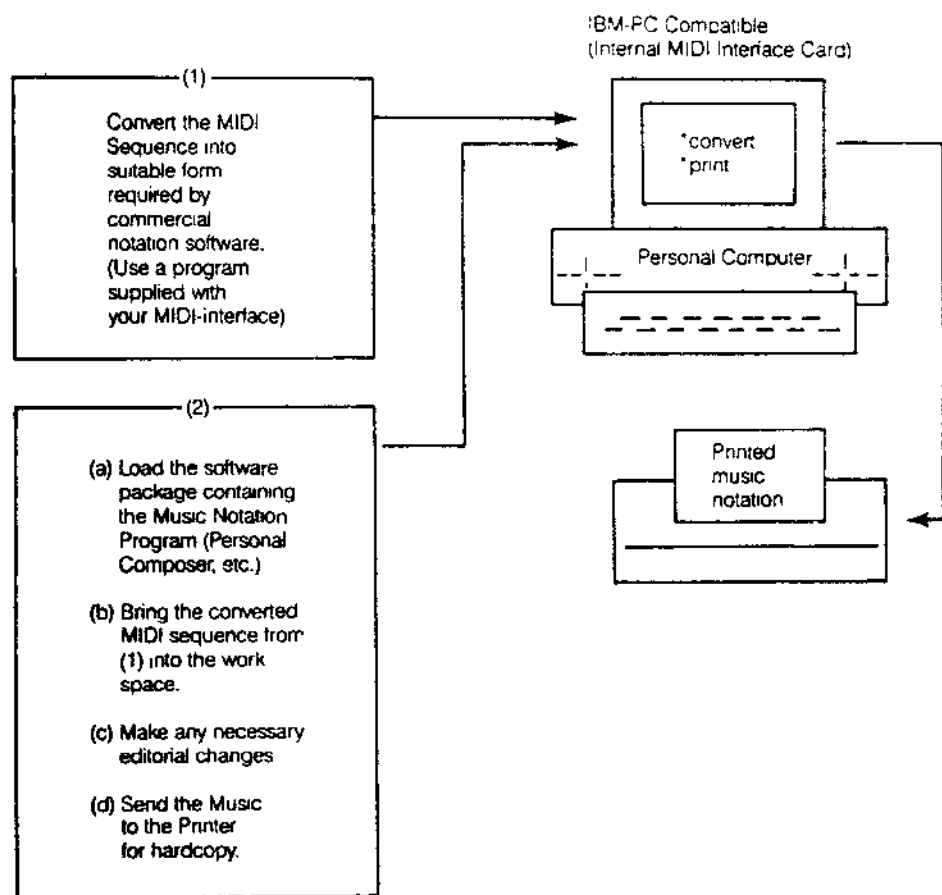


图 1-23 将算法作曲谱出的音乐数据转换为标准乐谱示意图

PERSONAL COMPOSER 软件，设计者是美国的 Jim Miller，该软件可在大多数生产与 IBM PC 机兼容的音乐合成器厂商处购得，建议与 PSS 系统及其转换软件 PCV.EXE 配套使用。

CAKEWALK 录音 / 编辑软件(可运行于与 IBM 兼容的配有 Roland MPU 401 MIDI 接口的音乐合成器)，该软件支持 ASCII-CAKEWALK 以及 CAKEWALK-ASCII 转换功能，目前还没有能实现 CAKEWALK 文件—标准乐谱转换功能的软件。

## 第二章 工具库

### 函数组: Tunings.c

1. Tunings() MIC 2.1
2. Fpower() MIC 2.2

### 功能

编译一个用 Hz 表示的同等节奏, 信频重复的微电子音阶频率表。

这个音阶编译器的一个直接应用是为混响器输出的音调生成微电子音阶。用户还可以把它用到另外一些程序, 比如, 把不同的调谐音阶变换到音量、节奏、音色和清晰度参数。

当把一组数据变换到另外一些参数时, 这个子程序把第一个数据表(数组)理解为一组指向第二个数据表的指针。

### 注释

参数“Octdiv”控制在一个八度中包含的等音程的数量, 参数“freq1”设置最初的(音调的)音距, 参数“numtones”设置了上升的音调数量。举例来说, 如果 numtones = 60 并且 octdiv = 12, 那么这个子程序就编译一个五个八度的音阶。

### 编程提示

1. 用音阶编译器的输出试验用户微机上的混芯片所能发出的的不同音调。
2. 把频率数据从程序中读入, 然后通过求模运算符(%)把这些值转换成一小组间歇时期。

### 程序清单

```
/* TUNINGS.C (equal-tempered, octave-repeating, microtonal  
scale compiler) */  
#include <stdio.h>  
#include <math.h>  
main()  
{
```

```

int tone[84], numtones, octdiv, j;
double freq1;
void Tunings();

numtones = 84;
octdiv = 19;
freq1 = 61.735; /*beginning frequency for scale computation*/
Tunings(tone, freq1, octdiv, numtones);
for (j = 0; j < numtones; j++)
{
    if(j % 10 == 0)
        printf("\n");
    printf("%d ", tone[j]);
}
} /* end of main */
/*===== MIC 2.1 =====*/
/* Tunings() function */
void Tunings(tone, freq1, octdiv, numtones)
int tone[], octdiv, numtones;
double freq1;
{
    double Fpower();
    double ccoeff, t9;
    double base=2.0;
    int j;

    t9 = 1.0/octdiv;
    ccoeff = Fpower(base, t9);
    tone[0]=freq1+.5;
    for (j = 1; j < numtones; j++)
        tone[j] = tone[j-1] * ccoeff + .5;
} /* end Tunings() function */
/*===== MIC 2.2 =====*/
/* Fpower() function */
double Fpower(value, tothe)
double value, tothe;
{
    int sign;
    double result;

    sign = (tothe < 0.0) ? -1 : 1;
    tothe = fabs(tothe);
    result = exp( log(value) * tothe);
    if(sign < 0)
        result = 1.0 / result;
    return(result);
} /* end of Fpower() function */
/*=====*/
/* END OF TUNINGS.C */

```

### 程序运行实例

TUNINGS.EXE

```

62 64 66 68 71 74 77 80 83 86
89 92 95 99 103 107 111 115 119 123
128 133 138 143 148 153 159 165 171 177
184 191 198 205 213 221 229 238 247 256
266 276 285 297 308 319 331 343 356 369
383 397 412 427 443 459 476 494 512 531

```

```

551 571 592 614 637 661 686 711 737 764
792 821 852 884 917 951 986 1023 1061 1100
1141 1183 1227 1273

```

**函数: Pitchtab() MIC2.3**

## 功能

提供一个用在低八度到高八度的范围内,把整数值转换成半音的注册音程元素表。

## 注释

这个函数在初始化一个音乐元素表时,用一种类似于 BASIC 语言中的 READ...DATA 方法。逐条列举所有在 DATA 语句中注明的音阶--C1,C#1,D1,D#1 等等——然后在程序执行中把它们读到一个元素数组表中去。其间的差别在于只用一个语句就把数值表赋给一个静态数组了,从而避免了为顺序地读数据到元素组中去而反复循环。

虽然这儿的代码形式是函数,但也可以在主函数中用 "in-line" 包含进去。

## 编程提示

在交互式的程序中,一个音阶表帮助用户把数据变换成熟悉的条目。

1. 写个程序生成一些随机的整数,同时可以让用户用音阶或数字的形式观察这些数据。
2. 修改这个函数,使它可以初始化另外形式的参数,例如节奏或音量。

## 程序清单

```

/* PITCHTAB.C (initializes a pitch data table
               corresponding to integer values 1-n) */
#include <stdio.h>
main()
{
    void Pitchtab();

    Pitchtab();

} /* end of main */
/*===== MIC 2.3 =====*/
/* Pitchtab() function */
void Pitchtab()
{
    int j,total = 85;
    static char *pitch[] = {
        "C0 ", "C#0 ", "D0 ", "D#0 ", "E0 ", "F0 ", "F#0 ", "G0 ", "G#0 ",
        "A0 ", "A#0 ", "B0 ", "C1 ", "C#1 ", "D1 ", "D#1 ", "E1 ", "F1 ",
        "F#1 ", "G1 ", "G#1 ", "A1 ", "A#1 ", "B1 ", "C2 ", "C#2 ", "D2 ",
        "D#2 ", "E2 ", "F2 ", "F#2 ", "G2 ", "G#2 ", "A2 ", "A#2 ", "B2 ",
        "C3 ", "C#3 ", "D3 ", "D#3 ", "E3 ", "F3 ", "F#3 ", "G3 ", "G#3 ",
        "A3 ", "A#3 ", "B3 ", "C4 ", "C#4 ", "D4 ", "D#4 ", "E4 ", "F4 ",

```

```

    "F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
    "D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
    "C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
    "A6 ", "A#6", "B6 ", "C7 " );

for (j = 0; j < total; j++)
    if (j % 15 == 0)
        printf("\n%s ", pitch[j]);
    else
        printf("%s ", pitch[j]);

} /* end of Pitchtab() function */
/*=====*/
/* END OF PITCHTAB.C */

```

## 程序运行实例

PITCHTAB.EXE

```

C0  C#0  D0  D#0  E0  F0  F#0  G0  G#0  A0  A#0  B0  C1  C#1  D1
D#1  E1  F1  F#1  G1  G#1  A1  A#1  B1  C2  C#2  D2  G#2  E2  F2
F#2  G2  G#2  A2  A#2  B2  C3  C#3  D3  D#3  E3  F3  F#3  G3  G#3
A3  A#3  B3  C4  C#4  D4  D#4  E4  F4  F#4  G4  G#4  A4  A#4  B4
C5  C#5  D5  D#5  E5  F5  F#5  G5  G#5  A5  A#5  B5  C6  C#6  D6
D#6  E6  F6  F#6  G6  G#6  A6  A#6  B6  C7

```

## 函数组: Vectors.c

1. Parstore() MIC2.4
2. Parxtret() MIC2.5

## 功能

配置一个最终矢量数组。在这一过程中调用了混合数据存贮[Parstore()]和复原[Parxtret()]函数。每个在数组中的地址都保留了单个音调的音阶、音量和节奏。

虽然这些程序处理特殊的音乐元素，但是，还会出现因为系统内存不够而需要压缩原始数据的情况，这些函数也可以用上。

## 注释

混淆数据存贮和复原的工作原始是把每个数的不同段分配给这些音乐参数：

```

000    000    000
---    ---    ---
音量   节奏   音阶

```

因为这个原因，在这儿需要双精度的数值。唯一的限制就是参数值的范围一定要小于999。任何用户想要控制的参数都要替换成程序变量“p”、“r”和“v”。

## 编程提示

1. 修改这个程序，使它可以存贮另外一些复合信息，例如结构密度、音色和清晰度。

2. 把程序从用于单个声音水平，转向用于宏格式水平，也就是贮存信息得到一个片段中的旋律个数，一个片段组中的片段个数和一个段中的片段组的个数。

## 程序清单

```

/* VECTORS.C (creates event vectors by storing pitch, rhythm
and volume data for a single musical event in
one double precision number.)*/
#include <stdio.h>
main()
{
    int total = 10;

    void Parstore(), Parxtret();
    double c[10];

    printf("%s %d %s", "attributes being stored for",
           total, "notes are:\n");
    Parstore(c, total);
    printf("\nexttracted note attributes are:\n");
    Parxtret(c, total);

} /* end of main */
/***** MIC 2.4 *****/
/* Parstore() function */
void Parstore(array, total)
double array[];
int total;
{
    int j;
    int v; /* note volume parameter */
    double p; /* note pitch parameter */
    double r; /* note rhythm parameter */

    for (j = 0; j < total; j++)
    {
        p = rand() % 88 + 1; /* load a random pitch number */
        r = rand() % 16 + 1; /* load a random rhythm value */
        v = rand() % 100 + 1; /* load a random volume value */
        printf("pitch = %.0f\n", p);
        printf("rhythm = %.0f\n", r);
        printf("volume = %d\n", v);
        p = p * 1000000.;
        r = r * 1000.;
        array[j] = p + r + v;
        printf("composite number = %.0f\n", array[j]);
    }

} /* end of Parstore() function */
/***** MIC 2.5 *****/
/* Parxtret() function */
void Parxtret(c, total)
double c[];
int total;
{
    double d;
    int p, r, v, j;
    for (j = 0; j < total; j++)
    {
        d = c[j];
        d = d / 1000000.;
        p = d;
        r = (d - p) * 1000;
    }
}

```

```

        d = d * 1000.;
        v = (d - (int)d) * 1000.+.5;
        printf("p = %d\r r = %d\t v = %d\n",p,r,v);
    }
} /* end of Parxtrot() function */
/*===== */
/* END OF VECTORS.C */

```

## 程序运行实例

### VECTORS.EXE

```

attributes being stored for 10 notes are:
pitch = 83      rhythm = 18      volume = 83
composite number = 83018083
pitch = 35      rhythm = 24      volume = 18
composite number = 35024018
pitch = 84      rhythm = 31      volume = 49
composite number = 84031049
pitch = 63      rhythm = 28      volume = 59
composite number = 63028059
pitch = 52      rhythm = 31      volume = 93
composite number = 52031093
pitch = 41      rhythm = 20      volume = 22
composite number = 41020022
pitch = 24      rhythm = 23      volume = 20
composite number = 24023020
pitch = 26      rhythm = 22      volume = 86
composite number = 26022086
pitch = 63      rhythm = 31      volume = 53
composite number = 63031083
pitch = 84      rhythm = 27      volume = 17
composite number = 84027017

```

```

extracted note attributes are
p = 83  r = 18  v = 83
p = 35  r = 24  v = 18
p = 84  r = 31  v = 49
p = 63  r = 28  v = 59
p = 52  r = 31  v = 93
p = 41  r = 20  v = 22
p = 24  r = 23  v = 20
p = 26  r = 22  v = 86
p = 63  r = 31  v = 53
p = 84  r = 27  v = 17

```

## 函数组: Curves.c

1. Lincurve() MIC2.6
2. Expcurve() MIC2.7
3. Logcurve() MIC2.9

## 功能

生成一个上升的整数组，这个数组与线性函数、指数函数、对数函数之一拟合。

虽然这组函数特别适合生成光渐地加快节奏、音量渐强、缓慢的音色变化等效果，但

是用户也可以把它直接用于任何需要离散的连续变化的整数的地方，还可以间接地用于指向一组数据。

## 注释

控制参数是“start”、“end”和“num”，这些值都必须是正整数。

## 编程提示

1. 写一个直接用于音阶、节奏和音量参数的程序。
2. 生成不可范围的曲线，用于当作指针指向包含没有排序的随机顺序的音乐参数值。
3. 修改曲线的算法，把返回正整数改为返回负整数。

## 程序清单

```
/* CURVES.C (returns positive integers along three
   discrete curves: linear, exponential, logarithmic) */
#include <stdio.h>
#include <math.h>
main()
{
    int x[300], start = 1, end = 200, num = 100, j1, j2;
    void Lincurve(), Expcurve(), Logcurve();

    printf("curve start will be %d, end will be %d ", start, end);
    printf("\nand number of elements will be %d.\n", num);
    for (j1 = 1; j1 <= 3; j1++)
    {
        if (j1 == 1)
            Lincurve(x, start, end, num);
        else
            if (j1 == 2)
                Expcurve(x, start, end, num);

            else Logcurve(x, start, end, num);
        for (j2 = 1; j2 <= num; j2++)
            if (j2 % 15 == 0)
                printf("\n%d ", x[j2]);
            else
                printf("%d ", x[j2]);
        printf("\n");
    }
    /* end of main */
    /*===== MIC 2.6 =====*/
    /* Lincurve() function */
    void Lincurve(x, start, end, num)
    int x[], start, end, num;
    {
        int j, l;
        float a;

        printf("\nlinear curve:\n");
        for (j = 1; j <= num; j+=10)
        {
```



3B 1-10

```

        for (l = j; l <= j + 9; l++)
        {
            a = (float) (l-1) / (num-1);
            x[l] = start + a * (end-start) +.5;
            if (l == num)
                break;
        }
    }

} /* end of Lincurve() function */
/***** MIC 2.7 *****/
/* Expcurve() function */
void Expcurve(x, start, end, num)
int x[], start, end, num;
{
    int j, l;
    float a;

    printf("\nexponential curve:\n");
    for (j = 1; j <= num; j += 10)
    {
        for (l = j; l <= j + 9; l++)
        {
            a = (float) (l-1) * (l-1) / ((num-1) * (num-1));
            x[l] = start + a * (end-start) +.5;
            if (l == num)
                break;
        }
    }

} /* end of Expcurve() function */
/***** MIC 2.8 *****/
/* Logcurve() function */
void Logcurve(x, start, end, num)
int x[], start, end, num;
{
    int j, l;
    float a, b, log();

    printf("\nlogarithmic curve:\n");
    for (j = 1; j <= num; j += 10)
    {
        for (l = j; l <= j + 9; l++)
        {
            a = 1;
            b = num;
            x[l] = start + log(a)/log(b) * (end-start) +.5;
            if (l == num)
                break;
        }
    }

} /* end of Logcurve() function */
/***** END OF CURVES.C *****/

```

### 程序运行实例

CURVES.EXE

curve start will be 1,end will be 200  
and number of elements will be 100.

linear curve:

1 3 5 7 9 11 13 15 17 19 21 23 25 27  
29 31 33 35 37 39 41 43 45 47 49 51 53 55 57  
59 61 63 65 67 69 71 73 75 77 79 81 83 85 87  
89 91 93 95 97 99 102 104 106 108 110 112 114 116 118  
120 122 124 126 128 130 132 134 136 138 140 142 144 146 148  
150 152 154 156 158 160 162 164 166 168 170 172 174 176 178  
180 182 184 186 188 190 192 194 196 198 200

exponential curve:

1 1 1 1 1 2 2 2 2 3 3 3 4 4  
5 6 6 7 8 8 9 10 11 12 13 14 15 16 17  
18 19 21 22 23 24 26 27 29 30 32 33 35 37 39  
40 42 44 46 48 50 52 54 56 58 60 62 65 67 69  
72 74 77 79 82 84 87 89 92 95 98 100 103 106 109  
112 115 118 121 125 128 131 134 138 141 144 148 151 155 158  
162 165 169 173 177 180 184 188 192 196 200

logarithmic curve:

1 31 48 61 71 78 85 91 96 101 105 108 112 115  
118 121 123 126 128 130 133 135 136 138 140 142 143 145 147  
148 149 151 152 153 155 156 157 158 159 160 161 163 164 165  
165 166 167 168 169 170 171 172 173 173 174 175 176 176 177  
178 179 179 180 181 181 182 183 183 184 185 185 186 186 187  
188 188 189 189 190 190 191 191 192 192 193 193 194 194 195  
195 196 196 197 197 198 198 199 199 200 200

## 函数: Fractab() MIC2.9

### 功能

统计用分数表示的( $1/2, 1/4, 1/8, 1/16, 1/32, 1/64$ , etc)一系列间歇时间。

有时, 一组随机生成的旋律节奏值必须同一个预设定的时间结构一致。这个子程序就可以提供一个生成值的总和, 然后反馈到程序中供单个时间的调整。

例如, 现有要求如下:

- 旋律必须包含八个音节。
- 音节值必须确切地与两个音节相等(总共三十二个十六分音符)。
- 子程序必须返回时间序列 $1/16, 1/2, 1/16, 1/4, 1/16, 1/8, 1/4, 1/2$ (总共二十九个十六分音符)。

然后总计和输出需求之间的误差用最小的值的整数倍来弥补。在这个例子中用三个十六分音符弥补三个随机选择的音节。

### 注释

如果用户想要生成如  $3/16, 2/5, 7/8$  等等之类的多分子节奏型, 就用 FRACSUM.C(MIC2.12 和 MIC2.13)中的 Fractab()函数。

## 编程提示

1. 写一个生成在一个旋律线上随机排列的音阶和节奏的程序。包含Fractab()函数和  
其它的语句使旋律节奏总和与一个用户输入时间结构相一致。

## 程序清单

```
/* FRACTAB.C (tabulates rhythmic duration values in cases where
each value can be expressed as a fraction which
has 1 as the numerator (e.g. 1/8). The sum of all
durations is returned in decimal.)*/
#include <stdio.h>
main()
{
    int x[50], j;
    int total = 50;
    float sum = 0.0;
    float Fractab();

    for (j = 0; j < total; j++)
    {
        x[j] = j + 1; /* load fraction denominators */
        if(j % 5 == 0)
            printf("\n1/%d\t", x[j]);
        else
            printf("1/%d\t", x[j]);
    }
    sum = Fractab(x, total); /* call tabulation */
    printf("\nThe sum of durations is %f whole notes", sum);
} /* end of main */
/*===== MIC 2.9 =====*/
/* Fractab() function */
float Fractab(x, total)
int x[], total;
{
    int j;
    float sum = 0.0;

    for (j = 0; j < total; j++)
        sum += 1.0 / x[j];
    return(sum);
} /* end of Fractab() function */
/*=====*/
/* END OF FRACTAB.C */
```

## 程序运行实例

FRACTAB.EXE

1/1	1/2	1/3	1/4	1/5
1/6	1/7	1/8	1/9	1/10
1/11	1/12	1/13	1/14	1/15
1/16	1/17	1/18	1/19	1/20
1/21	1/22	1/23	1/24	1/25
1/26	1/27	1/28	1/29	1/30
1/31	1/32	1/33	1/34	1/35
1/36	1/37	1/38	1/39	1/40
1/41	1/42	1/43	1/44	1/45
1/46	1/47	1/48	1/49	1/50

The sum of durations is 4.499206 whole notes

## 函数: Durred() MIC2.10

### 功能

把一组节拍约分到最低限度(与暂停符一致),然后把约分后的节拍作为一个字符串返回给程序。

### 注释

计算机生成的节奏必须用方便的符号记录下来。这种情况越来越普通,这个过程通过简化节拍到最低层来完成。例如,假设背景音乐为  $6/8$  次,而在一个音乐家看来  $12/16$  比  $6/8$  更确切,因为  $6/8$  更加适当地表达了背景与单独的音节的关系。

如果人们没有必要解释音节,比如,当数字是被送到一个数字式混响器中去时,用函数 Eucreduc()-MIC2.11 会更加有效。

### 编程提示

1. 转换主程序,使它生成随机顺序的分数分母,而不是分子,观察用这种约分方法作用以后生成的程序结果。
2. 写一个允许用户要求得到任意长的随机顺序的旋律,生成适合的节拍数,然后把它调整到与一个输入的参考量一致。

### 程序清单

```
/* DURRED.C (Reduces an array of rhythm duration
fractions to lowest terms [consistent with
rhythm pulse subdivision and places them
in a string array */
#include <stdio.h>
main()
{
    int j,x[20],y[20],total=20;
    void Durred();
    int seed = 2221;

    srand(seed);
    printf("random fraction numerators & denominators:\n");
    for (j = 0;j < total;j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 50 + 1;
        y[j] = 32;
        printf("%d/%d ",x[j],y[j]);
    }
    Durred(x,y,total);
    printf("\nthe reduced sequence:\n");
    for (j = 0;j < total;j++)
    {
```

```

        if (j % 10 == 0)
            printf("\n");
        printf("%d/%d ", x[j], y[j]);
    }
} /* end of main */
/*===== MIC 2.10 =====*/
/* Durred() function */
void Durred(x,y,total)
int x[],y[],total;
{
    int j;
    for (j = 0; j < total; j++)
    {
        if (x[j] / 2.0 != x[j] / 2 || y[j] / 2.0 != y[j] / 2)
            continue;
        else
        {
            x[j]=x[j] / 2;
            y[j]=y[j] / 2;
        }
    }
} /* end of Durred() function */
/*=====*/
/* END OF DURRED.C */

```

### 程序运行实例

DURRED.EXE

random fraction numerators & denominators:

41/32 40/32 28/32 9/32 29/32 15/32 16/32 18/32 20/32 10/32  
 32/32 5/32 25/32 21/32 34/32 21/32 50/32 40/32 36/32 31/32  
 the reduced sequence:

41/32 20/16 14/16 9/32 29/32 15/32 8/16 9/16 10/16 5/16  
 16/16 5/32 25/32 21/32 17/16 21/32 25/16 20/16 18/16 31/32

### 函数: Eucreduc() MIC2.1

#### 功能

把节拍约分到绝对最低限(当脉动小节的一致性没有提到时用)。把分数作为一个字符串返回程序。

#### 注释

当对整个整数脉动小节有限制地约分时, 用 Durred.MCI2.10 函数。(例如, 9/12 就不被再约分了。)。但是, Eucreduc() 函数是用欧几里得的最大分约数算法实现把一个分数(例如, 9/12)转换到更低的限度-3/4。当小节被直接送到一个混响器中去时, 最好使用这个子程序。

## 编程提示

参见 MIC2.10 Durred()函数应用.

## 程序清单

```
/* EUCREDUC.C (reduces rhythm durations to absolute
lowest terms using GCD - Euclid's algorithm;
i.e. it doesn't preserve metrical
pulse consistency. For example the fraction 9/12
reduces to 3/4 )*/
#include <stdio.h>
main()
{
    int x[50],y[50],total = 50,seed = 17114,j;
    void Eucreduce();

    srand(seed);
    printf("%s%s","Loading random fraction",
           " numerator and denominator arrays---\n");
    for (j = 0;j < total;j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % total + 1;
        y[j] = rand() % 16 + 1;
        printf("%d/%d ",x[j],y[j]);
    }
    printf("\nthe reduced fraction sequence ---\n");
    Eucreduce(x,y,total);
} /* end of main */
/*===== MIC 2.11 =====*/
/* Eucreduce() function */
void Eucreduce(x,y,total)
int x[],y[],total;
{
    int a,b,c,d,j,temp,count=0;

    for (j = 0;j < total;j++)
    {
        if (y[j] == 1)
        {
            printf("%d/%d ",x[j],y[j]);
            continue;
        }
        a = x[j];
        b = y[j];
        if (a > b)
        {
            temp = a;
            a = b;
            b = temp;
        }
        while(a > 0)
        {
            c = b / a;
            d = b - a * c;
            b = a;
            a = d;
        }
        printf("%d/%d ",x[j]/b,y[j]/b);
        count++;
    }
}
```

```

        if (count % 10 == 0)
            printf("\n");
    }
} /* end of Eucreduce() function */
/*=====*/
/* END OF EUCREDUC.C */

```

## 程序运行实例

EUCREDUC.EXE

Loading random fraction numerator and denominator arrays---

```

13/12 27/16 44/15 23/7 3/1 48/4 28/9 7/5 2/8 14/12
22/5 34/3 22/5 6/12 19/14 46/13 46/13 19/10 44/5 28/4
26/9 2/2 45/12 46/14 25/8 15/2 4/13 19/3 12/12 41/14
36/3 12/1 23/10 8/11 28/14 25/7 14/13 2/5 49/1 17/11
29/1 20/11 7/6 46/5 22/10 41/16 35/6 10/4 22/12 13/10
the reduced fraction sequence ---
13/12 27/16 44/15 23/7 3/1 12/1 28/9 7/5 1/4 7/6 22/5
34/3 22/5 1/2 19/14 46/13 46/13 19/10 44/5 7/1 26/9
1/1 15/4 23/7 25/8 15/2 4/13 19/3 1/1 41/14 12/1
12/1 23/10 8/11 2/1 25/7 14/13 2/5 49/1 17/11 29/1 20/11 7/6 46/5
11/5 41/16 35/6 15/2 11/6 13/10

```

## 函数组: Fractsum.c

1. LeastCommonMultiple() MIC2.12
2. GreatestCommonDivison() MIC2.13

## 功能

虽然在这个组里的函数各有单独的应用,但是在这儿用来计算一个多变量节拍数组的总和。

## 注释

另外一个函数, Fractab() MIC2.9, 也计算节拍的总数,但是,它用一个十进制数来表示整个小节值。然而,分数必须每个都是有分母1。

相反, FRACTSUM.C 对分数类型没有限制,它返回的总和是一个分数,而不是十进制数。(例如,  $2/1 + 11/16 = 2\ 11/16$ ) 函数 LeastCommonMultiple() 调用最大分约数函数来计算分数的总和。

但是,这种特殊的分数计和的方法有一种缺陷,就是当在一个小的字长的计算机上运行时,马上就会溢出。(需增加另外的代码来处理这种错误情况。)

## 编程提示

写一个在一个旋律线上生成随机顺序的音阶和节奏的程序，包含 Fractab() MIC2.9 函数和其它的语句使旋律节奏总和与一个用户输入时间结构一致。

#### 程序清单

```

/* FRACTSUM.C (sums a sequence of rhythm duration fractions;
               beware of potential overflow caused by overly
               complex fractions. */
#include <stdio.h>
main()
{
    unsigned long int x[50],y[50];
    int total = 50,seed = -231,j;
    unsigned long int GreatestCommonDivisor();
    void LeastCommonMultiple();

    srand(seed);
    printf("a sequence of fractions to be summed ----\n");
    for (j = 0;j < total;j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 10 + 1;

        do
            y[j] = rand() % 8 + 1;
        /* accept only simple denominators */
        while(y[j] % 3 != 0 && y[j] % 4 != 0);
        printf("%d/",x[j]);
        printf("%d ",y[j]);
    }
    LeastCommonMultiple(x,y,total);
} /* end of main */
/***** MIC 2.12 *****/
/* LeastCommonMultiple() function
   (determines the least common multiple of two integers
   by calling the greatest common divisor function to supply
   that value, then divides the product of the two integers
   by the greatest common divisor.) */

void LeastCommonMultiple(x,y,total)
unsigned long int x[],y[];
int total;
{
    unsigned long int whole = 0,num=0,den=0,LCM,GCD,j,m,n;
    unsigned long int GreatestCommonDivisor();

    for (j= 1;j < total;j++)
    {
        m = y[0]; /* y[0] stores current denominator sum */
        n = y[j]; /* next fraction denominator */
        GCD = GreatestCommonDivisor(m,n);
        LCM = m * n / GCD;
        m = LCM / y[0];
        n = LCM / y[j];
        /* x[0] stores current fraction denominator sum; x[j]
        holds next fraction denominator */
        num = (x[0] * m) + (x[j] * n);
        den = LCM;
        m = num;
        n = den;
        GCD = GreatestCommonDivisor(m,n);
        num = m / GCD; /*num stores reduced numerator total */
        den = n / GCD; /*den store reduced denominator total*/
    }
}

```



```

        whole += num / den;
        x[0] = num % den;
        y[0] = den;
    }
    printf("\nsum of fractions = ");
    printf("%d + ", whole);
    printf("%d/", num % den);
    printf("%d", den);

} /* end of LeastCommonMultiple() function */
/*===== MIC 2.13 =====*/
/* GreatestCommonDivisor() function */
unsigned long int GreatestCommonDivisor(m,n)
unsigned long int m,n;
{
    unsigned long int a,b,c,d,temp;

    a = m;
    b = n;
    if (a > b)        /* swap */
    {
        temp = a;
        a = b;
        b = temp;
    }
    while(a > 0)
    {
        c = b / a;
        d = b - a * c;
        b = a;
        a = d;
    }
    return(b);
} /* end of GreatestCommonDivisor() function */
/*=====*/
/* END OF FRACTSUM.C */

```

### 程序运行实例

```

FRACTSUM.EXE

a sequence of fractions to be summed ----
1/4 7/6 7/4 2/6 2/8 6/6 8/3 9/8 2/6 10/3
2/6 7/6 2/3 6/4 1/6 7/8 4/8 1/6 8/4 5/3
3/8 3/4 4/6 8/3 5/8 4/8 8/3 2/8 9/3 9/4
3/6 9/8 4/8 8/3 7/3 7/4 1/3 1/6 7/3 1/6
4/3 1/6 7/6 4/6 10/6 6/4 1/3 5/8 8/8 3/6
sum of fractions = 55 + 2/3

```

函数: Decfrac() MIC2.14

功能

转换一个十进制的数值(%), 用作节拍。

## 注释

主程序用欲转换到相关的最简分数去的十进制数填充浮点指针数组 X[]。为什么控制一个十进制数那么复杂呢? 因为一些混响器拒绝接受总数比  $1/128$  更小的分数输入。另外, 人类能感知的极限表达出来的特别复杂的分数, 它们根本毫无用处, 例如  $121347/291385$ 。也就是说, 人们不会把这类数写成分数, 而是直接写成小数。我们现在把最小节拍定为整个小节的  $1/100$ 。因为很少有人能够区别比这个数字小的节奏变化。

## 编程提示

生成一个用于一组音阶的十进制节拍, 用计算机音乐格式(每个十进制节拍都被打印出来了, 后面跟的是十进制的总计节拍。)用绘图仪在绘图纸上画出节奏和节拍, 然后把输出转化成可以演奏的图形符号, 例如箫的乐谱。

把被转换的分数的分子作为一个用 % 来衡量的标量, 它指明了清晰度或者音量参数; 也可以直接映射, 或者用随机的顺序。

把最后的数据送到一个混响器中, 用箫的音色演奏这个电子音乐。

## 程序清单

```
/* DECFRAC.C (converts decimals and decimal fractions
              to rationals) */
#include <stdio.h>
main()
{
    float x[50];
    int total = 50, prec, base, seed = 3243, j;
    void Decfrac();

    srand(seed);
    printf("%s", "Enter 1,2, or 3 for precision degree",
           " (1/10, 1/100, or 1/1000)\n");
    scanf("%d", &prec);
    printf("%s", "Loading random decimals containing integer",
           " and fractional\nparts:\n");

    switch(prec)
    {
        case 1 : base = 10;
                break;
        case 2 : base = 100;
                break;
        case 3 : base = 1000;
                break;
        default : base = 1000;
    }
    for (j = 0; j < total; j++)
    {
        x[j] = (rand() % 4) + (rand() % base) / (float)base;
        if (j % 6 == 0)
```

```

        printf("\n");
        printf("%g", x[j]);
    }
    printf("\n\nHere are the equivalent rational fractions:\n\n");
    Decfrac(x, total, base);
} /* end of main */
/*===== MIC 2.14 =====*/
/* Decfrac() function :NOTE for a higher degree of accuracy
                        when converting small decimal values,
                        raise base value exponentially,
                        ie. 1000,10000,etc. */
void Decfrac(x, total, base)
float x[];
int total, base;
{
    int orig, num, denom, a, b, c, d, temp, j;
    for (j = 0; j < total; j++)
    {
        orig = a = (x[j] * base) + .5;
        b = base;
        if (a > b)
        {
            temp = a;
            a = b;
            b = temp;
        }
        while(a > 0)
        {
            c = (b / a);
            d = b - a * c;
            b = a;
            a = d;
        }
        num = orig / b;
        denom = base / b;
        if (j % 6 == 0)
            printf("\n");
        printf("%d/%d", num, denom);
    }
} /* end of Decfrac() function */
/*=====*/
/* END OF DECFRAC.C */

```

### 程序运行实例

DECFRAC.EXE

Enter 1,2,or 3 for precision degree (1/10,1/100,or1/1000)  
Loading random decimals containing integer and fractional  
parts:

0.6	2.1	3.7	0.2	0.2	3.9
3.8	3.2	1.6	0.1	0.5	3.9
2.2	2.1	3.7	2.6	1.6	1.1
0.4	0	3.5	2.8	1.4	1.4
3.4	0.1	3.1	1.6	0.6	1.6
0.7	1.9	2.3	3.6	0.6	2.9
0.1	2	0.1	3.9	1.5	3.7
2.5	1.4	1.1	0.9	2.1	1.6
3.4	1				

Here are the equivalent rational fractions:

3/5	21/10	37/10	1/5	1/5	39/10
19/5	16/5	8/5	1/10	1/2	39/10
11/5	21/10	37/10	13/5	8/5	11/10
2/5	0/1	7/2	14/5	17/5	7/5
17/5	1/10	31/10	8/5	3/5	8/5
7/10	19/10	23/10	18/5	3/5	29/10
1/10	2/1	1/10	39/10	3/2	37/10
5/2	7/5	11/10	9/10	21/10	8/5
17/5	1/1				

## 函数: Fracdec() MIC2.15

### 功能

把一个分数的节拍值转换成用十进制值表示, 并且传递另外的音乐参数。

### 注释

用户可以翻译一系列分数型节拍——1/2, 3/4, 5/8 等等——成为十进制数, 用于变换节奏参数到另外一个参数。在变换时, 一组数值是与要转换的乐曲结构参数有一定的关系。例如, 把节奏变换到音量。首先把分数转换成十进制, 然后换算到音量参数的允许范围以内。(为了做到这一点, 用一个合适的换算因子乘以每个十进制数, 然后把它们取整。)

### 编程提示

生成一个随机顺序的、多变量的、分数型节拍的数组, 然后把分数转换成十进制数, 换算到允许范围内, 并且取整。再把这些数值作为混响器或者是独奏的乐器的清晰度参数。

用列出来的数组, 把它们转换并换算允许的范围, 作为指向另外音乐元素数组的指针。例如, 通过把指针按倒序排序或不排序, 生成音阶、音量、音色等参数。

### 程序清单

```
/* FRACDEC.C (converts fractional value to decimal
   for use as rhythm values; it also computes
   the sum of durations in whole notes) */
#include <stdio.h>
main()
{
    int j, total = 20, seed = 323;
    double x[20], y[20], z[20], sum, Fracdec();
```

```

srand(seed);
printf("loading random fractions:\n");
for (j = 0; j < total;j++)
{
    x[j] = rand() % total + 1; /*load numerators*/
    y[j] = rand() % total + 1; /*load denominators*/
}
sum = Fracdec(x,y,z,total);
for (j = 0; j < total;j++)
    printf("%.0f/%.0f = %g\n",x[j],y[j],z[j]);

    printf("sum = %f whole notes.",sum);
} /* end of main */
/*===== MIC 2.15 =====*/
/* Fracdec() function */
double Fracdec(x,y,z,total)
int total;
double x[],y[],z[];
{
    double sum = 0.0;
    int j;

    for (j = 0;j < total;j++)
    {
        z[j] = x[j] / y[j];
        sum += z[j];
    }
    return(sum);
} /* end of Fracdec() function */
/*=====*/
/* END OF FRACDEC.C */

```

### 程序运行实例

```

FRACDEC.EXE

loading random fractions:
13/13 =1
18/9 =2
14/4 =3.5
8/2 =4
6/16 =0.375
12/15 =0.8
7/17 =0.4117647
2/9 =0.2222222
19/18 =1.055556
3/9 =0.3333333
16/3 =5.333333
5/16 =0.3125
14/10 =1.4
20/18 =1.111111
4/15 =0.2666667
3/10 =0.3
4/1 =4
3/16 =0.1875
15/7 =2.142857
9/18 =0.5
sum = 29.251844 whole notes.

```

函数组: Dechidec.c

1. DecToBin()      MIC2.16
2. BinToDec()      MIC2.17
3. StringRev()      MIC2.18
4. StringLength()   MIC2.19

## 功能

把一组的十进制整数转换成二进制相等的字符数组，或者相反。

## 注释

二进制数有一些有意义的复杂的应用。一个特殊的应用是在一个微小的可测量的连续域中记录打击乐器。数字中的位的状态代表音调的 on / off 两个状态，一般是两个状态中的一个，开或者是关，在开的状态下，一个二进制是从左到右读入的。当一个 1 读到时，一个声音会开始响，直到读入另外一个 1(如果用户这样选择的话)。在关的状态下，每位的作用如下所述：当读到一个 1 时，一个小节开始；它在下一位读入时结束，而下一位可能是另外一个音调或者无声(0)，数字的每位都代表作曲家定义的一空拍，或者是一些被赋予的状态。

例如，给出二进制数 1100 / 0 / 0，音阶 C#, G, Ab C 和一个四分之一节拍的空拍，下面的关系就会在关状态中出现：

(R = Rest)

P(itch)	C#	G	R	R	Ab	R	C	R
R(hythm)	1 / 4	1 / 4	1 / 4	1 / 4	1 / 4	1 / 4	1 / 4	1 / 4

而在开状态中，会得出：

P(itch)	C#	G	Ab	C
R(hythm)	1 / 4	3 / 4	1 / 2	1 / 2

## 编程提示

写一个在开或关模式下应用二进制数的程序。用用户自己定义的规则实现它们。

## 程序清单

```
/* DECBIDEC.C (converts a decimal integer value
               to a binary string, then back to decimal)*/
#include <stdio.h>
#define ToAscii(x) ((x >= 0 && x <= 9) ? x + '0' : -1)
#define IsBinary(x) ((x == '0' || x == '1') ? 1 : 0)
#define ToDecimal(x) (x - '0')
#define TOTAL 20
main()
{
```

```

char binary[12];
unsigned int decnum;
int DecToBin();
int j;
printf("%s", "A sequence of decimal to binary",
      " to decimal conversions:\n");
for (j = 0; j < TOTAL; j++)
{
    decnum = (j+1) * 100; /*compute a decimal value*/
    DecToBin(decnum, binary);
    printf("\ndecimal integer %d ", decnum);
    printf(" in binary = %s\n", binary);
    BinToDec(binary, decnum);
    printf("converted back to decimal = %d", decnum);
} /* end of main */
/*===== MIC 2.16 =====*/
/* DecToBin() function (This function converts an array
of positive decimal integers to
an array of binary equivalents.) */

int DecToBin(decnum, binary)
unsigned int decnum;
char binary[12];
{
    int remainder, index = 0, count = 0;
    void StringRev();
    while (decnum / 2.0 != 0)
    {
        remainder = decnum % 2;
        binary[index++] = ToAscii(remainder);
        decnum = decnum / 2;
        count ++;
    }
    while (count++ <= 15)
        binary[index++] = '0';
    binary[index] = NULL;
    StringRev(binary, 0, index);
} /* end of DecToBin() function */
/*===== MIC 2.18 =====*/
/* StringRev() function */
void StringRev(string, start, end)
char string[];
int start, end;
{
    int length;
    char temp;
    int StringLength();
    length = StringLength(string);
    if (end >= length)
        end = length - 1;
    if (start >= end)
        return;
    else
    {
        temp = string[start];
        string[start] = string[end];
        string[end] = temp;
        StringRev(string, ++start, --end);
    }
} /* end of StringRev() function */

```

```

/*===== MIC 2.19 =====*/
/* StringLength() function */
int StringLength(string)
char *string;
{
    if (*string == NULL)
        return(0);
    else
        return(1 + StringLength(++string));
} /* end of StringLength() function */
/*===== MIC 2.17 =====*/
/* BinToDec() function (converts binary numbers to
                        their decimal equivalents.) */
int BinToDec(binary, decnum)
char *binary;
unsigned int *decnum;
{
    *decnum = 0;

    while (*binary)
        if (IsBinary(*binary))
            *decnum = *decnum * 2 + ToDecimal(*binary++);
        else
            return(-1);
    return(0);
}
/* end of BinToDec */
/*=====*/
/* END OF DECBIDEC.C */

```

## 程序运行实例

DECBIDEC.EXE

A sequence of decimal to binary to decimal conversions

```

decimal integer 100 in binary = 0000000001100100
converted back to decimal = 100
decimal integer 200 in binary = 0000000011001000
converted back to decimal = 200
decimal integer 300 in binary = 0000000100101100
converted back to decimal = 300
decimal integer 400 in binary = 0000000110010000
converted back to decimal = 400
decimal integer 500 in binary = 0000000111101000
converted back to decimal = 500
decimal integer 600 in binary = 0000001001011000
converted back to decimal = 600
decimal integer 700 in binary = 0000001010111100
converted back to decimal = 700
decimal integer 800 in binary = 0000001100100000
converted back to decimal = 800
decimal integer 900 in binary = 0000001110000100
converted back to decimal = 900
decimal integer 1000 in binary = 0000001111010000
converted back to decimal = 1000
decimal integer 1100 in binary = 0000010001001100
converted back to decimal = 1100
decimal integer 1200 in binary = 0000010010110000
converted back to decimal = 1200
decimal integer 1300 in binary = 0000010100010100

```



```

converted back to decimal = 1300
decimal integer 1400 in binary = 0000010101111000
converted back to decimal = 1400
decimal integer 1500 in binary = 0000010111011100
converted back to decimal = 1500
decimal integer 1600 in binary = 0000011001000000
converted back to decimal = 1600
decimal integer 1700 in binary = 0000011010100100
converted back to decimal = 1700
decimal integer 1800 in binary = 0000011100001000
converted back to decimal = 1800
decimal integer 1900 in binary = 0000011101101100
converted back to decimal = 1900
decimal integer 2000 in binary = 0000011110100000
converted back to decimal = 2000

```

## 函数: Stirling( ) MIC2.20

### 功能

提供一个分解整数因数的捷径。这个程序得到 *Stirling* 的近似值。在统计学应用程序中，可以用来计算概率、排列、组合。

### 注释

这个算法只用来计算自然对数大约大于十的因数，而遇到小于十一的数时，它就返回  $\ln(n!)$ 。这个算法是十分迅速的，并且在大多数情况下，它的精度是足够的。

### 编程提示

参见 MIC 2.21 Permut0( )和 MIC2.22 Combntot( )。

### 程序清单

```

/* STIRLING.C (computes factorial approximations) */
#include <stdio.h>
#include <math.h>
main()
{
    int j;
    double num; /*number for which factorial is to be computed*/
    double Stirling(), exp();
    double appr; /*Stirling's approximation of log of number*/

    for (j = 9; j <= 12; j++)
    {
        num = j;
        printf("%s%s%.0f", "Integer for which factorial",
            " will be computed is:", num);
        appr = Stirling(num);
        printf("\nlog of factorial is appr. %f\n", appr);
        if (num < 34.)
            printf("factorial is appr.%.0f\n", (exp(appr)+.5));
    }
}

```

```

    )
} /* end of main */
/*----- MIC 2.20 -----*/
/* Stirling() function */
double Stirling(num)
double num;
{
    double appr = 1.0;
    double log();
    int j;
    if (num <= 0.0)
    {
        appr = 0.0;
        return(appr);
    }
    for (j = 1; j <= 10; j++)
    {
        appr = appr * j;
        if (num == j)
        {
            appr = log(appr);
            return(appr);
        }
    }
    appr = log(6.283186L) / 2.0 + log(num) *
        (num + .5) - num + 1 / (12 * num);
    return(appr);
} /* end of Stirling() function */
/*-----*/
/* END OF STIRLING.C */

```

## 程序运行实例

### STIRLING.EXE

```

Integer for which factorial will be computed is:9
log of factorial is appr. 12.801827
factorial is appr.362880
Integer for which factorial will be computed is:10
log of factorial is appr. 15.104413
factorial is appr.3628801
Integer for which factorial will be computed is:11
log of factorial is appr. 17.502310
factorial is appr.39916886
Integer for which factorial will be computed is:12
log of factorial is appr. 19.987216
factorial is appr.479002395

```

## 函数组: Permuto.c

1. Permuto() MIC2.21
2. String() MIC2.20

## 功能

计算排列  $P_n^m$  的值。

### 注释

作曲家常常需要知道一组有限元素的可能的排列数。但这并不是说，在一个曲子中，可以把这些可能的排列全部用上。这仅仅是决策过程中的一个出发点。实际上，如果用一个人八度中的所有半音——只有十二个音节——的可能的排列来写曲子，这会需要一个人的一生来完成，然后再花费另一生来演奏这首曲子。因为这总共有十二个阶乘，大约 479001600 个可能的排列方法。

对于一组给定元素的排列，如果一次取少于元素总数的量进行排列，那么所得到的排列数就会少得多。例如，对十二个半音阶进行排列，每次取两个，这个排列数就减少到一百三十二。

这种计算可以用袖珍型计算器完成，但是为什么需要这个函数呢？关键是在于它为高级程序的数据结构提供了一些有用的信息。

### 编程提示

1. 写一个反馈排列信息给一个算法的程序，这个算法用以提供系统性半音阶组的排列。
2. 写一个程序用以观察一组随机产生的参数(比如是音阶)的所有可能的排列，然后向程序返回参数用以控制下一个参数。

### 程序清单

```
/* PERMUTOT.C (computes the number of permutations
               of n elements taken m at a time.) */
#include <stdio.h>
#include <math.h>
main()
{
    int elements = 1; /*number of items to permute*/
    int many = 1; /*items to be taken at a time*/
    int j;
    double permutations, Permutot();
    for (j = 0; j < 10; j++)
    {
        permutations = Permutot(elements, many);
        printf("permutations of %d elements", elements);
        printf(" taken %d at a time", many);
        printf("\nare %.0f\n", permutations);
        elements += 3;
        many += 2;
    }
} /* end of main */
/*===== MIC 2.21 =====*/
double Permutot(elements, many)
int elements, many;
{
```

```

double num;
double elemfact = 0.0; /*log of 'elements' factorial*/
double manyfact = 0.0; /*log of 'many' factorial*/
double permutations = 0.0;
double Stirling(),exp();
double appr;

num = elements;
elemfact = Stirling(num);
num = elements-many;
manyfact = Stirling(num);
permutations = exp(elemfact-manyfact);
return(permutations);
} /* end of Permutot() function */
/*===== MIC 2.20 =====*/
/* Stirling() function */
double Stirling(num)
double num;
{
double appr = 1.0;
double log();
int j;
if (num <= 0.0)
{
appr = 0.0;
return(appr);
}
for (j = 1; j <= 10; j++)
{
appr = appr * j;
if (num == j)
{
appr = log(appr);
return(appr);
}
}
appr = (log(6.283186L) / 2.0 + log(num) *
(num + .5) - num + 1 / (12 * num));
return(appr);
} /* end of Stirling() function */
/*=====*/
/* END OF PERMUTOT.C */

```

## 程序运行实例

PERMUTOT.EXE

```

permutations of 1 elements taken 1 at a time
are 1
permutations of 4 elements taken 3 at a time
are 24
permutations of 7 elements taken 5 at a time
are 2520
permutations of 10 elements taken 7 at a time
are 604800
permutations of 13 elements taken 9 at a time
are 259459542
permutations of 16 elements taken 11 at a time
are 174356710124
permutations of 19 elements taken 13 at a time
are 168951606028845
permutations of 22 elements taken 15 at a time
are 223016087856331552

```

```

permutations of 25 elements taken 17 at a time
are 3.84702719613062152e+020
permutations of 28 elements taken 19 at a time
are 8.40190696613826175e+023

```

## 函数组: Combntot.C

1. Combntot() MIC2.22
2. String() MIC2.20

### 功能

计算组合  $C_n^m$  的值, 在这里, 组合中各个元素的顺序是无关的。

### 注释

排列是对每个物体的位置的安排, 而组合只是简单地把它们选出来就可以了。举个例子来说, 就象是问这样一个问题: 十二个音阶有多少不同的方法在三个音节中出现? (这个答案是二百二十个)。如果用户只对几个里面取  $m$  个的唯一组合感兴趣, 那么马上就可以得到一组快速递减的数字。这些数字是在单个组合的范围里的。

如果用户希望通过在一个八度的半音阶中生成一个随机顺序的六音节组来写一个面向过程的乐段(这个乐段只由音阶构成)。不考虑排列的话, 可以生成九百二十四种不同的组合, 每个组合都由六个音节组成。在这个乐段中总共就有 5,544 个音阶。(对于一组给定  $n$  个元素的集合来说, 最大的组合数出现在  $m$  值接近于  $n$  的一半的时候。例如, 十二个元素中取七个元素进行唯一组合的种类有 792 种, 这相当于在十二个元素中取五个元素进行唯一组合。

### 编程提示

写一个计算并且记录所有可能的组合的程序。元素总数为十二个, 用户在命令行输入所取元素的个数。

### 程序清单

```

/* COMBNTOT.C (computes the number of combinations
               of n elements taken m at a time.) */
#include <stdio.h>
#include <math.h>
main()
{
    int elements = 10; /* number of items */
    int many = 2; /* number of items taken at a time */
    int j;
    double combinations, Combntot();
    for(j = 0; j < 10; j++)
    {

```

```

        combinations = Combntot(elements,many);
        printf("combinations of %d elements",elements);
        printf (" taken %d at a time",many);
        printf("\nare %.0f\n",combinations);
        elements += 3;
        many += 3;
    }
} /* end of main */
/***** MIC 2.22 *****/
/* Combntot() function */
double Combntot(elements,many)
int elements,many;
{
    double num;
    double elemfact; /*log of 'elements' factorial*/
    double manyfact; /*log of 'many' factorial*/
    double diffact; /*log of (elemfact-manyfact) factorial*/
    double combinations;
    double Stirling(),exp();
    double appr;

    num = elements;
    elemfact = Stirling(num);
    num = many;
    manyfact = Stirling(num);
    num = elements - many;
    diffact = Stirling(num);
    combinations = exp(elemfact - (manyfact + diffact));
    return(combinations);
} /* end of Combntot() function */
/***** MIC 2.20 *****/
/* Stirling() function */
double Stirling(num)
double num;
{
    double appr = 1.0;
    double log();
    int j;
    if(num <= 0.0)
    {
        appr = 0.0;
        return(appr);
    }
    for(j = 1;j <= 10;j++)
    {
        appr = appr * j;
        if(num == j)
        {
            appr = log(appr);
            return(appr);
        }
    }
    appr = log(6.283186L) / 2.0 + log(num) *
        (num + .5) - num + 1 / (12 * num);
    return(appr);
} /* end of Stirling() function */
/***** END OF COMBNTOT.C */

```

## 程序运行实例

COMBNTOT.EXE

```
combinations of 10 elements taken 2 at a time  
are 45  
combinations of 13 elements taken 5 at a time  
are 1287  
combinations of 16 elements taken 8 at a time  
are 12870  
combinations of 19 elements taken 11 at a time  
are 75582  
combinations of 22 elements taken 14 at a time  
are 319770  
combinations of 25 elements taken 17 at a time  
are 1081575  
combinations of 28 elements taken 23 at a time  
are 3108104  
combinations of 31 elements taken 23 at a time  
are 7888724  
combinations of 34 elements taken 26 at a time  
are 18156202  
combinations of 37 elements taken 29 at a time  
are 38608013
```

## 函数组: Normtbl.c

1. Freqtbl() MIC2.23
2. Datanorm() MIC2.24
3. Zeromat() MIC2.25

## 功能

对指定的音乐参数进行概率分布函数的测试。

## 注释

为了便于进行概率分布函数的测试，需要通过许多不同的例子得到一个特性曲线。虽然用户不想看一幅详细的分布图，但可能要看一直大概的分布图，以便和理想曲线进行比较。因此，用户有必要用 Datanorm()函数把得到的数据归整，以便在显示器上显示。另外，最重要的一点是必须用 Freqtbl()函数生成一张出现概率的表格，用它来提供画条形图或者类似图形的数据。

## 编程提示

1. 用在第四章“概率分布函数”中提到的函数修改主函数的数据生成部分。
2. 使 NORMTABL.C 程序用直方图观察分布函数。

## 程序清单

```

/* NORMTABL.C (create a data frequency
               table and normalize values for the table)*/
#include <stdio.h>
main()
{
    int x[20], range = 20, total = 1000, seed = -1011, j;
    float y[20];
    int srand();
    void Freqtabl(), Datanorm(), Zeromat();

    srand(seed);
    printf("integer value\tnormalized value\tfrequency\n");
    Zeromat(x, y, range);
    Freqtabl(x, range, total);
    Datanorm(x, y, range);
    for (j = 0; j < range; j++)
        printf("\t%d\t\t %.3f\t\t %d\n", j, y[j], x[j]);
} /* end of main */
/*===== MIC 2.23 =====*/
/* Freqtabl() function */
void Freqtabl(x, range, total)
int x[], range, total;
{
    int num, j, rand();

    for (j = 0; j < total; j++)
    {
        num = rand() % range;
        x[num] = x[num] + 1;
    }
} /* end of Freqtabl() function */
/*===== MIC 2.24 =====*/
/* Datanorm() function */
void Datanorm(x, y, range)
int x[], range;
float y[];
{
    int j;
    float sum = 0.0;

    for (j = 0; j < range; j++)
        sum += x[j];
    for (j = 0; j < range; j++)
        y[j] = x[j] / sum;
} /* end of Datanorm() function */
/*===== MIC 2.25 =====*/
/* Zeromat() function */
void Zeromat(x, y, range)
int x[], range;
float y[];
{
    int j;
    for (j = 0; j < range; j++)
    {
        x[j] = 0;
        y[j] = 0.0;
    }
} /* end of Zeromat() function */
/*===== MIC 2.26 =====*/
/* END OF NORMTABL.C */

```



## 程序运行实例

### NORMTABL.EXE

integer value	normalized value	frequency
0	0.059	59
1	0.044	44
2	0.043	43
3	0.052	52
4	0.053	53
5	0.056	56
6	0.049	48
7	0.049	49
8	0.041	41
9	0.036	36
10	0.051	51
11	0.042	42
12	0.047	47
13	0.045	45
14	0.052	52
15	0.063	63
16	0.067	67
17	0.060	60
18	0.043	43
19	0.049	49

### 函数组: Scaler.c

1. AllScaler() MIC2.35
2. Scaler() MIC2.36
3. Inverse() MIC2.37

### 功能

把一个整数数组的值归整到 84 个或者其中几个不同的范围中去。也就是把一个参数的数据转换到另一参数的范围中去。

### 注释

AllScaler()函数把所有的音域定义为从 1 到 84，然后每个都归整为零。Scaler()函数给出特定的音域。

### 注意

注意在编译时必须访问另外三个子程序文件(见附录 B)。

array.c MIC\_SP1.0  
synclavi.c MIC\_SP7.0  
randmain.c MIC\_SP19.0

这些文件都需要调用别的子程序，所以应该在 `determine dependencies` 一项顶部按顺序列出。

举个例子来说，为了便于测试以音阶参数生成的数据，作曲家有可能希望把这些归整了的序列转换成节奏或音品参数，以测试数据的可行性。因此，这些数据必须被转换成和目标参数的范围一致。

### 编程提示

1. 用第四章“概率分布函数”中提到的函数修改主函数手数据生成部分。
2. 使 `SCALER.C` 程序用直方图观察分布函数。
3. 修改主程序，使用户可以指明特殊的输出转换。

### 程序清单

```
SCALER.C

#include <stdio.h>
#define MAXDATA 100
#include "randmain.c"
#include "synclavi.c"
#include "array.c"
main()
{
    int datarray[MAXDATA], inverse[MAXDATA], j, size;

    size = RandMain(datarray);
    printf("\n\nInverse\n");
    Inverse(datarray, inverse, size);
    for(j = 0; j < size; j++)
        printf("%d ", inverse[j]);
    printf("\n\noriginal scaled to gamut of 20");
    scaler(datarray, size, 20);
    printf("\n\ninverse scaled to gamut of 20");
    scaler(inverse, size, 20);
    printf("\n\nall gamuts");
    AllScaler(datarray, size);
} /* end of main() */
/*===== MIC 2.35 =====*/
AllScaler(datarray, size)
    int datarray[], size;
{
    int j, max, min, pc[MAXDATA];
    float factor, pc_float, min1 = 214000.0;
    float k, range;

    max = getmax(datarray, size);
    min = getmin(datarray, size);
    range = max - min;
    for(k = 1; k <= 34; k++)
    {
        factor = k / (range + (range / k));
        printf("\n\ngamut = %f scaling factor = %f\n", k, factor);
        min1 = 210000.0;

        for(j = 0; j < size; j++)
        {
```

```

        pc_float = datarray[j] * factor;
        if (pc_float < min1)
            min1 = pc_float;
    }
    for(j = 0; j < size; j++)
    {
        pc_float = datarray[j] * factor - min1;
        pc[j] = (int)(pc_float + 0.5);
    }
    PutArray(pc, size);
    ScriptArray(pc, ' ', size);
}
}
/*===== MIC 2.36 =====*/
Scaler(int datarray[], int size, float gamut)
{
    int disarray[MAXDATA], j, min, max;
    float range, factor;

    max = getmax(datarray, size);
    min = getmin(datarray, size);
    range = max - min;
    factor = gamut / (range + (range / gamut));

    printf("\n\ngamut = %f scaling factor = %f\n", gamut, factor);
    for(j = 0; j < size; j++)
        printf("%d ", (int)(datarray[j] * factor - min + 0.5));
}
/*===== MIC 2.37 =====*/
Inverse(int datarray[], int inv[], int size)
{
    int j, min, max, range;

    max = getmax(datarray, size);
    min = getmin(datarray, size);
    range = max - min;
    for(j = 0; j < size; j++)
        inv[j] = abs(datarray[j] - (range + 1)); /* invert */
} /* end of scaler() function */
/*=====*/

```

### 程序运行实例

```

How many random numbers do you want?    10
Enter lowest number in gamut           0
Enter highest number in gamut          10
4 10 0 0 7 3 5 9 6 5

Inverse
7 1 11 11 4 8 6 2 5 6

original scaled to gamut of 20

gamut = 20.000000 scaling factor = 1.904762
8 19 0 0 13 6 10 17 11 10

inverse scaled to gamut of 20

gamut = 20.000000 scaling factor = 1.904762
12 1 20 20 7 14 10 3 9 10

```

```

all gamuts

gamut = 1.000000 scaling factor = 0.050000
0 1 0 0 0 0 0 0 0 0
C0 C#0 C0 C0 C0 C0 C0 C0 C0 C0
gamut = 2.000000 scaling factor = 0.133333
1 1 0 0 1 0 1 1 1 1
C#0 C#0 C0 C0 C#0 C0 C#0 C#0 C#0 C#0
gamut = 3.000000 scaling factor = 0.225000
1 2 0 0 2 1 1 2 1 1
C#0 D0 C0 C0 D0 C#0 C#0 D0 C#0 C#0
gamut = 4.000000 scaling factor = 0.320000
1 3 0 0 2 1 2 3 2 2
C#0 D#0 C0 C0 D0 C#0 D0 D#0 D0 D0
gamut = 51.000000 scaling factor = 5.001923
20 50 0 0 35 15 25 45 30 25
G#1 D4 C0 C0 B2 D#1 C#2 A3 F#2 C#2
gamut = 52.000000 scaling factor = 5.101887
20 51 0 0 36 16 26 46 31 26
G#1 D#4 C0 C0 C3 D#1 D2 A#3 G2 D2
gamut = 53.000000 scaling factor = 5.201852
21 52 0 0 36 16 26 47 31 26
A1 E4 C0 C0 C3 E1 D2 B3 G2 D2
gamut = 54.000000 scaling factor = 5.301818
21 53 0 0 37 16 27 48 32 27
A1 F4 C0 C0 C#3 E1 D#2 C4 G#2 D#2
gamut = 55.000000 scaling factor = 5.401786
22 54 0 0 38 16 27 49 32 27
A#1 F#4 C0 C0 D3 E1 D#2 C#4 G#2 D#2
gamut = 56.000000 scaling factor = 5.501754
22 55 0 0 39 17 28 50 33 28
A#1 G4 C0 C0 D#3 F1 E2 D4 A2 E2
gamut = 57.000000 scaling factor = 5.601724
22 56 0 0 39 17 28 50 34 28
A#1 G#4 C0 C0 D#3 F1 E2 D4 A#2 E2
gamut = 58.000000 scaling factor = 5.701695
23 57 0 0 40 17 29 51 34 29
B1 A4 C0 C0 E3 F1 F2 D#4 A#2 F2

```

gamut = 59.000000 scaling factor = 5.801667  
 23 58 0 0 41 17 29 52 35 29

B1 A#4 C0 C0 F3 F1 F2 54 B2 F2

gamut = 60.000000 scaling factor = 5.901639  
 24 59 0 0 41 18 30 53 35 30

C2 B4 C0 C0 F3 F#1 F#2 F4 B2 F#2

.  
.  
.  
.  
.

gamut = 76.000000 scaling factor = 7.501299  
 30 75 0 0 53 23 38 68 45 38

F#2 D#6 C0 C0 F4 B1 D3 G#5 A3 D3

gamut = 5.000000 scaling factor = 0.416667  
 2 4 0 0 3 1 2 4 3 2

D0 E0 C0 C0 D#0 C#0 D0 E0 D#0 D0

gamut = 6.000000 scaling factor = 0.514286  
 2 5 0 0 4 2 3 5 3 3

D0 F0 C0 C0 E0 D0 D#0 F0 D#0 D#0

gamut = 7.000000 scaling factor = 0.612500  
 2 6 0 0 4 2 3 6 4 3

D0 F#0 C0 C0 E0 D0 D#0 F#0 E0 D#0

gamut = 8.000000 scaling factor = 0.711111  
 3 7 0 0 5 2 4 6 4 4

D#0 G0 C0 C0 F0 D0 E0 F#0 E0 E0

gamut = 9.000000 scaling factor = 0.810000  
 3 8 0 0 6 2 4 7 5 4

D#0 G#0 C0 C0 F#0 D0 E0 G0 F0 E0

gamut = 10.000000 scaling factor = 0.909091  
 4 9 0 0 6 2 5 8 5 5

E0 A0 C0 C0 F#0 D#0 F0 G#0 F0 F0

gamut = 11.000000 scaling factor = 1.008333  
 4 10 0 0 7 3 5 9 6 5

E0 A#0 C0 C0 G0 D#0 F0 A0 F#0 F0

gamut = 12.000000 scaling factor = 1.107692  
 4 11 0 0 8 3 6 10 7 6

E0 B0 C0 C0 G#0 D#0 F#0 A#0 G0 F#0

gamut = 48.000000 scaling factor = 4.702041  
19 47 0 0 33 14 24 42 28 24

G1 B3 C0 C0 A2 D1 C2 F#3 E2 C2

gamut = 49.000000 scaling factor = 4.802000  
19 48 0 0 34 14 24 43 29 24

G1 C4 C0 C0 A#2 D1 C2 G3 F2 C2

gamut = 50.000000 scaling factor = 4.901961  
20 49 0 0 34 15 25 44 29 25

G#1 C#4 C0 C0 A#2 D#1 C#2 G#3 F2 C#2

gamut = 77.000000 scaling factor = 7.601232  
30 76 0 0 53 23 38 68 46 38

F#2 E6 C0 C0 F4 B1 D3 G#5 A#3 D3

gamut = 78.000000 scaling factor = 7.701266  
31 77 0 0 54 23 39 69 46 39

G2 F6 C0 C0 F#4 B1 D#3 A5 A#3 D#3

gamut = 79.000000 scaling factor = 7.801250  
31 78 0 0 55 23 39 70 47 39

G2 F#6 C0 C0 G4 B1 D#3 A#5 B3 D#3

gamut = 80.000000 scaling factor = 7.901235  
32 79 0 0 55 24 40 71 47 40

G#2 G6 C0 C0 G4 C2 E3 B5 B3 E3

gamut = 81.000000 scaling factor = 8.001220  
32 80 0 0 56 24 40 72 48 40

G#2 G#6 C0 C0 G#4 C2 E3 G6 C4 E1

gamut = 82.000000 scaling factor = 8.101205  
32 81 0 0 57 24 41 73 49 41

G#2 A6 C0 C0 A4 C2 F3 C#6 C#4 F3

gamut = 83.000000 scaling factor = 8.201191  
33 82 0 0 57 25 41 74 49 41

A2 A#6 C0 C0 A4 C#2 F3 D6 C#4 F3

gamut = 84.000000 scaling factor = 8.301176  
33 83 0 0 58 25 42 75 50 42

A2 B6 C0 C0 A#4 C#2 F#3 D#6 D4 F#3

## 第三章 数列与动态运算

### 函数组: Motforms.c

1. Printpitch() MIC3.1
2. Motretro() MIC3.2
3. Motinvrt() MIC3.3
4. Motrnpz() MIC3.4

### 功能

用三种标准化的方法处理一连串长短不一的音符(音乐主题)。

- a. 逆序
- b. 音程倒置
- c. 音程转换

### 注释

主程序变量“total”能被转换用来存放任意长度的音符，但是相应的数组维数必须进行相应的修改。

当修改驱动程序以允许使用者输入乐曲时，应特别注意防止音符超出音符元素表的限制。如果发生了上述的溢出，我们可以通过插入附加数码而使输出的数据回到限制的音高范围内，这就防止了程序运行出错。

### 编程提示

1. 将主程序变成交互式，允许用户输入音符并且选择程序。
2. 转换程序以从源文件中读入音高数据，并以不同的方法处理它，然后将其传送到目标文件。

### 程序清单

```
/* MOTFORMS.C (processes pitch sequences - motifs - by any of
   3 methods: retrograde, inversion, transposition) */
#include <stdio.h>
main()
{
```

```

int orig[12];      /*stores original pitch motif*/
int retro[12];     /*stores retrograde motif*/
int invert[12];    /*stores inverted motif*/
int motrans[12];   /*stores motif transposition*/
int transint = -3; /*transposition interval*/
int total = 12,j,seed = -2215;
void Printpitch(),Motretro(),
    Motinvrt(),Motrnpx();

srand(seed);
printf("\noriginal pitch motif --\n");
for (j = 0;j < total;j++)
    /*generate a midrange random-order pitch motif*/
    orig[j] = rand() % 12 + 30;
Printpitch(orig,total);
Motretro(orig,retro,total);
printf("\nretrograde motif --\n");
Printpitch(retro,total);
Motinvrt(orig,invert,total);
printf("\ninverted motif --\n");
Printpitch(invert,total);
Motrnpx(orig,motrans,transint,total);
printf("\nmotif transposed by %d interval--\n",transint);
Printpitch(motrans,total);
}/* end of main */
/*===== MIC 3.1 =====*/
/* Printpitch() function - an adaptation of MIC 2.3,
    Pitchtab() function */
void Printpitch(array,total)
int array[],total;
{
    static char *pitch[] = {
        "C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
        "A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
        "F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
        "D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
        "C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
        "A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
        "F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
        "D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
        "C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
        "A6 ", "A#6", "B6 ", "C7 " };
    int j;

    for (j = 0;j < total;j++)

        printf("%s ",pitch[array[j]]);
    printf("\n");
} /* end Printpitch() function */
/*===== MIC 3.2 =====*/
/* Motretro() function - accepts an input pitch
    sequence and outputs its
    retrograde form.*/
void Motretro(array1,array2,total)
int array1[],array2[],total;
{
    int j,l;

    for (j = 0;j < total;j++)
    {
        l = total - j - 1;
        array2[l] = array1[j];
    }
}

```



```

    } /* end Motretro() function */
    /*===== MIC 3.3 =====*/
    /* Motinvrt() function - accepts an input pitch sequence
                           and outputs its mirror image
                           form */
    void Motinvrt(array1,array2,total)
    int array1[],array2[],total;
    {
        int j;

        array2[0] = array1[0];
        for (j = 1;j < total;j++)
            array2[j] = array2[j-1] -
                (array1[j] - array1[j-1]);

    } /* end Motinvrt() function */
    /*===== MIC 3.4 =====*/
    /* Motrnpz() function - transposes an input motif by
                           a specified number of 1/2 steps. Transposition
                           direction is determined by the sign of the
                           transposition interval.*/
    void Motrnpz(array1,array2,tintval,total)
    int array1[],array2[],tintval,total;
    {
        int j;

        for (j = 0;j < total;j++)
            array2[j] = array1[j] + tintval;

    } /* end Motrnpz() function */
    /*=====*/
    /* END OF MOTFORMS.C */

```

### 程序运行实例

MOTFORMS.EXE

original pitch motif --

A#2 B2 G#2 A#2 D3 G2 A2 C3 F3 A#2 G2 G2

retrograde motif --

G2 G2 A#2 F3 C3 A2 G2 D3 A#2 G#2 B2 A#2

inverted motif --

A#2 A2 C3 A#2 F#2 C#3 B2 G#2 G#2 A#2 C#3 C#3

motif transposed by -3 interval--

G2 G#2 F2 G2 B2 E2 F#2 A2 D3 G2 E2 E2

### 函数组: Displace.c

1. Displace() MIC3.5
2. Printpitch() MIC2.3

### 功能

通过将音符移动指定数量的音程而变成音乐。

## 注释

该主程序可以通过重新声名数组 `orig[]`, `final[]` 和 `disp[]` 的维数来使其存贮更多的音符, 这样主程序就变成了一个交互式的程序, 加入适当的输入循环语句可以使用户更加迅速地输入音符和转换音程。

## 编程提示

1. 将主程序变成交互式可以使之接受任意长度的乐段(最多含有100个音符)。
2. 编写一个程序, 应用一系列节拍参数的无序数值对某一音列各音升或降八度, 但注意避免数字一样。例如, 音长值为1, (一个全音的对应升降)音程就不应该为一个八度。  
提示: 上述操作可通过将该列音长数组作为指向存放一系列无序音程值的数组元素的指针来实现。
3. 编写一交互式程序, 允许用户对输入音列进行反向、逆序、变调以及升(降)音程等各操作。
4. 将音程值数列的整数变为二进制形式, 试将这些二进制数灵活应用于节拍参数。

## 程序清单

```
srand(seed);
printf("\na random-order pitch sequence --\n");
for (j = 0; j < total; j++)
{
    orig[j] = rand() % 12+24; /*gen. a r-o sequence*/
    displint[j] = rand() % 5 - 2; /*gen. displacements*/
}
Printpitch(orig, total);
printf("\nrespective octave displacements:\n");
for (j = 0; j < total; j++)
    printf("%d ", displint[j]);
Displace(orig, final, displint, total);
printf("\nthe registered pitch sequence:\n");
Printpitch(final, total);
} /* end of main */
/*===== MIC 3.1 =====*/
/* Printpitch function */
void Printpitch(array, total)
int array[], total;
{
    static char *pitch[] = {
        "C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
        "A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
        "F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
        "D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
        "C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
        "A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
        "F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
        "D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
        "C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
    }
```

```

    "A6 ", "A#6", "B6 ", "C7 " );
    int j;
    for (j = 0; j < total; j++)
        printf("%s ", pitch[array[j]]);
    printf("\n");
} /* end Printpitch() function */
/*===== MIC 3.5 =====*/
/* Displace() function (individually displaces the pitches
of an input pitch sequence by a
specific number of octaves.) */
void Displace(array1, array2, array3, total)
int array1[], array2[], array3[], total;
{
    int j;
    for (j = 0; j < total; j++)
        array2[j] = array1[j] + 12 * array3[j];
} /* end Displace() function */
/*=====*/
/* END OF DISPLACE.C */

/* DISPLACE.C (individually displaces the pitches of an input
pitch sequence by a specific number of octaves) */
#include <stdio.h>
main()
{
    int orig[20];          /*prime sequence*/
    int final[20];         /*sequence after displacement*/
    int displint[20];      /*displacement interval array*/
    int total, j, seed = 554;
    void Printpitch(), Displace();
    total = 10;

```

### 程序运行实例

DISPLACE.EXE

```

a random-order pitch sequence --
F#2 C#2 C#2' C2 G2 G2 D2 C2 A2 A2

respective octave displacements:
0 -2 -1 1 0 -2 0 -2 -2 -1
the registrated pitch sequence:
F#2 C#0 C#1 C3 G2 G0 D2 C0 A0 A1

```

### 函数组: Alterseq.c

1. Alterseq() MIC3.6
2. Printpitch() MIC3.1

### 功能

通过一个常量系数，使间距扩展或压缩，来修正一个音调顺序。

## 注释

变更“intsize”如果为负，就会压缩节奏的间距(用 1/2 拍来测量)，这个间距的大小是在一个音调序列中相邻两个音符的音距，如果“intsize”为正，则这个间距将被扩展。

## 编程提示

1. 把主程序转换为一个交互式的程序，这个子程序允许用户输入一串整数值，它们代表存储的音调。用相同音调顺序运行几遍这个程序，但是在每次程序运行中，用一个有显著差异的系数来扩展和压缩间距。把主要的输出数据顺序画到音调参数中，然后把反常的形式中相同的数据画到旋律参数中。
2. 用相同输入整数序列，运行三次这个内部程序版本，但是要改变扩展/压缩间距，使原始的整数序列要适应音调的参数，把三个有变化的输出整数序列写到节奏、音量和清晰度参数中。

## 程序清单

```
/* ALTERSEQ.C {alters a value sequence by interval expansion
               or contraction. Positive integers expand the
               sequence,negative integers contract it.} */
#include <stdio.h>
#define abs_val(x) ({x >= 0 ? x : 0 - x})
main()
{
    int orig[10];      /* prime pitch sequence */
    int final[10];     /* expanded/contracted sequence */
    int intsize = 2;    /* interval-size for exp/contr */
    int total = 10,j,seed = 9154;
    void Printpitch(),Alterseq();

    srand(seed);
    printf("\na random-order pitch sequence --\n");
    for (j = 0;j <= total-1;j++)
        orig[j] = rand() % 22 + 22; /*gen. a r-o sequence*/
    Printpitch(orig,total);
    printf("\nthe sequence expanded by");
    printf(" %d semitones:\n",intsize);
    Alterseq(orig,final,intsize,total);
    Printpitch(final,total);
    printf("\nthe sequence contracted by");
    printf(" %d semitones:\n",intsize);
    intsize = -2;
    Alterseq(orig,final,intsize,total);
    Printpitch(final,total);
} /* end of main */
/***** MIC 3.1 *****/
/* Printpitch() function */
void Printpitch(array,total)
```

```

int array[],total;
{
static char *pitch[] = {
"C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
"A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
"F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
"D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
"C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
"A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
"F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
"D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
"C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
"A6 ", "A#6", "B6 ", "C7 " };

int j;

for (j = 0; j <= total-1; j++)
printf("%s ", pitch[array[j]]);
printf("\n");

} /* end Printpitch() function */
/***** MIC 3.6 *****/
/* Alterseq() function */
void Alterseq(array1,array2,expcont,total)
int array1[],array2[],expcont,total;
{
int j,oldintvl,intdir,newintvl;

array2[0] = array1[0];
for (j = 1; j <= total-1; j++)
{
if (array1[j] == array1[j-1])
{ array2[j] = array2[j-1];
continue;
}
oldintvl = array1[j] - array1[j-1];
if (oldintvl > 0)
intdir = 1;
else if (oldintvl < 0)
intdir = -1;
else intdir = 0;
newintvl = abs_val(oldintvl) + expcont;

if (newintvl < 1)
newintvl = 1;
array2[j] = array2[j-1] + intdir * newintvl;
}
} /* end Alterseq() function */
/*****
/* END OF ALTERSEQ.C */

```

### 程序运行实例

```

ALTERSEQ.EXE

a random-order pitch sequence --
C2 A#2 F3 C#2 E3 C2 E2 D#2 F#3 A#1

the sequence expanded by 2 semitones:
C2 C3 A3 D#2 G#3 D2 G#2 F2 A#3 C2

```

the sequence contracted by 2 semitones:  
C2 G#2 C#3 B1 C3 A#1 C2 B1 C3 F#1

## 函数组: Setflag.c MIC3.7

1. Setflag() MIC3.7
2. Zeromat() MIC2.27

### 功能

通过数组标记方法,生成一个无序的整数串,反复调用并返回一个无序的排列。

### 注释

这种无置换的取样是无用的,但是当被串行化的元素的数目相对地少一些时,那任务就非常好。它能生成一串在范围限制内的随机数,然后检查数组 cum[]地址指针的内容,以便通过这些随机数来决定是否一个标志(1)存在。如果地址为空,那么它将接收这个标志,并且把随机整数放入 set[]数组作为一系列的整数。如果这个地址已经有标记,这个数将被舍去并且另外生成一个。

为了生成许多串序列,就要反复地调用这个子程序,调用时,需要调用函数 MIC2.5 Zeromat(),在每次调用之前,把数组 cum[]的标志置位为 0。

在编程环境下,它可以更准确地激活两个混合子程序(MIC3.8 或者 MIC3.9)中的一个,以便生成一个元素组的无序排列。

### 编程提示

1. 写一个交互式的主程序,在这个程序中允许用户输入“低”和“高”的元素值,运行这个程序几次,设置“低”为 1,“高”为各种值(如 500),以得到一种有限使用试探算法。
2. 通过把函数输出作为一系列指针值放入一个元素表中,生或一串包括无连续标量的元素(12,14,23,45,47,61 等等)。

### 程序清单

```
/* SETFLAG.C (array-flag method random-order series generator) */
#include <stdio.h>
main()
{
    int set[12];
    int low = 0; /* first series element */
    int high = 11; /* last series element */
    int range = high - low + 1; /* series span */
    int total = 12; /*number of series elements */
    int j, seed = 21118;
```

```

void Setflag();

srand(seed);
Setflag(set,low,range,total);
printf("\na random-order 12-value series:\n");
for (j = 0;j < total;j++)
    printf("%d ",set[j]);
} /* end of main */
/*===== MIC 3.7 =====*/
/* Setflag() function */
void Setflag(array,bottom,span,total)
int array[],bottom,span,total;
{
    int j,u,cum[12];
    void Zeromat();

    Zeromat(cum,total);
    for (j = 0;j < total;j++)
    {
        do
            u = rand() % span + bottom;
        while (cum[u] == 1);
        cum[u] = 1;
        array[j] = u;
    }
} /* end of Setflag() function */
/*===== MIC 2.25 =====*/
/* Zeromat() function */
void Zeromat(x,range)
int x[],range;
{
    int j;
    for (j = 0;j < range;j++)
    {
        x[j] = 0;
    }
} /* end of Zeromat() function */
/*=====*/
/* END OF SETFLAG.C */

```

### 程序运行实例

SETFLAG.EXE

```

enter a random number generator seed: 9122
a random-order 12-value series:
3 5 10 11 7 1 8 9 6 4 0 2

```

### 函数组: Conshuff() MIC3.8

#### 功能

随机重组(混组)元素表数组的内容。

## 注释

这个算法通过第一次生成一个随机的指针序列来变序一个元素表，然后交换相应的数组地址内容。但是，变序之后，主要序列元素表不再对程序有用。(参见 MIC3.9 中的函数 Addshuf()它是一个无破坏性的数组地址随机化)。

## 编程提示

1. 写一个程序，它将要随机化(变序)任何合理长度变量值表，提供能获得任何四个元素的选项：音调、节奏和音量。
2. 加函数组:VECTORS.C是对上面函数，把输出数据有组织地写入音项向量。
3. 写一个程序，随机化一个数据表，它包括与六个八度微分音的音阶相应的数字，利用 4 元素输出生成一个四音结构，它的节奏也来源于一个变序宽度元素表。

## 程序清单

```
/* CONSHUFL.C (disruptive shuffle program - array contents
               reordered) */
#include <stdio.h>
main()
{
    int x[20]; /* stores the integer sequence to be shuffled */
    int total,j,seed = 441;
    void Conshuf1();

    srand(seed);
    total = 20;

    printf("\noriginal integer sequence:\n");
    for (j = 0;j < total;j++)
    {
        x[j] = j + 1;
        printf("%d ",x[j]);
    }
    printf("\nshuffled integer sequence:\n");
    Conshuf1(x,total);
    for (j = 0;j < total;j++)
        printf("%d ",x[j]);
    } /* end of main */

/*===== MIC 3.8 =====*/
/* Conshuf1() function */
void Conshuf1(array,total)
int array[],total;
{
    int j,u,s;
    for (j = 0;j < total;j++)
    {
        /* swap address contents */
        u = rand() % total;
        s = array[j];
        array[j] = array[u];
        array[u] = s;
    }
} /* end of Conshuf1() function */
/*=====*/
/* END OF CONSHUFL.C */
```



## 程序运行实例

CONSHUFL.EXE

```
original integer sequence:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
shuffled integer sequence:
13 6 14 9 7 2 3 11 12 8 5 1 15 17 20 10 18 19 16 4
```

## 函数: Addshufl() MIC3.9

### 功能

随机重组(变序)一个指针表到一个元素表中。

### 注释

这个函数与 MIC3.8 的区别在于:Conshufl()是无破坏性的,也就是说,数组的指针被变序了而不是数组的内容,因此脱离原表不影响将来对主程序的访问。

### 编程提示

1. 写一个程序,交替执行四个标准音调调节操作(后退、反向、后退一反向、调转)和主要音调顺序的无序排序。
2. 写一个程序,它建立在函数组: CURVES.C(MIC2.6,MIC2.7,MIC2.8),它们生成一个无序的变量数据表的无序排列,包括线性的、指数的、逻辑代数的和有范围的值的序列。

### 程序清单

```
/* ADDSHUFL.C (nondisruptive shuffle program: list of array
               address pointers reordered instead of array
               contents) */
#include <stdio.h>
main()
{
    int table[20], pointers[20], total, j, seed = -4491;
    void Addshufl();

    srand(seed);
    total = 20;

    printf("\noriginal array holding integer sequence:\n");
    for (j = 0; j < total; j++)
```

```

        {
            table[j] = j + 1;
            printf("%d ",table[j]);
            pointers[j] = j;
        }
        printf("%s%s","\\noriginal array accessed ",
            "by shuffled address pointers:\\n");
        Addshuf1(pointers,total);
        for (j = 0;j < total;j++)
            printf("%d ",table[pointers[j]]);
    } /* end of main */
    /***** MIC 3.9 *****/

/* Addshuf1() function */
void Addshuf1(ptrlist,total)
int ptrlist[],total;
{
    int j,u,s,rand();
    for (j = 0;j <= total-1;j++)
    {
        u = rand() % total;
        s = ptrlist[j];
        ptrlist[j] = ptrlist[u];
        ptrlist[u] = s;
    }
} /* end Addshuf1() function */
/***** */
/* END OF ADDSHUFL.C */

```

### 程序运行实例

ADDSHUFL.EXE

```

original array holding integer sequence:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
original array accessed by shuffled address pointers:
15 1 13 7 18 5 4 19 12 20 11 10 6 16 14 17 9 3 8 2

```

### 函数组: Rowforms.c

1. Printpitch MIC3.1
2. Conshuf MIC3.8
3. Rowretro MIC3.10
4. Rowinvrt MIC3.11
5. Rowtrapz MIC3.12

### 功能

生成普通的基本的 12 音列的形式: 初始的后退, 反向和调转。

### 注释

12 音调列形式是由一系列的子程序产生的, 它们没有寄存器具体化。但是, 八音度的分配来源于用很多方法产生的列。否则, 它们可能随机地被生成, 作为一个独立的、无联系的过程的结果。

## 编程提示

1. 写一个程序，生成无序的12音调音列，然后把音调频宽从一个浓缩的音阶(它表示列音调之间的间距大小)。

提示：因为最大的可能的1/2阶的间距是11，并且普通的音调范围是在1到6个八音曲调之间，一个易懂的方法把间距音“环绕”到这个八音曲调范围音阶上。

2. 扩展上面的程序，生成节奏宽度。节奏宽度是由无序的12音调列产生的[加函数 MIC3.21:Timpoint()]。
3. 用函数 MIC3.5:Displace()去生成曲调八音节的任务。

## 程序清单

```
/* ROWFORMS.C (produces commonly used 12-tone rowforms) */
#include <stdio.h>
main()
{
    int orig[12];      /* stores prime-order 12-tone series */
    int nextform[12];  /* stores related series forms */
    int transint;      /* interval of series transposition */
    int total=12;
    int j,seed = 5131;
    void Printpitch(), Conshufl(), Rowretro(),
        Rowinvrt(), Rowtrnpz();

    srand(seed);

    printf("\n(--loading integer array to be shuffled--)\n");
    for (j = 0; j < total; j++)
        orig[j] = j;
    printf("\na random-order 12-tone series:\n");
    Conshufl(orig,total);
    Printpitch(orig,total);
    printf("\nretrograde order:\n");
    Rowretro(orig,nextform,total);
    Printpitch(nextform,total);
    printf("\ninverted form:\n");
    Rowinvrt(orig,nextform,total);
    Printpitch(nextform,total);
    transint = 4;
    printf("\noriginal form transposed");
    printf(" by %d semitones:\n",transint);
    Rowtrnpz(orig,nextform,transint,total);
    Printpitch(nextform,total);
} /* end of main */
/*===== MIC 3.8 =====*/
/* Conshufl() function */
void Conshufl(array,total)
int array[],total;
{
    int j,u,s,rand();

    for (j = 0; j < total; j++)
    {
        u = rand() % total;
        s = array[j];
        array[j] = array[u];
        array[u] = s;
    }
}
```

```

    }
} /* end Conshuf1() function */
/***** MIC 3.10 *****/
/* Rowretro() function */
void Rowretro(array1,array2,total)
int array1[],array2[],total;
{
    int j,l;

    for (j = 0;j < total;j++)
    {
        l = (total - 1 - j) % total;
        array2[l] = array1[j];
    }
} /* end Rowretro() function */
/***** MIC 3.11 *****/
/* Rowinvrt() function */
void Rowinvrt(array1,array2,total)
int array1[],array2[],total;
{
    int j;

    for (j = 0;j < total;j++)
        array2[j] = (total-array1[j]) % total;
} /* end Rowinvrt() function */
/***** MIC 3.1 *****/
/* Rowtrnpz() function */
void Rowtrnpz(array1,array2,intval,total)
int array1[],array2[],intval,total;
{
    int j;

    for (j = 0;j < total;j++)
        array2[j] = (array1[j] + intval) % total;
} /* end Rowtrnpz() function */
/***** MIC 3.12 *****/
/* Printpitch() function */
void Printpitch(array,total)
int array[],total;
{
    int j;
    static char *pitch[] =
        {"C ","C# ","D ","D# ","E ","F ","F# ","
         "G ","G# ","A ","A# ","B "};

    printf("\n");
    for (j = 0;j < total;j++)
        printf("%s",pitch[array[j]]);
} /* end of Printpitch() function */
/***** MIC 3.13 *****/
/* END OF ROWFORMS.C */

```

## 程序运行实例

ROWFORMS.EXE

--loading integer array to be shuffled--

a random-order 12-tone series:

```

C A B D D# C# E G A# F F# G#
retrograde order:

G# F# F A# G E C# D# D B A C
inverted form:

C D# C# A# A B G# F D G F# E
original form transposed by 4 semitones:

E C# D# F# G F G# B D A A# C

```

## 函数组: Rowsquar.c

1. Setflag MIC3.7
2. Rowmat MIC3.13

## 功能

生成并显示一个方阵阵列，它包含了一个无序的 12 音调列形式的转调。

## 注释

使用排序算法的作曲家有时会喜欢用打印出一个特定曲谱的 48 转调的压缩版本来目测一排合成曲调的可能性。但是本系列矩阵将帮助作曲家很轻松地实现这一点。

将行表达为一个矩阵也已经被作曲家们运用到无序数列的应用中，尽管传统的 12 曲调作曲方式用一种硬件的、单一的样式来解释矩阵，但是作曲家完全可以将矩阵看作音项文件，仅凭借它实现作曲各项操作，从而赋以矩阵一种灵活、生动的解释。例如，12 音调中的每一个音可以不作为一个单音而看作一组乐音混合而成的复音；又例如，可以通过将一组旋律以不同的矩阵格式显示给演奏者，从而使演奏都能自行决定读谱的方向、方式，达到各种演奏效果。

## 编程提示

1. 将驱动程序变为交互式，允许用户输入音无。
2. 修改上述程序，先将矩阵存入一二维维数中，再加入语句生成一条在矩阵中含四个方向的随机遍历路径。
3. 设计其他方法寻找矩阵遍历路径。

## 程序清单

```

/* ROWSQUAR.C (produces square rowforms matrix) */
#include <stdio.h>
main()
{
    int orig[12]; /* stores original 12-tone series */
    int invers[12]; /* stores inverted 12-tone series */
    int seed = 665;
    void Rowmat(); /* configures and prints the matrix */
    void Setflag();

    srand(seed);
    printf("\nhere is a random-order, 12-tone series matrix:\n");

    printf("(of all available O,RO,I, and RI forms)\n");
    Setflag(orig,invers); /* call series generator */
    Rowmat(orig,invers); /* print the matrix */
} /* end of main */
/*===== MIC 3.7 =====*/
/* Setflag() function -- modified to program specs */
void Setflag(array1,array2)
int array1[12],array2[12];
{
    int cum[12],u,j,trans;
    for (j = 0;j <= 11;j++)
    {
        do u = rand() % 12;
        while (cum[u] == 1);
        cum[u]=1;
        if (j == 0)
            trans = u;
        if (u - trans < 0)
            array1[j] = u - trans + 12;
        else array1[j] = u - trans;
        array2[j] = (12-array1[j]) % 12;
    }
} /* end Setflag() function */
/*===== MIC 3.13 =====*/
/* Rowmat function */
void Rowmat(array1,array2)
int array1[],array2[];
{
    int j,k,nextnote,intval;
    static char *pitch[] =
        {"C ","C# ","D ","D# ","E ","F ","F# ","
         "G ","G# ","A ","A# ","B "};

    printf("\n");
    for (j = 0;j <= 11;j++)
    {
        intval = array2[j];
        for (k = 0;k <= 11;k++)
        {
            nextnote = (array1[k] + intval) % 12;
            printf("%s",pitch[nextnote]);
        }
        printf("\n");
    }
} /* end Rowmat() function */
/*=====*/
/* END OF ROWSQUAR.C */

```

## 程序运行实例

ROWSQUAR.EXE

here is a random-order, 12-tone series matrix:  
(of all available C,RO,I, and RI forms)

```
C B G C# A# F D# E A D F# G#
C# C G# D B F# E F A# D# G A
F E C F# D# A# G# A D G B C#
B A# F# C A E D D# G# C# F G
D C# A D# C G F F# B E G# A#
G F# D G# F C A# B E A C# D#
A G# E A# G D C C# F# B D# F
G# G D# A F# C# B C F A# D E
D# D A# E C# G# F# G C F A B
A# A F B G# D# C# D G C E F#
F# F C# G E B A A# D# G# C D
E D# B F D A G G# C# F# A# C
```

### 函数组: Modops.c

1. PrintPitch() MIC3.1
2. Conshuffl MIC3.8
3. M5setM7 MIC3.14

### 功能

通过乘法运算生成相互关联的 12 音调音列的变调形式。

### 注释

虽然 M5 和 M7 可算作能够生成一系列音的变调形式。但从广义上讲,他们实际上是生成了 12 级不同音阶组成的音列的有序排列(正如同音程变换操作实际上是生成音列的以一定方式相关的有序排列)。

仅仅排列某音列组的操作与对该音列进行变调操作的区别在于:排列某间列是根据参考音列集合生成新音列组。新音列组保留参考音列的基本特性不变。乘法运算的相关性表现在,其实质是有次序地将音列中的音对调位置。而且,M5 操作代表将一种音阶序列映射为第 4 音给功能图;M6 操作代表将一种音阶序列映射为第 5 音阶功能图。

### 编程提示

1. 寻找其他模数为 12 的乘法因子。不同的乘法因子会对音列统一性产生何影响?
2. 应用不同的乘法因子对各种不同长度的音列进行试运算。

### 程序清单

```

/* MODOPS.C (produces related transformations of a 12-tone row)*/
#include <stdio.h>
main()
{
    int orig[12];      /* stores original series */
    int perm[12];      /* stores series permutation */
    int op;            /* toggles M5 or M7 operation */
    int total,j,seed = -9112;
    void Conshuf1(), M5setM7(), Printpitch();

    srand(seed);
    total = 12;
    printf("\na random-order 12-tone series:\n");
    for (j = 0;j < total;j++)
        orig[j] = j;
    Conshuf1(orig,total);
    Printpitch(orig,total);
    printf("\nthe permutation resulting from M5 operation:\n");
    op = 5;
    M5setM7(orig,perm,op,total);
    Printpitch(perm,total);
    printf("\nthe permutation resulting from M7 operation:\n");
    op = 7;
    M5setM7(orig,perm,op,total);
    Printpitch(perm,total);
} /* end of main */
/*===== MIC 3.8 =====*/
/* Conshuf1() function */
void Conshuf1(array,total)
int array[],total;
{
    int j,u,s;

    for (j = 0;j < total;j++)
    {
        u = rand() % total;
        s = array[j];
        array[j] = array[u];
        array[u] = s;
    }
} /* end Conshuf1() function */
/*===== MIC 3.14 =====*/
/* M5setM7() function (outputs a permutation of an input
   series by the "M5" or "M7" operation.) */
void M5setM7(array1,array2,op,total)
int array1[],array2[],op,total;
{
    int j;

    for (j = 0;j < total;j++)
        array2[j] = (array1[j] * op) % 12;
} /* and M5setM7() function */
/*===== MIC 3.1 =====*/
/* Printpitch() function */
void Printpitch(array,total)
int array[],total;
{
    int j;
    static char *pitch[] =
        {"C ", "C# ", "D ", "D# ", "E ", "F ", "F# ",
         "G ", "G# ", "A ", "A# ", "B " };
    printf("\n");
    for (j = 0;j < total;j++)

```



```

        printf("%s",pitch[array[i]]);
    } /* end of Printpitch() function */
    /*-----*/
    /* END OF MODOPS.C */

```

## 程序运行实例

MODOPS.EXE

a random-order 12-tone series:

E G# A D# A# B F# C D F C# G

the permutation resulting from M5 operation:

G# E A D# D G F# C A# C# F B

the permutation resulting from M7 operation:

E G# D# A A# F F# C D B G C#

## 函数组: Pstnperm.c

1. Setflag() MIC3.7
2. Pstnperm() MIC3.15

## 功能

通过将集合中的数值元素变换为音符数据生成一组主序 12 音调的音列的排列。

## 注释

本算法是一系列包含生成音列变调形式的排列算法中的一种, 而且还有一特殊用法: 当升序(或降序)音阶是主序音列时, 函数返回其逆序音列。

这种系统化的集合组排列算法可用于任何长度的数列。而且, 如果你对每一排列的集合的四种格式的所有变调对进行变换操作, 便可生成无限多种新集合组。

## 编程提示

1. 修改驱动程序, 允许用户输入不同长度的数列。加入必要的语句生成原音列的逆序、逆向和逆序-逆向音列, 以及各种变调形式, 并加入一项功能, 允许用户将函数 Pstnperm() 应用于所有音列及其变调形式。
2. 编写一交互式程序, 应用程序 MIC.15 将输出的音列排列的各数据元素作为各种音乐参数的指针。

## 程序清单

```

/* PSTNPERM.C (generates a series permutation by swapping
   series position and pitch class number) */
#include <stdio.h>
main()
{
    int set[12];    /*stores original series */
    int posit[12]; /*stores permutation */
    int low = 0;
    int high = 11;
    int range = high - low + 1;
    int total = 12, seed = 9182;
    void Setflag(), Pstnperm(), Printpitch();

    srand(seed);
    Setflag(set, low, range, total);
    printf("\na random-order 12-value series:\n");

    Printpitch(set, total);
    printf("\npermutation resulting from ");
    printf("swap of PC number and position\n");
    Pstnperm(set, posit, total);
    Printpitch(posit, total);
} /* end of main */

/*----- MIC 3.7 -----*/
/* Setflag() function */
void Setflag(array, bottom, span, total)
int array[], bottom, span, total;
{
    int j, u, cum[12], rand();

    for (j = 0; j < total; j++)
    {
        do    u = rand() % span + bottom;
        while (cum[u] == 1);
        cum[u] = 1;
        array[j] = u;
    }
} /* end Setflag() function */

/*----- MIC 3.15 -----*/
/* Pstnperm() function (generates a permutation of a
   prime order series by swapping pitch class numbers
   with position-in-set numbers. If the series is
   the chromatic scale, then, of course, no alteration
   of the set will take place - the retrograde will
   be generated.) */
void Pstnperm(array1, array2, total)
int array1[], array2[], total;
{
    int j;

    for (j = 0; j < total; j++)
        array2[array1[j]] = j;
} /* end Pstnperm() function */

/*----- MIC 3.1 -----*/
/* Printpitch() function (adapted to print pitches
   as integers) */
void Printpitch (array, total)
int array[], total;
{
    int j;

```

```

        printf("%d\n");
        for (j = 0; j < total; j++)
            printf("%d ", array[j]);

    } /* end Printpitch() function */
    /*-----*/
    /* END OF PSTNPERM.C */

```

## 程序运行实例

PSTNPERM.EXE

a random-order 12-value series:

7 2 4 8 9 10 1 0 6 5 11 3

permutation resulting from swap of PC number and position

7 6 1 11 2 9 8 0 3 4 5 10

## 函数组: SampleSet() MIC3.16

### 功能

通过将原数据组按一定间隔进行循环, 穷举取样得取该数据组的排列(例如, 从一元素  $n$  开始, 将其后第  $n$  个元素取出移至目标数列中, 循环重复此过程直至所有的元素都被移至目标数列中。

### 注释

本算法是根据古代算术问题 Josephus 问题演变而来, 具体步骤如下:

1. 将一组人排成一个圆圈;
2. 派一个给圆圈沿圆圈走动、查看;
3. 每隔几个人将 1 个拉出圆圈;
4. 重复 2、3 步骤, 直至所有的人都被拉出圆圈。

当然, 还有许多其他穷举法取样的算法, 本函数仅通过两个“跳动”数组来存放样本组的暂态版本实现算法。在取样过程中, 目标序列的各元素将从原样本组的副本中取出并显示在屏幕上, 而原样本组保持不变。

如果本函数被一段程序调用, 而该程序希望实现将样本组排列并实时存储, 则必须引入另一数组存放目标序列以便将其显示在屏幕上。

### 编程提示

1. 编写一程序应用 MIC3.16 中的函数生成一组用户输入数据的排列并存储(并将生成的目标序列作为其他音乐参数的地址指针)。
2. 编写一程序, 从一组输入数列的排列中选取样本, 从而将根据原数据组通过取样生成新的数列的过程系统化。

## 程序清单

```

/* SAMPLSET.C (generates and prints numeric series
               permutations by set-sampling at a selected
               series-position interval) */
#include <stdio.h>
main()
{
    int orig[100];

    int currsetl = 12; /*current set length */
    int cycle;        /* permutation loop index */
    int target;        /* current targeted set sample */
    int factor = 2;    /* set sampling interval */
    int permut = 5;    /* total number of permutations */
    int j;             /* loop index,pointer to original set */
    void Samplset();

    printf("a sequential set:\n");
    for (j = 1;j <= currsetl;j++)
    {
        orig[j] = j;
        printf("%d ",orig[j]);
    }
    for (cycle = 1;cycle <= permut;cycle++)
    {
        printf("\npermutation %d\n",cycle);
        target = 1;
        printf("start position = %d\n",target);
        printf("target factor = %d\n",factor);
        Samplset(orig,target,factor,currsetl);
        factor = factor + 2;
    }
} /* end of main */
/*===== MIC 3.16 =====*/
/* Samplset() function */
void Samplset(arrayl,target,factor,currsetl)
int arrayl[],target,factor,currsetl;
{
    int a[100],b[100],acnt,bcnt,j;

    for (j = 1;j <= currsetl;j++)
        a[j] = arrayl[j];
    loop: bcnt = 0;
    for (j = 1;j <= currsetl;j++)
    {
        if (j == target)
        {
            printf("%d ",a[j]);
            target = target + factor;
        }
        else
        {
            bcnt = bcnt + 1;
            b[bcnt] = a[j];
        }
    }
    if (bcnt < 1)
    {
        printf("\n");
        return;
    }
    target = target - currsetl;
    acnt = 0;
}

```

```

if (target % bcnt != 0)
    target = target % bcnt;
else target = bcnt;
for (j = 1; j <= bcnt; j++)
    if (j == target)
    {
        printf("%d ", b[j]);

        target = target + factor;
    }
    else
    {
        acnt = acnt + 1;
        a[acnt] = b[j];
    }
if (acnt < 1)
{
    printf("\n");
    return;
}
currset1 = acnt;
target = target - bcnt;
if (target % currset1 != 0)
    target = target % currset1;
else target = currset1;
goto loop;
} /* end of Samplset() function */
/*-----*/
/* END OF SAMPLSET.C */

```

### 程序运行实例

#### SAMPLSET.EXE

```

a sequential set:
1 2 3 4 5 6 7 8 9 10 11 12
permutation 1
start position = 1
target factor = 2
1 3 5 7 9 11 2 6 10 4 12 8

permutation 2
start position = 1
target factor = 4
1 5 9 2 7 12 8 4 3 6 11 10

permutation 3
start position = 1
target factor = 6
1 7 2 9 5 3 12 4 8 6 11 10

permutation 4
start position = 1
target factor = 8
1 9 6 4 3 5 8 12 11 7 2 10

```

```

permutation 5
start position = 1
target factor = 10
1 11 20 12 3 6 2 9 5 7 4 8

```

## 函数组: Rotate.c

- |                            |         |
|----------------------------|---------|
| 1. Conshuff()              | MIC3.8  |
| 2. Setrotat()              | MIC3.17 |
| 3. Zerotrnz()              | MIC3.18 |
| 4. Segrotat()              | MIC3.19 |
| 5. GreatestCommonDivisor() | MIC2.13 |
| 6. Matprint()              | MIC3.20 |

## 功能

对一列含有 12 个元素的数列进行旋转和转置运算。

## 注释

集合元素的旋转是普通 12 音数列变换运算方式之一。在这组函数中，以整数而不是音符形式输出数据，以便于更容易将输出作为其他音乐参数的指针(各集合元素可作为其他音乐参数的地址指针)。数 0 到 11 代表 12 个半音阶的各音阶级，这种表示方式有利于(同 MIC3.14 函数中的)其后的取模运算。但若编程者希望用数 1-12 代表各音阶，也可通过稍加修改函数来实现。

函数 Setrotat()通过将一组数目给定、连续的集合元素从序列首移至序列尾部来实现其功能。它也可通过计算程序对元素列进行变换运算的次数来防止重复并使程序在该点停止运行。旋转运算后的结果存于数组 mat1[]中以各在函数 Zerotrnz()中使用。

函数 Segrotat()只能用于旋转分块的集合元素，各块所含元素数目必须相等(例如，2 组由每组 6 个音符组成的块，3 组由每组 4 个组成的块等)。在经过数次某种变换运算后将结果存入数组 mat3[]中以便打印。

## 编程提示

编写一交互式程序利用函数 ROTATE.C 实现，提示用户输入原集合元素数列、进行集合元素旋转运算，然后调用 MIC3.15: Pstnperm()函数以对下一集合进行旋转运算。

## 程序清单

```

/* ROTATE.C (set of functions to perform serial rotation
and transposition operations)*/
#include <stdio.h>

```

```

main()
{
    int orig[12];      /* stores original series */
    int mat1[12][12]; /* stores unpartitioned set rotation */
    int mat2[12][12]; /* stores '0' start-point transposition */
    int mat3[12][12]; /* stores partitioned set rotation */
    int settotal = 12; /* number of series elements */
    int rotegroup; /* number of contiguous elements to rotate */
    int numsegs;    /* symmetrical set partitions */
    int segsegs;    /* number of elements within each partition */
    int cycles,j,k; /* loop indices */

    int seed = 11;
    int GreatestCommonDivisor(),Segrotat();
    void Matprint(),Conshuf1(),Setrotat(),ZeroTrnp();

    srand(seed);
    printf("a random-order 12-value series:\n");
    for (j = 0;j <= settotal-1;j++) /*load set elements*/
        orig[j] = j;
    Conshuf1(orig,settotal); /*call shuffle function*/
    for (j = 0;j <= settotal-1;j++)
        printf("%d ",orig[j]);
    rotegroup = 4;
    cycles = GreatestCommonDivisor(settotal,rotegroup);
    printf("\ntotal number of unique cycles = %d\n",cycles);
    printf("here is the rotation matrix:\n");
    Setrotat(orig,mat1,rotegroup,settotal,cycles);
    Matprint(mat1,cycles,settotal);
    printf("\n'0' start point transposition:\n");
    ZeroTrnp(mat1,mat2,cycles,settotal);
    Matprint(mat2,cycles,settotal);
    numsegs = 4;
    printf("\npartitioned series,rotation within segment --\n");
    printf("-- divided into %d equal segments.\n",numsegs);
    cycles = Segrotat(orig,mat3,numsegs,settotal);
    Matprint(mat3,cycles,settotal);
} /* end of main */

/*===== MIC 3.20 =====*/
/* Matprint() function*/
void Matprint(matrix,cycles,settotal)
int matrix[][12],cycles,settotal;
{
    int l,k;
    for (l = 0;l <= cycles-1;l++)
    {
        for (k = 0;k <= settotal-1;k++)
            printf("%d ",matrix[l][k]);
        printf("\n");
    }
} /* end Matprint() function */

/*===== MIC 3.8 =====*/
/* Conshuf1() function */
void Conshuf1(array1,total)
int array1[],total;
{
    int m,u,s,rand();

    for (m = 0;m <= total-1;m++)
    {
        u = rand() % total;
        s = array1[m];
        array1[m] = array1[u];
    }
}

```

```

        array1[u] = s;
    }
} /* end Conshuf1() function */
/*----- MIC 2.13 -----*/
/* GreatestCommonDivisor() function */
int GreatestCommonDivisor(a,b)
int a,b;
{
    int temp,c,d,series,cycles;

    series = a;
    if (a > b) /*<<< swap function*/
    {
        temp = a;
        a = b;
        b = temp;
    }
    while(a > 0)
    {
        c = b/a;
        d = b - a * c;
        b = a;
        a = d;
    }
    cycles = series/b;
    return(cycles);
} /* end GreatestCommonDivisor() function */
/*----- MIC 3.17 -----*/
/* Setrotat() function */
void Setrotat(array1,array2,rotegroup,settotal,cycles)
int array1[],array2[][12],rotegroup,settotal;
{
    int l,k,u,s,t;

    rotegroup = rotegroup -1;
    u = 0;
    s = 0;
    for (k = rotegroup;k <= settotal-1+rotegroup;k++)
    {
        for (l = 1;l <= settotal;l++)
        {
            t = (k + l + u) % settotal;
            array2[s][l-1] = array1[t];
        }
        u = u + rotegroup;
        s = s + 1;
        if (s >= cycles)
            break;
    }
} /* end Setrotat() function */
/*----- MIC 3.18 -----*/
/* Zerotrnp() function */
void Zerotrnp(matrix1,matrix2,cycles,settotal)
int matrix1[][12],matrix2[][12],cycles,settotal;
{
    int intval,l,k,s;

    for (k = 0;k <= cycles-1;k++)
    {
        intval = matrix1[k][0];
        for (l = 0;l <= settotal-1;l++)

```



```

        {
            s = matrix1[k][1] - interval;
            if (s < 0)
                s = s + 12;
            matrix2[k][1] = s;
        }
    } /* end Zerotrnp() function */
    /*===== MIC 3.19 =====*/
    /* Segrotat() function */
    int Segrotat(array1, matrix, numsegs, settotal)
    int array1[], matrix[][12], numsegs, settotal;
    {
        int k, l, m, s, t, segmens;
        s = 0;

        segmens = settotal / numsegs;
        for (k = 1; k <= segmens; k++)
        {
            t = 0;
            for (l = 0; l <= settotal-1; l += segmens)
            {
                for (m = s + 1; m <= segmens + s; m++)
                {
                    matrix[s][t] = array1[m % segmens + 1];
                    t += 1;
                }
                s += 1;
            }
        }
        return(segmens);
    } /* end Segrotat() function */
    /*=====*/
    /* END OF ROTATE.C */

```

### 程序运行实例

#### ROTATE.EXE

```

a random-order 12-value series:
11 4 2 6 5 0 8 9 1 7 3 10
total number of unique cycles = 3
here is the rotation matrix:
5 0 8 9 1 7 3 10 11 4 2 6
1 7 3 10 11 4 2 6 5 0 8 9
11 4 2 6 5 0 8 9 1 7 3 10

'0' start point transposition:
0 7 3 4 8 2 10 5 6 11 9 1
0 6 2 9 10 3 1 5 4 11 7 8
0 5 3 7 6 1 9 10 2 8 4 11

```

```

partitioned series, rotation within segment -
-- divided into 4 equal segments.
4 2 11 5 0 6 9 1 8 3 10 7
2 11 4 0 6 5 1 8 9 10 7 3
11 4 2 6 5 0 8 9 1 7 3 10

```

## 函数组: Timpoint.c

1. Printpitch() MIC3.1
2. Setflag() MIC3.7
3. Timpoint() MIC3.21
4. Zeromat() MIC3.25

## 功能

应用节拍计时系统将一系列含 12 种音调的音符转化为一列音长值。

## 注释

有时作曲者希望将一系列含 12 种音调的音之间的音程关系转化为节拍值的形式。节拍计时系统就是被用来将各音符相隔的音程值转化为奏音的节拍。因为这一计时方法主要用到各音之间的时间间隔, 所以必须先用发音方式参数来给定音与音之间发声与休止的时间比。通常, 我们应用一模数来将数列计算限定在一个八度音程的音域内(0 到 11, 模数为 12), 同样地, 还可设置一时间模数与上述模数配合使用。

本函数先以  $1/2$  步长计算一组连续集合元素之间的距离, 然后返回一系列音长值, 其发音方式用一模数控制, 决定每小节中的时间脉冲数。

## 编程提示

1. 编写一程序, 允许用户输入一系列含 12 种音调的音列, 然后通过旋转取样(可用 MIC3.16:Sampleset()函数)对原音列进行变换运算, 将变换后的音列转化为节拍。
2. 修改上述程序, 应用函数 MIC3.18、MIC3.19 以及 MIC3.20 实现旋转、换位功能, 并允许用户将经过变换的输入数列作为节拍参数、音量参数以及发音方式参数的指针。
3. 编写一程序自动生成一系列无序含 12 音调的音符进行变换运算, 然后将运算结果作为四种音乐参数的指针并存盘。

## 程序清单

```

/* TIMPOINT.C (produces note duration values using the
   serial timepoint system) */
#include <stdio.h>
main()

```

```

{
int orig[12]; /* prime 12-element series */
int prpitch = 1; /* flag to print notenames or numbers */
int median; /* basic metrical pulse unit */
int modulus; /* transform modulus */
int seed = -9121;
int j; /* loop index */
void Setflag(), Timpoint(), Printpitch(), Zeromat();

srand(seed);
printf("\nEnter modulus (2,3,4,5, or 12):\n");
scanf("%d",&modulus);
printf("Enter median (2,4,8, or 16):\n");
scanf("%d",&median);
printf("\na random-order 12-tone pitch series,");
printf(" mod %d:\n",modulus);
Setflag(orig,modulus);
Printpitch(orig,prpitch);
printf("\nthe pitch series expressed as numbers,");
printf(" mod %d:\n",modulus);
prpitch = 0;
Printpitch(orig,prpitch);
printf("%s%s", "\ndurations representing the ",
"distance between time points,\n");
printf("%s%s", "using a 1/%d note pulse base",
" and a measure \n",median);
printf("of %d -1/%d notes.\n",modulus,median);
Timpoint(orig,modulus,median);
} /* end of main */
/*===== MIC 1.7 =====*/
/* Setflag() function */
void Setflag(array,modulus)
int array[],modulus;
{
int j,u,cum[12];
void Zeromat();

Zeromat(cum);
for (j = 0;j < 12;j++)
{
do u = rand() % 12;
while (cum[u] == 1);
cum[u] = 1;
array[j] = u % modulus;
}
} /* end of Setflag() function */
/*===== MIC 3.21 =====*/
/* Timpoint() function (produces rhythm duration values
using the serial 'timepoint system'.*/
void Timpoint(set,modulus,median)
int set[],modulus,median;
{
int tally,dur,j;
tally = 0;

for (j = 0;j < 11;j++)
{
if (set[j+1] <= set[j])
dur = modulus - set[j] + set[j+1];
else dur = set[j+1] - set[j];
tally = tally + dur;
}
}

```

```

        printf("%d/%d ",dur,median);
    }
    dur = modulus - set[j];
    printf("%d/%d ",dur,median);
} /* end of Timpoint() function */
/*----- MIC 3.1 -----*/
/* Printpitch() function (modified to program specs) */
void Printpitch(array,p)
int array[],p;
{
    int j;
    static char *pitch[] =
        {"C ","C# ","D ","D# ","E ","F ","F# ","
         "G ","G# ","A ","A# ","B "};

    printf("\n");
    for (j = 0;j < 12;j++)
        if (p)
            printf("%s",pitch[array[j]]);
            else printf("%d ",array[j]);
} /* end of Printpitch() function */
/*----- MIC 2.25 -----*/
/* Zeromat function */
void Zeromat(array)
int array[];
{
    int j;

    for (j = 0;j < 12;j++)
        array[j] = 0;
} /* end Zeromat function */
/*-----*/
/* END OF TIMPOINT.C */

```

## 程序运行实例

### TIMPOINT.EXE

```

enter modulus (2,3,4,6, or 12): 4
enter median (2,4,8, or 16): 4

a random-order 12-tone pitch series, mod 4:

D# D# D D C# C C# C C# C D D#
the pitch series expressed as numbers, mod 4:

3 3 2 2 1 0 1 0 1 0 2 3
durations representing the distance between time points,
using a 1/4 note pulse base and a measure
of 4 -1/4 notes.
4/4 3/4 4/4 3/4 3/4 1/4 3/4 1/4 3/4 2/4 1/4 1/4

```

## 函数组: Alintseq.c

1. Alintseq() MIC3.22
2. Consbuff() MIC3.8

## 功能

将一组含有 11 个音程值的数列(音程大小范围为从 1 到 11)转换成各个不同音阶的音列。

## 注释

虽然程序生成的无序音程值序列中的音程值各不相同, 函数运行后将该序列转换成的一系列音符都有可能同音重复(若要生成全域范围内的音符列需要用更复杂的算法), 然而, 这种现象可被有意识地加以利用, 并通过增、减音对加强某种音调的效果。

函数根据一系列音程值生成的一系列音符可以都升、降八度而同时保持其内部结构关系不变。要实现此操作, 只需在当某音升降八度后引起音列方向的改变时, 将相应音程值换算为其八度音程余数即可(例如, 小调升 3 换算为大调降 6)。

## 编程提示

1. 编写一程序用于测某一随机生成的音程值序列的音符缺音以及单音重复度, 并将缺省音符及其总数目和重复音符及其总数目列表显示。
2. 扩展1中所述程序以加强其智能的功能, 例如: 加入条件语句, 以便在根据一存有互相关联的乐音序列的文件的布局规划来选择音列时设置一连续因子。
3. 将步骤2中生成的文件作为音高参数、节拍参数、发音方式参数以及音量参数的指针。
4. 修改函数组ALINTSEQ.C加入一函数实现将音符音高升或降八度的功能, 并且此函数在因将某音升降八度引起的乐句方向变化时, 能自动将对应音程换算为其八度音程余数。

## 程序清单

```
/* ALINTSEQ.C (maps a set of 11 unique intervals (in
               semitones, size 1-11) to the pitch class
               numbers of the chromatic scale.) */
#include <stdio.h>
main()
{
    int intset[12]; /* array of unique interval-sizes */
    int pclass[12]; /* pitch class sequence array */
    int total = 10; /* number of sequences to generate */
    int intervals = 11; /* number of intervals between notes*/
    int j,k;        /* loop indices */
    int seed = 445;
    void Alintseq(), Conshuf1();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        for (k = 0; k < intervals; k++)
            intset[k] = k + 1;
    }
}
```

```

        Conshuf1(intset,intervals);
        printf("\n\nrandom-order interval-class series #1d:\n",j);
        for (k = 0;k < 11;k++)
            printf("%d ",intset[k]);
        Alintseq(pclass,intset);
        printf("\n\n-- mapped to pitchclass numbers --\n");
        for (k = 0;k < 12;k++)
            printf("%d ",pclass[k]);
    }
} /* end of main */
/***** MIC 3.22 *****/
/* Alintseq() function (generates
    all-interval pitch sequences) */
void Alintseq(pclass,intset)
int pclass[],intset[];
{
    int j;

    pclass[0] = 0;
    for (j = 1;j < 12;j++)
        pclass[j] = (pclass[j-1] + intset[j-1]) % 12;
} /* end of Alintseq() function */
/***** MIC 3.8 *****/
/* Conshuf1() function */
void Conshuf1(intset,intervals)
int intset[],intervals;
{
    int j,u,s,rand();

    for (j = 0;j < intervals;j++)
    {
        u = rand() % 11;
        s = intset[j];
        intset[j] = intset[u];
        intset[u] = s;
    }
} /* end of Conshuf1() function */
/***** MIC 3.8 *****/
/* END OF ALINTSEQ.C */

```

## 程序运行实例

### ALINTSEQ.EXE

```

random-order interval-class series #1:
1 11 6 5 10 4 7 2 8 9 3
-- mapped to pitchclass numbers --
0 1 0 6 11 9 1 8 10 6 3 6
random-order interval-class series #2:
5 6 10 1 3 11 8 9 2 4 7
-- mapped to pitchclass numbers --
0 5 11 9 10 1 0 8 5 7 11 6
random-order interval-class series #3:
3 4 6 1 10 5 3 7 2 9 11
-- mapped to pitchclass numbers --
0 8 0 6 7 5 10 1 8 10 7 6
random-order interval-class series #4:
5 8 6 3 9 7 10 11 4 1 2

```

```

-- mapped to pitchclass numbers --
0 5 1 7 10 7 2 0 11 3 4 6
random-order interval-class series #5:
2 1 11 5 8 6 3 9 10 4 7
-- mapped to pitchclass numbers --
0 2 3 2 7 3 9 0 9 7 11 6
random-order interval-class series #6:
6 4 11 2 10 3 7 9 8 1 5
-- mapped to pitchclass numbers --
0 6 10 9 11 9 0 7 4 0 1 6
random-order interval-class series #7:
11 7 2 4 8 3 9 5 10 1 6
-- mapped to pitchclass numbers --
0 11 6 8 0 8 11 8 1 11 0 6
random-order interval-class series #8:
1 6 4 10 9 11 8 2 3 7 5
-- mapped to pitchclass numbers --
0 1 7 11 9 6 5 1 3 6 1 6
random-order interval-class series #9:
2 4 9 6 3 10 8 5 11 7 1
-- mapped to pitchclass numbers --
0 2 6 3 9 0 10 6 11 10 5 6
random-order interval-class series #10:
2 8 5 3 11 9 7 1 6 10 4
-- mapped to pitchclass numbers --
0 2 10 3 6 5 2 9 10 4 2 6

```

## 函数组: Alintset.c

1. Alintset() MIC3.23
2. Zeromat() MIC2.25

## 功能

生成一全音程含 12 音调的音列。(不允许音阶重复。)

## 注释

函数 Alintset() 的算法主要借助于计算机可在短时间内对大量数据进行取样/测试以及删除(保留)功能。虽然其算法是一种“蛮力计算法”，但其过程大致与作曲者浏览音符并构思出全音程音列的过程相似。计算机随机生成一系列音符并测试该列音的音阶以及音程来判定这一乐句的特征。如果在测试过程中遇到重复音，则删除重复音后继续运行，这种算法将会遇到的困难在于当越来越多的音加入到音列中时，将会出现绝大多数新加入的音都是重复音并被冗余检测过程删掉的情况，这样实际上相当于很难再加入新的音。事实上，许多实验性乐句在谱到末尾时，不得不放弃。因为上述情况的发生导致无法选音谱定这一乐句。不过这种算法在 16 位微机上应用十分快捷。

## 编程提示

1. 改写程序以应用函数 MIC3.1: Printpitch().
2. 编写一段程序以实现将函数返回的全音程音列中的音程值进行旋转运算。注意音程旋转运算对重复音产生的效果并寻求调整音列中间的各种参数的方法。
3. 修改函数组ALINTSET.C, 加入一函数的实现将音符音高升、降八度, 并当此函数在因某音升降八度而引起乐句方向改变时, 能自动将对应音程值换算为其八度音程余数。

## 程序清单

```
/* ALINTSET.C (all-interval 12-tone series generator) */
#include <stdio.h>
main()
{
    int pclass[12]; /* pitch class array */
    int j,k;        /* loop indices */
    int seed = -11519;
    void Alintset();

    srand(seed);
    for (j = 0; j < 20; j++)
    {
        Alintset(pclass);
        printf("\n");
        for (k = 0; k < 12; k++)
            printf("%d ", pclass[k]);
        printf("\n");
    } /* end of main */
    /*----- MIC 3.23 -----*/
    /* Alintset() function ( generates all-interval 12-note
       series by the sample-test-discard/keep method. */
    void Alintset(pclass)
    int pclass[];
    {
        int pcum[12], icum[12], temp[12];
        int j,l,m,u,intsize;
        void Zeromat();

        loop: Zeromat(pclass);
        Zeromat(icum);
        Zeromat(pcum);
        pclass[0] = 0;
        pclass[11] = 6;
        for (j = 1; j < 11; j++)
        {
            do
            {
                u = (rand() % 11)+1;
                while (u == 6 || pcum[u] == 1);
                temp[j] = u;
                pcum[u] = 1;
            }
            for (l = 1; l < 12; l++)
            {
                for (m = 1; m < 11; m++)
```



```

{
    if (temp[m] > 0)
        intsize = temp[m] - pclass[l-1];
    else continue;
    if (intsize < 0)
        intsize = 12 + intsize;
    if (icum[intsize] < 1)
    {
        pclass[l] = temp[m];
        temp[m] = 0;
        icum[intsize] = 1;
        break;
    }
} /* end inner loop */
if (m > 10 && pclass[l] == 0)
    goto loop;
} /* end outer loop */
} /* end of Alintset() function */
/*----- NIC 2.25 -----*/
/* Zeromat function */
void Zeromat(array)
int array[];
{
    int j;

    for (j = 0; j < 12; j++)
        array[j] = 0;
} /* end of Zeromat function */
/*-----*/
/* END OF ALINTSET.C */

```

### 程序运行实例

ALINTSET.EXE

```

0 1 5 2 8 10 3 11 9 4 7 6
0 11 9 10 3 5 8 4 1 7 2 6
0 10 5 4 7 3 9 11 8 1 2 6
0 10 7 8 4 3 5 11 2 9 1 6
0 1 8 7 9 5 10 4 2 11 3 6
0 8 1 11 3 9 10 7 2 5 4 6
0 9 11 4 7 3 10 8 2 1 5 6
0 3 7 2 10 8 9 11 5 4 1 6
0 3 4 10 8 5 9 11 7 2 1 6
0 11 7 5 2 9 10 4 8 1 3 6
0 10 11 7 1 8 5 9 2 4 3 6
0 9 7 3 8 10 4 11 2 1 5 6
0 7 11 1 4 5 2 10 3 9 8 6
0 8 2 3 5 9 7 4 11 10 1 6
0 8 3 4 10 7 5 9 11 2 1 6
0 2 1 7 3 10 8 11 4 5 9 6
0 5 9 7 8 4 10 1 3 2 11 6
0 9 3 4 8 1 11 2 10 5 7 6
0 8 2 5 3 4 11 10 7 9 1 6
0 9 2 4 5 8 3 1 7 11 10 6

```

## 函数组: Intlink.c

1. Intlink() MIC3.24
2. Matprint() MIC3.20
3. Zeromat() MIC2.20

## 功能

生在一系列 12 音调的音列, 并在音程重复处理处连接。

## 注释

本组函数算法很简单, 首先, Intlink()函数生成一系列无序的(从 1 到 12 之间的数), 然后查找出出现重复音程的地方并将此点以后的音作为生成下一列音的核心, 随后再随机生成一些数与上述核心数据一起构成下一音列, 这样就生成不同长度的、间接相关的重复音符组(它们有一个总、前后次序关系), 并且这些音符组可作为任何音乐参数的指针。

虽然从理论上可以实现在连续生成的任何长度的音符序列之间保持次序关系(这些音列长度可从 2 到 12 个音), 但实际上音列长度越大这种次序关系越难保证。

## 编程提示

编写一程序实现下列功能:

1. 根据返回的音列定节拍拍数。
2. 随机对将音列中的音升或降八度。
3. 应用函数 MIC3.6:Alterseq()来加长或缩短 2 步骤中的乐句音列。
4. 将各音高、节拍数据建文件并存盘。

## 程序清单

```
/* INTLINK.C [Generates a stream of redundant-interval-linked
               12-tone series) */
#include <stdio.h>
main()
{
    int total = 10, seed = -2929;
    void Intlink();

    srand(seed);
    Intlink(total);
} /* end of main */
/*===== MIC 3.24 =====*/
/* Intlink function()
-- generates a random-order series, locates the
series position at which interval-size
redundancy occurs, then links subsequent set
permutations by moving remaining values from flag-
point in the current series to the beginning of the
```

next series to form its nucleus. Appropriate values are then permuted to complete the (now) current series. Thus, value-group repetitions of various lengths are produced - within an overall serial context - as a method of generating new related sequences for use as pointers to pitch, rhythm, volume, articulation, or other parameter elements. \*/

```
void Intlink(total)
int total;
{
    int pclass[12]; /* array holding current series */
    int pcum[12]; /* array of pitch class repetition flags */
    int icum[12]; /* array of interval class repetition flags */
    int j,k; /* loop indices */
    int u; /* random integer */
    int intsize; /* interval between pitch classes */
    int redpt; /* interval redundancy point marker */
    int shift; /* shiftpoint for remaining set members */
    void Zeromat(), Matprint();

    Zeromat(pclass);
    Zeromat(icum);
    Zeromat(pcum);
    for (j = 0; j < total; j++)
    {
        for (k = 0; k < 12; k++)
            if (pclass[k] > 0)
                pcum[pclass[k]] = 1;
        for (k = 0; k < 12; k++)
        {
            if (pclass[k] > 0)
                continue;
            do
                u = (rand() % 12);
            while (pcum[u] == 1);
            pcum[u] = 1;
            pclass[k] = u;
        }
        for (k = 0; k < 11; k++)
        {
            intsize = pclass[k+1] - pclass[k];
            if (intsize < 0)
                intsize = 12 + intsize;
            if (icum[intsize] == 1)
                break;
            icum[intsize] = 1;
        }
        redpt = k+1;
        printf("\ninterval redundancy occurs");
        printf(" at position %d\n", redpt+1);
        Matprint(pclass, redpt);
        Zeromat(icum);
        Zeromat(pcum);
        shift = 0;
        for (k = redpt; k <= 12; k++)
        {
            pclass[shift] = pclass[k];
            shift = shift + 1;
        }
        for (k = shift; k < 12; k++)
            pclass[k] = 0;
    }
}
```

```

} /* end of Intlink() function */
/*===== MIC 3.20 =====*/
/* Matprint function */
void Matprint(pclass,redpt)
int pclass[],redpt;
{
    int j;
    printf("pc set: ");
    for (j = 0;j < 12;j++)
    {
        printf("%d ",pclass[j]);
        if (j == redpt-1)
            printf("|");
    }
} /* end of Matprint() function */
/*===== MIC 2.25 =====*/
/* Zeromat() function */
void Zeromat(array)
int array[];
{
    int j;

    for (j = 0;j < 12;j++)
        array[j] = 0;
} /* end of Zeromat() function */
/*=====*/
/* END OF INTLINK.C */

```

## 程序运行实例

INTLINK.EXE

enter a random number generator: seed: 4912

```

interval redundancy occurs at position 6
pc set: 10 6 0 7 11 || 3 2 8 1 4 5 9
interval redundancy occurs at position 8
pc set: 3 2 8 1 4 5 9 || 0 11 7 6 10
interval redundancy occurs at position 4
pc set: 0 11 7 || 6 10 5 9 8 1 4 2 3
interval redundancy occurs at position 4
pc set: 6 10 5 || 9 8 1 4 2 3 7 0 11
interval redundancy occurs at position 8
pc set: 9 8 1 4 2 3 7 || 6 11 10 0 5
interval redundancy occurs at position 7
pc set: 6 11 10 2 5 0 || 3 4 7 8 9 1
interval redundancy occurs at position 4
pc set: 3 4 7 || 8 9 1 0 5 2 6 11 10
interval redundancy occurs at position 7
pc set: 8 9 1 0 5 2 || 6 11 10 4 3 7
interval redundancy occurs at position 5
pc set: 6 11 10 4 || 3 7 9 0 5 8 1 2
interval redundancy occurs at position 8
pc set: 3 7 9 4 5 8 1 || 2 11 6 10 0

```

## 函数: Valratio() MIC3.25

### 功能

控制一系列数中重复值的数目与唯一数值的数目的比例。

### 注释

本算法对一在固定范围内连续生成的数列产生一种“浮动序列”。也即，先设置一个限值  $n$ ，函数返回的数列在任何一组相邻  $n$  个数中间不可出现重复。例如，若函数返回 1-10 之间的含 100 个数的数列，而函数定的限值为 9，则输出数列中不会出现重复数值，但若定的限值为 4，则任一组 4 个相邻数中间不可有重复。

### 编程提示

1. 修改主程序，允许用户变量 total, range 和 window 的值。
2. 编写一程序应用函数 Valratio() (在完成每一次运行后) 返回一系列数据，作为音高、升、降音程值、节拍、音量、发音方式等各音乐参数的指针。
3. 扩展 2) 中所述程序引入函数 MIC3.6，加入写文件编码以便将程序运行结果存入四个单独的文件中，这四个文件分别存放某一乐句中各音的音高、节拍、发音方式以及音量数据。在数字式音乐合成器同时运行上述四文件。

### 程序清单

```
/* VALRATIO.C (Controls the ratio of repeated integers to
               unique integers in a random-order stream
               of values. */
#include <stdio.h>
#include <math.h>
int finalseq[100];
main()
{
    extern finalseq[]; /* array holding filtered values */
    int j; /* loop index */
    int total = 100;
    int seed = -1922;
    int range = 12; /* range of integers, start = 0 */
    int window = 4; /* size of repetition-prevent window */
    void Valratio();

    srand(seed);
    Valratio(range, window, total);
    printf("\napplied to random-order pitches (0-11)\n");
    printf("\nrepetition-prevent window is %d ;", window);
    for(j = 0; j < total; j++)
        if(j % 20 == 0)
            printf("\n");
}
```

```

        else
            printf("%d ",finalseq[j]);
    }
    /* end of main */
    /*===== MIC 3.25 =====*/
    /* Valratio() function */
    void Valratio(span,gap,total)
    int span,gap;
    {
        extern finalseq[];
        int pcum[12]; /* array of integer occurrence flags */
        int j;        /* loop index */
        int u;        /* random integer */

        for(j = 0;j < total;j++)
        {
            do
                u = rand() % span;
            while (pcum[u] == 1);
            pcum[u] = 1;
            finalseq[j] = u;
            if(j <= gap)
                continue;
            pcum[finalseq[j-gap]] = 0;
        }
    } /* end of Valratio() function */
    /*=====*/
    /* END OF VALRATIO.C */

```

### 程序运行实例

VALRATIO.EXE

applied to random-order pitches {0-11}

repetition-prevent window is 4 ;

```

11 4 6 0 10 11 8 1 4 6 11 0 5 1 9 8 4 5 3
10 11 7 5 9 3 10 7 1 6 8 0 4 11 1 10 1 0 8
3 6 11 1 10 8 9 7 3 0 10 4 7 11 9 10 4 1 0
6 3 10 7 0 9 11 3 10 0 7 11 5 3 6 0 8 9 1
6 3 9 8 5 6 0 10 4 5 3 1 0 9 7 3 5 4 9

```

### 函数组: Markov.c

1. Markov() MIC3.26
2. Zeromat() MIC3.25
3. Freqtab2() MIC3.27
4. Seedtest() MIC3.28

### 功能

根据主程序中给出的音乐参数数据生成一个 Markovian 整数求和余数图, 并列图中整数出现频率表。

### 注释

本算法是由作曲家 Peter Armstrong 设计的一种作曲算法演变而成。Peter 设计此算法的目的是为了将各种音列(包括 12 音音列和其他各种音列)数据应用于微分音阶、节拍值表等方面。

可以通过改变控制参量 order 和 apply 来使本算法用途更广泛。你可以根据源输入元素的数目以及数值和求和方式来生成各种类型的升序整数数列。(若参数 apply 被设置为 Exclusive, 则只有外层的源元素对相加生成新元素, 而若参量 apply 设置为 inclusive, 则所有源元素相加生成新元素。)应用 Markovian 算法构造数列的两个简单例子是众所周知的菲波那契数列和卢卡斯数列。

控制变量 modulus 将升序数列转化为可处理的格式(若不经转化, 直接将这些数列作为音乐参数数列将会引起列中元素数值大大超过参数的取值范围)。modulus 的值已在主程序中定为 12, 这样输出的数据是 0 到 11 之间的数, 可以适用于第二章中 12 音调音列的数值系统。取 modulus 为其他值则可将输出整数数列限定在某一数值范围内。

#### 编程提示

编写一交互式程序, 应用函数 MIC3.26:Markov 和 MIC2.1:Tunings()在音乐合成器上生成一组不同、八度重复\*的微分音音列。(可用 modulus 定每个八度内音符的个数。)

#### 程序清单

```
/* MARKOV.C (Markovian Integer Summation Residual Cycle
Generator) */
#include <stdio.h>
main()
{
    int cycle[5001]; /* stores generated integer cycle */
    int seed[10]; /* array of seeds (initial terms to sum) */
    int freq[88]; /* array of integer occurrence frequencies */
    int order; /* num. of preceding terms to sum for next term */
    int modulus; /* integer-range modulus imposed on each sum */
    int total; /* total terms generated */
    int apply; /* signal: inclusive or exclusive seed addition */
    int j; /* loop index */
    int Markov();
    void Zeromat(), Freqtab2();

    printf("enter modulus:\n");
    scanf("%d", &modulus);
    printf("%s%s", "enter 0 for inclusive or",
           " 1 for exclusive seed addition.\n");
    scanf("%d", &apply);
    printf("\nhow many seeds will you enter (min=2):\n");
    scanf("%d", &order);
    while (order < 2)
    {
        printf("\nyou must enter at least TWO seeds:\n");
        printf("how many seeds?");
        scanf("%d", &order);
    }
}
```

```

for (j = 1; j <= order-1; j++)
{
    printf("enter seed %d\n", j);
    scanf("%d", &seed[j]);
}
printf("\n");
for (j = 1; j <= order; j++)
    printf("seeds=%d ", seed[j]);
total = Markov(cycle, seed, order, apply, modulus);
Zeromat(freq, modulus, total);
Freqtab2(cycle, freq, modulus, total);
printf("\ntotal = %d\n", total);
for (j = 1; j <= total; j++)
{
    printf("%d ", cycle[j]);
    if (j % 15 == 0)
        printf("\n");
}
printf("\n\noccurrence frequency table\n");
for (j = 1; j <= modulus; j++)
{
    printf("%d : %d ", j-1, freq[j]);
    if (j % 5 == 0)
        printf("\n");
}
printf("\n\n");
/* end of main */
/***** MIC 3.26 *****/
/* Markov() function */
int Markov(cycle, seed, order, apply, mod)
int cycle[], seed[], order, apply, mod;
{
    int nxterm, total, done, j, l, Seedtest();
    for (j = 1; j <= order; j++)
        cycle[j] = seed[j];
    total = order;
    for (j = 1; j <= 5000; j++)
    {
        nxterm = seed[order];
        for (l = 1; l <= order-1; l++)
            nxterm += seed[l];
        if (apply == 1)
            nxterm = seed[order] + seed[1];
        nxterm = nxterm * mod;
        cycle[j+order] = nxterm;
        for (l = 1; l <= order-1; l++)
            seed[l] = seed[l+1];
        seed[order] = nxterm;
        total += 1;
        done = Seedtest(order, seed, cycle);
        if (done == 1)
        {
            total -= order;
            break;
        }
    }
    return(total);
}

```



```

    } /* end of Markov() function */
    /*===== MIC 2.25 =====*/
    /* Zeromat() function */
    void Zeromat(array,mod)
    int array[],mod;
    {
        int j;

        for (j = 0;j <= mod;j++)
            array[j] = 0;
    } /* end of Zeromat() function */
    /*===== MIC 3.27 =====*/
    /* Freqtab2() function */
    void Freqtab2(array1,array2,mod,total)
    int array1[],array2[],mod,total;
    {
        int j,k;

        for (j = 1;j <= mod;j++)
        {
            for (k = 1;k <= total;k++)
                if (array1[k] + 1 == j)
                    array2[j] = array2[j] + 1;
        }
    } /* end of Freqtab2() function */
    /*===== MIC 3.28 =====*/
    /* Seedtest() function */
    int Seedtest(order,seed,cycle)
    int order,seed[],cycle[];
    {
        int k,done=1;

        for (k = 1;k <= order;k++)
            if (seed[k] != cycle[k])
            {
                done = 0;
                break;
            }
        return(done);
    } /* end of Seedtest() function */
    /*===== MIC 3.29 =====*/
    /* END OF MARKOV.C */

```

## 程序运行实例

```

MARKOV.EXE

enter modulus: 12
enter 0 for inclusive or 1 for exclusive seed addition: 1

how many seeds will you enter (2-5): 5
enter seed 1
enter seed 2
enter seed 3
enter seed 4
enter seed 5

seeds=1 seeds=2 seeds=3 seeds=4 seeds=5
total =312

```

```

1 2 3 4 5 3 5 8 1 10 3 3 1 6 11
0 9 3 5 4 9 6 3 3 1 10 11 4 5 7
1 4 9 2 11 3 5 6 3 4 9 3 1 8 1
10 11 7 1 6 11 0 1 7 1 8 5 10 7 7
1 6 7 4 1 7 1 8 9 2 3 11 9 10 11
8 1 3 9 8 5 2 3 3 9 10 3 4 5 7
5 0 9 2 11 3 1 2 7 0 1 11 9 4 1
2 3 7 5 6 11 8 1 7 9 0 1 6 11 3
9 6 11 4 9 3 9 0 1 10 11 7 5 10 7
4 9 11 5 0 5 6 3 7 9 6 7 8 1 7
5 4 1 6 11 3 1 10 7 8 5 7 1 4 1
6 7 7 1 10 7 8 9 11 9 8 9 10 11 11
1 6 3 8 5 11 9 0 9 10 3 7 5 10 11
0 9 11 5 0 1 2 7 3 1 2 3 4 1 11
9 4 5 5 11 11 1 10 3 0 1 3 5 0 9
6 11 7 9 6 3 0 1 7 5 4 5 10 7 7
9 2 11 0 5 3 9 4 9 6 7 11 1 10 11
4 1 3 5 0 1 10 7 11 5 10 7 4 1 3
1 4 1 10 7 11 9 2 3 8 9 7 5 8 1
6 3 11 5 2 3 0 9 7 9 4 5 10 11 3
9 2 11 0 1 11 1 0 1 2 3 7

```

occurrence frequency table

```

0 : 18   1 : 43   2 : 14   3 : 36   4 : 20
5 : 28   6 : 18   7 : 34   8 : 14   9 : 33
10 : 20  11 : 34

```

### 函数组: Tropes.c

1. Tropes() MIC3.27
2. Zeromat() MIC2.25
3. Setflag() MIC3.7
4. Matprint() \* MIC3.20

### 功能

按一定规则生成一系列整数数列的排列。其规则是按统一的间隔数循环选取序列中的元素。

### 注释

函数 Tropes()通过每个循环逐渐增大选数间隔数生成质数数列的 11 种排列。实际上这相当于在第二次循环中按次序每隔一数选数，在第三次循环中按次序每隔两数选一数，以此类推。并且保持有效原质数数列的数组不变以便在下一步运行中应用。输出的排列有效于一静态数组中以使用 Matprint()函数显示结果。

### 编程提示

1. 修改函数组，使其可实现将所有排列存放在一个二维矩阵中。

2. 将主程序修改为交互式程序，允许用户选择排列的级别(根据选数间隔值)以及从键盘输入原始数列。
3. 编写一程序应用函数组CURVES.C (MIC2.6、MIC2.7和MIC2.8)，按一定规则生成参数值序列的排列。这些参数值序列包括线性定标、指数定标、对数定标的数值序列。
4. 在上述过程中加入一函数以实现对新生成的“奇异切面”再进行“切面”运算。

#### 程序清单

```

/* TROPES.C (returns systematic permutations of a numeric
   series array */
#define MAXDATA 12
main()
{
    int x[MAXDATA], y[MAXDATA];
    int j,k; /* loop indices */
    int bottom = 0; /* lowest integer value to generate */
    int range = 12; /* value range of integers to generate */
    int total = 12; /* number of integers in series */

    int seed = -9033;
    void Setflag();
    void Matprint();

    srand(seed);
    printf("generating a random-order number series:\n");
    Setflag(x,bottom,range,total);
    Matprint(x,total);
    printf("\nnow tropes (permutations) will be output:");
    for (j = 1; j < total; j++)
    {
        printf("\n order %d\n", j);
        Tropes(x, total, j);
    }
} /* end of main() */
/*===== MIC 3.27 =====*/
Tropes(array1, size, order)
int array1[], size, order;
{
    int j, k, offset, array2[MAXDATA];
    void Matprint();
    for (offset = 0, k = 0; offset < order; offset++)
        for (j = offset; j < size; j += order)
            array2[k++] = array1[j];
    Matprint(array2,size);
} /* end of Tropes() function */
/*===== MIC 2.25 =====*/
/* Zeromat() function ( adapted to program specs) */
void Zeromat(array,range)
int array[],range;
{
    int j;

    for (j = 0; j < range; j++)
        array[j] = 0;
} /* end of Zeromat() function */
/*===== MIC 3.7 =====*/
/* Setflag() function */

```

```

void Setflag(array,bottom,span,total)
int array[],bottom,span,total;
{
    int j,u,cum[12];
    void Zeromat();

    Zeromat(cum,total);
    for (j = 0;j < total;j++)
    {
        do
            u = rand() % span + bottom;
        while (cum[u] == 1);
        cum[u] = 1;
        array[j] = u;
    }
} /* end of Setflag() function */
/*===== MIC 3.20 =====*/
/* Matprint() function (adapted to program specs) */
void Matprint(array,total)
int array[],total;
{
    int k;
    for (k = 0;k < total;k++)
        printf("%d ",array[k]);
    printf("\n");
} /* end of Matprint() function */
/*=====*/
/* END OF TROPES.C */

```

#### 程序运行实例

```

TROPES.EXE

generating a random-order number series:
3 6 8 1 4 7 0 11 9 2 10 5

now tropes (permutations) will be output:
order 1
3 6 8 1 4 7 0 11 9 2 10 5

order 2
3 8 4 0 9 10 6 1 7 11 2 5

order 3
3 1 0 2 5 4 11 10 8 7 9 5

order 4
3 4 9 6 7 2 8 0 10 1 11 5

order 5
3 7 10 6 0 5 8 11 1 9 4 2

order 6
3 0 6 11 8 9 1 2 4 10 7 5

order 7
3 11 6 9 8 2 1 10 4 5 7 0

order 8
3 9 6 2 8 10 1 5 4 7 0 11

order 9
3 2 6 10 8 5 1 4 7 0 11 9

```

```

order 10
3 10 6 5 8 1 4 7 0 11 9 2

order 11
3 5 6 8 1 4 7 0 11 9 2 10

```

## 函数组: Mirror.c

1. PutPeform() MIC3.28
2. WriteComment() MIC3.29
3. loop() MIC3.30
4. PutChord() MIC3.31
5. GetChord() MIC3.32
6. GetPatterns() MIC3.33
7. OpenChord() MIC3.34
8. EndChord() MIC3.35
9. GetCenter() MIC3.36
10. PutRhythm() MIC3.37
11. OpenMot() MIC3.38
12. EndMot() MIC3.39
13. InvertShaper() MIC3.40

## 功能

根据输入音列、以某一音为中心生成一系列的弦音作为镜象音。

## 注释

用以生成和弦音的中心音符可以是固定的 F#3(电子乐器键盘中央键位音),也可以根据输入值来定。将每一音程序列进行旋转运算,生成的镜象和弦音可定义为“主题”。(用户必须通知机器是否用到输入的所有格式),通常采用较短音程的结构会收到极佳效果,节拍输出值可以是统一的音长值,也可以是一系列代表镜象和弦音格式的数值。

## 注意

我们在本函数组中所列程序与以上各函数不同。在本函数中,我们不是列出一段应用某种算法的非交互式源程序,而是着重列示了一种 C 语言编程者经常用来精简程序和长程序中加强数据类型检验功能的编程方法。为了能够正确运行本段程序,读者必须先掌握输入数据的大致期望值以及音项表这种特殊的乐谱形式(详见第七章中函数组: Scorform.c)。

注意在程序中多次使用了头文件引入指令`#include "文件名"`以便精简程序。因为本程序中用到的许多函数已在书中其他章节详细介绍，所以此处仅以头文件指令在编译过程中将其引入本程序以供使用，而不再将这些函数的程序清单列在本程序中，这一方法在编程中很有好处。

另外，请注意本程序中引入模型测试系统，函数模型以其对应的函数的名字存盘，但扩展名由.C 改为.PRO。模型测试是 C 语言的新成果，如果你所使用的编译系统中无此项功能，则只要将程序中的`#include "文件名"`伪指令删掉即可。但若你是在 Turbo C 运行环境下运行该程序，则最好利用模型文件来加强其查错功能。

本函数组中还通过写入多条`#include "文件名"`指令进一步精简程序。所有用该指令引入的非标准头文件都已在附录 A 列出，编译时，你的.H 子目录下或总目录下必须有这些头文件，否则，编译程序将会显示查找无效错误信息。

所有引入的扩展名为.C 的子程序都在附录 B 中列出。在编译前，这些子程序必须都已存入盘中以备主程序调用。这些子程序包括：

1. array.c      MIC\_SP1.0
2. scorform.c   MIC\_SP6.0
3. getpart.c     MIC\_SP3.0
4. getnam.c      MIC\_SP4.0
5. inputyou.c    MIC\_SP2.0

所有引入的模型文件都在附录 C 中列出。

## 程序清单

```
/* MIRROR.C (Builds chords around a central pitch as a mirror
             according to the interval sequence of a stream
             of input pitches) */
#include <stdio.h>
#define MAXDATA 200
#include "array.c"
#include "inputyou.c"
#include "getpart.c"
#include "scorform.c"
#include "getfnam.c"
#include "mirror.pro"

main()
{
    int shape[MAXDATA], datarray[MAXDATA];
    int inv[MAXDATA], disarray[MAXDATA];
    int size, ShapeSize, usescore, j, which;
    char motname[8];
    FILE *outputfile;

    outputfile = getfnam("Output", "w");
    size = InputYourOwn(disarray);
    size = getpart(disarray, datarray, size, 1);
    ShapeSize = size - 1;
    getshape(datarray, shape, size);
    printf("\nWhat do you want to call the motifs?\n");
    scanf("%s", &motname);
    printf("(1) Default center is F#3 (2) moving center\n");
}
```

```

scanf("%d", &which);
WriteComment(outputfile, datarray, size, "Pitches");
for(j = 0; j < ShapeSize; j++)
{
    RotateArray(shape, shape, ShapeSize, j);
    WriteComment(outputfile, shape, ShapeSize, "Intervals");
    printf("\nRotation no %d\n", j);
    if(which == 1)
        loop(outputfile, shape, ShapeSize, j, 42, motname);
    else
        loop(outputfile, shape, ShapeSize, j, datarray[j], motname);
    PutLn();
    FputLn(outputfile);
}
PutPerform(outputfile, ShapeSize, motname);
}
/*===== MIC 3.28 =====*/
PutPerform(fp, ShapeSize, motname)
FILE *fp;
int ShapeSize;
char motname[];
{
    int j;
    fprintf(fp, "\nNotelist using R 1-1\n");
    for(j = 0; j < ShapeSize; j++)
        fprintf(fp, "Perform %std\nP R\nR 4\n", motname, j);
}
/*===== MIC 3.29 =====*/
WriteComment(fp, data, size, comment)
FILE *fp;
int data[], size;
char comment[];
{
    printf("\n/*\n");
    printf("%s\n", comment);
    PutArray(data, size);
    printf("*/\n");

    fprintf(fp, "\n/*\n");
    fprintf(fp, "%s\n", comment);
    FputArray(fp, data, size);
    fprintf(fp, "*/\n");
}
/*===== MIC 3.30 =====*/
loop(fp, shape, ShapeSize, num, center, motname)
FILE *fp;
int shape[], ShapeSize, num, center;
char motname[];
{
    int up[MAXDATA], down[MAXDATA], chord[MAXDATA], ChordSize;

    fprintf(fp, "\n/* center = %d ", center);
    printf("\n/* center = %d ", center);
    PutPitch(center);
    FputPitch(fp, center);
    printf(" */\n");
    fprintf(fp, " */\n");
    GetCenter(up, down, center);
    GetPatterns(shape, up, down, ShapeSize);
    ChordSize = GetChord(up, down, chord, ShapeSize);
    OpenMot(fp, motname);
    printf("%d", num);
}

```

```

        fprintf(fp, "%d", num);
        OpenChord(fp);
        PutChord(fp, chord, ChordSize);
        EndChord(fp);
        PutRhythm(fp, chord, ChordSize);
        EndMot(fp, motname);
        printf("%d\n", num);
        fprintf(fp, "%d\n", num);
    }

/*===== MIC 3.31 =====*/
PutChord(fp, chord, ChordSize)
    FILE *fp;
    int chord[], ChordSize;
{
    int j;

    for(j = 0; j < ChordSize; j++)
        if(chord[j] != chord[j + 1]) /* no repeated pitches */
        {
            PutPitch(chord[j]);
            FputPitch(fp, chord[j]);
        }
}

/*===== MIC 3.32 =====*/
GetChord(up, down, chord, size)
    int up[], down[], chord[], size;
{
    int j, ChordSize;

    ChordSize = 0;
    for(j = size; j > 0; j--)
        if(down[j] >= 0 && down[j] <= 85)
            chord[ChordSize++] = down[j];
    for(j = 0; j <= size; j++)
        if(up[j] >= 0 && up[j] <= 85)
            chord[ChordSize++] = up[j];
    return(ChordSize);
}

/*===== MIC 3.33 =====*/
GetPatterns(disarray, up, down, size)
    int disarray[], up[], down[], size;
{
    int j;

    for(j = 0; j < size; j++)
        {
            up[j + 1] = up[j] + abs(disarray[j]);
            down[j + 1] = down[j] - abs(disarray[j]);
        }
}

/*===== MIC 3.34 =====*/
OpenChord(fp)
    FILE *fp;
{

```



```

printf("\n P { ").
fprintf(fp, "\n P { ");
}

```

```

/*===== MIC 3.35 =====*/

```

```

EndChord(fp)
FILE *fp;
{
    printf("\n\n");
    fprintf(fp, "\n\n");
}

```

```

/*===== MIC 3.36 =====*/

```

```

GetCenter(up, down, center)
int up[], down[], center;
{
    up[0] = center;
    down[0] = center;
}

```

```

/*===== MIC 3.17 =====*/

```

```

PutRhythm(fp, chord, ChordSize)
FILE *fp;
int chord[], ChordSize;
{
    int j, count;

    count = 0;
    printf(" R ");
    fprintf(fp, "/*R ");
    for( j = 0; j < ChordSize; j++)
    {
        if(chord[j] != chord[j + 1]) /* no repeated pitches */
        {
            printf("%3d ", chord[j] + 1); /* no 0 on rhythm lines */
            fprintf(fp, "%3d ", chord[j] + 1);
            count++;
        }
    }

    fprintf(fp, " */");
    fprintf(fp, "\n R (1)%d ", count);
}

```

```

/*===== MIC 3.38 =====*/

```

```

OpenMot(fp, motname)
FILE *fp;
char motname[];
{
    printf("\n Def Mot %s", motname);
    fprintf(fp, "\n Def Mot %s", motname);
}

```

```

/*===== MIC 3.39 =====*/

```

```

EndMot(fp, motname)
FILE *fp;
char motname[];
{
    printf("\n End %s", motname);
    fprintf(fp, "\n End %s", motname);
}

```

```

    )

/*===== MIC 3.40 =====*/
InvertShape(shape, inv, size)
    int shape[], inv[], size;
{
    int j;
    for(j = 0; j < size; j++)
    {
        if(shape[j] > 0)
            inv[j] = shape[j] - abs(shape[j] * 2);
        else
            inv[j] = shape[j] + abs(shape[j] * 2);
    }
}
/*=====*/

```

#### Mirror.c 输出实例

```

/*
Pitches
 5 6 7 8 9 10
*/

/*
Intervals
 1 1 1 1 1
*/

/* center = 42 F#3 */

Def Mot tune0
P [ C#3 D3 D#3 E3 F3 F#3 G3 G#3 A3 A#3 B3 ]
/*R 38 39 40 41 42 43 44 45 46 47 48 */
R (1)11
End tune0

/*-----*/

/*
Intervals
 1 1 1 1 1
*/

/* center = 42 F#3 */

Def Mot tune1
P [ C#3 D3 D#3 E3 F3 F#3 G3 G#3 A3 A#3 B3 ]
/*R 38 39 40 41 42 43 44 45 46 47 48 */
R (1)11
End tune1

/*-----*/

/*

```

```

Intervals
  1  1  1  1  1
*/

/* center = 42 F#3 */

Def Mot tune2
P [ C#3 D3 D#3 E3 F3 F#3 G3 G#3 A3 A#3 B3 ]
/*R 38 39 40 41 42 43 44 45 46 47 48 */
R (1)11
End tune2

/*-----*/

/*
Intervals
  1  1  1  1  1
*/

/* center = 42 F#3 */

Def Mot tune3
P [ C#3 D3 D#3 E3 F3 F#3 G3 G#3 A3 A#3 B3 ]
/*R 38 39 40 41 42 43 44 45 46 47 48 */
R (1)11
End tune3

/*-----*/

/*
Intervals
  1  1  1  1  1
*/

/* center = 42 F#3 */

Def Mot tune4
P [ C#3 D3 D#3 E3 F3 F#3 G3 G#3 A3 A#3 B3 ]
/*R 38 39 40 41 42 43 44 45 46 47 48 */
R (1)11
End tune4

/*-----*/

Notelist using R 1-1
Perform tune0
P R
R 4
Perform tune1
P R
R 4
Perform tune2
P R
R 4
Perform tune3
P R
R 4
Perform tune4
P R
R 4

```

## 函数组: Sets.c

(包含 MIC3.50-MIC3.81 中的所有函数)

### 功能

对一系列音符进行分析操作。

### 注释

Sets.c 是一组乐音分析函数。用户通过源文件或由键盘直接输入音符数据，并可调整数据集合大小和实现搭接功能。通过搭接输入数据可以发现一些不易看出的数据集合形式。输入音符被妥善排序，集合根据其所含元素类型进行标识。程序提供已标定集合的信息，甚至包括一部分集合结构消息，最后将被选定的数据集合以简表形式显示。

### 注意

为了正确运行本函数组中的函数，你必须加入先将用 include 伪指令引入的 sets.h 头文件以及其它为主程序认可的头文件放于编译目录中。因此，如果你仔细查看一下程序中的各条 include 语句，你会发现它们依次访问下列子程序(扩展名为.c)。

1. array.c        MIC\_\_SP1.0
2. inputyou.c    MIC\_\_SP2.0
3. scriptprs.c   MIC\_\_SP3.0
4. quicksort.c   MIC\_\_SP4.0
5. ansi.c        MIC\_\_SP5.0

### 编程提示

此处只给出了含 3、4、5 和 6 音的音符组，但实际上含有更高基本音的音符组也可加以应用。而且集合体可以是相邻的基本音符。用户可能希望将集合体列表显示出来，若用数组初始化的方法显示，虽然简单明了，但却会占用太多内存空间。这种情况下，可以位字段形式存储数据信息以节省内存。

### 程序清单

```

/* SETS.C (Series Analysis Routines) */

#include <stdlib.h>
#include "sets.h"
#include "sets.pro"

main()
{
    register int k;
    int index, imsize, imbeg, totsize, setnumber[MAXDATA], ask;
    int parray[MAXDATA], size;
    int setfq[50], setsize, original[12], vector[6];
    int imarray[MAXDATA], test[12], testshape[12], ordered[12];
    FILE *fopen();

    outputfile = fopen("set.dat", "w");
    Cls();
    centerf("Set Analysis\n");
    size = GetData(datararray, setfq, parray);
    showdata(outputfile, datarray, parray, size);
    imsize = GetImsize();
    ShowImbricated(outputfile, parray, imarray, imsize, size);
    setsize = GetSetSize();
    index = GetIndex(imsize, size, setsize);

    for(k = 0; k < index; k++)
    {
        GetTestSets(test, parray, datarray, original, imarray,
                    setsize, imsize, k);
        ShowOriginal(outputfile, original, test, setsize);
        Order(test, ordered, vector, setnumber, setsize, k);
        if(setnumber[k] == 0)
            NotSetError(outputfile, setsize);
        else if(setnumber[k] != 0)
            PutSetData(outputfile, test, setsize, vector,
                       setnumber[k], setfq);
    }
    PutSetSummary(outputfile, setfq, setsize);
    PutTable(outputfile);
    printf("\n\nAnalysis is in file: set.dat\n\n");
    fclose(outputfile);
}

/*===== MIC 3.40 =====*/
GetIndex(imsize, size, setsize)
    int imsize, size, setsize;
{
    if(imsize == 0)
        return(size / setsize);
    else
        return(size - (imsize-1));
}

/*===== MIC 3.41 =====*/
GetSetSize()
{
    int setsize;
    printf("\nEnter set size 3, 4, 5 or 6 ");
    scanf("%d", &setsize);
    return(setsize);
}

```

```

}
/*===== MIC 3.42 =====*/
GetImsize()
{
    int imsize;
    printf("\nEnter size of imbrication (0 for no imbrication) ");
    scanf("%d", &imsize);
    return(imsize);
}
/*===== MIC 3.43 =====*/
Order(test, ordered, vector, setnumber, setsize, k)
    int test[], ordered[], vector[], setnumber[], setsize, k;
{
    QuickSort(test, 0, setsize-1);
    NormalOrder(test, ordered, setsize);
    getshape(ordered, vector, setsize);
    endvector(vector, setsize);
    setnumber[k] = GetSetnumber(vector, setsize);
}
/*===== MIC 3.44 =====*/
ShowOriginal(fp, original, test, setsize)
    FILE *fp;
    int original[], test[], setsize;
{
    PutLn();
    FputLn(fp);
    printf("\nOriginal pitches:");
    fprintf(fp, "\nOriginal pitches:");
    ScriptArray(original, ' ', setsize);
    FscriptArray(fp, original, ' ', setsize);
    printf("Pc array:          ");
    PutArray(test, setsize);
    fprintf(fp, "Pc array:          ");
    FputArray(fp, test, setsize);
}
/*===== MIC 3.45 =====*/
GetTestSets(test, parray, datarray, original, imarray,
    setsize, imsize, k)
    int test[], parray[], datarray[], original[], imarray[],
    setsize, imsize, k;
{
    int j;

    if (imsize == 0)
        for(j = 0; j < setsize; j++)
        {
            test[j] = parray[j + k * setsize];
            original[j] = datarray[j + k * setsize];
        }
    else
        for(j = 0; j < imsize; j++)
        {
            test[j] = imarray[j + k * imsize];
            original[j] = datarray[j + k];
        }
}
/*===== MIC 3.46 =====*/
GetSetnumber(ordered, setsize)
    int ordered[], setsize;
{
    int setnumber;

```

```

switch(setsize)
{
case 3 : setnumber = set3cmp(ordered);
        break;
case 4 : setnumber = set4cmp(ordered);
        break;
case 5 : setnumber = set5cmp(ordered);
        break;
case 6 : setnumber = set6cmp(ordered);
        break;
}
return(setnumber);
}
/***** MIC 3.47 *****/
NotSetError(fp, setsize)
FILE *fp;
int setsize;
{
    PutLn();
    FputLn(fp);
    printf("\n\t\t\t\t\t**** NOT A %d NOTE SET ***\n",setsize);
    fprintf(fp, "\n\t\t\t\t\t**** NOT A %d NOTE SET ***\n",setsize);
    putchar(7);
    notset++;
}
/***** MIC 3.48 *****/
PutSetData(fp, test, setsize, ordered, setnumber, setfq)
FILE *fp;
int test[], setsize, ordered[], setnumber, setfq[];
{
    printf("Ordered pc array:      ");
    PutArray(test, setsize);
    fprintf(fp, "Ordered pc array:      ");
    FputArray(fp, test, setsize);
    printf("As pitches:");
    fprintf(fp, "As pitches:");
    ScriptArray(test, ' ', setsize);
    FscriptArray(fp, test, ' ', setsize);
    printf("Ordered interval array:");
    PutArray(ordered, setsize);
    fprintf(fp, "Ordered interval array:");
    FputArray(fp, ordered, setsize);
    printf("Set = %d-%d\n", setsize, setnumber);
    fprintf(fp, "Set = %d-%d\n", setsize, setnumber);
    setfq[setnumber - 1]++;
    switch(setsize)
    {
        case 3: PrintSetInfo(set3, setnumber, setsize);
                FprintSetInfo(fp, set3, setnumber, setsize);
                break;
        case 4: PrintSetInfo(set4, setnumber, setsize);
                FprintSetInfo(fp, set4, setnumber, setsize);
                break;
        case 5: PrintSetInfo(set5, setnumber, setsize);
                FprintSetInfo(fp, set5, setnumber, setsize);
                break;
        case 6: PrintSetInfo(set6, setnumber, setsize);
                FprintSetInfo(fp, set6, setnumber, setsize);
                break;
    }
}

```

```

/*===== MIC 3.49 =====*/
PutSetSummary(fp, setfq, setsize)
    FILE *fp;
    int setfq[], setsize;
{
    int j;
    PutLn();
    FputLn(fp);
    printf( "\n\nSet summary: \n");
    fprintf(fp, "\n\nSet summary: \n");
    for(j = 0; j < amt[setsize - 3]; j++)
    {
        if(setfq[j] != 0)
        {
            printf("\nSet %d-%d occurs %d ", setsize, j+1 ,
                setfq[j]);
            fprintf(fp, "\nSet %d-%d occurs %d ", setsize, j+1,
                setfq[j]);
            if(setfq[j] == 1)
            {
                printf("time");
                fprintf(fp, "time");
            }
            else
            {
                printf("times");
                fprintf(fp, "times");
            }
        }
        printf("\nThere were %d non-sets", notset);
        fprintf(fp, "\nThere were %d non-sets", notset);
        PutLn();
        FputLn(fp);
    }
}
/*===== MIC 3.50 =====*/
PutTable(fp)
    FILE *fp;
{
    int ask, j;
    do
    {
        printf("Do you want a set table? yes = 1, no = 2\n");
        scanf("%d", &ask);
    } while(ask < 1 || ask > 2);

    if(ask == 1)
    {
        printf( "\n\nHere is the 3 note set table\n");
        fprintf(fp, "\n\nHere is the 3 note set table\n");
        PrintSetTable(set3, 3);
        FprintSetTable(fp, set3, 3);
        printf( "\n\nHere is the 4 note set table\n");
        fprintf(fp, "\n\nHere is the 4 note set table\n");
        PrintSetTable(set4, 4);
        FprintSetTable(fp, set4, 4);
        printf( "\n\nHere is the 5 note set table\n");
        fprintf(fp, "\n\nHere is the 5 note set table\n");
        PrintSetTable(set5, 5);
        FprintSetTable(fp, set5, 5);
        printf( "\n\nHere is the 6 note set table\n");
    }
}

```



```

        fprintf(fp, "\n\nHere is the 6 note set table\n");
        PrintSetTable(set6, 6);
        FprintSetTable(fp, set6, 6);
    }
}

/*===== MIC 3.51 =====*/
GetData(datarray, setfq, parray)
    int datarray[], setfq[], parray[];
{
    int ask, j, size;
    psize = rsize = asize = vsize = tsize = 0;

    do {
        printf("\nDo you wish to type in integers (1) or ");
        printf("Read in a Script score (2) ");
        printf("\nType a 1 or 2\t?\b");
        scanf("%d", &ask);
    } while (ask < 1 || ask > 2);
    if(ask == 1)
        size = InputYourOwn(datarray);
    else
    {
        inputfile = getfnam("input", "r");
        GetLines(inputfile);
        fclose(inputfile);
        switch(WhichArray())
        {
            case 1 : size = psize;
                     printf("\nUsing Pitch Array \n\n");
                     for(j = 0; j < psize; j++)
                         datarray[j] = parray[j];
                     break;
            case 2 : size = rsize;
                     printf("\nUsing Rhythm Array \n\n");
                     for(j = 0; j < rsize; j++)
                         datarray[j] = rarray[j];
                     break;
            case 3 : size = asize;
                     printf("\nUsing Articulation Array \n\n");
                     for(j = 0; j < asize; j++)
                         datarray[j] = aarray[j];
                     break;
            case 4 : size = vsize;
                     printf("\nUsing Volume Array \n\n");
                     for(j = 0; j < vsize; j++)
                         datarray[j] = varray[j];
                     break;
            case 5 : size = tsize;
                     printf("\nUsing Rte Array \n\n");
                     for(j = 0; j < tsize; j++)
                         datarray[j] = tarray[j];
                     break;
        }
    }
    for(j = 0; j < 50; j++)
        setfq[j] = 0;
    for(j = 0; j < size; j++)
        parray[j] = datarray[j] % 12;
    printf("\nUsing %d pitches\n", size);
    return(size);
}

```

```

/*===== MIC 3.52 =====*/
WhichArray()
{
    char string[80];
    int j = 0;

    printf("\nWhich data set do you wish to analyze\n");
    if(psize != 0)
        printf("(1) Pitch data ");
    if(rsize != 0)
        printf("(2) Rhythm data ");
    if(ysize != 0)
        printf("(3) Articulation data ";
    if(vsize != 0)
        printf("(4) Volume data ");
    if(tsize != 0)
        printf("(5) Rte data ");
    do {
        gets(string);
        j = atoi(string);
    } while(j < 1 || j > 5);
    return(j);
}

/*===== MIC 3.53 =====*/
showdata(fp, datarray, parray, size)
FILE *fp;
int datarray[], parray[], size;
{
    printf("\nInput pitches\n");
    fprintf(fp, "\nInput pitches\n");
    ScriptArray(datarray, ' ', size);
    FscriptArray(fp, datarray, ' ', size);
    PutLn();
    FputLn(fp);
    printf("\nPitch classes of input data:\n");
    PutArray(parray, size);
    fprintf(fp, "\nPitch classes of input data:\n");
    FputArray(fp, parray, size);
    PutLn();
    FputLn(fp);
}

/*===== MIC 3.54 =====*/
ShowImbricated(fp, parray, imarray, imsize, size)
FILE *fp;
int parray[], imarray[], imsize, size;
{
    int j, tosize, imbeg;

    if(imsize == 0)
        return;
    for(j = 0, imbeg = 0; j < size - (imsize - 1); j++, imbeg += imsize)
        imbricate(parray, imarray, j, imsize, imbeg);

    tosize = imsize * (size - (imsize - 1));
    printf("\nImbricated array (in blocks of %d)\n", imsize);
    PutArray(imarray, tosize);
    printf("\n");
    fprintf(fp, "\nImbricated array (in blocks of %d)\n", imsize);
    FputArray(fp, imarray, tosize);
    fprintf(fp, "\n");
    PutLn();
}

```

```

    FputcLn(fp);
    fprintf(fp, "\nImbricated array as pitches\n");
    printf("\nImbricated array as pitches\n");
    ScriptArray(imarray, ' ', tosize);
    FscriptArray(fp, imarray, ' ', tosize);
    PutLn();
    FputcLn(fp);
}
/*===== MIC 3.55 =====*/
NormalOrder(parray, ordered, setsize)
    int parray[], ordered[], setsize;
{
    int j, min, temp, test, t1, k, norm;
    int temparray[6], ordered2[6];

    t1 = 0;
    test = 0; /* to see if requirement 2 is needed */
    for(j = 0; j < setsize; j++)
    {
        ordered[j] = parray[j];
        temparray[j] = parray[j];
    }
    j = 0;
    min = abs(ordered[0] - ordered[setsize-1]);
    do {
        rotate(ordered, ordered, setsize);
        temp = abs(ordered[0] - ordered[setsize-1]);
        if(temp < min)
        {
            min = temp;
            for(k = 0; k < setsize; k++)
                temparray[k] = ordered[k];
        }
        else if (temp == min)
        {
            test = 1;
            for(k = 0; k < setsize; k++)
                ordered2[k] = ordered[k];
            break;
        }
        j++;
    } while (j < setsize-1 && test == 0);
    if(min < temp)
    {
        rotate(ordered, ordered, setsize);
        rotate(temparray, temparray, setsize);
    }
    if(test == 1) /* compare temparray with ordered */
    {
        for(j = 0; j < setsize-1; j++)
        {
            if(abs(ordered2[j+1] - ordered2[j]) >
                abs(temparray[j+1] - temparray[j]))
            {
                t1 = 1;
                break; /* use ordered */
            }
        }
        if(t1 == 0)
        {
            norm = ordered2[0];
        }
    }
}

```

```

        for(j = 0; j < setsize; j++)
            ordered[j] = ordered2[j] - norm;
        return;
    }
    else
    {
        norm = temparray[0];
        for(j = 0; j < setsize; j++)
            ordered[j] = abs(temparray[j] - norm);
        return;
    }
}

/*===== MIC 3.56 =====*/
endvector(vector, size)
int vector[], size;
{
    int j, temp=0;
    for(j = 0; j < size-1; j++)
        temp += vector[j];
    vector[size-1] = 12 - temp;
}

/*===== MIC 3.57 =====*/
rotate(datarray, disarray, size)
int datarray[];
int disarray[];
int size;
{
    int j;

    disarray[size] = datarray[0] + 12;
    for(j = 0; j < size; j++)
        disarray[j] = datarray[j+1];
}

/*===== MIC 3.58 =====*/
set3cmp(test)
int test[];
{
    int j = 0;
    int k = 0;
    int setname = 0;

    for(j = 0; j < 12; j++)
    {
        if(
            (test[k] == set3[j].set_array[k]) &&
            (test[k+1] == set3[j].set_array[k+1])
        )
        {
            setname = j + 1;
            return(setname);
        }
    }
    return(0); /* not found */
}

/*===== MIC 3.59 =====*/
set4cmp(test)
int test[];
{
    int j = 0;
    int k = 0;

```

```

int setname = 0;
for(j = 0; j < 29; j++)
{
    if(
        (test[k] == set4[j].set_array[k]) &&
        (test[k+1] == set4[j].set_array[k+1]) &&
        (test[k+2] == set4[j].set_array[k+2])
    )
    {
        setname = j + 1;
        return(setname);
    }
}
return(0); /* not found */
}
/*===== MIC 3.60 =====*/
set5cmp(test)
int test[];
{
    int j = 0;
    int k = 0;
    int setname = 0;

    for(j = 0; j < 38; j++)
    {
        if(
            (test[k] == set5[j].set_array[k]) &&
            (test[k+1] == set5[j].set_array[k+1]) &&
            (test[k+2] == set5[j].set_array[k+2]) &&
            (test[k+3] == set5[j].set_array[k+3])
        )
        {
            setname = j + 1;
            return(setname);
        }
    }
    return(0); /* not found */
}
/*===== MIC 3.61 =====*/
set6cmp(test)
int test[];
{
    int j = 0;
    int k = 0;
    int setname = 0;

    for(j = 0; j < 50; j++)
    {
        if(
            (test[k] == set6[j].set_array[k]) &&
            (test[k+1] == set6[j].set_array[k+1]) &&
            (test[k+2] == set6[j].set_array[k+2]) &&
            (test[k+3] == set6[j].set_array[k+3]) &&
            (test[k+4] == set6[j].set_array[k+4])
        )
        {
            setname = j + 1;
            return(setname);
        }
    }
}

```

```

    return(0); /* not found */
}
/*===== MIC 3.62 =====*/
PrintSetTable(set, setsize)
    struct sets set[];
    int setsize;
{
    register int j = 0;
    int k;

    for(j = 0; j < amt[setsize - 3]; j++)
    {
        k = 0;
        printf("\nSet %d-%d", setsize, j+1);

        printf("\nSet array:\t ");
        for(k = 0; k < setsize; k++)
            printf("%d ", set[j].set_array[k]);

        printf("\nInversion:\t ");
        if(set[j].array_inversion[0] != 0)
            for(k = 0; k < setsize; k++)
                printf("%d ", set[j].array_inversion[k]);

        printf("\nVector: \t ");
        for(k = 0; k < 6; k++)
            printf("%d ", set[j].icvector[k]);

        PutSubset(j, setsize);
        PutSimilar(j, setsize);
        PutContains(j, setsize);
    }
}
/*===== MIC 3.63 =====*/
PrintSetTable(fp, set, setsize)
    FILE *fp;
    struct sets set[];
    int setsize;
{
    register int j = 0;
    int k;

    for(j = 0; j < amt[setsize - 3]; j++)
    {
        k = 0;
        fprintf(fp, "\nSet %d-%d", setsize, j+1);

        fprintf(fp, "\nSet array:\t ");
        for(k = 0; k < setsize; k++)
            fprintf(fp, "%d ", set[j].set_array[k]);

        fprintf(fp, "\nInversion:\t ");
        if(set[j].array_inversion[0] != 0)
            for(k = 0; k < setsize; k++)
                fprintf(fp, "%d ", set[j].array_inversion[k]);

        fprintf(fp, "\nVector: \t ");
        for(k = 0; k < 6; k++)
            fprintf(fp, "%d ", set[j].icvector[k]);

        fputSubset(fp, j, setsize);
    }
}

```

```

    }
}
/*===== MIC 3.64 =====*/
PutSubset(setnumber, setsize)
int setnumber, setsize;
{
    int k = 0;
    int subset;

    if(setsize == 6)
        return;
    setsize += 1;
    printf("\nIs a subset of: ");
    switch(setsize - 1)
    {
        case 3 : while(subset3of[setnumber][k] != 0 &&
                        k <= 9) /* 0 = flag */
        {
            subset = subset3of[setnumber][k++];
            printf("\n\t %d-%d\t", setsize, subset);
            PutSetArray(set4[subset-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set4[subset-1].set_array,
                               setsize);
            printf("\n\t\t");
            PutSetInversion(set4[subset-1].array_inversion,
                             setsize);
            printf("\t");
            PutSetInversionPitches(set4[subset-1].array_inversion,
                                   setsize);
        };
        printf("\n");
        break;
        case 4 : while(subset4of[setnumber][k] != 0 && k <= 9)
        {
            subset = subset4of[setnumber][k++];
            printf("\n\t %d-%d\t", setsize, subset);
            PutSetArray(set5[subset-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set5[subset-1].set_array,
                               setsize);
            printf("\n\t\t");
            PutSetInversion(set5[subset-1].array_inversion,
                             setsize);
            printf("\t");
            PutSetInversionPitches(set5[subset-1].array_inversion,
                                   setsize);
        }
        printf("\n");
        break;
        case 5 : while(subset5of[setnumber][k] != 0 && k <= 9)
        {
            subset = subset5of[setnumber][k++];
            printf("\n\t %d-%d\t", setsize, subset);
            PutSetArray(set6[subset-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set6[subset-1].set_array, setsize);
            printf("\n\t\t");
            PutSetInversion(set6[subset-1].array_inversion,
                             setsize);
            printf("\t");
        }
    }
}

```

```

        PutSetInversionPitches(set6[subset-1].array_inversion,
                               setsize);
    };
    printf("\n");
    break;
}
)
/***** MIC 1.65 *****/
FputSubset(fp, setnumber, setsize)
FILE *fp;
int setnumber, setsize;
{
    int k = 0;
    int subset;

    if(setsize == 6)
        return;
    setsize += 1;
    fprintf(fp, "\nIs a subset of: ");
    switch(setsize-1)
    {
        case 3 : while(subset3of(setnumber)[k] != 0 && k <= 9)
            {
                subset = subset3of(setnumber)[k++];
                fprintf(fp, "\n\t %d-%d\t", setsize, subset);
                FputSetArray(fp, set4[subset-1].set_array, setsize);
                fprintf(fp, "\t");
                FputSetArrayPitches(fp, set4[subset-1].set_array,
                                    setsize);
                fprintf(fp, "\n\t\t");
                FputSetInversion(fp, set4[subset-1].array_inversion,
                                setsize);
                fprintf(fp, "\t");
                FputSetInversionPitches(fp, set4[subset-1].array_inversion,
                                        setsize);
            };
            fprintf(fp, "\n");
            break;
        case 4 : while(subset4of(setnumber)[k] != 0 && k <= 9)
            {
                subset = subset4of(setnumber)[k++];
                fprintf(fp, "\n\t %d-%d\t", setsize, subset);
                fprintf(fp, "\t");
                FputSetArray(fp, set5[subset-1].set_array, setsize);
                fprintf(fp, "\t");
                FputSetArrayPitches(fp, set5[subset-1].set_array,
                                    setsize);
                fprintf(fp, "\n\t\t");
                FputSetInversion(fp, set5[subset-1].array_inversion,
                                setsize);
                fprintf(fp, "\t");
                FputSetInversionPitches(fp,
                                        set5[subset-1].array_inversion, setsize);
            }
            fprintf(fp, "\n");
            break;
        case 5 : while(subset5of(setnumber)[k] != 0 && k <= 9)
            {
                subset = subset5of(setnumber)[k++];
                fprintf(fp, "\n\t %d-%d\t", setsize, subset);
                FputSetArray(fp, set6[subset-1].set_array, setsize);
            }
    }
}

```



```

        fprintf(fp, "\t");
        FputSetArrayPitches(fp, set6[subset-1].set_array,
                           setsize);
        fprintf(fp, "\n\t\t");
        FputSetInversion(fp, set6[subset-1].array_inversion,
                          setsize);
        fprintf(fp, "\t");
        FputSetInversionPitches(fp, set6[subset-1].array_inversion,
                                setsize);
    };
    fprintf(fp, "\n");
    break;
}
)
/*===== MIC 3.56 =====*/
PutSetArray(SetArray, setsize)
int SetArray[], setsize;
{
    int k;
    for(k = 0; k < setsize; k++)
        printf("%d ", SetArray[k]);
}
/*===== MIC 3.67 =====*/
PutSetInversion(SetInversion, setsize)
int SetInversion[], setsize;
{
    int k;
    if(SetInversion[0] == 0)
    {
        printf("No inversion");
        return;
    }
    else
        for(k = 0; k < setsize; k++)
            printf("%d ", SetInversion[k]);
}
/*===== MIC 3.68 =====*/
PutSetArrayPitches(SetArray, setsize)
int SetArray[], setsize;
{
    int k, pitch = 0;
    PutPitch(pitch);
    for(k = 0; k < setsize; k++)
    {
        pitch += SetArray[k];
        PutPitch(pitch);
    }
}
/*===== MIC 3.69 =====*/
PutSetInversionPitches(SetInversion, setsize)
int SetInversion[], setsize;
{
    int k, pitch = 0;
    if(SetInversion[0] == 0)
        return;
    else
    {
        PutPitch(pitch);
        for(k = 0; k < setsize; k++)
        {

```

```

        pitch += SetInversion[k];
        PutPitch(pitch);
    }
}

/*===== MIC 3.70 =====*/
FputSetArrayPitches(fp, SetArray, setsize)
FILE *fp;
int SetArray[], setsize;
{
    int k, pitch = 0;
    FputPitch(fp, pitch);
    for(k = 0; k < setsize; k++)
    {
        pitch += SetArray[k];
        FputPitch(fp, pitch);
    }
}

/*===== MIC 3.71 =====*/
FputSetInversionPitches(fp, SetInversion, setsize)
FILE *fp;
int SetInversion[], setsize;
{
    int k, pitch = 0;
    if(SetInversion[0] == 0)
        return;
    else
    {
        FputPitch(fp, pitch);
        for(k = 0; k < setsize; k++)
        {
            pitch += SetInversion[k];
            FputPitch(fp, pitch);
        }
    }
}

/*===== MIC 3.72 =====*/
PutSetVector(SetVector)
int SetVector[];
{
    int k;
    for(k = 0; k < 6; k++)
        printf("%d ", SetVector[k]);
}

/*===== MIC 3.73 =====*/
FputSetArray(fp, SetArray, setsize)
FILE *fp;
int SetArray[], setsize;
{
    int k;
    for(k = 0; k < setsize; k++)
        fprintf(fp, "%d ", SetArray[k]);
}

/*===== MIC 3.74 =====*/
FputSetInversion(fp, SetInversion, setsize)
FILE *fp;
int SetInversion[], setsize;
{
    int k;
    if(SetInversion[0] == 0)
    {

```

```

        fprintf(fp, "No inversion");
        return;
    }
    else
        for(k = 0; k < setsize; k++)
            fprintf(fp, "%d ", SetInversion(k));
    }
    /*===== MIC 3.75 =====*/
    FputSetVector(fp, SetVector)
    FILE *fp;
    int SetVector[];
    {
        int k;
        for(k = 0; k < 6; k++)
            fprintf(fp, "%d ", SetVector[k]);
    }
    /*===== MIC 3.76 =====*/
    PrintSetInfo(set, setnumber, setsize)
    struct sets set[];
    int setnumber, setsize;
    {
        int j;

        j = setnumber - 1;
        setnumber -= 1;
        printf("\nSet info:");
        printf("\nSet array:\t ");
        PutSetArray(set[j].set_array, setsize);
        printf("\nInversion:\t ");
        PutSetInversion(set[j].array_inversion, setsize);
        printf("\nVector: \t ");
        PutSetVector(set[j].icvector);
        PutSubset(j, setsize);
        PutSimilar(j, setsize);
        PutContains(j, setsize);
    }
    /*===== MIC 3.77 =====*/
    FprintSetInfo(fp, set, setnumber, setsize)
    FILE *fp;
    struct sets set[];
    int setnumber, setsize;
    {
        int j;
        j = setnumber - 1;

        fprintf(fp, "\nSet info:");
        fprintf(fp, "\nSet array:\t ");
        FputSetArray(fp, set[j].set_array, setsize);
        fprintf(fp, "\nInversion:\t ");
        FputSetInversion(fp, set[j].array_inversion, setsize);
        fprintf(fp, "\nVector: \t ");
        FputSetVector(fp, set[j].icvector);
        FputSubset(fp, j, setsize);
        FputSimilar(fp, j, setsize);
        FputContains(fp, j, setsize);
    }
    /*===== MIC 3.78 =====*/
    PutContains(setnumber, setsize)
    int setnumber, setsize;
    {
        int k = 0;

```

```

int contained;

if (setsize == 3)
    return;
printf("Contains: ");
switch(setsize)
{
    case 4 : while(contains3[setnumber][k] != 0 && k <= 11)
        {
            contained = contains3[setnumber][k++];

            printf("\n\t %d-%d\t", setsize - 1, contained);
            PutSetArray(set3[contained-1].set_array, 3);
            printf("\t ");
            PutSetArrayPitches(set3[contained-1].set_array, 3);
            printf("\n\t\t");
            PutSetInversion(set3[contained-1].array_inversion, 3);
            printf("\t ");
            PutSetInversionPitches(set3[contained-1].array_inversion, 3);
        };
    printf("\n")
    break;
    case 5 : while(contains4[setnumber][k] != 0 && k <= 11)
        {
            contained = contains4[setnumber][k++];
            printf("\n\t %d-%d\t", setsize - 1, contained);
            PutSetArray(set4[contained-1].set_array, 4);
            printf("\t");
            PutSetArrayPitches(set4[contained-1].set_array, 4);
            printf("\n\t\t");
            PutSetInversion(set4[contained-1].array_inversion, 4);
            printf("\t");
            PutSetInversionPitches(set4[contained-1].array_inversion, 4);
        };
    printf("\n")
    break;
    case 6 : while(contains5[setnumber][k] != 0 && k <= 11)
        {
            contained = contains5[setnumber][k++];
            printf("\n\t %d-%d\t", setsize - 1, contained);
            PutSetArray(set5[contained-1].set_array, 5);
            printf("\t");
            PutSetArrayPitches(set5[contained-1].set_array, 5);
            printf("\n\t\t");
            PutSetInversion(set5[contained-1].array_inversion, 5);
            printf("\t");
            PutSetInversionPitches(set5[contained-1].array_inversion, 5);
        };
    printf("\n")
    break;
}

/*===== MIC 3.79. =====*/
PutSimilar(setnumber, setsize);
int setnumber, setsize;
{
    int k = 0;
    int similar;
    if (setsize == 3)
        return;

```

```

printf("\nis similar to: ");
switch(setsize)
{
    case 4 : while(SimilarTo4[setnumber][k] != 0 &&
                  k <= 12)
        {
            similar = SimilarTo4[setnumber][k++];
            printf("\n\t %d-%d\t", setsize , similar);
            PutSetArray(set4[similar-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set4[similar-1].set_array, setsize);
            printf("\n\t\t");
            PutSetInversion(set4[similar-1].array_inversion, setsize);
            printf("\t");
            PutSetInversionPitches(set4[similar-1].array_inversion,
                                  setsize);
        };
        printf("\n");
        break;
    case 5 : while(SimilarTo5[setnumber][k] != 0 && k <= 12)
        {
            similar = SimilarTo5[setnumber][k++];
            printf("\n\t %d-%d\t", setsize , similar);
            PutSetArray(set5[similar-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set5[similar-1].set_array,
                              setsize);
            printf("\n\t\t");
            PutSetInversion(set5[similar-1].array_inversion,
                              setsize);
            printf("\t");
            PutSetInversionPitches(set5[similar-1].array_inversion,
                                  setsize);
        };
        printf("\n");
        break;
    case 6 : while(SimilarTo6[setnumber][k] != 0 && k <= 15)
        {
            similar = SimilarTo6[setnumber][k++];
            printf("\n\t %d-%d\t", setsize , similar);
            PutSetArray(set6[similar-1].set_array, setsize);
            printf("\t");
            PutSetArrayPitches(set6[similar-1].set_array,
                              setsize);
            printf("\n\t\t");
            PutSetInversion(set6[similar-1].array_inversion,
                              setsize);
            printf("\t");
            PutSetInversionPitches(set6[similar-1].array_inversion,
                                  setsize);
        };
        printf("\n");
        break;
}

/*===== MIC 3.80 =====*/
FputcContains(fp, setnumber, setsize)
FILE *fp;
int setnumber, setsize;
{
    int k = 0;

```

```

int contained;

if (setsize == 3)
    return;
fprintf(fp, "Contains: ");
switch(setsize)
{
    case 4 : while(contains3[setnumber][k] != 0 && k <= 11)
        {
            contained = contains3[setnumber][k++];
            fprintf(fp, "\n\t %d-%d\t", setsize - 1, contained);
            FputSetArray(fp, set3[contained-1].set_array, 3);
            fprintf(fp, "\t ");
            FputSetArrayPitches(fp, set3[contained-1].set_array, 3);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                set3[contained-1].array_inversion, 3);
            fprintf(fp, "\t\t");
            FputSetInversionPitches(fp,
                set3[contained-1].array_inversion, 3);
        };
        fprintf(fp, "\n");
        break;
    case 5 : while(contains4[setnumber][k] != 0 && k <= 11)
        {
            contained = contains4[setnumber][k++];
            fprintf(fp, "\n\t %d-%d\t", setsize - 1, contained);
            FputSetArray(fp, set4[contained-1].set_array, 4);
            fprintf(fp, "\t");
            FputSetArrayPitches(fp, set4[contained-1].set_array, 4);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                set4[contained-1].array_inversion, 4);
            fprintf(fp, "\t");
            FputSetInversionPitches(fp,
                set4[contained-1].array_inversion, 4);
        };
        fprintf(fp, "\n");
        break;
    case 6 : while(contains5[setnumber][k] != 0 && k <= 11)
        {
            contained = contains5[setnumber][k++];
            fprintf(fp,
                "\n\t %d-%d\t", setsize - 1, contained);
            FputSetArray(fp, set5[contained-1].set_array, 5);
            fprintf(fp, "\t");
            FputSetArrayPitches(fp,
                set5[contained-1].set_array, 5);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                set5[contained-1].array_inversion, 5);
            fprintf(fp, "\t");
            FputSetInversionPitches(fp,
                set5[contained-1].array_inversion, 5);
        };
        fprintf(fp, "\n");
        break;
    }
}
/*===== MIC 3.81 =====*/
FputsSimilar(fp, setnumber, setsize)

```

```

FILE *fp;
int setnumber, setsize;

{
    int k = 0;
    int similar;
    if (setsize == 3)
        return;
    fprintf(fp, "\nIs similar to: ");
    switch(setsize)
    {
        case 4 : while(SimilarTo4[setnumber][k] != 0 &&
                        k <= 12)
        {
            similar = SimilarTo4[setnumber][k++];
            fprintf(fp, "\n\t %d-%d\t", setsize, similar);
            FputSetArray(fp, set4[similar-1].set_array,
                        setsize);
            fprintf(fp, "\t");
            FputSetArrayPitches(fp, set4[similar-1].set_array,
                        setsize);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                        set4[similar-1].array_inversion, setsize);
            fprintf(fp, "\t");
            FputSetInversionPitches(fp,
                        set4[similar-1].array_inversion, setsize);
        };
        fprintf(fp, "\n");
        break;
        case 5 : while(SimilarTo5[setnumber][k] != 0 && k <= 12)
        {
            similar = SimilarTo5[setnumber][k++];
            fprintf(fp, "\n\t %d-%d\t", setsize, similar);
            FputSetArray(fp, set5[similar-1].set_array,
                        setsize);
            fprintf(fp, "\t");
            FputSetArrayPitches(fp, set5[similar-1].set_array,
                        setsize);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                        set5[similar-1].array_inversion, setsize);
            fprintf(fp, "\t");
            FputSetInversionPitches(fp,
                        set5[similar-1].array_inversion, setsize);
        };
        fprintf(fp, "\n");
        break;
        case 6 : while(SimilarTo6[setnumber][k] != 0 && k <= 15)
        {
            similar = SimilarTo6[setnumber][k++];
            fprintf(fp, "\n\t %d-%d\t", setsize, similar);
            FputSetArray(fp, set6[similar-1].set_array,
                        setsize);
            fprintf(fp, "\t");
            FputSetArrayPitches(fp, set6[similar-1].set_array,
                        setsize);
            fprintf(fp, "\n\t\t");
            FputSetInversion(fp,
                        set6[similar-1].array_inversion, setsize);
            fprintf(fp, "\t");
            FputSetInversionPitches(fp,

```

```

        set6[similar-1].array_inversion, setsize);
    );
    fprintf(fp, "\n");
    break;
)
} /* END OF MAIN */
/*=====*/

```

### 程序输出实例(sets.exe)

```

no imbrication, 5 note sets

Input pitches

D#0 E0 C0 C#0 D0 C#0 D0 D#0 F0 C0 D0 E0 F0 C0 C#0
/*-----*/

Pitch classes of input data:
3 4 0 1 2 1 2 3 5 0 2 4 5 0 1
/*-----*/
/*-----*/

Original pitches:
D#0 E0 C0 C#0 D0
Pc array:          3 4 0 1 2
Ordered pc array:  0 1 2 3 4
As pitches:
C0 C#0 D0 D#0 E0
Ordered interval array: 1 1 1 1 8
Set = 5-1

Set info:
Set array:      1 1 1 1 8
Inversion:      No inversion
Vector:         4 3 2 1 0 0
Is a subset of:
6-1  1 1 1 1 1 7      C0 C#0 D0 D#0 E0 F0 C1
      No inversion
6-2  1 1 1 1 2 6      C0 C#0 D0 D#0 E0 F#0 C1
      2 1 1 1 1 6      C0 D0 D#0 E0 F0 F#0 C1

Is similar to:
5-2  1 1 1 2 7      C0 C#0 D0 D#0 F0 C1
      2 1 1 1 7      C0 D0 D#0 E0 F0 C1

Contains:
4-1  1 1 1 9      C0 C#0 D0 D#0 C1
      No inversion
4-2  1 1 2 8      C0 C#0 D0 E0 C1
      2 1 1 8      C0 D0 D#0 E0 C1
4-3  1 2 1 8      C0 C#0 D#0 E0 C1
      No inversion
/*-----*/

Original pitches:
C#0 D0 D#0 F0 C0

```



Pc array: 1 2 3 5 0  
 Ordered pc array: 0 1 2 3 5  
 As pitches:  
 C0 C#0 D0 D#0 F0

Ordered interval array: 1 1 1 2 7  
 Set = 5-2

Set info:  
 Set array: 1 1 1 2 7  
 Inversion: 2 1 1 1 7  
 Vector: 3 3 2 1 1 0

Is a subset of:

6-1	1 1 1 1 1 7	C0	C#0	D0	D#0	E0	F0	C1
	No inversion							
6-2	1 1 1 1 2 6	C0	C#0	D0	D#0	E0	F#0	C1
	2 1 1 1 1 6	C0	D0	D#0	E0	F0	F#0	C1
6-8	2 1 1 1 2 5	C0	D0	D#0	E0	F0	G0	C1
	No inversion							
6-9	1 1 1 2 2 5	C0	C#0	D0	D#0	F0	G0	C1
	2 2 1 1 1 5	C0	D0	E0	F0	F#0	G0	C1

Is similar to:

5-1	1 1 1 1 8	C0	C#0	D0	D#0	E0	C1
	No inversion						
5-3	1 1 2 1 7	C0	C#0	D0	E0	F0	C1
	1 2 1 1 7	C0	C#0	D#0	E0	F0	C1
5-4	1 1 1 3 6	C0	C#0	D0	D#0	F#0	C1
	3 1 1 1 6	C0	D#0	E0	F0	F#0	C1
5-23	2 1 2 2 5	C0	D0	D#0	F0	G0	C1
	2 2 1 2 5	C0	D0	E0	F0	G0	C1

Contains:

4-1	1 1 1 9	C0	C#0	D0	D#0	C1
	No inversion					
4-2	1 1 2 8	C0	C#0	D0	E0	C1
	2 1 1 8	C0	D0	D#0	E0	C1
4-4	1 1 3 7	C0	C#0	D0	F0	C1
	3 1 1 7	C0	D#0	E0	F0	C1
4-10	2 1 2 7	C0	D0	D#0	F0	C1
	No inversion					
4-11	1 2 2 7	C0	C#0	D#0	F0	C1
	2 2 1 7	C0	D0	E0	F0	C1

/\*-----\*/

Original pitches:

D0 E0 F0 C0 C#0  
 Pc array: 2 4 5 0 1  
 Ordered pc array: 0 1 2 4 5  
 As pitches:  
 C0 C#0 D0 E0 F0  
 Ordered interval array: 1 1 2 1 7  
 Set = 5-3

Set info:  
 Set array: 1 1 2 1 7  
 Inversion: 1 2 1 1 7  
 Vector: 3 2 2 2 1 0

Is a subset of:

6-1	1 1 1 1 1 7	C0	C#0	D0	D#0	E0	F0	C1
	No inversion							
6-14	1 2 1 1 3 4	C0	C#0	D#0	E0	F0	G#0	C1

	3 1 1 2 1 4	C0	D#0	E0	F0	G0	G#0	C1
6-15	1 1 2 1 3 4	C0	C#0	D0	E0	F0	G#0	C1
	3 1 2 1 1 4	C0	D#0	E0	F#0	G0	G#0	C1

Is similar to:

5-2	1 1 1 2 7	C0	C#0	D0	D#0	F0	C1
	2 1 1 1 7	C0	D0	D#0	E0	F0	C1
5-4	1 1 1 3 6	C0	C#0	D0	D#0	F#0	C1
	3 1 1 1 6	C0	D#0	E0	F0	F#0	C1
5-11	2 1 1 3 5	C0	D0	D#0	E0	G0	C1
	3 1 1 2 5	C0	D#0	E0	F0	G0	C1
5-27	1 2 2 3 4	C0	C#0	D#0	F0	G#0	C1
	2 2 1 4 3	C0	D0	E0	F0	A0	C1

Contains:

4-2	1 1 2 8	C0	C#0	D0	E0	C1
	2 1 1 8	C0	D0	D#0	E0	C1
4-3	1 2 1 8	C0	C#0	D#0	E0	C1
	No inversion					
4-4	1 1 3 7	C0	C#0	D0	F0	C1
	3 1 1 7	C0	D#0	E0	F0	C1
4-7	1 3 1 7	C0	C#0	E0	F0	C1
	No inversion					
4-11	1 2 2 7	C0	C#0	D#0	F0	C1
	2 2 1 7	C0	D0	E0	F0	C1

/\*-----\*/

Set summary:

Set 5-1 occurs 1 time  
Set 5-2 occurs 1 time  
Set 5-3 occurs 1 time  
There were 0 non-sets

/\*-----\*/

## 函数组: Constel.c

1. con() MIC3.82
2. FindFreq() MIC3.83
3. GetPitchStats() MIC3.84
4. FusionMain() MIC3.85
5. PutFusPitch() MIC3.86

## 功能

音程分析。

## 注释

本函数对某一给定音列进行基本统计分析。函数对线性音程序列进行抽数并列出音符、音程出现频率表。程序的末尾部分将相邻音程混和以便展示部分音列中音程或结构音程的结构。混音程度分几级，从最简单的双音混和到三音混和(如果音列足够长的话)。

## 注意

不要忘记查看 include 指令引入了哪些附加的头文件(扩展名为.c,.h,.pro 等)。

## 编程提示

1. 加一段程序以实现用图表显示统计数据。
2. 将混音函数以数据生成方式而不是数据分析方式应用。

## 程序清单

```
/* CONSTELL.C (Intervallic Analysis Program) */

#include <stdio.h>
#include <ctype.h>
#define MAXDATA 500
#include "array.c"
#include "synclavie.c"
#include "statfunc.c"
#include "inputyou.c"
#include "matrix.c"
#include "screen.c"
#include "getnum.c"
#include "fusarray.c"

struct pitchrec
```

```

        int pc;
        int freq;
    } pit[MAXDATA], regist[MAXDATA], odat[MAXDATA];

#include "constel.pro"
/* ARRAYS */
int datarray[MAXDATA];
int statarray[MAXDATA];
int constellation[MAXDATA];
int fusarray[MAXDATA][2];
int fufreq[MAXDATA];
int pccarray[MAXDATA];
int imarray[MAXDATA];
int sorted[MAXDATA];
int pcsort[MAXDATA];
int shapearray[MAXDATA];

int size, i, consize, fusize, j, s;

FILE *inputfile;
FILE *outputfile;
main()
{
    outputfile = fopen("const.dat", "w");
    size = InputYourOwn(datarray);
    con();
    FusionMain();
    fclose(outputfile);
    printf("\nAnalysis is in file: const.dat\n");
}
/***** MIC 3.82 *****/
/*****/
con()
{
    register int j = 0;
    register int x = 0;
    int form, start, last, incr, nt, trans;

    Cls();
    PutLn();
    fprintf(outputfile, "\n\tConstellation Analysis \n\n");
    FputLn(outputfile);
    fprintf(outputfile, "\n\nInput data :\n");
    FscriptArray(outputfile, datarray, ' ', size);
    FputLn(outputfile);
    fprintf(outputfile, "\nInput data as integers :\n");
    FputArray(outputfile, datarray, size);
    FputLn(outputfile);
    fprintf(outputfile, "\nLinear interval sequence:\n");
    getshape(datarray, shapearray, size);
    FputArray(outputfile, shapearray, size-1);
    printf("\nHow many transpositions do you want (0-12)\n");
    scanf("%d", &nt);
    for(trans = 0; trans <= nt; trans++)
    {
        if(trans != 0)
        {
            PutLn();
            FputLn(outputfile);
            printf("\n\nTransposition no. %d \n", trans);
        }
    }
}

```

```

fprintf(outputfile, "\n\nTransposition no. %d \n", trans);
}
for(j = 0; j < size; j++)
{
    pit[j].freq = 0;
    pit[j].pc = 0;
    regist[j].pc = 0;
    regist[j].freq = 0;
    odat[j].pc = 0;
    odat[j].freq = 0;
    pcsort[j] = (datarray[j] + trans) % 12;
    sorted[j] = datarray[j] + trans;
}

QuickSort(sorted, 0, size - 1);
QuickSort(pcsort, 0, size - 1);
k = 0;
for(j=0; j < size; j++)
{
    if(sorted[j] != sorted[j + 1])
        constellation[k++] = sorted[j];
}

consize = k;
if(trans == 0)
{
    fprintf(outputfile, "\n\nThere are %d pitches in the constellation\n", consize);
    printf("\n\nThere are %d pitches in the constellation\n", consize);
}

fprintf(outputfile, "Constellation:\n");
printf("Constellation:\n");
ScriptArray(constellation, ' ', consize);
FscriptArray(outputfile, constellation, ' ', consize);

getshape(constellation, shapearray, consize);

for(j = 0; j < size; j++)
{
    pit[j].pc = pcsort[j];
    regist[j].pc = sorted[j] / 12;
    odat[j].pc = sorted[j];
    starray[j] = datarray[j] + trans;
}

GetPitchStats(outputfile, starray, size);
getshape(constellation, shapearray, consize);
FindFreq(pit, size);
FindFreq(regist, size);
FindFreq(odat, size);
PutLn();
FputLn(outputfile);

printf("\n\npitch class\t\tfrequency\n");
printf("-----\t\t-----\n");
fprintf(outputfile, "\n\npitch class\t\tfrequency\n");
fprintf(outputfile, "-----\t\t-----\n");
for(j=0; j < size; j++)
{
    if(pit[j].pc != pit[j+1].pc)

```

```

        {
            printf("      %d", pit[j].pc);
            printf("\t\t\t %d\n", pit[j].freq);
            fprintf(outputfile, "      %d", pit[j].pc);
            fprintf(outputfile, "\t\t\t %d\n", pit[j].freq);
        }
    }

    PutLn();
    FputLn(outputfile);
    k = consize - 2;
    printf("\n\npitch\t\t\tfrequency\t\t\tinterval\n");
    printf("-----\t\t\t-----\t\t\t-----\n");
    fprintf(outputfile, "\n\npitch\t\t\tfrequency\t\t\tinterval\n");
    fprintf(outputfile, "-----\t\t\t-----\t\t\t-----\n");
    for(j = size; j >= 0; j--)
    {
        if(odat[j].pc != odat[j + 1].pc)
        {
            PutPitch(odat[j].pc);
            FputPitch(outputfile, odat[j].pc);
            printf("\t\t\t %d\n", odat[j].freq);
            fprintf(outputfile, "\t\t\t %d\n", odat[j].freq);
            if(j > 0)
            {
                printf("\t\t\t\t\t %d\n", shapearray[k]);
                fprintf(outputfile, "\t\t\t\t\t %d\n", shapearray[k--]);
            }
        }
    }
    PutLn();
    FputLn(outputfile);
    printf("\n\nregister\t\t\tfrequency\n");
    printf("-----\t\t\t-----\n");
    fprintf(outputfile, "\n\nregister\t\t\tfrequency\n");
    fprintf(outputfile, "-----\t\t\t-----\n");
    for(j=0; j < size; j++)
    {
        if(regist[j].pc != regist[j + 1].pc)
        {
            printf("      %d", regist[j].pc);
            printf("\t\t\t %d\n", regist[j].freq);
            fprintf(outputfile, "      %d", regist[j].pc);
            fprintf(outputfile, "\t\t\t %d\n", regist[j].freq);
        }
    }
}
}
/*===== MIC 3.83 =====*/
FindFreq(pit, size)
    struct pitchrec pit[];
    int size;

{
    int j, k, key;

    for(j = 0; j < size; j++)
    {
        key = pit[j].pc;
        for(k = 0; k < size; k++)
        {

```

```

        if(key == pit[k].pc)
            pit[k].freq += 1;
    }
}

/*===== MIC 3.84 =====*/
GetPitchStats(outputfile, statarray, size)
    FILE *outputfile;
    int statarray[];
    int size;
{
    float mea, sdev;
    int mod, med, min, max, dev;

    med = median(statarray, size);
    printf("\nmedian = ");
    PutPitch(med);
    fprintf(outputfile, "\nmedian = ");
    FputPitch(outputfile, med);

    mea = mean(statarray, size);
    printf("\nmean = ");
    i = mea; /* cast */
    PutPitch(i);
    fprintf(outputfile, "\nmean = ");
    FputPitch(outputfile, i);
    sdev = StdDev(statarray, size);
    printf("\nstandard deviation from the mean = %3.2f", sdev);
    fprintf(outputfile, "\nstandard deviation from the mean = %3.2f", sd);
    dev = sdev; /* cast */
    printf("\nThat means that the tessitura is from ");
    PutPitch(i - dev / 2);
    printf("to ");
    PutPitch(i + dev / 2);
    printf("\n");
    fprintf(outputfile, "\nThat means that the tessitura is from ");
    FputPitch(outputfile, i - dev / 2);
    fprintf(outputfile, "to ");
    FputPitch(outputfile, i + dev / 2);
    fprintf(outputfile, "\n");

    mod = FindMode(statarray, size);
    printf("\nstatistical mode = ");
    PutPitch(mod);
    fprintf(outputfile, "\nstatistical mode = ");
    FputPitch(outputfile, mod);

    max = getMax(statarray, size);
    printf("\nhighest pitch = ");
    PutPitch(max);
    fprintf(outputfile, "\nhighest pitch = ");
    FputPitch(outputfile, max);

    min = getmin(statarray, size);
    printf("\nlowest pitch = ");
    PutPitch(min);
    fprintf(outputfile, "\nlowest pitch = ");
    FputPitch(outputfile, min);

    regress(statarray, size);

```

```

    )
    /*===== MIC 3.85 =====*/
    FusionMain()
    (
        int last, incr;

        fprintf(outputfile, "\n\tFusions\n");
        FputLn(outputfile);
        fprintf(outputfile, "\n\nInput data :\n");
        FscriptArray(outputfile, datarray, ' ', size);
        FputLn(outputfile);
        fprintf(outputfile, "\nInput data as integers :\n");
        FputArray(outputfile, datarray, size);
        FputLn(outputfile);

        PutLn();
        FputLn(outputfile);
        ZeroMatrix(fusarray, size, 2);
        getshape(datarray, shapearray, size-1);
        fusize = 0;
        {
            fusize = FusePair(outputfile, shapearray, fusarray, size - 1);
            printf("\nInterval pairs\n");
            fprintf(outputfile, "\nInterval pairs\n");
            PutMatrix(fusarray, fusize, 2);
            MatrixSort(fusarray, fusize, 2);
            FindMatrixFreq(fusarray, fusfreq, fusize);
            PutMatrixFq(fusarray, fusfreq, fusize, 2);
            FputMatrixFq(outputfile, fusarray, fusfreq, fusize, 2);
            PutFusPitch(outputfile, fusarray, fusize);
            ZeroMatrix(fusarray, fusize, 2);

        }

        PutLn();
        FputLn(outputfile);

        fusize = fusion(datarray, fusarray, size - 1, 24);
        printf("\nSingle datum with multiple fusions\n");
        printf("upper limit = 24\n");
        fprintf(outputfile, "\nSingle datum with multiple fusions");
        fprintf(outputfile, "\nupperlimit = 24\n");
        PutMatrix(fusarray, fusize, 2);
        MatrixSort(fusarray, fusize, 2);
        FindMatrixFreq(fusarray, fusfreq, fusize);
        PutMatrixFq(fusarray, fusfreq, fusize, 2);
        FputMatrixFq(outputfile, fusarray, fusfreq, fusize, 2);
        PutFusPitch(outputfile, fusarray, fusize);
        ZeroMatrix(fusarray, fusize, 2);

        PutLn();
        FputLn(outputfile);

        fusize = fusion2(datarray, fusarray, size - 1);
        printf("\nDouble fusions\n");
        fprintf(outputfile, "\nDouble fusions\n");
        PutMatrix(fusarray, fusize, 2);
        MatrixSort(fusarray, fusize, 2);
        FindMatrixFreq(fusarray, fusfreq, fusize);
        PutMatrixFq(fusarray, fusfreq, fusize, 2);
        FputMatrixFq(outputfile, fusarray, fusfreq, fusize, 2);
    }

```



```

        PutFusPitch(outputfile, fusarray, fusize);
        ZeroMatrix(fusarray, fusize, 2);

    PutLn();
    FputLn(outputfile);

    fusize = fusion3(dataarray, fusarray, size - 1);
    if(fusize > 0)
    {
        printf("\nTriple fusions\n");
        fprintf(outputfile, "\nTriple fusions\n");
        PutMatrix(fusarray, fusize, 2);
        MatrixSort(fusarray, fusize, 2);
        FindMatrixFreq(fusarray, fusfreq, fusize);
        PutMatrixFq(fusarray, fusfreq, fusize, 2);
        FputMatrixFq(outputfile, fusarray, fusfreq, fusize, 2);
        PutFusPitch(outputfile, fusarray, fusize);
        ZeroMatrix(fusarray, fusize, 2);
    }

    PutLn();
    FputLn(outputfile);
}

/*===== MIC 3.86 =====*/
PutFusPitch(fp, fusarray, size)
FILE *fp;
int fusarray[][2], size;
{
    int pc, j;

    printf("\nAs pitches from C1\n\n");
    fprintf(fp, "\nAs pitches from C1\n\n");
    for(j = 0; j < size; j++)
    {
        pc = 12;
        PutPitch(pc);
        PutPitch(pc += fusarray[j][0]);
        PutPitch(pc += fusarray[j][1]);
        printf("\n");
        pc = 12;
        FputPitch(fp, pc);
        FputPitch(fp, pc += fusarray[j][0]);
        FputPitch(fp, pc += fusarray[j][1]);
        fprintf(fp, "\n");
    }
} /* end of function */

```

### Sample output: constel.c

```

        Constellation Analysis
    /*-----*/

    Input data :

        C0 D#0 E0 C#1 C3 D4 G3

```

```
/*-----*/
```

```
Input data as integers :  
0 3 4 13 36 50 43
```

```
/*-----*/
```

```
Linear interval sequence:  
3 1 9 23 14 -7
```

```
There are 7 pitches in the constellation  
Constellation:
```

```
C0 D#0 E0 C#1 C3 G3 D4
```

```
median = C#1
```

```
mean = A1
```

```
standard deviation from the mean = 19.52
```

```
That means that the tessitura is from C1 to F#2
```

```
statistical mode = C0
```

```
highest pitch = D4
```

```
lowest pitch = C0
```

```
/*-----*/
```

pitch class	frequency
0	2
1	1
2	1
3	1
4	1
7	1

```
/*-----*/
```

pitch	frequency	interval
D4	1	7
G3	1	7
C3	1	23
C#1	1	9
E0	1	1
D#0	1	3
C0	1	

```
/*-----*/
```

register	frequency
0	3
1	1

3	2
4	1

# Fusions

/\*-----\*/

Input data :

C0 D#0 E0 C#1 C3 D4 G3

/\*-----\*/

Input data as integers :

0 3 4 13 36 50 43

/\*-----\*/

/\*-----\*/

# FUSIONS

Intervals to be fused :

3 1 9 23 14 7

Interval pairs

fusion	frequency
1 9	1
3 1	1
9 23	1
14 7	1
23 14	1

As pitches from C1

C1 C#1 A#1  
C1 D#1 E1  
C1 A1 G#3  
C1 D2 A2  
C1 B2 C#4

/\*-----\*/

Single datum with multiple fusions

upperlimit = 24

fusion	frequency
0 7	1
0 20	1
3 17	1
4 3	1
13 7	2
36 17	1
36 20	2

As pitches from C1

C1 C1 G1  
C1 C1 G#2

C1 D#1 G#2  
 C1 E1 G1  
 C1 C#2 G#2  
 C1 C#2 G#2  
 C1 C4 F5  
 C1 C4 G#5  
 C1 C4 G#5

/\*-----\*/

#### Double fusions

fusion		frequency
-----		-----
3	17	1
7	49	1
17	86	1

#### As pitches from C1

C1 D#1 G#2  
 C1 G1 G#5  
 C1 F2 R

/\*-----\*/

#### Triple fusions

fusion		frequency
-----		-----
0	20	1
3	53	1
7	13	1
20	36	1

#### As pitches from C1

C1 C1 G#2  
 C1 D#1 G#5  
 C1 G1 G#2  
 C1 G#2 G#5

/\*-----\*/

## 第四章 概率分布函数

许多作曲算法要求测试主机中均匀分布随机数据生成程序生成的大量数据。因为按一定句法规则作曲常常需要通过取样(测试)或删除(保留)的方式调用整型化 RND 函数,这一过程实际上不是从一系列数据中筛选掉所有不符合某一给定规则的数。通常,这种借助于机器内部的均匀分布算法谱写出的音乐结构至多只能算是可用,很难达到最佳艺术效果。

相反,非均匀分布算法是一种很重要的作曲方法的基础。这种方法直接计算出所音列的各种音乐参数,而不必遵循那些因不同音乐参量而异的句法规则。其目的是为了能够将函数返回的一条概率分布特性曲线应用于多种音乐参量。应用时只需将返回数据进行恰当定标,无需太多的前后、上下关系就可以达到符合音乐参数取值范围的要求。本章虽然不可能将所有此类算法一一列出,但已举出的各例已足以满足大多数作曲方法的要求。

本章中各函数应用于连续变化或不连续变化的随机变量。随机变量的取值通常根据某一随机事件在一个随机过程中的发生概率大小而定。连续变化的随机变量的取值可无限细分,依机器计算精度而定,而且取从 0 到 1 之间的一个实数值。不连续变化随机变量的取值不可无限细分,通常取整数值。

一个随机变量过程中所有可能发生的随机事件的概率之和必为 1。各随机事件的概率大小通常可用概率分布图来直观表示。连续变化随机变量的概率分布是一条曲线,而非连续变化随机变量的概率分布则类似于直方图,但二者略有不同。直方图记录一段程序运行过程中某一数值的实际出现频率,而概率分布图则表示在一段程序假想运行过程中一系列不连续的数值中出现某一数的概率大小。通过对比二者的不同之处可以验证子程序操作是否正确。这样做还可以搞清取样点的个数与理想概率分布的保真度之间的关系(通常需要先生成大量数据才能绘出特性曲线)。

连续变化随机变量的实型值可以转换为整型值并换算成各种取值范围内的数值。具体方法是将各实型数先与一恰当的换算因子相乘,然后取整。

### 函数组: Bet.c

1. Beta() MIC 4.1
2. Fpower() MIC2.2

### 功能

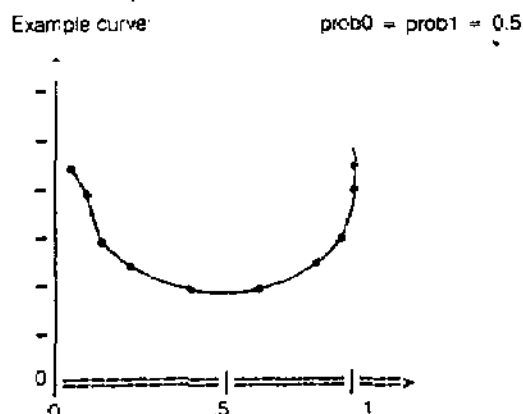
生成一组随机数据构成贝塔分布特性线。

## 注释

本算法的返回值是大于 0、小于或等于 1 的连续、无序实数。通过改变控制变量“prob0”和 prob1 可调整特性曲线的形状，参数 prob0 决定 0 值附近的概率值，参数 prob1 决定 1 值附近的概率值，参数值越小，概率值越大。如果参数 prob0 的值与 prob1 相等，则特性曲线是一条以  $x=0.5$  的直线或对称图形的曲线。若二者的值不相等，则曲线在较小的参数对应的横坐标值附近曲率更大。中值计算公式如下：

$$\text{prob0} / (\text{prob0} + \text{prob1})$$

如果参数 prob0 和 prob1 的值都大于 1，则函数生成的特性曲线将类似于钟形的高斯曲线。还可以将参数 prob0 和 prob1 都置为 1 以生成均匀分布特性曲线。



## 程序清单

```
/* BETA.C (Eulerian Beta Probability Distribution Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int seed = -43;
    int j;          /* loop index */
    float parm0 = .4; /* controls values nearer to 0 */
    float parm1 = .2; /* controls values nearer to 1 */
    double x[100];   /* array storing Beta distribution */
    double Beta();

    srand(seed);
    for (j = 0; j < 100; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Beta(parm0, parm1);
        printf("%f ", x[j]);
    }
    /* End of main */
    /*===== MIC 4.1 =====*/
    /* Beta() Function */
}
```

```

double Beta(prob0,prob1)
double prob0,prob1;
{
    double u1;      /* random real number 1 ( > 0 < 1.) */
    double u2;      /* random real number 2 ( > 0 < 1.) */
    double t1;      /* computed probability 1 */
    double t2;      /* computed probability 2 */
    double sum;     /* total of probabilities 1 and 2 */
    double result;  /* final Beta value */
    double Fpower();

    prob0 = 1. / prob0;
    prob1 = 1. / prob1;
    do
    {
        u1 = rand() / 32767.;
        u2 = rand() / 32767.;
        t1 = Fpower(u1,prob0);
        t2 = Fpower(u2,prob1);
        sum = t1 + t2;
    }
    while (sum > 1.0);
    result = t1 / sum;
    return result;
} /* end of Beta() function */
/***** MIC 2.2 *****/
/* Fpower() function */
double Fpower(value, tothe)
double value, tothe;
{
    int sign;
    double result;

    sign = (tothe < 0.0) ? -1 : 1;
    tothe = fabs(tothe);
    result = exp( log(value) * tothe);
    if (sign < 0)
        result = 1.0 / result;
    return(result);
} /* end of Fpower() function */
/***** */
/* END OF BETA.C */

```

## 程序运行实例

BETA.EXE

```

0.010990 0.359585 0.999128 0.223860 0.999807
0.704022 0.001031 0.817014 0.999484 0.553148
0.399195 0.752755 0.992644 0.813917 0.964797
0.010715 0.979119 0.813143 0.898935 0.997926
0.014646 0.589775 0.988215 0.936789 0.685888
0.957545 0.961579 0.000084 0.177708 0.880190
0.131819 0.974886 0.063196 0.645864 0.007146
0.982604 0.104106 0.799257 0.057739 0.009879
0.909136 0.183087 0.999559 0.057684 0.950676
0.940050 0.916770 1.000000 0.989211 0.893537
0.107457 0.960971 0.469452 0.553147 0.992217
0.998132 0.993824 0.778849 0.214022 0.999997
0.328246 0.999494 0.159273 0.826219 0.894953

```

```

0.021496 0.164498 0.954909 0.020334 0.489878
0.999218 0.970606 0.841927 0.926379 0.951947
0.289238 0.316517 0.019442 0.467412 0.723952
0.875365 0.432815 0.016803 0.997326 0.963250
0.023208 0.000203 0.999992 0.080765 0.117176
0.999984 0.882339 0.634970 0.045995 0.000184
0.995582 0.000001 0.990730 0.565506 0.065421

```

## 函数: Bilexp() MIC4.2

### 功能

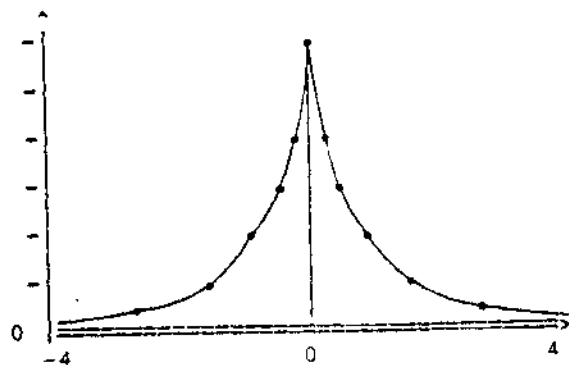
生成一组随机数构成双边指数分布特性曲线。

### 注释

本函数的返回值是一些在以 0 点为中心的连续、随机数。正、负数值按指数坐标分布在 0 点两侧，取值无上、下界。当通过改变参数 spread 的值增大生成数据的取值范围时，根据拉普拉斯第一定律，其曲线幅值减小(幅值必须大于 0)。

Example curve.

spread = 1.5



### 程序清单

```

/* BILEXP.C (Bilexp Probability Distribution Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int total;
    int j;
    int seed = 1231;
    double x[100];
    double parm = .43;
    double Bilexp();

    srand(seed);
    for (j = 0; j < 100; j++)
    {
        if (j % 5 == 0)

```



```

        printf("\n");
        x[j] = Bilexp(para);
        printf("%f ",x[j]);
    }
} /* end of main*/
/***** MIC 4.2 *****/
/* Bilexp function */
double Bilexp(spread)
double spread; /* horizontal scaling parameter */
{
    double u; /* stores random number > 0 & < 1.0 */

    u = (rand() / 32767.) * 2.;
    if (u > 1)
    {
        u = 2. - u;
        u = -log(u);
        u = u / spread;
        return u;
    }
    else
    {
        u = log(u);
        u = u / spread;
        return(u);
    }
} /* end of Bilexp() function */
/*****
/* END OF BILEXP.C */

```

### 程序运行实例

#### BILEXP.EXE

```

-9.167392 3.958152 1.022059 1.434329 5.896396
2.602305 2.699034 0.132822 -1.119226 3.941086
3.685882 4.670400 1.026690 -0.146841 3.263379
7.529966 -1.578920 0.944291 2.946929 1.676083
0.609170 -2.078071 3.870297 0.029786 0.909192
1.703087 3.531165 0.695605 2.618004 0.824504
1.078923 -0.425182 -0.592079 2.712210 0.912972
3.035803 2.853083 4.297001 8.846820 -0.438513
0.172841 -5.431838 1.076441 5.679309 -3.017546
0.509518 3.933370 5.396888 -1.176897 -0.088462
-0.349235 12.001710 -1.333447 2.050826 2.027968
1.243777 -0.324134 3.142967 -4.409566 -1.921140
1.913694 -0.824301 2.136380 -2.165372 -0.952828
-1.011729 3.226710 0.918233 -0.386982 -2.437891
-3.826480 -0.001207 1.592393 -1.965653 -2.653718
6.766148 -0.978414 1.015241 0.798746 -3.436567
-2.990685 -2.170781 5.507898 -1.109370 0.472875
-0.848095 0.851163 4.914167 0.656311 -0.916970
-1.901765 -0.512170 0.138702 -0.202539 0.668011
-0.628061 0.387317 -0.632715 1.519237 1.196048

```

## 函数: Cauchy() MIC4.3

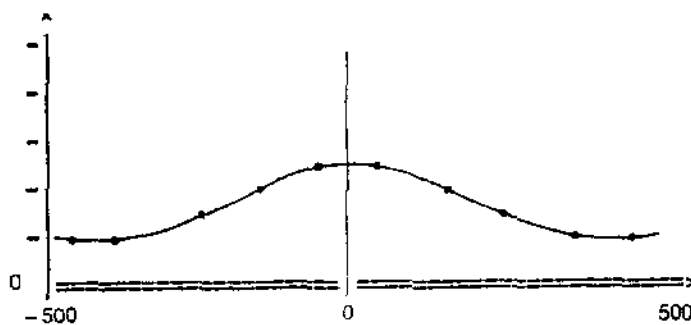
### 功能

生成一组随机数据构成柯西分布特性曲线。

### 注释

本函数的返回值是一些对称分布在中值点两侧的实数。它类似于高斯分布，因为它也是在中值点两侧无上、下界，但此概率曲线在中值点附近较为平缓。这实际上相当于高中值点较远的取样点的概率值比高斯分布上相应点的概率值大。改变控制变量 spread 可调整曲线的取样点在水平方向上的分布。

Example curve: spread = 5



### 程序清单

```
/* CAUCHY.C (Cauchy Probability Distribution Function */
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = -7743;
    float parml = 2.0; /* horizontal scaling factor */
    double x[100], Cauchy();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Cauchy(parml);
        printf("%f ", x[j]);
    }
    /* end of main */
    /*===== MIC 4.3 =====*/
}
```

```

/* Cauchy() function */
double Cauchy(spread)
double spread; /* horizontal scaling factor */
{
    double PI= 3.1415927;
    double u; /* random number > 0 & < 1.0 */
    double result; /* final Cauchy value */

    do {
        u = rand() / 32767.;
        u = u * PI;
        result = spread * tan(u);
    }
    while (u == .5);
    return result;
} /* end of Cauchy() function */
/*=====*/
/* END OF CAUCHY.C */

```

### 程序运行实例

CAUCHY.EXE

```

-1.656243 -3.923872 29.991776 1.649789 -0.193504
-1.220800 1.274069 -0.379132 1.709961 166.208393
-2.871423 4.153552 -0.587291 3.542630 -3.298190
-7.998833 0.614049 0.835203 1.325619 -0.070594
-2.643495 0.673091 -1.684563 9.269187 -1.267878
0.784315 -0.819707 2.680699 0.689140 -0.445706
-1.862281 0.651387 -0.901293 37.114822 -1.486634
-3.061565 -3.951926 -27.344761 1.132015 -7.810809
-0.576683 5.213639 -1.365731 -3.399563 -3.362575
2.484867 -1.282442 -2.418642 -3.585061 0.001142
-0.348426 -0.725384 -0.523466 1.745793 -99.557471
-1.506074 2.471255 -87.448481 4.539351 1.890403
-1.453261 0.171462 1.810333 1.468552 14.542320
1.252875 -0.220444 0.958665 2.047835 1.238769
0.910769 2.536231 0.403222 -5.209158 -10.168957
-0.527566 1.943379 2.570015 0.006136 2.681771
144.351696 0.736905 -1.859777 2.979780 -0.730378
-43.655568 8.373116 11.068622 -7.056781 0.798287
6.920014 0.820603 1.367419 0.501232 -6.877964
0.758542 1.238238 -0.869416 -2.999004 -0.179385

```

### 函数: Expon() MIC4.4

#### 功能

生成一组数据构成单边指数分布特性曲线。

#### 注释

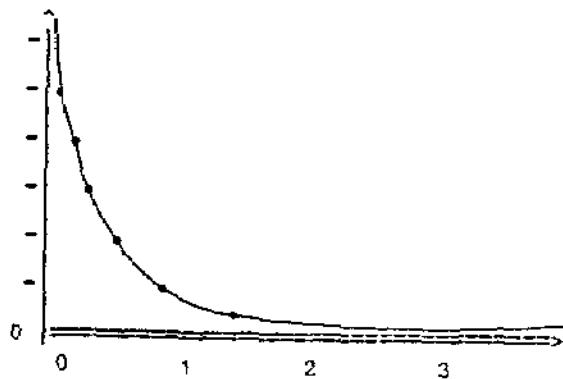
本算法的返回值是一些连续的大于 0 的无序实数。零点附近的取样点的概率值较大，随着取样点的指数坐标值增大，其概率减小。

可以通过改变控制变量“spread”调整曲线的取样点的分布。spread 值越大，返回值较

小的概率也就越大。虽然取样点的横坐标值无上限，但当其横坐标值取得很大时，其出现概率几乎为 0，也即该取样点对应的横坐标值几乎不可能被选入返回数值组中。分布中值的计算公式为  $0.6935 / \text{spread}$ 。

Example curve:

spread = 2.0



### 程序清单

```
/* EXPON.C (Exponential Probability Distribution Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = 1296;
    double x[100];
    double parm = 2.0; /* horizontal scaling factor */
    double Expon();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
        {
            printf("\n");
            x[j] = Expon(parm);
            printf("%f ", x[j]);
        }
    } /* end of main */
    /*===== MIC 4.4 =====*/
    /* Expon() function */
    double Expon(spread)
    double spread; /* horizontal scaling factor */
    {
        double u; /* random number > 0 & < 1.0 */
        double result; /* final Expon value */

        u = rand() / 32767.;
        result = -log(u) / spread;
        return result;
    } /* end of Expon() function */
    /*=====*/
    /* END OF EXPON.C */
}
```

## 程序运行实例

EXPON.EXE

```
0.180739 0.365910 0.794703 1.110815 0.042149
0.091451 0.232745 0.524669 1.125089 1.021958
2.075535 0.025336 0.062012 0.130476 0.874127
0.001268 0.617594 0.033362 0.442071 0.062548
0.160141 0.125176 0.112022 0.006961 0.088674
0.445965 0.551570 0.558935 0.401843 0.223618
0.475922 0.470184 1.118146 0.026509 0.657392
0.075263 0.534616 0.209291 0.740604 0.380785
0.618276 0.134294 0.116595 0.195792 0.224406
0.447681 0.281466 0.420242 0.214841 0.401196
0.003384 0.196018 0.691122 1.125814 0.636633
0.074093 0.109148 0.948260 0.320428 0.848748
0.193922 0.256131 0.222998 0.159805 0.831309
0.472102 0.930189 0.628147 2.353409 0.000779
0.065758 1.706694 0.222021 0.360358 0.060253
0.115490 1.418809 0.044496 0.981031 1.105078
1.229544 0.235206 1.553101 0.086746 0.229161
0.547129 0.110573 0.340763 0.503426 0.316562
0.341669 1.216163 0.046183 1.079452 0.242886
0.327693 0.828420 0.120960 1.543648 0.281788
```

函数: Gamma() MIC4.5

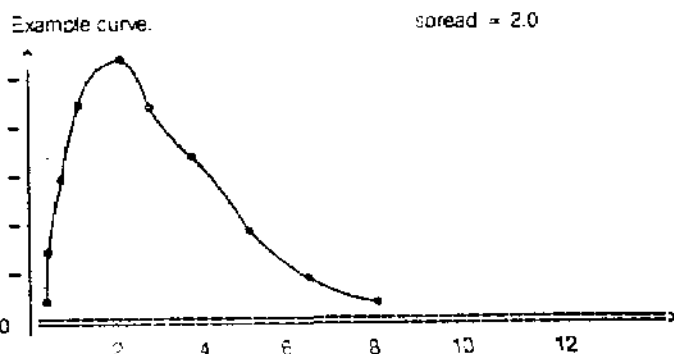
### 功能

生成一组随机数构成伽马分布特性曲线。

### 注释

本算法可返回是对应于一条不对称曲线的连续、无序实数。它常被用于在定音乐节拍时产生一种“自由节奏”效果。可通过改变控制变量 **spread** 的值来显著地改变曲线形状。通常 **spread** 的取值不大于 10, 因为随着 **spread** 的值增大, 曲线越来越接近于高斯分布曲线。

因为有时会出现特殊情况(比如说 **spread** 值的微小变化引起曲线形状突变), 读者最好先将 **spread** 试置为各种不同的值, 然后根据各点实现出现频率值画出条线图。



## 程序清单

```
/* GAMMA.C (Gamma Probability Distribution Function)*/
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = -296;
    double x[100];
    double parm = 5.0; /* governs shape of curve */
    double Gamma();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
        {
            printf("\n");
            x[j] = Gamma(parm);
            printf("%f ", x[j]);
        }
    } /* end of main*/
    /*===== MIC 4.5 =====*/
    /* Gamma function */
    double Gamma(spread)
    double spread; /* governs shape of curve */
    {
        int j;
        double result;
        double u;
        double sum = 1.0;

        for (j = 0; j < spread; j++)
        {
            u = rand() / 32767.;
            sum = sum * u;
        }
        result = -log(sum);
        return result;
    } /* end of Gamma() function */
    /*=====*/
    /* END OF GAMMA.C */
}
```

## 程序运行实例

### PROGRAM EXECUTION:

```
4.848656 5.199660 3.569251 4.148325 3.887295
6.240203 2.291942 8.424481 2.148828 4.512343
4.736967 2.425011 10.381698 5.558272 5.695244
4.181691 10.828009 2.878646 3.905727 6.334919
5.569885 5.586389 5.296786 6.745270 3.607900
6.461507 3.272248 5.986170 5.965548 4.750306
8.802918 4.376763 5.606261 5.760528 5.642366
3.233463 3.460231 3.356096 13.866655 11.360364
5.034211 4.132257 7.248348 3.646231 5.223305
```

```

3.665702 2.708764 5.592431 6.282181 4.858055
6.351133 7.570575 4.328890 3.382579 4.026595
4.739250 8.057736 5.242775 5.275028 4.829192
5.234377 5.567001 13.246912 4.799805 9.367443
1.652347 3.675683 7.101449 8.029843 3.478084
1.757624 3.910659 6.071557 6.232306 5.305021
4.664670 6.978143 1.963472 4.177363 1.742774
2.888380 8.907016 4.846449 3.649949 5.951984
4.907827 6.991382 3.371839 5.651329 5.365534
8.148632 4.952009 4.453359 10.597573 2.977761
1.790738 3.856396 4.101259 5.823742 1.009400

```

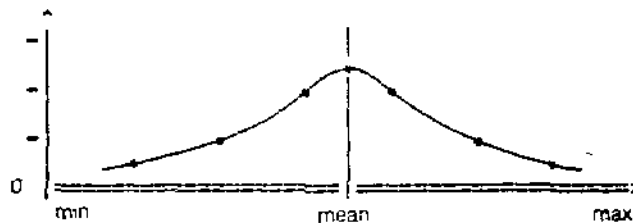
**函数: Gauss() MIC4.6**

**功能**

生成一组随机数据构成高斯分布特性曲线。

**注释**

本概率分布有时也被称为标准概率分布或高斯——接普接斯分布。本算法生成一条钟形曲线，其取样点的横坐标值是以中值为中心点的无序正实数。本分布可以通过对均匀分布随机数求和得到。控制变量是标准偏移 dev 和中值 mean。



**程序清单**

```

/* GAUSS.C (Gaussian Probability Distribution Function)*/
#include <stdio.h>
main()
{
    int j, total = 100, seed = -30261;
    double x[100];
    double parn1 = 2.0; /* controls standard deviation */
    double parn2 = 10.0; /* controls statistical mean */
    double Gauss();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)

```

```

        printf("\n");
        x[j] = Gauss(parm1,parm2);
        printf("%f ",x[j]);
    }
} /* end of main*/
/*===== MIC 4.6 =====*/
/* Gauss() function */
double Gauss(dev,mean)
double dev,mean; /* control std. deviation and stat. mean */
{
    int j;
    int num = 12; /* number of random values used to compute */
    double result; /* final gaussian value */
    double u; /* random number > 0 & < 1.0 */
    double sum = 0.0; /* sum of random numbers */
    double scale = 1.0; /* internal scaling factor */
    double halfnum = num / 2.0;

    for (j = 0; j < num;j++)
    {
        u = rand() / 32767.;
        sum = sum + u;
    }
    result = dev * scale * (sum - halfnum) + mean;
    return result;
} /* end of Gauss() function */
/*=====*/
/* END OF GAUSS.C */

```

#### 程序运行实例

```

12.314890 11.178503 9.634877 9.220191 11.122166
8.813440 11.201880 10.544145 7.378399 10.718406
9.766533 9.392193 11.623035 7.404767 12.876492
10.408704 12.660665 11.900815 10.844020 9.254128
10.561968 8.749046 11.747734 9.604541 7.663137
11.009064 10.431471 12.151799 7.566149 9.254494
9.504807 5.602710 10.138371 10.576189 12.343272
9.680593 12.512589 7.579394 10.087466 12.124088
12.087832 9.385601 8.644002 11.686636 9.528184
10.631855 7.684927 13.138890 7.763482 8.470412
9.648222 12.265572 12.989715 9.569933 8.736778
8.868007 11.667776 11.113926 9.920225 11.762871
7.524155 7.801813 11.932737 11.782220 10.551958
10.849147 10.274422 10.774316 8.008606 8.986236
10.211615 10.602374 11.418500 9.459456 7.775445
8.145451 11.570360 13.030915 12.763390 12.298898
11.519150 8.978057 7.737785 8.972320 9.585559
10.093875 9.027192 14.092532 8.355785 9.687735
7.725517 8.600543 11.681753 8.583819 12.692770
11.454146 11.680715 10.122257 9.553697 6.625690

```

#### 函数: Hypcos() MIC4.7

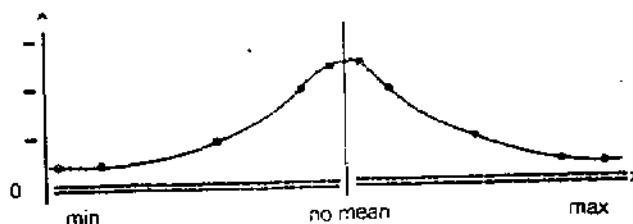


## 功能

生成随机数据构成双曲线型余弦分布特性曲线。

## 注释

本算法生成一条对称曲线，其取样点的横坐标是一些无序、连续的实数(包括正、负实数)，虽然取样点分布以零点为中心，本曲线没有中值点。



## 程序清单

```
118 1-10
/* Hypcos.c (Hyperbolic Cosine Prob. Distr. Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = 602;
    double x[100], Hypcos();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Hypcos();
        printf("%f ", x[j]);
    }
    /* end of main */
    /*===== MIC 4.7 =====*/
    /* Hypcos() function */
    double Hypcos()
    {
        double PI = 3.1415927;
        double result;
        double u;

        u = rand() / 32767.;
        result = log(tan(PI * u / 2.0));
        return result;
    }
    /* end of Hypcos() function */
    /*=====*/
    /* END OF HYPCOS.C */
}
```

## 程序运行实例

HYPCOS.EXE

```
-0.447231 -2.572166 -0.420027 0.345758 -1.470294
2.936152 0.727509 -1.509629 2.373300 2.281548
0.118348 -0.614872 0.280250 1.323721 -1.892836
1.298349 -0.068269 0.444909 0.291530 -1.382082
1.333989 -0.947991 -0.242169 -2.306029 2.470149
1.679118 -0.950269 -0.702250 -1.222725 3.248058
1.628220 -0.402434 -1.260111 -0.351151 1.598450
-2.672303 3.205711 -1.852299 -0.061352 -0.993282
0.081634 -0.782678 2.320216 -2.191268 -0.516507
0.337636 5.456957 4.408213 0.581376 -2.312364
-0.538412 -1.208829 -2.419484 1.055535 2.295396
0.287633 0.131779 -0.435220 -3.175371 1.022303
0.016156 -1.158313 0.635731 0.756782 -0.583516
-1.519456 0.733271 1.784108 1.133905 -0.995934
1.111261 0.837414 0.562116 0.272882 0.603109
1.737879 -3.736815 -0.512806 -0.561559 -1.382488
3.085284 0.673695 0.787758 0.938209 0.077402
1.989971 -0.312087 0.890590 1.006447 -0.320245
-0.859009 -2.078297 0.001870 0.554442 1.180315
-0.571498 -0.517161 -0.614185 -0.134874 0.097806
```

函数: **Linear()** MIC4.8

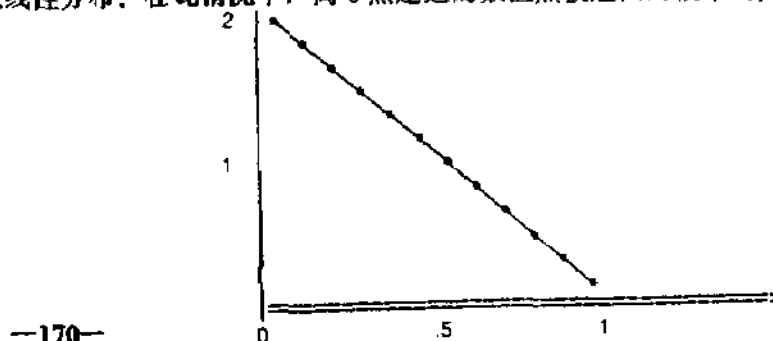
### 功能

生成一组随机数构成线性分布曲线。

### 注释

本函数的返回值是一些 0 与 1 之间的连续、无序实数。0 点附近的值被返回的概率最大。

只需对算法稍加改动，使基在同样概率下选较大的数而不是较小的数就可以得到一个反线性分布。在此情况下，离 0 点越近的数值点被返回的概率越大。



## 程序清单

```
/* LINEAR.C (Linear Probability Distribution Function) */
#include <stdio.h>
main()
{
    int j, total = 100, seed = -29454;
    float x[100], Linear();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Linear();
        printf("%f ", x[j]);
    }
} /* end of main */
/*----- MIC 4.8 -----*/
/* Linear() function */

float Linear()
{
    float result; /* final Linear value */
    float u1;     /* random number > 0 & < 1.0 */
    float u2;     /* random number > 0 & < 1.0 */

    u1 = rand() / 32767.;
    u2 = rand() / 32767.;
    if (u2 < u1)
        u1 = u2;
    result = u1;
    return result;
} /* end of Linear() function */
/*-----*/
/* END OF LINEAR.C */
```

## 程序运行实例

LINEAR.EXE

```
0.329173 0.172735 0.249733 0.592700 0.308481
0.279214 0.241523 0.391552 0.548906 0.316172
0.280038 0.931639 0.049013 0.370434 0.064028
0.753685 0.092654 0.807520 0.307291 0.687429
0.439161 0.011170 0.160588 0.550127 0.631031
0.874813 0.412732 0.048433 0.072085 0.460189
0.444960 0.278146 0.512284 0.115085 0.192480
0.135044 0.875149 0.660482 0.765038 0.761589
0.070711 0.035188 0.395642 0.149815 0.110202
0.157750 0.186773 0.577593 0.135319 0.549364
0.679556 0.051119 0.700674 0.481307 0.292367
0.848354 0.195502 0.081668 0.940184 0.075472
0.599048 0.533006 0.456160 0.374554 0.220649
0.554918 0.216803 0.289743 0.115116 0.065889
0.167730 0.456160 0.666372 0.066744 0.087008
0.002960 0.073061 0.428480 0.148473 0.483139
0.225043 0.249672 0.189795 0.625721 0.213416
0.243507 0.498978 0.043641 0.563585 0.216010
```

0.140690 0.377972 0.222083 0.467910 0.267037  
0.458419 0.296976 0.050081 0.179754 0.016602

## 函数: Logistic() MIC4.9

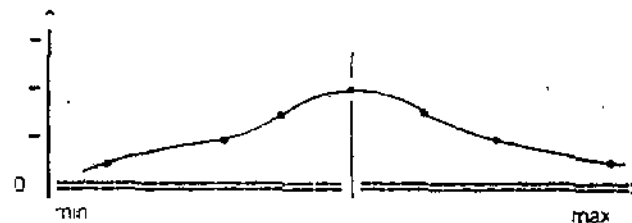
### 功能

生成一组随机数构成对数分布特性曲线。

### 注释

本函数返回值是一些连续、无序实数(包括正、负数)控制变量 par1 和 par2 决定中值和曲线取样点的分布。

Example curve.



### 程序清单

```
/* LOGISTIC.C (Logistic Probability Distribution Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = 742;
    double x[100], val1, val2, Logist();
    val1 = 1.0; /* controls mean of distribution */
    val2 = 5.0; /* controls dispersion of curve values */

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Logist(val1, val2);
        printf("%f ", x[j]);
    }
} /* end of main */

/*===== MIC 4.9 =====*/
/* Logistic() function */
double Logist(mean, dispersion)
double mean, dispersion;
{
    double result; /* final Logistic value */
    double u; /* random number > 0 & < 1.0 */
```

```

      u = rand() / 32767.;
      result = (-mean * -log(1.0 / u + 1.0)) / dispersion;
      return result;
} /* end of Logistic() function */
/*****
/* END OF LOGISTIC.C */

```

## 程序运行实例

### LOGISTIC.EXE

```

-0.334279 0.385297 -0.203285 0.127533 0.183139
-0.059083 0.015939 0.105260 -0.063907 -0.386510
0.748888 0.606583 -0.119194 0.142856 0.123525
-0.111957 -0.253996 -0.148164 0.061190 0.281807
-0.159932 -0.171647 -0.290905 0.446928 -0.044533
0.579250 -0.609667 -0.052188 -0.281729 0.005970
-0.341987 -0.173226 0.438072 0.162793 0.734264
-0.625013 0.425829 0.515534 -1.032069 0.905213
-0.237870 -0.006068 0.140399 0.026064 -0.000110
-0.130833 0.647482 -0.117517 0.398488 0.521853
0.229236 0.133987 -0.326322 0.243300 -0.324898
0.338804 0.299817 -0.680044 0.341563 0.022106
0.266573 0.655870 0.201163 0.319182 -0.261182
-0.069228 0.894464 -0.032556 0.105600 -0.238758
0.157768 -0.414888 0.563798 0.315804 -0.546233
-0.203472 0.188492 0.153015 0.227505 0.351556
0.894464 -0.516280 1.202541 0.370372 -0.236474
0.284291 -0.600000 -0.070410 0.170335 -0.209556
0.443941 -0.181169 0.152227 -0.036147 0.077902
0.024237 0.591258 0.004798 -0.646812 -0.548048

```

## 函数: Poisson() MIC4.10

### 功能

生成一组随机数构成的泊松分布曲线。

### 注释

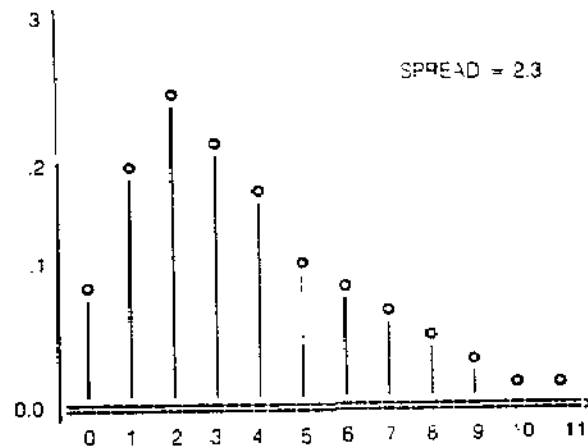
本算法生成无序的非负整数，取样点的横坐标值无上界，且由控制变量 spread 决定其分布。

本分布的中值点的横坐标值就是变量 spread 所取的值(必须 > 0)。

不连续随机变量

概率分布示意图:

## 程序清单



```

/* POISSON.C (Poisson Probability Distribution Function */
#include <stdio.h>
#include <math.h>
main()
{
    int x[100],j,total = 100,seed = 9213;
    int Poisson();
    double parml = 4.5; /* controls distribution of values */

    srand(seed);
    for (j = 0;j < total;j++)
    {
        if (j % 10 == 0)
            printf("\n");

        x[j] = Poisson(parml);
        printf("%d ",x[j]);
    }

    /* end of main */
    /*===== MIC 4.10 =====*/
    /* Poisson() function */
    int Poisson(spread)
    double spread; /* controls distribution of values */
    {
        int num = 0;
        double u,t;

        u = rand() / 32767.;
        t = exp(-spread);
        while (u > t)
        {
            num = num + 1;
            u = u * (rand() / 32767.);
        }
        return num;
    } /* end of Poisson() function */
    /*=====*/
    /* END OF POISSON.C */
}

```

## 程序运行实例

# POISSON.EXE

```

5 6 2 4 2 6 8 4 3 2
3 7 0 5 2 1 7 1 1 4
5 6 6 3 6 5 6 5 5 0
4 2 5 7 4 3 6 1 3 8
1 2 3 8 0 6 6 9 1 1
7 4 4 6 4 9 5 3 1 7
2 7 5 1 5 4 4 4 6 2
1 4 6 1 4 3 6 5 1 4
4 5 1 5 5 8 2 4 2 6
4 3 3 2 6 5 4 4 6 2

```

函数: Rnd-rnd() MIC4.11

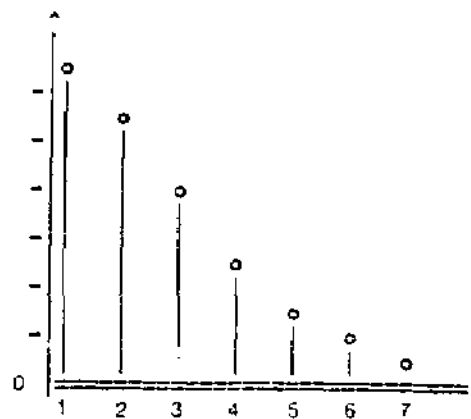
## 功能

生成一组随机数构成 Rnd-rnd()分布特性曲线。

## 注释

本函数应用一随机衰变函数来决定其不连续取样点的分布。其分布类似于指数概率分布，即数值越大出现频率越小。此外，只需对随机数生成程序实行递归调用即可生成无序整数。

不连续随机变量  
概率分布示意图:



## 程序清单

```

/* RND_RND.C (Recursive Random Distribution Function) */
#include <stdio.h>
main()

```

```

{
int x[300],j,total = 300,srand(),Rnd_rnd(),seed = -3112;
int bottom = 1; /* lowest possible random integer */
int top = 5; /* highest possible random integer */
int span; /* range of random integers */
span = top - bottom + 1;
srand(seed);
for (j = 0;j < total;j++)
{
if (j % 15 == 0)
printf("\n");
x[j] = Rnd_rnd(bottom,span);
printf("%d ",x[j]);
}
} /* end of main */
/*===== MIC 4.11 =====*/
/* Rnd_rnd() function */
int Rnd_rnd(low,range)
int low,range;
{
long int u;

u = ((rand(rand()) / 32767.) *
(rand(rand()) / 32767.)) * range + low;
return u;
} /* end of Rnd_rnd() function */
/*=====*/
/* END OF RND_RND.C */

```

#### 程序运行实例

```

1 4 2 3 1 2 4 2 2 2 2 2 1 1
4 2 2 1 2 2 1 3 3 1 3 2 2 1
2 4 2 1 1 4 3 1 1 1 2 1 1 3 1
1 3 3 1 1 1 2 2 5 1 2 2 1 1 4
2 4 2 2 1 1 3 1 3 4 4 3 3 1 1
1 3 2 1 1 1 1 1 1 4 1 1 3 1 4
1 1 1 3 2 1 3 2 5 1 1 1 1 1 3
1 1 1 1 1 1 2 2 4 1 4 1 1 3 1
2 4 4 1 2 3 1 2 1 1 1 1 3 1 3
1 3 3 1 1 3 1 2 1 3 1 2 2 1 2
4 1 1 1 1 1 1 5 1 1 1 1 1 1 1
4 2 1 4 2 1 1 3 2 1 2 2 3 1 3
1 1 1 1 1 1 1 2 1 2 4 1 2 2 1
1 1 1 1 2 1 2 1 2 3 1 1 1 1 2
3 2 1 2 1 4 1 1 2 1 2 1 1 3 3
1 5 1 3 2 1 1 4 1 3 3 1 2 1 1
1 1 2 1 1 4 2 2 4 2 2 4 1 2 1
2 4 1 3 3 1 1 1 1 2 1 1 3 4 1
1 2 1 1 1 3 1 2 4 1 1 1 3 2 1
3 1 3 1 1 1 2 2 1 1 4 2 2 1 1

```

#### 函数组 Weibull.c

1. Weibull() MIC4.12
2. Fpower() MIC2.2

#### 功能



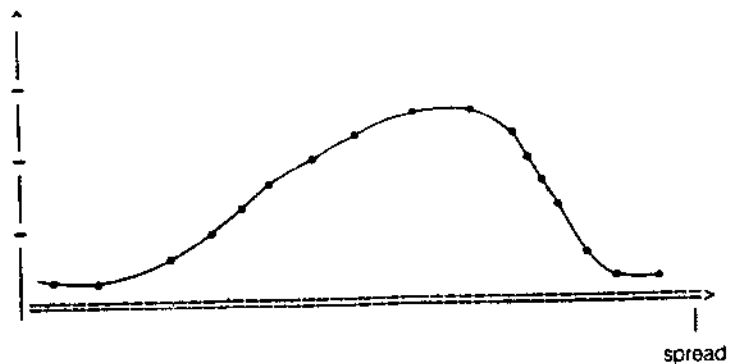
生成一组随机数构成韦伯尔分布特性曲线。

### 注释

本函数的返回值是一些无序、连续正实数。通过改变输入参数 denshape 的值来调整曲线形状。参量 spread 只控制横坐标值，且取样点横坐标无上限。

Example curve.

denshape = 3



### 程序清单

```
/* WEIBULL.C (Weibull Probability Distribution Function) */
#include <stdio.h>
#include <math.h>
main()
{
    int j, total = 100, seed = -212;
    double x(100), Weibull();
    double parm1 = 20.0; /* controls horizontal scale */
    double parm2 = 1.0; /* controls curve shape */

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");

        x[j] = Weibull(parm1, parm2);
        printf("%f ", x[j]);
    }
    /* end of main */
    /*===== MIC 4.12 =====*/
    /* Weibull() function */
    double Weibull(spread, denshape)
    double spread, denshape;
    {
        double result, u, s, t, Fpower();

        u = rand() / 32767.;
        u = 1. / (1. - u);
        s = -log(u);
```

```

t = 1. / denshape;
result = spread * Fpower(s,t);
return result;
} /* end of Weibull() function */
/*===== MIC 2.2 =====*/
/* Fpower() function */
double Fpower(value,tothe)
double value, tothe;
{
    int sign;
    double result;

    sign = (tothe < 0.0) ? -1 : 1;
    tothe = fabs(tothe);
    result = exp( log(value) * tothe);
    if (sign < 0)
        result = 1.0 / result;
    return(result);
} /* end of Fpower() function */
/*=====*/
/* END OF WEIBULL.C */

```

## 程序运行实例

### WEIBULL.EXE

```

9.256420 1.783166 15.610108 34.076964 6.735427
19.655075 7.211167 10.621601 1.357983 52.974343
13.436001 43.025359 23.690789 56.055919 8.077893
6.189065 7.029050 11.152833 18.695120 38.007816
79.543645 13.530627 2.979317 5.078500 12.622286
16.670149 7.720935 5.551414 1.866082 10.388356
13.914892 20.878987 6.146691 0.482472 16.970183
16.528776 0.825547 0.865023 18.412676 16.580450
30.764799 11.036971 40.634751 16.881971 20.357077
1.325999 2.702130 5.801103 32.787871 4.461040
11.886781 4.059244 6.891604 2.410134 13.801391
12.433870 3.633861 1.957418 16.889070 0.155638
64.757706 2.649799 5.305600 5.924664 6.595733
6.439314 55.431642 13.284815 5.387735 5.853380
10.810385 12.529569 30.022734 1.943960 5.393329
22.488511 15.304719 11.708425 7.636705 30.993471
5.898415 11.543599 32.819342 21.242887 11.070916
18.298062 2.244831 13.819654 3.896902 27.620571
11.292975 15.761214 1.588600 10.376047 25.559315
31.585740 41.637108 19.486188 6.024230 3.691772

```

## 函数: Triangle() MIC4.13

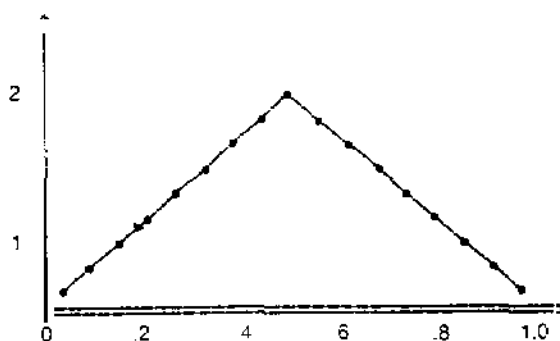
### 功能

生成一组随机数构成三角分布特性曲线。

### 注释

Triangle()函数的返回值是一些0到1之间的连续、无序实数。0.5附近的数值点被返回的概率最大。本算法通过将两个均匀分布求平均值的方法得出本分布。

Example distribution



### 程序清单

```
/* TRIANGLE.C (Triangular Prob. Distr. Function */
#include <stdio.h>
main()
{
    int j, total = 100, seed = 62;
    double x[100];
    double Triangle();

    srand(seed);
    for (j = 0; j < total; j++)
    {
        if (j % 5 == 0)
            printf("\n");
        x[j] = Triangle();
        printf("%f ", x[j]);
    }
} /* end of main */
/*===== MIC 4.13 =====*/
/* Triangle() function */

double Triangle()
{
    double result;
    double u1;
    double u2;

    u1 = rand() / 32767.;
    u2 = rand() / 32767.;
    result = .5 * (u1 + u2);
    return result;
} /* end of Triangle() function */
/*=====*/
/* END OF TRIANGLE.C */
```

### 程序运行实例

# TRIANGLE.EXE

0.628880	0.577883	0.396084	0.212683	0.325755
0.438734	0.648640	0.273400	0.595340	0.666875
0.345592	0.075426	0.483612	0.493866	0.740211
0.477966	0.303598	0.188696	0.479629	0.488952
0.482910	0.624317	0.140080	0.519089	0.286431
0.218863	0.637074	0.510666	0.414014	0.752235
0.510422	0.441496	0.973098	0.057405	0.584567
0.203803	0.326273	0.586886	0.503586	0.358211
0.314219	0.854076	0.641469	0.762551	0.853038
0.249657	0.387188	0.575732	0.407086	0.233100
0.804895	0.636799	0.836619	0.398984	0.669057
0.718787	0.620334	0.418439	0.874676	0.657476
0.453246	0.355480	0.722297	0.355495	0.719169
0.358562	0.578356	0.661214	0.838313	0.726859
0.410199	0.122440	0.490173	0.351650	0.196265
0.615619	0.211005	0.032868	0.711310	0.349467
0.366543	0.670415	0.686453	0.744087	0.499634
0.330485	0.844417	0.684759	0.552904	0.687185
0.809671	0.650899	0.466399	0.338710	0.393536
0.719932	0.150304	0.498917	0.754418	0.446211

## 第五章 排序与查找

函数: Shellart() MIC5.1

功能

将一组整数按升序排列。

注释

通常希法排序是一种简单、快捷、实用的算法。虽然在某些场合用普通的冒泡法排序可能更快,但因音乐文件中含有许多长而不连续的随机数,所以冒泡法不适于音乐文件的排序。当然,假若一系列数据只有 10 个数而且本身已有一定次序,用冒泡法排序还是更快一些。

在二叉树排序算法中,Shellsrt 函数通过不断地将一系列无序数一分为二,并用二数交换算法不断重排数列来实现排序。如果原数列还须保留,则可在用 Shellart 函数排序前将其存入数组 X[]。

Shellsrt 函数(以及本章的其他排序函数)还有另一有趣的应用。比如说,如果将待排序数列的数都在音调坐标轴上标出来,则排序算法的每一步就变成了一系列音调变化。如果需要进一步查看于程序的内部操作步骤,只须修改源程序以打印出数组 X[]中的原数列,在排序算法每一步操作中与之对照即可。

编程提示

1. 修改主程序及排序函数以便输出实型数据。
2. 将数列中的每一数表示为音调轴上一点,则排序过程转化为音调变化过程。

程序清单

```
/* SHELLSRT.C (Shell Sort routine, quite fast for long  
lists)*/  
#include <stdio.h>  
main()  
{  
    int x[200]; /* stores list to be sorted */  
    int total = 200, seed = 16213, j;  
    void ShellSrt();
```

```

srand(seed);
printf("an array of random-order integers to be sorted:\n");
for (j = 0; j < total; j++)
{
    if (j % 10 == 0)
        printf("\n");
    x[j] = rand() % 1000 + 1;
    printf("%d ", x[j]);
}
printf("\n\nthe array sorted in ascending order:\n");
ShellSrt(x, total);
for (j = 0; j < total; j++)
{
    if (j % 10 == 0)
        printf("\n");
    printf("%d ", x[j]);
}
} /* end of main */
/***** MTC 5.1 *****/
/* ShellSrt() function */
void ShellSrt(x, total)
int x[];
int total;
{
    register int j, k, l, s, w, y;
    int sortinc[5];

    sortinc[0] = 9; sortinc[1] = 5; sortinc[2] = 3;
    sortinc[3] = 2; sortinc[4] = 1;

    for (w = 0; w < 5; w++)
    {
        l = sortinc[w]; s = -1;
        for (j = 1; j < total; ++j)
        {
            y = x[j];
            k = j - 1;
            if (s == 0)
            {
                s = -1;
                s++;
                x[s] = y;
            }
            while (y < x[k] && k >= 0 && k <= total)
            {
                x[k+1] = x[k];
                k = k - 1;
            }
            x[k+1] = y;
        }
    }
} /* end of ShellSrt() function */
/***** */
/* END OF SHELLSRT.C */

```

### 程序运行实例

SHELLSORT.EXE

an array of random-order integers to be sorted:

```
324 23 160 518 156 997 606 119 486 527
284 949 401 554 689 959 56 95 158 630
475 46 368 371 331 887 673 695 416 949
26 299 518 507 869 913 688 971 705 810
196 268 349 385 505 923 762 384 348 30
715 961 714 820 239 224 18 889 706 54
617 473 15 345 67 978 681 18 742 334
291 542 562 273 493 225 392 736 472 368
119 195 300 262 605 316 314 129 477 767
189 501 828 980 989 217 642 411 245 561
678 528 731 67 816 62 369 621 422 424
27 215 504 462 314 525 782 373 214 630
556 554 145 677 67 759 331 141 75 343
713 566 630 790 211 585 655 485 836 166
731 900 523 909 800 496 890 342 420 703
222 451 264 609 800 516 590 506 82 627
663 333 973 425 48 843 105 48 630 107
716 442 1000 108 176 650 230 753 671 905
8 963 505 4 768 245 880 324 267 788
634 333 556 198 416 594 227 741 70 89
```

the array sorted in ascending order:

```
4 8 15 18 18 23 26 27 30 46
48 48 54 56 62 67 67 67 70 75
82 89 95 105 107 108 119 119 129 141
145 156 158 160 166 176 189 195 196 198
211 214 215 217 222 224 225 227 230 239
245 245 262 264 267 268 273 284 291 299
300 314 314 316 324 324 331 331 333 333
334 342 343 345 348 349 368 368 369 371
373 384 385 392 401 411 416 416 420 422
424 425 442 451 462 472 473 475 477 485
486 493 496 501 504 505 505 506 507 516
518 518 523 525 527 528 542 554 554 556
556 561 562 566 585 590 594 605 606 609
617 621 627 630 630 630 634 642 650 655
663 671 673 677 678 680 681 688 689 695
703 705 706 713 714 715 716 731 731 736
741 742 753 759 762 767 768 782 788 790
800 800 810 816 820 828 836 843 869 880
887 889 890 900 905 909 913 923 949 949
959 961 963 971 973 978 980 989 997 1000
```

函数: Shaksort() MIC5.2

功能

将一系列数值升序排列。

注释

Shaker 法排序是传统冒泡法排序的改进算法。如果程序要求保留原数列,可在调用排序函数前先将原数列存入一数组中。该函数适用于较短数列的排序。

如果希望查看 Shaksort 函数排序的每一步骤, 只要修改源程序, 打印出未经排序的原数列, 然后排序过程中与之对照即可。

#### 编程提示

1. 如前函数 MIC5.1 中所述, 可通过将数列中的数标在音量坐标轴上, 则排序算法的每一步骤就变成了一系列变量变化。
2. 编制一程序, 产生一随机干扰将一系列 1 到 100 之间的整数随机排列, 然后先将这些数作为音调坐标轴上的点来进行排序, 再作为音量、音调坐标轴上的点来进行排序。(这样做的结果是产生如第二章函数组 Curves.C 所生成的一系列曲线源程序。)

#### 程序清单

```
/* SHAKSORT.C (Improved version of Bubble sort, but still
               good only for short lists)*/
#include <stdio.h>
main()
{
    int x[200]; /* stores list to be sorted */
    int total = 200, seed = -11933, j;
    void ShakSort();

    srand(seed);
    printf("an array of random-order integers to be sorted:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 1000 + 1;
        printf("%d ", x[j]);
    }

    printf("\n\nthe array sorted in ascending order:\n");
    ShakSort(x, total);
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        printf("%d ", x[j]);
    }
    printf("\n");
} /* end of main */
/*=====*/
/* ShakSort() function */
void ShakSort(x, total)
int x[];
int total;
{
    register int j, k, l, m, temp;
    l = 1;
    k = total - 1;
    m = total - 1;
    do
    {
        for (j = m; j >= l; --j)
```



```

        if (x[j-1] > x[j])
        {
            /* swap */
            temp = x[j-1];
            x[j-1] = x[j];
            x[j] = temp;
            k = j;
        }
    }
    l = k + 1;
    for (j = l; j < n + 1; ++j)
    {
        if (x[j-1] > x[j])
        {
            /* swap */
            temp = x[j-1];
            x[j-1] = x[j];
            x[j] = temp;
            k = j;
        }
    }
    n = k - 1;
}
while (l <= n):
} /* end of ShakSort() function */
/*-----*/
/* END OF SHAKSORT.C */

```

### 程序运行实例

SHAKSORT.EXE

an array of random-order integers to be sorted:

326 414 325 137 606 456 479 733 908 578  
363 585 30 124 325 519 166 55 821 127  
672 595 99 479 303 137 864 404 492 107  
358 281 279 738 916 166 481 672 795 333  
834 855 837 602 96 656 496 233 312 523  
958 364 920 980 672 308 572 17 746 775  
681 126 113 99 471 348 500 905 875 490  
504 22 573 425 975 966 46 560 673 969  
53 393 500 114 66 835 569 77 459 66  
32 42 766 851 935 215 416 220 229 970  
484 140 932 798 110 455 927 357 185 284  
321 616 118 905 407 163 832 767 967 68  
548 298 834 204 229 988 286 342 592 323  
872 616 124 839 873 456 397 176 695 865  
937 303 830 781 507 364 487 468 360 64  
969 278 927 131 502 670 205 992 24 520  
984 527 740 971 194 716 742 810 122 190  
822 215 614 298 290 401 188 904 21 400  
610 462 607 574 652 116 6 720 433 999  
700 44 312 770 922 877 519 881 420 164

the array sorted in ascending order:

```

6 17 21 22 24 30 32 42 44 46
53 55 64 66 66 68 77 96 99 99
107 110 113 114 116 118 122 124 124 126
127 131 137 140 163 164 166 166 176 185
188 190 194 204 205 215 215 220 229 229
233 278 279 281 284 286 290 298 298 303
303 308 312 312 321 323 325 325 326 333
337 342 348 357 358 360 363 364 364 393
397 400 401 404 407 413 414 416 420 425
455 456 456 459 462 468 471 479 479 481
484 487 490 492 496 500 500 502 504 507
519 519 520 523 527 548 560 569 572 573
574 578 585 592 595 602 606 607 610 614
616 616 652 666 670 672 672 672 673 681
695 700 716 720 733 738 740 742 746 766
767 770 775 781 795 798 810 821 822 830
832 834 834 835 837 839 851 855 864 865
872 873 875 877 881 904 905 905 908 916
920 922 927 927 932 935 937 958 966 967
969 969 970 971 975 980 984 988 992 999

```

### 函数: Quicksort() MIC5.3

#### 功能

将一系列数升序排列。

#### 注释

虽然 Quicksort 函数是应用最快的排序算法，它的缺点在于程序复杂，冗长而且不易看懂。

若要详细查看 Quicksort 函数排序的每一步骤，只须如下所述修改源程序，并打印出排序算法每一步骤进行后数列中各数的位置。

#### 编程提示

1. 如在函数CTB5.1中所述，修改主程序及函数Quicksort()，将排序算法各步骤转化为一系列指向坐标轴上的点的指针的移动，并用此指针来调整一段旋律的各音符的音高。
2. 编程序，以实现MIC4.11:Rnd随机干扰功能产生一系列1到64之间的无序整数并存入一数组中。用 Quicksort 函数对数组元素排序，然后将数列中各数看作节拍坐标轴上的点，从而排序过程转化为一系列节拍值的变化。还可用函数组MOTFORMS(如第三章中所述)生成一文件实现音长的轮回、逆序、回转、变调等功能。

#### 程序清单

```

/* QUIKSORT.C (one of the fastest sorts for long lists) */
#include <stdio.h>
main()
{
    int x[800]; /* stores list to be sorted */
    int total = 800, seed = 1993, j;
    void QuikSort();

    srand(seed);
    printf("an array of random-order integers to be sorted:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 1000 + 1;
        printf("%d ", x[j]);
    }
    printf("\n\nthe array sorted in ascending order:\n");

    QuikSort(x, 0, total-1);
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        printf("%d ", x[j]);
    }
} /* end of main */
/*=====*/
/* QuikSort() function */
void QuikSort(x, left, right)
int x[];
int left, right;
{
    register int j, k, y, z;
    j = left;
    k = right;
    y = x[(left + right)/2];
    do
    {
        while (x[j] < y && j < right) j++;
        while (y < x[k] && k > left) k--;
        if (j <= k)
        {
            z = x[j];
            x[j] = x[k];
            x[k] = z;
            j++;
            k--;
        }
    }
    while (j <= k);
    if (left <= k)
        QuikSort(x, left, k);
    if (j < right)
        QuikSort(x, j, right);
} /* end of QuikSort() function */
/*=====*/
/* END OF QUIKSORT.C */

```

## 程序运行实例

QUICKSORT.C

an array of random-order integers to be sorted:

```
59 288 35 47 685 302 431 640 678 378
357 85 659 531 897 535 539 48 683 682
398 508 281 408 28 593 280 788 759 56
172 209 763 615 721 129 118 387 625 793
988 552 144 341 301 839 357 96 570 87
131 15 148 472 560 838 207 939 84 162
174 765 965 817 159 461 832 543 552 875
707 41 411 92 738 159 975 828 744 53
992 683 120 243 825 122 355 414 239 63
155 710 689 422 143 449 934 635 729 567
261 358 965 463 987 830 228 26 282 865
352 982 382 193 113 638 396 179 937 784
437 964 140 62 216 909 603 865 248 816
50 234 141 915 191 739 837 237 891 182

980 806 679 523 554 691 768 920 990 850
620 124 729 819 344 299 596 823 977 889
109 629 859 721 86 115 680 613 392 425
290 779 223 300 723 917 174 585 19 25
198 136 914 514 513 443 424 414 596 95
865 409 838 106 574 490 618 389 697 893
92 916 176 953 163 251 221 234 218 459
995 641 772 524 604 794 575 61 690 917
449 907 907 682 298 614 961 151 953 354
778 291 762 99 884 824 806 122 948 147
668 304 872 368 392 832 442 708 474 1
502 2 10 931 711 26 129 234 677 62
233 185 195 279 464 313 871 424 162 98
578 393 590 984 17 790 899 361 451 296
315 70 646 612 367 400 620 363 51 917
963 801 300 851 419 466 345 996 945 769
862 690 893 74 895 308 616 171 132 441
393 430 513 645 256 66 232 338 715 211
865 24 480 87 154 807 732 522 322 764
570 859 3 131 806 824 968 420 971 295
20 192 226 351 28 277 589 510 218 488
348 620 378 786 711 696 721 302 844 488
933 874 794 375 840 980 37 178 308 407
507 859 489 989 719 593 900 463 485 927
192 155 607 308 479 844 22 315 395 608
740 933 953 990 552 211 325 518 42 735
191 998 304 541 861 1 840 830 946 498
440 125 654 898 780 989 599 782 135 82
959 20 676 791 6 647 365 480 803 353
12 625 973 792 258 470 384 884 293 260
829 695 70 288 665 70 194 278 482 519
656 348 126 859 571 368 493 558 409 743
616 812 876 415 787 102 152 223 768 853
821 989 221 398 342 352 112 278 914 623
184 152 314 217 302 932 923 494 125 74
212 310 967 784 67 665 576 807 891 127
887 380 834 879 107 198 578 701 579 155
972 768 306 184 188 473 594 144 658 139
694 188 654 407 517 412 983 71 365 732
500 189 541 730 314 932 782 791 216 703
811 125 904 57 759 992 480 581 263 118
977 798 769 67 270 9 628 633 296 929
185 109 585 403 688 655 446 979 390 747
121 406 479 635 750 202 232 127 740 523
```

735 503 48 871 461 216 414 3 51 638  
 416 295 684 273 178 672 754 466 382 916  
 548 575 436 843 183 853 517 979 259 14  
 409 964 897 739 79 285 498 619 92 31  
 2 644 949 665 750 273 129 419 322 732  
 81 239 487 662 499 724 756 45 487 228  
 154 252 815 785 797 289 477 338 81 595  
 433 180 90 890 80 527 120 492 940 750  
 905 45 937 44 215 566 460 746 884 960  
 906 157 309 82 102 312 180 874 387 925  
 25 641 148 986 154 641 657 505 73 259  
 700 984 478 252 612 491 821 719 128 590  
 722 377 759 38 122 384 302 858 181 237  
 458 470 922 892 839 737 658 953 800 56  
 433 5 385 889 641 529 507 572 400 962  
 127 847 115 94 69 744 611 909 102 184  
 936 683 375 533 677 661 456 633 531 117  
 133 466 337 107 323 985 458 5 756 879  
 432 7 505 92 305 935 62 745 32 122  
 239 265 438 235 493 507 505 595 608 858  
 628 530 247 856 232 813 498 705 922 136  
 5 604 869 277 662 887 227 749 696 579

the array sorted in ascending order:

1 1 2 2 3 3 5 5 5 6  
 7 9 10 12 14 15 17 19 20 20  
 22 24 25 25 26 26 28 28 31 32  
 35 37 38 41 42 44 45 45 47 48  
 48 50 51 51 53 56 56 57 59 61  
 62 62 62 63 66 67 67 69 70 70  
 70 71 73 74 74 79 80 81 81 82  
 82 84 85 86 87 87 90 92 92 92  
 92 94 95 96 98 99 102 102 102 106  
 107 107 109 109 112 113 115 115 118 118  
 120 120 121 122 122 122 122 124 125 125  
 125 126 127 127 127 128 129 129 129 131  
 131 132 133 135 136 136 137 139 140 141  
 143 144 144 147 148 148 151 152 152 154  
 154 154 155 155 155 157 159 159 162 162  
 163 171 172 174 174 176 178 178 179 180  
 180 181 182 183 184 184 184 185 185 188  
 188 189 191 191 192 192 193 194 195 198  
 198 202 207 209 211 211 212 215 216 216  
 216 217 218 221 221 223 223 226 227 228  
 228 232 232 232 233 234 234 234 235 237  
 237 239 239 239 243 247 248 251 252 252  
 256 258 259 259 260 261 263 265 270 273  
 273 277 277 278 278 279 280 281 282 285  
 288 288 289 290 291 293 295 295 296 296  
 298 299 300 300 301 302 302 302 302 304  
 304 305 306 308 308 309 310 312 313 314  
 314 315 315 318 322 322 323 325 337 338  
 338 341 342 344 345 348 348 351 352 352  
 353 354 355 357 357 358 361 363 365 365  
 367 368 368 375 375 377 378 378 380 382

382	384	384	385	387	387	389	390	392	392
393	393	395	396	398	398	400	400	403	406
407	407	408	409	409	409	411	412	414	414
414	415	416	419	419	420	422	424	424	425
430	431	432	433	433	436	437	438	440	441
442	443	446	449	449	451	456	458	458	459
460	461	461	463	463	464	466	466	466	470
470	472	473	474	477	478	479	479	480	480
480	482	485	487	487	488	488	489	490	491
492	493	493	494	498	498	498	499	500	502
503	505	505	505	507	507	507	508	510	513
513	514	517	517	518	519	522	523	523	524
527	529	530	531	531	533	535	539	541	541
543	548	552	552	552	554	558	560	566	567
570	570	571	572	574	575	575	576	578	578
579	579	581	585	585	589	590	590	593	593
594	595	595	596	596	599	603	604	604	607
608	608	611	612	612	613	614	615	616	616
618	619	620	620	620	623	625	625	628	628
629	633	633	635	635	638	638	640	641	641
641	641	644	645	646	647	654	654	655	656
657	658	658	659	661	662	662	665	665	665
668	672	676	677	677	678	679	680	682	682
683	683	683	684	685	688	689	690	690	691
694	695	696	696	697	700	701	703	705	707
708	710	711	711	715	719	719	721	721	721
722	723	724	729	729	730	732	732	732	735
735	737	738	739	739	740	740	743	744	744
745	746	747	749	750	750	750	754	756	758
759	759	759	762	763	764	765	768	768	768
769	769	772	778	779	780	782	782	784	784
785	786	787	788	790	791	791	792	793	794
794	797	798	800	801	803	806	806	806	807
807	808	811	812	813	815	816	817	819	821
821	823	824	824	825	828	829	830	830	832
832	834	837	838	838	839	839	840	840	843
844	844	847	850	851	853	853	856	858	858
859	859	859	859	861	862	865	865	865	865
869	871	871	872	874	874	875	876	879	879
884	884	884	887	887	889	889	890	891	891
892	893	893	895	897	897	898	899	900	904
905	906	907	907	909	909	914	914	915	916
916	917	917	917	920	922	922	923	925	927
929	931	932	932	933	933	934	935	936	937
937	939	940	945	946	948	949	953	953	953
953	959	960	961	962	963	964	964	965	965
967	968	971	972	973	975	977	977	979	979
980	980	982	983	984	984	985	986	987	988
989	989	989	990	990	992	992	995	996	998

函数: Insertst() MIC5.4

功能

用插入排序算法将一系列数升序排列。

注释

插入法排序速度相当快，而且简单易懂。它有时被称为“扑克牌法排序”，因为在排序过程中，每一未经排序的元素根据已排好的序的元素位置插入到恰当的地方。

#### 编程提示

1. 编写一程序应用 CTB3.6 中的 Alterseq 函数及 Insertst 函数实现下列功能：
  - a. 将一系列无序数排序。
  - b. 将未经排序的数列中的数表示为音高坐标轴上的点。
  - c. 将已排序的数列及其逆序数表示为音量坐标轴上的点。
  - d. 将数列中的数表示为节拍坐标轴上的点。
2. 编写一程序应用 MIC3.25 中的 Valratio 函数、MIC3.5 中的 Displace 函数及 Insertst 函数实现下列功能：
  - a. 将一系列无序的经过冗余控制处理的整数存入一数组中。
  - b. 将数组排序，并用 Displace 函数处理它。
  - c. 将经过处理的数组各元素表示成指向节拍轴上的各点的指针。

#### 程序清单

```
/* INSERTST.C (insertion sort, aka card player's sort) */
#include <stdio.h>
main()
{
    int x[200]; /* stores list to be sorted */
    int total = 200, seed = 31213, j;
    void InsertSt();

    srand(seed);
    printf("an array of random-order integers to be sorted:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 1000 + 1;
        printf("%d ", x[j]);
    }
    printf("\n\nthe array sorted in ascending order:\n");
    InsertSt(x, total);
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        printf("%d ", x[j]);
    }
} /* end of main */
/*=====*/
/* InsertSt() function */
void InsertSt(x, total)
int x[];
int total;
{
    register int j, k, temp;
    for (j = 1; j < total; j++)
    {
```

```

        temp = x[j];
        k = j - 1;
        while (k >= 0 && temp < x[k])
        {
            x[k+1] = x[k];
            k--;
        }
        x[k+1] = temp;
    }
} /* end of InsertSt() function */
/*=====*/
/* END OF INSERTST.C */

```

### 程序运行实例

INSERTST.EXE

an array of random-order integers to be sorted:

```

562 151 603 434 419 178 340 309 769 652
951 769 173 829 200 873 429 444 592 204
93 886 530 396 414 725 542 43 508 739
397 187 293 575 13 797 56 358 279 605
211 638 774 203 640 862 200 52 393 970
58 578 947 413 584 617 288 660 356 892
752 530 808 183 569 664 974 541 628 551
14 100 395 206 856 456 57 137 556 957
538 444 503 457 252 397 303 137 740 388
442 162 156 807 282 547 196 281 948 465
536 762 563 371 266 805 407 558 96 302
235 523 287 474 453 288 296 699 415 828
411 428 991 455 738 921 811 825 377 546
832 129 723 946 596 208 695 675 137 447
739 412 92 135 688 693 477 583 448 57
232 706 404 165 236 970 218 865 811 7
541 605 794 229 214 157 370 157 482 311
727 586 52 164 800 623 486 756 111 918

83 383 722 4 984 994 841 708 698 38
49 303 514 391 274 293 39 91 29 673

```

the array sorted in ascending order:

```

4 7 13 14 29 38 39 43 49 52
52 56 57 57 58 83 91 92 93 96
100 111 129 135 137 137 151 156 157
157 162 164 165 173 178 183 187 196 200
200 203 204 206 208 211 214 218 229 232
235 236 252 266 274 279 281 282 287 288
288 293 293 296 302 303 303 309 311 340
356 358 370 371 377 383 388 391 393 395
396 397 397 404 407 411 412 413 414 415
419 428 429 434 442 444 444 447 448 453
455 456 457 465 474 477 482 486 503 508
514 523 530 530 536 538 541 541 542 546
547 551 556 558 562 563 563 575 578 583
584 586 592 596 603 605 605 617 623 628
638 640 652 664 673 675 680 688 693 695
698 699 706 722 723 725 727 738 738 739
739 740 752 756 762 769 769 774 794 797

```



```

800 805 807 808 811 811 812 823 829 832
841 856 862 865 873 886 892 918 921 946
947 948 951 957 970 970 974 984 991 994

```

## 函数: Selecsrt() MIC5.5

### 功能

将一系列数升序排列。

### 注释

选择法排序比冒泡法排序要快一些，但用于较长数列的排序时仍然很慢。在用 Selecsrt 函数排序过程中，先在数列中查找出最小的数，然后将其与第一个元素调换位置，而后再在从第二个元素起的数列中重复上述过程，直至排序结束。

### 编程提示

编一程序应用 MIC3.25 中的 Valratio 函数、MIC3.5 中的 Displace 函数和 Insertst 函数实现下列功能：

1. 将一系列无序的、经过的冗余控制处理的整数存入一数组中。
2. 将数组元素排序，并用 Displace 函数处理之。
3. 将经过上述处理的数组元素表示为指向节拍轴上各点的指针。

### 程序清单

```

/* SELECSRT.C (selection sort, faster than the bubble sort,
               but slow for long lists.)/
#include <stdio.h>
main()
{
    int x[200]; /* stores list to be sorted */
    int total = 200, seed = 9226, j;
    void Selecsrt();

    srand(seed);
    printf("an array of random-order integers to be sorted:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 1000 + 1;
        printf("%d ", x[j]);
    }
    printf("\n\nthe array sorted in ascending order:\n");
    Selecsrt(x, total);
    for (j = 0; j < total; j++)

```

```

        {
            if (j % 10 == 0)
                printf("\n");
            printf("%d ", x[j]);
        }
    } /* end of main */

    /*=====*/
    /* SelecSrt() function (First finds the item with the
       lowest value, then swaps it with the first
       list item; repeats the action with next lowest,
       second list item, and so on.) */
    void SelecSrt(x, total)
    int x[];
    int total;
    {
        register int j, k, l, temp;
        for (j = 0; j < total-1; ++j)
        {
            l = j;

            temp = x[j];
            for (k = j+1; k < total; ++k)
            {
                if (x[k] < temp)
                {
                    l = k;
                    temp = x[k];
                }
            }
            x[l] = x[j];
            x[j] = temp;
        }
    } /* end of SelecSrt() function */
    /*=====*/
    /* END OF SELECSRT.C */

```

### 程序运行实例

SELECSRT.EXE

an array of random-order integers to be sorted:

```

519 513 274 512 179 902 969 467 693 485
584 564 834 793 433 716 141 969 374 801
687 134 295 150 896 255 815 700 908 371
561 376 58 352 431 246 641 891 796 933
499 56 530 908 38 795 597 212 576 574
121 22 275 533 641 912 633 255 994 962
9 407 816 79 299 938 214 945 192 786
363 756 485 644 859 463 42 950 300 822
645 44 242 909 188 883 517 385 891 877
840 272 770 543 461 955 443 153 712 721
234 229 360 31 781 682 309 421 997 585
630 91 445 705 393 76 67 79 985 211
77 457 507 477 358 124 809 782 785 611
611 462 876 442 226 942 840 410 678 286
160 990 427 258 632 462 527 221 424 443
326 725 624 531 210 273 118 542 157 31
342 665 432 369 801 537 849 905 305 731
832 643 963 499 442 27 349 883 730 584
714 645 269 424 756 210 997 596 569 612
15 566 153 402 760 953 935 748 797 33

```

the array sorted in ascending order:

```

9 15 22 27 31 31 33 38 42 44
56 58 67 76 77 79 79 91 118 121
124 134 141 150 153 153 157 160 179 188
192 210 210 211 212 214 221 226 229 234
242 246 255 255 258 269 272 273 274 275
286 295 299 300 305 309 326 342 349 352
358 360 363 369 371 374 376 385 393 402
407 410 421 424 424 427 431 432 433 442
442 443 443 445 457 461 462 462 463 467
477 485 485 499 499 507 512 513 517 519
527 530 531 533 537 542 543 561 564 566
569 574 576 584 584 585 596 597 611 611
612 624 630 632 633 641 641 643 644 645
645 665 678 682 687 693 700 705 712 714
716 721 725 730 731 748 756 756 760 770
781 782 785 786 793 795 796 797 801 801
809 815 816 822 832 834 840 840 849 859
876 877 883 883 891 891 896 902 905 908
908 909 912 933 935 938 942 945 950 953
955 962 963 969 969 985 990 994 997 997

```

## 函数: Tsearch1() MIC5.6

### 功能

在一个无序表中查找某一文件记录或关键字(线性方法)。

### 注释

无序表线性查找法是本章四种查找法中最简单、低级的算法。它从前到后地扫描输入数组以查找目标文件记录。在主程序中,本查找算法用于将一系列无序数中的最大值与最小值查找出并存入数组指定位置,作为该列数的范围——此范围值将被程序的其他部分调用,或作为被为线性查找的运行区间值显示。函数将每一目标文件记录的位置以及出现频率返回主程序,如果未查找到,则显示相应信息。

在本函数中,数据与关键字码被同等对待,但在其他函数中,有时需要专门定义关键字码为另一数组的地址指针。

### 编程提示

编写一接口程序实现以下功能:

1. 生成并显示一系列无序整数(从 1 到 88),该列整数将被转换为相应音符。
2. 查找数列,找出一组音阶的各音所在位置,并以音量逐渐升高的次序排列。(可应用操作模块将选择的数组元素转换为一系列含八度音程的音阶。)
3. 将各音按百分比形式定音量值。

#### 4. 将其余数组元素代表的音定较低音量值(20%到 60%).

#### 程序清单

```
/* TSEARCH1.C (sequential table searching routine) */
/*useful in situations where a secondary event
or parameter change is tied to the occurrence
of a specific value */

#include <stdio.h>
main()
{
    int x[200]; /* stores list to be sorted */
    int total = 200, tablekey, tally, seed = 1993, j;
    int Tsearch1();

    srand(seed);
    printf("an array of random-order integers to be searched:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 12; /* pitch-classes of the scale */
        printf("%d ", x[j]);
    }

    tablekey = 9;
    printf("\nsearch item will be the integer %d\n", tablekey);
    tally = Tsearch1(x, total, tablekey);
    if (tally == 0)
        printf("\nnot found\n");
    else
        printf("\n%d appeared %d times", tablekey, tally);
    printf("\n");
} /* end of main */

/*=====*/
/* Tsearch1() function {sequential search returns frequency &
location(s) of keys */
int Tsearch1(x, total, key)
int x[], total, key;
{
    int j, sum = 0;
    for (j = 0; j < total; j++)
        if (x[j] == key)
        {
            sum++;
            printf("\nitem found at location %d", j);
        }
    return (sum);
} /* end of Tsearch1() function */
/*=====*/
/* END OF TSEARCH1.C */
```

#### 程序运行实例

## TSEARCH1.EXE

an array of random-order integers to be searched:

```
6 7 2 6 0 1 2 11 9 1
8 0 6 2 8 10 10 11 10 9
9 11 8 7 7 8 7 7 10 11
11 8 10 2 0 8 1 6 0 0
11 3 7 4 4 10 0 11 5 10
2 6 11 7 7 5 6 10 11 1
1 4 8 0 2 8 3 6 7 2
2 4 6 11 1 6 6 7 3 0
11 10 7 6 0 9 6 5 10 6
10 9 4 9 10 4 1 2 0 2
0 9 4 2 2 1 7 5 9 8
7 1 1 0 4 5 7 6 4 3
8 7 11 9 3 4 10 8 3 11
5 5 8 10 2 2 8 8 6 5
11 5 6 2 1 6 3 3 9 9
7 3 4 6 11 10 7 2 4 0
0 4 6 4 5 2 3 0 11 8
1 10 10 3 2 8 1 4 2 0
9 7 5 1 0 6 7 9 11 2
8 4 1 5 5 5 5 0 0 4
search item will be the integer 9
```

```
item found at location 8
item found at location 19
item found at location 20
item found at location 85
item found at location 91
item found at location 93
item found at location 101
item found at location 108
item found at location 123
item found at location 148
item found at location 149
item found at location 180
item found at location 187
9 appeared 13 times
```

## 函数组: Tsearch2.c

1. Quicksort() MIC5.3
2. Tsearch2() MIC5.7

## 功能

在一已排序的文件中查找指定的文件记录或关键字码(二叉树法)。

## 注释

本折半法查找函数只能应用于一系列已排好序的文件记录。它不但速度很快,而且常用,但由于它要求先将文件记录排序,大大加长了运行时间,所以在音乐文件中不如

MTC5.6 中的 Tsearch1 函数应用广泛。而且因为本函数的算法实际上是每次将其处理的文件记录列一分为二，直到查找到目标记录。所以列中重复出现的记录必须被预先删除，否则程序运行到查找出这些记录第一次出现的位置后就结束查找，并返回主程序。

上述程序可通过下列方法来防止出现：将含有重复记录的文件记录列存入一个数组中，再设一组指向该数组各元素的地址指针。查找时对这组指针进行操作。试举一例说明。若在一个字符串数组中存放一些文件记录，再用一组无序整数作为该字符串数组各元素的地址指针，该字符串数组中的各文件记录有可能是重复出现的，但由于设置了地址指针，使得每一个重复出现的记录的地址指针关键代码各不相同，而且由于这些关键代码将被升序排列，所以也不必是连续值。这样，排序后的数据文件就即可作为一种音乐参数来进行目标记录查找，亦可作为其他音乐参数来实现同一过程。

### 编程提示

1. 可编一程序将某一音乐参数的一组数据进行排序与查找，然后将该组数据作为指向另一音乐参数坐标轴上各点的指针，因而这组数据的排序查找过程实际上转化为另一音乐参数的数据排序与查找。
2. 编写一程序，实现下述操作：现有一列已排序的音高数据，其中某些指定音符需要被剔除，则先在该列音高数据中查找出这些指定音符，然后将此列音高数据作为指向音量坐标轴上各点的指针，并将要剔除的音符代表的指针指向的音量值变为零。这样就实现了指定音符的剔除。

### 程序清单

```
/* TSEARCH2.C (binary SORTED table searching routine
   finds 1st occurrence of key in list)*/

#include <stdio.h>
main()
{
    int x[200]; /* stores list to be sorted */
    int total = 200, tablekey, position, seed = 4132, j;
    int Tsearch1();
    void QuikSort();

    srand(seed);
    printf("an array of random-order integers to be searched:\n");
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 50; /* pitch-classes of the scale */
        printf("%d ", x[j]);
    }
    printf("\n---array now being sorted (ascending-order)---\n");
    QuikSort(x, 0, total-1);
    tablekey = 21;
    printf("\nsearch item will be the integer %d\n", tablekey);
}
```

```

position = Tsearch1(x,total,tablekey);
if (position == -1)
    printf("\nnot found\n");
else
    printf("\nid found at location %d",tablekey,position);
printf("\n");
} /* end of main */
/*=====MIC 5.3 =====*/
/* QuikSort() function */
void QuikSort(x,left,right)
int x[];
int left,right;
{
    register int j,k,y,z;
    j = left;
    k = right;
    y = x[(left + right)/2];
    do
    {
        while (x[j] < y && j < right) j++;
        while (y < x[k] && k > left) k--;
        if (j <= k)
        {
            z = x[j];
            x[j] = x[k];
            x[k] = z;
            j++;
            k--;
        }
    }
    while (j <= k);
    if (left <= k)
        QuikSort(x,left,k);
    if (j < right)
        QuikSort(x,j,right);
} /* end of QuikSort() function */
/*=====*/
/* Tsearch2() function (binary SORTED table search returns
location of key */
int Tsearch1(x,total,key)
int x[],total,key;
{
    int bottom,mid,top;

    bottom = 0; top = total-1;
    while (bottom <= top)
    {
        mid = (bottom + top) / 2;
        if (key < x[mid])
            top = mid-1;
        else if (key > x[mid])
            bottom = mid+1;
        else
            return mid;
    }
    return -1;
} /* end of Tsearch2() function */
/*=====*/
/* END OF TSEARCH2.C */

```

## 程序运行实例

TSEARCH2.EXE

an array of random-order integers to be searched:

```
10 42 46 6 22 23 27 19 19 0
36 5 45 15 47 35 46 22 36 40
24 11 14 7 48 19 28 34 39 11
11 46 20 35 21 24 31 34 8 11
0 13 17 44 22 21 46 34 21 44
44 1 42 11 15 6 34 37 40 23
49 6 0 4 15 20 0 9 17 26
15 43 45 25 26 45 19 42 48 39
14 34 35 2 23 7 14 17 3 26
6 48 43 6 39 43 47 42 9 43
13 20 31 33 34 11 16 46 16 13
47 25 49 5 27 29 35 37 26 35
1 23 27 47 49 2 45 30 30 43
13 10 48 43 13 1 20 39 44 38
3 43 37 8 49 27 47 0 21 21
34 23 45 30 37 11 9 8 45 22
45 44 35 18 36 39 43 4 9 9
29 15 7 9 7 19 17 21 30 8
29 23 1 49 29 10 33 5 15 8
36 31 22 27 39 26 6 43 24 36
```

---array now being sorted (ascending-order)---

search item will be the integer 21

21 found at location 80

## 函数组: Tsearch3.c

1. Quicksort() MIC5.3
2. Tsearch3() MIC5.8

## 功能

用逼近查找法在一已排序记录列中查找目标文件记录。

## 注释

本算法要求先将文件记录排序，它比折半算法和插值法更有效，并且该算法无论是否查到目标记录，都按其值大于还是小于目标记录将原序列分为两个子序列。

查算法适用于下述情形：即依前后关系要求而定的一系列有序数据解必须是符合规定



线性或重向数据集的一系列规则的。例如，一段旋律可以通过利用输入乐段的休止音阶根据某一原则(以逼近目标值的方式存储)生成后续段落的休止音阶的方法来谱出，或者说，可用一些随机整数表示出一系列无序数构成的数表中的随机路径。

### 编程提示

1. 编写一接口程序实现下列功能:
  - a. 将 20 个从 1 到 76 的无序整数(已经过冗余处理)存入一驱动数组。
  - b. 将一列从 1 到 88 的无序数存入一参考数组。
  - c. 应用逼近查找算法驱动数组中的各元素的值。
  - d. 在查找出的整数数表中找一随机路径。
  - e. 将此随机路径转化为一系列音符。
2. 编写一接口程序实现根据存在参考数组中的一系列规则来确定一段旋律发音间断方式。可以应用其他参数数据作为指向该发音方式参数坐标轴上各点的指针进行运算，如在 CTB5.8 中所述。

### 程序清单

```
/* TSEARCH3.C (SORTED table proximity search routine
returns subset around 1st occurrence of
search key )*/

#include <stdio.h>
main()
{
    int x[100]; /* stores list to be sorted */
    int total = 100, tablekey, swidth, seed = 1993, j;
    void Tsearch3();
    void QuikSort();
    srand(seed);
    printf("an array of random-order integers to be searched:\n")
    for (j = 0; j < total; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        x[j] = rand() % 200; /* pitch-classes of the scale */
        printf("%d ", x[j]);
    }
    printf("\narray now being sorted (ascending-order)\n");
    QuikSort(x, 0, total-1);

    tablekey = 21;
    swidth = 5;
    printf("\nsearch item will be the integer %d\n", tablekey);
}
```

```

    Tsearch3(x,total,tablekey,swidth);
} /* end of main */
/*=====MIC 5.1 =====*/
/* QuikSort() function */
void QuikSort(x,left,right)
int x[];
int left,right;
{
    register int j,k,y,z;
    j = left;
    k = right;
    y = x[(left + right)/2];
    do
    {
        while (x[j] < y && j < right) j++;
        while (y < x[k] && k > left) k--;
        if (j <= k)
        {
            z = x[j];
            x[j] = x[k];
            x[k] = z;
            j++;
            k--;
        }
    } while (j <= k);
    if (left <= k)
        QuikSort(x,left,k);
    if (j < right)
        QuikSort(x,j,right);
} /* end of QuikSort() function */
/*=====*/
/* Tsearch3() function (proximity search returns
   a subset of the list on either side
   of the search key */
void Tsearch3(x,total,key,subset)
int x[],total,key,subset;

```

```

(
    int bottom,top,j,flag = 1;
    for (j = 0;j < total;++j)
        if (key <= x[j])
        {
            flag = 0;
            break;
        }
    if (flag )
        j = total + 1;
    if (x[j] == key)
        top = j + subset;
    else
        top = j + subset - 1;
    bottom = j - subset;
    if (top > total)
        top = total;
    if (bottom < 1)
        bottom = 1;
    for (j = bottom;j <= top;j++)
        printf("%d ",x[j]);
} /* end of Tsearch3() function */
/*****
/* END OF TSEARCH3.C */

```

## 程序运行实例

```
TSEARCH1.EXE
an array of random-order integers to be searched:

58 87 34 46 84 101 10 39 77 177
156 84 58 130 96 134 138 47 82 81
197 107 80 7 27 192 79 187 158 55
171 8 162 14 120 128 117 186 24 192
187 151 143 140 100 38 156 95 169 86
130 14 147 71 159 37 6 138 83 161
173 164 164 16 158 60 31 142 151 74
106 40 10 91 137 158 174 27 143 52
191 82 119 42 24 121 154 13 38 62
154 109 88 21 142 48 133 34 128 166
array now being sorted (ascending-order)

search item will be the integer 21
10 13 14 14 16 21 24 24 27 27 30
```

## 第六章 歌词 / 配乐设计

歌词文本的运用与处理已为作曲者们广泛用于从根据歌词谱写曲子到用计算机在电声乐器上奏出的音乐的歌词部分,无所不包。本章中的四个子程序主要是关于下面方面:

1. Poem.c(歌词)
2. Phone.c(语言)
3. Txtparse.c(配乐)
4. Drivel.c(叠唱)

歌词文本中要调用的各函数应存于同一文件 textlib.c 中。若要调用该文件中的函数,应该使用并 include "textlib.c" 为伪指令,把 textlib.c 头文件包含到源程序文件中,或者如下所示,将这条伪指令写入文本处理头文件:

```
/* text.h (definitions for text processing)
/* #include "textlib.c" (add this line for automatic inclusion */
/* of all text-processing functions) */

#include <stdio.h>
#include <ctype.h>
#include <alloc.h>
#include "randmain.c"
#define MAXDATA 100
#define MAXVOCAB 100
#define MAXWORDSIZE 80
#define TRUE 1
#define FALSE -1
static char vowels[6] = {"AEIOU"};
static char punctuation[10] = {".,:;!-/?"};
char *phones[MAXVOCAB];
char *diphth[MAXVOCAB];
char *words[MAXVOCAB];
char *verbs[MAXVOCAB];
char *adjectives[MAXVOCAB];
char *nouns[MAXVOCAB];
char *articles[3] = {"a", "the", "an"};
int wct = 0;
int dip = 0;
int ph = 0; /* size of arrays */
int vbct = 0;
int adct = 0;
int nct = 0;
int act = 3;
/******/
```

### 函数组: Poem.c

1. GetVocab() MIC.6.1
2. assign() MIC.6.2

- 3. MakePoem() MIC.6.4
- 4. VocabSummary() MIC.6.3
- 5. SelectWorld() MIC.6.5

## 功能

撰写歌词。

## 注释

本程序从数据文件中读出词语的词性，并根据词性排列出一张单词表。然后用 MakePoem 函数从单词表中选词构成句子。可以通过改变 MakePoem 函数的选词方式来构成句子的不同格式。

POEM.C 函数以此方式作出句子的句法结构和韵律都比较呆板，不见得会人人欣赏，而且经常会出现风、马、牛不相及的词词排列在一起，使诗句变得毫无意义。为了消除这种“紊乱”，可以尽量减少句子中的形容词、名词、动词和介词并尽量选用那些在任何场合都不致引起歧义的词语。与此相反，仅凭臆想而盲目扩充程序单词表中的词只会导致作出的歌词含义模糊，词不搭义。

## 编程提示

可以编制一接口程序如下：

1. 从文件单词表读取词语
2. 设置一个有关单词表词数多少的变量，通过改变其值来调整生成的句子的意义清晰程度。
3. 将作出的歌词的句子转换成为音乐数据，并根据数据定出其音高、节拍、发音方式，然后构成一乐句作为高音声部。
4. 将最终生成的音项表存盘。

## 程序清单

```
POEM.C
#include <stdio.h>
#include "fgetword.c"
#include "text.h"
int GetVocab(FILE *fp, char *part[]);
void MakePoem(FILE *fp);
void VocabSummary(void);
main()
{
    FILE *fp;
    extern char *adjectives[];

    fp = fopen("adj.dat", "r");
    adct = GetVocab(fp, adjectives);
}
```

```

    fclose(fp);
    fp = fopen("verbs.dat", "r");
    vbct = GetVocab(fp, verbs);
    fclose(fp);
    fp = fopen("nouns.dat", "r");
    nct = GetVocab(fp, nouns);
    fclose(fp);
    VocabSummary();
    fp = fopen("poem.dat", "w");
    MakePoem(fp);
    fclose(fp);
}
/*****
GetVocab(fp, part)
    FILE *fp;
    char *part[];
{
    int j = 0;

    while(!feof(fp))
        j = assign(fp, part, j);

    return(j);
}
*****/
assign(fp, part, position)
    FILE *fp;
    char *part[];
    int position;
{
    char *string;

    string = (char *) malloc(MAXWORDSIZE);
    fgetword(string, MAXWORDSIZE, fp);
    if(!isspace(string[0]))
        part[position++] = string;
    return(position);
}
/*****
void VocabSummary()
{
    int j;

    printf("\nVocabulary:\n");
    for(j = 0; j < nct; j++)
        printf("nouns %d = %s\n", j, nouns[j]);
    for(j = 0; j < adct; j++)
        printf("adjectives %d = %s\n", j, adjectives[j]);
    for(j = 0; j < vbct; j++)
        printf("verbs %d = %s\n", j, verbs[j]);
    for(j = 0; j < 3; j++)
        printf("articles %d = %s\n", j, articles[j]);
}
*****/
void MakePoem(fp)
    FILE *fp;
{
    int j, num, hownany, prt.seed;

```

```

printf("\nHow many lines do you want?\n");
howmany = getnum();
printf("\nWrite to file? (1) = yes\n");
prt = getnum();
randomize();
for(j = 0; j < howmany; j++)
{
    SelectWord(fp, articles, 3, prt);
    SelectWord(fp, adjectives, adct, prt);
    SelectWord(fp, nouns, nct, prt);
    SelectWord(fp, verbs, vbct, prt);
    SelectWord(fp, articles, 3, prt);
    SelectWord(fp, adjectives, adct, prt);
    SelectWord(fp, nouns, nct, prt);

    printf("\n");
    if(prt == 1)
        fprintf(fp, "\n");
}
}
/*****
SelectWord(fp, part, ct, flag)
FILE *fp;
char *part[];
int ct, flag;
{
    int num;
    num = irand(0, ct - 1);
    printf("%s ", part[num]);
    if(flag == 1)
        fprintf(fp, "%s ", part[num]);
}
*****/

```

## 程序运行实例

POEM.EXE

```

Vocabulary:
nouns 0 = thing
nouns 1 = book
nouns 2 = computer
nouns 3 = desk
nouns 4 = sky
nouns 5 = ocean
nouns 6 = floor
nouns 7 = ceiling
nouns 8 = girl
nouns 9 = boy
nouns 10 = tree
nouns 11 = cat
nouns 12 = table
nouns 13 = car
nouns 14 = ball
adjectives 0 = blue
adjectives 1 = wind-swept
adjectives 2 = short
adjectives 3 = long
adjectives 4 = fat
adjectives 5 = skinny
adjectives 6 = azure
adjectives 7 = pretty
adjectives 8 = handsome

```



```

adjectives 9 = fast
verbs 0 = insert
verbs 1 = show
verbs 2 = licks
verbs 3 = flick
verbs 4 = walk
verbs 5 = talk
verbs 6 = run
verbs 7 = acts
verbs 8 = asked
verbs 9 = wrote
verbs 10 = executed
articles 0 = a
articles 1 = the
articles 2 = an

How many lines do you want?      5
Write to file? (1) - yes         2

the blue computer show the pretty thing
the handsome book run a wind-swept sky
a blue table licks a pretty ball
the blue ocean acts a short floor
the azure floor talk the long ball

```

### 函数组: Phone.c

- |                   |          |
|-------------------|----------|
| 1. chop()         | MIC 6.35 |
| 2. SoundSummary() | MIC 6.29 |
| 3. MakeText()     | MIC 6.32 |
| 4. getchonews()   | MIC 6.38 |
| 5. isvowel()      | MIC 6.36 |
| 6. getdiphthws    | MIC 6.37 |
| 7. ExtracWord()   | MIC 6.39 |

### 功能

语音调整。

### 注释

本程序从“stdin”文件中每次读出一行，然后将句中的词语的字母按单音、复音(或简单地按元音、辅音)分类，并分别存储在全局数组 phones[]和 diph[]中。MakeText()函数可通过按元音辅音结合的方式从上述数组中选取元素构成新词，词的语音结构可通过改变函数选取元素的方式来进行调整。

### 维护提示

1. 可以修改程序以便根据用户自定义的元音、辅音搭配形式来构造语音结构更复杂的词。
2. 可用函数组TEXTPARSE.C中的函数将单音、复音词配曲，并定音高、节拍。

### 程序清单

```

PHONE.C
#include "text.h"
#include "fgetline.c" /* MICSP_15.0 */
main()
{
    int j, k, howmany;
    char string[80];
    FILE *fp;

    fp = fopen("junk.txt", "w");
    printf("\nInclude newline? (1) yes\t");
    j = getnum();
    printf("\nHow many lines?\t");
    howmany = getnum();
    for(k = 0; k < howmany; k++)
    {
        printf("Input string\t");
        if(j == 1)
            fgetline(stdin, string, 80);
        else
            gets(string);
        puts(string);
        chop(string);
    }
    SoundSummary();
    MakeText(fp);
    fclose(fp);
} /* end of main */
/*===== MIC 6.35 =====*/
chop(string)
char string[];
{
    int position = 0;

    while(position < strlen(string))
    {
        if(isvowel(string[position]) == FALSE)
            position = getphonews(string, position);
        else if(isvowel(string[position]) == TRUE)
            position = getdipthws(string, position);
        else
            position++;
    }
}
/*===== MIC 6.29 =====*/
SoundSummary()
{
    int j;

    printf("\nSounds\n");
    for(j = 0; j < ph; j++)
        printf("phones %d = %s\n", j, phones[j]);
    printf("%d phones\n", ph);
    for(j = 0; j < dip; j++)

```

```

        printf("diphthongs %d = %s\n", j, dipth[j]);
        printf("%d diphthongs\n", dip);
    }
    /*===== MIC 6.32 =====*/
    MakeText(fp)
    FILE *fp;
    {
        int j, num, howmany, prt;

        printf("\nHow many words do you want?\n");
        howmany = getnum();
        printf("\nWrite to file? (1) = yes\n");
        prt = getnum();

        for(j = 0; j < howmany; j++)
        {
            num = irand(0, ph - 1);
            printf("%s", phones[num]);
            if(prt == 1)
                fprintf(fp, "%s", phones[num]);
            num = irand(0, dip - 1);
            printf("%s", dipth[num]);
            if(prt == 1)
                fprintf(fp, "%s", dipth[num]);
        }
    }
    /*===== MIC 6.36 =====*/

    isvowel(ch)
    char ch;
    {
        int j;

        for(j = 0; j < 5; j++)
            if(toupper(ch) == vowels[j])
                return(TRUE);
        return(FALSE);
    }
    /*===== MIC 6.37 =====*/
    getdiphthws(string, position)    /* keep the spaces */
    char string[];
    int position;
    {
        char *vstr;
        int j = 0;

        vstr = (char *) malloc(80);
        while(isvowel(string[position]) == TRUE)
            vstr[j++] = string[position++];
        vstr[j] = '\0';
        if(strlen(vstr) > 0)
            dipth[dip++] = vstr;    /* dip is global counter */
        return(position);
    }
    /*===== MIC 6.38 =====*/
    getphonews(string, position)
    char string[];
    int position;
    {
        char *cstr;
        int j = 0;
    }

```

```

    cstr = (char *) malloc(80);
    while(isvowel(string[position]) == FALSE)
    {
        cstr[j++] = string[position++];
        if(ispunct(cstr[j - 1]))
            cstr[j++] = ' ';
    }
    cstr[j] = '\0';
    if(strlen(cstr) > 0)
        phones[ph++] = cstr; /* ph is global counter */
    return(position);
}
/*===== NIC 6.19 =====*/
ExtractWord(string, position)
char string[];
int position;
{
    int j = 0;
    char *word;

    word = (char *) malloc(80);
    while(!isspace(string[position]))
        word[j++] = string[position++];

    word[j++] = '\0';
    words[wct++] = word;
    return(position);
}
/*=====*/

```

## 程序运行实例

PHONE.DAS

```

Include newline? (1) yes      1
How many lines? 1
Input string
four score and seven years ago

```

```

Sounds
phones 0 = f
phones 1 = r sc
phones 2 = r
phones 3 =
phones 4 = nd s

```

```

phones 5 = v
phones 6 = n y
phones 7 = rs
phones 8 = g
phones 9 =

```

```

10 phones
diphthongs 0 = ou
diphthongs 1 = o
diphthongs 2 = e
diphthongs 3 = a
diphthongs 4 = e
diphthongs 5 = e
diphthongs 6 = ea
diphthongs 7 = a
diphthongs 8 = o
9 diphthongs

```

```

How many words do you want?      100
Write to file? (1) = yes         2

```

```

n yeron yavagend ser sccrorou ou
efond sera
ar scevend so
or scou
en yearan you
ors ovogars and sa
en yeafaveafe ovo en yefen yearours en yan yegor scoveveafears
a ero end sean yon yend ser scen yond sorea and segou
eveva agors eage a
efouva
ond sefour scou ero arer scarearago
eareavou
ora
o
afa egou ouve

```

### 函数组: Txtparse.c

1. copytxt( ) MIC 6.6
2. PutTotals( ) MIC 6.7
3. Putrhythm( ) MIC 6.8
4. PutConsonants( ) MIC 6.9
5. PutPunctuation( ) MIC 6.10
6. PutVowels( ) MIC 6.11
7. transpose( ) MIC 6.12
8. PutCharSummary( ) MIC 6.13
9. PutData( ) MIC 6.14
10. gettrans( ) MIC 6.15
11. getstring( ) MIC 6.16
12. getarray( ) MIC 6.17
13. showarray( ) MIC 6.18
14. Zero( ) MIC 6.19
15. PutScore( ) MIC 6.20
16. numvowel( ) MIC 6.21
17. numpunctuation( ) MIC 6.22

18. numconsonants( )	MIC 6.23
19. WhichVowels( )	MIC 6.25
20. WhichPunctuation( )	MIC 6.26
21. WhichConsonants( )	MIC 6.27
22. hasvowel( )	MIC 6.28

## 注释

本程序用于读入歌词文件。程序中打开两个文件：一文件用于歌词分析，另一文件为分析过的歌词谱曲。前一文件每次分析一行歌词，而后一文件谱出的曲子是一 ASCII 歌词文件并列出了每一音符的主要参数值，即 P(音高)、R(节拍)、A(发音方式)、V(变量)、T(音质)等。下例是一段由五个音组成的旋律：

P	C#4	D4	F4	A4	E4	字母为音符名，数字为音阶名
R	1	8	8	1	1	以全音的几分之几来代替节拍
A	100	50	50	100	100	以百分比表示发音时间
V	40	80	80	40	40	以百分比表示音量
T	12	43	2	55	73	以乐器代号代表不同乐器或音调

## 编程提示

修改程序，实现为歌词文件定恰当节拍的功能，并为最终输出的旋律定节拍。

## 程序清单

```

TXTPARSE.C
#include "textlib.c"
#include "getfnam.c"
#include "array.c"
#include "synclavi.c"
#include "fgetline.c"
int wordcnt, lnwordcnt, globaltrans;
FILE *input;
FILE *output;
FILE *score;
FILE *getfnam();
/*****
main()
{
    int j, k, size, numv, numc, nump, trans, ask, max, min,
        datarray[80];
    int punctarray[80], carray[80], varray[80], slen, usepunct;
    char string[80];

    input = getfnam("input text","r");
    output = getfnam("output data","w");
    score = getfnam("Script score","w");
    globaltrans = trans = wordcnt = 0;

```

```

trans = gettrans();
printf("\nTransposition factor = %d\n", trans);
printf("\nKeep the punctuation? (1) = yes ");
scanf("%d", &usepunct);
if(usepunct != 1)
    usepunct = FALSE;
copytxt(input, output, TRUE); /* echo text */
copytxt(input, score, FALSE);
fprintf(score, "\nNotelist using R 1-1\n");
do {
    lnwordcnt = 0;
    Zero(datarray);
    getstring(string);
    slen = strlen(string);
    size = getarray(string, slen, datarray, usepunct);
    if(size != 0)
    {
        showarray(datarray, size);
        max = getmax(datarray, size);
        min = getmin(datarray, size);
        printf("\nMax = %d Min = %d\n", max, min);
        fprintf(output, "\nMax = %d Min = %d\n", max, min);
        if(globaltrans == FALSE)
        {
            trans = 0 - min;
            transpose(datarray, size, trans);
            fprintf(output, "\nData normalized to 0, rhythm to 1\n");
            fprintf(score, "\n/* Data normalized to 0, rhythm to 1 */\n");
            fprintf(stdout, "\n/* Data normalized to 0, rhythm to 1 */\n");
            PutData(datarray, size, string);
        }
        else
        {
            transpose(datarray, size, trans);
            fprintf(stdout, "\nData transposed %d\n", trans);
            fprintf(output, "\nData transposed %d\n", trans);
            fprintf(score, "\nData transposed %d\n", trans);
            PutData(datarray, size, string);
        }
        numv = numvowel(string);
        numc = numconsonants(string);
        nump = numpunctuation(string);
        PutCharSummary(numv, numc, nump);
        PutVowels(varray, numv, trans, string);
        PutConsonants(carray, numc, trans, string);
        PutPunctuation(punctarray, nump, trans, usepunct, string);
    }
    PutTotals(wordcnt, lnwordcnt);
} while (!feof(input));

fclose(input);
fclose(output);
fclose(score);
}
/*****/
copytxt(fpi, fpo, echo)
FILE *fpi, *fpo;
int echo;
{
    char string[80];

    if(echo)

```

```

        printf("\nOriginal complete text:\n");
        fprintf(fpo, "\n/* Original complete text:\n");
        do {
            fgetline(fpi, string, 80);
            if(echo)
                puts(string);
            fputs(string, fpo);
        } while (!feof(fpi));
        fprintf(fpo, "\n*/\n");
        rewind(fpi);
    }
    /*****
PutTotals(wordcnt, lnwordcnt)
    int wordcnt, lnwordcnt;
    {
        printf("\nThere are %d words total\n", wordcnt);
        printf("There are %d words in this line\n\n", lnwordcnt);
        printf("-----\n");
        fprintf(output, "\nThere are %d words total\n", wordcnt);
        fprintf(output, "There are %d words in this line\n\n",
            lnwordcnt);
        fprintf(output, "-----\n");
    }
    /*****
Putrhythm(fp, darray, size)
    FILE *fp;
    int darray[], size;
    {

        int j;

        for(j = 0; j < size; j++)
            darray[j] += 1;
        FscriptArray(fp, darray, 'R', size);
    }
    /*****
PutConsonants(carray, numc, trans, string)
    int carray[], numc, trans;
    char string[];
    {
        printf("\nThe consonants are:\n");
        fprintf(output, "\nThe consonants are:\n");
        fprintf(score, "\n/* The consonants are: */\n");
        WhichConsonants(output, string, carray, trans);
        ScriptArray(carray, 'P', numc);
        FscriptArray(score, carray, 'P', numc);
        Putrhythm(score, carray, numc);
    }
    /*****
PutPunctuation(punctarray, nump, trans, usepunct, string)
    int punctarray[], nump, trans, usepunct;
    char string[];
    {
        if(usepunct)
        {
            printf("\nThe punctuations marks are:\n");
            fprintf(score, "\n/* The punctuation marks are: */\n");
            fprintf(output, "\nThe punctuation marks are: \n");
            WhichPunctuation(output, string, punctarray, trans);
            ScriptArray(punctarray, 'P', nump);
            FscriptArray(score, punctarray, 'P', nump);
            Putrhythm(score, punctarray, nump);
        }
        else

```



```

        {
            printf("\nThe punctuations marks are not used\n\n");
            fprintf(score, "\n/* The punctuation marks are not used */\n\n");
            fprintf(output, "\nThe punctuation marks are not used \n\n");
        }
    }
    /*****
PutVowels(varray, numv, trans, string)
    int varray[], numv, trans;
    char string[];
    {
        printf("\nThe vowels are:\n");
        fprintf(output, "\nThe vowels are:\n");
        fprintf(score, "\n/* The vowels are: */\n");
        WhichVowels(output, string, varray, trans);
        ScriptArray(varray, 'P', numv);
        FscriptArray(score, varray, 'P', numv);
        Putrhythm(score, varray, numv);
    }
    /*****
transpose(datarray, size, trans)
    int datarray[], size, trans;
    {
        int k;
        for(k = 0; k < size; k++)
            datarray[k] += trans;
    }
    /*****
PutCharSummary(numv, numc, nump)
    int numv, numc, nump;
    {
        printf("\nit has %d vowels", numv);
        printf("\nit has %d consonants", numc);
        printf("\nit has %d punctuation", nump);
        fprintf(output, "\nit has %d vowels", numv);
        fprintf(output, "\nit has %d consonants", numc);
        fprintf(output, "\nit has %d punctuation", nump);
    }
    /*****
PutData(datarray, size, string)
    int datarray[], size;
    char string[];
    {
        PutArray(datarray, size);
        FputArray(output, datarray, size);
        ScriptArray(datarray, 'P', size);
        PutScore(score, string, datarray, size);
    }
    /*****
gettrans()
    {
        int trans;
        printf("\nDo you want a global transposition value? (1) yes ");
        scanf("%d", &globaltrans);
        if(globaltrans != 1)
        {
            globaltrans = FALSE;
            return(0);
        }
    }

```

```

        if(globaltrans)
        {
            printf("\nEnter transposition value for the array ");
            scanf("%d", &trans);
        }

        return(trans);
    }
    /*****
getstring(string)
    char string[];
    {
        printf("\nCurrent string:\n");
        fprintf(output, "\nCurrent string:\n");
        fgetline(input, string, 80);
        puts(string);
        fputs(string, output);
        printf("\nIt is %d characters long (including spaces)",
                strlen(string) );
        fprintf(output,
                "\nIt is %d characters long (including spaces)",
                strlen(string) );
    }
    /*****/
getarray(string, slen, datarray, usepunct)
    char string[];
    int slen, datarray[], usepunct;
    {
        int k, size;

        size = 0;
        for(k = 0; k < slen; k++)
        {
            if(string[k] == ' ' || string[k] == '\n')
            {
                lnwordcnt++;
                wordcnt++;
            }
            if(usepunct != FALSE)
            {
                if( isalpha(string[k]) || ispunct(string[k]) )
                    datarray[size++] = string[k];
            }
            else
                if( isalpha(string[k]) )
                    datarray[size++] = string[k];
        }
        return(size);
    }
    /*****/
showarray(datarray, size)
    int datarray[], size;
    {
        printf("\nThe string as integers:\n");
        fprintf(output, "\nThe string as integers:\n");
        PutArray(datarray, size);
        FputArray(output, datarray, size);
    }
    /*****/
Zero(datarray)
    int datarray[];
    {
        int j;

```

```

        for(j = 0; j < 80; j++)
            datarray[j] = 0;
    }
    /*****
PutScore(fp, string, datarray, size)
    FILE *fp;
    char string[];
    int datarray[], size;
    {
        fprintf(fp, "\n\n/* Text: \n");
        fprintf(fp, " \nts\n", string);
        fprintf(fp, " As pitch and rhythm\n*/\n");
        FscriptArray(fp, datarray, 'P', size);
        Putrhythm(fp, datarray, size);
    }
    /*****/
numvowel(string)
    char string[];
    {
        int j, k, numvowels;

        numvowels = 0;
        for(j = 0; j < strlen(string); j++)
            for(k = 0; k < 5; k++)
                if(toupper(string[j]) == vowels[k])
                    numvowels++;
        return(numvowels);
    }
    /*****/
numpunctuation(string)
    char string[];
    {
        int j, k, numpunct;

        numpunct = 0;
        for(j = 0; j < strlen(string); j++)
            for(k = 0; k < 10; k++)
                if(string[j] == punctuation[k])
                    numpunct++;
        return(numpunct);
    }
    /*****/
numconsonants(string)
    char string[];
    {
        int j, k, numconsonants, isvowel;

        numconsonants = 0;
        for(j = 0; j < strlen(string); j++)
        {
            isvowel = FALSE;
            for(k = 0; k < 5; k++)
                if(toupper(string[j]) == vowels[k] ||
                    isalpha(string[j]) == 0)
                    isvowel = TRUE;
            if(isvowel != TRUE)
                numconsonants++;
        }
        return(numconsonants);
    }
    /*****/
WhichVowels(fp, string, disarray, trans)

```

```

FILE *fp;
char string[];
int disarray[], trans;

{
    int j, k, m;

    m = 0;
    for(j = 0; j < strlen(string); j++)
        for(k = 0; k < 5; k++)
            if(toupper(string[j]) == vowels[k])
            {
                printf("%c ", string[j]);
                fprintf(fp, "%c ", string[j]);
                disarray[m++] = string[j] + trans;
            }
}
/*****
WhichPunctuation(fp, string, disarray, trans)
FILE *fp;
char string[];
int disarray[], trans;

{
    int j, k, m;

    m = 0;
    for(j = 0; j < strlen(string); j++)
        for(k = 0; k < 10; k++)
            if(string[j] == punctuation[k])
            {
                printf("%c ", string[j]);
                fprintf(fp, "%c ", string[j]);
                disarray[m++] = string[j] + trans;
            }
}
*****/
WhichConsonants(fp, string, disarray, trans)
FILE *fp;
char string[];
int disarray[], trans;

{
    int j, k, m, isvowel;

    m = 0;
    for(j = 0; j < strlen(string); j++)
    {
        isvowel = FALSE;
        for(k = 0; k < 5; k++)
            if(toupper(string[j]) == vowels[k] ||
               isalpha(string[j]) == 0)
                isvowel = TRUE;
        if(isvowel != TRUE)
        {
            printf("%c ", string[j]);
            fprintf(fp, "%c ", string[j]);
            disarray[m++] = string[j] + trans;
        }
    }
}

```

```

/*****
hasvowel(string)
char string[];
{
    int j, k, isvowel;
    for(j = 0; j < strlen(string); j++)
    {
        isvowel = FALSE;
        for(k = 0; k < 5; k++)
            if(toupper(string[j]) == vowels[k] ||
               isalpha(string[j]) == 0)
                isvowel = TRUE;
        if(isvowel == TRUE)
            return(TRUE);
    }
    return(FALSE);
}
/* END OF TXTPARSE.C */
*****/

```

### 程序运行实例

TXTPARSE.EXE

Filename for input text ? test.txt

Filename for output data ? junk

Filename for Script score ? junk.sc

Do you want a global transposition value? (1) yes 1

Enter transposition value for the array -45

Keep the punctuation? (1) = yes 1

Original complete text:

So i went to a concert the other day.

They were playing a piece by you know who.

It was the same old thing - bash bash bash

clang clang clang

c'mon baby give it to me.

Current string:

So i went to a concert the other day.

It is 38 characters long (including spaces)  
 The string as integers:  
 83 111 105 119 101 110 116 116 111 97 99 111 110 99 101 114  
 116 116 104 101 111 116 104 101 114 100 97 121 46

Max = 121 Min = 46

it has 11 vowels  
 it has 17 consonants  
 it has 1 punctuation  
 The vowels are:

o i a o a o e e o e a  
 P F5 B4 G4 F5 D#4 F5 G4 G4 F5 G4 D#4

The consonants are:

S w n t t c n c r t t h t h r d y  
 P C#3 C#6 E5 A#5 A#5 F4 E5 F4 G#5 A#5 A#5 A#4  
 P A#5 A#4 G#5 F#4 D#6

The punctuations marks are:

P C0

There are 9 words total  
 There are 9 words in this line

-----  
 Current string:  
 They were playing a piece by you know who.

It is 43 characters long (including spaces)  
 The string as integers:  
 84 104 101 121 119 101 114 101 112 108 97 121 103 110 103 97  
 112 105 101 99 101 98 121 121 111 117 107 110 111 119 119 104  
 111 46

Max = 121 Min = 46

it has 13 vowels  
 it has 20 consonants  
 it has 1 punctuation  
 The vowels are:

e e e a i a i e e o u o o  
 P G4 G4 G4 D#4 B4 D#4 B4 G4 G4 F5 B5 F5  
 P F5

The consonants are:

T h y w r p l y n g p c b y y k n w w h  
 P D3 A#4 D#6 C#6 G#5 F#5 D5 D#6 E5 A4 F#5 F4  
 P E4 D#6 D#6 C#5 E5 C#6 C#6 A#4

The punctuations marks are:

P C0

There are 18 words total  
 There are 9 words in this line

-----  
 Current string:  
 It was the same old thing - bash bash bash

It is 43 characters long (including spaces)

The string as integers:

73 116 119 97 115 116 104 101 115 97 109 101 111 108 100 116  
104 105 110 103 45 98 97 115 104 98 97 115 104 98 97 115  
104

Max = 119 Min = 45

it has 10 vowels

it has 22 consonants

it has 1 punctuation

The vowels are:

I a e a e o i a a a

P D#2 D#4 G4 D#4 G4 F5 B4 D#4 D#4 D#4

The consonants are:

t w s t h s m l d t h n g b s h b s h b s h

P A#5 C#6 A5 A#5 A#4 A5 D#5 D5 F#4 A#5 A#4 E5

P A4 E4 A5 A#4 E4 A5 A#4 E4 A5 A#4

The punctuations marks are:

-

P R

There are 28 words total

There are 10 words in this line

Current string:

clang clang clang

It is 18 characters long (including spaces)

The string as integers:

99 108 97 110 103 99 108 97 110 103 99 108 97 110 103

Max = 110 Min = 97

it has 3 vowels

it has 12 consonants

it has 0 punctuation

The vowels are:

a a a

P D#4 D#4 D#4

The consonants are:

c l n g c l n g c l n g

P F4 D5 E5 A4 F4 D5 E5 A4 F4 D5 E5 A4

The punctuations marks are:

There are 11 words total  
 There are 3 words in this line

---

Current string:  
 c'mon baby give it to me.

It is 26 characters long (including spaces)  
 The string as integers:  
 99 39 109 111 110 98 97 98 121 103 105 118 101 105 116 116  
 111 109 101 46

Max = 121 Min = 39

it has 7 vowels  
 it has 11 consonants  
 it has 2 punctuation  
 The vowels are:  
 o a i e i o e  
 P F5 D#4 B4 G4 B4 F5 G4

The consonants are:  
 c m n b b y g v t t m  
 P F4 D#5 E5 E4 E4 D#6 A4 C6 A#5 A#5 D#5

The punctuation marks are:  
 ' ,  
 P R C0

There are 37 words total  
 There are 6 words in this line

\*(text c.1987 Rodney Waschka II. Used by permission)

## 函数组: Drivel.c

- |                  |          |
|------------------|----------|
| 1. WordSummary   | MIC 6.30 |
| 2. MakeDrivel()  | MIC 6.31 |
| 3. ChopWords()   | MIC 6.40 |
| 4. ExtracWorrd() | MIC 6.39 |



## 功能

将输入的歌词文件中的词重新排列。

## 注释

本程序每次从歌词文件中读五行，然后将每行句子分解成单个的词(以空格隔开)并存盘。这些词可被随机读取构成“叠句”歌词。

1. 在程序中加入函数以实现随机重组输出歌词段落中的句子次序的功能。
2. 修改 MakeDrivel 函数以使用户可建一个表记录歌词中每一词出现的频率。
3. 通过运用类似于函数 Loopgen1()MIC 7.1 以及函数 Loopgen2()MIC 7.2 的功能，实现在原歌词文件的句子基础上，形成反复叠唱以及歌词同一结构循环出现，从而将原函数组变成一组格式/过程调整函数。

## 程序清单

```
DRIVEL.C
#include "text.h"
#include "fgetline.c"
main()
{
    int j;
    char string[80];
    FILE *fp;

    printf("\nFilename: ");
    gets(string);
    fp = fopen(string, "r");
    while(!feof(fp))
    {
        fgetline(fp, string, 80);
        puts(string);
        ChopWords(string);
    }
    WordSummary();
    fclose(fp);
    fp = fopen("junk.txt", "w");
    MakeDrivel(fp);
    fclose(fp);
}
/*===== MIC 6.30 =====*/
WordSummary()
{
    int j;
    for(j = 0; j < wct; j++)
        printf("%s\n", words[j]);
} /* end of WordSummary() function */
/*===== MIC 6.31 =====*/
MakeDrivel(fp)
    FILE *fp;
{
    int j, num, howmany, prt;
```

```

printf("\nHow many words do you want?\t");
howmany = getnum();
printf("\nWrite to file? (1) = yes\t");
prt = getnum();
randomize();
for(j = 0; j < howmany; j++)
{
    num = irand(0, wct - 1);
    printf("%s ", words[num]);
    if((j % 6) == 5)
        printf("\n");
    if(prt == 1)
    {
        fprintf(fp, "%s ", words[num]);
        if((j % 6) == 5)
            fprintf(fp, "\n");
    }
}
} /* end of MakeDrivel() function */
/*===== MIC 6.39 =====*/
ExtractWord(string, position)
char string[];
int position;
{
    int j = 0;
    char *word;

    word = (char *) malloc(80);
    while(!isspace(string[position]))
        word[j++] = string[position++];
    word[j++] = '\0';
    words[wct++] = word;
    return(position);
} /* end of ExtractWord() function */
/*===== MIC 6.40 =====*/
ChopWords(string)
char string[];
{
    int position = 0;
    while(position < strlen(string))
    {
        if(isspace(string[position]))
            position++;
        else
            position = ExtractWord(string, position);
    }
} /* end of ChopWords() function */
/* END OF DRIVEL.C */

```

## 程序运行实例

Filename: test.txt

So i went to a concert the other day.

They were playing a piece by you know who.

It was the same old thing - bash bash bash

clang clang clang

c'mon baby give it to me.

So  
i  
went  
to  
a  
concert  
the  
other  
day.  
They  
were  
playing  
a  
piece  
by  
you  
know  
who.  
It  
was  
the  
same  
old  
thing  
-  
bash  
bash  
bash  
clang  
clang  
clang  
clang  
c'mon  
baby  
give  
it  
to  
me.

How many words do you want? 100  
Write to file? (1) = yes 2

piece was clang who. i piece  
the by day. They piece who.  
was piece clang clang were other  
a to to clang a me.  
thing It thing concert know same  
me. a piece - it bash  
the who. same it bash to  
piece the They bash playing me.  
bash - to bash the the  
a a bash - know clang  
went went They bash give it  
thing to same the know to  
clang who. day. the It old  
old the who. the you thing  
clang clang the clang They -  
thing thing bash They It a  
So baby you baby

## 第七章 作曲总论

函数: Loopgen1() MIC7.1

### 功能

实现以两种方式将音符反复循环重奏——增音法与减音法。

### 注释

本函数适用于音乐中出现反复重奏并逐渐变化的某乐段的情形，它在改变循环长度的同时复制数份源乐段的整数代码。在增音法循环重奏方式中，循环先从一“核旋律”开始，而后在重奏过程中不断加长这段旋律。在减音法循环重奏方式中，则是先从核旋律开始而后每次重奏减短这段旋律。变量 Startnum 决定循环起始核旋律的长度，变量 fator 则控制每次重奏在核旋律中增加或删除的 6 音的代码值。变量 period 通知程序何时进入到下一次循环。函数中对某一乐段复制的循环用副本只在本函数内有效，并不返回到主程序。

### 编程提示

编写一程序应用 Loopgen1 函数实现下列功能：

1. 生成一系列无序、有 12 种音调的标准音符。
2. 根据步骤 1 生成的一系列音符定乐段的节拍值。
3. 先根据增音法或减音法复制出乐段的一系列循环副本，再定每一副本的节拍值。
4. 先将节拍值数据列分级并按逆序排列，然后将其作为指向音量坐标轴上各点的地址指针进行运算。

### 程序清单

```
/* LOOPGEN1.C ( additive/subtractive loop alterations) */
#include <stdio.h>
main()
{
    int num[15];
    int j;
    int period = 2;      /*alter loop every other iteration*/
    int factor = 3;      /* alter loop by group of three elements*/
    int startnum = 15;   /* sequence nucleus */
    int notes = 15;      /*number of values in original sequence*/
    int many = 10;       /*number of loop copies to generate*/
    int mode = 0;        /*toggle subtractive mode*/
    void Loopgen1();
}
```

```

printf("a array of ascending integers\n");
for (j = 0; j < notes; j++)
{
    num[j] = j + 1;
    printf("%d ", num[j]);
}

/* call loop generator in subtractive mode */
Loopgen1(num, period, factor, startnum, notes, many, mode);

mode = 1;          /* toggle additive mode */
startnum = 1;      /* sequence nucleus */
factor = 4;        /* change elements factor */

/* call loop generator in additive mode */
Loopgen1(num, period, factor, startnum, notes, many, mode);

) /* end of main */
/*===== MIC 7.1 =====*/
/* Loopgen1() function (generates a given number of copies
of an input sequence while processing it in
one of two length-altering modes: additive,
or subtractive */

void Loopgen1(num, period, factor, startnum, notes, many, mode)
int num[], period, factor, startnum, notes, many, mode;
{
    int alter;      /* stores current sequence length */
    int incr = factor; /* stores # of elements to add/subtract */
    int iter = period; /* counts cycles, controls loop-length */
    int copies;     /* loop index */
    int dup;        /* loop index */

    if (mode)
    {
        alter = startnum;

        printf("\nadditive mode loop processing\n");
    }
    else
    {
        alter = notes;
        printf("\nsubtractive mode loop processing\n");
    }
    for (copies = 0; copies < many; copies++)
    {
        if (copies >= iter)
        {
            if (mode)
                alter = startnum + incr;
            else
                alter = notes - incr;
            if (alter > notes)
                alter = notes;
        }
        for (dup = 0; dup < alter; dup++)
            printf("%d ", num[dup]);
        printf("\n");
        if (copies == many)
            break;
        if (copies < iter)
            continue;
    }
}

```

```

        incr += factor;
        iter += period;
    }
} /* end of Loopgen1() function */
/*=====*/
/* END OF LOOPGEN1.C */

```

## 程序运行实例

```

LOOPGEN1.EXE
a array of ascending integers
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
subtractive mode loop processing
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3
1 2 3
additive mode loop processing
1
1
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10 11 12 13
1 2 3 4 5 6 7 8 9 10 11 12 13
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

```

## 函数: Loopgen2() MIC7.2

### 功能

将一些乐段副本进行系统化循环重奏，改换过的循环体将以逆序排列。

### 注释

本函数用于音乐的不同型態格式，它在将一些指定的循环元素进行(顺向或逆向)循环的同时，将该列整数复制出一定数目的副本。控制变量如下：变量 many——要返回的循环体副本的数目，变量 factor——要从循环终止转换到循环起始处的循环元素的数目，变量 reverse——将经过换位的循环元素逆序排列的控制信号。

当本函数用于侧重于将不同音阶的音符进行循环调整的音乐作曲中时，最好再增加一控制变量(同 MIC7.1 中的 Loopgen1 函数)来控制所有副本的循环频率。例如，如果你想将一某一段旋律的音符的各项参数各自独立地进行循环，就可应用上述控制变量来实现。

比如说, 音高参数可能每隔一次循环中将两个元素变位, 而同时节拍参数在每一次循环中将三个元素变位等等。(注意函数中的数组 temp() 必须与主程序中的数组 x[] 相配)。

### 编程提示

编写一交互式程序应用 Loopgen2() 实现以下功能:

1. 允许用户输入所有控制变量的值。
2. 允许用户对某一段旋律的各音的音高、节拍、发音方式和音量等各参数进行旋转运算。
3. 将一含有旋律同奏的乐谱文件进行格式化并存盘。

### 程序清单

```
/* LOOPGEN2.C (Loop-element rotation function) */
#include <stdio.h>
main()
{
    int x[16];          /* stores loops */
    int j;              /* loop index */
    int numvals=15;     /* sequence total */
    int factor=6;       /* number of values to shift */
    int many=6;         /* number of loops to return */
    int reverse=1;      /* toggles reverse order shift */
    void Loopgen2();

    printf("original integer sequence:\n\n");
    for (j = 1; j <= numvals; j++)
    {
        x[j] = j;
        printf("%d ", x[j]);
    }
    printf("\n\n%d %s %d %s %s", many, "copies, with", factor,
        "values shifted \n(reverse order)",
        "during each cycle:\n\n");

    Loopgen2(x, numvals, factor, many, reverse);
} /* end of main */
/*===== MIC 7.2 =====*/
/* Loopgen2() function (shifts a specific number of values
   from the end of a sequence to the beginning
   while generating a given number of loop copies;
   shifted values can be placed in retrograde.)*/

void Loopgen2(x, numvals, factor, many, reverse)
int x[], numvals, factor, many, reverse;
{
    int temp[16]; /* stores values to be shifted; it must be
                   the same size as main routine array x[] */
    int copies;   /* loop index */
    int hold;     /* loop index, pointer to array temp[] */
    int group;    /* loop index */
    int shift;    /* loop index, pointer to array x[] */
    int replace;  /* loop index, pointer to array x[] */
    int currloop; /* loop index, pointer to array x[] */
```



```

    for (copies = 1; copies <= many; copies++)
    {
        for (hold = 0; hold < factor; hold++)
            temp[hold + 1] = x[numvals - hold];
        for (group = 1; group <= factor; group++)
        {
            for (shift = numvals; shift >= 1; shift--)
                x[shift] = x[shift - 1];
        }
        for (replace = 1; replace <= factor; replace++)
            if (reverse)
                x[replace] = temp[replace];
            else
                x[replace] = temp[factor + 1 - replace];
        for (currloop = 1; currloop <= numvals; currloop++)
            printf("%d ", x[currloop]);
        printf("\n\n");
    }
} /* end of Loopgen2() function */
/*-----*/
/* END OF LOOPGEN2.C */

```

### 程序运行实例

```

LOOPGEN2.EXE

original integer sequence:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

6 copies, with 6 values shifted
(reverse order) during each cycle:

15 14 13 12 11 10 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 15 14 13 12 11 10 1 2 3
3 2 1 10 11 12 9 8 7 6 5 4 15 14 13
13 14 15 4 5 6 3 2 1 10 11 12 9 8 7
7 8 9 12 11 10 13 14 15 4 5 6 3 2 1
1 2 3 6 5 4 7 8 9 12 11 10 13 14 15

```

### 函数组: Primops.c

1. Primintv() MIC7.3
2. Primnum() MIC7.4

### 功能

生成两个数组:

1. 存放一系列限定范围的质数数列。

2. 存放上述质数数列各数之间的间隔数。

### 注释

函数 `Primintv()` 通过调用函数 `Primnum()` 来查看给定范围内以升序连续排列的一系列整数中各数是否是质数。如果是，则存入数组 `X[]` 中，然后函数 `Primintv` 将上一个质数与该质数相减，结果存入数组 `Y[]` 中。

有趣的是，当扩大函数查看整数的范围时，在上限附近质数与质数之间的间隔数加大而前面的质数间的间隔数却保持不变，利用这一特点，我们可在节拍调整中通过连续改变函数查看整数上限得到一系列不断加大的质数与质数之间的间隔数，将此间隔数转化为节拍值就得到了一种音长逐渐加长的节拍形式。

### 编程提示

1. 修改主程序以连续产生在下述范围内进行质数查找而得到的一系列质数间隔序列：1-300, 300-600, 600-900, 900-1200, 1200-1500。仔细观察不同间隔数的变化频率。
2. 编写一交互式程序实现下列功能：
  - a. 允许用户输入变量 `low` 和变量 `high` 的值。
  - b. 可以将返回的质数间隔数数组按不同的音乐参数要求的不同查找范围分级。
  - c. 为指定元素表设置指针。

### 程序清单

```
/* PRIMOPS.C (prime number operations) */
#include <stdio.h>
#include <math.h>
main()
{
    int x[1000]; /* list of prime numbers */

    int y[1000]; /* list of numeric intervals between primes */
    int j;      /* loop index */
    int low = 1; /* smallest integer within test range */
    int high = 1000; /* largest integer within test range */
    int pcnt;    /* prime number counter, pointer to x[], y[] */
    int Primintv();

    printf("prime numbers within range %d to %d : \n", low, high);
    pcnt = Primintv(x, y, low, high);
    for (j = 0; j < pcnt; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        printf("%d ", x[j]);
    }
    printf("\n\nnumeric intervals between primes: \n");
    for (j = 0; j < pcnt - 1; j++)
    {

```

```

        if (j % 10 == 0)
            printf("\n");
        printf("%d ", y[j]);
    }
    printf("\n\n\n");
} /* end of main */
/*===== MIC 7.3 =====*/
/* Primintv() function (creates an array of primes and
and an array of numeric intervals separating
the primes */

int Primintv(x,y,low,high)
int x[],y[],low,high;
{
    int pcnt = 0; /* prime number counter */
    int num;      /* number to be tested for prime */
    int prim;     /* flag indicating a prime number */
    int Primnum();

    for (num = low; num <= high; num++)
    {
        prim = Primnum(num);
        if (prim != 1)
            continue;
        x[pcnt] = num;
        if (pcnt > 0)
            y[pcnt - 1] = x[pcnt] - x[pcnt - 1];
        pcnt += 1;
    }
    return(pcnt);
} /* end of Primintv() function */
/*===== MIC 7.4 =====*/
/* Primnum() function (locates and returns prime numbers
from a stream of ascending integers*/

int Primnum(num)
int num;
{
    int prim = 1; /* prime number flag */
    int divisor; /* loop index, divisor for prime test */
    int limit;   /* prime test limit */
    float sqrt();
    limit = sqrt(num * 1.0);
    for (divisor = 2; divisor <= limit; divisor++)
        if (num % divisor == 0)
        {
            prim = 0;
            break;
        }
    return(prim);
} /* end of Primnum() function */
/*=====*/
/* END OF PRIMOPS.C */

```

### 程序运行实例

PRIMOPS.EXE

prime numbers within range 1 to 1000 :

```

1 2 3 5 7 11 13 17 19 23
29 31 37 41 43 47 53 59 61 67
71 73 79 83 89 97 101 103 107 109
113 127 131 137 139 149 151 157 163 167
173 179 181 191 193 197 199 211 223 227
229 233 239 241 251 257 263 269 271 277
281 283 293 307 311 313 317 331 337 347
349 353 359 367 373 379 383 389 397 401
409 419 421 431 433 439 443 449 457 461
463 467 479 487 491 499 503 509 521 523
541 547 557 563 569 571 577 587 593 599
601 607 613 617 619 631 641 643 647 653
659 661 673 677 683 691 701 709 719 727
733 739 743 751 757 761 769 773 787 797
809 811 821 823 827 829 839 853 857 859
863 877 881 883 887 907 911 919 929 937
941 947 953 967 971 977 983 991 997

```

numeric intervals between primes:

```

1 1 2 2 4 2 4 2 4 6
2 6 4 2 4 6 6 2 6 4
2 6 4 6 8 4 2 4 2 4
14 4 6 2 10 2 6 6 4 6
6 2 10 2 4 2 12 12 4 2
4 6 2 10 6 6 6 2 6 4
2 10 14 4 2 4 14 6 10 2
4 6 8 6 6 4 6 8 4 8
10 2 10 2 6 4 6 8 4 2
4 12 8 4 8 4 6 12 2 18
6 10 6 6 2 6 10 6 6 2
6 6 4 2 12 10 2 4 6 6
2 12 4 6 8 10 8 10 8 6
6 4 8 6 4 8 4 14 10 12
2 10 2 4 2 10 14 4 2 4
14 4 2 4 20 4 8 10 8 4
6 6 14 4 6 6 8 6

```

## 函数: TraNtabl() MIC7.5

### 功能

生成一个选音变化概率的一阶表, 并根据此表定旋律中的各音符次序。

### 注释

用一个二维矩阵存放各音的出现频率。在选音过程中, 当前音符选择就通过音的出现频率, 与前一次选音相关。各音出现频率可通过矩阵 P() 存入, 或作为程序数据文件读入, 也可以通过用户 INPUT 指令输入。

为了使输出数据准确无误, 矩阵中每一列中各数之和应为 11.0, 而且, 数据表中的数不可随意输入以防止出现死循环。例如, 若将表 7-1 中的 F 音后选 C 音的概率变为 0.0, 则 F 音后选 C 音的概率变为 100%, 而且 F 音后选 G 音的概率也变为 0.0, 这样, 一旦选中了 F 音就陷入了死循环。

		Last pitch			
		C	F	G	B
Next pitch	C	0.0	0.5	0.1	0.0
	F	0.1	0.2	0.5	0.1
	G	0.6	0.3	0.1	0.5
	B	0.3	0.0	0.3	0.4

表 7-1 图例

每一行中各数代表上一次选为某音的概率，每一列中各数代表下一次选音为某音的概率。例如，若上一次选的音为 B(第 4 列)，则本次选音仍选 B 的概率即为 0.4%，这一原则可应用于三维、四维或多维矩阵，但随着矩阵维数的增大，理解起来也将愈加困难。

### 编程提示

编写一交互式程序应用 Trantabl()函数，并允许用户输入矩阵各元素值并将输入值作为各种音乐参数的地址指针。

### 程序清单

```
/* TRANTABL.C (table of 1st-order transitional probs) */
#include <stdio.h>
#include <math.h>

main()
{
    int startpitch = 2, total = 100;
    void Trantabl();
    Trantabl(startpitch, total);
} /* end of main */

/*===== MIC 7.5 =====*/
/* Trantabl() function (generates pitch sequences in
conformity with probabilities stored in
a 2-dimensional matrix; to change the probs,
edit the static float ptabl[][] array. */

void Trantabl(nextpitch, total)
int nextpitch;
{
    int j9, k; /* loop indices */
    int seed = -3944;
    double u; /* random number > 0 & < 1.0 */
    float t; /* threshold test value */

    /* probability table */
    static float ptabl[4][4] =
        { 0.0, .5, 0.0, .1, .2, .5, .1, .6, .3, .1, .5, .3, 0.0, .3, .4 };

    /* pitch table */
    static char *pitch[] = {"C ", "F ", "G ", "B "};

    srand(seed);
    printf("\ncurrent probability table:\n\n");
    for (j9 = 0; j9 < 4; j9++)
```

```

        {
            for (k = 0; k < 4; k++)
                printf("%1f ", ptabl[j9][k]);
            printf("\n");
        }
    printf("\n%s%s", "pitch sequence beginning with ",
           pitch[nextpitch], "\n");
    for (j9 = 0; j9 < total; j9++)
    {
        if (j9 % 10 == 0)
            printf("\n");
        printf("%s ", pitch[nextpitch]);
        u = rand() / 32767.;
        t = 0;
        for (k = 0; k < 4; k++)
        {
            t = ptabl[k][nextpitch] + t;
            if (u <= t)
            {
                nextpitch = k;
                break;
            }
        }
    }
} /* end of Trantabl() function */
/*-----*/
/* END OF TRANTABL.C */

```

### 程序运行实例

TRANTABL.C

current probability table:

0.0	0.5	0.0	0.1
0.2	0.5	0.1	0.6
0.3	0.1	0.5	0.3
0.0	0.3	0.4	0.0

pitch sequence beginning with G :

G	B	F	F	C	C	C	C	C	F
F	F	C	F	F	C	G	B	F	F
F	F	C	F	C	G	G	B	F	C
C	F	F	F	C	G	B	G	G	G
G	B	F	C	C	G	G	F	C	C
F	F	C	C	G	G	B	F	F	C
C	C	C	F	C	G	F	C	C	C
F	C	F	C	G	B	F	F	F	F
C	C	F	F	F	F	F	C	G	G
G	G	B	F	C	F	C	G	G	G

### 函数组: Valuprob.c

1. Probtbl( ) MIC7.6
2. Probcalc( ) MIC7.7

## 功能

根据由出现频率而定的加权值生成一系列无序整数。

## 注释

函数 Probtabl() 可根据存放在静态整型数组 wtabl[] 中的各元素加权值建立一个各元素出现频率的数表。因为所有加权值都是相对的，所以其具体数值无关紧要(函数 Probcalc() 根据随机数或程序来定其值)。

函数 Probcalc() 将一系列从 1 到 n 的 n 个无序整数根据存于数组 wtabl() 中的加要值返回其出现频率数。输出数组 x[] 作为多音乐参数的指针变量数组。

## 编程提示

1. 编写一程序实现下列功能:
  - a. 允许用户输入变量 probs、变量 total，以及数组 wtabl() 中各数组元素的值。
  - b. 允许用户从磁盘文件中读取概率加权值并存入数组 wtabl[] 中。
  - c. 将输出的指针数组作为任一指定音乐参数的一系列指针。
2. 修改 MIC2.23 中的 Freqtabl() 函数，以记录一源文件中各阿拉伯字母的出现的频率。先输入一段英文诗句，然后根据字母出现频率作为加权值生成一系列音符。在法文诗、意大利诗、德文诗上反复重复上述过程，并注意上述过程用于各种不同文字的诗时的音调的不同变化。

## 程序清单

```
/* VALUPROB.C (reads relative weights into a cumulative
   probability table, then generates an
   integer sequence with occurrence fre-
   quencies in relative proportion to table
   weights.)*/
#include <stdio.h>
main()
{
    int x[200];
    int j;
    int probs = 5; /* number of probabilities */
    int total = 200; /* number of integers to return */
    int wsum; /* sum of probabilities */
    int seed = 31231;
    int Probtabl();
    void Probcalc();
    static int wtabl[5] = {1,5,10,15,20};

    srand(seed);
    wsum = Probtabl(wtabl,probs);
    Probcalc(x,wtabl,probs,total,wsum);
    printf("\nhere is the sequence generated by the table:\n");
```

```

    for (j = 0; j < total; j++)
    {
        if (j % 20 == 0)
            printf("\n");
        printf("%d ", x[j]);
    }
} /* end of main */
/*===== MIC 7.5 =====*/
/* Probtbl() function (converts probability table
                      weights to cumulative form) */
int Probtbl(wtbl[], probs)
int wtbl[], probs;
{
    int wsum = 0; /* sum of probabilities */
    int k;        /* loop index */

    for (k = 0; k < probs; k++)
    {
        wtbl[k] = wtbl[k] + wsum;
        wsum = wtbl[k];
    }
    return(wsum);
} /* end of Probtbl() function */
/*===== MIC 7.7 =====*/
/* Probcalc() function (generates integers in relative
                      proportion to weights contained
                      in a cumulative table. */
void Probcalc(array1, array2, probs, total, wsum)
int array1[], array2[], probs, total, wsum;
{
    int k, l; /* loop indices */
    int u;    /* random integers */

    for (k = 0; k < total; k++)
    {
        u = rand() % wsum; /* get normalized random integer */
        for (l = 0; l < probs; l++)
            if (u < array2[l])
            {
                array1[k] = l + 1;
                break;
            }
    }
} /* end Probcalc() function */
/*=====*/
/* END OF VALUPROB.C */

```

## 程序运行实例

VALUPROB.EXE

here is the sequence generated by the table:

```

5 4 4 3 5 4 5 3 5 4 3 3 2 4 5 2 3 4 3 4
3 5 3 5 1 2 5 5 4 4 5 5 5 2 2 4 3 5 4 5
4 5 5 5 3 5 5 5 5 5 1 4 2 1 3 4 2 1 5 5
4 4 5 5 5 4 4 4 5 3 4 3 3 5 5 5 5 4 5 3
5 4 1 4 3 4 4 3 3 5 4 3 5 4 5 5 2 4 4 5
3 2 5 4 5 3 5 3 5 5 5 2 5 4 3 4 4 2 5 2
5 5 2 4 4 4 5 4 4 3 4 3 5 3 1 4 5 5 5 4

```



```

5 2 5 2 4 4 5 4 2 3 5 5 2 4 3 4 5 4 3 2
5 3 3 2 5 2 4 4 4 5 5 2 5 4 3 4 5 5 4 4
5 5 3 4 2 3 4 4 5 2 4 5 4 4 4 4 5 4 5 5

```

## 函数: Randwalk() MIC7.8

### 功能

模拟一在上下限范围内的简单、双向随机遍历路径。

### 注释

本函数在从一起点遍历某一连续整数序列时，以“掷硬币法”决定每一步是沿数值增大的方向还是沿数值减少的方向。在而且指针不可能在某一步停在原处不动，因为每一次“掷硬币法”必有一确定结果决定前进还是后退。

如果整数列中含有重复值，则可选根据作曲规则将该列整数存入一数组，然后以 Randwalk()函数的输出作为地址指针遍历该数组。

### 编程提示

1. 编写一程序实现以下功能:
  - a. 从磁盘中存储的文件读取一系列经过谱曲的音符。
  - b. 将这些音符的代码存入一数组中。
  - c. 允许用户任选一遍历起始位置。
  - d. 生成一包含有 10 个值的任意遍历路径并以数组形式输出。
  - e. 在输出数组中的整数作为生成其他音乐参数的概率加权值(可用函数 VALUPROB.C 实现此项功能)。
2. 改编上述程序以从磁盘中读取一个歌词文本文件(只需加入第六章中所述的字符串处理功能即可)，允许用户生成一含有遍历输入歌词文件中的各词的随机路径的输出文件。那么，MIC7.8 中的 Randwalk 函数的输出就将与 MIC6.3 中的 Randline()函数以及 MIC6.7 中的 Randword()函数的输出完全相同。

### 程序清单

```

/* RANDWALK.C (simulates a simple, bi-directional,
   random walk which is bounded at both ends)*/
#include <stdio.h>
main()
{
    int total = 200; /* take 200 steps */
    int currloc = 7; /* start at position seven */
    int seed = -8675;
    void Randwalk();

    srand(seed);
    printf("here is the random walk:\n");

```

```

    Randwalk(total,currloc);
} /* end of main */
/***** MIC 7.8 *****/
/* Randwalk() function */
void Randwalk(total,currloc)
int total,currloc;
{
    int dir;      /* step direction */
    int nextstep; /* step location */
    int j;        /* loop index */
    float u;      /* random number > 0 & < 1.0 */

    for (j = 0; j < total; j++)
    {
        if (j % 15 == 0)
            printf("\n");
        printf("%d ",currloc);
        dir = 1;
        u = rand() / 32767.;
        if (u < .5)
            dir = -1;
        nextstep = currloc + dir;
        if (nextstep > 15)
            nextstep = 14;
        else
            if (nextstep < 1)
                nextstep = 2;
        currloc = nextstep;
    }
} /* end of Randwalk() function */
/***** */
/* END OF RANDWALK.C */

```

### 程序运行实例

here is the random walk:

```

7 8 9 10 11 10 11 10 11 12 13 14 13 12 11
10 11 12 11 10 9 10 9 10 9 10 11 12 13 12
13 14 15 14 13 14 13 12 11 10 9 10 9 8 9
10 11 12 13 14 13 14 13 12 11 10 9 8 7 8
9 10 11 12 13 14 15 14 13 12 13 12 13 12 13
12 13 14 15 14 13 14 13 12 13 14 13 12 13 12
11 10 11 12 11 12 11 12 13 14 13 12 13 12 11
12 11 10 9 8 9 8 9 10 11 10 11 12 13 12
11 12 11 12 13 12 11 12 11 12 13 14 15 14 13
12 13 14 15 14 13 12 13 14 15 14 15 14 13 14
15 14 15 14 15 14 13 14 15 14 15 14 15 14 15
14 13 14 15 14 13 14 15 14 13 14 15 14 13 14
15 14 15 14 13 12 11 12 13 14 13 12 13 12 13
14 13 14 15 14

```

函数: Matwalk() MIC7.9

功能

模拟一个在二阶矩阵范围内的多向随意遍历路径。

### 注释

本函数不同于 MIC7.8 中的 Randwalk 函数，因为其遍历指针可以在矩阵方块内任意向上、向下沿各边、或对角线移动以及原地不动。矩阵的行数、列数以及指针的起始点在主程序中给定。编程者可通过改变矩阵的行数、列数来调整由任一随机遍历生成的一系列数的特征。例如，当一随机遍历生成的一系列数作为指向音高坐标轴的指针时，改变矩形行、列数将会引起显著变化。如下所示：

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90

这一 6 行到 15 列矩阵可生成一段旋律，其间隔数(+或-)为 1,14,15 和 16。

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32
33	34	35	36

这一 9 行到 4 列矩阵也生成一段旋律，间隔数为 3, 4 和 5。

### 编程提示

1. 修改全程序以便将一整型数据文件读入一个二维矩阵。
2. 将不同行、列数的矩阵的输出数据分别作为节拍参数、音量参数以及发音方式参数的地址指针。

### 程序清单

```
/* MATWALK.C (2-dimensional random walk around a matrix)*/
#include <stdio.h>

#define rows 10 /* (edit row and column sizes to try */
#define cols 10 /* other matrix proportions) */
main()
{
    int w[rows][cols]; /* random walk matrix */
    int count = 0; /* aids screen printing of matrix*/
    int total = 200; /* take 200 steps */
    int j,k; /* loop indices */
    int xloc = 5,yloc = 5; /* walk start coordinates */
    int seed = 19274;
    void Matwalk();
```

```

srand(seed);
printf("the matrix for this walk:\n");
for (j = 0; j < rows; j++)
{
    for (k = 0; k < cols; k++)
    {
        w[j][k] = (j * cols) + k + 1;
        count++;
        if (count % cols == 0)
            printf("%d\n", w[j][k]);
        else
            printf("%d ", w[j][k]);
        if (count <= 9)
            printf(" ");
    }
    printf("\nstart coordinates are: row %d: column %d;\n",
        xloc+1, yloc+1);
    Matwalk(w, total, xloc, yloc);
} /* end of main */
/*===== MIC 7.9 =====*/
/* Matwalk() function */
void Matwalk(w, total, xloc, yloc)
int w[rows][cols], total, xloc, yloc;
{
    int xstep; /* step direction random integer: -1,0,or +1 */
    int ystep; /* step direction random integer: -1,0,or +1 */
    int k; /* loop index */

    for (k = 0; k < total; k++)
    {
        if (k % 15 == 0)
            printf("\n");
        printf("%d ", w[xloc][yloc]);
        xstep = rand() % 3 - 1;
        ystep = rand() % 3 - 1;
        xloc += xstep;
        yloc += ystep;
        if (xloc < 0)
            xloc = 1;
        else
            if (xloc >= rows)
                xloc = rows-1;
        if (yloc < 0)
            yloc = 1;
        else
            if (yloc >= cols)
                yloc = cols-1;
    }
    printf("\n\n");
} /* end of Matwalk() function */
/*=====*/
/* END OF MATWALK.C */

```

## 程序运行实例

MATWALK.EXE

```

the matrix for this walk:
1  2  3  4  5  6  7  8  9  10
11 12 13 14 15 16 17 18 19 20

```

```

21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100

```

start coordinates are: row 6; column 6;

```

56 47 47 58 58 59 48 57 26 25 14 25 14 14 23
23 24 13 3 3 2 11 2 11 1 2 11 2 1 2
2 13 24 25 35 25 14 4 13 4 13 14 13 17 21
21 11 12 11 2 12 23 23 12 33 33 33 42 33 43
33 43 54 43 43 54 64 75 72 74 74 85 84 73 63
64 53 52 63 74 73 62 63 54 63 53 63 72 71 72
72 83 83 92 91 91 81 81 81 71 72 63 64 65 64
53 63 62 63 54 54 53 62 61 72 71 72 71 81 81
82 81 81 72 72 71 62 63 52 52 51 42 51 52 52
51 62 51 42 42 33 42 32 32 33 44 14 43 52 51
62 62 72 81 72 73 82 92 91 92 93 84 84 75 64
54 53 62 52 43 34 24 13 2 1 12 13 13 23 24
35 24 34 23 23 13 23 14 25 35 24 25 15 5 16
5 15 25 14 14

```

## 函数: Voss MIC7.10

### 功能

生成一系列有分数特性( $1/f$  噪音分贝数)的整数。

### 注释

本函数生成一系列自相似的有序整数列。许多作曲工作者认为有分数特性的整数列比其他各种随机生成的整数列更有用,因为它们更接近于古典音乐中的一些结构模式。

本程序的输出数据总在微观上或宏观上表现出很高程度的相关性,因为宏观上的模式总是一内部模式相似的。而且,一组从一系列数的尾部选出的数值总是跟随从一系列数的首部选出的数值有很大相似性。(若要验证这一点,只需比较各数列块内数值的个数以及乐音音程变化型式即可。)

函数返回的每一  $1/f$  值都是相对于前一返回值的参考值,而且所有的数值元素都以对数形式与以前的返回值相关联,这就是为什么本过程被称为有长期记忆功能的道理。

### 编程提示

1. 编写一交互式程序实现下述功能: 允许用户输入元素表及控制变量,并提供为一具分数特性的乐句生成音高、节拍、音量以及发音方式等各参数并存盘的功能。

将此乐句改编为可在独奏乐器上演奏(例如, 单簧管)。

2. 编写一交互式程序用分数算法建立一复音音乐结构并存盘。在一电子乐器上演奏这段曲子。
3. 运行上述程序, 设计输入数据以便某输出文件可分别在独奏乐器上和音乐合成器上演奏。

## 程序清单

```
/* VOSS.C (Fractals routine)*/
#include <stdio.h>
main()
{
    int x[200];
    int last = 24; /* last value generated */
    int total = 200, seed = 3111;
    void Voss();

    srand(seed);
    printf("\nhere is the resulting fractal sequence:\n");
    Voss(x, total, last);
} /* end of main */
/***** MIC 7.10 *****/
/* Voss() fractals function (generates 1/f fractional
                             noise patterns) */
void Voss(x, total, last)
int x[50], total, last;
{
    int fract; /* current 1/f value */
    int halfvals; /* 1/2 the number of possible values */
    int temp; /* temporary computation storage */
    int k; /* loop index */
    float prob; /* 1/number of possible values */
    float u; /* random number > 0 & < 1.0 */

    for (k = 0; k < total; k++)
    {
        if (k % 15 == 0)
            printf("\n");
        fract = 0;
        halfvals = 16;
        prob = .03125;
        while (halfvals >= 1)
        {
            temp = last/halfvals;
            if (temp == 1)
                last = last - halfvals;
            u = rand() / 32767.;
            if (u < prob)
                temp = 1 - temp;
            fract = fract + temp * halfvals;
            halfvals /= 2;
            prob *= 2.;
        }
        x[k] = fract;
        printf("%d ", x[k]);
        last = fract;
    }
} /* end of Voss() fractals function */
/***** */
/* END OF VOSS.C */
```

## 程序运行实例

VOSS.EXE

here is the resulting fractal sequence:

```
27 30 29 28 29 29 30 30 31 31 30 28 28 30 29
31 25 28 31 30 25 26 26 26 27 30 26 26 26 26
27 27 24 26 27 26 25 25 8 12 12 13 12 13 13
12 31 25 26 8 0 21 21 20 23 23 23 22 23 21
23 21 22 23 18 17 16 21 22 20 23 31 30 29 29
31 30 31 28 28 30 30 30 30 22 31 30 31 31
31 31 29 30 30 28 24 26 26 24 24 25 25 8 15
14 14 12 13 15 15 13 12 12 12 10 8 10 5 9
9 8 9 9 11 15 15 15 15 14 15 30 28 24 25

29 25 25 25 24 25 26 28 28 31 30 31 30 28 28
29 31 28 20 20 25 24 25 25 24 24 25 24 24 25
28 31 29 28 29 31 30 31 29 30 30 21 21 22 22
23 23 30 31 29 28 29 14 14 12 12 15 15 12 28
29 29 25 29 28
```

## 函数组: Rdintchd.c

1. Rdintchd() MIC7.11
2. Printpitch() MIC3.1

### 功能

生成一组声音密度不同的和弦。用一系列规定范围、任意音程范围的音组成。

### 注释

本函数指定在提供一个脚本和弦文件以便从中精炼出一段输出旋律。Rdintchd()函数通过从一规定范围、音程连续变化的高、低音中随机选取一些音来构成不同长度的音阶结构。和弦基音(从 C 到 B)、始音阶(1-3)、和弦内的音数(3-9)以及音程范围都在主程序中给定。逐渐增加的一系列音程的最后选一将根据主程序给出的值来定。

例如,若音量 small 的值被定为 4,而音量 large 定为 7,则最后生成的和弦将全部由无序的主三度音程、全四度音程、增四度音程以及全五度音程组成。

### 编程提示

1. 修改主程序以使用户能够输入主要控制变量的值,不要忘记加入提示及查错语句,以便在当用户输入了将使程序运行无效的值时,显示出错信息(例如,若输入了太多的和弦音以及太大的音程数将很快使算出的音超出音符表的范围)。

2. 加一段程序以实现将生成的和弦文件存盘。在文件中每一和弦的后面标志位以有表明和弦发音方式。
3. 加一段程序从一和弦文件中随机选取和弦并在屏幕上显示，然后根据这列和弦生成一系列各音符为任意次序的旋律。
4. 将和弦文件的输出数据作为其他音乐参数的地址指针。

## 程序清单

```

/* RDINTCHD.C (generates a group of variable-density chords
   having in common a random-order interval
   aggregate. */
#include <stdio.h>
main()
{
    int j; /* loop index */

    int chordtotal = 15;
    int seed = 9121;
    void Rdintchd();
    int root; /* lowest chordtone "C" - "B" */
    int octave; /* chord root octave register */
    int chordmems; /* number of tones in chord */
    int small; /* smallest allowable interval in chord */
    int large; /* largest allowable interval in chord */
    int intvalrange; /* range of allowable interval-sizes */

    srand(seed);
    for (j = 0; j < chordtotal; j++)
    {
        printf("chord %d:", j+1);
        /* select random values */
        /* for primary parameters */
        root = rand() % 12 + 1;
        octave = rand() % 3 + 1;
        chordmems = rand() % 7 + 2;
        small = rand() % 2 + 1;
        large = rand() % 6 + 5;
        intvalrange = large - small + 1;

        Rdintchd(root, octave, chordmems, small, intvalrange);
        printf("\n");
    }
} /* end of main */
/***** MIC 7.11 *****/
/* Rdintchd() function */
void Rdintchd(int root, int octave, int chordmems, int small, int intvalrange)
{
    int intval, chordtone, k;

    static char *pitch[] = {
        "C0 ", "C#0 ", "D0 ", "D#0 ", "E0 ", "F0 ", "F#0 ", "G0 ", "G#0 ",
        "A0 ", "A#0 ", "B0 ", "C1 ", "C#1 ", "D1 ", "D#1 ", "E1 ", "F1 ",
        "F#1 ", "G1 ", "G#1 ", "A1 ", "A#1 ", "B1 ", "C2 ", "C#2 ", "D2 ",
        "D#2 ", "E2 ", "F2 ", "F#2 ", "G2 ", "G#2 ", "A2 ", "A#2 ", "B2 ",
        "C3 ", "C#3 ", "D3 ", "D#3 ", "E3 ", "F3 ", "F#3 ", "G3 ", "G#3 ",
        "A3 ", "A#3 ", "B3 ", "C4 ", "C#4 ", "D4 ", "D#4 ", "E4 ", "F4 ",
        "F#4 ", "G4 ", "G#4 ", "A4 ", "A#4 ", "B4 ", "C5 ", "C#5 ", "D5 ",
        "D#5 ", "E5 ", "F5 ", "F#5 ", "G5 ", "G#5 ", "A5 ", "A#5 ", "B5 ",
        "C6 ", "C#6 ", "D6 ", "D#6 ", "E6 ", "F6 ", "F#6 ", "G6 ", "G#6 ",
        "A6 ", "A#6 ", "B6 ", "C7 " };
}

```



```

    chordtone = root;
    for (k = 0; k < chordmems; k++)
    {
        printf("%s ", pitch[chordtone + ((octave-1)*12)]);
        intval = rand() % intvalrange + small;
        chordtone = chordtone + intval;
    }
    printf("\n");
} /* end of Rdintchd() function */
/*=====*/
/* END OF RDINTCHD.C */

```

## 程序运行实例

```

RDINTCHD.EXE

chord 1:C#0 G#0 D1
chord 2:E0 F#0 B0
chord 3:A2 C3 E3 D4 A4 C5 A#5 D#6 F6
chord 4:A#2 C3 F3 A#3
chord 5:B1 D2 F#2 G2 B2 D3 E3 F3 A3
chord 6:G#1 C#2 D#2 G2 B2 D3
chord 7:F0 A#0 D#1 F1
chord 8:C3 F3 A#3 C4 F#4 B4 C5 F#5
chord 9:E1 A1 C2
chord 10:B0 D1 A1
chord 11:A0 B0 C#1 G1 A#1 F2 C3 F3
chord 12:C2 D2 F2 C3 D#3 F3 G#3 D#4
chord 13:F#2 C#3 F#3 B3 D#4 B4 D5
chord 14:D#1 F1 B1 C2 F#2 A2 A#2
chord 15:F0 G#0 D#1 F1 B1

```

## 函数: Octscale() MIC7.12

### 功能

根据一系列输入的音程值生成一系列音程值重复、标准的音阶值。

## 注释

函数 Octscale 类似于 MIC7.11 的 Rdintchd 函数，因为它的返回值是长度不同的、从低音到高音排列的音符列。只不过此函数中生成音程相同、音阶不同的音的目的是为了可用本函数之外的作曲过程。

本函数先选一起始音以及起始音阶，然后在计算本音阶各音(通过运用音程数组 gamints[])之前先选定音域。重复上述过程直至选出的元素数目达到变量 segtotal 所要求的值。因此需要对音域中所有的音程序列都重复以上过程。

本函数组可以很容易地改编为交互式程序，但必须加入附加语句防止用户输入的音域值超出音符表的范围。

## 编程提示

1. 修改主程序以提示用户输入数组 gamuts 各元素值，以及变量 segtotal 和 inttotal 的值。
2. 修改函数 MIC 7.12 Octscl() 提示用户输入变量 tonic, startoctave 和 scalemens 的值。
3. 编写一交互式程序应用 MIC7.11 中的 Rdintchd 函数以及 MIC7.12 中的 Octscale 函数，以为输出文件提供辅编码。

## 程序清单

```
/* OCTSCALE.C (generates octave-repeating scales/gamuts) */
#include <stdio.h>
main()
{
    int gamints[11]; /* array of interval sizes in 1/2 steps */
    int j, j1, k;    /* loop indices */
    int inttotal = 6; /* current # of intervals in gamints[] */
    int gamuts = 1;  /* number of unique gamuts to generate */
    int segtotal = 3; /* scale segments to return per gamut */
    int highoctave;  /* upper octave-register limit */
    int exceed;      /* flags interval set octave over-run */
    int seed = 9132;
    void Octscale();

    srand(seed);

    for (j = 0; j < gamuts; j++)
    {
        printf("scale gamut %d interval set:\n", j+1);
        highoctave = 7;
        exceed = 0;
        do {
            exceed = 0;
            for (k = 0; k < inttotal; k++)
            {
                gamints[k] = rand() % 3 + 1;
                exceed += gamints[k];
            }
        } while (exceed > 11);
        for (j1 = 0; j1 < inttotal; j1++)
```

```

        printf("%d ", gamints[j]);
        Octscale(gamints, highoctave, inttotal, segtotal);
    }
} /* end of main */
/*===== MIC 7.12 =====*/
/* Octscale() function (generates octave-repeating scales
   from a set of intervals) */
void Octscale(gamints, highoctave, inttotal, segtotal)
int gamints[], highoctave, inttotal, segtotal;
{
    int count; /* tabulates chordtone generation */
    int tonic; /* lowest scale/gamut tone */
    int scaletone; /* next scale/gamut member */
    int startoctave; /* scale segment tonic register */
    int scalemens; /* number of tones in current segment */
    int k,l,m; /* loop indices */
    static char *pitch[] = {
        "C0 ", "C#0 ", "D0 ", "D#0 ", "E0 ", "F0 ", "F#0 ", "G0 ", "G#0 ",
        "A0 ", "A#0 ", "B0 ", "C1 ", "C#1 ", "D1 ", "D#1 ", "E1 ", "F1 ",
        "F#1 ", "G1 ", "G#1 ", "A1 ", "A#1 ", "B1 ", "C2 ", "C#2 ", "D2 ",
        "D#2 ", "E2 ", "F2 ", "F#2 ", "G2 ", "G#2 ", "A2 ", "A#2 ", "B2 ",
        "C3 ", "C#3 ", "D3 ", "D#3 ", "E3 ", "F3 ", "F#3 ", "G3 ", "G#3 ",
        "A3 ", "A#3 ", "B3 ", "C4 ", "C#4 ", "D4 ", "D#4 ", "E4 ", "F4 ",
        "F#4 ", "G4 ", "G#4 ", "A4 ", "A#4 ", "B4 ", "C5 ", "C#5 ", "D5 ",
        "D#5 ", "E5 ", "F5 ", "F#5 ", "G5 ", "G#5 ", "A5 ", "A#5 ", "B5 ",
        "C6 ", "C#6 ", "D6 ", "D#6 ", "E6 ", "F6 ", "F#6 ", "G6 ", "G#6 ",
        "A6 ", "A#6 ", "B6 ", "C7 " };

    for (k = 0; k < segtotal; k++)
    {
        /* select random values for the control parameters */
        tonic = rand() % 12 + 1;
        startoctave = rand() % 3 + 1;
        scalemens = rand() % 12 + 4;
        printf("\nscale segment %d = %d notes\n", k+1, scalemens);
        count = 0;
        for (l = startoctave - 1; l < highoctave; l++)
        {
            scaletone = tonic;
            for (m = 0; m <= inttotal; m++)
            {
                printf("%s ", pitch[scaletone + (l * 12)]);
                count++;
                if (count == scalemens)
                    break;
                if (m >= inttotal)
                    break;
                scaletone = scaletone + gamints[m];
            }
            if (count == scalemens)
                break;
        }
        printf("\n");
    }
} /* end of Octscale() function */
/*=====*/
/* END OF OCTSCALE.C */

```

## 程序运行实例

```
scale gamut 1 interval set:
3 1 3 2 1 1
scale segment 1 = 11 notes
B1 D2 D#2 F#2 G#2 A2 A#2 B2 D3 D#3 F#3
scale segment 2 = 5 notes
D2 F2 F#2 A2 B2
scale segment 3 = 6 notes
G0 A#0 B0 D1 E1 F1
scale gamut 2 interval set:
2 3 1 1 1 3
scale segment 1 = 13 notes
C#0 D#0 F#0 G0 G#0 A0 C1 C#1 D#1 F#1 G1 G#1 A1
scale segment 2 = 7 notes
D2 E2 G2 G#2 A2 A#2 C#3
scale segment 3 = 6 notes
D1 E1 G1 G#1 A1 A#1
scale gamut 3 interval set:
1 1 2 3 1 3
scale segment 1 = 13 notes
G1 G#1 A1 B1 D2 D#2 F#2 G2 G#2 A2 B2 D3 D#3
scale segment 2 = 5 notes
D1 D#1 E1 F#1 A1
scale segment 3 = 13 notes
B2 C3 C#3 D#3 F#3 G3 A#3 B3 C4 C#4 D#4 F#4 G4
```

## 函数: **Intgam()** MIC 7.13

### 功能

根据输出数组生成一组音符密度不同的音阶以及和弦。

### 注释

本函数集 MIC 7.11 的 **Rdintchd** 函数与 MIC7.12 的 **Octscale** 函数一体, 它根据一组固定的无序音程返回一组音阶, 但该组音程不一定要指定是在哪一音阶级, (该组音程只用于计算当前音符与前一输出音符之间的间隔。)

虽然, **gamints[]** 各数组元素值, 以及变量 **root**, **octave** 和 **chordments** 的值可在主程序中任意给定, 但是我们可以将此函数改写为交互式由用户自定义上述各值。

### 编程提示

#### 1. 编写一交互式程序实现下列功能:

- 应用函数 MIC7.11 中的 **Rdintchd()**、MIC7.12 中的 **Octscale()** 以及 MIC7.13 中的 **Intgam**。

- b. 为用户提供三种方式来生成音阶/和弦以及可使用户在各音列中加入单个音符。
  - c. 在输出数组中设置字段结束标志符。
  - d. 将程序运行后生成的音阶/和弦序列以文件形式存盘。
2. 编写一程序, 以实现可从一文件中单独读出一列记录, 并将其转入乐曲程序中。

## 程序清单

```

/* INTGAM.C (generates non-octave-repeating gamuts/chords
   derived from an interval set) */
#include <stdio.h>
main()
{
    int gamints[12]; /* array of interval-sizes in 1/2-steps */
    int j;           /* loop index */
    int inttotal = 6; /* number of intervals in gamints[] */
    int chdtotal = 5; /* number of chord/gamuts to return */
    int root;        /* lowest chord/gamut member */
    int octave;      /* start octave register */
    int chordmems;    /* number of pitches in each chord/gamut */
    int seed = 21;
    void Intgam();

    srand(seed);
    for (j = 0; j < inttotal; j++)
        gamints[j] = rand() % 3 + 1; /* get intervals */
    for (j = 0; j < chdtotal; j++)
    {
        printf("chord %d\n", j+1);
        root = rand() % 12 - 1; /* get random start pitch */
        octave = rand() % 3 + 1; /* get random start octave */
        chordmems = rand() % 10 + 6; /* get gamut-length */
        Intgam(gamints, root, octave, chordmems, inttotal);
        printf("\n");
    }
} /* end of main */
/*===== MID 7.13 =====*/
/* Intgam() function */
void Intgam(gamints, root, octave, chordmems, inttotal)
int gamints[], root, octave, chordmems, inttotal;
{
    int count, chordtone, k, l;

    static char *pitch[] = {
        "C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
        "A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
        "F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
        "D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
        "C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
        "A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
        "F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
        "D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
        "C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
        "A6 ", "A#6", "B6 ", "C7 " };

    count = 0;
    chordtone = root + ((octave-1) * 12);
    do
    {
        for (l = 0; l < inttotal; l++)

```

```

        {
            printf("%s ",pitch[chordtone]);
            count++;
            if (count == chordmens)
                break;
            chordtone += gamints[1];
        }
        while (count != chordmens);
    }/* end of Intgam() function */
    /*=====*/
    /* END OF INTGAM.C */

```

## 程序运行实例

INTGAM.EXE

```

chord 1
G0 G#0 A#0 C1 C#1 D1 D#1 E1 F#1 G#1 A1 A#1 B1
chord 2
A#0 B0 C#1 D#1 E1 F1
chord 3
B0 C1 D1 E1 F1 F#1 G1 G#1 A#1 C2 C#2 D2 D#2 E2
chord 4
D1 D#1 F1 G1 G#1 A1 A#1 B1 C#2 D#2 E2 F2 F#2 G2
chord 5
A#2 B2 C#3 D#3 E3 F3 F#3 G3 A3 B3 C4 C#4 D4 D#4 F4

```

## 函数组: Rhyprops.c

1. Parstore() MIC2.4
2. Parxtret() MIC2.5
3. Rhyprops() MIC2.6

## 功能

将节拍的分数值转化为各音的音长值。

## 注释

我们可以通过将一系列节拍分数值转化为一段旋律中各音的音长值来处理作曲中有关发音方式方面的问题。虽然音乐中连续的打点计时系统是用一系列的整数值来计时，但分数值都应用得更广泛，它通过分析节拍值之间的关系来计算出各音的实际音长。因为节拍变化并不是一个连续、有序过程，因为结果也不应按打点计时系统方法输出(可参见程序清单后的运行结果)。

节拍的分数值信息由三部分组成，例如，1:4:8的意思是某一音的音长相当于4个

1/8 音的时间。相反, 4:1:8 意思是 4 个音符, 每个音符的音长都是一个 1/8 的四分之一。输出会都表示成一个全音的几分之几形式。例如, 4/1 表示相当于 4 个全音的音长, 而 1/4 表示相当于一个全音四分之一的音长。

为了进一步说明这一点, 可用 Parstore() 函数选取 3 个节拍切数值以便转换为输出节拍数据。你可以通过设置一专门数组存放输入数据来修改程序使其可接受一系列分数值作为输入数据。

为了防止返回值为太复杂的分数, 可遵循下列方法:

若分子值大于分母值(例如, 9:8), 则须将分母值转换为下列各数之一:

1,2,4,8,16

当分母值大于分子值时(例如, 4:9), 须将分子值转换为上述各值。

## 编程提示

1. 修改主程序以便可从键盘直接输入各控制变量的值。
2. 增加文件存储编码以存储输出数据。

## 程序清单

```
/* RHYPROPS.C (rhythmic proportion conversion program) */
#include <stdio.h>
main()
{
    double c[20]; /* list of rhythm proportions */
    int total = 3; /* number of items in each proportion */
    int e = 0; /***** loop *****/
    int f = total-1; /***** control *****/
    int i = 0; /***** parameters *****/
    int g = 1; /***** *****/
    void Parstore(), Parxtret();

    Parstore(c, total);
    printf("\noriginal form:\n");
    Parxtret(c, i, f, e, g);
    printf("\nretrograde form:\n");
    i = total-1; /***** adjust *****/
    f = 0; /***** loop control *****/
    g = -1; /***** parameters *****/
    Parxtret(c, i, f, e, g);
    printf("\ninverted form:\n");
    e = 1; /***** adjust *****/
    i = 0; /***** loop control *****/
    f = total-1; /***** parameters *****/
    g = 1; /***** *****/
    Parxtret(c, i, f, e, g);
    printf("\nretrograde inverted form:\n");
    i = total-1; /***** adjust *****/
    f = 0; /***** loop control *****/
    g = -1; /***** parameters *****/
    Parxtret(c, i, f, e, g);
} /* end of main */
/*===== MIC 2.4 =====*/
/* Parstore() function (adapted to program specs) */
void Parstore(array, total)
double array[20];
```

```

int total;
{
    int k,base;
    double r1 = 0,r2 = 0; /* part1 & part 2 of proportion */
    for (k = 0;k < total;k++)
    {
        r1 = (k+1) * (k+1); /* load arbitrary proportion*/
        r2 = (k+1) + 3.0; /* load arbitrary proportion*/
        base = 8; /*load metrical base*/
        printf("selected proportion = %.0f:",r1);
        printf("%.0f\n",r2);
        printf("metrical base = %d\n",base);
        r1 = r1 * 1000000.;
        r2 = r2 * 1000.;
        array[k] = r1 + r2 + base;
    }
} /* end of Parstore() function */
/*===== MIC 2.5 =====*/
/* Parxtrect() function (adapted to program specs) */
void Parxtrect(c,i,f,e,g)
double c[];
int i,f,e,g;
{
    double d;
    int r1,r2,base,k,Rhyprops();

    for (k = i;k +=g)
    {
        d = c[k];
        d = d / 1000000.;
        r1 = d;
        r2 = (d - r1) * 1000;
        d = d * 1000.;
        base = (d - (int)d) * 1000. + .5;
        Rhyprops(r1,r2,base,e);
        if (k == f)
            break;
    }
} /* end of Parxtrect() function */
/*===== MIC 7.14 =====*/
/* Rhyprops() function (convert rhythm proportions to
note duration values expressed
as fractions of a whole note) */
int Rhyprops(r1,r2,base,e)
int r1,r2,base,e;
{
    int num; /* converted duration fraction numerator */
    int temp; /* temporary computation storage */
    int l; /* loop index */
    float den; /* converted duration fraction denominator */

    if (e == 1)
    {
        temp = r1;
        r1 = r2;
        r2 = temp;
    }
    if (r1 < r2)
    {
        den = (float)r2 / r1;
        while ((int)den != den;
            {

```



```

        den = den * 2.;
        base = base * 2;
    }
    for (l = 1; l <= r1; l++)
        printf("%.0f/%d ", den, base);
    printf("\n");
    return;
}
num = 1;
den = r1 * base / r2;
while ((int)den != den)
{
    num = num * 2;
    den = den * 2.;
}
for (l = 1; l <= r1; l++)
    printf("%d/%.0f ", num, den);
printf("\n");
return;
} /* end of Rhyprops() function */
/*-----*/
/* END OF RHYPROS.C */

```

### 程序运行实例

RHYPROPS.EXE

```

selected proportion - 1:4
metrical base = 8
selected proportion - 4:5
metrical base = 8
selected proportion - 9:6
metrical base = 8

```

original form:

```

4/8
5/32 5/32 5/32 5/32
1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12

```

retrograde form:

```

1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12
5/32 5/32 5/32 5/32
4/8

```

inverted form:

```

1/32 1/32 1/32 1/32
1/10 1/10 1/10 1/10 1/10
3/16 3/16 3/16 3/16 3/16 3/16

```

retrograde inverted form:

```

3/16 3/16 3/16 3/16 3/16 3/16
1/10 1/10 1/10 1/10 1/10
1/32 1/32 1/32 1/32

```

## 函数组: Polyrhy.c

1. Polyrhy() MIC7.15
2. Durred() MIC2.10

### 功能

在四种可供选择的复杂程度上生成多音的、多节拍的 / 标准节拍的、无序的、同步修正的音符音长序列。

### 注释

Polyrhy()函数相当于音乐家计算某一节拍形式的逆过程。它根据变量 timescale 的值,通过切合、重组全音符所要求的标准脉冲数来计算音长值。变量 level 根据要求的组合度细分脉冲(将脉冲细分为越小的等份以及此后的微脉重组过程将会使生成的节拍形式具有越强的非周期性),变量 ml 为中度——即音长序列的节拍基准值。例如,4 表示全音符的  $1/4$ ; 5 表示全音符的  $1/5$ ; 7 表示全音符的  $1/7$ , 等等。最后,用 Durred()函数将计算出的音长换算为最简分数对应的值。

主程序输出 4 级音长序列,每一组中的字长的长度、所含的脉冲数,以及复合度不同,变量 total、median 以及 level 在每次循环重奏中逐渐变大,而变量 timescale 的值保持不变以保证不同长度音长序列能同步起止。

### 编程提示

1. 修改主程序,使其成为交互式,并增加一函数将输出数列存盘。
2. 编写一程序适用 MIC7.15 中的 Polyrhy 函数以及 MIC7.2 中的 Loopgen2 函数来处理音长序列。然后将该音长序列与其他函数(例如, MIC7.5 中的 Trantabl()函数)谱出的音符序列相组合以生成一段完整的曲子。

### 程序清单

```
/* POLYRHY.C (Polyrhythmic/metric duration
sequence generator)*/
#include <stdio.h>
main()
{
    int j;                /* loop index */
    int total = 10;        /* number of durations to return */
    int ml = 3;            /* metrical base value */
    int level = 1;         /* rhythm complexity level (1-2-4-8) */
    int timescale = 10;    /* arbitrary time frame */
    int seed = 4544;
    void Polyrhy();
}
```

```

srand(seed);
for (j = 0; j < 4; j++)
{
    printf("\npass %d\n",j+1);
    printf("number of durations in rhythm sequence =%d\n",
        total);
    printf("median = %d\n",m1);
    printf("timescale = %d (scale = 1 - 10)\n",timescale);
    printf("level = %d (from a choice of 1-2-4-8)\n",level);
    Polyrhy(total,timescale,m1,level);
    level += level;
    total += 10;
    m1++;
}
} /* end of main */
/***** MIC 7.15 *****/
/* Polyrhy() function */
void Polyrhy(total,timescale,m1,level)
int total,timescale,m1,level;
{
    int dur[200]; /* stores duration numerators */
    int newmed; /* reduced duration fraction denominator */
    int k,l; /* loop indices */
    int odd; /* flag for odd number of values in sequ.*/
    int sum; /* total micropulses in raw sequence */
    int base; /* number of micropulses to distribute */
    int median; /* metrical base value (3,4,5,7,9,etc.)*/
    int factor; /* determinant of random integer range */
    int range; /* span of random integers */
    int wedge; /* low limit of random integers */
    int rcomp; /* stores computation over- or under-run*/
    int u1, u2, u3; /* random integers */
    int Durred();

    odd = 0;
    sum = 0;
    median = m1;
    base = median * timescale;
    if (total == 1)
    {
        printf("%d\n",timescale);
        return;
    }
    if (timescale - total > 0)
    {
        if (base / 2.0 != base / 2)
        {
            base = base - 1;
            odd = m1;
        }
        for (k = 0; k < 100; k++)
        {
            if (base/2.0 != base/2 || median/2.0 != median/2)
                break;
            if (total <= base / 2)
            {
                median /= 2;
                base /= 2;
            }
        }
    }
    else

```

```

        break;
    }
else
{
    for (k = 1; k <= 30; k++)
    {
        if (total > base)
        {
            median *= 2;
            base *= 2;
        }
        else
            break;
    }
}
base *= level;
median *= level;
if (level == 4) factor = 3;
else if (level == 8) factor = 4;
else factor = level;
wedge = factor;
range = (factor + level) * base / total;
for (k = 1; k <= total; k++)
{
    u1 = rand() % range + wedge;
    dur[k] = u1;
    sum += u1;
}
rcomp = base - sum;
while (rcomp != 0)
{
    for (l = 1; l <= total; l++)
    {
        if (rcomp < 0)
        {
            if (dur[l] > 1)
            {
                dur[l] = dur[l] - 1;
                rcomp = rcomp + 1;
            }
            continue;
        }
        if (rcomp == 0)
            break;
        else
        {
            dur[l] = dur[l] + 1;
            rcomp = rcomp - 1;
            break;
        }
    }
}
if (odd == 1)
{
    u2 = rand() % total + 1;
    do
    {
        u3 = rand() % total + 1;
    }
}

```

```

        while (total > 1 && u3 == u2);
        dur[u2] = dur[u2] + dur[u3];
        dur[u3] = -1;
    }
    for (k = 1; k <= total; k++)
    {
        if (k % 10 == 0)
            printf("\n");
        newmed = median;
        if (dur[k] / 2.0 == dur[k]/2 && newmed/2.0 == newmed/2)
            newmed = Durred(dur,newmed,k);
        if (dur[k] < 0)
            printf("1/%d ",odd);
        else
            printf("%d/%d ",dur[k],newmed);
    }
    printf("\n");
} /* end of Polyrhy() function */
/*===== MIC 2.10 =====*/
/* Durred() function (adapted to program specs) */
int Durred(dur,median,k)
int dur[],median,k;
{
    do
    {
        dur[k] = dur[k]/2;
        median = median / 2;
    }
    while (dur[k]/2.0 == dur[k]/2 && median/2.0 == median/2);
    return(median);
} /* end of Durred() function */
/*=====*/
/* END OF POLYRHY.C */

```

### 程序运行实例

```

pass 1
number of durations in rhythm sequence =10
median = 3
timescale = 10 (scale = 1 - 10)
level = 1 (from a choice of 1-2-4-8)
3/3 1/3 1/3 4/3 5/3 2/3 2/3 5/3 2/3
5/3

pass 2
number of durations in rhythm sequence =20
median = 4
timescale = 10 (scale = 1 - 10)
level = 2 (from a choice of 1-2-4-8)
3/8 1/1 1/2 1/2 3/4 3/8 1/8 1/2 5/8
1/2 1/4 3/8 5/8 1/4 1/8 1/1 1/2 1/2 7/8
1/4

pass 3
number of durations in rhythm sequence =30
median = 5
timescale = 10 (scale = 1 - 10)

```

```

level = 4 {from a choice of 1-2-4-8}
11/20 1/20 11/20 1/20 5/10 11/20 3/5 3/10 1/20
3/5 7/20 3/10 1/20 9/20 3/5 5/20 5/10 5/10 1/20
3/10 1/10 7/20 1/20 11/20 3/20 1/20 5/20 11/20 3/5
1/5

pass 4
number of durations in rhythm sequence = 40
median = 6
timescale = 10 {scale = 1 - 10}
level = 8 {from a choice of 1-2-4-8}
1/3 3/24 13/48 17/48 9/48 1/24 19/48 1/48 9/48
3/48 3/24 3/48 17/48 21/48 15/48 19/48 17/48 21/48 11/48
17/48 9/24 11/48 17/48 9/48 11/48 7/48 1/6 3/24 1/48
3/48 11/48 1/48 9/48 15/48 7/24 9/24 11/24 9/24 9/24
19/48

```

### 函数组: Meline.c

1. Rdmelint()     MIC7.16
2. Rest()        MIC 7.17
3. Printpitch()   MIC3.1

### 功能

生成一系列无序、音程范围给定包含一部分休止音的音符。

### 注释

本函数组的主要函数是 Rdmelint()。大致与函数 Rdintchd()相同，但用法不一样。首先，它的返回值是一双向乐句而不是一列无序无向的和弦音，其次，它还有一个变量 up 控制乐句的演奏方向是正向还是逆向。

函数 Rest()和允许用户输入一休止音出现频率(以百分比形式)来控制乐句发音方式。只要输出音乐数据文件中的某数小于某一指定值，就在此处插入休止符。

### 编程提示

编写一程序应用函数组 MELINE.C 实现以下功能：

1. 引入一控制变量以便在程序运行过程中增大或减小音程值的范围。
2. 允许用户在每一次运行后生成一些音程各异的乐音序列。
3. 将生成的乐段命名并存盘，以便将来从中单独选取某一乐段。
4. 为实现不同过程而单独检索各乐段。
5. 交替打印出乐段的原始形式以及其 ABACADAE 形式(回旋曲形式)。

### 程序清单

```

/* MELINE.C (generates a random-order, interval-constrained,
melodic line containing a percentage of
rest values) */
#include <stdio.h>
main()
{
    int j,k;          /* loop indices */
    int total = 20;   /* length of sequence */
    int small = 1;    /* smallest allowable interval-size */
    int large = 3;    /* largest allowable interval-size */

    int up = 80;      /* prob. (in %) of ascending intervals */
    int rest = 10;    /* prob. (in %) of rest interpolation */
    int start = 48;   /* sequence start value */
    int seed = 3923;
    void Rdmelint();

    srand(seed);
    for (j = 0; j < 4; j++)
    {
        printf("pass %d\n", j+1);
        printf("total=%d small=%d large=%d\n", total, small, large);
        printf("up=%d rest=%d start=%d\n", up, rest, start);
        Rdmelint(start, small, large, up, rest, total);
        total += 10;
        small += 1;
        large += 2;
        up -= 20;
        rest += 20;
    }
} /* end of main */
/*===== MIC 7.16 =====*/
/* Rdmelint() function (computes an interval-constrained
pitch sequence) */
void Rdmelint(note, small, large, up, restprob, total)
int note, small, large, up, restprob, total;
{
    int j;          /* loop index */
    int flag;       /* rest note indicator */
    int intvalrange; /* span of allowable interval-sizes */
    int u;          /* random integers */
    int Restgen();
    static char *pitch[] = {
        "C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
        "A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
        "F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
        "D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
        "C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
        "A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
        "F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
        "D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
        "C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
        "A6 ", "A#6", "B6 ", "C7 ", "R " };
    intvalrange = large - small + 1;
    printf("%s ", pitch[note]);
    for (j = 0; j < total-1; j++)
    {
        if (j % 12 == 0)
            printf("\n");
        flag = Restgen(restprob);
        if (flag)
        {

```

```

        printf("%s ",pitch[85]);
        continue;
    }
    u = rand() % intvalrange + small;
    if (rand() % 100 < up)
        note = note + u;
    else
        note = note - u;
    if (note > 85)
        note -= 12;

        else if (note < 1)
            note += 12;
        printf("%s ",pitch[note]);
    }
    printf("\n\n");
} /* end of Admelint function */
/***** MIC 7.17 *****/
/* Restgen() function */
int Restgen(restprob)
int restprob;
{
    int flag;

    flag = 0;
    if (rand() % 100 < restprob)
        flag = 1;
    return(flag);
} /* end of Restgen() function */
/***** */
/* END OF MELINE.C */

```

## 程序运行实例

MELINE.EXE

```

pass 1
total=20 small=1 large=3
up=80 rest=10 start=48
C4 A#3 B3 D4 F4 G#4 G4 A4 C5 D5 E5 D#5
F#5 R A5 B5 C5 D6 D#6 F6

```

```

pass 2
total=30 small=2 large=5
up=60 rest=10 start=48
C4 R D4 E4 A4 F4 C#4 R G#3 A#3 R C4
F4 A#4 R C5 R D#5 F#5 E5 F#5 G#5 C#6 G#5
E5 A5 R R C6 R

```

```

pass 3
total=40 small=3 large=7
up=40 rest=50 start=48
C4 F3 A#2 F#2 B1 F1 R R A1 R R R
F1 A#0 E1 B0 R R R G0 R R R C#0
F#0 R A#0 R R R R R F1 R R
A#0 R F#0 B0

```

```

pass 4
total=50 small=4 large=9

```



```

up=20 rest=70 start=48
C4 R R R R R R R R R R
D#3 R F#2 R R R C2 E1 R R R G#0
R R C#1 R R G#0 R R R R R
B0 R R R R F0 R R R R R
R R

```

## 函数: Partspan() MIC7.18

### 功能

以三种方式计算一组乐音的音程范围。这三种方式包括: 分层方式、连锁方式、以及普通方式。

### 注释

在有些情况下, 需要将一源表中选音诸曲的过程跟每音的音程 / 标定控制过程协调统一起来, 用 Partspan() 函数可实现这一功能。它可将总音域分成几个子音程范围, 或者将互相重叠的音程分配到各部分, 还可以为每部分提供全音域范围以供选音。

主程序可在每部分的音程范围内随机选音, 只须稍加改动就可以调程序中的函数改为可调用其他作曲算法的函数。

MIC 7.18 函数的应用还可以扩展到任何一系列数据(升序、降序或无序数据), 并通过将程序输出数据作为一组音乐参数指针的范围以供选音时参考。

### 编程提示

编写一交互式程序应用函数 MIC7.13: Intgam() 生成一个和弦音文件, 定其各乐段音程范围使各段可分别在四种乐器上演奏, 并将选出的各声部的音排序, 然后用函数组 POLYRHY.C 生成每声部的各音音长值序列。

将输出数据文件转化为标准乐谱形式以便在四种乐器上演奏。

### 程序清单

```

/* PARTSPAN.C (Computes instrumental part ranges from a
common pitch aggregate in one of 2 modes:
stratified or interlocking.) */
#include <stdio.h>
main()
{
    int note[40];          /* stores source pitchbank */
    int part[4];           /* list of part ranges */
    int adjust[4];         /* list of part lower limits */
    int u;                 /* random integer */
    int chordlength = 40; /* # of notes in source pitchbank */

```

```

int voices = 4; /* number of score parts to return */
int melength= 80; /* length of melody to generate */
int interlock; /* part range relationship indicator */

int seed = 2113;
int j,k,l; /* loop indices */
void Partspan();
static char *pitch[] = {
"C0 ", "C#0", "D0 ", "D#0", "E0 ", "F0 ", "F#0", "G0 ", "G#0",
"A0 ", "A#0", "B0 ", "C1 ", "C#1", "D1 ", "D#1", "E1 ", "F1 ",
"F#1", "G1 ", "G#1", "A1 ", "A#1", "B1 ", "C2 ", "C#2", "D2 ",
"D#2", "E2 ", "F2 ", "F#2", "G2 ", "G#2", "A2 ", "A#2", "B2 ",
"C3 ", "C#3", "D3 ", "D#3", "E3 ", "F3 ", "F#3", "G3 ", "G#3",
"A3 ", "A#3", "B3 ", "C4 ", "C#4", "D4 ", "D#4", "E4 ", "F4 ",
"F#4", "G4 ", "G#4", "A4 ", "A#4", "B4 ", "C5 ", "C#5", "D5 ",
"D#5", "E5 ", "F5 ", "F#5", "G5 ", "G#5", "A5 ", "A#5", "B5 ",
"C6 ", "C#6", "D6 ", "D#6", "E6 ", "F6 ", "F#6", "G6 ", "G#6",
"A6 ", "A#6", "B6 ", "C7 " };

srand(seed);
printf("%s %s %s", "pitchbank (chord or scale) from which"
"random-order\nmelodies will be derived to test",
"the distribution:\n");
for (j = 0; j < chordlength; j++)
/* load and display example scale/chord structure */
{
if (j % 11 == 0)
printf("\n");
note[j] = (j+1) * 2;
printf("%s ", pitch[note[j]]);
}
for (j = 0; j < 2; j++)
{
if (j > 0)
{
printf("\nPASS 2, INTERLOCKING:\n");
interlock = 1;
}
else
{
printf("\n\nPASS 1, STRATIFIED:\n");
interlock = 0;
}
Partspan(part, adjust, voices, chordlength, interlock);
for (k = 0; k < voices; k++)
{
printf("\nvoice %d melody:\n", k+1);
printf("range = %s to ", pitch[note[adjust[k]]]);
printf("%s \n", pitch[note[adjust[k]+part[k]-1]]);
for (l = 0; l < melength; l++)
{
if (l % 11 == 0)
printf("\n");
u = (rand() % part[k]) + adjust[k];
printf("%s ", pitch[note[u]]);
}
printf("\n");
}
printf("\n");
}
} /* end of main */
/*===== MIC 7.18 =====*/

```

```

/* Partspan() function */
void Partspan(part,adjust,voices,chordlength,interlock)
int part[],adjust[],voices,chordlength,interlock;
{
    int span; /* subdivision of total pitch bank range */
    int extra; /* any remainder from subdivision */
    int comp; /* compensatory value added to parts */
    int j; /* loop index */

    span = chordlength / voices;
    extra = chordlength - voices * span;
    for (j = 0; j < voices; j++)
    {
        if (extra <= 0)
            comp = 0;
        else
            comp = 1;
        part[j] = span + comp;
        extra -= 1;
        adjust[0] = 0;
        if (j < voices-1)
            adjust[j+1] = adjust[j] + part[j];
    }
    if (interlock < 1)
        return;
    for (j = 0; j < voices-1; j++)
        part[j] = part[j] + part[j] * .5;
} /* end of Partspan() function */
/*=====*/
/* END OF PARTSPAN.C */

```

## 程序运行实例

### PARTSPAN.EXE

pitchbank (chord or scale) from which random-order  
melodies will be derived to test the distribution:

```

D0 E0 F#0 G#0 A#0 C1 D1 E1 F#1 G#1 A#1
C2 D2 E2 F#2 G#2 A#2 C3 D3 E3 F#3 G#3
A#3 C4 D4 E4 F#4 G#4 A#4 C5 D5 E5 F#5
G#5 A#5 C6 D6 E6 F#6 G#6

```

PASS 1, STRATIFIED:

voice 1 melody:  
range = D0 to G#1

```

E1 F#1 D1 G#0 G#1 E0 G#1 G#1 D0 C1 G#1
D1 D1 E1 G#1 D0 E1 D0 E1 A#0 D0 E1
E0 C1 G#0 A#0 G#0 G#1 E0 C1 E0 G#0 E1
C1 F#1 G#0 A#0 E1 C1 F#1 G#0 C1 G#1 G#1
E1 C1 E0 C1 G#0 C1 E0 E1 E1 A#0 G#0
E0 C1 D0 D0 E0 F#1 E1 D1 F#1 G#0 D0
E0 C1 F#0 A#0 C1 F#1 A#0 D1 D1 D0 E1
F#1 C1 E1

```

voice 2 melody:  
range = A#1 to E3

D2 A#1 A#1 C3 F#2 G#2 A#2 G#2 G#2 C2 C3  
A#1 E3 A#2 E2 D2 D3 G#2 D2 A#1 D3 F#2  
E2 C3 A#2 C2 A#1 E3 F#2 A#1 E2 D2 G#2  
E3 A#2 D3 C3 D3 D2 D3 D3 C2 D2 D2  
A#1 E2 F#2 C3 C3 G#2 A#1 G#2 E3 D2 D2  
A#1 A#1 G#2 E3 D2 A#1 D2 C2 A#2 E2 E3  
E3 D3 C2 C3 E2 D2 D2 E2 D2 G#2 E3  
G#2 D2 G#2

voice 3 melody:  
range = F#3 to C5

G#3 G#3 C5 A#3 A#4 A#4 E4 G#4 F#3 F#4 A#3  
G#4 C5 F#4 D4 G#3 F#3 F#4 D4 G#4 G#4 D4  
C4 A#3 G#3 C5 C4 G#3 C5 G#3 C5 A#4 F#4  
F#4 C4 E4 G#4 C5 E4 C5 C5 D4 F#3 F#3  
E4 D4 A#4 G#3 E4 C5 D4 G#4 G#4 F#4 A#3  
A#4 A#4 F#4 E4 F#3 C5 D4 C5 D4 A#3 C4  
F#4 A#4 F#4 F#3 C5 A#4 A#3 A#3 G#3 G#3 A#4  
D4 G#4 C4

voice 4 melody:  
range = D5 to G#6

E6 A#5 E5 D5 F#6 A#5 C6 A#5 F#5 C6 E6  
A#5 F#5 E5 C6 D5 A#5 E5 G#6 C6 D5 D6  
D5 E6 G#6 F#6 C6 F#5 F#5 G#5 D5 D6 F#5  
A#5 E5 F#6 E5 C6 E6 D5 E5 C6 G#5 F#6  
D5 C6 E5 D6 E5 G#6 F#6 C6 D6 F#5 E5  
A#5 G#5 E5 A#5 F#6 G#6 C6 F#6 F#6 C6 D6  
F#6 D5 F#6 G#5 D5 D5 F#5 G#5 G#6 D5 G#6  
F#5 E5 E6

PASS 2, INTERLOCKING:

voice 1 melody:  
range = D0 to F#2

D2 D2 D0 C2 E1 F#2 F#0 E0 F#2 E2 G#0  
E1 E0 D2 C1 C1 A#1 D2 C1 D0 G#0 G#0  
G#0 E1 F#1 C1 G#0 F#1 C1 E0 D1 C2 E0  
D0 G#1 A#1 C2 G#0 D2 E1 C1 C2 F#1 D2  
C1 A#1 C2 E0 F#0 E0 E2 C2 F#2 E1 D1  
C2 F#2 F#1 C1 D2 D2 G#1 E2 A#0 D2 F#0  
A#0 E0 E2 A#0 F#0 E1 E0 D1 F#1 C1 F#2  
C1 C1 C1

voice 2 melody:  
range = A#1 to D4

A#1 E2 A#2 E2 C3 D2 F#3 F#3 A#2 D4 D3  
E2 D2 D3 C4 F#3 F#2 C4 C3 A#1 F#2 E2  
A#3 D4 D2 A#3 C3 F#2 C3 C2 D2 C2 D4  
A#2 C2 A#3 E2 D4 F#2 A#1 C2 C4 D2 E3  
C2 A#3 F#2 E2 A#1 E2 C4 E2 C4 C2 E2  
C4 D2 D2 C3 A#2 A#3 C3 A#3 F#2 E3 E3

```
A#3 F#3 D4 C3 A#1 C4 E2 E3 D3 C2 D4
A#1 A#1 E3
```

```
voice 3 melody:
range = F#3 to A#5
```

```
D4 G#5 A#3 E5 D5 F#5 A#3 D5 A#5 D5 F#3
A#3 G#3 D4 G#3 C4 C5 F#3 E5 C5 F#4 A#5
F#3 F#5 A#4 G#5 E5 E5 F#3 C4 C4 A#5 A#3
E4 G#4 E5 C5 F#3 D4 D4 F#4 F#5 F#5 A#4
F#4 G#4 F#5 C5 C5 D4 G#3 E4 F#4 E4 G#3
E5 C5 G#3 G#5 G#4 G#3 D4 E4 F#4 D5 F#4
F#5 D5 D4 C4 G#4 G#5 G#5 E4 A#4 F#3 E4
G#4 G#5 G#4
```

```
voice 4 melody:
range = D5 to G#6
```

```
G#5 A#5 E5 A#5 E5 G#5 E6 E6 G#6 G#5 E6
D5 E6 F#6 D6 G#6 D6 G#5 E6 D6 E5 C6
F#5 G#6 A#5 G#5 C6 G#6 E5 E5 G#5 A#5 D6
A#5 G#5 G#6 E6 G#5 F#5 D6 F#6 E5 C6 F#6
C6 D6 F#6 E5 F#5 E6 G#6 E5 F#6 C6 D6
G#6 A#5 F#6 F#5 D6 E5 E5 A#5 F#6 F#5 G#5
F#5 G#6 E5 G#5 F#5 E6 C6 F#5 C6 G#6 E6
A#5 E5 F#5
```

## 函数组: Ornament.c

1. Ornselec() MIC 7.19
2. Addorn() MIC 7.20

## 功能

生成一列无序音符，并加入装饰音，以便根据音的出现概率加权值，以及四种配装饰音模式来选择音符。

## 注释

主程序旋转一存放有无序的暂态数组，然后将其转入函数 Ornselec() 中，函数读入一概率权值表，表中包含四种配装饰音模式的各加权值，然后调用函数 Addorn() 以便在音符数组中加入选定的装饰音。

四种配装饰音模式有可能被扩展到包含用户要求的任何模式。只需将数组 cmbtable[], dec[] 以及 orn[] 的维数加大装入输入各种模式即可。

## 编程提示

1. 修改主程序段以便允许用户输入乐音序列作为装饰音。
2. 修改主程序段以便从一乐音数据文件中读取音符作为装饰音。

3. 在程序中增加一函数, 以便在乐段中加入休止音。

4. 编写一交互式程序实现下列功能:

a. 用 MIC7.5 中的 Trantabl() 函数生成一旋律。

b. 应用函数组 ORNAMENT.C。

c. 应用 MIC7.2 中的 Loopgnc2() 函数。

d. 并将多音、循环重奏乐段写入磁盘。

### 程序清单

```
/* ORNAMENT.C (Generates a random-order pitch sequence,
then adds embellishments to a percentage
of notes in accordance with probability
weights assigned to each of four ornament
types.)*/
#include <stdio.h>
main()
{
    int temp[200]; /* sequence prior to decoration */
    int final[500]; /* final embellished pitch seq. */
    int embseq; /* number of pitches in final sequence */
    int deco = 70; /* prob. (in %) of note decoration */
    int melength * 30; /* length of original pitch sequence */
    int j; /* loop index */
    int seed = -18763;
    int Ornselec();

    static char *pitch[] = {
        "C0 ", "C#0 ", "D0 ", "D#0 ", "E0 ", "F0 ", "F#0 ", "G0 ", "G#0 ",
        "A0 ", "A#0 ", "B0 ", "C1 ", "C#1 ", "D1 ", "D#1 ", "E1 ", "F1 ",
        "F#1 ", "G1 ", "G#1 ", "A1 ", "A#1 ", "B1 ", "C2 ", "C#2 ", "D2 ",
        "D#2 ", "E2 ", "F2 ", "F#2 ", "G2 ", "G#2 ", "A2 ", "A#2 ", "B2 ",
        "C3 ", "C#3 ", "D3 ", "D#3 ", "E3 ", "F3 ", "F#3 ", "G3 ", "G#3 ",
        "A3 ", "A#3 ", "B3 ", "C4 ", "C#4 ", "D4 ", "D#4 ", "E4 ", "F4 ",
        "F#4 ", "G4 ", "G#4 ", "A4 ", "A#4 ", "B4 ", "C5 ", "C#5 ", "D5 ",
        "D#5 ", "E5 ", "F5 ", "F#5 ", "G5 ", "G#5 ", "A5 ", "A#5 ", "B5 ",
        "C6 ", "C#6 ", "D6 ", "D#6 ", "E6 ", "F6 ", "F#6 ", "G6 ", "G#6 ",
        "A6 ", "A#6 ", "B6 ", "C7 " };

    srand(seed);
    printf("%s%s", "unembellished pitch sequence = ",
        melength, " notes\n");
    for (j = 0; j < melength; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        temp[j] = rand() % 80 + 2;
        printf("%s ", pitch[temp[j]]);
    }
    embseq = Ornselec(temp, final, melength, deco);
    printf("%s%s", "\n\nembellished pitch sequence: \n(",
        deco, "% of original melody notes",
        " have been ornamented)\n");
    for (j = 0; j < embseq; j++)
    {
        if (j % 10 == 0)
            printf("\n");
        printf("%s ", pitch[final[j]]);
    }
}
```

```

    } // end of main */
    /*===== MIC 7.19 =====*/
    /* Ornselec() function (computes ornament-type probabilities
        and invokes Addorn() function to add
        selected ornaments */

    int Ornselec(temp,final,melength,deco)
    int temp[],final[],melength,deco;
    {
        int j;          /* loop index */
        int wsum = 0;    /* sum of ornament prob. weights */
        int embseq = 0; /* pointer to array final[] */

        int orn[] =      /* ornament patterns */
            {1002,2003,2004,4006};
        int dec[] =      /* values for pattern reference */
            {-1,0,-1,0,1,0};
        int embtable[] = /* ornament pattern probabilities */
            {4,3,2,1};
        int rand(),Addorn();

        if (deco == 0)
        {
            printf("\nfinished\n");
            return(0);
        }
        printf("%s%s","\\n\\nornaments being added",
            " (prob. weights 4,3,2,1):\\n");
        printf("1) - 2) - 3) - 4) -");
        for (j = 0; j < 4; j++)
        {
            embtable[j] += wsum;
            wsum = embtable[j];
        }
        for (j = 0; j < melength; j++)
        {
            if (rand() % 100 < deco)
                embseq = Addorn(temp,final,embtable,orn,
                    dec,wsum,j,embseq);
            else
            {
                final[embseq] = temp[j];
                embseq += 1;
            }
        }
        return(embseq);
    } /* end of Ornselec() function */

    /*===== MIC 7.20 =====*/
    /* Addorn() function (interpolates selected ornament
        patterns into original pitch
        sequence) */

    int Addorn(temp,final,embtable,orn,dec,wsum,j,embseq)
    int temp[],final[],embtable[],orn[],dec[],wsum,j,embseq;
    {
        int k,l,m,n; /* loop indices */
        float b;      /* loop index computation */

        for (m = 0; m < 4; m++)
        {
            if (rand() % wsum >= embtable[m])

```

```

        continue;
    else
    {
        k = orn[m] / 1000.;
        b = orn[m] / 1000.;
        l = (b - k) * 1000. + .5;
    }
    for (n = k-1; n <= l-1; n++)
    {
        final[embseq] = temp[j] + dec[n];
        embseq += 1;
    }
    return(embseq);
}
return(embseq);
} /* end of Addorn() function */
/*=====*/
/* END OF ORNAMENT.C */

```

### 程序运行实例

```

ORNAMENT.EXE

unembellished pitch sequence = 30 notes

C3  F#4  B3  B5  C#1  G#5  E3  A#4  D3  B1
B5.  D1  B3  G3  A1  A3  A#1  C#3  D#5  F3
D3  F#6  D#4  F#0  B3  C4  C#1  D3  F#0  F5

ornaments being added (prob. weights 4,3,2,1):
1) _ - 2) - _ 3) - _ 4) _ -

embellished pitch sequence:
(70% of original melody notes have been ornamented)

C3  B2  F4  F#4  B3  A#3  A#5  B5  C#1  D1
C#1  G#5  G5  G#5  E3  D#3  A#4  D3  C#3  B1
A#1  B5  A#5  D1  C#1  B3  F#3  G3  G#1  A1
A3  G#3  A#1  C#3  C3  C#3  D#5  E3  F3  C#3
D3  F#6  D#4  F0  F#0  B3  B3  C4  C#1  C1
D3  C#1  F#0  F5  E5

```

### 函数组: Seqstore.c

1. Setstore() MIC7.21
2. Seqxtret() MIC7.22

### 功能

提供存储、检索一链路结构的整数序列组。

### 注释



函数 Seqstore()生成并存储一组整数序列。并根据序列长度设关键代码(本段中的函数可被替换为可调用其他数据生成算法的函数)。而应用数组 x[]可以调用进行下一步操作的子程序。

函数 Seqxtrct()检索没有序列长度关键代码的多序列数组,并显示数组中各元素。

### 编程提示

编写一交互式程序应用函数组 SEQSTORE.C 实现下列功能:

1. 调用 MIC7.11 中的 Rdintchd()函数来生成一扩展型和弦音数组。
2. 检索数组中存放的和弦音使各和弦可应用于作曲函数中。
3. 然后根据上述和弦音生成一组音程范围给定,无序的乐段。

### 程序清单

```
/* SEQSTORE.C (store and retrieve an articulated group
   of integer sequences.)*/
#include <stdio.h>
main()
{
    int x[200]; /* stores sequence-group with keys */
    int seqtotal=5; /* number of sequences in group to store */
    int notetotal; /* number of values in sequence-group */
    int seed = -912;
    int Seqstore();
    void Seqxtrct();

    srand(seed);
    printf("a group of random-integer sequences are now being stored.\n");
    printf("here is how they are represented in the array:\n");
    notetotal = Seqstore(x,seqtotal);
    printf("\n\nextracted sequence-group looks like this:");
    Seqxtrct(x,notetotal);
} /* end of main */

/*===== MIC 7.21 =====*/
/* Seqstore() function */
int Seqstore(x,seqtotal)
int x[],seqtotal;
{
    int j,k,notecount,seqlen,storeval;

    notecount = 0;
    for (j = 0;j < seqtotal;j++)
    {
        seqlen = rand() % 10 + 5;
        for (k = 0;k < seqlen;k++)
        {
            storeval = (k+1) * (j+1);
            if (k == 0)
            {
                x[k+notecount] = storeval * 1000 + seqlen;
                printf("%d ",x[k+notecount]);
            }
        }
    }
}
```

```

    }
    else
    {
        x[k+notecount] = storeval;
        printf("%d ",x[k+notecount]);
    }
    notecount = notecount + seqlen;
    printf("\n");
}
return(notecount);
} /* end of Seqstore() function */
/***** MIC 7.22 *****/
void Seqxtot(x,notetotal)
int x[],notetotal;
{
    int j,seqcounter;
    seqcounter = 0;
    for (j = 0; j < notetotal; j++)
    {
        if (x[j] < 1000)
            printf("%d ",x[j]);
        else
        {
            seqcounter +=1;
            printf("\nsequence %d \n",seqcounter);
            printf("%d ",x[j]/1000);
        }
    }
    printf("\n");
} /* end of Seqxtot() function */
/***** END OF SEQSTORE.C *****/

```

## 程序运行实例

### SEQSTORE.EXE

a group of random-integer sequences are now being stored.  
here is how they are represented in the array:  
1005 2 3 4 5

2010 4 6 8 10 12 14 16 18 20  
3014 6 9 12 15 18 21 24 27 30 33 36 39 42  
4012 8 12 16 20 24 28 32 36 40 44 48  
5008 10 15 20 25 30 35 40

extracted sequence-group looks like this:

sequence 1  
1 2 3 4 5  
sequence 2  
2 4 6 8 10 12 14 16 18 20  
sequence 3  
3 6 9 12 15 18 21 24 27 30 33 36 39 42  
sequence 4  
4 8 12 16 20 24 28 32 36 40 44 48  
sequence 5  
5 10 15 20 25 30 35 40

## 函数组: Scorform.c

1. IntergerToPitch MIC7.23
2. PutPitch( ) MIC7.24
3. ScriptArray( ) MIC7.25

## 功能

将某一音乐参数的一组数据整理为适用于某一乐器的乐谱表。

## 注释

SCORFORM.C 函数组的功能如下: 主程序段先将一列升序整数存入一音符特性数组, 然后进行各种数据类型转换以一组音符名称和整数的形式将各音乐参数值显示在屏幕上。每一行都以 P、R、A、V 或 T 开头分别代表音高、节拍、发音方式、音量或音调等各音乐参数。其中音高的值以一个阿拉伯字母(代表音符)加上一个数字(代表所在音阶)来表示。

节拍值以相当于全音的几分之几的分数值来表示(例如, 22 表示全音符的  $1/22$ , 以此类推)。发音方式、音量、音质各参数值分别以相对的最大值的百分比来表示——譬如, V 为 450 表示音量相当于最大音量的一半。这种表示方法对应用于电子乐器很适宜, 但是若输出文件是作为人工演奏的乐谱, 则还需要将各输出值转译为传统乐谱的表示形式, 在这种情况下, 需要采用一个标准比例符号来将输出的百分比数值转换传统形式, 以及应用于音质参数数据。

## 编程提示

1. 编写一交互式程序, 为用户提供各种作曲算法以应用于各种音乐参数。
2. 在上述交互式程序中加入建栏、存盘功能以将谱好的音符表存盘。
3. 修改上述交互式程序使其可谱出含复合结构的曲子(比如说, 多声部、多段落等)。

## 程序清单

```
/* SCORFORM.C (compile a notelist score of data for 5
               musical parameters for output to the screen
               or printer) */
#include <ctype.h>
main()
{
    int j, datarray[87];
```

```

for(j = 0; j < 87; j++)
    datarray[j] = j; /* generate some integers for
                        parameter data; substitute
                        your own algorithm. */

printf("Notelist using R 1-1\n");
ScriptArray(datarray, 'P', j);
ScriptArray(datarray, 'R', j);
ScriptArray(datarray, 'A', j);
ScriptArray(datarray, 'V', j);
ScriptArray(datarray, 'T', j);
} /* end of main() */
/*=====*/
/* This module incorporates a recent C language addition:
   FUNCTION PROTOTYPES; delete this module if your C
   compiler doesn't support prototyping. */

char *IntegerToPitch(int num);
PutPitch(int num);
ScriptArray(int datarray[], char lnhead, int size);
/*===== MIC 7.23 =====*/
/* IntegerToPitch() function (an adaptation of
   MIC 2.3 Pitchtab(), which converts numbers
   to pitches */
#define MAXPIT 84
char *IntegerToPitch(num)
int num;
{
    static char pitch[4];
    static char *names[] = { "C", "C#", "D", "D#", "E", "F", "F#",
                              "G", "G#", "A", "A#", "B" };
    static char *rest = ("R");
    int pc, oct;

    if (num < 0 || num > MAXPIT)
        return (rest);
    else {
        pc = num % 12;
        oct = (num/12);
        sprintf(pitch, "%s%d", names[pc], oct);
        return (pitch);
    }
} /* end of IntegerToPitch() function */
/*===== MIC 7.24 =====*/
/* PutPitch() function */
PutPitch(num)
int num;
{
    printf("%4s", IntegerToPitch(num));
} /* end of PutPitch() function */
/*===== MIC 7.25 =====*/
/* ScriptArray() function */
ScriptArray(datarray, lnhead, size)
int datarray[];
char lnhead;
int size;
{
    int k;
    int j = 0;
    int offset = 0;

    while(j + offset < size)

```

```

printf("\n%c ", lnhead);
while(j < 10 && j + offset < size)
    switch(lnhead)
    {
        case 'P' :
            printf(" ");
            PutPitch(datarray[j++ + offset]);
            break;
        case 'R' :
            printf(" ");
            printf("%-4d", datarray[j++ + offset]);
            break;
        case 'A' :
            printf(" ");
            printf("%-4d", datarray[j++ + offset]);
            break;
        case 'V' :
            printf(" ");
            printf("%-4d", datarray[j++ + offset]);
            break;
        case 'T' :
            printf(" ");
            printf("%-4d", datarray[j++ + offset]);
            break;
        case ' ' :
            PutPitch(datarray[j++ + offset]);
            break;
    }
    offset += j;
    j = 0;
}
printf("\n");
} /* end of ScriptArray() function */
/*-----*/
/* END OF SCORFORM.C */

```

### 程序运行实例

SCORFORM.EXE

Notelist using R 1-1

P	C0	C#0	D0	D#0	E0	F0	F#0	G0	G#0	A0
P	A#0	B0	C1	C#1	D1	D#1	E1	F1	F#1	G1
P	G#1	A1	A#1	B1	C2	C#2	D2	D#2	E2	F2
P	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3
P	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4
P	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4
P	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5
P	A#5	B5	C6	C#6	D6	D#6	E6	F6	F#6	G6
P	G#6	A6	A#6	B6	C7	R	R			
R	0	1	2	3	4	5	6	7	8	9
R	10	11	12	13	14	15	16	17	18	19
R	20	21	22	23	24	25	26	27	28	29
R	30	31	32	33	34	35	36	37	38	39
R	40	41	42	43	44	45	46	47	48	49
R	50	51	52	53	54	55	56	57	58	59
R	60	61	62	63	64	65	66	67	68	69

R	70	71	72	73	74	75	76	77	78	79
R	80	81	82	83	84	85	86			
A	0	1	2	3	4	5	6	7	8	9
A	10	11	12	13	14	15	16	17	18	19
A	20	21	22	23	24	25	26	27	28	29
A	30	31	32	33	34	35	36	37	38	39
A	40	41	42	43	44	45	46	47	48	49
A	50	51	52	53	54	55	56	57	58	59
A	60	61	62	63	64	65	66	67	68	69
A	70	71	72	73	74	75	76	77	78	79
A	80	81	82	83	84	85	86			
V	0	1	2	3	4	5	6	7	8	9
V	10	11	12	13	14	15	16	17	18	19
V	20	21	22	23	24	25	26	27	28	29
V	30	31	32	33	34	35	36	37	38	39
V	40	41	42	43	44	45	46	47	48	49
V	50	51	52	53	54	55	56	57	58	59
V	60	61	62	63	64	65	66	67	68	69
V	70	71	72	73	74	75	76	77	78	79
V	80	81	82	83	84	85	86			
T	0	1	2	3	4	5	6	7	8	9
T	10	11	12	13	14	15	16	17	18	19
T	20	21	22	23	24	25	26	27	28	29
T	30	31	32	33	34	35	36	37	38	39
T	40	41	42	43	44	45	46	47	48	49
T	50	51	52	53	54	55	56	57	58	59
T	60	61	62	63	64	65	66	67	68	69
T	70	71	72	73	74	75	76	77	78	79
T	80	81	82	83	84	85	86			

## 生成 MIDI 格式的输出乐谱文件的过程

实现将 BASIC 程序的输出文件在 MIDI 音乐合成器上演奏的最简捷的方法是选用商用时序软件。这种软件可以处理格式化 ASCII 文件并对其进行转换。我们使用的软件是系统设置协会开发的 PSS 系统。该系统软件被作为可插在任一种(与 IBM 兼容的)个人计算机扩展板上的 MIDI 接口卡形式,含两根传输线:输入线与输出线用以连接 MIDI 接口至音乐合成器。PSS 系统含有一套完整的录音、放音、编辑软件以及 TOPM.EXE 程序用以将 BASIC 程序的输出文件转化为可应用于音乐合成器的音乐序列形式。而且,该系统还含有一 PCV.EXE 程序以直接将输出文件转化为标准乐谱形式,经过转换后的文件就可以通过 IBM 兼容软件包 PERSONAL COMPOSER 打印成一页页的乐谱形式。这种软件包在各类音像书店均有出售。

## 将程序改写为 MIDI 格式

生成一输出文件将其转化为 MIDI 格式,然后在音乐合成器上奏出的步骤如下:

1. 在 BASIC 程序中写入 ASCII 建立文件、存盘指令。
2. 将 ASCII 输出文件送入一个乐句转换程序。

3. 输入商用时序软件环境。

4. 在 MIDI 音乐合成器上演奏此乐句。

步骤 1 要求在程序中的恰当位置加入指令将音项数据记录存盘。该文件可被看作乐谱，因为它包含了演奏某一乐句所需的所有信息。

下列指令生成并打开一文件：

`f1 = fopen("mididata", "w");` 然后，必须给每一音项记录所要求的一系列参变量赋值。若只希望试听一下用某一算谱出的曲子，那么，对音量参数、节拍参数、发音方式参数以及发声通道参数赋定值即可。

如下例所示：

```
int j, notes, seed;
unsigned long int start_time = 0; /* noteon start time in clock
                                     ticks */
unsigned int channel = 0; /* MIDI channels can be 0 - 15 */
unsigned port = 0; /* for PROMIDI ASCII file, port must be
                    set to 0 */
unsigned int pitch; /* MIDI note number (0-127) */
unsigned int velocity; /* MIDI "loudness" (0-127) */
unsigned int artdur; /* articulated note-duration */
float duration; /* total time in clock ticks allowed for note */
float articulation; /* factor to shorten (articulate) the note */
```

接下来，在程序中通常需要将音高参数显示在屏幕上的地方，加入语句以显示出当前音的音高、音量、音长、发音方式以及发声通道等各项参数的值。增加变量 `noteon` 的值（音乐起始时刻），并将当前记录存盘。

```
pitch = rand() % 65 + 30; /* lowest pitch will be 30 */
channel = rand() % 15; /* randomly assign MIDI channel */
velocity = rand() % 50 + 75; /* randomly assign volume */
duration = 192 / (rand() % 9 + 2); /* 192 clock ticks per
                                     quarter-note */
artdur = duration * articulation; /* shorten the time

fprintf(f1, "%lu %u %u %u %u\n", start_time, pitch, velocity,
                                     artdur, channel, port);
start_time += duration; /* advance noteon time */
```

注意，在此例中，变量 `duration` 的不同取值并不用于本文件中，而是用于加到原起始时间（单位是 T，即时钟信号数）上以计算下一音的发音时间。变量 `artdur` 用于存放缩短了的音长值（音断式发音）。缩短音长是为了在当一个 8 声部的复音合成器的每一声部单独发音时，给乐器提供选通时间。若乐器被设置为可演奏和弦，则每一音符的起始时刻允许相互重叠。

MIC7.26 中的 `Midifile()` 函数展示了如前所述的 MIDI 格式 ASCII 文件的读写原则，请仔细研读以便透彻理解函数中各变量的作用，并对不同的音程序列应用上述以求熟练掌握。

## 函数：Midifile() MIC7.27

功能

整理一音项表, 该表中应含有四种音乐参数(音高、音长、速度、发音方式), 并以 PSS 系统所要求的 ASCII 格式文件形式存盘。

## 注释

输出文件必须通过 TOPM.EXE 程序转换成为音乐数据文件, 该程序可由系统设计协会(SDA)的音乐软件提供。

## 编程提示

1. 编写一交互式程序, 为用户提供各种数据生成算法以便应用于各音乐参数。
2. 修改 Midifile() 函数的以实现谱出具有复合结构的曲子(比如, 多声部曲、多个乐段曲等等)。

## 程序清单

```
/* MIDIFILE.C (compiles a notelist score of data for 4 musical
parameters and writes it to a file in FROMIDI STUDIO SYSTEM ASCII
format)..
```

```
NOTE: File must then be converted to a sequence file using
TOPM.EXE, provided as part of sequencing software by Systems
Design Associates, 5068 Plano Parkway, Suite 121, Plano, Texas
75075 */
```

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    void Midifile();
    Midifile();
} /* end of main */
/*----- MIC 7.26 -----*/
/* Midifile() function */
void Midifile()
{
    FILE *f1, *fopen();
    int j, notes, seed;
    unsigned int pitch, velocity, channel, port, artdur;
    float duration, articulation;
    unsigned long int start_time = 0;
    channel = 0; /* assign notelist to MIDI Channel 1 */
    port = 0; /* for FROMIDI ASCII file port must be set to 0 */

    f1 = fopen("mididata", "w");
    printf("enter an integer seed for the random generator\n");
    scanf("%d", &seed);
    printf("how many events in your notelist?\n");
    scanf("%d", &notes);
    printf("enter articulation as a real > 0 & <= .85\n");
    scanf("%f", &articulation);
    srand(seed);
    for(j = 0; j < notes; j++)
```



```

    {
        pitch = rand() % 65 + 30; /* lowest pitch will be 30 */
        channel = rand() % 15; /* randomly assign MIDI channel */
        velocity = rand() % 50 + 75; /* randomly assign volume */
        duration = 192/(rand() % 9 + 2); /* 768 clock ticks per
                                         whole-note */
        artdur = duration * articulation; /* shorten the time
                                         note sounds */
        printf("%1u %u %u %u %u %u\n", start_time, pitch, velocity,
            artdur, channel, port);
        fprintf(f1, "%1u %u %u %u %u %u\n", start_time, pitch, velocity,
            artdur, channel, port);
        start_time += duration; /* advance noteon time */
    }
    fclose(f1);
} /* end of Midifile() function */

```

### 程序运行实例

#### MIDIFILE.EXE

```

enter an integer seed for the random generator    301
how many events in your notelist? 45
enter articulation as a real > 0 & <= .85    .50

```

```

0 83 87 130 2 0
153 77 84 92 12 0
262 70 78 81 4 0
358 36 93 326 5 0
742 60 108 81 6 0
838 42 103 108 5 0
966 93 103 217 2 0
1222 82 115 326 6 0
1606 84 121 81 11 0
1702 57 77 163 4 0
1894 44 92 92 6 0
2003 59 96 130 1 0
2156 94 88 163 10 0
2348 62 96 326 9 0
2732 52 100 130 5 0
2885 88 108 92 11 0
2994 54 75 326 9 0
3378 85 114 81 13 0
3474 33 75 326 9 0
3858 43 103 81 5 0
3954 83 93 72 12 0
4039 62 121 130 14 0
4192 92 109 108 13 0
4320 37 78 64 2 0
4396 44 117 81 6 0

```

4492 52 97 81 6 0  
4588 43 123 108 8 0  
4716 33 114 64 8 0  
4792 91 121 108 6 0  
4920 40 84 217 11 0  
5176 90 78 92 13 0  
5285 71 96 72 7 0  
5370 74 117 72 13 0  
5455 33 85 108 0 0  
5583 33 112 326 13 0  
5967 45 119 217 13 0  
6223 48 81 81 7 0  
6319 35 76 326 3 0  
6703 49 118 163 1 0  
6895 92 85 326 4 0  
7279 87 100 108 12 0  
7407 39 104 92 4 0  
7516 88 77 163 5 0  
7708 42 122 217 5 0  
8599 45 102 108 1 0

## 附录 A 程序头文件(.h)

```

/*===== MIC_H3 =====*/
/* FILENAME: SCPTPRSR.H */
#include <stdio.h>
#include <ctype.h>
#include "fgetline.c"
#ifndef SYNCLAVIE_C
#include "synclavi.c"
#endif
#include "getfnam.c"
#include "stoi.c"
#ifndef MAXDATA
#define MAXDATA 1000
#endif
#define MAXSIZE 80;
int parray[MAXDATA];
int rarray[MAXDATA];
int aarray[MAXDATA];
int varray[MAXDATA];
int tarray[MAXDATA];
int psize, rsize, asize, vsize, tsize;
#ifndef inputfile
FILE *inputfile;
#endif
FILE *getfnam();

/*===== MIC_H4 =====*/
/* TEXT.H */
#include <stdio.h>
#include <ctype.h>
#include <alloc.h>
#include "randmain.c"
#define MAXDATA 100
#define MAXVOCAB 100
#define MAXWORDSIZE 80
#define TRUE 1
#define FALSE -1
static char vowels[6] = ("AEIOU");
static char punctuation[10] = (".,:;!-/?");
char *phones[MAXVOCAB];
char *dipth[MAXVOCAB];
char *words[MAXVOCAB];

char *verbs[MAXVOCAB];
char *adjectives[MAXVOCAB];
char *nouns[MAXVOCAB];
char *articles[3] = {"a", "the", "an"};
int wct = 0;
int dip = 0;
int ph = 0;
int vbct = 0;
int adct = 0;
int nct = 0;
int act = 3;
/* size of arrays */

```

## 附录 B 子程序库(.C)

```
/******  
/* MIC_SP1.0  ARRAY.C  
/* purpose: function-group to handle array  
/* processing (retro, invert, rotate, etc.) */  
#include "array.pro"  
/****** MIC_SP1.1 ******/  
invshape(shape, inv, size)  
{  
    int shape[], inv[], size;  
    {  
        int j;  
        for(j = 0; j < size; j++)  
        {  
            if(shape[j] > 0)  
                inv[j] = shape[j] - abs(shape[j] * 2);  
            else  
                inv[j] = shape[j] + abs(shape[j] * 2);  
        }  
    }  
}  
/****** MIC_SP1.2 ******/  
getshape(datarray, shapearray, size)  
{  
    int datarray[];  
    int shapearray[];  
    int size;  
    {  
        int a, b, j;  
        for (j = 0; j < size - 1; j++)  
        {  
            a = datarray[j];  
            b = datarray[j + 1];  
            shapearray[j] = b - a;  
        }  
    }  
}  
/****** MIC_SP1.3 ******/  
fputArray(fp, datarray, size)  
{  
    FILE *fp;  
    int datarray[];  
    int size;  
    {  
        int k;  
        int j = 0;  
        int offset = 0;  
    }  
}
```

```

        while(j + offset < size)
        {
            while(j < 18 && j + offset < size)
                fprintf(fp, "%3d ", datarray[j++ + offset]);

            offset += j;
            j = 0;
            fprintf(fp, "\n");
        }
    }
/*===== MIC_SP1.4 =====*/
PutArray(datarray, size)
    int datarray[];
    int size;
{
    int k;
    int offset = 0;

    while(j + offset < size)
    {
        while(j < 18 && j + offset < size)
            printf("%3d ", datarray[j++ + offset]);

        offset += j;
        j = 0;
        printf("\n");
    }
}
/*===== MIC_SP1.5 =====*/
PutLn()
{
    int j;
    printf("\n/*");
    for(j = 0; j < 68; j++)
        printf("%c", '-');
    printf("*/\n");
}
/*===== MIC_SP1.6 =====*/
FputLn(fp)
    FILE *fp;
{
    int j;
    fprintf(fp, "\n/*");
    for(j = 0; j < 68; j++)
        fprintf(fp, "%c", '-');
    fprintf(fp, "*/\n");
}
/*===== MIC_SP1.7 =====*/
RetroArray(datarray, size)
    int datarray[];
    int size;
{
    int retrograde[MAXDATA];
    int ask;
    int j, k;

    for(j = size-1, k = 0; j >= 0; j--, k++)
    {
        PutPitch(datarray[j]);
        retrograde[k] = datarray[j];
    }
}

```

```

printf("\n");
printf("\nDo you want to store it in the data array? yes (1) no (2) ");
scanf("%d", &ask);
if( ask == 1)
{
    for(j = 0; j < size; j++)
        datarray[j] = retrograde[j];
}
/*===== MIC_SP1.8 =====*/
getmax(datarray, size)
int datarray[];
int size;
{
    int j, max;
    for(max = datarray[0], j = 1; j < size; j++)
        if(datarray[j] > max)
            max = datarray[j];
    return max;
}
/*===== MIC_SP1.9 =====*/
getmin(datarray, size)
int datarray[], size;
{
    int j, min;
    for(min = datarray[0], j = 1; j < size; j++)
        if(datarray[j] < min)
            min = datarray[j];
    return min;
}
/*===== MIC_SP1.10 =====*/
float getmaxf(datarray, size)
float datarray[];
int size;
{
    int j;
    float max;
    for(max = datarray[0], j = 1; j < size; j++)
        if(datarray[j] > max)
            max = datarray[j];
    return max;
}
/*===== MIC_SP1.11 =====*/
float getminf(datarray, size)
float datarray[];
int size;
{
    int j;
    float min;
    for(min = datarray[0], j = 1; j < size; j++)
        if(datarray[j] < min)
            min = datarray[j];
    return min;
}
/*===== MIC_SP1.12 =====*/
FrotateArray(fp, size, datarray)
FILE *fp;
int size;
int datarray[];
{
    register int rotation, j, k, m, tran, ask;

```

```

    int disarray[MAXDATA];
    tran = 0;
    printf("ROTATIONS");
    for(rotation=0; rotation < size; rotation++)
    {
        printf("\nArray rotation no %d\n", rotation);
        fprintf(fp, "\n/* Array rotation no %d */\n", rotation);
        for(j=0; j < size - rotation; j++)
            disarray[j] = datarray[j + rotation] + tran;

        for(k=0; k < rotation; k++)
            disarray[j + k] = datarray[k] + tran;

        ScriptArray(disarray, 'P', size);
        PscriptArray(fp, disarray, 'P', size);
    }
}

/*===== MIC_SP1.13 =====*/
RotateArray(datarray, disarray, size, rotation)
int datarray[];
int disarray[];
int size, rotation;
{
    register int j, k;
    int temp[MAXDATA];
    for(j = 0; j < size - rotation; j++)
        temp[j] = datarray[j + rotation];

    for(k = 0; k < rotation; k++)
        temp[j + k] = datarray[k];

    for(j = 0; j < size; j++)
        disarray[j] = temp[j];
}

/*===== MIC_SP1.14 =====*/
imbricate(darray, imarray, start, imsize, imbeg)
int darray[], imarray[];
int start, imsize, imbeg;
{
    int j;
    for(j= start; j < start + imsize; j++)
        imarray[imbeg++] = darray[j];
}

/*===== MIC_SP1.15 =====*/
sumarray(array, size)
int array[];
int size;
{
    int j;
    int sum = 0;

    for (j = 0; j < size; j++)
        sum += abs(array[j]);

    return (sum);
}

/*===== MIC_SP2.0 INPUTYOU.C =====*/
/* MIC_SP2.0 INPUTYOU.C
   purpose: user input of data to array */
/*===== MIC_SP2.1 =====*/
InputYourOwn(datarray)
int datarray[];

```

```

    int j, k, size;
#ifdef debug
printf("\n\t\t\t\t\tentering inputyourown\n");
#endif

printf("\nHow many numbers do you want to input? ");
scanf("%d", &size);
for(j = 0; j < size; j++)
{
    printf("Input integer # %d \t-\b", j+1);
    scanf("%d", &k);
    if(k >= 0 && k < 85)
        datarray[j] = k;
    else
    {
        do{
            printf("\nBetween 0 and 84, remember?\n");
            printf("Input integer # %d \t-\b", j+1);
            scanf("%d", &k);
            datarray[j] = k;
        } while ( k < 0 || k > 84);
    }
}
printf("\nHere is what you typed in\n");
for(j = 0; j < size; j++)
    printf("%d ", datarray[j]);
printf("\n\n");
return (size);
}

/*****
 * MIC_SP4.0 GETFNAM.C
 * purpose: obtain filename from console for
 *           reading, writing, or appending */
/***** MIC_SP4.1 *****/
FILE *getfnam(use,mode)

char *use,*mode;
{
    FILE *fp,*fopen();
    char name[80],c;
    int ask;

    printf("\n\nFilename for %s ? ",use);
    scanf("%s",name);

    if (name[0] == '='){
        c = *mode;
        switch(c){
            case 'r': return (stdin);
            case 'w':
            case 'a': return (stdout);
        }
    }

    if ((fp = fopen(name,mode)) == NULL){
        printf("Error opening %s for %s.",name,use);
        printf("\nEnter 1 to try again; any other key to abort.");
        scanf("%d", &ask);
        if(ask == 1)
            fp = getfnam(use,mode);
    }
}

```



```

        else exit(0);
    }
    return (fp);
}
/*=====*/
/*****
/* MIC_SP6.0      SCORFORM.C
    purpose: to convert integer data to score
              (notelist) format for use by
              synthesizers.

#include <ctype.h>
#include "scorform.pro"
/*===== MIC_SP6.1 =====*/
ValidPitch(pitch)
    char *pitch;
{
    int j, valid;

    for(j = 0; j < 3; j++)
    {
        if(pitch[j] <= 'G' || pitch[j] >= 'A' || pitch[j] >= '0' ||
           pitch[j] <= '9' || pitch[j] == '#' || pitch[j] == '-')
            valid = 1;
        else
            return(0);
    }
    return(valid);
}
/*===== MIC_SP6.2 =====*/
#define REST 85
#define INVALID 86
int PitchToInteger(pitch)
    char *pitch;
{
    int PitchNumber = 24, oct = 3, accidental = 0, i, isvalid;
    static int letternums[] = {9,11,0,2,4,5,7};
    /* A B C D E F G */
    char c;

    isvalid = ValidPitch(pitch);
    if(isvalid == 0)
        return(INVALID);
    if(pitch[1] == 'F') /* convert all flats to - */
    {
        pitch[1] = '-';
        if(pitch[0] == 'C')
            pitch[2] = 1;
    }

    while( (c = *pitch++) != '\0')
    {
        if(iscntrl(c))
            return (INVALID);
        if(c == '\0')
            return (INVALID);
        if (c == 'R')
            return REST;

        else if (c >= 'A' && c <= 'G')
        {

```

```

        i = c - 'A';
        PitchNumber = letternums[i];
    }
    else if (c >= '0' && c <= '9')
        oct = c - '0';
    else switch(c)
    {
        case '#': accidental += 1; break;
        case '+': accidental += 1; break;
        case 'b': accidental -= 1; break;
        case '-': accidental -= 1; break;
        case 'x': accidental += 2; break;
        default: printf("\ninvalid %c", c);
                return (INVALID);
    }

    PitchNumber = ((PitchNumber + 12 + accidental) % 12) +
        (oct * 12);
    return (PitchNumber);
}

/***** MIC_SP6.3 *****/
#define MAXPIT 84
char *IntegerToPitch(num)
int num;
{
    static char pitch[4];
    static char *names[] = { "C", "C#", "D", "D#", "E", "F", "F#",
                              "G", "G#", "A", "A#", "B" };
    static char *rest = {"R"};
    int pc, oct;

    if (num < 0 || num > MAXPIT)
        return (rest);
    else {
        pc = num % 12;
        oct = (num / 12);
        sprintf(pitch, "%s%d", names[pc], oct);
        return (pitch);
    }
}

/***** MIC_SP6.4 *****/
PutPitch(num)
int num;
{
    printf("%-4s", IntegerToPitch(num));
}

/***** MIC_SP6.5 *****/
ScriptArray(datarray, lnhead, size)
int datarray[];
char lnhead;
int size;
{
    int k;
    int j = 0;
    int offset = 0;

    while(j + offset < size)
    {
        printf("\n%c ", lnhead);
        while(j < 15 && j + offset < size)

```

```

        switch(lnhead)
        {
            case 'P' :
                printf(" ");
                PutPitch(datarray[j++ + offset]);
                break;
            case 'R' :
                printf(" ");
                printf("%t-4d", datarray[j++ + offset]);
                break;
            case 'A' :
                printf(" ");
                printf("%t-4d", datarray[j++ + offset]);
                break;
            case 'V' :
                printf("%t-4d", datarray[j++ + offset]);
                break;
            case 'T' :
                printf(" ");
                printf("%t-4d", datarray[j++ + offset]);
                break;
            case ' ' :
                PutPitch(datarray[j++ + offset]);
                break;
        }
        offset += j;
        j = 0;
    }
    printf("\n");
}
/*****
/* MIC_SP7.0 SYNCLAVI.C
    purpose: provide screen output and data filing
            in computer music notelist format; an
            expansion of SCORFORM.C */

#include "synclavi.pro"
#include <stdio.h>
#include <ctype.h>
/***** MIC_SP6.1 *****/
ValidPitch(pitch)
    char *pitch;
{
    int j, valid;

    for(j = 0; j < 3; j++)
    {
        if(pitch[j] <= 'G' || pitch[j] >= 'A' || pitch[j] >= '0' ||
           pitch[j] <= '9' || pitch[j] == '*' || pitch[j] == '-')
            valid = 1;
        else
            return(0);
    }
    return(valid);
}

/***** MIC_SP6.2 *****/
#define REST 85
#define INVALID 86
/*
 * PitchToInteger: convert a pitch string to an integer value

```

```

/*
int PitchToInteger(pitch)
char *pitch;
{
    int PitchNumber = 24, oct = 3, accidental = 0, i, isvalid;
    static int letternums[] = {9,11,0,2,4,5,7}; /* A B C D E F G */
    char c;

    isvalid = ValidPitch(pitch);
    if(isvalid == 0)
        return(INVALID);
    if(pitch[1] == 'F') /* convert all flats to - */
    {
        pitch[1] = '-';
        if(pitch[0] == 'C')
            pitch[2] -= 1;
    }

    while( (c = *pitch++) != '\0')
    {
        if(iscntrl(c))
            return (INVALID);
        if(c == '\0')
            return (INVALID);
        if (c == 'R')
            return REST;

        else if (c >= 'A' && c <= 'G')
        {
            i = c - 'A';
            PitchNumber = letternums[i];
        }
        else if (c >= '0' && c <= '9')
            oct = c - '0';
        else switch(c)
        {
            case '#': accidental += 1; break;
            case '+': accidental += 1; break;
            case 'b': accidental -= 1; break;
            case '-': accidental -= 1; break;
            case 'x': accidental += 2; break;
            default: printf("\ninvalid %c", c);
                    return (INVALID);
        }
    }

    PitchNumber = ((PitchNumber + 12 + accidental) % 12) + (oct * 12);
    return (PitchNumber);
}

/*===== NIC_SP6.3 =====*/
#define MAXPIT 84
char *IntegerToPitch(num)
int num;
{
    static char pitch[4];
    static char *names[] = { "C","C#","D","D#","E","F","F#",
                             "G","G#","A","A#","B" };
    static char *rest = {"R"};
    int pc,oct;

    if (num < 0 || num > MAXPIT)

```

```

        return (rest);
    else {
        pc = num % 12;
        oct = (num/12);
        sprintf(pitch,"%s%d", names[pc], oct);
        return (pitch);
    }
}

/*===== MIC_SF6.4 =====*/
PutPitch(num)
    int num;
{
    printf("%-4s", IntegerToPitch(num));
}

/*===== MIC_SF7.1 =====*/
FputPitch(fp,num)
    FILE *fp;
    int num;
{
    fprintf(fp,"%-4s", IntegerToPitch(num));
}

/*===== MIC_SF6.5 =====*/
ScriptArray(datarray, lnhead, size)
    int datarray[];
    char lnhead;
    int size;
{
    int k;
    int j = 0;
    int offset = 0;

    while(j + offset < size)
    {
        printf("\n%c ", lnhead);
        while(j < 15 && j + offset < size)
            switch(lnhead)
            {
                case 'P' :
                    printf(" ");
                    PutPitch(datarray[j++ + offset]);
                    break;
                case 'R' :
                    printf(" ");
                    printf("%-4d", datarray[j++ + offset]);
                    break;
                case 'A' :
                    printf(" ");
                    printf("%-4d", datarray[j++ + offset]);
                    break;
                case 'V' :
                    printf(" ");
                    printf("%-4d", datarray[j++ + offset]);
                    break;
                case 'T' :
                    printf(" ");
                    printf("%-4d", datarray[j++ + offset]);
                    break;
                case ' ' :
                    PutPitch(datarray[j++ + offset]);
                    break;
            }
    }
}

```

```

        offset += j;
        j = 0;
    }
    printf("\n");
}
/*===== MIC_SP7.2 =====*/
FscriptArray(fp, datarray, lnhead, size)
    FILE *fp;
    int datarray[];
    char lnhead;
    int size;
{
    int k;
    int j = 0;
    int offset = 0;

    while(j + offset < size)
    {
        fprintf(fp, "\n%c ", lnhead);
        while(j < 15 && j + offset < size)
            switch(lnhead)
            {
                case 'P' :
                    fprintf(fp, " ");
                    FputPitch(fp, datarray[j++ + offset]);
                    break;
                case 'R' :
                    fprintf(fp, " ");
                    fprintf(fp, "%-4d", datarray[j++ + offset]);
                    break;
                case 'A' :
                    fprintf(fp, " ");
                    fprintf(fp, "%-4d", datarray[j++ + offset]);
                    break;
                case 'V' :
                    fprintf(fp, " ");
                    fprintf(fp, "%-4d", datarray[j++ + offset]);
                    break;
                case 'T' :
                    fprintf(fp, " ");
                    fprintf(fp, "%-4d", datarray[j++ + offset]);
                    break;
                case ' ' :
                    FputPitch(fp, datarray[j++ + offset]);
                    break;
            }
        offset += j;
        j = 0;
    }
    fprintf(fp, "\n");
}
/*===== MIC_SP8.0 SCREEN.C =====*/
/* MIC_SP8.0 SCREEN.C
   purpose: ansi screen escape characters */

#include "screen.pro"
#define BOLD      "\033[1m"
#define UNDER    "\033[4m"
#define BLINK     "\033[5m"
#define REVERSE   "\033[7m"

```

```

#define NOBOLD      "\033[2;2m"
#define NOUNDER    "\033[2;4m"
#define NOBLINK    "\033[2;5m"
#define NOREVERSE  "\033[2;7m"
#define NORM       "\033[0m"      /* cancels special modes */
#define CURSUP     "\033[A"
#define CURDN     "\033[B"
#define CURR      "\033[C"
#define CURL      "\033[D"
#define HOME      "\033[H"
#define BIGTOP    "\033#1"
#define BIGBOT    "\033#4"
#define NORMAL    "\033#5"
#define WIDE      "\033#6"

int vt;
/*===== MIC_SP8.1 =====*/
/* centerf      centers string      */
centerf(string)
char *string;
{
    int l;
    l = (80 - strlen(string)) / 2;
    while (l-- > 0) putchar(' ');
    printf("%s\n", string);
}
/*===== MIC_SP8.2 =====*/
centerwdef(string)
char *string;
{
    int l;
    l = (40 - strlen(string)) / 2;
    while (l-- > 0) putchar(' ');
    printf("%s%s\n", WIDE, string, NORM);
}
/*===== MIC_SP8.3 =====*/
centerbigf(string)
char *string;
{
    int l;
    l = (40 - strlen(string)) / 2;
    while (l-- > 0) putchar(' ');
    printf("%s%s\n", BIGTOP, string, NORM);
    l = (40 - strlen(string)) / 2;
    while (l-- > 0) putchar(' ');
    printf("%s%s\n", BIGBOT, string, NORM);
}
/*===== MIC_SP8.4 =====*/
cls()
{
    if (vt == 1)
        vt52cls();
    else
        vt100cls();
}
/*===== MIC_SP8.5 =====*/
ScrCurs(row, col)
int row, col;
{
    if (vt == 1)
        vt52scrCurs(row, col);
    else

```

```

        vt100scrCurs(row, col);
    }
    /*===== MIC_SP8.6 =====*/
    vt52cls()
    {
        printf("\033H\033J");
    }
    /*===== MIC_SP8.7 =====*/
    vt52scrCurs(row, col)
    int row, col;
    {
        printf("\033Y%c%c", row + 31, col + 31);
    }
    /*===== MIC_SP8.8 =====*/
    vt100cls()
    {
        printf("\033[H\033[2J");
    }
    /*===== MIC_SP8.9 =====*/
    vt100scrCurs(row, col)
    int row, col;
    {
        printf("\033[%d;%dH", (row) + 1, (col) + 1);
    }
    /*===== MIC_SP8.10 =====*/
    getvt()
    {
        vt = 1; /* for ibm pc with ansi driver installed */
    }
    /*=====*/
    /*=====*/
    /* MIC_SP9.0      GETNUM.C */
    getnum()
    {
        int j;
        char string[10];

        gets(string);
        j = atoi(string);
        return(j);
    }
    /*=====*/
    /* MIC_SP10.0     FUSARRAY.C */

    #include "fusarray.pro"
    /*===== MIC_SP10.1 =====*/
    FusePair(fp, dat, fusarray, size)
    FILE *fp;
    int dat[];
    int fusarray[][2];
    int size;
    {
        int j, row, col;
        printf("\n\tFUSIONS\n");
        fprintf(fp, "\n\tFUSIONS\n");
        printf("\n\n\tIntervals to be fused : \n\t");
        fprintf(fp, "\n\n\tIntervals to be fused : \n\t");
        PutArray(dat, size);
        FputArray(fp, dat, size);
    }

```



```

row = 0;
for(j=0; j < size - 1; j++)
{
    col = 0;
    fusarray[row][col++] = dat[j];
    fusarray[row+1][col] = dat[j + 1];
}
}
/*===== MIC_SP10.2 =====*/
fusion(dat, fusarray, size, limit)
int dat[];
int fusarray[][2];
int size;
int limit;
{
    int x, j, k, col;
    int row = 0;
    int m = 0;
    int n = 0;

    for(x = 2; x <= size - 1; x++)
    {
        for(j = 0, k = size - 1; j <= size - x, k >= x; j++, k--)
        {
            m = summation(dat, j, x); /* sum dat from j to x */
            n = negsum(dat, k, x); /* sum dat from k down to x */
            if(m <= limit)
            {
                col = 0;
                fusarray[row][col++] = dat[j];
                fusarray[row+1][col] = m;
            }
            if(n <= limit)
            {
                col = 0;
                fusarray[row][col++] = dat[k];
                fusarray[row+1][col] = n;
            }
        }
        row++;
    }
    return(row);
}
/*===== MIC_SP10.3 =====*/
fusion2(dat, fusarray, size)
int dat[];
int fusarray[][2];
int size;
{
    int j, k, m, n, row, col;

    row = 0;
    for(j=0; j < size - 3; j++)
    {
        k = dat[j] + dat[j + 1];
        m = dat[j + 2] + dat[j + 3];
        col = 0;
        fusarray[row][col++] = k;
        fusarray[row+1][col] = m;
    }
    return(row);
}

```

```

fusion1(dat, fusarray, size)
    int dat[];
    int fusarray[][2];
    int size;
{
    int j, k, m, n, row, col;

    row = 0;
    for(j=0; j < size - 4; j++)
    {
        k = dat[j] + dat[j + 1] + dat[j + 2];
        m = dat[j + 1] + dat[j + 2] + dat[j + 3];

        col = 0;
        fusarray[row][col++] = k;
        fusarray[row++][col] = dat[j + 3];

        col = 0;
        fusarray[row][col++] = dat[j];
        fusarray[row++][col] = m;
    }
    return(row);
}

/*===== MIC_SP10.4 =====*/
summation(darray, start, num)
    int darray[];
    int start;
    int num;
{
    int j;
    int k = 0;
    for(j = 1; j <= num; j++)
        k += darray[start + j];
    return(k);
}

/*===== MIC_SP10.5 =====*/
negsum(darray, start, num)
    int darray[];
    int start;
    int num;
{
    int j = 0;
    int k;
    for(k = 1; k <= num; k++)
        j += darray[start - k];
    return(j);
}

/*****
/* MIC_SP11.1    MATRIX.C */

#include "quicksort.c"
#include "matrix.pro"
/*===== MIC_SP11.1 =====*/
/* should be sorted first */
PutMatrixFq(matrix, fq, rsize, csize)
    int matrix[][2];
    int fq[];

```

```

        int rsize;
        int csize;

    {
        int j, k;
        printf("\n fusion\t\t\tfrequency");
        printf("\n ----- \t\t\t-----\n");
        for(j = 0; j < rsize; j+=fq[j])
        {
            for(k = 0; k < csize; k++)
                printf("%d\t", matrix[j][k]);
            printf("\t\t\t%d\n", fq[j]);
        }
    }

/*===== MIC_SP11.2 =====*/
FputMatrixFq(fp, matrix, fq, rsize, csize)
    FILE *fp;
    int matrix[][2];
    int fq[];
    int rsize;
    int csize;

{
    int j, k;
    fprintf(fp, "\n fusion\t\t\tfrequency");
    fprintf(fp, "\n ----- \t\t\t-----\n");

    for(j = 0; j < rsize; j+=fq[j])
    {
        for(k = 0; k < csize; k++)
            fprintf(fp, "%d\t", matrix[j][k]);
        fprintf(fp, "\t\t\t%d\n", fq[j]);
    }
}

/*===== MIC_SP11.3 =====*/
FindMatrixFreq(matrix, fq, size)
    int matrix[][2];
    int fq[];
    int size;

{
    int j, k, key, key2;
    printf("\nFinding matrix frequency\nPlease be patient\n");
    for(j = 0; j < size; j++)
        fq[j] = 0;

    for(j = 0; j < size; j++)
    {
        key = matrix[j][0];
        key2 = matrix[j][1];
        for(k = 0; k < size; k++)
        {
            if(key == matrix[k][0] && key2 == matrix[k][1])
                fq[k] += 1;
        }
    }
}

/*===== MIC_SP11.4 =====*/
MatrixSort(matrix, rsize, csize)
    int matrix[][2];
    int rsize;
    int csize;

```

```

(
    int i, j, k, temp, temp2;
    printf("\nSorting matrix\nPlease be patient\n");
    /* sort rows */
    for(i = 0; i < rsize-1; ++i)
        for(j = i + 1; j < rsize; ++j)
            if(matrix[i][0] > matrix[j][0])
            {
                temp = matrix[i][0];
                temp2 = matrix[i][1];
                matrix[i][0] = matrix[j][0];
                matrix[i][1] = matrix[j][1];
                matrix[j][0] = temp;
                matrix[j][1] = temp2;
            }
    /* sort cols */
    for(i = 0; i < rsize-1; ++i)
        for(j = i + 1; j < rsize; ++j)
            if(matrix[i][1] > matrix[j][1] && matrix[i][0]
                == matrix[j][0])
            {
                temp = matrix[i][1];
                matrix[i][1] = matrix[j][1];
                matrix[j][1] = temp;
            }
}
/*===== MIC_SP11.5 =====*/
MatrixQuickSort(matrix, rsize, csize)
    int matrix[][2];
    int rsize;
    int csize;
{
    int i, j, tmp;
    int temp[5000];
    int temp2[5000];
    printf("\nSorting matrix\nqqs\n");
    /* sort rows */
    for(i = 0; i < rsize; i++)
        temp[i] = matrix[i][0];
    QuickSort(temp, 0, rsize-1);
    /* sort cols */
    for(i = 0; i < rsize-1; ++i)
        for(j = i + 1; j < rsize; ++j)
            if(matrix[i][1] > matrix[j][1] && matrix[i][0]
                == matrix[j][0])
            {
                tmp = matrix[i][1];
                matrix[i][1] = matrix[j][1];
                matrix[j][1] = tmp;
            }
    for(i = 0; i < rsize-1; ++i)
        matrix[i][0] = temp[i];
}
/*===== MIC_SP11.6 =====*/

```

```

PutMatrix(matrix, nrow, ncol)
int matrix[][2];
int nrow;
int ncol;

{
    int j, k;
    for(j = 0; j < nrow; j++)
    {
        for(k = 0; k < ncol; k++)
            printf("%d\t", matrix[j][k]);
        printf("\n");
    }
}

/*===== MIC_SP11.7 =====*/
FputMatrix(fp, matrix, nrow, ncol)
FILE *fp;
int matrix[][2];
int nrow;
int ncol;

{
    int j, k;
    for(j = 0; j < nrow; j++)
    {
        for(k = 0; k < ncol; k++)
            fprintf(fp, "%d\t", matrix[j][k]);
        fprintf(fp, "\n");
    }
}

/*===== MIC_SP11.8 =====*/
ZeroMatrix(matrix, rowsize, colsize)
int matrix[];
int rowsize;
int colsize;

{
    int j, k;
    for (j = 0; j < rowsize; j++)
    {
        for(k = 0; k < colsize; k++)
            matrix[j, k] = 0;
    }
}

/*===== MIC_SP12.0 QUICKSOR.C =====*/

/*===== MIC_SP12.0 =====*/
/* call : quicksort(sorted, 0, size - 1); */
/* sorted = int array, size = length of array */
Quicksort(item, left, right)
int item[], left, right;
{
    register int i, j;
    char x, y;

    i = left;
    j = right;
    x = item[(left + right)/2];
    do {
        while(item[i] < x && i < right) i++;
        while(x < item[j] && j > left) j--;
        if(i <= j)

```

```

        {
            y = item[i];
            item[i] = item[j];
            item[j] = y;
            i++; j--;
        }
    } while(i <= j);
    if(left < j) QuickSort(item, left, j);
    if(i < right) QuickSort(item, i, right);
}
/*===== MIC_SP11.9 =====*/
quick(item, count)
    int item[];
    int count;
{
    QuickSort(item, 0, count-1);
}

/*===== MIC_SP12.0 STATFUNC.C =====*/
/* calls:
        a = mean(data, num);
        std = StdDev(data, num);
        m = median(data, num);
        md = FindMode(data, num);
        regress(data, num);
*/
/*===== MIC_SP12.1 =====*/
#define MAX 1000
float mean(), StdDev();

float
mean(data, num)
    int data[];
    int num;
{
    int t, sum = 0;
    float avg;

    for(t = 0; t < num; ++t)
        sum += data[t];
    avg = sum/num;
    return avg;
}
/*===== MIC_SP12.2 =====*/
float
StdDev(data, num)
    int data[];
    int num;
{
    register int t;
    float std, avg;
    double temp, sqrt();
    avg = mean(data, num);
    std = 0;
    for(t = 0; t < num; ++t)
    {
        std += ((data[t] - avg) * (data[t] - avg));
    }
}

```

```

        std /= num;
        temp = std;
        temp = sqrt(temp);
        std = temp;
        return std;
    }
    /*===== MIC_SP12.3 =====*/
    median(data, num)
        int data[];
        int num;
    {
        register int t;
        int dtemp[MAX];

        for(t = 0; t < num; ++t)
            dtemp[t] = data[t];
        quick(dtemp, num);
        return dtemp[num/2];
    }
    /*===== MIC_SP12.4 =====*/
    FindMode(data, num)
        int data[];
        int num;
    {
        register int t, w;
        int oldmode;
        int count, oldcount, md;

        oldmode = 0;
        oldcount = 0;
        for(t = 0; t < num; ++t)
        {
            md = data[t];
            count = 1;
            for(w = t + 1; w < num; ++w)
                if(md == data[w]) count++;
            if(count > oldcount)
            {
                oldmode = md;
                oldcount = count;
            }
        }
        return oldmode;
    }
    /*===== MIC_SP12.5 =====*/
    regress(data, num)
        int data[];
        int num;
    {
        float a, b, x, x_avg, y_avg, temp, temp2;
        float cor;
        int data2[580];
        float StdDev();
        int t, min, max;
        char s[80];

        y_avg = 0;
        for(t = 0; t < num; ++t)
            y_avg += data[t];
        y_avg /= num;
    }

```

```

x_avg = 0;
for(t = 0; t <= num; ++t)
    x_avg += t;
x_avg /= num;

temp=0;
temp2=0;
for(t = 0; t <= num; ++t)
{
    temp += (data[t]- y_avg) * (t-x_avg);
    temp2 += (t-x_avg) * (t-x_avg);
}
b = temp/temp2;
a = y_avg - (b * x_avg);
for(t = 0; t < num; ++t)
    data2[t] = t + 1;
cor = temp/(num);
cor=cor/(StdDev(data, num) * StdDev(data2, num));

printf("\nRegression equation is Y = %f + %f * X\n",a,b);
printf("Correlation coefficient: %f\n",cor);
/*printf("\nPlot data points and regression line? (y/n)");
gets(s);
if(toupper(*s) == 'N') return;
for(t = 0; t < num * 2; ++t)
    data2[t] = a+(b*(t+1));
min = getmin(data, num) * 2;
max = getmax(data, num) * 2;
scatterplot(data,num, min, max, num*2);
scatterplot(data2, num*2, min, max, num*2);
gets(s);*/
}

/*===== MIC_SP12.6 =====*/
get_num()
{
    char s[80];
    gets(s);
    return(atoi(s));
}

/*****
/* MIC_SP13.0    SCRIPTPR.S */

/*===== MIC_SP13.1 =====*/
#include "scriptprs.h"

GetLines(fp)
    FILE *fp;
{
    char *line;
    int lsize, j;

    line = (char *) malloc(80);
    printf("\nReading file:\n");

    while (!feof(fp))
    {
        lsize = fgetline(fp, line, 80);
        printf("%s",line);
        getlnhead(line, lsize);
    }
}

```



```

        printf("File read complete");
    }
    /*===== MIC_SP13.2 =====*/
    PutData()
    {
        int j;
        printf("\n\nHere is the pitch array size= %d\n", psize);
        for(j = 0; j < psize; j++)
            printf("%d ", parray[j]);
        printf("\n\nHere is the rrythm array size= %d\n", rsize);
        for(j = 0; j < rsize; j++)
            printf("%d ", rarray[j]);
        printf("\n\nHere is the articulation array size= %d\n",
            asize);
        for(j = 0; j < asize; j++)
            printf("%d ", aarray[j]);
        printf("\n\nHere is the volume array size= %d\n", vsize);
        for(j = 0; j < vsize; j++)
            printf("%d ", varray[j]);
        printf("\n\nHere is the rte array size= %d\n", tsize);
        for(j = 0; j < tsize; j++)
            printf("%d ", tarray[j]);
    }
    /*===== MIC_SP13.3 =====*/
    getlnhead(line, lnsiz)
    char line[];
    int lnsiz;
    {
        int j, size;
        char lnhead;

        size = 0;
        j = 0;
        while (line[j] == ' ')
            j++;
        lnhead = toupper(line[j]);
#ifdef DEBUG
        printf("\nlnhead %c ", lnhead);
#endif
        switch(lnhead)
        {
            case 'P' : size = getparray(line, lnsiz, parray, psize);
                        psize = size;
                        break;
            case 'R' : size = getnarray(line, lnsiz, rarray, rsize);
                        rsize = size;
                        break;
            case 'A' : size = getnarray(line, lnsiz, aarray, asize);
                        asize = size;
                        break;
            case 'V' : size = getnarray(line, lnsiz, varray, vsize);
                        vsize = size;
                        break;
            case 'T' : size = getnarray(line, lnsiz, tarray, tsize);
                        tsize = size;
                        break;
            case '/' : j = SkipComment(line, j, lnsiz);
                        break;
            case 'D' : printf("\nDefines not yet supported");
                        break;
        }
    }

```

```

/*===== MIC_SP13.4 =====*/
SkipComment(line, offset, lnsz)
char *line;
int offset;
int lnsz;
{
    do {
        switch(line[offset])
        {
            case '*' : if(line[offset+1] == '/')
                        return(offset + 2);
                        break;

            case '\n': lnsz = fgetline(inputfile, line, 80);
                        printf("line read:\n");
                        puts(line);
                        getlnhead(line, lnsz);
                        break;

            default : offset++;
                        break;
        }
    } while(--lnsz > 0);
    return(offset);
}

/*===== MIC_SP13.5 =====*/
getparray(line, lnsz, datarray, xsize)
char line[];
int lnsz;
int datarray[];
int xsize;
{
    int i, j, k, pc, isp;
    char pitch[4];

    i = j = k = 0;
#ifdef DEBUG
    printf("\ngetparray size=%d \n", xsize);
    puts(line);
#endif
    do {
        j = 0;
        for(i = 0; i < 4; i++)
            pitch[i] = ' ';
        isp = 0;
        pc = 86;
        while(line[k] != ' ' && j < 3 && line[k] != 'P')
        {
            pitch[j++] = line[k++];
            --lnsz;
            isp = 1;
        }
        if(isp == 1)
        {
            pitch[j] = '\0';
            pc = PitchToInteger(pitch);
        }
        k++;
        if(pc != 86)
        {
            datarray[xsize++] = pc;
        }
    } while(lnsz > 0);
#ifdef DEBUG
    printf("\ngetparray done\n");
    puts(line);
#endif
}

```

```

        printf("\nparray[%d] = %d pc=%d " ,xsize-1,
               datarray[xsize-1],pc);
        PutPitch(datarray[xsize-1]);
    #endif
    )
    ) while (line[k] != '\n' && --lnsize > 0);
    return(xsize);
}
/*===== MIC_SPI3.6 =====*/
getnarray(line, lnsize, datarray, xsize)
char *line;
int lnsize;
int datarray[];
int xsize;
{
    int i, j, k;
    char num[4];

    j = k = 0;
    do{
        j = 0;
        for(i = 0; i < 4; i++)
            num[i] = ' ';
        while(line[k] != ' ' && j < 3)
        {
            num[j++] = line[k++];
            --lnsize;
        }
        num[3] = '\0';
        k++;
        if(isdigit(num[0]) && (!isspace(num[0])) )
        {
            datarray[xsize++] = atoi(num);
        }
    #ifdef DEBUG
        printf("\ndatarray[%d]=%d num = %s ",xsize-1,
               datarray[xsize-1],num);
    #endif
    }
    ) while (line[k] != '\n' && --lnsize > 0);

    return(xsize);
}

/*===== MIC_SPI5.0 FGETLINE.C =====*/
fgetline(fp, line, MaxSize)
FILE *fp;
char *line;
int MaxSize;
{
    int ch;
    int ReturnSize = 0;

    if ( MaxSize == 0 )
    {
        *line = '\0';
        return( 0 );
    }

    while ( ( --MaxSize > 0 ) && ((ch = getc(fp)) != EOF )

```

```

        line[ReturnSize++] = ch;
    if (ch == '\n')
        line[ReturnSize++] = ch;

    line[ReturnSize] = '\0';
    return(ReturnSize);
}
/*****

/* MIC_SP16.0 STOI.C */

stoi(string)
char string[];
{
    int j, ival, retval;
    retval = 0;

    for(j = 0; string[j] >= '0' && string[j] <= '9'; ++j)
    {
        ival = string[j] - '0';
        retval *= 10;
        retval += ival;
    }
    return(retval);
}
/*****

/* MIC_SP19.0 RANDMAIN.C */
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include "getnum.c"
/***** MIC_SP19.1 *****/
RandMain(datarray)
int datarray[];
{
    int i, hi, lo;
    int size = 0;

    printf("\nHow many random numbers do you want?\t");
    size = getnum();
    printf("Enter lowest number in gamut\t");
    lo = getnum();
    printf("Enter highest number in gamut\t");
    hi = getnum();

    randomize();
    for(i = 0; i < size; i++)
    {
        datarray[i] = irand(lo, hi);
        printf("%d ", datarray[i]);
    }
    return(size);
}
/***** MIC_SP19.2 *****/
irand(l, h)
int l, h;
{
    int range, min;
    long tm;

```

```

        min = (l <= h) ? l : h;
        range = abs(h - l) - 1;
        return (rand() % range + min);
    }
}
/*****

/*****===== MIC_SP20.0 =====*/
/* GPRINTF: Used like PRINTF except the output is sent to the */
/* screen in graphics mode at the specified co-ordinate.      */
/*                                                              */
int gprintf( int *xloc, int *yloc, char *fmt, ... )
{
    va_list argptr;          /* Argument list pointer */
    char str[140];           /* Buffer to build sting into */
    int cnt;                 /* Result of SPRINTF for return */

    va_start( argptr, format ); /* Initialize va_ functions */
    cnt = vsprintf( str, fmt, argptr ); /* prints string to buffer */
    outtextxy( *xloc, *yloc, str ); /* Send string in graphics mode */
    *yloc += textheight( "H" ) + 2; /* Advance to next line */

    va_end( argptr );        /* Close va_ functions */

    return( cnt );           /* Return the conversion count */
}
/*****===== MIC_SP21.0 =====*/
#include <ctype.h>
fgetword( Word, WordLength, fp )
char *Word;
int WordLength;
FILE *fp;

{
    int ch;

    if ( WordLength <= 0 )
    {
        *Word = '\0';
        return;
    };
    ch = *Word++ = getc( fp );
    if ( isspace( ch ) )
        while ( --WordLength > 0 ) /* skip white space */
        {
            ch = *Word++ = getc( fp );
            if ( !isspace( ch ) )
            {
                ungetc( ch, fp );
                break;
            }
        }
    else
        while ( --WordLength > 0 )
        {
            ch = *Word++ = getc( fp );
            if ( isspace( ch ) )
            {
                ungetc( ch, fp );
                break;
            }
        }
}

```

```

    }
    if ( WordLength == 0 )
        ungetc( *( Word - 1 ), fp );
    *( Word - 1 ) = '\0';
}

/*****
===== MIC_SP23.0 =====
/* TEXTLIB.C */
#include "text.h"
SoundSummary()
{
    int j;

    printf("\nSounds\n");
    for(j = 0; j < ph; j++)
        printf("phones %d = %s\n", j, phones[j]);
    printf("%d phones\n", ph);
    for(j = 0; j < dip; j++)
        printf("diphthongs %d = %s\n", j, diphth[j]);
    printf("%d diphthongs\n", dip);
}
/*****
WordSummary()
{
    int j;
    for(j = 0; j < wct; j++)
        printf("%s\n", words[j]);
}
/*****
MakeDrivel(fp)
FILE *fp;
{
    int j, num, howmany, prt;

    printf("\nHow many words do you want?\n");
    howmany = getnum();
    printf("\nWrite to file? (1) = yes\n");
    prt = getnum();
    for(j = 0; j < howmany; j++)
    {
        num = irand(0, wct - 1);
        printf("%s ", words[num]);
        if((j % 6) == 5)
            printf("\n");
        if(prt == 1)
        {
            fprintf(fp, "%s ", words[num]);
            if((j % 6) == 5)
                fprintf(fp, "\n");
        }
    }
}
/*****
MakeText(fp)
FILE *fp;
{
    int j, num, howmany, prt;

```

```

printf("\nHow many words do you want?\t");
howmany = getnum();
printf("\nWrite to file? (1) = yes\t");
prt = getnum();

for(j = 0; j < howmany; j++)
{
    num = irand(0, ph - 1);
    printf("%s", phones[num]);
    if(prt == 1)
        fprintf(fp, "%s", phones[num]);
    num = irand(0, dip - 1);
    printf("%s", diph[num]);
    if(prt == 1)
        fprintf(fp, "%s", diph[num]);
}
}
/*****
getphone(string, position)
char string[];
int position;
{
    char *cstr;
    int j = 0;

    cstr = (char *) malloc(80);
    while(isvowel(string[position]) == FALSE)
    {
        if(!isspace(string[position]))
            cstr[j++] = string[position++];
        else
            position++;
        if(ispunct(cstr[j]))
        {
            cstr[j+1] = ' ';
            j++;
        }
    }
    cstr[j] = '\0';
    if(strlen(cstr) > 0)
        phones[ph++] = cstr; /* ph is global counter */
    return(position);
}
*****/
getdiph(string, position)
char string[];
int position;
{
    char *vstr;
    int j = 0;

    vstr = (char *) malloc(80);
    while(isvowel(string[position]) == TRUE)
    {
        if(!isspace(string[position]))
            vstr[j++] = string[position++];
        else
            position++;
    }
    vstr[j] = '\0';
    if(strlen(vstr) > 0)

```

```

        depth[dip++] = vstr;      /* dip is global counter
    return(position);
}
/*****
chop(string)
    char string[];
{
    int position = 0;

    while(position < strlen(string))
    {
        if(isvowel(string[position]) == FALSE)
            position = getphonews(string, position);
        else if(isvowel(string[position]) == TRUE)
            position = getdiphws(string, position);
        else
            position++;
    }
}
/*****
isvowel(ch)
    char ch;
{
    int j;

    for(j = 0; j < 5; j++)
        if(toupper(ch) == vowels[j])
            return(TRUE);
    return(FALSE);
}
/*****
getdiphws(string, position)    /* keep the spaces */
    char string[];
    int position;
{
    char *vstr;
    int j = 0;

    vstr = (char *) malloc(80);
    while(isvowel(string[position]) == TRUE)
        vstr[j++] = string[position++];
    vstr[j] = '\0';
    if(strlen(vstr) > 0)
        depth[dip++] = vstr;      /* dip is global counter */
    return(position);
}
/*****
getphonews(string, position)
    char string[];
    int position;
{
    char *cstr;
    int j = 0;

    cstr = (char *) malloc(80);
    while(isvowel(string[position]) == FALSE)
    {
        cstr[j++] = string[position++];
        if(ispunct(cstr[j - 1]))
            cstr[j++] = ' ';
    }
}

```



```

        cstr[j] = '\0';
        if(strlen(cstr) > 0)
            phones(ph++) = cstr;          /* ph is global counter */
        return(position);
    }
    /*****
ExtractWord(string, position)
    char string[];
    int position;
    {
        int j = 0;
        char *word;

        word = (char *) malloc(80);
        while(!isspace(string[position]))
            word[j++] = string[position++];
        word[j++] = '\0';
        words[wct++] = word;
        return(position);
    }
    *****/
ChopWords(string)
    char string[];
    {
        int position = 0;
        while(position < strlen(string))
        {
            if(isspace(string[position]))
                position++;
            else
                position = ExtractWord(string, position);
        }
    }
    /*****
===== MIC_SP24.0 =====
**FGETWORD.C**
#include <ctype.h>
fgetword( Word, WordLength, fp )
    char *Word;
    int WordLength;
    FILE *fp;

    {
        int ch;

        if ( WordLength <= 0 )
        {
            *Word = '\0';
            return;
        }
        ch = *Word++ = getc( fp );
        if ( isspace( ch ) )
            while ( --WordLength > 0 )          /* skip white space */
            {
                ch = *Word++ = getc( fp );
                if ( !isspace( ch ) )
                {
                    ungetc( ch, fp );
                    break;
                }
            }
    }

```

```

else
    while ( --WordLength > 0 )
    {
        ch = *Word++ = getc( fp );
        if ( isspace( ch ) )
        {
            ungetc( ch, fp );
            break;
        }
    }
    if ( WordLength == 0 )
        ungetc( *( Word - 1 ), fp );
    *( Word - 1 ) = '\0';
}
/*===== MIC_SP25.0 =====*/
/*GETFNAM.C*/
FILE *getfnam(use,mode)
char *use,*mode;
{
    FILE *fp,*fopen();
    char name[80],c;
    int ask;

    printf("\n\nFilename for %s ? ",use);
    scanf("%s",name);

    if (name[0] == '='){
        c = *mode;
        switch(c){
            case 'r': return (stdin);
            case 'w':
            case 'a': return (stdout);
        }
    }

    if ((fp = fopen(name,mode)) == NULL){
        printf("Error opening %s for %s.",name,use);
        printf("\nEnter 1 to try again; any other key to abort.");
        scanf("%d", &ask);
        if(ask == 1)
            fp = getfnam(use,mode);
        else exit(0);
    }
    return (fp);
}

```

## 附录 C 函数源文件(.pro)

```
/*===== MIC_P1 =====*/
/* MIRROR.PRO */

/* Source file: mirror.c */
/* function prototypes */
main(void);
PutPeform(FILE *fp, int ShapeSize, char motname[]);
WriteComment(FILE *fp, int data[], int size, char comment[]);
loop(FILE *fp, int shape[], int ShapeSize, int num, int center,
char motname[]);
PutChord(FILE *fp, int chord[], int ChordSize);
GetChord(int up[], int down[], int chord[], int size);
GetPatterns(int disarray[], int up[], int down[], int size);
OpenChord(FILE *fp);
EndChord(FILE *fp);
GetCenter(int up[], int down[], int center);
PutRhythm(FILE *fp, int chord[], int ChordSize);

/*===== MIC_P2 =====*/
/* SETS.PRO */

/* Source file: sets.c */
/* function prototypes */

GetIndex( int imsize, int size, int setsize);
GetSetSize(void);
GetImsize(void);
Order( int test[], int ordered[], int vector[], int setnumber[],
int setsize, int k);
ShowOriginal( FILE *fp, int original[], int test[], int setsize);
GetTestSets( int test[], int parray[], int datarray[],
int original[], int imarray[], int setsize, int imsize, int k);
GetSetnumber( int ordered[], int setsize);
NotSetError( FILE *fp, int setsize);
PutSetData( FILE *fp, int test[], int setsize, int ordered[],
int setnumber, int setfq[]);
```

```

PutSetSummary( FILE *fp, int setfq[], int setsize);
PutTable( FILE *fp);
GetData( int datarray[], int setfq[], int parray[]);
WhichArray(void);
showdata( FILE *fp, int datarray[], int parray[], int size);
ShowImbricated( FILE *fp, int parray[], int imarray[],
               int imsize, int size);
NormalOrder( int parray[], int ordered[], int setsize);
endvector( int vector[], int size);
rotate( int datarray[], int disarray[], int size);
set3cmp( int test[]);
set4cmp( int test[]);
set5cmp( int test[]);
set6cmp( int test[]);
PrintSetTable( struct sets set[], int setsize);
FprintSetTable( FILE *fp, struct sets set[], int setsize);
PutSubset( int setnumber, int setsize);
FputSubset( FILE *fp, int setnumber, int setsize);
PutSetArray( int SetArray[], int setsize);
PutSetInversion( int SetInversion[], int setsize);
PutSetArrayPitches( int SetArray[], int setsize);
PutSetInversionPitches( int SetInversion[], int setsize);
FputSetArrayPitches( FILE *fp, int SetArray[], int setsize);
FputSetInversionPitches( FILE *fp, int SetInversion[],
                        int setsize);
PutSetVector( int SetVector[]);
FputSetArray( FILE *fp, int SetArray[], int setsize);
FputSetInversion( FILE *fp, int SetInversion[], int setsize);
FputSetVector( FILE *fp, int SetVector[]);
PrintSetInfo( struct sets set[], int setnumber, int setsize);
FprintSetInfo( FILE *fp, struct sets set[], int setnumber,
              int setsize);
PutContains( int setnumber, int setsize);
PutSimilar( int setnumber, int setsize);
FputContains( FILE *fp, int setnumber, int setsize);
FputSimilar( FILE *fp, int setnumber, int setsize);

/*===== MIC_P3 =====*/
/* FILENAME: ARRAY.PRO
   purpose: function prototypes
*/
invshape(int shape[], int inv[], int size);
getshape(int datarray[], int shapearray[], int size);
FputArray(FILE *fp, int datarray[], int size);
PutArray(int datarray[], int size);
PutLn();
FputLn(FILE *fp);
RetroArray(int datarray[], int size);
getmax(int datarray[], int size);
getmin(int datarray[], int size);
float getmaxf(float datarray[], int size);
float getminf(float datarray[], int size);
FrotateArray(FILE *fp, int size, int datarray[]);
RotateArray(int datarray[], int disarray[], int size,
            int rotation);
imbricate(int darray[], int imarray[], int start, int imsize,
          int imbeg);
sumarray(int array[], int size);

```

```

/*===== MIC_P4 =====*/
/* FILENAME: POEM.PRO */
/* Source file: poem.c */
/* function prototypes */
int GetVocab(void);
void MakePoem(void);
void VocabSummary(void);
main(void);
GetVocab(FILE *fp, char *part[]);
assign(FILE *fp, char *part[], int position);
void VocabSummary(void);
void MakePoem(FILE *fp);
SelectWord(FILE *fp, char *part[], int ct, int flag);

```

```

/*===== MIC_P5 =====*/
/* FILENAME PHONE.PRO */
/* Source file: phone.c */
/* function prototypes */
SoundSummary(void);
WordSummary(void);
MakeDrivel(FILE *fp);
MakeText(FILE *fp);
getphone(char string[], int position);
getdiph(char string[], int position);
chop(char string[]);
isvowel(char ch);
getdiphths(char string[], int position);
getphonews(char string[], int position);
ExtractWord(char string[], int position);
ChopWords(char string[]);

```

```

/*===== MIC_P8 =====*/
/* FILENAME MIRROR.PRO */
/* Source file: mirror.c */
/* function prototypes */
main(void);
PutPeform(FILE *fp, int ShapeSize, char motname[]);
WriteComment(FILE *fp, int data[], int size, char comment[]);
loop(FILE *fp, int shape[], int ShapeSize, int num, int center,
char motname[]);
PutChord(FILE *fp, int chord[], int ChordSize);
GetChord(int up[], int down[], int chord[], int size);
GetPatterns(int disarray[], int up[], int down[], int size);
OpenChord(FILE *fp);
EndChord(FILE *fp);
GetCenter(int up[], int down[], int center);
PutRhythm(FILE *fp, int chord[], int ChordSize);

```

```

/*===== MIC_P9 =====*/
/* FILENAME MATRIX.PRO */
/* file: matrix.pro */
/* function prototypes */
PutMatrixFq(int matrix[][2], int fq[], int rsize, int csize);
PputMatrixFq(FILE *fp, int matrix[][2], int fq[], int rsize,
int csize);
FindMatrixFreq(int matrix[][2], int fq[], int size);
MatrixSort(int matrix[][2], int rsize, int csize);

```

```

MatrixQuickSort(int matrix[][2], int rsize, int csize);
PutMatrix(int matrix[][2], int nrow, int ncol);
FputMatrix(FILE *fp, int matrix[][2], int nrow, int ncol);
ZeroMatrix(int matrix[], int rowsize, int colsize);

```

```

/*===== MIC_P12 =====*/
/* FILENAME CONSTEL.PRO */
/* Source file: constel.c */
/* function prototypes */
main(void);
con(void);
FindFreq(struct pitchrec pit[], int size);
GetPitchStats(FILE *outputfile, int statarray[], int size);
FusionMain(void);
PutFusPitch(FILE *fp, int fusarray[][2], int size);

```

```

/*===== MIC_P13 =====*/
/* FILENAME FUSARRAY.PRO */
/* Source file: fusarray.c */
/* function prototypes */
FusePair(FILE *fp, int dat[], int fusarray[][2], int size);
fusion(int dat[], int fusarray[][2], int size, int limit);
fusion2(int dat[], int fusarray[][2], int size);
fusion3(int dat[], int fusarray[][2], int size);
summation(int darray[], int start, int num);
negsum(int darray[], int start, int num);

```

```

/*===== MIC_P14 =====*/
/* FILENAME SCRIPTPRS.PRO */
/* function prototypes */
GetLines(FILE *fp);
PutData(void);
getlnhead(char line[], int lsize);
SkipComment(char *line, int offset, int lsize);
getparray(char line[], int lsize, int datarray[], int xsize);
getnarray(char *line, int lsize, int datarray[], int xsize);

```

```

/*===== MIC_P15 =====*/
/* FILENAME SETS.PRO */
/* source file sets.c */

GetIndex( int imsize, int size, int setsize);
GetSetSize(void);
GetImsize(void);
Order( int test[], int ordered[], int vector[], int setnumber[],
      int setsize, int k);
ShowOriginal( FILE *fp, int original[], int test[], int setsize);
GetTestSets( int test[], int parray[], int datarray[],
      int original[], int imarray[], int setsize, int imsize, int k);
GetSetnumber( int ordered[], int setsize);
NotSetError( FILE *fp, int setsize);
PutSetData( FILE *fp, int test[], int setsize, int ordered[],
      int setnumber, int setfq[]);
PutSetSummary( FILE *fp, int setfq[], int setsize);
PutTable( FILE *fp);
GetData( int datarray[], int setfq[], int parray[]);
WhichArray(void);
showdata( FILE *fp, int datarray[], int parray[], int size);

```

```

ShowImbricated( FILE *fp,int parray[],int imarray[], int imsize,
               int size);
NormalOrder( int parray[], int ordered[], int setsize);
endvector( int vector[], int size);
rotate( int datarray[], int disarray[], int size);
set3cmp( int test[]);
set4cmp( int test[]);
set5cmp( int test[]);
set6cmp( int test[]);
PrintSetTable( struct sets set[], int setsize);
FprintSetTable( FILE *fp, struct sets set[], int setsize);
PutSubset( int setnumber, int setsize);
FputSubset( FILE *fp, int setnumber, int setsize);
PutSetArray( int SetArray[], int setsize);
PutSetInversion( int SetInversion[], int setsize);
PutSetArrayPitches( int SetArray[], int setsize);
PutSetInversionPitches( int SetInversion[], int setsize);
FputSetArrayPitches( FILE *fp, int SetArray[], int setsize);
FputSetInversionPitches( FILE *fp, int SetInversion[], int setsize);
PutSetVector( int SetVector[]);
FputSetArray( FILE *fp, int SetArray[], int setsize);
FputSetInversion( FILE *fp, int SetInversion[], int setsize);
FputSetVector( FILE *fp, int SetVector[]);
PrintSetInfo( struct sets set[], int setnumber, int setsize);
FprintSetInfo( FILE *fp, struct sets set[], int setnumber,
               int setsize);
PutContains( int setnumber, int setsize);
PutSimilar( int setnumber, int setsize);
FputContains( FILE *fp, int setnumber, int setsize);
FputSimilar( FILE *fp, int setnumber, int setsize);
/*****

```