计算机知识普及系列丛书

TURBOC 音乐编程指南

李雯　王真华　编写
张明　王　旭　审校

学苑出版社

计算机知识普及系列丛书

TURBO C 音乐编程指南

李 雯 王真华 编写
张 明 审校

学苑出版社
1994.

(京)新登字 151 号

内 容 提 要

本书是一本介绍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 Borland'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。

欲购本书的用户，请直接与北京 8721 信箱联系，电话：2562329，邮码：100080。

计算机知识普及系列丛书

TURBO C 音乐编程指南

编　　写：李　雯　王真华
审　　校：张　明
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：施园印刷厂
开　　本：787×1092　1/16
印　　张：20.375　　**字　数**：483 千字
印　　数：1～4000 册
版　　次：1994 年 2 月北京第 1 版第 1 次
ISBN7－5077－0821－7/TP・19
本册定价：19.00 元

目　录

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 Tubo 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，在此表示最诚致的谢意。

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观用途．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，若希望符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 Trubo 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[illegible]，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 prinf 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，比如在程序开头如入下列语句:

```
#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．某一个程序可前命名为 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 fnuction 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．这种格式最普通，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。人类还无法更深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，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机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传来决定的．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．