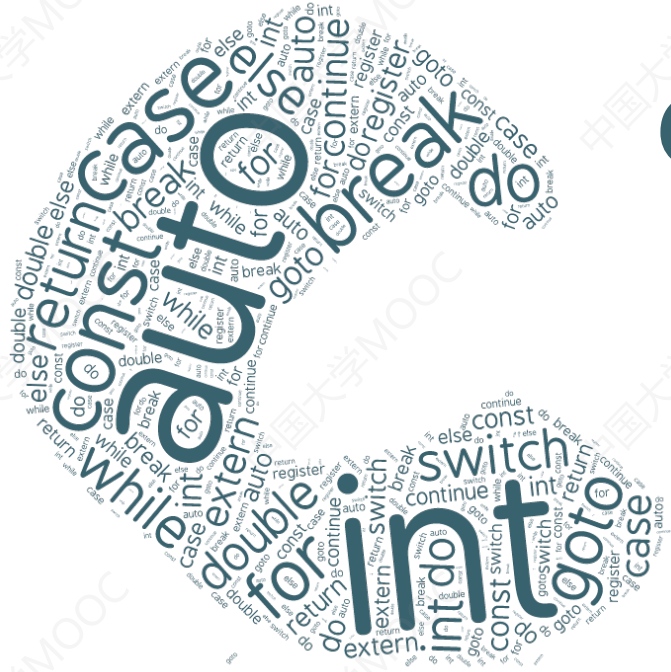


chapter 9

文件与图形

目录
content

9.1 文件的基本概念

文件：指存放在外部介质上的一组相关数据的有序集合。

这个数据集有一个名称，叫做文件名。实际上在前面的各章中我们已经多次使用了文件，例如源程序文件、目标文件、可执行文件、库文件 (头文件)等。

文件
- **文件编码的方式**
  - 文本文件
  - 二进制文件
- **文件缓冲类型**
  - 缓冲型文件
  - 非缓冲型文件

9.1.1 文本文件与二进制文件

文本文件:

文本文件在磁盘中存放时每个字符对应一个字节，用于存放对应的ASCII码。

例如，数1357在内存中的存储形式为:

```
00000101  01001101
```

在文本文件中的保存形式为:

```
ASCII: 00110001   00110011   00110101   00110111
对应的    ↓          ↓          ↓          ↓
十进制码:  1          3          5          7
```

共占用4个字节。

文本文件可在屏幕上按字符显示，例如源程序文件就是文本文件，由于是按字符显示，因此能读懂文件内容。但一般占存储空间较多，而且要花费转换时间。

9.1.1 文本文件与二进制文件

二进制文件:

二进制文件是按二进制的编码方式来存放文件的。

例如， 数1357的存储形式为:

```
00000101    01001101
```

只占二个字节。二进制形式输出数值，可以节省外存空间和转换时间，但不便于阅读。

9.1.2 缓冲性文件系统

ANSI C语言中把文件的输入/输出功能作为标准库函数的一部分，以提高程序的可移植性。各种C语言系统都遵循ANSI C标准定义了一组完整的标准输入/输出操作函数，这组标准输入/输出操作函数称为缓冲性文件系统(Buffered File System)。

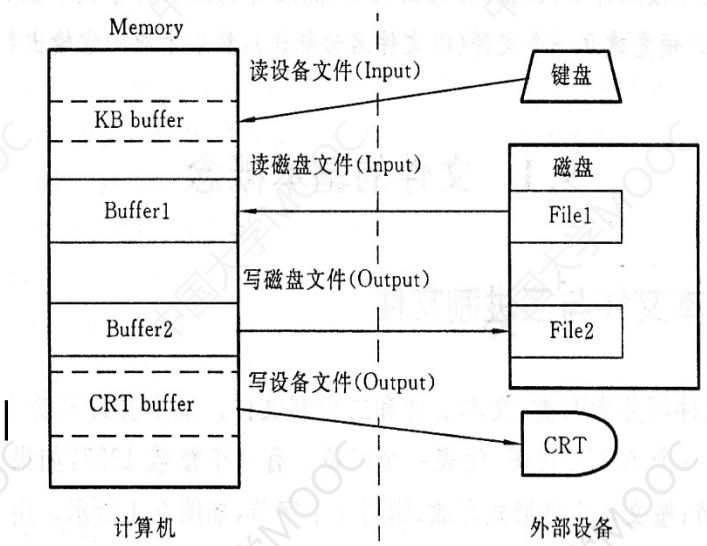
所谓**缓冲性文件系统**是指能够自动地在内存区为每个正在使用的文件名开辟一个缓冲区的系统。

1. 当从磁盘向内存读取数据时，

首先一次性从磁盘文件中将一批数据读入内存，

再从缓冲区中逐个将数据送到对应的内存空间中。

2.从内存向磁盘输出数据时，必须首先输出到缓冲区，再一起输出到磁盘文件中。



9.2 文件类型指针

在C语言中用一个指针变量指向一个文件，这个指针称为文件指针。通过文件指针就可对它所指的文件进行各种操作。该结构类型由系统定义，取名FILE,并在stdio.h中定义（#include “stdio.h”）：

```
typedef struct
{
        int             level;          //缓冲区“满”或“空”的程度
        unsigned        flags;          //文件状态标志
        char            fd;             //文件描述符号
        unsigned char   hold;           //如无缓冲区则不读取字符串
        int             bsize;          //缓冲区大小
        unsigned char   _FAR* buffer;   //数据传输缓冲区指针
        unsigned char   _FAR* curp;     //文件缓冲区位置
        unsigned        istemp;         //临时文件指示器
        short           token;          //用于有效性检查
}FILE;
```

9.2 文件类型指针

定义说明文件指针的一般形式为：

```
FILE *指针变量标识符;
```

其中**FILE**应为大写，它实际上是由系统定义的一个结构，该结构中含有文件名、文件状态和文件当前位置等信息。在编写源程序时不必关心**FILE**结构的细节。

```
FILE *fp;
```

表示**fp**是指向**FILE**结构的指针变量，通过**fp**即可找存放某个文件信息的结构变量，然后按结构变量提供的信息找到该文件，实施对文件的操作。习惯上也笼统地把**fp**称为指向一个文件的指针。

9.3 文件操作与相关函数

文件处理过程通常要经历如下三个步骤：

打开文件 → 文件读/写 → 关闭文件

文件在进行读写操作之前要先**打开**，使用完毕要**关闭**。所谓打开文件，实际上是建立文件的各种有关信息，并使文件指针指向该文件，以便进行其它操作。关闭文件则断开指针与文件之间的联系，也就禁止再对该文件进行操作。

在C语言中，**文件操作**都是由**库函数**来完成的。在本章内将介绍主要的文件操作函数。

9.3.1 文件的打开（fopen函数）：

fopen函数用来打开一个文件，其函数原形为：

```
FILE* fopen(const char *filename, const char *mode);
```

其调用的一般形式为：

```
if((文件指针名=fopen(文件名,使用文件方式))==NULL)
```

其中，

“**文件指针名**”必须是被说明为FILE 类型的指针变量；

“**文件名**”是被打开文件的文件名，为字符串常量或字符串数组；

“**使用文件方式**”是指文件的类型和操作要求。

9.3.1 文件的打开（fopen函数）：

例9.1 文件的打开1

```
FILE *fp；
if((fp=("c:\\config.sys","r"))==NULL);
```

其意义是检查文件指针**fp**所指向的文件存在否，如果存在，则打开C盘目录下的config.sys文件，只允许进行“读”操作，并使**fp**指向该文件。

例9.2 文件的打开2

```
FILE *fphzk；
If((fphzk=("ccbp.dat","rb"))==NULL);
```

其意义是先检查文件ccbp.dat文件是否存在，如果存在，则打开当前目录下的文件ccbp.dat，这是一个二进制文件，只允许按二进制方式进行读操作。

9.3.1 文件的打开（fopen函数）：

使用文件的方式共有12种，符号和意义如下：

文件使用方式	意义
“rt”	只读打开一个文本文件，只允许读数据
“wt”	只写打开或建立一个文本文件，只允许写数据
“at”	追加打开一个文本文件，并在文件末尾写数据
“rb”	只读打开一个二进制文件，只允许读数据
“wb”	只写打开或建立一个二进制文件，只允许写数据
“ab”	追加打开一个二进制文件，并在文件末尾写数据
“rt+”	读写打开一个文本文件，允许读和写
“wt+”	读写打开或建立一个文本文件，允许读写
“at+”	读写打开一个文本文件，允许读，或在文件末追加数据
“rb+”	读写打开一个二进制文件，允许读和写
“wb+”	读写打开或建立一个二进制文件，允许读和写
“ab+”	读写打开一个二进制文件，允许读，或在文件末追加数据

9.3.1 文件的打开（fopen函数）：

1） 文件使用方式由r，w，a，t，b，+ 六个字符拼成，各字符的含义是：

r(read): 读	w(write): 写
a(append): 追加	t(text):文本文件，可省略不写
+: 读和写	b(banary): 二进制文件

2）凡用“r”打开一个文件时，该文件必须已经存在，且只能从该文件读出。

3）用“w”打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件，若打开的文件已经存在，则将该文件删去，重建一个新文件。

9.3.1 文件的打开（fopen函数）：

4) 若要向一个已存在的文件追加新的信息，只能用“a”方式打开文件。但此时该文件必须是存在的，否则将会出错。

5) 在打开一个文件时，如果出错，fopen将返回一个空指针值NULL。在程序中可以用这一信息来判别是否完成打开文件的工作，并作相应的处理。

例9.3 文件打开错误

```
if((fp=fopen("c:\\hzk16","rb")==NULL)
{
        printf("\nerror on open c:\\hzk16 file!");
        getch();
        exit(1);
}
```

如果返回的指针为空，表示不能打开C盘根目录下的hzk16文件，则给出错误提示信息。敲键后执行exit(1)退出程序。

6) 标准输入文件(键盘)，标准输出文件(显示器)，标准出错输出(出错信息)是由系统打开的，可直接使用。

9.3.2 文件关闭函数（fclose函数）

文件一旦使用完毕，应用关闭文件函数把文件关闭，以避免文件的数据丢失等错误。

fclose函数调用的一般形式是：

```
fclose(文件指针);
```

正常完成关闭文件操作时，fclose函数返回值为0。如返回非零值则表示有错误发生。

9.3.3 数据块读写函数fread和fwtrite

C语言提供了用于整块数据的读写函数。可用来读写一组数据，如一个数组元素，一个结构变量的值等。

函数fread和fwrite的原形分别如下：

```
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
```

```
size_t fwrite(const void *ptr, size_t size, size_t n, FILE *stream);
```

对于函数fread而言，ptr是存放所读入数据的内存区域的指针，而对于函数fwrite而言，prt是写入到那个文件的信息的指针。变量n的值确定将读/写多少项，而每项的长度是由size决定的，size的类型为size_t，一般代表无符号整数。形参stream是指针变量，是指向原先打开的文件。

9.3.3 数据块读写函数fread和fwtrite

函数fread和fwrite都有返回值。函数fread返回读入的项数，如果出错或者达到文件的尾部，则返回值可能会小于n；函数fwrite返回写入的项数，如果出错，则该值将等于n。

例如：

```
read(fa,4,5,fp);
```

其意义是从fp所指的文件中，每次读4个字节(一个实数)送入实数组fa中，连续读5次，即读5个实数到fa中。

9.3.4 格式化读写函数fscanf和fprintf

fscanf函数，fprintf函数与前面使用的scanf和printf 函数的功能相似，都是格式化读写函数。两者的区别在于fscanf函数和fprintf函数的读写对象不是键盘和显示器，而是磁盘文件。

函数fscanf和fprintf的原型分别如下：

```
int fscanf(FILE *fp,const char *format[,address,...]);
```

```
int fprintf(FILE *fp,const char *format[,address,...]);
```

变元fp是函数fopen（）返回的文件指针，而函数fprintf（）和fscanf（）是把I/O操作导向指明的文件。这两个函数的调用格式为：

```
fscanf(文件指针,格式字符串，输入列表);
```

```
fprintf(文件指针,格式字符串，输出列表);
```

9.3.5 字符读写函数fgetc和fputc

字符读写函数是以字符(字节)为单位的读写函数。 每次可从文件读出或向文件写入一个字符。

● 读字符函数fgetc

fgetc函数的功能是从指定的文件中读一个字符，函数调用的形式为：

```
字符变量=fgetc(文件指针);
```

例如：

```
ch=fgetc(fp);
```

其意义是从打开的文件fp中读取一个字符并送入ch中。

9.3.5 字符读写函数fgetc和fputc

1) 在fgetc函数调用中，读取的文件必须是以读或读写方式打开的。

2) 读取字符的结果也可以不向字符变量赋值，

例如：

```
fgetc(fp);
```

但是读出的字符不能保存。

3) 在文件内部有一个位置指针。用来指向文件的当前读写字节。在文件打开时，该指针总是指向文件的第一个字节。使用fgetc 函数后，该位置指针将向后移动一个字节。 因此可连续多次使用fgetc函数，读取多个字符。应注意文件指针和文件内部的位置指针不是一回事。文件指针是指向整个文件的，须在程序中定义说明，只要不重新赋值，文件指针的值是不变的。文件内部的位置指针用以指示文件内部的当前读写位置，每读写一次，该指针均向后移动，它不需在程序中定义说明，而是由系统自动设置的。

9.3.5 字符读写函数fgetc和fputc

● 写字符函数fputc

fputc函数的功能是把一个字符写入指定的文件中，函数调用的形式为：

```
fputc(字符量，文件指针);
```

其中，待写入的字符量可以是字符常量或变量，例如：

```
fputc('a',fp);
```

其意义是把字符a写入fp所指向的文件中。

1) 被写入的文件可以用写、读写、追加方式打开，用写或读写方式打开一个已存在的文件时将清除原有的文件内容，写入字符从文件首开始。如需保留原有文件内容，希望写入的字符以文件末开始存放，必须以追加方式打开文件。被写入的文件若不存在，则创建该文件。

2) 每写入一个字符，文件内部位置指针向后移动一个字节。

3) fputc函数有一个返回值，如写入成功则返回写入的字符，否则返回一个**EOF**。可用此来判断写入是否成功。

9.3.6 字符串读写函数fgets和fputs

● 读字符串函数fgets

函数的功能是从指定的文件中读一个字符串到字符数组中，函数调用的形式为：

```
fgets(字符数组名,n,文件指针);
```

其中的n是一个正整数。表示从文件中读出的字符串不超过 n-1个字符。在读入的最后一个字符后加上串结束标志'\0'。

例如：

```
fgets(str,n,fp);
```

的意义是从fp所指的文件中读出n-1个字符送入字符数组str中。

1) 在读出n-1个字符之前，如遇到了换行符或EOF，则读出结束。

2) fgets函数也有返回值，其返回值是字符数组的首地址。

9.3.6 字符串读写函数fgets和fputs

● 写字符串函数fputs

fputs函数的功能是向指定的文件写入一个字符串，其调用形式为：

```
fputs(字符串,文件指针);
```

其中字符串可以是字符串常量，也可以是字符数组名，或指针变量，例如：

```
fputs(“abcd“,fp);
```

其意义是把字符串“abcd”写入fp所指的文件之中。

9.3.7 文件定位

前面介绍的对文件的读写方式都是顺序读写，即读写文件只能从头开始，顺序读写各个数据。但在实际问题中常要求只读写文件中某一指定的部分。为了解决这个问题可移动文件内部的位置指针到需要读写的位置，再进行读写，这种读写称为随机读写。

实现随机读写的关键是要按要求移动位置指针，这称为文件的定位。

移动文件内部位置指针的函数主要有两个，

即 **rewind** 函数和**fseek**函数。

文件定位

● 文件首位置定位rewind函数

rewind函数前面已多次使用过，其调用形式为：

```
rewind(文件指针);
```

功能: 把文件内部的位置指针移到文件首。

● 文件任意位置定位fseek函数

fseek函数用来移动文件内部位置指针，其调用形式为：

```
fseek(文件指针,位移量,起始点);
```

其中：“文件指针”指向被移动的文件。

“位移量”表示移动的字节数，要求位移量是long型数据，以便在文件长度大于64KB 时不会出错。当用常量表示位移量时，要求加后缀“L”。

9.3.7 文件定位

“起始点”表示从何处开始计算位移量，规定的起始点有三种：文件首，当前位置和文件尾。

起始点	表示符号	数字表示
文件首	SEEK_SET	0
当前位置	SEEK_CUR	1
文件末尾	SEEK_END	2

例如：

```
fseek(fp,100L,0);
```

其意义是把位置指针移到离文件首100个字节处。

还要说明的是fseek函数一般用于二进制文件。在文本文件中由于要进行转换，故往往计算的位置会出现错误。

在移动位置指针之后，即可用前面介绍的任一种读写函数进行读写。由于一般是读写一个数据据块，因此常用fread和fwrite函数。

9.3.8 文件检测函数

C语言中常用的文件检测函数有以下几个：

- 文件结束检测函数feof
- 读写文件错误检测函数ferror
- 文件出错标志和文件结束标志置零函数clearerr

文件结束检测函数feof函数

调用格式：

feof(文件指针);

功能：判断文件是否处于文件结束位置，如文件结束，则返回值为1，否则为0。

9.3.8 文件检测函数

- **读写文件出错检测函数**

ferror函数调用格式：

ferror(文件指针);

功能：检查文件在用各种输入输出函数进行读写时是否出错。如ferror返回值为0表示未出错，否则表示有错。

- **文件出错标志和文件结束标志置0函数**

clearerr函数调用格式：

clearerr(文件指针);

功能：本函数用于清除出错标志和文件结束标志，使它们为0值。

9.4 文件函数应用综合举例

例9.4文件函数应用综合应用

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct stu
{
   char name[10];
   int num;
   int age;
   char addr[15];
};
main()
{
   char ch;
   int i;

   FILE *fp;
   Struct stu boya[2],boyb[2],*pp,*qq;
            pp=boya;
   qq=boyb;

   if((fp=fopen("stu_list.txt","wb+"))==NULL)
   {
      printf("不能打开文件，任意键退出!");
      getch();
      exit(1);
   }
   printf("\ninput data\n");
   for(i=0;i<2;i++,pp++)
   scanf("%s%d%d%s",pp->name,&pp->num,&pp->age,pp->addr);
   pp=boya;
   for(i=0;i<2;i++,pp++)
   {
      fprintf(fp,"%s %d %d %s\n",pp->name,pp->num,pp->age,pp-> addr);
   }
   fclose(fp);

   if((fp=fopen("stu_list.txt","r"))==NULL)
   {
      printf("不能打开文件，任意键退出");
      getch();
      exit(1);
   }
   rewind(fp);
   for(i=0;i<2;i++,qq++)
   {
      fscanf(fp,"%s %d %d %s\n",qq->name,&qq->num,&qq->age,qq->addr);
   }
   printf("\n\nname\tnumber age addr\n");
   qq=boyb;
   for(i=0;i<2;i++,qq++)
   {
   printf("%s\t%5d %7d %s\n",qq->name,\
               qq->num, qq->age,qq->addr);
   }
   fclose(fp);
}
```

9.5 C语言图形程序设计基本概念

图形化界面是软件的趋势，图形化界面具有界面友好、交互性强的特点，但图形界面的开发相对来说会麻烦一些，因此难度也就大一些，一般来说，图形界面的开发，经常与屏幕的分辨率有关。

在屏幕上进行绘制图形，一般要按以下几个步骤执行：

（1）把屏幕设置为图形模式；

（2）选择背景与显示实体的颜色；

（3）计算图形显示坐标；

（4）调用绘图语句绘制实体

C语言图形程序设计基本概念

显示器的工作方式

图形方式

一像素作为屏幕上的最小单元，以屏幕的左上角为原点,位置为(0,0)。

文本方式

屏幕上显示的最小单位是字符。不同的文本方式对应相应的行数、列数和颜色。

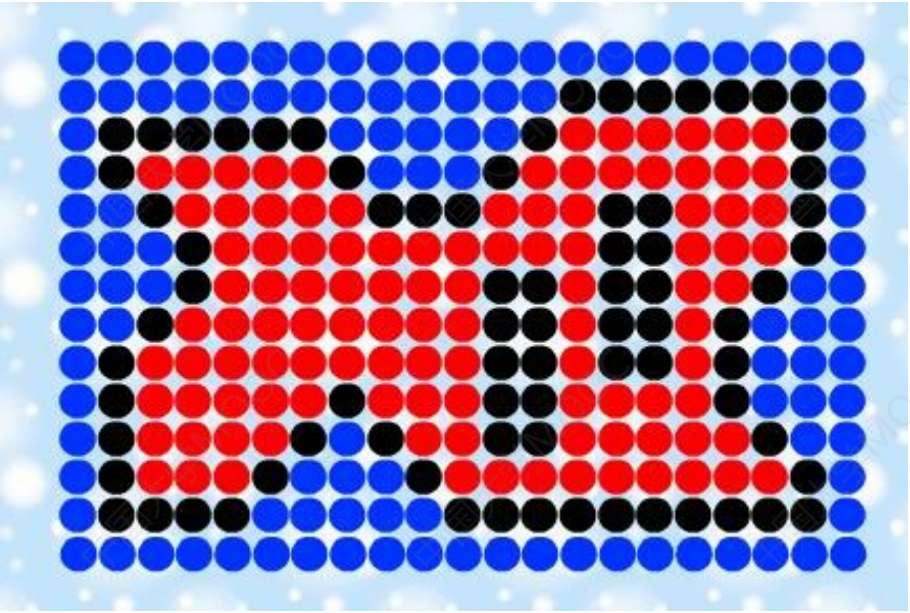
9.5 C语言图形程序设计基本概念

在屏幕上显示图形的方式称为图形模式。在图形模式下,屏幕是由像素点组成的,像素点的多少决定了屏幕的分辨率。分辨率越高,显示图形越细致,质量越好。

如右图所示

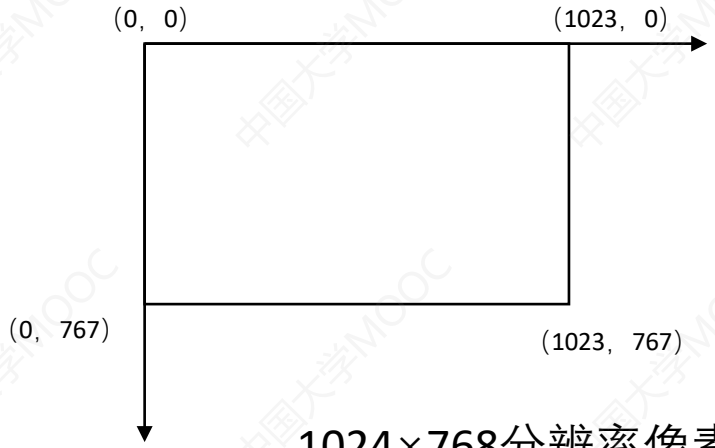
1.屏幕由许多像素点组成;

2.图形的显示效果取决于分辨率的高低;



C语言图形程序设计基本概念

在图形模式下，屏幕上每个像素的显示位置用点　坐标来描述的。点坐标系是以屏幕左上角为坐标原点（0，0），水平方向为X轴，自左向右；垂直方向为Y轴，自上向下。如下图所示：



1024×768分辨率像素坐标

9.6 C语言中的图形函数

C提供了十分丰富的图形库函数，共有70多个，所有图形函数的原型均在头文件“graphics.h”中定义。所以，在任何使用图形函数的程序中，都要求其头部包含文件graphics.h，即在程序文件的开头写上文件包含命令：

```
#include <graphics.h>
```

同时将集成开发环境option/Linker中的graphics.lib选项设置为”on”。

9.6.1 初始化图形系统函数initgraph

● 初始化图形系统函数

void initgraph（int* graphdriver,int* graphmode,char*pathtodriver）

- **功能**：将图形驱动程序装入内存，使屏幕显示适配器设置为图形模式，即图形系统初始化；
- **参数说明**：1. **graphdriver**是一个整型值,用来指定要装入的图形驱动程序，指向存有显卡类型编号（整数）的整型变量;该值在头文件**graphics.h**中以枚举的形式定义；

2. **graphmode**是一个整型值,用来设置图形显示模式。图形显示模式决定了显示的分辨率和显示的颜色的多少等；

3. **pathtodriver**是一个字符串，用来指明图形驱动程序文件所在的路径，图形驱动程序文件以bgi为扩展名。

9.6.1 初始化图形系统函数initgraph

```
enum graphics_drivers
{
DETECT,    //自动检测
CGA,MCGA,EGA,EGA64,EGAMONO,IBM8
514,  //1-6
HERCMONO,ATT400,VGA,PC3270,//7-10
CURRENT_DRAIVER=-1
};
```

```
enum graphics_modes
{
…   //省略CGA，VGA等显示模式
VGALO = 0,   //640×200 16color 4pages
VGAMED = 1,// 640×350 16color 2pages
VGAHI = 2,  // 640×480 16color 1pages
PC3270HI = 0, // 720×350 1pages
IBM8514LO = 0, // 640×480 256color
IBM8514HI = 1, // 1024×768 256color
};
```

✓1.若驱动程序就在用户当前目录下，则path可为空字符串，否则应给出具体路径名。以BC为例，一般情况下, 在BC安装目录（假设暗转在C盘），则该路径为: C:\BC31,若写在参数中则为“C:\\BC31”；

✓2.前两个参数实际上是整型指针，调用时应加上地址运算符“&”；

9.6.1 初始化图形系统函数initgraph

例9.5 已知显示器类型的图形系统初始化

```
#include"graphics.h"
   main()
  {
      int gdriver,gmode;
      gdriver=CGA;    //设置显示模式CGA
     gmode=CGAC0;  //选用CGA图形模式

initgraph(&gdriver,&gmode,“c:\\bc31\\bgi
");
                          //初始化图形系统
     bar3d(10,20,50,80,0,0); //画一条形
图
    getch();                  //等待按一键结束
    closegraph();  //关闭图形系统，回到
文本模式
    }
```

例9.6 未知显示器类型的图形系统初始化

```
  #include"graphics.h"
   main()
  {
      int gdriver,gmode;
     detectgraph(&gdriver,&gmode); //测试结果
存放于 gdriver,gmode中
if(gdriver<0)
 {
    printf("there is not graphics displayer\n");
   exit(1);
 }  //无图形显示模式时，显示信息，停止程序

printf("detect graphics driver is # %d,mode is #
%d \n",gdriver,gmode);       //显示硬件测试结果
     getch();                    //等待按一键结束
     initgraph(&gdriver,&gmode,“D:\\TC");
                               //初始化图形系统
      bar3d(10,20,50,80,0,0); //画一条形图
      getch();                  //等待按一键结束
      closegraph();  //关闭图形系统，回文本模式
     }
```

9.6.1 初始化图形系统函数initgraph

例9.7 自动初始化图形系统

```
#include"graphics.h"
   main()
  {
      int gdriver,gmode;
      gdriver=DETECT;
     initgraph(&gdriver,&gmode,“ ");
                              //初始化图形系统
      bar3d(10,20,50,80,0,0); //画一条形图
     getch();                 //等待按一键结束
     closegraph(); //关闭图形系统，回到文本模式
     }
```

关闭图形系统函数closegraph

关闭图形系统函数

void closegraph（void）

- **功能**：关闭图形系统，释放图形系统所占的内存空间，并返回调用initgraph函数之前的显示模式（通常是文本模式）；

  另外，将系统从图形方式切换到文本方式的函数还有：

  void far restorecrtmode（void）

9.6.3 设置画笔当前颜色及屏幕背景色

- **设置当前画笔颜色函数** **void setcolor（int color）**
  - **功能**：用来设置当前画笔的颜色，将影响待画出的直线、圆、矩形等线条的颜色；
- **设置屏幕背景色函数** **void setbkcolor（int color）**
  - **功能**：用来设置屏幕背景的颜色；

在头文件graphics.h中，以枚举的形式定义了颜色：

```
enum
COLORS{BLACK,BLUE,GREEN,CYAN,RED,MAGENTA,BROWN,LIGHTGRAY,DARKGRAY,
LIGHTBLUE,LIGHTGREEN,LIGHTCYAN,LIGHTRED,LIGHTMAGENTA,YELLOW,WHITE};
```

例如：设置当前画笔颜色为绿色，可执行语句：

setcolor(GREEN) 或者 setcolor(2)

设置屏幕背景颜色为蓝色，可执行语句：

setbkcolor(BLUE) 或者 setbkcolor(1)

9.6.4 画点及获取屏幕点颜色

- **画点函数** **void putpixel（int x, int y, int color）**
  - **功能**：向屏幕指定坐标处画一个给定颜色的点；

  例如： putpixel(100,80,RED)

- **获取屏幕点颜色函数** **unsigned getpixel（int x, int y）**
  - **功能**：用来指出屏幕上某一点的颜色是什么；

  例如： c = getpixel(100,80)

9.6.5 设置线型及画直线

- **设置线型函数** void setlinestyle（int linestyle, unsigned user_pattern，int thickness）

  - **功能**：用于设置画笔的当前线型及宽度；
  - **参数说明**：参数linestyle即线型的枚举常量如下：

枚举常量名	整数值	线型
SOLID_LINE	0	实线
DOTTED_LINE	1	虚线
CENTER_LINE	2	中心线
DASHE_LINE	3	破折号
USERBIT_LINE	4	用户自定义线型

如果linestyle的参数取值为USERBIT_LINE，则参数user_pattern的值就是用户自定义的线型。线型以二进制（共16位）的形式存放于参数user_pattern中，其中1点画出，0不画出。当linestyle取非USERBIT_LINE，user_pattern取0值。

参数thickness只有两个值：

NORM_WIDTH(整数值为1)和THICK_WIDTH(数值为3)

NORM_WIDTH画细线， THICK_WIDTH表示画粗线

9.6.5 设置线型及画直线

● 画直线的相关函数

void line（int x1, int y1， int x2, int y2）

void lineto（int x, int y）

void **moveto**（int x, int y）

➢ **功能**：line函数在屏幕上的任意两点之间画一条直线段；

如：在坐标（10，20）、（150，50）的两点画一条直线

line(10，20，150，50 ）；

函数moveto用于移动画笔的当前位置到指定坐标位置，但移动过程中不画线。

函数lineto配合函数moveto使用，则可在屏幕上画出连续的折线。如画一个以点（50，0），（100，0）及（60，80）为顶点的三角形：

```
moveto（50，0）；
lineto（100，50）；
lineto（60，80）；
Lineto（50，0）；
```

9.6.6 画圆、椭圆、矩形及多边形

void circle（int x, int y, int radius）
void ellips（int x, int y, int stangle,int endangle,int xradius,int yradius）
void rectangle（int left, int top, int right, int bottom）
void drawpoly（int numpoints, int* polypoints）

- **函数说明：**

1）函数circle的功能是以某点为圆心，用当前线型画一个指定半径的圆，如：
circle（100，80，30）即为以（100，80）为圆心，画一个半径30的圆；

2）函数ellips用于在屏幕上画一个椭圆。参数（x,y）是椭圆的中心坐标，stangle是起始角度，endangle结束角度，xradius和yradius分别为x轴和y轴的椭圆半径；
ellips（150，100，0，180，40，30）；

3）函数rectangle的功能是画一个矩形，参数（left，top）是左上角的坐标，参数（right，bottom）是矩形右下角的坐标。
rectangle（10，20，80，50）；

画圆、椭圆、矩形及多边形

➢ **函数说明：**

4）函数drawpoly的功能是按给定的顶点，画一条连续折线。如果所画的折线式封闭的，那么画出的便是一个多边形。参数numpoints为折线的顶点数，参数polypoints指向有顶点坐标的 一维整型数组的第一个元素，数组中顶点的坐标按（x,y）依次存放。

如画一个梯形，可执行语句如下:

```
int v[] = {50,10,100,10,120,60,30,60,50,10};
drawpoly(5,v);
```

填充图形函数

设置填充模式和填充颜色函数

void setfillpattern（char* upattern, int color）;
void setfillstyle(int pattern,int color);

- **功能**：setfillpattern是设置用户定义的填充图案及填充颜色；
- **参数说明：**参数upattern指向存有表述填充图案（8×8方块）的8个字节存储区域，参数color为填充颜色：

如：

```
char pattern[] = {0xFF,0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0xFF};
setfillpattern(pattern,BLUE);
```

9.6.7 填充图形函数

Setfillstyle的功能是设置系统预设填充图案及用户指定的填充颜色。参数pattern的取值如右表，

如果用户想使用自定义填充模式，应该使用setfillpattern函数而不是用setfillstyle函数的USER_FILL模式。

枚举常量名	整数值	填充图案
EMPTY_FILL	0	背景色填充
SOLID_FILL	1	单色填充
LINE_FILL	2	…填充
LTSLASH_FILL	3	///填充
SLASH_FILL	4	粗///填充
BKSLASH_FILL	5	粗\\\填充
LTBKSLASH_FILL	6	\\\填充
HATCH_FILL	7	淡影线填充
XHATCH_FILL	8	交叉线填充
INTERLEAVE_FILL	9	间隔线填充
WID_DOT_FILL	10	稀疏空白点填充
COLOSE_DOT_FILL	11	密集空白点填充

9.6.7 填充图形函数

- **填充指定区域函数** **void floodfill（int x,int y,int border）；**

  - **功能**：将指定边界色为border的封闭区域，用当前填充图案的填充色来填充；
  - **参数说明：**形参（x,y）为填充区域中的一点，参数border为区域的边界颜色值。如：

    用绿色填充一个圆形区域（边界红色），可执行语句：

    setcolor（GREEN）；

    circle（80，60，30）；

    floodfile（80，63，RED）；

9.6.7 填充图形函数

- **填充矩形、多边形、椭圆、圆和扇形等函数：**

```
void  bar （ int left, int top, int right, int bottom ） ;
void fillpoly(int numpoints, int* polypoints );
void fillellips(int x, int y, int xradius,int yradius);
void peislice(int x, int y, int stangle,int endangle,int radius);
void sector(int x, int y, int stangle,int endangle,int xradius,int yradius);
```

- **功能**：函数bar和函数rectangle相似，不同的是函数bar用当前线型和颜色画完矩形边框后，还会用当前填充图案和填充色填充该矩形；

函数fillpoly与函数drawpoly相似，不同的也是在画完折线后，还会用当前填充图案和填充色填充折线所围起的区域；

函数fillellips和ellips也是填充和不填充的区别；

函数pieslice和函数sector的区别： pieslice是填充圆形扇形， sector是填充椭圆形扇形；

9.7 图形方式下的文本常见操作函数

在图形方式下主要绘制图形，但还要输出文本。为了有效地进行图形操作，在图形方式下开设“视口”（也称“窗口”或“视见区”）也是极重要的，图像方式下的文本操作函数是只在图形区进行文本输入/输出的函数，BC++3.1只提供了对图形进行字符串输出的函数，但输出字符的字型等是可控的。

9.7.1 视口操作函数

视口操作函数是指用于图形输出的屏幕矩形区域，初始化图形系统时，视口默认为整个屏幕。视口函数有：

void far setviewport（int left, int top, int right, int bottom ,int clip）；

void far getviewport(struct viewporttype* viewport);

void far clearviewport(void);

- **功能**：函数setviewport 用于设置视口在屏幕中的位置和视口区的大小，参数（left，top）是视口左上角的坐标（屏幕坐标系中的坐标），参数（right，bottom）是视口右下角的坐标（屏幕坐标系中的坐标）。参数clip用来确定当绘制的图形越过视口边界时，是否对其裁剪。0-裁剪；非0-不裁剪；

函数getviewport返回当前视口区的信息，结果存入viewport中，viewport结构定义：

```
Struct viewporttype{
        int left, top, right, bottom;
        int clip;};
```

函数clearviewport清除当前视口；

9.7.2 图形方式下的文字输出

- **相关函数：**

```
settextstyle(int font,int direction,int charsize);
outtext(char* textstring);
outtextxy(int x,int y, char* textstring);
```

- **功能**：函数settextstyle用于指定输出字符串的字体、大小和方向。
- **参数说明**：函数中font确定所用字体的类型，bc31中包含了几种英文字体，如表，其中8×8点阵字体放在图形驱动程序中，而其他的矢量字体都是以CHR为扩展名，存放在系统初始化函数initgraph指定的目录中或存放在当前目录下。

参数direction确定输入文字的方向，它只有两个值：HORIZ_DIR(整数值为0)和VERT_DIR(整数值为1)。

参数charsize确定文字的大小，取值范围是0~10.

9.7.2 图形方式下的文字输出

枚举常量名	整数值	字体类型
DEFAULT_FONT	0	8×8点阵字体
TRIPLEX_FONT	1	三重矢量字体
SMALL_FONT	2	小号矢量字体
SANS_SERIF_FONT	3	无衬线矢量字体
GOTHIC_FONT	4	哥特矢量字体

文本操作函数outtext和outtextxy主要功能是在屏幕上输出字符串，区别在于outtext在当前位置输出， outtextxy是在指定位置输出。

9.7.3 屏幕图形的保存和恢复

- **相关函数：**

```
getimage(int left,int top,int right,int bottom,void* bitmap);
imagesize(int left,int top,int right,int bottom,void);
putimage(int left,int top, void* bitmap,int op);
```

- **功能及参数说明**：

函数**getimage**可以把屏幕某一矩形区域的图像保存到指定的内存中。参数（left,top）为矩形左上角坐标，参数(right,bottom)为矩形右下角坐标，参数bitmap为用于保存图像的内存区地址。

函数imagsize用于计算保存图形所需的存储空间大小；

函数putimage是恢复函数getimage保存的图像，并将其显示到屏幕上。参数（left,top）为屏幕目标矩形区域的左上角坐标，参数bitmap为由函数getimage保存的图像存放区域的地址，参数op为恢复方式，如下：

9.7.3 屏幕图形的保存和恢复

枚举常量名	整数值	恢复方式
COPY_PUT	0	原样拷贝到屏幕上
XOR_PUT	1	与屏幕上的点“异或”后写入
OR_PUT	2	与屏幕上的点“或”后写入
AND_PUT	3	与屏幕上的点“与”后写入
NOT_PUT	4	原图像取反后写到屏幕

例：要将屏幕某一区域图像复制到屏幕另一位置：

```
void * buffer;                    //定义指针用于存放图像存储区地址;
unsigned s;                       //用于存放存储区大小
s = imagesize(20,30,50,65); //计算所需内存的大小
buffer = malloc(s);               //动态分配所需的内存
getimage(20,30,50,65,buffer)j; //保存矩形区图像到buffer指向的内存区;
putimage(100,100,buffer,COPY_PUT);//以复制方式将图像恢复到指定位置
free(buffer);
```

9.8 综合举例

例9.8　画出一个搜索引擎的界面，用一个二维数组来存储鼠标的画图信息（存放在cursor中）。

```
void main()
{
        //存储画鼠标信息的二维数组
        int Cursor[16][10] = {
    {1,0,0,0,0,0,0,0,0,0},
    {1,1,0,0,0,0,0,0,0,0},
    {1,1,0,0,0,0,0,0,0,0},
    {1,1,1,0,0,0,0,0,0,0},
    {1,1,1,1,0,0,0,0,0,0},
    {1,1,1,1,1,0,0,0,0,0},
    {1,1,1,1,1,1,0,0,0,0},
    {1,1,1,1,1,1,1,0,0,0},
    {1,1,1,1,1,1,1,1,0,0},
    {1,1,1,1,1,1,1,1,1,0},
    {1,1,1,1,1,1,1,1,1,1},
    {1,1,1,1,1,0,0,0,0,0},
    {1,0,0,0,0,1,1,0,0,0},
    {0,0,0,0,0,1,1,0,0,0},
    {0,0,0,0,0,0,1,0,0,0},
    };
```

```
int driver = VGA;
int mode = VGAHI,I,j,x,y;
initgraph(&driver,&mode,"d:\\bc31\\bgi");
cleardevice();
setbkcolor(LIGHTGRAY);
setviewport(20,100,570,450,1);
x = 500;
y = 200;
for( i =0;i<16;i++)
    for(j=0;j<10;j++)
   {
      if(Cursor[i][j] !=0)
         putpixel(x+j,y+i,WHITE);
   }
setfillstyle(1,WHITE);
bar(100,100,400,120);
setcolor(BLUE);
circle(250,50,30);
circle(250,10,8);
```

Bc31安装路径

综合举例

```
 circle(220,15,5);
 circle(280,15,5);
 setfillstyle(1,BLUE);
 floodfill(250,50,BLUE);
floodfill(250,10,BLUE);
floodfill(220,15,BLUE);
floodfill(280,15,BLUE);
settextstyle(1,0,3);
setcolor(RED);
outtextxy(150,35,”searching engine”);
setfillstyle(1,LGHTGRAY);
bar(420,100,460,120);
setcolor(WHITE);
line(419,99,461,99);
line(419,99, 419,121);
setcolor(RED);
settextstyle(SMALL_FONT,HORIZ_DIR,4);
outtextxy(423,102,”search”);
outtextxy(125,130,”help”);
outtextxy(225,130,”more”);
outtextxy(325,130,”about”);
    setcolor(DARKGRAY);
    line(102,101,102,119);
    getch();
    closegraph();
}
```

本章小结

- C语言中文件的处理过程通常要经历“打开文件-〉文件的读/写-〉关闭文件”等3个步骤，C标准函数库为此都配备有相应的操作函数。
- 按文件内的数据组织形式，可把文件分为文本流文件和二进制流文件，编写程序时应注意这两种流文件在完成上述3个存取操作步骤的不同之处。
- 文件可按只读、只写、读/写、追加4种操作方式打开，同时还必须指定文件的类型是二进制文件还是文本文件。
- 学习C语言文件读/写函数，重点需要掌握**fopen**，**fclose**，**fread**，**fwrite**，**fscanf**和**fprintf**函数。
- 掌握C语言开发各种图形，任何图形都可以归结为点、线、矩形、圆形等基本图形处理上。