C语言的基本输入与输出 函数

1.1.1 格式化输入输出函数

Turbo C2.0 标准库提供了两个控制台格式化输入、 输出函数printf () 和scanf(), 这两个函数可以在标准输入输出设备上以各种不同的格式读写数据。printf()函数用来向标准输出设备(屏幕)写数据; scanf() 函数用来从标准输入设备(键盘)上读数据。

一、printf()函数

printf()函数是格式化输出函数, 一般用于向标准输出设备按规定格式输出信息。在编写程序时经常会用到此函数。printf()函数的调用格式为:printf("<格式化字符串>", <参量表>);其中格式化字符串包括两部分内容: 一部分是正常字符, 这些字符将按原样输出，例如printf("风雨兼程\n"); 另一部分是格式化规定字符, 以"%"开始, 后跟一个或几个规定字符,用来确定输出内容格式。参量表是需要输出的一系列参数, 其个数必须与格式化字符串所说明的输出参数个数一样多, 各参数之间用","分开, 且顺序一一对应, 否则将会出现意想不到的错误。

1. 格式化规定符

Turbo C2.0提供的格式化规定符如下:

符号	作用
%d	十进制有符号整数
%u	十进制无符号整数
%f	浮点数
%s	字符串
%c	单个字符
%p	指针的值
%e	指数形式的浮点数
%x, %X	无符号以十六进制 表示的整数
%0	无符号以八进制表示的整数
%g	自动选择合适的表示法

说明:

(1). 可以在"%"和字母之间插进数字表示最大场宽。例如: %9.2f 表示输出场宽为9的浮点数, 其中小数位为2, 整数位为6,小数点占一位, 不够9位右对齐。%8s 表示输出8个字符的字符串, 不够8个字符右对齐。

(2). 可以控制输出左对齐或右对齐, 即在"%"和字母之间加入一个"-" 号可说明输出为左对齐, 否则为右对齐。例如: %-7d 表示输出7位整数左对齐。如果字符串的长度或整型数位数超过说明的场宽, 将按其实际长度输出。但对浮点数, 若整数部分位数超过了说明的整数位宽度, 将按实际整数位输出;若小数部分位数超过了说明的小数位宽度, 则按说明的宽度以四舍五入输出。另外, 若想在输出值前加一些0, 就应在场宽项前加个0。例如: d 表示在输出一个小于4位的数值时, 将在前面补0使其总宽度为4位。如果用浮点数表示字符或整型量的输出格式, 小数点后的数字代表最大宽度,小数点前的数字代表最小宽度。例如: %6.9s 表示显示一个长度不小于6且不大于9的字符串。若大于9, 则第9个字符以后的内容将被删除。

(3). 可以在"%"和字母之间加小写字母l, 表示输出的是长型数。例如: %ld 表示输出long整数%lf 表示

输出double浮点数。
2. 一些特殊规定字符

字符	作用
\n	换行
\f	清屏并换页
\r	回车
\t	Tab符
\xhh	表示一个ASCII码用16进表示, 其中hh是1到2个16进制数

编制下面的程序，以加深对TurboC2.0数据的了解

例1

```
#include<stdio.h>

#include< string    .h>

intmain()

{

    char c,s[20], *p;

    inta=1234, *i;

    floatf=3.141592653589;

    doublex=0.12345678987654321;

    p="How do you do";

    strcpy(s,"Hello, Comrade");

    *i=12;

    c='\x41';

    printf("a=%d\n", a);    /*结果输出十进制整数a=1234*/

    printf("a=%6d\n", a);   /*结果输出6位十进制数a= 1234*/

    printf("a=%06d\n", a);  /*结果输出6位十进制数a=001234*/

     printf("a=%2d\n", a);   /*a超过2位, 按实际值输出a=1234*/
```

```
    printf("*i=%4d\n", *i); /*输出4位十进制整数*i= 12*/
    printf("*i=%-4d\n",*i); /*输出左对齐4位十进制整数*i=12*/
    printf("i=%p\n", i);    /*输出地址i=06E4*/
    printf("f=%f\n", f);    /*输出浮点数f=3.141593*/
    printf("f=6.4f\n", f);  /*输出6位其中小数点后4位的浮点数f=3.1416*/
    printf("x=%lf\n", x);   /*输出长浮点数x=0.123457*/
    printf("x=%18.16lf\n", x);/*输出18位其中小数点后16位的长浮点数x=0.1234567898765432*/

    printf("c=%c\n", c);    /*输出字符c=A*/
    printf("c=%x\n", c);    /*输出字符的ASCII码值c=41*/
    printf("s[]=%s\n", s);  /*输出数组字符串s[]=Hello, Comrade*/
    printf("s[]=%6.9s\n", s);/*输出最多9个字符的字符串s[]=Hello, Co*/
    printf("s=%p\n", s);    /*输出数组字符串首字符地址s=FFBE*/
    printf("*p=%s\n", p);   /* 输出指针字符串p=How do you do*/
    printf("p=%p\n", p);    /*输出指针的值p=0194*/
    getch();
     retunr0;
}
```

二、scanf()函数

scanf()函数是格式化输入函数, 它从标准输入设备(键盘) 读取输入的信息。其调用格式为:scanf(“<格式化字符串>”, <地址表>);格式化字符串包括以下三类不同的字符;

1. 格式化说明符: 格式化说明符与printf()函数中的格式说明符基本相同。
2. 空白字符: 空白字符会使scanf()函数在读操作中略去输入中的一个或多个空白字符。
3. 非空白字符: 一个非空白字符会使scanf()函数在读入时剔除掉与这个非空白字符相同的字符。地址表是需要读入的所有变量的地址, 而不是变量本身。这与printf()函数完全不同, 要特别注意。各个变量的地址之间同“,”分开。

例1:

```
int i,j;
scanf(“%d, %d”, &i, &j);
```

上例中的scanf()函数先读一个整型数, 然后把接着输入的逗号剔除掉, 最后读入另一个整型数。如果“,”这一特定字符没有找到, scanf()函数就终止。若参数之间的分隔符为空格, 则参数之间必须输入一

个或多个空格。
说明:
(1). 对于字符串数组或字符串指针变量, 由于数组名和指针变量名本身就是地址, 因此使用scanf()函数时, 不需要在它们前面加上”&”操作符。
例2

```
main()
{
	char *p, str[20];
	scanf(“%s”, p);
	scanf(“%s”, str);
	printf(“%s\n”, p);
	printf(“%s\n”, str);
}
```

(2). 可以在格式化字符串中的“%”各格式化规定符之间加入一个整数, 表示任何读操作中的最大位数。如例2中若规定只能输入10字符给字符串指针p, 则第一条scanf() 函数语句变为scanf(“s”, p);程序运行时一旦输入字符个数大于10, p就不再继续读入, 而后面的一个读入函数即scanf(“%s”, str)就会从第11个字符开始读入。实际使用scanf()函数时存在一个问题, 下面举例进行说明:当使用多个scanf()函数连续给多个字符变量输入时, 例如:

```
main()
{
	char c1, c2;
	scanf  (“%c”, &c1);
	scanf(“%c”, &c2);
	printf(“c1 is %c, c2 is %c”, c2, c2);
}
```

运行该程序, 输入一个字符A后回车 (要完成输入必须回车), 在执行scanf(“%c”, &c1)时, 给变量c1赋值”A”, 但回车符仍然留在缓冲区内, 执行输入语句scanf(”%c”, &c2)时, 变量c2输出的是一空行, 如果输入AB后回车, 那么输出结果为: c1 is A, c2 is B。要解决以上问题, 可以在输入函数前加入清除函数fflush()。修改以上程序变成:

```
#include<stdio.h>
main()
{
	char c1, c2;
	scanf(“%c”, &c1);
	fflush(stdin);
	scanf(“%c”, &c2);
	printf(“c1 is %c, c2 is %c”, c1, c2);
}
```

1.1.2 非格式化输入输出函数
非格式化输入输出函数可以由上面讲述的标准格式化输入输出函数代替, 但这些函数编译后代码少, 相对占用内存也小, 从而提高了速度, 同时使用也比较方便。下面分别进行介绍。

一、puts()和gets()函数

1. puts()函数puts()函数用来向标准输出设备(屏幕)写字符串并换行, 其调用格式为:puts(s);其中s为字符串变量(字符串数组名或字符串指针)。puts()函数的作用与语printf(”%s\n”, s)相同。

例3:

```
main()
{
	char s[20], *f;
	strcpy(s, “Hello! Turbo C2.0″);
	f=“Thank you”;
	puts(s);
	puts(f);
}
```

说明:

(1). puts()函数只能输出字符串, 不能输出数值或进行格式变换。

(2). 可以将字符串直接写入puts()函数中。如:puts(“Hello, Turbo C2.0”);

2. gets()函数

gets()函数用来从标准输入设备(键盘)读取字符串直到回车结束, 但回车符不属于这个字符串。其调用格式为:gets(s);其中s为字符串变量(字符串数组名或字符串指针)。gets(s)函数与scanf(“%s”, &s)相似, 但不完全相同, 使用scanf(“%s”,&s)函数输入字符串时存在一个问题, 就是如果输入了空格会认为输入字符串结束,空格后的字符将作为下一个输入项处理, 但gets() 函数将接收输入的整个字符串直到回车为止。

例4

```
main()
{
	char s[20], *f;
	printf(“What’s your name?\n”);
	gets(s);
	puts(s);
	puts(“How old are you?”);
	gets(f);
	puts(f);
}
```

说明:(1). puts()函数只能输出字符串, 不能输出数值或进行格式变换。(2). 可以将字符串直接写入puts()函数中。如:puts("Hello, Turbo C2.0");

2. gets()函数

gets()函数用来从标准输入设备(键盘)读取字符串直到回车结束, 但回车符不属于这个字符串。其调用格式为:gets(s);其中s为字符串变量(字符串数组名或字符串指针)。 gets(s)函数与scanf("%s", &s)相似, 但不完全相同, 使用scanf("%s",&s)函数输入字符串时存在一个问题, 就是如果输入了空格会认为

输入字符串结束,空格后的字符将作为下一个输入项处理, 但**gets() 函数将接收输入的整个字符串直到回车为止。**

例5

```
main()
{
        char s[20], *f;
        printf("What's your name?\n");
        gets(s);                    /*等待输入字符串直到回车结束*/
        puts(s);                    /*将输入的字符串输出*/
        puts("How old are you?");
        gets(f);
        puts(f);
 }
```

说明:(1). gets(s)函数中的变量s为一字符串。如果为单个字符, 编译连接不会有错误, 但运行后会出现"Null pointer asignmemt"的错误。

二、putchar()、getch()、getche()和getchar()函数

1. putchar()函数

putchar()函数是向标准输出设备输出一个字符, 其调用格式为:putchar(ch);其中ch为一个字符变量或常量。putchar()函数的作用等同于printf("%c", ch);

例6:

```
#include<stdio.h>
main()
   {
        char c:             /*定义字符变量*/
        c='B';              /*给字符变量赋值*/
        putchar(c);         /*输出该字符*/
        putchar('\x42');    /*输出字母B*/
```

```
    putchar(0x42);      /*直接用ASCII码值输出字母B*/
}
```

从本例中的连续四个字符输出函数语句可以分清字符变量的不同赋值方法。

2. getch()、getche()和getchar()函数

(1) getch()和getche()函数 这两个函数都是从键盘上读入一个字符。其调用格式为:getch(); getche(); 两者的区别是:getch()函数不将读入的字符回显在显示屏幕上, 而getche()函数却将读入的字符回显到显示屏幕上。

例7:

```
#include<stdio.h>
main()
{
    char c, ch;
    c=getch();     /*从键盘上读入一个字符不回显送给字符变量c*/
    putchar(c);    /*输出该字符*/
    ch=getche();   /*从键盘上带回显的读入一个字符送给字符变量ch*/
    putchar(ch);
}
```

利用回显和不回显的特点, 这两个函数经常用于交互输入的过程中完成暂停等功能。

例8:

```
#include<stdio.h>
main()
{
    char c, s[20];
    printf("Name:");
    gets(s);
    printf("Press  any     key to confinue...");
    getch(); /*等待输入任一键*/
```

}

(2) getchar()函数

getchar()函数也是从键盘上读入一个字符, 并带回显。它与前面两个函数的区别在于: getchar()函数等待输入直到按回车才结束, 回车前的所有输入字符都会逐个显示在屏幕上。但只有第一个字符作为函数的返回值。getchar()函数的调用格式为:getchar();

例9:

```
#include<stdio.h>

main()

{

        char c;

        c=getchar();   /*从键盘读入字符直到回车结束*/

        putchar(c);    /*显示输入的第一个字符*/

        getch();       /*等待按任一健*/

 }
```

1.2 文件的输入输出函数

键盘、显示器、打印机、磁盘驱动器等逻辑设备, 其输入输出都可以通过文件管理的方法来完成。而在编程时使用最多的要算是磁盘文件, 因此本节主要以磁盘文件为主, 详细介绍Turbo C2.0提供的文件操作函数, 当然这些对文件的操作函数也适合于非磁盘文件的情况另外, TurboC2.0提供了两类关于文件的函数。一类称做标准文件函数也称缓冲型文件函数, 这是ANSI标准定义的函数; 另一类叫非标准文件函数, 也称非缓冲型文件函数。这类函数最早公用于UNIX操作系统, 但现在MS-DOS3.0 以上版本的操作系统也可以使用。下面分别进行介绍。

1.2.1 标准文件函数

标准文件函数主要包括文件的打开、关闭、读和写等函数。不象BASIC 、FORTRAN语方有顺序文件和随机文件之分， 在打开时就应按不同的方式确定。Turbo C2.0并不区分这两种文件, 但提供了两组函数, 即顺序读写函数和随机读写函数。

一、文件的打开和关闭

任何一个文件在使用之前和使用之后, 必须要进行打开和关闭, 这是因为操作系统对于同时打开的文件数目是有限制的, DOS操作系统中,可以在DEVICE.SYS中定义允许同时打开的文件数n(用files=n定义)。其中n 为可同时打开的文件数, 一般n<=20。因此在使用文件前应打开文件, 才可对其中的信息进行存取。用完之后需要关闭, 否则将会出现一些意想不到的错误。Turbo C2.0提供了打开和关闭文件的函数。

1. fopen()函数

fopen函数用于打开文件, 其调用格式为:FILE *fopen(char *filename, *type);在介绍这个函数之;前, 先了解一下下面的知识。

(1) 流(stream)和文件(file)

流和文件 在Turbo C2.0中是有区别的, TurboC2.0 为编程者和被访问的设备之间提供了一层抽象的东西, 称之为"流", 而将具体的实际设备叫做文件。流是一个逻辑设备, 具有相同的行为。因此, 用来进行磁盘文件写的函数也同样可以用来进行打印机的写入。在Turbo C2.0中有两种性质的流: 文字流(textstream)和二进制(binary stream)。对磁盘来说就是文本文件和二进制文件。本软件 为了便于让读者易理解Turbo C2.0语言而没有对流和文件作特别区分。

(2) 文件指针FILE

实际上FILE是一个新的数据类型。它是TurboC2.0的基本数据类型的集合,称之为结构指针。有关结构的概念将在第四节中详细介绍, 这里只要将FILE理解为一个包括了文件管理有关信息的数据结构 , 即在打开文件时必须先定义一个文件指针。

(3) 以后介绍的函数调用格式将直接写出形式参数的数据类型和函数返回值的数据类型。例如: 上面打开文件的函数, 返回一个文件指针, 其中形式参数有两个, 均为字符型变量(字符串数组或字符串指针)。本软件不再对函数的调用格式作详细说明。现在再来看打开文件函数的用法。fopen()函数中第一个形式参数表示文件名, 可以包含路径和文件名两部分。如:

"B:TEST.DAT"

"C:\\TC\\TEST.DAT"

如果将路径写成"C:\TC\TEST.DAT"是不正确的, 这一点要特别注意。第二个形式参数表示打开文件的类型。关于文件类型的规定参见下表。

表 文件操作类型

字符	含义
"r"	打开文字文件只读
"w"	创建文字文件只写
"a"	增补, 如果文件不存在则创建一个
"r+"	打开一个文字文件读/写
"w+"	创建一个文字文件读/写

"a+"　　打开或创建一个文件增补

"b"　　二进制文件(可以和上面每一项合用)

"t"　　文这文件(默认项)

如果要打开一个CCDOS子目录中, 文件名为CLIB的二进制文件, 可写成:fopen("c:\\ccdos\\clib", "rb");

如果成功的打开一个文件,fopen()函数返回文件指针， 否则返回空指针(NULL)。由此可判断文件打开是否成功。

2. fclose()函数

fclose()函数用来关闭一个由fopen()函数打开的文件 , 其调用格式为:int fclose(FILE *stream); 该函数返回一个整型数。当文件关闭成功时, 返回0， 否则返回一个非零值。可以根据函数的返回值判断文件是否关闭成功。

例10:

```
#iclude<stdio.h>

main()

{

    FILE *fp;              /*定义一个文件指针*/

    int i;

    fp=fopen("CLIB", "rb"); /*打开当前目录名为CLIB的文件只读*/

    if(fp==NULL)           /*判断文件是否打开成功*/

      puts("File openerror");/*提示打开不成功*/

    i=fclose(fp);          /*关闭打开的文件*/

    if(i==0)               /*判断文件是否关闭成功*/

      printf("O,K");       /*提示关闭成功*/

    else

      puts("File close error");/*提示关闭不成功*/

}
```

二、有关文件操作的函数

1. 文件的顺序写函数

fprintf()、fputs()和fputc()函数函数fprintf()、fputs()和fputc()均为文件的顺序写操作函数,其调用格式如下:int fprintf(FILE *stream, char *format, <variable-list>);int fputs(char *string, FILE *steam);int fputc(int ch, FILE *steam);

上述三个函数的返回值均为整型量。fprintf() 函数的返回值为实际写入文件中的字罕个数(字节数)。如果写错误, 则返回一个负数, fputs()函数返回0时表明将string指针所指的字符串写入文件中的操作成功, 返回非0时, 表明写操作失败。fputc()函数返回一个向文件所写字符的值, 此时写操作成功, 否则返回EOF(文件结束结束其值为-1, 在stdio.h中定义)表示写操作错误。fprintf( ) 函数中格式化的规定与printf( ) 函数相同， 所不同的只是fprintf()函数是向文件中写入。而printf()是向屏幕输出。

下面介绍一个例子, 运行后产后一个test.dat的文件。

例11:

```
#include<stdio.h>

 main()

{

        char *s="That's goodnews"); /*定义字符串指针并初始化*/

        int i=617;                  /*定义整型变量并初始化*/

        FILE *fp;                   /*定义文件指针*/

        fp=fopne("test.dat", "w");    /*建立一个文字文件只写*/

        fputs("Your score of TOEFLis", fp);/*向所建文件写入一串字符*/

        fputc(':', fp);             /*向所建文件写冒号:*/

        fprintf(fp, "%d\n", i);      /*向所建文件写一整型数*/

        fprintf(fp, "%s", s);        /*向所建文件写一字符串*/

        fclose(fp);                 /*关闭文件*/

}
```

用DOS的TYPE命令显示TEST.DAT的内容如下所示:屏幕显示

```
    Your score of TOEFL is: 617

    That's good news
```

2. 文件的顺序读操作函数

fscanf()、fgets()和fgetc()函数函数fscanf()、fgets()和fgetc()均为文件的顺序读操作函数,其调用格式如下:int fscanf(FILE *stream, char *format, <address -list>);char fgets(char *string, int n, FILE *steam);int fgetc(FILE *steam);

fscanf()函数的用法与scanf()函数相似， 只是它是从文件中读到信息。fscanf()函数的返回值为EOF(即-1), 表明读错误, 否则读数据成功。fgets()函数从文件中读取至多n-1个字符(n用来指定字符数), 并把它们放入string指向的字符串中, 在读入之后自动向字符串末尾加一个空字符, 读成功返回string指针,失败返回一个空指针。fgetc()函数返回文件当前位置的一个字符, 读错误时返回EOF。

下面程序读取例11产生的test.dat文件 , 并将读出的结果显示在屏幕上。

例12

```
#include<stdio.h>

main()

{

        char *s, m[20];

        int i;

        FILE *fp;

        fp=fopen("test.dat", "r");

         fgets(s, 24, fp);

        printf("%s", s);

        fscanf(fp, "%d", &i);

        printf("%d", i);

        putchar(fgetc(fp));

        fgets(m, 17, fp);

        puts(m);

        fclose(fp);

        getch();

}
```

运行后屏幕显示:

Your score of TOEFL is: 617

That's good news

如果将上例中fscanf(fp,"%d", &i)改为fscanf(fp, "%s", m), 再将其后的输出语句改为printf("%s",m), 则可得出同样的结果。由此可见Turbo C2. 0中只要是读文字文件, 则不论是字符还是数字都将按其ASCII值处理。另外还要说明的一点就是fscanf()函数读到白符时, 便自动结束, 在使用时要特别注意。

3. 文件的随机读写

有时用户想直接读取文件中间某处的信息, 若用文件的顺序读写必须从文件头开始直到要求的文件位置再读, 这显然不方便。Turbo C2.0提供了一组文件的随机读写函数, 即可以将文件位置指针定位在所要求读写的地方直接读写。文件的随机读写函数如下:

int fseek (FILE *stream, long offset, int fromwhere);

int fread(void *buf, int size, int count, FILE *stream);

int fwrite(void *buf, int size, int count, FILE *stream);

long ftell(FILE *stream);

fseek()函数的作用是将文件的位置指针设置到从fromwhere开始的第offset字节的位置上, 其中fromwhere是下列几个宏定义之一:

符号常数	数值	含义
SEEK_SET	0	从文件开头
SEEK_CUR	1	从文件指针的现行位置
SEEK_END	2	从文件末尾

offset是指文件位置指针从指定开始位置(fromwhere指出的位置)跳过的字节数。它是一个长整型量, 以支持大于64K字节的文件。fseek()函数一般用于对二进制文件进行操作。当fseek()函数返回0时表明操作成功, 返回非0表示失败。

下面程序从二进制文件test_b.dat中读取第8个字节。

例13:

#include<stdio.h>

```
main()
{
        FILE *fp;
        if((fp=fopen("test_b.dat", "rb"))==NULL)
          {
             printf("Can't openfile");
             exit(1);
          }
        fseek(fp, 8. 1, SEEK_SET);
        fgetc(fp);
        fclose(fp);
}
```

fread()函数是从文件中读count个字段, 每个字段长度为size个字节, 并把它们存放到buf指针所指的缓冲器中。fwrite()函数是把buf指针所指的缓冲器中, 长度为size个字节的count个字段写到stream指向的文件中去。随着读和写字节数的增大, 文件位置指示器也增大, 读多少个字节, 文件位置指示器相应也跳过多少个字节。读写完毕函数返回所读和所写的字段个数。ftell()函数返回文件位置指示器的当前值, 这个值是指示器从文件头开始算起的字节数, 返回的数为长整型数, 当返回-1时, 表明出现错误。