第 3 章　顺序结构

3.1　学前思考

从程序流程的角度来看，程序可以分为三种基本结构，即顺序结构、分支结构、循环结构。这三种基本结构可以组成各种复杂的程序。

本项目使用到了基本的输入输出语句，采用了顺序结构的方式来实现。具体实现 35 选 7 彩票模拟系统在进入之前的欢迎界面。如图 3.1 所示。

图 3.1　欢迎界面

3.2　知识储备

结构化程序的三种结构：顺序结构、选择结构、循环结构。本章介绍这些基本语句及其在顺序结构中的应用，使读者对 C 程序有一个初步的认识，为后面各章的学习打下基础。

顺序结构的程序即：程序由上而下一步一步地执行。主要由赋值语句、输入输出操作组成，在 C 语言中没有输入输出语句，输入输出的操作是由函数完成的。

3.2.1 赋值语句

赋值语句是由赋值表达式再加上分号构成的表达式语句。

其一般形式为：

变量＝表达式；

赋值语句的功能和特点都与赋值表达式相同。它是程序中使用最多的语句之一。

在赋值语句的使用中需要注意以下几点：

(1)由于在赋值符“＝”右边的表达式也可以是一个赋值表达式，

因此，形式变量＝(变量＝表达式)；是成立的，从而形成嵌套的情形。

其展开之后的一般形式为：

```
变量＝变量＝…＝表达式；
```

例如：

```
a＝b＝c＝d＝e＝5；
```

按照赋值运算符的右接合性，因此实际上等效于：

```
e＝5；
d＝e；
c＝d；
b＝c；
a＝b；
```

(2)注意在变量说明中给变量赋初值和赋值语句的区别。

给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其他同类变量之间仍必须用逗号间隔，而赋值语句则必须用分号结尾。

例如：

```
int a＝5,b,c；
```

(3)在变量说明中，不允许连续给多个变量赋初值。

如下述说明是错误的：

```
int a＝b＝c＝5
```

必须写为

```
int a＝5,b＝5,c＝5；
```

而赋值语句允许连续赋值。

(4)注意赋值表达式和赋值语句的区别。

赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

下述语句是合法的：

```
if((x＝y＋5)＞0) z＝x；
```

语句的功能是，若表达式 x＝y＋5 大于 0 则 z＝x。

下述语句是非法的：

```
if((x＝y＋5；)＞0) z＝x；
```

因为 x＝y＋5；是语句，不能出现在表达式中。

3.2.2　数据的输入输出

(1)所谓输入输出是以计算机为主体而言的。

(2)本章介绍的是向标准输出设备显示器输出数据的语句。

(3)在 C 语言中,所有的数据输入/输出都是由库函数完成的。因此都是函数语句。

(4)在使用 C 语言库函数时,要用预编译命令。

#include

将有关“头文件”包括到源文件中。

使用标准输入输出库函数时要用到“stdio.h”文件,因此源文件开头应有以下预编译命令:

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

或

```
#include"stdio.h"
```

stdio 是 standard input &outupt 的意思。

(5)考虑到 printf 和 scanf 函数使用频繁,系统允许在使用这两个函数时可不加 #include<stdio.h>或#include"stdio.h"

3.2.3　字符数据的输入输出

1. putchar 函数(字符输出函数)

putchar 函数是字符输出函数,其功能是在显示器上输出单个字符。

其一般形式为:putchar(字符变量)

例如:

```
putchar('A');    (输出大写字母 A)
putchar(x);      (输出字符变量 x 的值)
putchar('\101');(也是输出字符 A)
putchar('\n'); (换行)
```

对控制字符则执行控制功能,不在屏幕上显示。

使用本函数前必须要用文件包含命令:

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

或

```
#include"stdio.h"
```

【例 3.1】 输出单个字符。

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    char a='B',b='o',c='k';
    putchar(a);putchar(b);putchar(b);putchar(c);putchar('\t');
    putchar(a);
```

```
    putchar(b);
    putchar('\n');
    putchar(b);
    putchar(c);
    system("pause");
    return 0;
}
```

该程序的运行结果如图 3.2 所示。

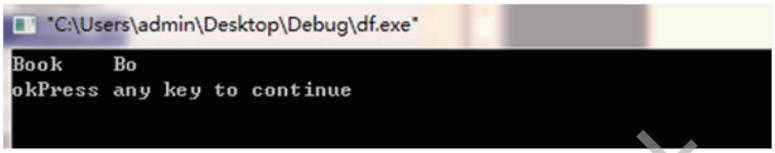


图 3.2 例 3.1 运行结果

2. getchar 函数(键盘输入函数)

getchar 函数的功能是从键盘上输入一个字符。

其一般形式为:

```
getchar();
```

通常把输入的字符赋予一个字符变量,构成赋值语句,如:

```
char c;
c=getchar();
```

【例 3.2】 输入单个字符。

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    char c;
    printf("input a character\n");
    c=getchar();
    putchar(c);
    system("pause");
    return 0;
}
```

该程序的运行结果如图 3.3 所示。

图 3.3 例 3.2 运行结果

使用 getchar 函数还应注意几个问题:

(1)getchar 函数只能接受单个字符,输入数字也按字符处理。输入多于一个字符时,只

接收第一个字符。

(2)使用本函数前必须包含文件"stdio.h"。

(3)在 TC 屏幕下运行含本函数程序时,将退出 TC 屏幕进入用户屏幕等待用户输入。输入完毕再返回 TC 屏幕。

(4)程序最后两行可用下面两行的任意一行代替:

```
putchar(getchar());
printf("%c",getchar());
```

3.2.4　输入与输出的格式

1. printf 函数(格式输出函数)

printf 函数称为格式输出函数,其关键字最末一个字母 f 即为"格式"(format)之意。其功能是按用户指定的格式,把指定的数据显示到显示器屏幕上。在前面的例题中我们已多次使用过这个函数。

printf 函数是一个标准库函数,它的函数原型在头文件"stdio.h"中。但作为一个特例,不要求在使用 printf 函数之前必须包含 stdio.h 文件。

printf 函数调用的一般形式为:

printf("格式控制字符串",输出列表)

其中格式控制字符串用于指定输出格式。格式控制串可由格式字符串和非格式字符串两种组成。格式字符串是以%开头的字符串,在%后面跟有各种格式字符,以说明输出数据的类型、形式、长度、小数位数等。例如:

"%d"表示按十进制整型输出;

"%ld"表示按十进制长整型输出;

"%c"表示按字符型输出等。

非格式字符串在输出时原样照印,在显示中起提示作用。

【例 3.3】　请写出该程序的运行结果。

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    int a=88,b=89;
    printf("%d %d\n",a,b);
    printf("%d,%d\n",a,b);
    printf("%c,%c\n",a,b);
    printf("a=%d,b=%d",a,b);
    system("pause");
    return 0;
}
```

该程序的运行结果如图 3.4 所示。

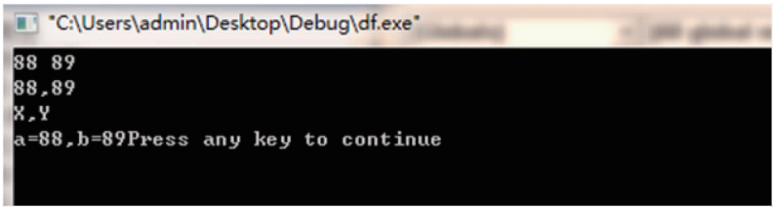


图 3.4 例 3.3 运行结果

本例中四次输出了 a,b 的值,但由于格式控制串不同,输出的结果也不相同。第四行的输出语句格式控制串中,两格式串%d 之间加了一个空格(非格式字符),所以输出的 a,b 值之间有一个空格。第五行的 printf 语句格式控制串中加入的是非格式字符逗号,因此输出的 a,b 值之间加了一个逗号。第六行的格式串要求按字符型输出 a,b 值。第七行中为了提示输出结果又增加了非格式字符串。

格式字符串的一般形式为:

[标志][输出最小宽度][.精度][长度]类型

其中方括号[]中的项为可选项。

各项的意义介绍如下:

(1)类型:类型字符用以表示输出数据的类型,其格式符和意义如表 3.1 所示:

表 3.1 输出数据类型格式符

格式字符	意 义
d	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 0)
x,X	以十六进制形式输出无符号整数(不输出前缀 Ox)
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e,E	以指数形式输出单、双精度实数
g,G	以%f 或%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

(2)标志:标志字符为-、+、#、空格四种,其意义如表 3.2 所示:

表 3.2 标识符意义

标 志	意 义
-	结果左对齐,右边填空格
+	输出符号(正号或负号)
空格	输出值为正时冠以空格,为负时冠以负号
#	对 c,s,d,u 类无影响;对 o 类,在输出时加前缀 o;对 x 类,在输出时加前缀 0x;对 e,g,f 类当结果有小数时才给出小数点

(3)输出最小宽度:用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度,则按实际位数输出,若实际位数少于定义的宽度则补以空格或 0。

(4)精度:精度格式符以“.”开头,后跟十进制整数。本项的意义是:如果输出数字,则表示小数的位数;如果输出的是字符,则表示输出字符的个数;若实际位数大于所定义的精度数,则截去超过的部分。

(5)长度:长度格式符为 h,l 两种,h 表示按短整型量输出,l 表示按长整型量输出。

2. scanf 函数(格式输入函数)

scanf 函数称为格式输入函数,即按用户指定的格式从键盘上把数据输入到指定的变量之中。

scanf 函数是一个标准库函数,它的函数原型在头文件“stdio.h”中,与 printf 函数相同,C 语言也允许在使用 scanf 函数之前不必包含 stdio.h 文件。

scanf 函数的一般形式为:

```
scanf("格式控制字符串",变量地址列表);
```

其中,格式控制字符串的作用与 printf 函数相同,但不能显示非格式字符串,也就是不能显示提示字符串。地址表列中给出各变量的地址。地址是由地址运算符“&”后跟变量名组成的。

例如:

```
&a, &b
```

分别表示变量 a 和变量 b 的地址。

这个地址就是编译系统在内存中给 a,b 变量分配的地址。在 C 语言中,使用了地址这个概念,这是与其他语言不同的。应该把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是 C 编译系统分配的,用户不必关心具体的地址是多少。

变量的地址和变量值的关系如下:

在赋值表达式中给变量赋值,如:

```
a=567
```

则 a 为变量名,567 是变量的值,&a 是变量 a 的地址。

但在赋值号左边是变量名,不能写地址,而 scanf 函数在本质上也是给变量赋值,但要求写变量的地址,如 &a。这两者在形式上是不同的。& 是一个取地址运算符,&a 是一个表达式,其功能是求变量的地址。

【例 3.4】　请写出该程序的运行结果,输入 10 12 20。

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    int a,b,c;
    printf("input a,b,c\n");
    scanf("%d%d%d",&a,&b,&c);
    printf("a=%d,b=%d,c=%d",a,b,c);
    system("pause");
    return 0;
}
```

该程序的运行结果如图 3.5 所示。

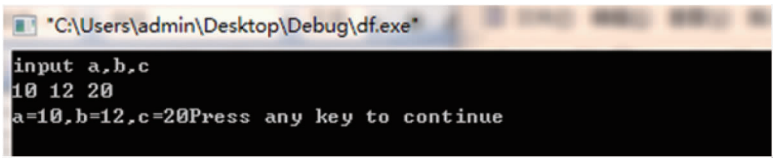


图 3.5　例 3.4 运行结果

在本例中，由于 scanf 函数本身不能显示提示串，故先用 printf 语句在屏幕上输出提示，提醒用户输入 a、b、c 的值。执行 scanf 语句时，则退出 TC 屏幕进入用户屏幕等待用户输入。用户输入 10　12　20 后按下回车键，此时，系统又将返回显示屏幕。在 scanf 语句的格式串中由于没有非格式字符在“%d%d%d”之间作输入时的间隔，因此在输入时要用一个以上的空格或回车键作为每两个输入数之间的间隔。如：

10 12 20

或

10

12

20

格式字符串的一般形式为：

%[*][输入数据宽度][长度]类型

其中有方括号[]的项为任选项。各项的意义如下：

(1)类型：表示输入数据的类型，其格式符和意义如表 3.3 所示。

表 3.3　　输入格式

格式	字符意义
d	输入十进制整数
o	输入八进制整数
x	输入十六进制整数
u	输入无符号十进制整数
f 或 e	输入实型数(用小数形式或指数形式)
c	输入单个字符
s	输入字符串

(2)“*”符：用以表示该输入项，读入后不赋予相应的变量，即跳过该输入值。

如：scanf("%d %*d %d",&a,&b);

当输入为：1　2　3 时，把 1 赋予 a，2 被跳过，3 赋予 b。

(3)宽度：用十进制整数指定输入的宽度(即字符数)。

例如：

```
scanf("%5d",&a);
```

输入：12345678

只把 12345 赋予变量 a，其余部分被截去。

又如：

```
scanf("%4d%4d",&a,&b);
```

输入:12345678

只把 1234 赋予 a,而把 5678 赋予 b。

(4)长度:长度格式符为 l 和 h,l 表示输入长整型数据(如%ld)和双精度浮点数(如%lf)。h 表示输入短整型数据。

使用 scanf 函数还必须注意以下几点:

①scanf 函数中没有精度控制,如:scanf("%5.2f",&a);是非法的。不能企图用此语句输入小数为 2 位的实数。

②scanf 中要求给出变量地址,如给出变量名则会出错。如 scanf("%d",a);是非法的,应改为 scanf("%d",&a);才是合法的。

③在输入多个数值数据时,若格式控制串中没有非格式字符作输入数据之间的间隔则可用空格,Tab 或回车作间隔。C 编译在碰到空格、Tab、回车或非法数据(如对"%d"输入"12A"时,A 即为非法数据)时即认为该数据结束。

④在输入字符数据时,若格式控制串中无非格式字符,则认为所有输入的字符均为有效字符。

例如:

```
scanf("%c%c%c",&a,&b,&c);
```

输入为:

```
d  e  f
```

则把'd'赋予 a,'f'赋予 b,'e'赋予 c。

当输入为:

```
def
```

才能把'd'赋于 a,'e'赋予 b,'f'赋予 c。

如果在格式控制中加入空格作为间隔,如:

```
scanf ("%c %c %c",&a,&b,&c);
```

则输入时各数据之间可加空格。

3.3　C 刀小试

1. 实验目的

(1)理解 C 语言程序的三种基本结构。

(2)掌握变量定义和基本数据处理。

(3)掌握输入输出函数的功能、格式及使用方法、设计简单的顺序结构程序。

2. 实验内容

按要求编写程序或将程序填充完整。

(1)求判别式的值

在花括号{}的 scanf 与 printf 语句之间填写适当的语句,将程序补充完整实现下述功能,并上机运行验证。

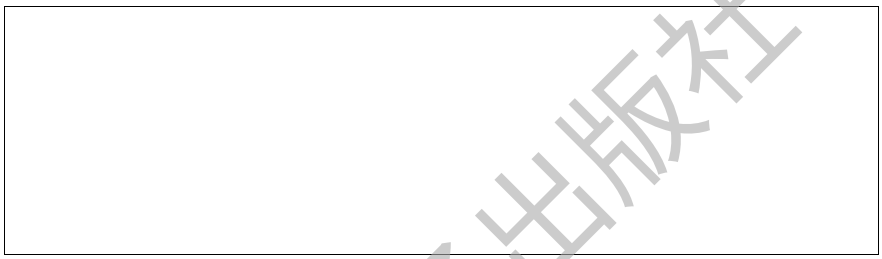
输入 a,b,c 的值,计算 s=b * b-4ac 的计算结果并输出。

例如:

输入:1.5

输出:5.3

```
/ * example3_1.c * /
# include<stdio.h>
# include<stdlib.h>
int main(int argc, char * argv[])
{
    inta, b,c,s;
    scanf(" % d % d % d", &a,&b,&c);
    / * 可根据需要填写多条语句 * /
```



```
    printf("s= % d\n", s);
system("pause");
return 0;

}
```

(2)计算求和求平均值

在花括号{}之间填写适当的语句,将程序补充完整实现下述功能,并上机运行验证。

程序的功能是从键盘上输入三个整数代表一个学生三个科目的期末考试成绩,计算出该学生的成绩总分和平均分。

例如:

输入:80,70,90

输出:sum=240.00 average=80.00

```
/ * example3_2.c * /
# include<stdio.h>
# include<stdlib.h>
int main(int argc, char * argv[])
{
    int a,b,c;
    float s,d;
    printf("Input math,english,c_program:");
    scanf(" % d, % d, % d",&a,&b,&c);
    / * 可以根据需要填入任意多条语句 * /
```

```
    printf("sum= %f\taverage= %f\n",s,d);
system("pause");
return 0;

}
```

【解答提示】

注意求和、平均值的数据类型。

(3)分解一个整数各个位数字

在花括号{}之间填写适当的语句,将程序补充完整实现下述功能,并上机运行验证。

程序的功能是从键盘上输入一个两位数的正整数,将其个位数和十位数互换后变成一个新的数,并输出这个数。

例如:

输入:89

输出:new_num=98

```
/*example3_3.c*/
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    int a,b,c,d;
    printf("Input num:");
    scanf("%d",&a);
    /*可以根据需要填入任意多条语句*/
```

```
    printf("new_num= %d\n",d);
system("pause");
return 0;

}
```

3.4 实作强化

1. 实验目的

(1)熟练掌握 putchar、getchar、printf、scanf 函数的使用方法。

(2)掌握各种类型数据的输入输出的方法,能正确使用各种格式转换符。

2. 实验内容

(1)字符的输入与输出

编程实现由键盘输入一个字符后,在屏幕上输出该字符。

键盘输入:a↙

输出:a

[参考答案]

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    char ch;
    ch=getchar();
    putchar(ch);
    system("pause");
    return 0;

}
```

(2)计算加法

编程实现由键盘输入一个加法式,输出正确的结果。(两个加数均为整数)

键盘输入:10+20↙

输出:30

[参考答案]

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    int a, b;
    scanf("%d+%d", &a,&b);
    printf("%d", a+b);
    system("pause");
    return 0;

}
```

(3)求圆面积

由键盘输入圆半径 r,请计算该圆的面积。(注:π 取 3.14159,结果保留两位小数位;另外,程序只要能对 r 在 0 到 10000 范围的情况输出正确答案即可)

键盘输入:65.2↙

输出:13355.02

[提示]结果保留两位小数,可采用 printf 函数的格式控制字符来实现。

[参考答案]

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    float area,r;
    scanf("%f",&r);
    area=3.14159*r*r;
    printf("%0.2f",area);
    system("pause");
    return 0;

}
```

(4)计算摄氏温度值

从键盘输入一个华氏温度值,要求按格式输出其对应的摄氏温度值,精确到小数点后两位。

数学公式描述为:

$$C=\frac{5}{9}(F-32)$$

键盘输入:100↙

输出:37.78

[提示]注意公式中的除为普通除法。

[参考答案]

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    float f,c;
    scanf("%f",&f);
    c=5.0/9*(f-32);
    printf("%.2f",c);
    system("pause");
    return 0;

}
```

3.5 总结分享

1. 数据的输出

printf 函数格式：printf(格式控制字符串，输出列表)；例：printf("a＝%d，b＝%f"，a，b)；

格式控制字符串的几点说明：(1)由双引号括起，两部分组成。(2)%后加格式字符的表示输出的数值类型和格式。(3)其他原样输出。

2. 数据的输入

scanf 函数格式：scanf("%[＜修饰符＞]＜格式字＞"，变量地址列表)；例：scanf("%d"，&a)；

可以指定输入数据的域宽，系统自动按域宽截取输入数据，例如：scanf("%3d"，&a)；按宽度 3 输入一个整数给变量 a。

注意：输入 long 型数据必须用%ld，输入 double 型数据必须用%lf 或%le。

在完成以上实验的基础上，写下你的收获。