# 第3章 顺序结构程序设计

**学习目的和要求**

- 辨别程序的三种控制结构及其特点，能对任何一种结构绘制流程图。
- 掌握数据的定义和表达式赋值方式，能够设计顺序结构的应用程序。
- 灵活应用C语言基本输入/输出函数的基本格式及其主要用法。
- 认识C++中简单的输入/输出控制。
- 理解C/C++语言提供的预处理命令：宏定义和文件包。

**思政目标和思政点**

在解三角形的过程中，如何利用三条边直接求面积是一个比较难的问题。中国宋代的数学家秦九韶在1247年独立提出了“三斜求积术”。“三斜求积术”与海伦公式虽然在形式上有所不同，但与海伦公式完全等价，填补了中国数学史中的一个空白，通过此案例引导学生体会数学的简洁美与中国古代数学的辉煌成就，厚植爱国主义情怀。

结构化程序设计是面向过程程序设计的基本原则。其观点是采用“自顶向下、逐步细化、模块化”的程序设计方法，任何程序都由顺序、选择、循环三种基本结构程序构造而成。C语言提供了丰富的语句，来实现程序的各种结构方式。

顺序结构是最简单、最常用的基本结构，学好顺序结构程序设计，能够为后续学习选择结构和循环结构程序设计打下坚实的基础。顺序结构的程序是自上而下顺序执行各条语句。本章介绍C语言提供的几种常用语句、预处理命令及其在顺序结构程序中的应用，读者可对C程序有一个初步的认识，为后面各章的学习打下基础。

## 3.1 C语言常见的数据处理语句

一个C语言程序结构如图3.1所示，即C程序可以由若干源程序文件组成，一个源文件可以由若干函数和预处理命令以及全局变量声明部分组成，一个函数由数据定义部分和执行语句组成。C程序对用到的所有数据都必须先定义，并且需要指定其数据类型。在C程序中，程序算法处理的对象是数据，数据是以常量或变量的形式表示的。执行部分是由语句组成的，程序的功能是由执行语句实现的。

C语言中提供了丰富的语句，可分为以下几类。

(1) 表达式语句。

(2) 复合语句。

(3) 空语句。

(4) 控制语句：C语言提供了9种控制语句，在后续章节会陆续介绍。

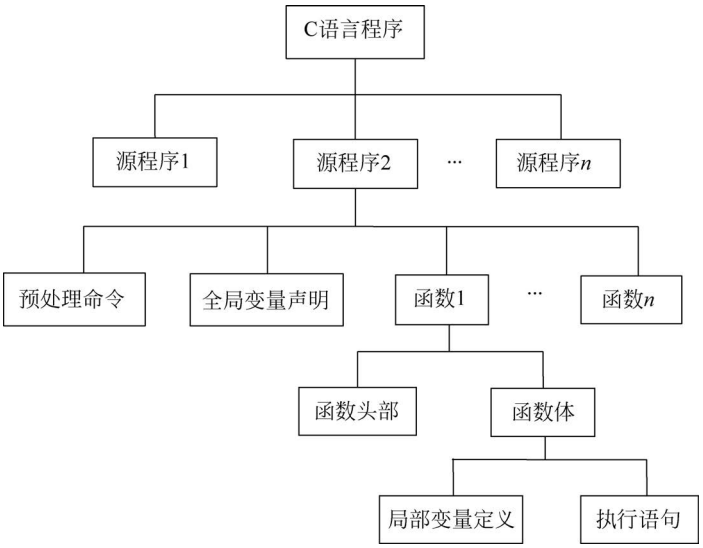


图 3.1　C 语言程序结构

### 3.1.1　数据定义和赋值语句

在 C 语言中，最常用的语句是赋值语句和输入/输出语句，其中最基本的是赋值语句。程序中的计算功能大部分是由赋值语句实现的，几乎每一个有实用价值的程序都包括赋值语句，有的程序中的大部分语句都是赋值语句。

**1. 数据定义**

这里仅以变量定义为例进行介绍。

C 语言中，要求对所有的变量，必须先定义后使用；在定义变量的同时可以给变量赋初值的操作称为变量的初始化。

变量定义的一般格式为

```
[存储类型]  数据类型  变量名 1 [,变量名 2,…];
```

例如：

```
float radius, length, area;
```

变量初始化的一般格式为

```
[存储类型]  数据类型  变量名 1[ = 初值 1] [,变量名 2[ = 初值 2] …];
```

例如：

```
float radius = 2.5,length,area;
```

**说明：**

(1) 对变量的定义，可以出现在函数中的任何行，也可以放在函数之外(详见 8.5 节)。

(2) 变量定义和变量声明的区别。

① 变量定义：用于为变量分配空间，还可以为变量初始化。程序中，变量有且仅有一次定义。

② 变量声明：用于向程序表明变量的类型和名字，不分配空间。定义也是声明(因为

定义变量时也声明了它的类型和名字),但声明不是定义。在一个程序中,变量只能定义一次,却可以声明多次(如在多文件程序中,详见 8.5 节)。

**2. 赋值语句**

赋值语句是由赋值表达式加上分号构成的表达式语句,详见 3.1.2 节。

一般格式为

```
变量 = 表达式;
```

功能:先计算赋值运算符右边表达式的值再赋给左边的变量。

几点注意:

(1) 由于在赋值运算符"="右边的表达式可以又是一个赋值表达式,因此下述形式:

```
变量 1 = (变量 2 = 表达式);
```

是成立的,从而形成嵌套的情形。

其展开后的等价形式为

```
变量 1 = 变量 2 = … = 表达式;
```

例如:

```
a = b = c = d = 5;
```

由于赋值运算符的优先级最低,其结合性为右结合,因此实际上等价于:

```
a = (b = (c = (d = 5)));
```

或者理解为

```
d = 5;
c = d;
b = c;
a = b;
```

(2) 注意逗号运算符的使用。

```
x = a = 5, b = 6,c = 7;          为逗号表达式语句,表达式的结果为 7,变量 x = 5,a = 5
x = a = 5;b = 6,c = 7;           为两条语句: 赋值语句 x = a = 5;和逗号表达式语句 b = 6,c = 7;
x = (a = 5,b = 6,c = 7);         为赋值语句,变量 x = 7,a = 5
```

(3) 在变量定义中给变量赋初值和赋值语句的区别。

给变量赋初值是变量定义的一部分,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;
```

在变量定义中,不允许连续给多个变量赋初值。如下述变量定义是错误的。

```
int a = b = c = 5;
```

必须写为

```
int a = 5,b = 5,c = 5;
```

而赋值语句允许给已定义变量连续赋值。例如:

```
a = b = c = d = 5;
```

(4) 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。

如下语句是合法的。

```
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。

而如下语句就是非法的。

```
if((x = y + 5;)> 0) z = x;
```

因为 x=y+5;是语句,不能出现在表达式中。

**【案例 3.1】** 变量的定义与赋值。

```
#include <stdio.h>
#define M 0                                    //定义符号常量(详见 3.3 节)
int n;                                         //变量 n 定义在函数外
int main()
{ int x;                                       //变量 x 定义在函数内
 int a = M,b,c = 5;                            //变量定义和赋值语句
 b = a + c;                                    //赋值语句
 printf("a = %d,b = %d,c = %d\n",a,b,c);       //输出语句(详见 3.2 节)
 printf("x = %d\n",x);
 printf("n = %d\n",n);
 printf("y = %d\n",y);
 return 0;
}
```

程序运行时出现如图 3.2 所示的错误,在源程序的第 11 行,提示变量 y 没有定义。对于 C 语言中所有的变量,必须先定义后使用。

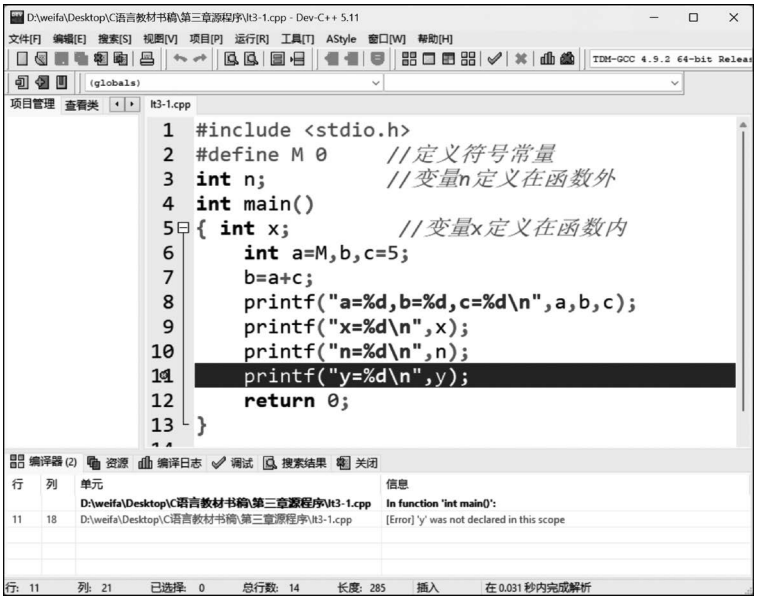


图 3.2 运行结果

### 3.1.2 表达式语句

C语言规定：一个表达式后面加上分号“;”就是一条表达式语句。

其一般格式为

```
表达式;
```

执行表达式语句就是计算表达式的值。

表达式语句有多种，包括赋值语句、关系表达式语句、逻辑表达式语句、函数调用语句等。

例如：

```
x = 6;                  赋值语句
x == 6;                 关系表达式语句
x = x == 6;             赋值语句
i -- ;                  自减 1 语句,i 值减 1;
printf("hello world");函数调用语句
```

### 3.1.3 空语句

空语句仅由一个分号“;”组成，不做任何操作。在程序中，空语句可用来作空循环体。

例如：

```
while((getchar())!= '\n')
    ;
```

本循环语句的功能是，只要从键盘输入的字符不是回车则重新输入。这里的循环体为空语句。

又如：

```
if (a % 3 == 0) ;
     i++;
```

本意是，如果a是3的倍数，则i加1。但由于if(a%3==0)后多加了分号，则if语句到此结束，程序将继续执行i++;语句，不论3是否整除a，i都将自动加1。

### 3.1.4 复合语句

复合语句是由花括号{}将多条语句组合在一起而成的，语法上应把复合语句看成一条语句，而不是多条语句。

其一般格式为

```
{
 [说明部分]
  语句部分
}
```

说明部分可以任选。语句部分可以是一条或多条语句，也可以是嵌套复合语句。

例如，要把a和b中的大数放在a中，小数放在b中，可用如下程序片段实现。

```
int a = 5,b = 6;                    //定义变量
if(a < b)                           //if 语句,当 a 小于 b 成立时,执行复合语句
```

```
{                          }
  int t;                   |
  t = a;                   |
  a = b;                   }        复合语句中包含变量 t 的声明和多条执行语句
  b = t;                   |
}                          }
printf("%d%d\n",a,b);          //函数调用语句
```

复合语句内的各语句都必须以分号结尾，在}括号后不能再加分号。

## 3.2 常用的输入/输出库函数

为了让计算机处理各种数据，首先应该把源数据输入计算机中；计算机处理结束后，再将目标数据信息以人能够识别的方式输出。C 语言没有自己的输入/输出语句，输入/输出操作是由 C 语言编译系统提供的库函数来实现的。C 语言在头文件 stdio.h 中提供了输入/输出函数：putchar()、getchar()、printf()和 scanf()。因此，在使用输入/输出函数前必须要用文件包含命令：

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

表示要使用的函数，包含在标准输入/输出头文件 stdio.h 中。

或

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

### 3.2.1 格式输入/输出函数

**1. 格式输出函数 printf()**

printf()函数称为格式输出函数，其功能是按用户指定的格式，把指定的一个或多个任意类型的数据输出到显示器屏幕上。

1）一般格式

printf()的一般格式为

```
printf("格式控制字符串"[,输出项表]);
```

其中，“格式控制字符串”用于指定输出格式。它包含格式字符串、普通字符及转义字符三种信息。输出项表是可选的。

（1）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。

（2）普通字符是需要原样输出的字符，在显示中起提示作用。例如：

```
printf("%d,%c\n",a,c);
```

其中，双引号内的“,”就是普通字符，调用函数时会原样输出。

（3）转义字符用于控制输出的样式，如常用到的\n、\t、\b 等。

输出项表中给出了各个输出项，要求格式字符串和各输出项在数量和类型上应该一一对应。如果要输出的数据不止一个，相邻两个之间用逗号分开。例如：

```
printf("a=%f,b=%5d\n", a, b);
```

2）printf()的格式字符

格式字符的一般形式为

% [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。下面分别介绍各项的意义。

（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　printf()函数格式字符及举例**

| 格式字符 | 意　义 | 举　例 | 输出结果 |
|---|---|---|---|
| d | 以十进制形式输出带符号整数(正数不输出符号) | int a=567;<br>printf("%d",a); | 567 |
| o | 以八进制形式输出无符号整数(不输出前缀 0) | int a=65;<br>printf(" %o",a); | 101 |
| x,X | 以十六进制形式输出无符号整数(不输出前缀 0x) | int a=255;<br>printf(" % x",a); | ff |
| u | 以十进制形式输出无符号整数 | int a=567;<br>printf("%u",a); | 567 |
| f | 以小数形式输出单、双精度实数，系统默认输出 6 位小数 | float a=567.789;<br>printf("%f",a); | 567.789000 |
| e,E | 以指数形式输出单、双精度实数 | float a=567.789;<br>printf("%e",a); | 5.677890e+02 |
| g,G | 以%f 或%e 中较短的输出宽度输出单、双精度实数 | float a=567.789;<br>printf("%g",a); | 567.789 |
| c | 输出单个字符 | char a=65<br>printf("%c",a); | A |
| s | 输出字符串 | printf(" %s","ABC") | ABC |

（2）标志：常用标志字符有－、＋、＃、空格 4 种，其意义如表 3.2 所示。

**表 3.2　printf()函数标志字符**

| 标志字符 | 意　义 |
|---|---|
| m | 输出数据域宽。若数据长度小于 m，左边补空格，否则按实际输出 |
| .n | 对于实数，指定小数点后的位数(四舍五入)；对于字符串，指定实际输出位数 |
| － | 输出数据左对齐，右边填空格(默认则为右对齐) |
| ＋ | 指定在有符号数的正数前面显示正号(＋) |
| 0 | 输出数据时指定左边不使用的空位自动填 0 |
| 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 0；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 |
| l 或 h | 在 d、o、x、u 格式字符前，指定输出精度为 long 型；<br>在 e、f、g 格式字符前，指定输出精度为 double 型 |

**【案例 3.2】** 输出函数举例。

```
#include <stdio.h>
int main()
{   int a = 15;
    float b = 123.1234567;
```

```
    double c = 12345678.1234567;
    char d = 'p';
    printf("a = %d, %5d, %o, %x\n",a,a,a,a);
    printf("b = %c, %8c\n",b,b);
    printf("c = %f, %lf, %5.4lf, %e\n",c,c,c,c);
    printf("d = %lf, %f, %8.4lf\n",d,d,d);
    printf("%f%%\n",1.0/5);          //输出字符'%',格式控制中连续出现两个%%
    return 0;
}
```

运行结果：

```
a = 15,    15,17,f
b = p,         p
c = 123.123459,123.123459,123.1235,1.231235e+002
d = 12345678.123457,12345678.123457,12345678.1235
0.200000%
```

**2. 格式输入函数 scanf()**

1) scanf()的一般格式

scanf()的一般格式为

```
scanf("格式控制字符串",地址表列);
```

(1) 格式控制字符串：与 printf()函数含义相同，但不能显示非格式字符串，也就是不能显示提示字符串。

(2) 地址表列：是由若干地址组成的表列，可以是变量的地址，也可以是字符串首地址。变量的地址由地址运算符“&”后跟变量名组成。

例如，scanf("%d%d", &a,&b)；中 &a、&b 分别表示变量 a 和变量 b 的地址。

至于地址表列为字符串首地址的示例，在介绍了字符数组后会用到。

2) scanf()的格式字符

格式字符的一般形式为

```
%[*][输入数据宽度][长度]类型
```

其中有方括号[]的项为任选项。各项的意义如表 3.3 所示。

**表 3.3　scanf()格式字符**

| 格式字符 | 字符意义 |
|---|---|
| d | 输入十进制整数 |
| o | 输入八进制整数(前缀 o 不输入) |
| X,x | 输入十六进制整数(前缀 OX 不输入，大小写作用相同) |
| u | 输入无符号十进制整数 |
| f | 输入实型数(用小数形式或指数形式) |
| e、E、g、G | 与格式字符 f 作用相同，e 与 f、g 可以相互替换(大小写作用相同) |
| c | 输入单个字符 |
| s | 输入字符串 |

有关 scanf()函数的使用有以下几点说明。

(1) scanf()函数可以一次输入多个数据，如 scanf("%f%f%f",&a,&b,&c)。scanf()函数的格式控制串"%f%f%f "之间没有非格式字符，输入数据时，不应连续给出，应以空

格、回车或 Tab 键来隔开这三个十进制数(键盘输入 3.0 4.0 5.0),否则系统不知道应该怎样区分这三个数。

(2) scanf()函数中可以包含普通字符,如 scanf("a=%d,b=%d,c=%d)函数的格式控制串"a=%d,b=%d,c=%d "中除了格式字符外还有其他普通字符"a=,b=,c=",则在输入数据时,要在对应位置输入与这些字符相同的字符。键盘输入数据时应为

```
a=3,b=4,c=5↙
```

(3) 当输入的数据有字符时,如 scanf("%d%c%d%c",&a,&c1,&b,&c2),应当按照以下输入方式输入。

```
3a5b↙
```

此时,系统把 3 赋值给变量 a,字符'a'赋值给变量 c1,5 赋值给变量 b,字符'b'赋值给变量 c2。在使用%c 作为格式控制字符输入数据时,空格字符和转义字符都作为有效字符处理。

**说明**:当输入的数据全部是数值数据时,在两个数字之间需要插入空格、回车或 Tab 键,以使系统能区分两个数值。当输入的数据有字符时,系统可以识别数字和字符,因此不再需要以空格来区分数字和字符。

(4) 输入数据时不能指定精度,如下面的写法是错误的。

```
scanf("%6.2f",&a);
```

(5) 可以指定输入数据所占的列数,系统将自动按指定列宽来截取所需数据。如下所示。

```
scanf("%2d%3d%2d",&a,&b,&c);
printf("%d, %d, %d\n",a,b,c);
```

若输入 123456789,则输出结果为 12,456,78。a 为 12,b 为 456,c 为 78。

(6) 赋值抑制字符“*”。

“*”表示本输入项对应的数据读入后,不赋给相应的变量(该变量由下一个格式指示符输入)。

例如:

```
scanf("%2d%*2d%3d",&num1,&num2);
printf("num1=%d,num2=%d\n",num1,num2);
```

假设输入 123456789,则系统将读取“12”并赋值给 num1;读取“34”但舍弃掉(“*”的作用);读取“567”并赋值给 num2。所以,printf()函数的输出结果为 num1=12,num2=567。

(7) 在输入数值数据时,遇到字母等非数值符号系统认为该数据结束。

**【案例 3.3】** 交换变量 a 和 b 的值。

```
#include<stdio.h>
int main()
{   int a,b;                                    //变量定义
    scanf("a=%d,b=%d",&a,&b);                   //输入数据
    a=a+b;
    b=a-b;
    a=a-b;
    printf("a=%d b=%d\n",a,b);
```

```
    return 0;
}
```

运行结果：

```
a = 3,b = 5 ↙
a = 5 b = 3
```

**【案例 3.4】** 输入函数举例。

```
#include < stdio.h >
int  main()
{int a,b,c;
 scanf(" % 2d % 3d % 2d",&a,&b,&c);
 printf(" % d,  % d,  % d \n",a,b,c);
 fflush(stdin);                    //清空输入缓冲区
 scanf(" % d",&a);
 printf(" % d\n",a);
 return 0;
}
```

运行结果：

```
123456789 ↙
12,345, 67
123 ↙
123
```

**思考**：若去掉 fflush(stdin);这条语句，结果如何？

## 3.2.2 字符输入/输出函数

**1. putchar()函数**

putchar()函数是单个字符输出函数，其功能是在显示器上输出一个字符。

其一般格式为

```
putchar(ch);
```

ch 可以是一个字符变量或整型变量或常量，也可以是一个转义字符。

例如：

```
putchar('a');        输出小写字母 a
putchar(c);          输出字符变量 c 的值
putchar('\101');     输出大写字母 A
putchar('\n');       输出转义字符换行
```

对控制字符则执行控制功能，不在屏幕上显示。

**2. getchar()函数**

getchar()函数的功能是从键盘上输入一个字符。它是一个无参函数，其一般格式为

```
getchar();
```

通常把输入的字符赋予一个字符变量，构成赋值语句，例如：

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

**注意**：getchar()函数只能接收单个字符，输入的数字也按字符处理。输入多于一个字

符时，只接收第一个字符。用 getchar()函数得到的字符可以赋给一个字符变量或整型变量，也可以不赋值给任何变量，仅作为表达式的一部分。例如：

```
putchar(getchar());          输入一个字符并输出
```

**【案例 3.5】** 字符输入/输出。

```
#include <stdio.h>
main()
{ char c1,c2,c3,c4,c5,c6;
  scanf("%c%c%c%c",&c1,&c2,&c3,&c4);
  c5 = getchar();        c6 = getchar();
  putchar(c1);         putchar(c2);
  printf("%c%c\n",c5,c6);
}
```

运行结果：

```
123↙
45678↙
1245
```

**说明**：在用键盘输入信息时，并不是在键盘上输入一个字符，该字符就立即送到计算机中的。这些字符先暂存在键盘的缓冲区中，只有按了 Enter 键才把这些字符一起输入计算机中，按先后顺序分别赋给相应的变量。上例中，字符'1'赋值给变量 c1，字符'2'赋值给变量 c2，字符'3'赋值给变量 c3，换行符赋值给变量 c4，字符'4'赋值给变量 c5，字符'6'赋值给变量 c6，字符'7'、字符'8'、字符'9'没有赋值给任何变量。

**【案例 3.6】** 编程从键盘先后输入 int 型、char 型和 float 型数据，要求每输入一个数据就显示出这个数据的类型和数据值。

```
#include <stdio.h>
int main()
{
    int a;
    char b;
    float c;
    printf("Please input an integer:");
    scanf("%d",&a);
    printf("integer:%d\n",a);
    printf("Please input a character:");
    scanf("%c",&b);
    printf("character:%c\n",b);
    printf("Please input a float number:");
    scanf("%f",&c);
    printf("float:%f\n",c);
    return 0;
}
```

运行结果：

```
Please input an integer:5↙
integer:5
Please input a character:character:

Please input a float number:3.5↙
float:3.500000
```

这个程序的运行结果显然有问题，为什么会输出这样的结果呢？

错误的原因在于第二个字符数据没有正确读入。接下来分析是如何输入数据的。程序首先提示输入第一个整型数据，执行输入整数 5，紧接着输入一个回车符，这时，系统把输入的回车符作为一个有效字符赋值给变量 b，所以当执行到语句 printf("character:%c\n",b);时，此时变量 b 是回车符，执行换行操作，运行结果的第四行是空行，执行的是转义字符'\n'。接着程序提示输入一个浮点型数据，输入数据 3.5，实际上这是第二次输入数据。

这个问题的解决方法有以下两种。

方法 1：用函数 getchar()将数据输入时存入缓冲区中的回车符读入，以避免被后面的字符型变量作为有效字符读入。

```
#include <stdio.h>
int main()
{
    int a;
    char b;
    float c;
    printf("Please input an integer:");
    scanf("%d",&a);
    printf("integer:%d\n",a);
    getchar();      //将存于缓冲区中的回车符读入,避免在后面作为有效字符读入
    printf("Please input a character:");
    scanf("%c",&b);
    printf("character:%c\n",b);
    printf("Please input a float number:");
    scanf("%f",&c);
    printf("float:%f\n",c);
    return 0;
}
```

方法 2：在%c 前面加一个空格，忽略前面数据输入时存入缓冲区中的回车符，避免被后面的字符型变量作为有效字符读入。与方法 1 相比，方法 2 更简单，程序可读性也更好。

```
#include <stdio.h>
int main()
{
    int a;
    char b;
    float c;
    printf("Please input an integer:");
    scanf("%d",&a);
    printf("integer:%d\n",a);
    printf("Please input a character:");
    scanf(" %c",&b);        //在%c 前面加一个空格,将存于缓冲区中的回车符读入
    printf("character:%c\n",b);
    printf("Please input a float number:");
    scanf("%f",&c);
    printf("float:%f\n",c);
    return 0;
}
```

以上两个程序的运行结果均为

```
Please input an integer:5
integer:5
Please input a character:a
character:a
Please input a float number:3.5
float:3.500000
```

### 3.2.3 C++的输入/输出控制

C++中数据的输入/输出是通过I/O流来实现的。所谓“流”,是指数据从一个位置流向另一个位置。流的操作包括建立流、删除流、提取(读操作/输入)、插入(写操作/输出)。流在使用前要被建立,使用后要被删除。从流中获取数据称为提取操作,向流中插入数据称为插入操作。cin和cout是C++中预定义的流类对象,cin用来处理标准输入,cout用来处理标准输出,有关流对象cin、cout和流运算符的定义等存放在C++的输入/输出流库(iostream.h)中。

**1. 预定义的插入符“<<”**

“<<”是预定义的流插入运算符,其作用是将需要输出的数据插入输出流中,默认的输出设备是显示器。一般的屏幕输出是将插入符作用在流类对象cout上。其语法形式为

```
cout <<表达式1 <<表达式2 <<…<<表达式 n;
```

在上面的输出语句中,可以串联多个插入运算符,输出多个数据项。在插入运算符后面可以写任意复杂的表达式,系统自动计算出它的值并传给插入符。例如:

```
cout <"Hello the world! "<< endl;
```

将字符串"Hello the world!"输出到屏幕上并换行。

```
cout <<"2 + 8 = "<< 2 + 8 <<"\n";
```

将字符串"2+8="和表达式2+8的计算结果10显示在屏幕上并换行。

**2. 预定义的提取符“>>”**

“>>”是预定义的流提取运算符,其作用是从默认的输入设备(一般为键盘)的输入流中提取若干字节送到计算机内存中指定的变量。一般的键盘输入是将提取符作用在流类对象cin上。其语法形式为

```
cin >>变量1 >>变量2 >>…>>变量 n;
```

在上面的输入语句中,提取符后边可以有多个,每个后面跟一个变量。例如:

```
int a,b;
cin >> a >> b;
```

先要求从键盘上输入两个整型数,两个数间用空格分隔。若键盘输入

```
5  6
```

这时变量a的值为5,变量b的值为6。

再执行语句cout << a <<"+"<< b <<"="<< a+b << endl;

输出结果为

```
5 + 6 = 11
```

### 3. 简单的 I/O 格式控制(iomanip.h)

C++中要实现输出数据的格式控制需要使用相关函数来实现,如表 3.4 所示。

**表 3.4 常用的 I/O 流类库格式操纵符**

| 操 纵 符 | 作 用 |
|---|---|
| dec | 设置数值数据的基数为 10 |
| hex | 设置数值数据的基数为 16 |
| oct | 设置数值数据的基数为 8 |
| setfill(c) | 设置填充字符 c,c 可以是字符常量或字符变量 |
| setprecision(n) | 设置浮点数的小数位数为 n(包括小数点) |
| setw(n) | 设置数据字段宽度为 n |
| endl | 输出换行符,与转义字符'\n'等价 |

**【案例 3.7】** 要求输出浮点数 3.14159,占 6 个字符宽度,小数点后保留 3 位有效数字,空格用“0”填充。

```
#include <iostream.h>          //C++中也可使用命令#include <stdio.h>
#include <math.h>
#include <iomanip.h>           //包含格式控制头文件
#define M 9                    //C/C++中使用预处理命令定义符号常量(详见 3.3.1 节)
const double PI =3.14159;      //C++中提供了符号常量声明语句
void main()
{  cout<<"M = "<<setfill('0')<<setw(6)<<setprecision(4)<<sqrt(M)<<endl;
   cout<<"PI = "<<setfill('0')<<setw(6)<<setprecision(4)<<PI<<"\n";
   cout<<"\nPI = "<<setfill('0')<<setw(6)<<setprecision(4)<<PI<<endl;
}
```

运行结果:

```
M = 000003
PI = 03.142
PI = 03.142
```

**【案例 3.8】** 求 $ax^2+bx+c=0$ 方程的根,a、b、c 由键盘输入,假设 $b^2-4ac>0$,$a\neq 0$。

```
#include <iostream.h>
#include <math.h>
#include <iomanip.h>            //包含格式控制头文件 iomanip.h
void main()
{
 float a,b,c,disc,x1,x2,p,q;
 cout<<"a = " ;cin>>a;
 cout<<"b = " ;cin>>b;
 cout<<"c = " ;cin>>c;
 //cout<<"请输入系数 a、b、c" ;
 //cin>>a>>b>>c;                //以空格、Tab 键或回车间隔输入
 disc = b*b-4*a*c;
 p= -b/(2*a);
 q = sqrt(disc)/(2*a);
 x1 = p+q; x2 = p-q;
 //printf("\nx1 = %5.2f\nx2 = %8.3f\n",x1,x2);
 cout<<"\nx1 = "<<setfill('0')<<setw(8)<<setprecision(3)<<x1;
 cout<<"\nx2 = "<<setfill('0')<<setw(8)<<setprecision(3)<<x2<<endl;
}
```

运行结果：

```
a = 1 ↙
b = 5 ↙
c = 3 ↙

x1 = 00 - 0.697
x2 = 0000 - 4.3
```

## 3.3 编译预处理

编译预处理是C语言编译系统的一个组成部分。所谓编译预处理是指在对源程序进行编译之前，先由预处理程序对源程序中的编译预处理命令进行处理（例如，程序中用#include<stdio.h>命令包含一个文件stdio.h，则在预处理时将文件stdio.h中的实际内容代替该命令），然后再将处理的结果和源程序一起进行编译，生成目标代码。合理地使用预处理命令，可以改进程序的设计环境，提高编程效率。

所有预处理命令在程序中必须以“#”号开头，每一条预处理命令单独占一行；因为它不是C语言中的语句，不以“;”结束。预处理命令都放在函数之外，而且一般都放在源文件的前面，它们称为预处理部分。

C语言提供的预处理功能主要有以下三种。

（1）文件包含。

（2）宏定义。

（3）条件编译。

### 3.3.1 文件包含

文件包含是指一个源文件可以将另一个源文件的全部内容包含进来，即将另外的文件包含到本文件之中。C语言中提供了#include命令来实现文件包含操作。文件包含命令有以下两种格式。

```
#include <文件名>
```

或

```
#include "文件名"
```

**说明：**

（1）被包含的文件一般指定为头文件（*.h），也可以为C程序等文件；文件包含允许嵌套，即在一个被包含的文件中又可以包含另一个文件。

（2）两种格式的区别：使用尖括号表示直接在系统指定的“包含文件目录”（包含文件目录由用户在设置环境时设置）中去查找被包含文件，而不在源文件目录中去查找，这称为标准方式；使用双引号则表示系统先在当前源文件目录中查找被包含文件，若未找到，再到包含文件目录中去查找。

（3）一个include指令只能指定一个被包含文件，若要包含$n$个文件，则要用$n$个指令。

（4）一般系统提供的头文件用尖括号，自定义的文件用双引号。

(5) 被包含文件与当前文件在预编译后变成同一个文件,而非两个文件。

### 3.3.2 宏定义

在C语言源程序中允许用一个标识符来表示一个字符串,称为“宏”。被定义为“宏”的标识符称为“宏名”。在编译预处理时,对程序中所有出现的“宏名”都用宏定义中的字符串去代换,这称为“宏代换”或“宏展开”。

宏定义是由源程序中的宏定义命令完成的。宏展开是由预处理程序自动完成的。

在C语言中,“宏”分为无参数的宏(简称为无参宏)和有参数的宏(简称为有参宏)两种。

**1. 无参宏定义**

一般格式为

```
#define  标识符  字符串
```

在前面介绍过的符号常量的定义就是一种无参宏定义。

例如:

```
#define PI 3.14159
```

**说明:**

(1) 宏名通常用大写字母表示,以便与变量区别。

(2) 宏用“#define”来定义,宏名和它所代表的字符串之间用空格分隔开。

例如:

```
#define MAX(a,b)  a*b
```

(3) “字符串”可以是常量、表达式、格式串等。例如:

```
#define R 5
#define L 2*PI*R
#define S 3.14159*R*R
```

(4) 在编译预处理时把宏名替换成字符串(也称为宏展开)。

下例中,用PI来代替3.14159,在预编译时先由预处理程序进行宏替换,即用3.14159代替所有的宏名PI,然后再进行编译。

**【案例3.9】** 输入圆的半径,求圆的周长和面积。

```
#include<stdio.h>     //预处理命令必须用#号开头,单独占用一个书写行,尾部无;号
#define PI 3.14159
void main()
{ float r,l,s;          //定义半径r、周长l、面积s
  printf("输入圆的半径:");
  scanf("%f",&r);
  l=2.0*PI*r;
  s=PI*r*r;
  printf("l=%10.4f\ns=%10.4f\n",l,s);
}
```

运行结果:

```
输入圆的半径:4↙
```

```
l = 25.1328
s = 50.2655
```



(5) 一个定义过的宏可以出现在其他新定义宏中,但应注意其中括号的使用,因为括号也是宏代替的一部分。

例如:

```
#define WIDTH 50
#define LENGTH (WIDTH + 20)
```

宏 LENGTH 等价于#define LENGTH (50+20)。

有没有括号其意义截然不同,例如:

```
area = LENGTH * WIDTH;
```

若宏体中有括号,则宏展开后变成:

```
area = (50 + 20) * 50;
```

若宏体中没有括号,即#define LENGTH 50+20,则宏展开后变成:

```
area = 50 + 20 * 50;
```

显然二者的结果是不一样的。

(6) 运算符、括号和已定义过的宏等。因为宏操作仅仅是替换字符串,不涉及其他数据类型,所以在其后面可以出现数值、运算符、已定义的宏等,宏把它们都作为字符串的一部分。

**2. 有参宏定义**

C 语言允许宏带有参数。在宏定义中的参数称为形式参数,简称为形参。在宏调用中的参数称为实际参数。

对带参数的宏,在调用中不仅要宏展开,而且要用实参去替换形参。

定义有参宏的一般格式为

```
#define 宏名(形参表)  字符串
```

这里的宏名和无参数宏的定义一致,也是一个标识符,形参表中可以有一个或多个参数,多个参数之间用逗号分隔。被替换的字符串称为宏体,含有各个形参。

带参宏调用的一般形式为

```
宏名(实参表);
```

将程序中出现宏名的地方均用宏体替换,并用实参代替宏体中的形参。

例如:

```
#define MAX(a,b)  ((a)>(b)?(a):(b))   /* 宏定义 */
```

其中,(a,b)是宏 MAX 的参数表,如果有下面宏调用语句:

```
max = MAX(3,9);                       /* 宏调用 */
```

则在出现 MAX 处用宏体((a)>(b)? (a):(b))替换,并用实参 3 和 9 代替形参 a 和 b。

这里的 max 是一个变量的名称,用来接收宏 MAX 带过来的数值,宏展开如下。

```
max = (3 > 9?3:9);
```

语句运行的结果为 9。

使用带参的宏定义要注意以下几点。

(1) 在定义有参宏时,宏名与右边括号之间不能出现空格,否则系统将空格以后的所有字符均作为替代字符串,而将该宏视为无参宏。

例如:

```
#define MAX (a,b) ((a)>(b)?(a):(b))
```

可以看出,在宏名 MAX 和(a,b)之间存在一个空格,这时将把(a,b)((a)>(b)?(a):(b))作为宏名 MAX 的字符串。宏展开时,宏调用语句:

```
max = MAX(x,y);
```

将变为

```
max = (a,b)(a > b)?a:b(x,y);
```

这样就不会实现原来的功能。正确的书写应该是:

```
#define MAX(a,b) ((a)>(b)?(a):(b))
```

其中,MAX(a,b)是一个整体。

**【案例 3.10】** 计算两个数值之和。

```
#include < stdio.h >
#define SUM(a,b) a + b
//定义的宏带参数,并且宏名和右边括号是一体的
//宏的功能是实现两个数的求和
int main()
{
  int a,b;
  int k;
  printf("输入两个整型数据: ");
  scanf(" % d, % d",&a,&b);
  k = SUM(a,b);
  printf("两个整数之和是: % d\n",k);
}
```

运行结果:

```
输入两个整型数据:
3,5↙
两个整数之和是: 8
```

在这里,因为宏就是简单的替换,是没有数据类型的,所以这个宏既可以实现整数的运算,也适用于浮点数的运算。

(2) 由于运算符优先级不同,定义带参宏时,宏体中与参数名相同的字符序列带圆括号与不带圆括号的意义有可能不一样。

例如:

```
#define S(a,b) a * b
area = S(2,5);
```

宏展开后为

```
area = 2 * 5;
```

如果 area＝S(w,w＋5);,宏展开后 area＝w＊w＋5;,由于乘法的优先级高于加法的优先级,显然得不到希望的值。

如果将宏定义改为

```
#define S(a,b)  (a)*(b)
```

无论是 area＝S(2,5);还是 area＝S(w,w＋5);,都将得到希望的值。

由此可以看出在宏体中适当加圆括号所起的作用。

**【案例 3.11】** 输出数据,体会括号的用途。

```
#include <stdio.h>
#define P1(a,b)  a*b
#define P2(a,b)  (a)*(b)                    //括号的使用
#define P3(a,b)  (a*b)
#define P4(a,b)  ((a)*(b))
void main()
{
  int x=2,y=6;
  //输出运算结果,比较各自的不同
  printf("%5d,%5d\n",P1(x,y),P1(x+y,x-y));
  printf("%5d,%5d\n",P2(x,y),P2(x+y,x-y));
  printf("%5d,%5d\n",P3(x,y),P3(x+y,x-y));
  printf("%5d,%5d\n",P4(x,y),P4(x+y,x-y));
}
```

运行结果:

```
12,    8
12,  -32
12,    8
12,  -32
```

### 3.3.3 条件编译

预处理程序提供了条件编译的功能。可以按不同的条件去编译不同的程序部分,因而产生不同的目标代码文件。这对于程序的移植和调试是很有用的。

常用的条件编译命令有下列三种形式。

形式一:

```
#ifdef 标识符
    程序段 1
#else
    程序段 2
#endif
```

它的功能是:如果标识符已被#define 命令定义过,则对程序段 1 进行编译;否则对程序段 2 进行编译。如果没有程序段 2(它为空),本格式中的#else 可以没有,即可以写为

```
#ifdef 标识符
      程序段 1
#endif
```

形式二:

```
#ifndef 标识符
      程序段 1
#else
      程序段 2
#endif
```

与形式一的区别是将“ifdef”改为“ifndef”。它的功能是：如果标识符未被 #define 命令定义过，则对程序段 1 进行编译，否则对程序段 2 进行编译。这与形式一的功能正相反。

形式三：

```
#if 常量表达式
      程序段 1
#else
      程序段 2
#endif
```

它的功能是：如果常量表达式的值为真(非 0)，则对程序段 1 进行编译，否则对程序段 2 进行编译。因此可以使程序在不同条件下，完成不同的功能。

## 3.4 顺序结构程序设计

顺序结构是最简单、最常用的基本结构。顺序结构的程序是自上而下顺序执行各条语句。下面介绍几个顺序结构程序设计的例子。为了更好地理解程序的执行过程，可以使用流程图将程序的逻辑以图形的方式直观地展示，帮助开发者更好地厘清思路，规划程序的整体架构。

### 3.4.1 流程图

C 语言流程图是一种以图形化方式来展示 C 语言程序执行步骤和逻辑结构的工具，它借助特定的图形符号和连接线，清晰地呈现程序中各个操作环节之间的关系与执行顺序，帮助开发者更好地设计、理解和交流程序逻辑。图 3.3 是 C 语言常见的流程图符号。

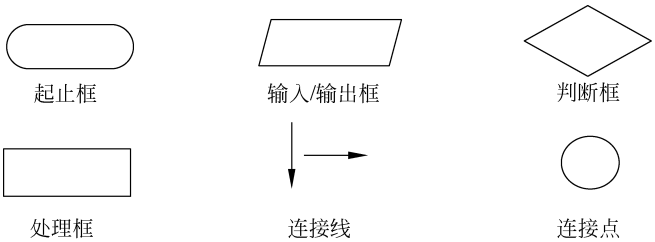


图 3.3 常见的流程图符号

(1) 顺序结构。最基本、最常用的结构，主要包括数据的输入、处理和输出三个步骤，每条语句按照顺序从上向下依次执行，其流程图的基本形态如图 3.4 所示。顺序结构的 C 语言程序一般由三部分组成：预处理命令部分、主函数部分以及自定义函数部分。以下是简化的程序格式(不包含自定义函数)。

```
#include <stdio.h>
#include <头文件>
int  main()
```

```
{
  //数据定义和赋值(输入数据);
  //语句序列(处理数据);
  return 0;
}
```

(2) 选择结构。选择结构又称为分支结构,根据是否满足来确定是否执行若干操作之一,或者确定若干操作中选择哪个操作执行,如图 3.5 所示。虚线框内是双分支选择结构。此结构中包含一个判断表达式,根据给定的表达式条件是否成立而选择执行语句 A 操作或语句 B 操作。详见第 4 章。

(3) 循环结构。在一定条件下反复执行某一部分的操作。如图 3.6 所示的是其中的一种循环结构——while 循环结构。其执行过程是:当给定的条件表达式成立时,执行语句,执行完语句后,再判断条件表达式是否成立,如果仍然成立,再执行语句,如此反复执行语句,直到某一次条件表达式不成立为止,此时不执行语句,而脱离循环结构。详见第 5 章。

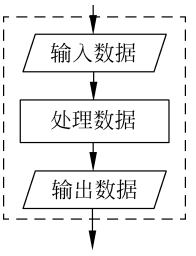


图 3.4 顺序结构基本流程图

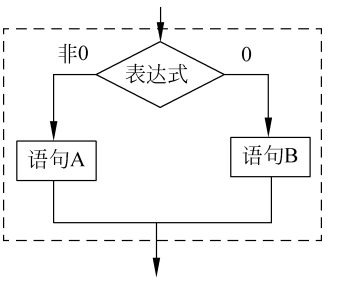


图 3.5 双分支选择结构

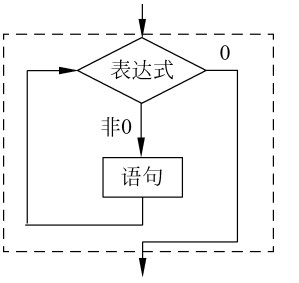


图 3.6 while 循环结构

## 3.4.2 顺序结构程序设计举例

**【案例 3.12】** 键盘输入三角形的三边长 a、b、c,求三角形面积(假设能构成三角形)。

该题是一个基本的顺序结构程序设计问题,包含三个主要步骤:输入数据,处理数据,输出数据。利用海伦-秦九韶公式得到面积为 area=$\sqrt{s(s-a)(s-b)(s-c)}$ 其中,s=(a+b+c)/2。图 3.7 是该题的流程图,依据流程图设计的源程序如下。

```
#include <stdio.h>
//为使用求平方根函数 sqrt(),包含数学库函数文件 math.h
#include <math.h>
int main()
{
 float a,b,c,s,area;
 scanf("%f%f%f",&a,&b,&c);              //scanf()函数默认以空格、Tab 键或回车间隔输入数据
 s=1.0/2*(a+b+c);
 area=sqrt(s*(s-a)*(s-b)*(s-c)); //调用函数 sqrt()
 printf("a=%7.2f,b=%7.2f,c=%7.2f,s=%7.2f\n",a,b,c,s);
 printf("area=%7.2f\n",area);
 return 0;
}
```

运行结果:

```
3 4 5↙
a=   3.00,b=   4.00,c=   5.00,s=   6.00
area=   6.00
```

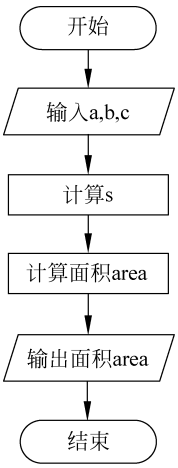


图 3.7　案例 3.12 流程图

**思考**：语句 s＝**1.0**/2＊(a＋b＋c);中为什么是 1.0?

**【案例 3.13】** 将两个两位数的正整数 a、b 合并形成一个整数放在 c 中。合并的方式是：将 a 数的十位数和个位数依次放在 c 数的千位和十位上，b 数的十位数和个位数依次放在 c 数的个位和百位上。

```
#include <stdio.h>
int  main()
{
  int number1,number2;
  int c;
  int dig1_1,dig1_2;                                    //number1 的十位、个位
  int dig2_1,dig2_2;                                    //number2 的十位、个位
  printf("input number1,number2:");
  scanf("%d%d",&number1,&number2);
  dig1_1 = number1/10;
  dig1_2 = number1%10;
  dig2_1 = number2/10;
  dig2_2 = number2%10;
  c = dig1_1*1000+dig1_2*10+dig2_1+dig2_2*100;          //组合新数
  printf("%d",c);
  return 0;
}
```

运行结果：

```
input number, number2:
12 45↙
1524
```

**【案例 3.14】** 输入圆锥的半径 r 和高 h，求圆锥的体积。

```
#include <stdio.h>                                      //预处理命令,尾部不加分号
#define PI 3.14159                                      //预处理命令,定义符号常量 PI
int main()
{
    float r,h,v;                                        //定义半径 r、高 h、体积 v
    printf("请输入圆锥的半径: ");
```

```
    scanf("%f",&r);
    printf("请输入圆锥的高:");
    scanf("%f",&h);
    v=1.0/3*3.14159*r*r*h;
    printf("v=%f\n",v);
    return 0;
}
```

运行结果:

```
请输入圆锥的半径:2↙
请输入圆锥的高:1.5↙
v=6.283180
```

## 3.5 知识要点和常见错误列表

本章知识要点如下。

(1) 结构化程序设计是面向过程程序设计的基本原则。其观点是采用“自顶向下、逐步细化、模块化”的程序设计方法,任何程序设计都由顺序、选择、循环三种基本程序构造而成。

(2) 本章主要介绍了顺序结构程序设计的基本语句(数据定义和赋值语句等),以及常用输入/输出库函数。重点掌握 printf()、scanf()函数和赋值语句,要理解赋值号“=”将右边的值赋给左边变量的实质。

(3) C 语言提供了三种预处理命令:宏定义、文件包含和条件编译。重点掌握文件包含、宏定义的方法以及预处理的应用。

(4) 在顺序结构程序中,一般包括以下几部分。

① 程序开头的编译预处理命令。

在程序中要使用标准函数(又称为库函数),除 printf()和 scanf()外,其他的都必须使用编译预处理命令,将相应的头文件包含进来。

② 顺序结构程序的函数体中,是完成具体功能的各条语句和运算,主要包括:

- 定义需要的变量或常量。
- 输入数据或变量赋初值。
- 完成具体的数据处理。
- 输出结果(尽量放在程序的最后)。

(5) “缩进式”是良好的代码书写习惯,应正确反映计算机执行的逻辑关系。

本章知识常见错误列表如表 3.5 所示。

**表 3.5 本章知识常见错误列表**

| 序号 | 错误类型 | 错误举例 | 分析 |
|---|---|---|---|
| 1 | 变量使用前没定义 | n=a+b | 变量必须先定义后使用,否则出现语法错误 |
| 2 | 赋值语句写错 | n+1=n | 赋值号左边只能是变量,不能是表达式,这一点和数学上的等号是不同的 |
| 3 | 变量没有赋值就使用 | int n,a,b;<br>n=a+b; | 变量 a、b 在使用之前一定要有明确的值,否则会出现一个随机数 |

续表

| 序号 | 错误类型 | 错误举例 | 分析 |
|---|---|---|---|
| 4 | 语句结尾少了分号“;” | int n,i<br>n=1 | 分号“;”在C/C++语句中表示一个语句的结束,在单独的语句中一定要加“;” |
| 5 | 输出变量的格式描述要与变量类型一致 | float a;<br>printf("%d",a); | 变量a类型为float,描述有误,输出0 |
| 6 | 输入语句中变量之间的分隔符 | scanf("%d%d",&a,&b) | 输入数据时以空格或回车分隔 |
| 7 | 输入变量没有加地址符号就使用 | scanf("%d",num) | 变量输入列表里,变量名前有地址符& |
| 8 | 输入控制字符串错误 | scanf("%6.2f",num) | 输入语句中不能指定精度 |
| 9 | 运行程序一次正确后误以为程序正确 | 程序只运行一次得到正确结果就以为完成一个题目了 | 专门设计一些输入数据(如选择程序要检查每个分支、循环程序要检查循环边界等),要多次运行均能得到正确结果 |
| 10 | 输入文件包含命令时拼写错误 | # include<stdio.h><br>void main()<br>{ } | #后不该有空格,include之后该有空格 |
| 11 | 预处理命令后多了分号“;” | #include<stdio.h>;<br>#include<iostream.h>; | #include包含命令不是语句,后面不该有分号 |

# 实训3　格式输入与输出函数的应用

## 一、实训目的

(1) 掌握printf()函数的用法。

(2) 掌握scanf()函数的用法。

(3) 熟悉C/C++程序中输入/输出控制的用法。

## 二、实训任务

(1) 给出程序中printf()函数的输出结果。

(2) 用scanf()函数输入数据,使a=3,b=7,x=8.5,y=71.82,c1='A',c2='a',在键盘上应如何输入?

(3) PM2.5(细颗粒物)的暴露剂量对于评估健康风险至关重要。PM2.5由于其粒径小,能够深入人体肺部甚至进入血液循环,从而对呼吸系统、心血管系统等造成危害。根据环境中的PM2.5浓度、暴露时间和呼吸速率,计算一个人在特定时间段内吸入的PM2.5总量,以此作为暴露剂量的评估指标。已知暴露剂量=污染物浓度×呼吸速率×暴露时间。

## 三、实训步骤

(1) 参考源程序sx3-1.cpp如下。

```
#include<stdio.h>
int main()
{  int a=5,b=7;
   float x=6738564,y=-789.124;
   char c='A';
   long n=1234567;
```

```
    unsigned u = 65535;
    printf("%d%d\n",a,b);
    printf("%3d%3d\n",a,b);
    printf("%f, %f\n",x,y);
    printf("%-10f, %-10f\n",x,y);
    printf("%8.2f, %8.2f, %.4f, %.4f, %3f\n",x,y,x,y,x,y);
    printf("%e, %10.2e\n",x,y);
    printf("%c, %d, %o, %x\n",c,c,c,c);
    printf("%ld,%lo, %x\n",n,n,n) ;
    printf("%u, %o, %x,%d\n",u,u,u,u);
    printf("%s,%5.3s\n","COMPUTER","COMPUTER");
}
```

运行结果：

```
57
  5  7
6738564.000000, -789.124023
6738564.000000, -789.124023
6738564.00,  -789.12, 6738564.0000, -789.1240, 6738564.000000
6.738564e+006, -7.89e+002
A, 65, 101, 41
1234567,4553207, 12d687
65535, 177777, ffff,65535
COMPUTER,  COM
```

（2）参考源程序 sx3-2.cpp 如下。

```
#include<stdio.h>
int main()
{ int a,b;
  float x,y;
  char c1,c2;
  scanf("a=%db=%d",&a,&b);
  scanf("%f%e",&x,&y);
  scanf(" %c%c",&c1,&c2);          //第一个%前为空格
  printf("a=%d,b=%d, x=%f,y=%e,c1=%c,c2=%c ",a,b,x,y,c1,c2);
}
```

运行结果：

```
a=3b=7↙
8.5 71.82↙
Aa↙
a=3,b=7, x=8.500000,y=7.182000e+001,c1=A,c2=a
```

**思考**：上例中第三个 scanf()函数双引号中第一个字符为空格。请上机验证：若没有这个空格字符，输出结果会怎样？为什么？

（3）参考源程序 sx3-3.cpp 如下。

```
#include<stdio.h>
#define BREATHING_RATE 0.5          //假设成年人的平均呼吸速率(单位：立方米/小时)
int main()
 {
    double pm25Concentration;       //PM2.5 浓度(单位：毫克/立方米)
    double exposureTime;            //暴露时间(单位：小时)
```

```
    double exposureDose;            //暴露剂量(单位: 毫克)
    printf("请输入环境中 PM2.5 的浓度(毫克/立方米): ");
    scanf("%lf", &pm25Concentration);
    printf("请输入暴露在该环境中的时间(小时): ");
    scanf("%lf", &exposureTime);
    //计算暴露剂量
    exposureDose = pm25Concentration * BREATHING_RATE * exposureTime;
    printf("在 %.2f 小时内,您吸入的 PM2.5 总量(暴露剂量)为 %.2f 毫克.\n", exposureTime,
exposureDose);
    return 0;
}
```

运行结果：

```
请输入环境中 PM2.5 的浓度(毫克/立方米): 0.25↙
请输入暴露在该环境中的时间(小时): 2↙
```

在 2.00 小时内,您吸入的 PM2.5 总量(暴露剂量)为 0.25 毫克。

# 习　题　3

## 一、选择题

1. 若 a 为 int 类型,且其值为 3,则执行完表达式 a+=a-=a*a 后,a 的值是(　　)。

A. -3　　B. 9　　C. -12　　D. 6

2. 结构化程序设计的三种基本结构是(　　)。

A. 输入、处理、输出　　B. 树状、网状、环状

C. 顺序、选择、循环　　D. 主程序、子程序、函数

3. 若 x 和 y 都是 int 型变量,x=100、y=200,且有下面的程序片段：

```
printf("%d",(x,y) );
```

上面程序片段的输出结果是(　　)。

A. 200　　B. 100

C. 100 200　　D. 输入格式符不够,输出不确定的值

4. 若 int k, g;均为整型变量,则下列语句的输出为(　　)。

```
k=017;  g=111;  printf("%d\t",++k);  printf("%x\n",g++);
```

A. 15 6f　　B. 16 70　　C. 15 71　　D. 16 6f

5. 若有定义 int a; float b; double c;,程序运行时输入 1,2,3 <回车>,能把 1 输入给变量 a、2 输入给变量 b、3 输入给变量 c 的输入语句是(　　)。

A. scanf("%d,%f,%lf", &a,&b,&c);

B. scanf("%d%f%lf", &a,&b,&c);

C. scanf("%d,%lf,%lf", &a,&b,&c);

D. scanf("%d,%f,%f", &a,&b,&c);

6. 在宏定义#define A 3.897678 中,宏名 A 代替一个(　　)。

A. 单精度数　　B. 双精度数

C. 常量　　D. 字符串

7. 设变量定义为 int a，b;，执行下列语句时，输入(　　)，则 a 和 b 的值都是 10。

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

A. 10 10　　B. 10，10

C. a=10 b=10　　D. a=10，b=10

8. 以下叙述中正确的是(　　)。

A. 在 scanf()函数中的格式控制字符是为了输入数据用的，不会输出到屏幕上

B. 在使用 scanf()函数输入整数或实数时，输入数据之间只能用空格来分隔

C. 在 printf()函数中，各个输出项只能是变量

D. 使用 printf()函数无法输出百分号%

9. 在文件包含命令中，被包含文件名用"<>"括起时，寻找被包含文件的方式是(　　)。

A. 直接按系统设定的标准方式搜索目录

B. 先在源程序所在目录搜索，再按系统设定的标准方式搜索

C. 仅在源程序所在目录搜索

D. 仅搜索当前目录

10. 若程序中有宏定义行：

```
#define  N  100
```

则以下叙述中正确的是(　　)。

A. 宏定义行中定义了标识符 N 的值为整数 100

B. 在编译程序对 C 源程序进行预处理时用 100 替换标识符 N

C. 上述宏定义行实现将 100 赋给标识符 N

D. 在运行时用 100 替换标识符 N

11. 设有定义 int a=0，b=1，c=1;，以下选项中，表达式值与其他三个不同的是(　　)。

A. b=a==c　　B. a=b=c　　C. a=c==b　　D. c=a!=c

12. 若有以下程序：

```
#include <stdio.h>
#define  S(x)  x*x
#define  T(x)  S(x)*S(x)
main()
{  int  k=5, j=2;
   printf("%d, %d\n", S(k+j),T(k+j));
}
```

则程序的输出结果是(　　)。

A. 17,289　　B. 492,401　　C. 17,37　　D. 49,289

13. 有以下程序：

```
#include  <stdio.h>
#define  N      5
#define  M      N+1
#define  f(x)  (x*M)
main()
{ int  i1,i2;
```

```
  i1 = f(2);
  i2 = f(1 + 1);
  printf ("%d  %d\n",i1,i2);
  }
```

程序运行后的输出结果是(　　)。

A. 12 12　　B. 11 7　　C. 11 11　　D. 12 7

14. 以下程序的输出结果是(　　)。

```
#include <stdio.h>
 main()
 {  int k = 11;
    printf("%d, %o, %x\n",k,k,k);
}
```

A. 12,11,11　　B. 11,13,13　　C. 11,013,0xb　　D. 11,13,b

15. 有以下程序:

```
#include <stdio.h>
#define F(x,y) (x) * (y)
main()
{ int a = 3,b = 4;
  printf("%d\n",F(a++,b++));
}
```

程序运行后输出结果是(　　)。

A. 12　　B. 15　　C. 16　　D. 20

16. 设有定义 int a=0,b=1;,以下表达式中,会产生"短路"现象,致使变量 b 的值不变的是(　　)。

A. +a||++b　　B. a++||b++

C. ++a&&b++　　D. a++&&b++

17. 以下正确的叙述是(　　)。

A. 在程序的一行中可以出现多个有效的预处理命令行

B. 使用带参宏时,参数的类型应与宏定义时的一致

C. 宏替换不占用运行时间,只占编译时间

D. 宏定义不能出现在函数内部

18. 若变量 a 与 i 已正确定义,且 i 已正确赋值,合法的语句是(　　)。

A. a==1　　B. ++i;　　C. a=a++=5;　　D. a=int(i)

19. 以下选项中正确的定义语句是(　　)。

A. double, a, b;　　B. double, a, b;

C. double a; b;　　D. double a=7, b=7;

20. 以下叙述中正确的是(　　)。

A. 可以把 define 和 if 定义为用户标识符

B. 可以把 define 定义为用户标识符,但不能把 if 定义为用户标识符

C. 可以把 if 定义为用户标识符,但不能把 define 定义为用户标识符

D. define 和 if 都不能定义为用户标识符

## 二、填空题

1. 语句 x++；++x；x=x+1；x=1+x；执行后都使变量 x 中的值增 1，请写出一条同一功能的赋值语句：_____。

2. 语句 b=a=6,a*3；执行后整型变量 b 的值是_____。

3. 语句 b=(a=6,a*3)；执行后整型变量 b 的值是_____。

4. getchar()函数只能接收一个_____。

5. 已知 i=5，语句 a=(i>5)?0:1.6;执行后整型变量 a 的值是_____。

6. 有如下程序：

```
#include <stdio.h>
main()
{
  int x = 072;
  printf(">%d<\n", x+1);
}
```

程序运行后的输出结果是_____。

7. 有以下程序：

```
#include <stdio.h>
main()
{ char  c1='1',c2='2';
  c1=getchar();  c2=getchar();  putchar(c1);  putchar(c2);
}
```

当运行时输入：a <回车>后，输出的 c1 的值是_____，c2 的值是_____。

8. 有以下程序：

```
main()
{ char a,b,c,d;
  scanf("%c,%c,%d,%d",&a,&b,&c,&d);
  printf("%c,%c,%c,%c\n",a,b,c,d);
}
```

若运行时从键盘上输入：6,5,65,66↙，则输出结果是_____。

9. 以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{  int  a=3;
   printf("%d\n", (a+=a-=a*a));
}
```

10. 设 a、b、c、d、m、n 均为 int 型变量，且 a=5、b=6、c=7、d=8、m=2、n=2，则逻辑表达式(m=a>b)&&(n=c>d)运算后，n 的值为_____。

11. 设有定义 int n = 1234; double x = 3.1415;，则执行语句 printf("%3d,%1.3f\n", n, x)；后 n 的值是_____，x 的值是_____。

12. 设 int x=1, y=1;，执行语句!x&&y--;后 y 的值是_____。

13. 若有以下程序：

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

```
main()
{  int  a = 0,b = 0,c = 0;
   c = (a += ++b, b += 4);
   printf(" % d, % d, % d\n",a,b,c);
}
```

运行程序,a 的值是_____,b 的值是_____,c 的值是_____。

14. 设有宏定义:#define MYSWAP(z,x,y) {z=x;x=y;y=z;}

以下程序段通过宏调用实现变量 a、b 内容交换,请填空。

float a=5,b=16,c; MYSWAP(_____,a,b);

15. 下面程序的输出结果是_____。

```
#define  CIR(r)  r * r
void main()
{ int a = 1, b = 2, t;
  t = CIR(a + b);
  printf(" % d\n",t);
}
```

**三、判断题**

1. C 语言本身不提供输入/输出语句,输入和输出操作是由函数来实现的。 ( )

2. 语句 scanf("%7.2f",&a);是一个合法的 scanf()函数。 ( )

3. 若 int i =3;,则 printf("%d",-i++);输出的值为-4。 ( )

4. 语句 printf("%f%%",1.0/3);输出为 0.333333。 ( )

5. 若有变量定义和语句:int a;char c;float f;scanf("%d,%c,%f",&a,&c,&f);

若通过键盘输入 10,A,12.5,则 a=10,c='A',f=12.5。 ( )

6. 若有宏定义:#define S(a,b)t=a;a=b;b=t,由于变量 t 没定义,则此宏定义是错误的。 ( )

7. 一个 include 命令可以指定多个被包含的文件。 ( )

**四、程序阅读题**

1. 写出以下程序的运行结果。

```
#include <stdio.h>
#define  S(x)  4 * (x) * x + 1
main()
{  int k = 5, j = 2;
   printf(" % d\n", S(k + j) );
}
```

2. 写出以下程序的运行结果。

```
#include <stdio.h>
#define    F(x)       2.84 + x
#define    PR(a)      printf(" % d" ,(int)(a))
#define    PRINT(a)   PR(a);putchar('\n')
main()
{
   PRINT( F(5) * 2 );
}
```

3. 写出以下程序的运行结果。

```
#define N 10
#define s(x)  x * x
#define f(x)  (x * x)
void main()
{
 int i1,i2;
 i1 = 1000/s(N);
 i2 = 1000/f(N);
 printf("%d  %d\n",i1,i2);
}
```

**五、编程题**

1. 输入矩形的长和宽，求矩形的周长和面积。

要求：矩形的长、宽、周长和面积均为 float 型或 double 型数据。

2. 已知将华氏温度 F 转换为摄氏温度 C 的公式为 C＝5÷9×(F－32)，请编写程序，将输入的华氏温度转换为摄氏温度，温度保留 1 位小数。

3. 编写一个程序，模拟简单的购物结算过程。用户需要依次输入商品的单价、购买数量以及折扣率(折扣率为 0～1 的小数，如 0.8 表示 8 折)，计算并输出商品的总价(总价＝单价×数量×折扣率)。