第 3 章　程序的结构

【本章内容】

- 从 C 到 C++；
- C++ 的简单程序与使用；
- 程序结构与效率。

3.1　从 C 到 C++

3.1.1　概述

本章要求了解 C 和 C++ 的基本语法，掌握基础的程序结构和简单算法，并且能够编写简单的 C++ 程序。

3.1.2　C 语言的语法

C 和 C++ 的语法有很多共性，C++ 可以完全兼容 C 语言，使用 C 语言编写的程序可以在 C++ 的编译器上运行，C 和 C++ 混编的程序也可以在 C++ 的编译器上通过编译，反之则不一定可以，因为 C++ 有很多 C 语言标准没有定义的扩展内容。C 语言语法还有一定的代码规范，在命名方面，可以借用下画线和词汇做到简明表达功能的目的，使程序易读易懂；在语句方面，应该尽量每条语句单独占一行，增强程序的易读性。

1. 常见文件的扩展名

C 语言常见文件的扩展名有以下 3 种：

(1) .c 是 C 语言源文件，在编写代码时创建；

(2) .o 是目标文件，在编译成功时产生；

(3) .out 是可执行文件，在链接成功时产生。

2. 数据类型

数据类型是 C 语言数据结构的表达形式，选择合适的数据类型对程序来说是非常重要的。

(1) 基本类型。

① 整型。整型分为以下 6 种。

- 整型(int)：长度为 4B(在 Turbo C 2.0 环境中长度为 2B)，取值范围是－2147483648～2147483647。
- 短整型(short int)：长度为 2B，取值范围是－32768～32767。
- 长整型(long int)：长度为 4B，取值范围是－2147483648～2147483647。
- 无符号整型(unsigned int)：长度为 4B，取值范围是 0～4294967295。
- 无符号短整型(unsigned short int)：长度为 2B。

• 无符号长整型(unsigned long int)：长度为 4B。

不同的编译环境，整型数据在内存中所占的字节数不一样。可以使用 sizeof 运算符，确定整型数据在编译器中所占的字节数。例如：

```
int a;
```

表示定义一个类型为整型的变量 a。

```
sizeof(a);
```

表示计算变量 a 所占的字节数，等价于

```
sizeof(int)
```

② 字符型(char)。字符型包括中文字符、英文字符、数字字符和其他 ASCII 字符，一共 256 个字符。C 语言的任何一个字符都可以用转义字符来表示。例如：\101 表示字符'A'，\134 表示反斜杠(\)，\XOA 表示换行。

③ 浮点型。浮点型包括两种。

单精度(float)：长度为 4B，取值为 3.4×10^{-38}～3.4×10^{38}，可提供 6～7 位有效数字。

双精度(double)：长度为 8B，取值为 1.7×10^{-308}～1.7×10^{308}。

说明：printf 输出 float 和 double 都可以用%f，double 还可以用%lf。scanf 输入 float 用%f，double 输入用%lf，不能混用。

【例 3-1】 用 printf 语句输出一个 float 型和一个 double 型变量的和。代码如下：

```
1.    #include<stdio.h>
2.    int main()
3.    {
4.        float x1,x2;double y1,y2;
5.        x1=1111111111111.111111111;
6.        x2=2222222222222.222222222;
7.        y1=1111111111111.111111111;
8.        y2=2222222222222.222222222;
9.        printf("x1+x2=%f\ny1+y2=%lf\n",x1+x2,y1+y2);
10.   }
```

运行结果如图 3-1 所示。

```
x1+x2=3333333450752.000000
y1+y2=3333333333333.333000

--------------------------------
Process exited after 0.5264 seconds with return value 0
请按任意键继续. . .
```

图 3-1　运行结果

④ 枚举(enum)。枚举就是将一个变量可能存在的值都一一列举出来，变量的值只限于列举出来值的范围内。枚举类型的定义形式为：

```
enum typeName{valueName1,valueName2,…,valueNamen};
```

enum 是一个新的关键字,专门用来定义枚举类型,这也是它在 C 语言中的唯一用途;typeName 是枚举类型的名字;valueName1,valueName2,…,valueNamen 是每个值对应的名字列表。

例如,一个星期有 7 天,在此可以为每天定义一个名字并且为其赋值:

```
enum week{ Mon, Tues, Wed, Thurs, Fri, Sat, Sun};
```

因为枚举值默认从 0 开始,往后逐个加 1,所以 week 中的 Mon,Tues,…,Sun 对应的值分别为 0,1,…,6。

上述例子也可以写成:

```
enum week{ Mon=1, Tues=2, Wed=3, Thurs=4, Fri=5, Sat=6, Sun=7 };
```

也可以只给第一个变量指定值:

```
enum week{ Mon=1, Tues, Wed, Thurs, Fri, Sat, Sun };
```

这样枚举值就从 1 开始递增,week 中的 Mon,Tues,…,Sun 对应的值分别为 1,2,…,7。

定义枚举变量:

```
enum week a, b, c;
```

也可以在定义枚举类型的同时定义枚举变量:

```
enum week{ Mon=1, Tues, Wed, Thurs, Fri, Sat, Sun } a, b, c;
```

可以把列表中的值赋给枚举变量:

```
enum week{ Mon=1, Tues, Wed, Thurs, Fri, Sat, Sun };
enum week a=Mon, b=Wed, c=Sat;
```

或者

```
enum week{ Mon=1, Tues, Wed, Thurs, Fri, Sat, Sun } a=Mon, b=Wed, c=Sat;
```

(2) 构造类型。

① array(数组)。为了方便处理数据,把具有相同类型的若干变量按有序的形式组织起来,这些按序排列的同类数据元素的集合称为数组。

一个数组包含多个数组元素,这些数组元素可以是基本数据类型或是构造类型。按数组元素的类型不同,数组又可分为数值数组、字符数组、指针数组、结构数组等各种类别。

数组的形式为

```
arrayName[index]
```

其中，arrayName 为数组名称，index 为下标。例如：

```
int a[4]
```

定义了一个数组长度为 4 的整型数组，它的名字是 a。数组中的每个元素都有一个序号，这个序号从 0 开始，而不是 1 开始，这个序号则是下标。使用数组元素时，指明下标即可，例如 a[0]表示第 0 个元素，a[2] 表示第 2 个元素。

提示：数组中每个元素的数据类型必须相同，对于 int a[4]每个元素都必须为 int 型。数组是一个整体，它的内存是连续的，表 3-1 是 int a[4]的内存示意表。

表 3-1　int a[4]的内存示意表

a[0]	a[1]	a[2]	a[3]

② struct(结构体)。C 语言提供了一种构造类型——结构体，用来处理一组类型不同或相同的数据。结构体的定义形式如下：

```
struct 结构体名
{
    结构体
};
```

结构体是一种集合，它里面包含了多个变量或数组，它们的类型可以相同，也可以不同，每个这样的变量或数组都称为结构体的成员。例如：

```
1.    struct stu
2.    {
3.        char *name;                    //姓名
4.        int num;                       //学号
5.        int age;                       //年龄
6.        char group;                    //所在学习小组
7.        float score;                   //成绩
8.    };
9.    struct student
10.   {
11.       char *name;                    //姓名
12.       int num;                       //学号
13.       float score;                   //成绩
14.   };
```

student 为结构体名，它包含了 3 个成员：name、num 和 score。结构体成员的定义方式与变量和数组的定义方式相同，只是不能初始化。

③ union(共用体)。在 C 语言中，有一种和结构体非常类似的语法，叫做共用体，又称

为联合体。它的定义格式如下：

```
union 共用体名
{
成员列表
};
```

结构体和共用体的区别在于，结构体的各个成员会占用不同的内存，互相之间没有影响；而共用体的所有成员占用同一段内存，修改一个成员会影响其余所有成员。结构体占用的内存大于等于所有成员占用的内存的总和（成员之间可能会存在缝隙），共用体占用的内存等于数据长度最长的成员占用的内存。

共用体的特点如下：

- 共用体使用了内存覆盖技术，即同一时刻只能保存一个成员的值。
- 在共用体变量中起作用的是最后一次存放的成员，在存入一个新成员后，原有成员就会失去作用。
- 共用体变量的地址和共用体中各成员的地址是同一地址。
- 不能对共用体变量名赋值，也不能企图引用变量名来得到一个值。
- 共用体类型可以出现在结构体类型的定义中，反之，结构体也可以出现在共用体类型的定义中。

共用体也是一种自定义类型，可以通过它来创建变量，例如：

```
1.    union data                                    //定义一个名称为 date 的共用体
2.    {
3.        int n;                                    //定义整型变量 n
4.        char c;                                   //定义字符型变量 c
5.        double f;                                 //定义双精度浮点型变量 f
6.    };
7.    union data a, b, c;                           //创建共用体变量 a,b,c
```

也可以在定义共用体的同时创建变量：

```
1.    union data
2.    {
3.        int n;
4.        char ch;
5.        double f;
6.    } a, b, c;
7.    union data
8.    {
9.        int n;
10.       char c;
11.       double f;
12.   } a, b, c;
```

如果后续程序不再定义新的变量，也可以将共用体的名字省略：

```
union                    //省略共用体名字 date
{
    int n;
    char ch;
    double f;
} a, b, c;
```

(3) 指针类型。想要理解指针的概念，首先必须要了解计算机中的数据都是存在内存中的，不同类型的数据在内存中占用的字节数不一样，例如整型 int 占用 4B 空间，为了正确的访问这些数据，必须为每个字节编上号码，就像每个人要有自己的名字(默认不重名)。每个字节的编号都是唯一的，根据编号可以很容易找到某个字节，这些编号被称为地址或者指针。

如果一个变量用来存放另一变量的地址，则称它为指针变量。指针变量可以存放基本类型数据的地址，也可以存放数组、函数以及其他指针变量的地址。使用指针变量时，应该对其进行初始化。常见指针变量的定义如表 3-2 所示。

表 3-2　常见指针变量的定义

定　义	含　义
int * p;	p 可以指向 int 类型的数据，也可以指向类似 int arr[n] 的数组
int * * p;	p 为二级指针，指向 int * 类型的数据
int * p[*n*];	p 为指针数组。[ ] 的优先级高于 *，所以应该理解为 int *(p[*n*]);
int(* p)[*n*];	p 为二维数组指针
int * p();	p 是一个函数，它的返回值类型为 int *
int(* p)();	p 是一个函数指针，指向原型为 int func() 的函数
void *	无类型指针，可以指向任何类型的数据

3. 输入输出函数

(1) printf。printf 有两种格式。

① 格式字符串(以%开头)，形式如下：

```
printf(格式控制,输出表列)
```

② 非格式字符串(原样输出)组成，形式如下：

```
printf(输出表列)
```

例如：

```
#include<stdio.h>
int  main()
{
    char ch='a';
```

```
5.        printf("请输出一个字符:\n");        //原样输出" "中的内容
6.        printf("%c",ch)                    //按%c的格式输出ch变量代表的内容
7.    }
```

(2) scanf。形式如下：

```
scanf(格式控制,地址表列)
```

例如：

```
1.    #include<stdio.h>
2.    int main()
3.    {
4.        int a;
5.        float b;
6.        double c;
7.        scanf("%d,%f,%lf",&a,&b,&c);
8.        //输入不同类型的变量a,b,c的值
9.    }
```

在以上程序中，将

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

改为

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

注意，此时输入时的格式应该和" "内的对应，即

```
a=1,b=1,c=1
```

在 scanf 函数中，“&”千万不能省略。

(3) putchar。putchar 等同于 printf(%c)，位于<stdio.h>中，形式如下：

```
putchar(字符变量)
```

【例 3-2】 用 putchar 输出不同字符变量的值。代码如下：

```
1.    #include<stdio.h>
2.    int main()
3.    {
4.        char a='B',b='o',c='k';                    //定义3个字符变量并赋值
5.        //以下4行代表分别输出字符变量a、b、c的值
6.        putchar(a);
7.        putchar(b);
8.        putchar(b);
9.        putchar(c);
10.   }
```

运行结果如图 3-2 所示。

```
Book
--------------------------------
Process exited after 0.1319 seconds with return value 0
请按任意键继续. . .
```

图 3-2　例 3-2 的运行结果

(4) getchar。getchar 等同于 scanf(%c,&mchar),形式如下：

```
getchar();
```

【例 3-3】 用 getchar 输入一个字符。代码如下：

```
1.    #include<stdio.h>
2.    int main()
3.    {
4.        char c;
5.        printf("请输入一个字符:");
6.        c=getchar();
7.        putchar(c);
8.    }
```

运行结果如图 3-3 所示。

```
请输入一个字符：A
A
--------------------------------
Process exited after 3.319 seconds with return value 0
请按任意键继续. . .
```

图 3-3　例 3-3 的运行结果

4. 语句

C 语言的程序功能部分是由各种执行语句实现的,基本语句可以分为 5 类：表达式语句、函数调用语句、控制语句、复合语句和空语句。

(1) 表达式语句。表达式语句用于计算表达式的值,表达式语句由表达式和分号(;)组成。其一般形式如下：

```
表达式;
```

例如：

```
a=x+y;                    //赋值表达式语句
```

(2) 函数调用语句。执行函数语句就是调用函数体并把实际参数赋给函数定义中的形式参数,然后执行被调用的函数体中的语句,求取函数值。函数调用语句由函数名、实际参数和分号(;)组成。其一般形式如下：

```
函数名(实际参数表);
```

例如：

```
printf("hello C++");            //调用 printf 函数
```

(3) 控制语句。控制语句用于控制程序的流程，是三大程序结构的重要组成部分。控制语句分为 3 类。条件判断语句：if 语句和 switch 语句；循环执行语句：do…while 语句、while 语句和 for 语句；转向语句：break 语句、goto 语句、continue 语句和 return 语句。

(4) 复合语句。用{ }把一些语句和声明括起来的语句，称为复合语句(又称语句块)。在程序中应把复合语句看成是一条语句，而不是多条语句。例如：

```
{a+b=c;k=i * j;}                    //这是一条复合语句
```

(5) 空语句。只有分号(;)组成的语句称为空语句。空语句不执行任何操作，可用来作空循环体。例如：

```
while(getchar()!='\n');                    //while 语句的循环体为空语句
```

5. 表达式与运算符

(1) 逻辑运算符与逻辑表达式如表 3-3 所示。

表 3-3　逻辑运算符及其运算规则

值		逻辑非(!)		逻辑与(&&)	逻辑或(||)
a	b	!a	!b	a&&b	a||b
真	真	假	假	真	真
真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

与运算符(&&)以及或运算符(||)均为双目运算符，具有左结合性。非运算符(!)为单目运算符，具有右结合性。它们的优先级次序是“非→与→或”依次由高到低。

表示常量或者变量之间的逻辑关系的逻辑表达式形势如下：

```
表达式 逻辑运算符 表达式
```

例如：

```
a&&b&&c                    //表示变量 a 与 b 与 c
```

(2) 算术运算符与算术表达式如表 3-4 所示。

表 3-4　算术运算符

加	减	乘	除	自增	自减	求余
+	−	*	/	−−	++	%

＋＋i：i 自增 1 后再参与其他运算。－－i：i 自减 1 后再参与其他运算。i＋＋：i 参与运算后，i 的值再自增 1。i－－：i 参与运算后，i 的值再自减 1。

算术表达式：用算术运算符和括号将运算对象连接起来的式子称为算术表达式。

例如：

```
p=++i                    //假设 i 的原值为 3,i 的值先加 1 变成 4 再赋给变量 p,p 为 4
q=i--                    //将 i 的值 3 赋给变量 q,q 为 3,然后 i 的值加 1 变为 4
```

(3) 关系运算符与关系表达式如表 3-5 所示。

表 3-5　关系运算符

大于	小于	等于	大于等于	小于等于	不等于
＞	＜	＝＝	＞＝	＜＝	!＝

关系运算符都是双目运算符，其结合性均为左结合。关系运算符的优先级低于算术运算符，高于赋值运算符。在 6 个关系运算符中，＜、＜＝、＞、＞＝的优先级相同，高于＝＝和!＝，＝＝和!＝的优先级相同。

关系表达式用于表示常量或者变量间的关系，合法的关系表达式如下：

```
表达式 关系运算符 表达式
```

例如：

```
a+b>i+j
```

(4) 位操作运算符与位操作表达式如表 3-6 所示。

表 3-6　位操作运算符与位操作表达式

位与	位或	位非	位异或	左移	右移
&	|	～	^	＜＜	＞＞

例如：

```
a&b;                     //表示变量 a 位与变量 b
```

(5) 赋值运算符与赋值表达式如表 3-7 所示。

表 3-7　赋值运算符与赋值表达式

简单赋值	＝				
复合算数赋值	＋＝	－＝	＊＝	/＝	%＝
复合位运算赋值	&＝	|＝	^＝	＞＞＝	＜＜＝

赋值表达式用于对变量进行赋值的表达式。例如：a＝2 和 a＋＝b。

(6) 条件运算符与条件表达式。

条件运算符(？：)：三目运算符，要求有 3 个操作对象。

条件表达式：用于表达 3 个变量或常量，在某一条件下进行赋值的表达式。形式如下：

```
表达式 1 ?  表达式 2 : 表达式 3
```

例如：

```
max=(a>b) ?a : b                    //如果 a>b 为真，则把 a 赋予 max，否则把 b 赋予 max
```

3.1.3 C++ 对 C 的扩充

1. 关于运算符的扩充

(1) 作用域运算符。C++ 提供作用域运算符(::)，它能指定用户所需要的作用域。作用域，顾名思义，就是作用的领域。在 C++ 中，作用域运算符可以用来解决局部变量和全局变量重名的问题。局部变量是指在函数或模块内部定义的变量，其作用域为定义该局部变量的模块或函数内。全局变量是指在函数或者模块之外定义的变量，其作用域是整个程序。作用域运算符也可被用来访问命名空间中的名字。如果需要在局部变量的作用域内使用同名的全局变量，需要在该变量的前面加上“::”，此时“::num”就代表全局变量。

【例 3-4】 作用域运算符举例。代码如下：

```
1.   #include<iostream>                          //使用头文件 iostream
2.   int avar;                                   //定义一个全局变量 avar
3.   int main()
4.   {
5.       int avar;                               //定义一个同名的局部变量 avar
6.       avar=25;                                //将 25 赋值给局部变量 avar
7.       ::avar=10;                              //通过::将 10 赋值给全局变量
8.       std::cout<<"local avar="<<avar<<std::endl;
9.       //使用::访问命名空间 std 中 cout 和 endl，并且输出局部变量 avar 的值
10.      std::cout<<"global avar="<<::avar<<std::endl;
11.      //使用::访问命名空间 std 中 cout 和 endl，并且输出全局变量 avar 的值
12.      return 0;                               //main 函数要求必须返回 0
13.  }
```

运行结果如图 3-4 所示。

```
local avar = 25
global avar = 10

--------------------------------

Process exited after 4.258 seconds with return value 0
请按任意键继续. . .
```

图 3-4　例 3-4 运行结果

程序分析：在所有花括号之外定义的全局变量 avar，从声明语句开始，直到整个程序结束都可以被访问，而在 main 函数中定义的同名局部变量 avar，从声明语句开始，直到 main

函数结束都可以被访问，但在 main 函数之外则无法访问。

（2）动态分配/撤销内存运算符 new 和 delete。在 C 语言中 malloc 和 free 的作用分别是分配和撤销内存空间，但是 malloc 函数在使用时需要求出所需字节数，并且由于该函数是一个指针型函数，指针的基类型为 void，必须进行强制类型转换才能使返回的指针指向具体的数据。C++ 则提供了简便而功能较强的 new 和 delete 运算符取代 malloc 和 free 函数。

new 使用的一般格式如下：

```
new 类型[初值];
```

delete 使用的一般格式如下：

```
delete[]指针变量;
```

例如：

```
1.    new int(10);     //开辟一个存放整数的空间,并指定该整数的初值为 10
2.    float *pt=new char[10];
3.    //开辟一个存放字符数组的空间并指定该数组有 10 个元素
4.    //将返回的指向数组的指针赋给指针变量 pt
5.    float *p=new float(3.14);
6.    //开辟一个存放实数的空间,并指定该实数初值为 3.14
7.    //将返回的指向实型数据的指针赋给指针变量 p
8.    delete p;        //撤销上面用 new 开辟的实数空间
9.    delete []pt;     //在指针变量 pt 前加一对括号,表示撤销上面用 new 开辟的数组空间
```

2. 关于变量的扩充

（1）用 const 定义常变量。在 C 语言中常用宏定义符号常量，使用 # define 指令实现，例如：

```
#define PI 3.14159
```

即在系统预编译时将程序中出现的所有字符串 PI 置换成 3.14159，宏定义不分配存储单元。在 C++ 中可以使用 const 限定符定义常变量，const 对象一旦创建后就不能被改变，所以必须对其进行初始化。例如：

```
1.    const int bufsize=512; //定义整型变量 bufsize,并限定缓冲区的大小为 512
2.    bufsize=500;           /*试图改变 bufsize 的大小,由于上面 const 将 bufsize 限
                                 定为 512 大小的常量,所以这是错误的,不能实现*/
3.    int a=50;              //定义一个整型变量 a,并且初始化为 50
4.    const int b=a;         //利用整型变量 a 初始化 b,将 b 初始化值为 50 的整型常量
5.    int j=b;               //用整型常量 b 初始化整型变量 j 也是合法的
```

（2）字符串变量。在 C++ 中不仅可以使用字符数组处理字符串，还可以使用另一种数据类型——string 类型。使用 string 类型需要使用 string 头文件，即应在文件开头加上“#

include <string>”。

例如：

```
string str1="hello";              //定义一个字符串变量 str1,并对其初始化,内容为 hello
string str2="love";               //定义一个字符串变量 str2,并对其初始化,内容为 love
```

字符串常量以\0'作为结束符,所以字符串常量"love"共有 5 个字符,但使用字符串常量初始化字符串变量时,字符串变量就不包括\0'了,所以字符串变量 str2 中的字符为"love",共 4 个字符。

```
str2=str1;                        //将字符串变量 str1 复制给 str2
```

在定义字符串变量时不需要指定长度,其长度随初始化的字符串长度而改变,所以即使 str1 与 str2 长度不一样,也可以进行这种复制,即 str2 变量的内容变为“hello”。

```
str1=str1+str2;                   //连接 str1 和 str2,连接后 str1 为"hello love"
```

(3) 变量的引用。在 C++ 中引用的作用是为变量起另一个名字,可以通过引用声明符“&”来实现这个功能。具体用法如下：

```
1.    int i;          //定义一个整型变量 i
2.    int j;          //定义一个整型变量 j
3.    int &a=i;       /*声明 a 是一个整型变量的引用变量,被初始化为 i,即 a 是 i 的别名,它
                        们都代表同一变量*/
4.    int &a=j;       //错误,a 已经是变量 i 的别名,不能再当作其他变量的别名
5.    int &a=1;       //错误,引用类型的初始值必须是一个对象
6.    double k;       //定义一个浮点型变量 k
7.    int &b=k;       /*错误,此处引用类型的初始值应该是整型变量,而 k 在上一句中已经被
                        定义成浮点型变量*/
```

3. C++ 的输入输出

在 C++ 中使用 cin 函数进行输入,cout 函数进行输出。输入输出函数都被定义在标准库 iostream 中,标准库中又包含两个基础类型 istream 和 ostream,分别表示输入和输出流,而 cin 函数和 cout 函数分别是这两个类型的对象。要想使用输入输出功能,需要在程序开始使用#include<iostream>。cout 函数必须和插入运算符(<<)一起使用,cin 函数必须和提取运算符(>>)一起使用。

【例 3-5】 使用 cout 函数输出不同类型的数据。代码如下：

```
1.    #include<iostream>                                    //使用头文件 iostream
2.    using namespace std;                                  //使用命名空间 std
3.    int main()
4.    {int a=3;
5.    float b=3.14;
6.    char c='B';
```

```
7.    cout<<"a="<<a<<","<<"b="<<b<<","<<"c="<<c<<endl; //输出不同类型的数据
8.    return 0;
9.    }
```

运行结果如图 3-5 所示。

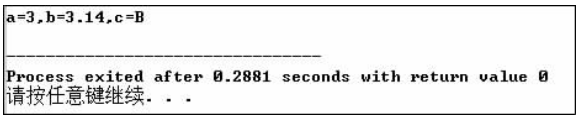

图 3-5　例 3-5 的运行结果

如果要指定输出所占的列数,可以使用控制符 setw 进行设置。例如 setw(5)的作用是为下一个输出项预留 5 列的空间,如果下一个输出项的长度不足 5 列,则输出项数据向右对齐,若超过 5 列则按实际长度输出。

【例 3-6】 修改例 3-5 的程序,观察运行结果。代码如下:

```
1.    #include<iostream>                    //使用头文件 iostream
2.    #include<iomanip>                     //使用头文件 iomanip,才能使用控制符 setw
3.    using namespace std;                  //使用命名空间 std
4.    int main()
5.    {
6.        int a=3;
7.        float b=3.14;
8.        char c='B';
9.        cout<<"a="<<setw(6)<<a<<endl<<"b="<<setw(6)<<b<<endl<<"c="<<
          setw(6)<<c<<endl;
10.       //输出不同类型的数据,并且指定输出项列数为 6
11.       return 0;
12.   }
```

运行结果如图 3-6 所示。

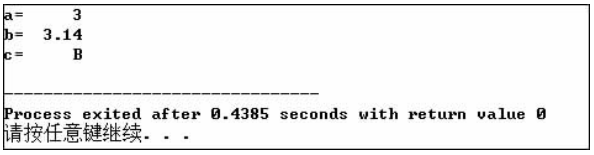

图 3-6　例 3-6 的运行结果

【例 3-7】 使用 cin 进行输入。代码如下:

```
1.    #include<iostream>
2.    using namespace std;
3.    int main()
4.    {
```

```
5.        int i;
6.        float j;
7.        cin>>"i="">>i>>"j="">>j;        //从键盘输入 i 和 j 的值,用空格隔开,如 1 1.1
8.        cout<<i<<","<<j<<endl;         //输出以上输入的值
9.        return 0;
10.   }
```

运行结果如图 3-7 所示。

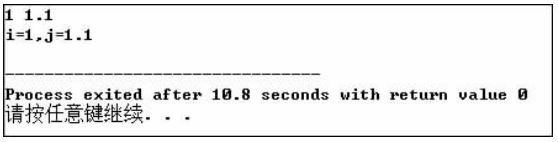

图 3-7　例 3-7 的运行结果

3.2　C++ 的简单程序使用

本节将介绍几个简单的 C++ 程序,并作出详细解释。

【例 3-8】 输出一行字符“Hello C++ !”。代码如下:

```
1.    #include<iostream>                          //使用 iostream 头文件
2.    using namespace std;                        //使用命名空间 std
3.    int main()                                  //定义一个整型的 main 函数
4.    {
5.        cout<<"Hello C++!"<<endl;               //输出"Hello C++!"
6.        return 0;                               //若程序正常结束,则向系统返回 0
7.    }
```

运行结果如图 3-8 所示。

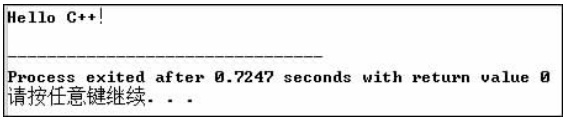

图 3-8　例 3-8 的运行结果

程序分析:

在本例中,#include 是一个预处理指令,它通常与头文件写在同一行,本例中用于告诉编译器要使用 iostream 库。

using namespace std 是指使用命名空间,如果要使用 C++ 标准库中的内容(如 #include 指令)则需使用“using namespace std”,因为 C++ 标准库中的类和函数是在命名空间 std 中声明的。为了方便,在编写程序时,都可以在程序前面写上“using namespace std”。

接下来就是定义主函数。标准 C++ 规定,main 函数必须定义成 int 型(整型),同时需要在 main 函数最后一行加一句“return 0;”。如果程序执行正常,则向操作系统返回数值

0，否则返回－1。

在 C++ 中可以使用 cout 进行输出，当然也可以用 printf 进行输出。输出语句最后的“endl”是控制符，作用是换行，与'\n'的作用相同。

【例 3-9】 输出字符变量经过运算后的值。代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    char a='a',b='b',c='c',d=65;          //分别定义 4 个不同的字符变量
    cout<<a<<b<<c<<d<<endl;               //输出 4 个字符变量的原值
    a=a+1;
    b=b+2;
    c=c+3;
    d=d+4;
    cout<<a<<b<<c<<d<<endl;               //输出 4 个字符变量经过运算后的值
    return 0;                             //若程序正常结束，则向系统返回 0
}
```

运行结果如图 3-9 所示。

```
abcA
bdfE

--------------------------------
Process exited after 0.6079 seconds with return value 0
请按任意键继续. . .
```

图 3-9　例 3-9 的运行结果

程序分析：以上程序是对 ASCII 码的一个简单了解，由于 ASCII 编码统一规定了所有的大写和小写字母，数字 0～9、标点符号，以及在美式英语中使用的特殊控制字符所用的二进制数。

4 个字符变量 abcd 的原值分别是“a，b，c，A”，再进行加法运算，例如“a＝a＋1”表示变量 a 的值加 1，然后赋值给变量 a，此时变量 a 的值变为字符 b，以此类推，执行完所有表达式语句后，输出为“bdfE”。

【例 3-10】 任意输入两个数，并求这两个数的和。代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    int i,j,sum;
    cin>>i>>j;                          //从键盘输入变量 i 和 j 的值，如 1 和 2
    cout<<"i="<<i<<"j="<<j<<endl;       //分别输出变量 i 和 j 的值
    sum=i+j;                            //变量 i 和 j 做加法运算后的值赋给变量 sum
    cout<<"i+j="<<sum<<endl;            //输出变量 sum 的值
```

```
    return 0;                                    //若程序正常结束,则向系统返回 0
}
```

运行结果如图 3-10 所示。

```
1 2
i=1,j=2
i+j=3

--------------------------------
Process exited after 22.93 seconds with return value 0
请按任意键继续. . .
```

图 3-10　例 3-10 的运行结果

程序分析：这个程序的作用是求 i 和 j 的和 sum。在执行输入语句时,从键盘输入的第一个数据赋给变量 i,第二个数据赋给变量 j。执行表达式语句

```
sum=i+j;
```

时,先计算 i+j 的值,再将这个值赋值给变量 sum,这样就得到了两数之和,最后再执行输出语句就能在屏幕上显示出来。

【例 3-11】 从键盘输入两个整数 a 和 b,比较它们的大小并输出大者。代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    int max(int x,int y);                        //对 max 函数作声明
    int a,b,c;
    cin>>a>>b;
    c=max(a,b);                                  //调用 max 函数
    cout<<"max="<<c<<endl;
    return 0;
}
int max(int x,int y)                             //定义 max 函数
{
    int z;
    if(x>y)z=x;                                  //if 判断语句
    else z=y;
    return (z);
}
```

若输入 a,b 的值分别为 15,20,则运行结果如图 3-11 所示。

```
15 20
a=15,b=20,max=20

--------------------------------
Process exited after 3.095 seconds with return value 0
请按任意键继续. . .
```

图 3-11　例 3-11 的运行结果

程序分析：比较两个数大小的问题相当于两个装满水的杯子交换各自的液体的问题，都需要通过第三方，即另一个变量或者另一个杯子才能解决。本程序包括两个函数，分别是主函数 main 和被调用的函数 max。其中函数 max 的作用是将 x 和 y 中较大的值赋给变量 z，变量 z 就相当于另一个杯子，作用是存放较大数。

使用函数 max 有几点需要注意，即需要为函数 max 作出声明，然后才可以对其进行定义，无论声明还是定义都需要指出其形式参数 x 和 y，在对其进行调用时将实际参数 a 和 b 的值分别传送给形参 x 和 y，此时系统会根据函数声明检查二者是否匹配，即实参和形参的个数或者类型是否一致，若一致则经过执行 max 函数得到一个返回值，即变量 z 的值，再把这个值赋值给 c。最后再输出 c 的值，这个值就是较大值。

3.3 程序结构与效率

3.3.1 顺序结构

顺序结构就是一条一条地从上到下执行语句，每条语句都会被执行，执行过的语句不会再次执行。

【例 3-12】 一元二次方程式 $ax^2+bx+c=0$ 的根为 x_1，x_2，求它们的值，其中 a、b 和 c 的值由键盘输入。代码如下：

```
#include<iostream>
#include<cmath>                                    //使用头文件 cmath
using namespace std;
int main()
{
    float a,b,c,x1,x2;
    cin>>a>>b>>c;
    x1=(-b+sqrt(b*b-4*a*c))/(2*a);                 //这是求解 x1 的表达式语句
    x2=(-b-sqrt(b*b-4*a*c))/(2*a);                 //这是求解 x2 的表达式语句
    cout<<"a="<<a<<",b="<<b<<",c="<<c<<endl<<"x1="<<x1<<",x2="<<x2
    <<endl;
    return 0;
}
```

若输入 a、b、c 的值分别为 4.5、8.8 和 2.4，则运行结果如图 3-12 所示。

```
4.5 8.8 2.4
a=4.5,b=8.8,c=2.4
x1=-0.327612,x2=-1.62794

--------------------------------
Process exited after 17.94 seconds with return value 0
请按任意键继续. . .
```

图 3-12 运行结果

程序分析：由于程序中要用到数学函数 sqrt，所以需要使用头文件 cmath，也可以使用 C 语言的写法“math.h”。

顺序结构的程序在运行时，各条执行语句就是按顺序执行的。在定义了所需变量之后，进行输入操作，然后根据题目要求将数学表达式转换成合法的 C++ 表达式，再进行赋值操作，最后将两个值输出。

【例 3-13】 从键盘输入三角形的 3 条边长 a、b 和 c，计算三角形的面积 s。代码如下：

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int a,b,c;
    double p,s;
    cin>>a>>b>>c;
    cout<<"a="<<a<<",b="<<b<<",c="<<c<<endl;
    if (a<0||b<0||c<0)                          //判断边长是否都大于 0
    {
        cout<<"error"<<endl;
        return 1;
    }
    p=(a+b+c)/2;
    s=sqrt(p*(p-a)*(p-b)*(p-c));                //计算三角形面积的表达式语句
    cout<<"s="<<s<<endl;
    return 0;
}
```

若输入 a、b、c 的值分别为 6、8 和 10 则运行结果如图 3-13 所示。

```
6 8 10
a=6,b=8,c=10
s=24

--------------------------------
Process exited after 14.81 seconds with return value 0
请按任意键继续. . .
```

图 3-13　例 3-13 的运行结果

3.3.2　选择结构

选择结构就是根据条件来判断执行哪一条语句，如果给定的条件成立，就执行相应的语句，如果不成立，就执行其他的一些语句。理解选择结构，首先要了解 bool 类型（也叫 boolean 类型或布尔类型），它只有两个值，即真和假。判断表达式最终的值就是一个 bool 类型，这个判断表达式的 bool 值直接决定了选择结构该如何进行选择，循环结构如何进行循环。实现选择结构的语句有 if 语句和 switch 语句。

1. if 语句

if 单分支语句的格式如下：

```
if(bool 值 1)                /* 如果 bool 值 1 为真则执行代码段 1,否则不执行代码段 1,而继续
                                执行后面的程序 */
{
    代码段 1
}
…
```

if…else 双分支语句的格式如下：

```
if (bool 值 1)                //如果 bool 值 1 为真,则执行代码段 1,否则执行代码段 2
{
    代码段 1
}
else
{
    代码段 2
}
```

if…else if 多分支语句的格式如下：

```
if (bool 值 2)              //如果 bool 值 2 为真,则执行代码段 1,否则判断 bool 值 3 是否为真
{
    代码段 1
}
else if (bool 值 3)         //若 bool 值 3 为真则执行代码段 2,否则直接执行代码段 3
{
    代码段 2
}
else
{
    代码段 3
}
//开头的 if 和结尾的 else 都只能有一个,但是中间的 else if 可以有很多
```

2. switch 语句

switch 语句的格式如下：

```
switch(变量) //执行到这一句时,变量的值是已知的
{             /* switch case 语句执行时,会用该变量的值依次和各个 case 后的常数去对比,
                 找到第一个匹配项 */
    case 1:  //若变量的值是 1 则执行该 case 对应的代码段 1
    代码段 1;
    break;   //break 的作用是执行完当前 case 后跳出 switch 语句,不再执行后面其他 case
    case 2:
    代码段 2;
```