



# 第1章 C语言基础与顺序结构

## 学习目标

1. 掌握 C 语言程序的结构。
2. 合理选用数据类型。
3. 掌握算术运算符和赋值运算符。
4. 掌握控制数据的输入 / 输出方法。
5. 掌握顺序结构以及顺序结构的流程。
6. 掌握 Visual C++ 6.0 集成环境的使用方法。
7. 掌握转换大小写字母的方法。
8. 掌握产生随机整数的方法。

计算机由硬件系统和软件系统组成,其中硬件是物质基础,而软件是计算机的灵魂。没有软件的计算机是什么也干不了的“裸机”。所有软件要用计算机语言编写。

计算机语言是人和计算机交换信息的工具。随着计算机技术的发展,计算机语言逐步得到完善。最初使用的计算机语言是用一串串 0 和 1 组成的数字表达的语言——机器语言,后来使用的计算机语言是用简洁的英文字母或符号串表达的语言——汇编语言。机器语言和汇编语言都是低级语言,用这种语言编写的程序执行效率高,但程序代码很长,又都依赖于具体的计算机,因此编码、调试、阅读程序都很困难,通用性也差。

目前使用最广泛的计算机语言是用接近于人们自然语言表达的语言——高级语言。用高级语言编写的程序完全不依赖于计算机硬件,编码相对短,可读性强。C 语言属于高级语言。用高级语言编写的程序叫作源程序。

## 1.1 认识 C 语言程序

### 1.1.1 了解 C 语言程序的构成

任何一个 C 语言程序都是由若干个函数构成的,所以编写一个程序的过程,就是根据功能要求并按照 C 语言语法规则逐个编写各函数的过程。下面给出一个较完整的程序,以便读者尽早了解 C 语言程序。程序中的各行含义将在后续章节中详细介绍,在实例 6.1 中给出其执行过程。

**【实例 1.1】** 观察下面的程序,认识一个完整的 C 语言程序,了解 C 语言程序的结构。

```
// 下面 3 行是预处理命令部分
#include <stdio.h>
#include <math.h>
```

```
#define PI 3.14159

// 下面 2 行是函数的原型说明部分
double sup_area(double r);
double volume(double r);

// 下面是主函数部分
int main(void)
{   double a=-5,b,c,d;

    b=fabs(a);
    c=sup_area(b);
    d=volume(b);
    printf("c=%lf,d=%lf\n",c,d);
    return 0;
}

// 下面是 sup_area() 函数的定义部分,该函数的功能是计算球的表面积
double sup_area(double r)
{   double s;

    s=4*PI*r*r;
    return s;
}

// 下面是 volume() 函数的定义部分,函数功能是计算球的体积
double volume (double r)
{   double v;

    v=4.0/3.0*PI*r*r*r;
    return v;
}
```

## 1. 运行结果

```
c=314.159000,d=523.598333
```

## 2. 归纳分析

(1) 任何一个 C 语言程序都由若干个函数构成,而且必须有且仅有一个主函数(函数名必须是 main),其他函数的多少由实际情况而定,处理简单问题时也可以没有其他函数。本程序包括主函数 main()、sup\_area() 函数和 volume() 函数。各个函数完成各自的功能,分工协作完成预定任务。如同生活中,每个人完成各自的职责。

(2) 程序中用“//”引导的部分叫作注释部分。注释部分对程序的运行不起作用。



在程序中添加注释是为了便于阅读,让读者明晰程序设计思路。

(3) C语言提供了丰富的标准库函数以便直接使用,但要求在程序的开头加上包含该函数信息的命令行(参见附录D)。本程序中使用了库函数 `printf()` (功能是输出数据,是输出函数)和 `fabs()` (功能是求绝对值,是数学函数)。C语言系统将所有输入/输出函数的信息存放在 `stdio.h` 文件中,而将所有数学函数的信息存放在 `math.h` 文件中,所以程序的开头加了两个命令行 `#include <stdio.h>` 和 `#include <math.h>`。

(4) 程序的开头除了上述两个命令行外,又有了命令行 `#define PI 3.14159`。有此命令行后,程序中所用到的所有 `PI` 均用 `3.14159` 代替。

(5) 程序中编写了3个函数,除了主函数外,在使用其他自定义的函数前,应对这些函数逐一进行函数原型说明,因此程序中加了“`double sup_area(double r);`”和“`double volume(double r);`”。

### 1.1.2 熟悉主函数框架

在C语言中可以编写程序进行算术运算,就像日常生活中人们使用计算器计算一样。

**【实例 1.2】** 编写程序,计算两个整数的和与差,要求从键盘输入两个数。

#### 1. 编程思路

在C语言中,数据的输入操作使用标准库函数 `scanf()` 实现,而通过标准库函数 `printf()` 实现输出功能。加法操作和减法操作分别使用算术运算符 `+` 和 `-`。

#### 2. 程序代码

```
#include <stdio.h>
int main(void)
{   int x,y,a;                               // 定义三个变量

    printf("Input x and y:");               // 显示提示信息
    scanf("%d%d",&x,&y);                   // 要求从键盘输入两个整数
    a=x+y;                                   // 计算两个数的和
    printf("The sum of the two numbers:%d\n",a); // 输出两个数的和

    a=x-y;                                   // 计算两个数的差
    printf("The difference:%d\n",a);        // 输出两个数的差
    return 0;
}
```

#### 3. 运行结果

```
Input x and y:1200 180
The sum of the two numbers:1380
The difference:1020
```

#### 4. 归纳分析

(1) 本程序只包含主函数,主函数的一般框架是:

```
int main(void)
```

```
{ 定义变量部分  
  功能语句部分  
  return 0;  
}
```

定义变量和功能语句部分均可以是多条,而且每条都以“;”结束。

**说明:** C99 标准提供两种主函数形式,一种是不带参数的主函数,另一种是带参数的主函数,本书只介绍第一种主函数。

(2) 在屏幕上显示内容要使用 printf() 函数。此函数有两种形式,参见实例 1.8。

语句“printf("Input x and y:");”的功能是屏幕上显示字符串“Input x and y:”。而语句“printf("The sum of the two numbers:%d\n",a);”的功能是在显示双引号中字符的同时,%d 的位置上用 a 的值替换,其中,%d 是格式说明符,在输出整数时使用;\n 是换行符。

(3) 从键盘输入数据要使用 scanf() 函数。参见实例 1.8。

语句“scanf("%d%d",&x,&y);”的功能是要求用户从键盘输入两个整数,并把它们分别存放在变量 x 和 y 中。在输入整数时也要使用格式说明符 %d。

(4) 编写程序后应上机验证。学习者可以按照如下顺序尝试上机。更多的上机操作方法将在后续章节中陆续介绍。

## 5. 操作步骤

(1) 安装 Visual C++ 6.0。如果已经安装,则跳过此步。

(2) 启动 Visual C++ 6.0。为了编写 C 语言程序,首先应启动 Visual C++ 6.0 集成环境,其方法是:在 Windows 的“开始”菜单中,依次选择“程序”| Microsoft Visual Studio 6.0 | Microsoft Visual C++6.0,此时弹出 Visual C++ 6.0 的主窗口。

(3) 新建 C 源程序。在 Visual C++ 6.0 的主窗口选择“文件”|“新建”命令,出现“新建”对话框,在此选择“文件”选项卡下的 C++ Source File,在右侧“位置”栏中输入“D:\C 语言”或通过单击按钮选择 D 盘上的“C 语言”文件夹,再在“文件名”文本框中输入“实例 1\_2.c”(见图 1.1),然后单击“确定”按钮,便建立了新的 C 源程序,如图 1.2 所示。该界面的右侧为编辑窗口,这时的编辑窗口是空的。

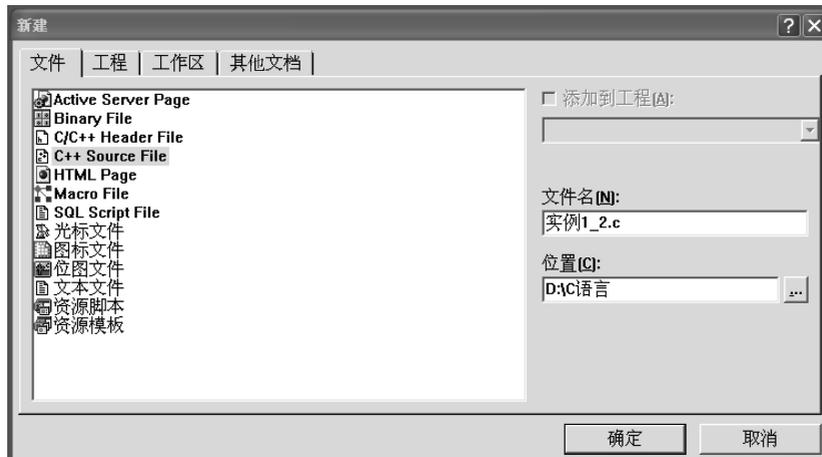


图 1.1 “新建”对话框

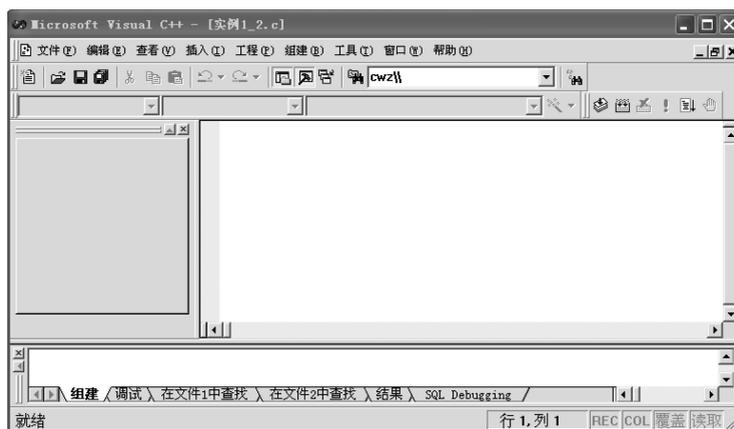


图 1.2 新建 C 源程序后的界面

(4) 编辑源程序。在编辑窗口中输入实例 1.2 的代码。可以不输入注释部分。

(5) 保存文件。因为上机时经常会发生预料不到的事情,一定要养成随时存盘的好习惯。单击工具栏中的“保存”按钮,按原名将文件存盘。

(6) 编译和连接程序。C 源程序通过编译和连接之后才能运行。单击“编译”按钮,进行程序的编译(在出现的提示信息框中单击“是”按钮)。系统在编译前会自动将程序保存,然后进行编译。如果在编译的过程中发现错误,将在下方窗口中列出所有错误和警告(附录 F 中提供编译错误的中英文对照表)。双击显示错误或警告的第 1 行,则光标自动跳到代码的错误行。修改该错误后,对程序重新进行编译,若程序还有错误或警告,可继续修改和编译,直到没有错误为止。编写程序往往不会第一次就很完美,一般都需要多次修正迭代,这就要求具有勇于尝试、不怕挫折的品质。编译本实例时没有出现任何错误和警告,所以错误和警告数都为 0,如图 1.3 所示。

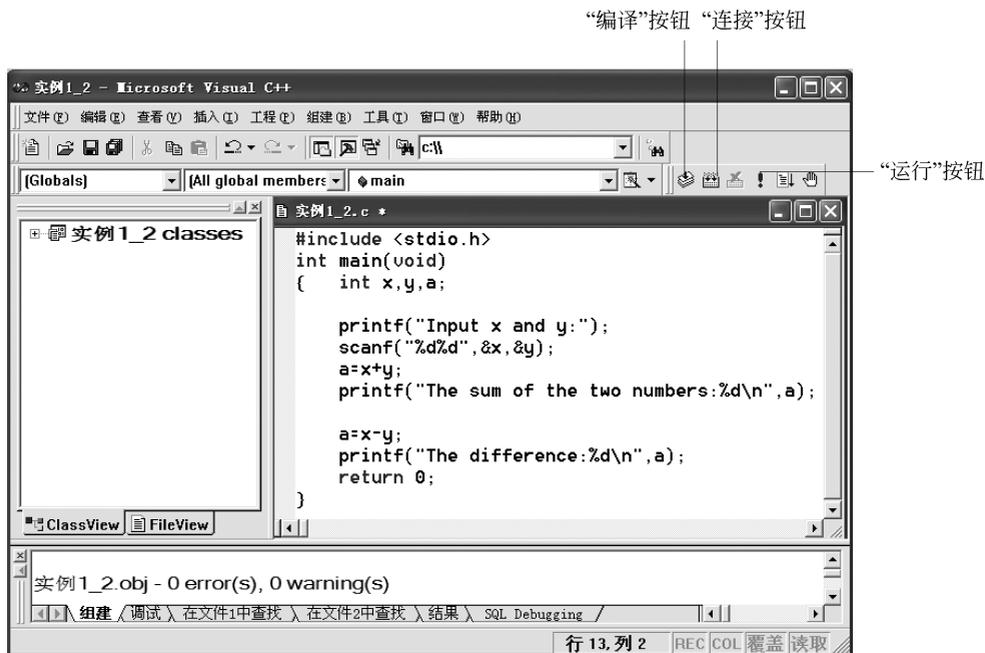


图 1.3 编译后的情况

如果在图 1.1 右侧“文件名”文本框中输入文件名时省略扩展名,则系统建立的文件不是 C 语言程序,而是 C++ 语言程序(默认扩展名为 .cpp)。

单击“连接”按钮后,与编译时一样,如果在连接的过程中系统发现错误,将在下面的窗口中列出所有错误和警告。修改错误后重新编译和连接,直到编译和连接都没有错误为止。

(7) 运行程序。单击“运行”按钮,出现如图 1.4 所示的界面,并要求用户输入  $x$  和  $y$  的值。输入 1200 和 180 后按 Enter 键,看到实例 1.2 的运行情况,如图 1.5 所示。运行程序时可以按 Shift 键切换中英文输入法。按任意键回到编辑窗口。

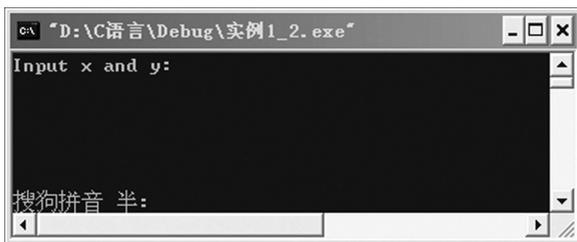


图 1.4 实例 1.2 的运行界面

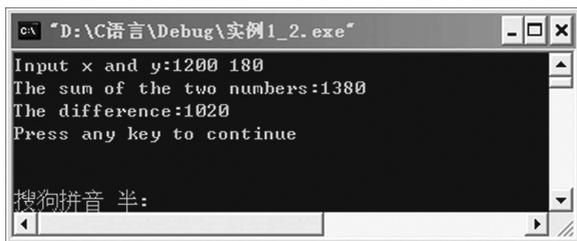


图 1.5 实例 1.2 的运行情况

如果运行结果与预期的结果不相符,则修改程序后重复第 6 步和第 7 步的操作,直到结果正确为止。

从以上操作步骤可以看到,要得到 C 程序的运行结果,首先选择一种集成环境,将源程序输入计算机内,然后把源程序翻译成机器能识别的目标程序,再把目标程序和系统提供的库函数等链接起来产生可执行文件,这时才可以运行程序。C 程序的编辑、编译、连接、运行过程如图 1.6 所示。



图 1.6 C 程序的编辑、编译、连接、运行过程

C 程序的编辑、编译、连接、运行过程可以在不同的环境中进行,本书选用的是 Visual C++ 6.0 集成环境。附录 E 中补充介绍了 Visual C++ 2010 环境。

如果退出 Visual C++ 6.0 环境后需要重新打开已建立的 C 程序实例 1\_2.c,则在资源管理器中双击“实例 1\_2.c”,或先启动 Visual C++ 6.0 环境后通过“文件”|“打开”菜单项打开文件。

## 1.2 合理选用数据类型

在编写 C 语言程序时,都需要处理大量的数据,其中最常用的数据类型为整型、实型、字符型,而常用的整型又有基本整型和长整型,常用的实型有单精度实型和双精度实型。不同类型的数据有不同的特性和处理方法,因此编写程序时合理选用相应的数据类型是至关重要的。

### 1.2.1 合理选用整型数据

在日常生活中我们经常需要处理整数,即不带小数点的数。例如,计算某人的年龄、统计学生人数等。在 C 语言中,常用基本整型(int)或长整型(long)变量存放整型数据。由于在 Visual C++ 6.0 环境中,基本整型和长整型所占字节数相同(4 字节),因此本书只介绍基本整型。定义变量时必须根据需要给出其类型。

**【实例 1.3】** 编写程序,计算正方形铁板的面积,铁板边长为 150,如图 1.7 所示。

#### 1. 编程思路

利用公式“面积 = 边长 × 边长”计算正方形的面积,而且定义变量 area 存放正方形的面积。由于需要处理的数据是整数,所以选用 int 型。



图 1.7 正方形铁板

#### 2. 程序代码

```
#include <stdio.h>
int main(void)
{   int area;                // 定义基本整型变量 area

    area=150;                // 把 150 赋给变量 area
    area=area*area;          // 计算正方形的面积
    printf("area=%d\n",area); // 输出正方形的面积
    return 0;
}
```

#### 3. 运行结果

**area=22500**

#### 4. 归纳分析

(1) 程序中使用了变量 area,所谓变量就是在程序执行过程中其值可以变化的量。C 语言将数据分为常量和变量,常量是在程序执行过程中其值永远不变的量。变量的实质是内存中的一个存储单元,因此在使用变量前应向系统申请存储单元,这一过程就是定义变量的过程。本程序用“int area;”定义了变量 area。

定义变量的一般形式如下：

类型名 变量名 1, 变量名 2, ..., 变量名 n;

变量名由编程者自己给出,但必须遵循如下变量名的命名规则。

- ① 变量名中只能出现数字、大小写英文字母和下划线。
- ② 变量名必须以字母或下划线开头。
- ③ 变量名与关键字不能相同。
- ④ 不提倡使用预定义字符。

C 语言区分大小写,所以变量 `area` 和 `Area` 是两个不同的变量。所谓关键字是 C 语言中已有特定用途的标识符 (参见附录 A),如 `int`、`float` 等。预定义符有:库函数名 (如 `printf`)、预编译处理命令 (如 `include`、`define`) 等。如果把预定义符作为变量名,则该预定义符将失去原有的含义。

**注意：**变量必须先定义后使用,定义变量后其中的值是不确定的。

变量如同容器,其中可以存放数据,也可以从中取出数据使用。例如,用 “`int area;`” 定义变量 `area` 后,变量 `area` 中的值是不确定的,但执行 “`area=150;`” 后, `area` 中的值变为 150,再执行 “`area=area*area;`” 后, `area` 中的值又变为 22500,如图 1.8 所示。

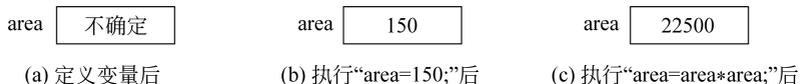


图 1.8 变量 `area` 值的变化情况

编写程序时经常使用形如 “`area=area*area;`” 的语句,其特点是运算符 “`=`” 两侧均出现相同的变量名,但两侧的变量名所代表的含义是不同的。例如,该语句中,运算符 “`=`” 右侧的 `area` 代表变量 `area` 中的值 (即 150),而左侧的 `area` 代表存储单元。该语句的执行过程是先计算 `area*area` 的值 (即  $150 \times 150$ ),再把计算结果 (即 22500) 存放在运算符 “`=`” 左侧的 `area` 中。在 C 语言中 “`=`” 是赋值运算符,参见 1.3.2 小节。

(2) 输出基本整型数据时在 `printf()` 函数中要使用格式说明符。

**【讨论题 1.1】** 在实例 1.3 的程序中,如果需要计算任意一个边长的正方形铁板面积,应该如何修改程序?

## 1.2.2 合理选用实型数据

在日常生活中经常需要处理实型数据。例如,计算学生的平均成绩、统计一年的销售额等。在 C 语言中,常用 `float` 型或 `double` 型变量存放实型数据。

**【实例 1.4】** 编写程序,计算半径为 15.67 的圆面积。要求分别使用单精度型和双精度型数据计算。

### 1. 编程思路

通过求圆面积公式  $\pi r^2$  计算。单精度实型变量用 `float` 定义,双精度实型变量用 `double` 定义。

## 2. 程序代码

```
#include <stdio.h>
#define PI 3.14159          // 程序中的所有PI 均用 3.14159 代替
int main(void)
{   float s1;
    double s2;

    s1=PI*15.67*15.67;
    s2=PI*15.67*15.67;
    printf("s1=%f,s2=%lf\n",s1,s2);
    return 0;
}
```

## 3. 运行结果

```
s1=771.413940,s2=771.413969
```

## 4. 归纳分析

(1) 在存储整型数据时没有误差,但存储实型数据时就有误差。例如,通过“a=5;”将5赋给整型变量a后,a中存放的值就是5,但通过“b=345.6789;”将345.6789赋给单精度实型变量b后,b中存放的值不是345.6789,而是带误差的值(如345.678894)。单精度实型数据的有效位是6~7位,它只保证前6~7位是正确的;而双精度实型数据的有效位是15~16位,它能够保证前15~16位是正确的。从本程序的运行结果中也可以观察。

(2) 在单精度实型和双精度实型中,通常按如下原则选择使用具体数据类型。

① 当要处理 $-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$ 的实数时,选用单精度实型(float)(也可以选用双精度实型,但会浪费存储单元,因为双精度实型占8字节,单精度实型只占4字节)。

② 当要处理 $-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$ 的实数时,选用双精度实型,不能选用单精度实型。

③ 当要处理的实型数据精度要求高时,选用双精度实型。

**注意：**在整型变量中不能存放实型数。若有“int a; a=3.6;”,则执行该程序段后,a中存放的值是3,而不是3.6。

(3) 输出单精度实型数据时在printf()函数中要使用格式说明符%f,而输出双精度实型数据时使用%lf。用%f或%lf输出数据时,小数点后总是输出六位,若改用%.3f或%.3lf形式,则小数点后只输出三位,若用%.0f或%.0lf形式,则不输出小数点后的数据。

**【讨论题 1.2】** 假设一个学生的总评成绩由平时成绩和期末成绩构成(平时、期末和总评成绩均为整数),其中平时成绩占总评成绩的40%,期末成绩占总评成绩的60%。

计算总评成绩时可按如下步骤进行。

① 输入平时成绩和期末成绩

计算成绩 1 :                    成绩 1= 平时成绩 × 0.4

计算成绩 2 :                    成绩 2= 期末成绩 × 0.6

② 计算总评成绩

总评成绩 = 成绩 1+ 成绩 2

③ 输出总评成绩

应需要定义哪些变量? 如何选择各变量类型?

### 1.2.3 合理选用字符型数据

在日常生活中经常遇到一串串字符,如姓名、性别、通信地址等,在 C 语言中也同样需要处理大量的字符数据,例如,打字练习时计算正确率、连接两篇文章等。单个字符(即字符常量)要用 char 型变量存放。在程序中将字符常量用单引号括起来,如 'a'、'B' 等。字符常量可以是 ASCII 码(参见附录 B)中的任意一个字符,它在内存中占一个字节。

**【实例 1.5】** 假设变量 ch 中已存放字母 'H',编写程序,将 ch 中的字母转换成小写字母后重新存放在该变量中。

#### 1. 编程思路

要处理字符,必须使用字符型类型,用 char 定义字符型变量。通过 'H'+32 可以得到字母 'H' 所对应的小写字母。

#### 2. 程序代码

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

    ch='H';
    ch=ch+32;
    printf("ch=%d,ch=%c\n",ch,ch);
    return 0;
}
```

#### 3. 运行结果

**ch=104,ch=h**

#### 4. 归纳分析

(1) 字符常量在内存中存放的是其 ASCII 码值,所以像整型数据一样可以直接参与算术运算。如大写字母 'H' 在内存中存的是其 ASCII 码值 72,因此 'H'+32 的值是 104,即为小写字母 'h' 的 ASCII 码值。实际上,所有大写字母和对应小写字母的 ASCII 码值都相差 32。

(2) 按字符形式输出字符型数据时,在 printf() 函数中要使用格式说明符 %c,在屏