3 变量、代数与计算机解题

教学目标

C/C ++ 程序的基本结构

变量的定义和使用

内容要点

程序的基本结构

程序说明部分

预编译命令

主函数

声明部分

执行部分

变量与数据类型

变量的基本概念

数据类型与变量的地址空间

定义变量和赋初值

变量赋值的 5 个要点

指针变量

指针定义与初始化

指针赋值

在讲述这一章之前,先举一个例子。

【任务 3.1】 王小二同学是一个聪明的孩子,他到超市去买东西,看到电子计价算账方便快捷,就想编程模拟操作一下。下面先请读者看程序,然后再做解释。

3.1 程序的基本结构

先来读任务 3.1 的程序，为了易于说明，程序清单的左边加注了表示行号的数字。

```
1  //*********************************************
2  //*  程 序 名:3_1.cpp                          *
3  //*  作    者:王小二                          *
4  //*  编制时间:2002 年 7 月 7 日                *
5  //*  主要功能:计算应付款                       *
6  //*********************************************
7  #include <iostream>                    // 预编译命令
8  using namespace std;
9  int main()                             // 主函数
10 {                                      // 主函数开始
11     float ApplePrice=3.5;              // 苹果单价,3.5 元/kg
12     float BananaPrice=4.2;             // 香蕉单价,4.2 元/kg
13     float AppleWeight=0.0;             // 苹果重量,初始化为 0
14     float BananaWeight=0.0;            // 香蕉重量,初始化为 0
15     float Total=0.0;                   // 总钱数,初始化为 0
16     cout << "请输入苹果重量" << endl;   // 提示信息
17     cin >> AppleWeight;                // 输入苹果重量
18     cout << "请输入香蕉重量"<< endl;    // 提示信息
19     cin >> BananaWeight;               // 输入香蕉重量
20     Total=ApplePrice * AppleWeight+BananaPrice * BananaWeight;
21                                        // 计算应付款
22     cout << "应付款" << Total << endl;  // 输出应付款
23     return 0;
24 }                                      // 主函数结束
```

1. 程序说明部分

写好的程序通常还会再修改，或与他人交流。由于程序语言与人类自然语言在表达上的差异，程序代码示意图不太容易从代码直接反映出来，有些重要信息（如作者、时间、目的等）也不属于源码性质，所以，一般要在源程序开头几行对程序作一个简要的说明。这些说明是以C++ 的注释形式出现的。所谓注释，是指计算机不对其进行计算的部分，在C++ 中以//符号引导的同一行内的文字都是注释。

在读程序时，首先要看程序说明，这件事十分重要。此程序的说明有 4 项：程序名、作者、编制时间、主要功能。在源程序清单中占 6 行，这 6 行纯粹是为了说明用的，不属于机器要计算的内容，因此，在每一行的前面冠以注释符号//。

2. 预编译命令

在第7行以#开头的是预编译命令。

```
#include <iostream>
```

意思是将程序库中的输入输出流文件作为头文件加入到现在要编写的程序中。

3. 主函数

从第9行到第24行是主函数。主函数是以main()为标识的,这是每一个程序都必须有的标识。在这个程序中,int main() 所起的作用仅是执行一系列的操作。

主函数 main() 的函数体由一对大括号{}括起,函数体包含两个部分:前面是声明部分,后面是执行部分。按规定声明在前,执行在后。不声明者,不能执行。

在任务3.1中声明了以下5项内容,依次在程序的第11～15行:

(1) 变量名 ApplePrice 是苹果单价,在其前的 float 是该变量所取的数据类型,3.2节将详细讲解。在其后的"=3.5"是赋初值给该变量,这里苹果的单价是每千克3.5元。

(2) 变量名 BananaPrice 是香蕉单价。

(3) 变量名 AppleWeight 是苹果重量。

(4) 变量名 BananaWeight 是香蕉重量。

(5) 变量名 Total 是总钱数。

声明部分之后是对5个变量的操作,即执行部分。

第16行和第18行是让屏幕显示提示信息,告诉程序的使用者下面准备用键盘输入苹果和香蕉的重量。这两条语句用 cout 输出流。

第17行和第19行是用 cin 输入流将键盘输入的实数分别赋给变量 AppleWeight(苹果重量)和 BananaWeight(香蕉重量)。

第20行用来计算应付款 Total,这是一条赋值语句,计算购买香蕉和苹果的钱,相加后将总钱数赋给 Total 变量。

第22行是输出语句,将应付款显示到屏幕上。第23行要有 return 0;

第10行与第24行所包含的一对大括号是主函数 main()所必需的,被这一对大括号所括起的语句,就是主函数的内容。任务3.1是一个完整的例子,在对变量有了一些初步的感性认识之后,下面再深入讲述有关变量的重要概念和一些特点。

3.2 变量与数据类型

3.2.1 变量的基本概念

变量是相对于常量而言的,在程序中经过操作其值允许改变的量称为变量。

变量在使用前必须加以定义,一般在程序的声明部分定义。

每一个变量要有一个与其他变量不相同的名字,称为变量名。

变量在计算机中需要占据存储空间，这些空间都有各自不同的内部编号。这些编号称为变量在计算机内存中存储单元的地址，简称变量的地址。

变量名的第一个字符必须是字母或下划线，其后的字符只能是字母、数字和下划线，且所用的名字不得与 C/C++ 语言系统所保留的关键字相同(见附录 B)。

建议

在给变量命名时应考虑实际含义，以便提高程序的易读性。比如任务 3.1 中的苹果单价用 ApplePrice。

3.2.2 数据类型与变量的地址空间

程序中的变量取什么数据类型是由工程任务的需要决定的。C/C++ 中的数据类型可分为以下两大类：第一类是基本数据类型，包括整型、浮点型和字符型；第二类是构造数据类型，包括数组、结构、联合、枚举等。所谓构造数据类型，是指这种类型的数据是由若干个基本数据类型的变量按特定的规律组合构造而成的。

各种数据所能表示的数据精度不同，因而它所占用的内存空间的大小不同。下面仅就基本数据类型来分析所能表示的数的精度和所占用的内存空间。

基本数据类型有：

(1) 整型。即整数类型，它又可分为 4 种。

① int 整型　占用 4 字节，数的表示范围是 −2 147 483 648～2 147 483 647。

② unsigned int 无符号整型　占用 4 字节，数的表示范围是 0～4 294 967 295。

③ long int 长整型　占用 4 字节，数的表示范围是 −2 147 483 648～2 147 483 647。

④ unsigned long int 无符号长整型　占用 4 字节，数的表示范围是 0～4 294 967 295。

(2) 实型。即实数类型，它又可分为 3 种。

① float 浮点型　占用 4 字节，数的表示范围是 $-3.4\times10^{38}\sim-2.2\times10^{-38}$，$1.2\times10^{-38}\sim3.4\times10^{38}$，有效位为 7 位。

② double 双精度型　占用 8 字节，数的表示范围是 $-1.7\times10^{308}\sim-2.2\times10^{-308}$，$2.2\times10^{-308}\sim1.7\times10^{308}$，有效位为 15 位。

③ long double 长双精度型　占用 16 字节，数的表示范围是 $-1.2\times10^{4932}\sim-3.3\times10^{-4932}$，$3.3\times10^{-4932}\sim1.2\times10^{4932}$，有效位为 19 位。

(3) bool 逻辑型。占用 1 字节。

(4) char 字符型。占用 1 字节。

3.3 定义变量和赋初值

在主函数 main() 中的声明部分要对一些变量进行定义，提出合适的精度要求，指出这些变量的数据类型，目的是为变量分配内存单元并赋以初始值。比如定义变量名为 a

的整型变量,程序写成:

```
int a=30;
```

系统会根据这个精度要求,安排4个字节的内存单元存放"a"变量的整数值,见图3.1。在图中变量名"a"是这个内存单元的符号地址。在本节的学习中建立起变量与变量地址的概念会对以后的学习大有用处。一讲到变量就要想到有一个地址与之联系。

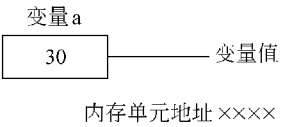

图3.1 变量的定义和内存地址的关系

为了更清楚地讲清变量内存单元的大小与存储单元地址的关系,我们来看如下的示例程序:

```
//*****************************************
//*程 序 名:3_2.cpp                        *
//*作    者:王小二                        *
//*编制时间:2010年5月                      *
//*主要功能:测试变量取地址操作             *
//*****************************************

#include <iostream>
using namespace std;

int main()
{
    int i;
    long m;
    float f;
    double d;
    cout <<"sizeof(int)=" <<sizeof (int)<<", sizeof(i)="<<sizeof (i)<<",&i="<<&i<<endl;
    cout <<"sizeof(long)=" <<sizeof (long)<<", sizeof(m)="<<sizeof (m)<<",&m="<<&m<<endl;
    cout <<"sizeof(float)=" <<sizeof (float)<<", sizeof(f)="<<sizeof (f)<<",&f="<<&f<<endl;
    cout <<"sizeof(double)=" <<sizeof (double)<<", sizeof(d)="<<sizeof (d)<<",&d="<<&d<<endl;
    return 0;
}
```

在上面的程序中,"&"表示取变量地址的操作符,运算结果是返回变量在内存中存储

单元的位置，即地址值。

程序运行结果如下：

```
sizeof(int)=4,sizeof(i)=4,&i=0x7fff5fbffa9c
sizeof(long)=8,sizeof(m)=8,&m=0x7fff5fbffa90
sizeof(float)=4,sizeof(f)=4,&f=0x7fff5fbffa98
sizeof(double)=8,sizeof(d)=8,&d=0x7fff5fbffa88
```

在上面的运行结果中，地址是通过 cout 输出到屏幕的，通常是以十六进制的格式表示的。

3.4 变 量 赋 值

给变量赋值是一个非常重要的概念。

3.4.1 赋值符号与赋值表达式

在 C/C++ 中赋值符号为“=”，赋值表达式的一般格式为

<变量>=<表达式>

例如

```
PI=3.14159;                    //读作将表达式的值 3.14159 赋给变量 PI
C=sin(PI/4);                   //读作将表达式 π/4 的正弦函数值赋给变量 C
```

3.4.2 变量赋值的 5 要素

给变量赋值时要注意以下 5 点：

(1) 变量必须先定义再使用。

(2) 在变量定义时就赋初值，是好的编程习惯。

(3) 对变量的赋值过程是“覆盖”过程。所谓“覆盖”是在变量地址单元中用新值去替换旧值。

(4) 读出变量的值后，该变量保持不变，相当于从中复制一份出来。

(5) 参与表达式运算的所有变量都保持原来的值不变。

下面举例说明上述特点。

```
#include <iostream>
using namespace std;
int main()
```

```
{                                    //主程序开始
int a=0, b=0, c=0;                   // 声明 a,b,c 为整型变量
                                     // 均初始化为 0
a=7;                                 // a 赋值为 7,覆盖了原来的 0
b=a;                                 // b 赋值为 a,a 中的值覆盖了 b 中的值
                                     // 但 a 中的值不变
c=a+b;                               // 将 a+b 的值赋给 c,a+b 的值为 14
                                     // 去覆盖 c 中的 0,a 与 b 保持 7 不变
a=a+1;                               // 将 a+1 的值赋给 a,a+1 的值为 8
                                     // 覆盖了原来的 7
cout≪a≪''≪b≪''≪c≪endl;
return 0;
}                                    //主程序结束
```

上例中,a=a+1 可简化写作 a ++ ,图 3.2 说明了这 5 条语句的执行过程。

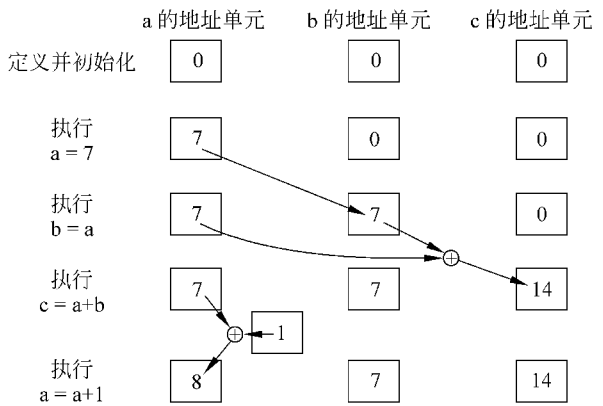

图 3.2　变量赋值过程

3.5　指针变量

指针是 C 语言中的一个重要概念。掌握指针的用法,可使程序简洁、高效、灵活。指针看似复杂,但并不难学。

为了了解什么是指针,先看一个小故事。

地下工作者阿金接到上级指令,要去寻找打开密电码的密钥,这是一个整数。几经周折,才探知如下线索:密钥藏在一栋 3 年前就被贴上封条的小楼中。一个风雨交加的夜晚,阿金潜入了小楼,房间很多,不知该进哪一间,正在一筹莫展之际,忽然走廊上的电话

铃声响起。艺高人胆大，阿金毫不迟疑，抓起听筒。只听一个陌生人说："去打开 211 房间，那里有线索。"阿金疾步上楼，打开 211 房间，用电筒一照，只见桌上赫然 6 个大字：地址 1000。阿金眼睛一亮，迅速找到 1000 房间，取出重要数据 66，完成了任务。

可用图 3.3 来描述这几个数据之间的关系。

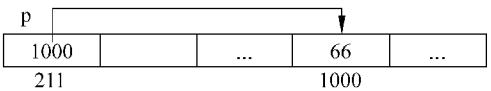

图 3.3　数据存放

说明

(1) 数据藏在一个内存地址单元中，地址是 1000。

(2) 地址 1000 又由 p 单元所指认，p 单元的地址为 211。

(3) 66 的直接地址是 1000；66 的间接地址是 211；211 中存的是直接地址 1000。

(4) 称 p 为指针变量，1000 是指针变量的值，实际上是有用数据在存储器中的地址。指针变量就是用来存放另一变量地址的变量(变量的指针就是变量的地址)。

3.5.1　指针定义与初始化

指针是一种特殊的变量，特殊性表现在类型和值上。从变量角度看，指针也具有变量的 3 个要素：

(1) 变量名，这与一般变量取名相同，由英文字符开始。

(2) 指针变量的类型，是指针所指向的变量的类型，而不是自身的类型。

(3) 指针的值是某个变量在内存中的地址，简称变量的内存地址。

例如下面的语句是定义一个名为 p 的指针，该指针指向一个整数类型的变量，且被初始化为 NULL。

```
int * p=NULL;
```

一旦指针 p 被定义，系统会为 p 分配一个内存单元，该单元的地址可以用 &p 表示(符号 &p 表示 p 的地址)，如图 3.4 所示。

在 p 中赋予一个符号化的常量 NULL，称之为将指针 p 初始化为 0。这个符号化的常量 NULL 是在头文件<iostream>中定义了的。将指针初始化为 NULL 等于将指针初始化为 0，在这里整数 0 是 C/C++ 系统惟一一个允许赋给指针类型变量的整数值。除 0 以外的整数值是不允许赋给指

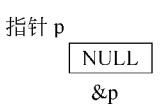

图 3.4　指针 p 被分配一个内存单元

针变量的,因为指针变量的数据类型是内存的地址,而不是任何整数。要记住:值为NULL的指针不指向任何变量。在定义时让指针初始化为NULL可以防止其指向任何未知的内存区域,以避免产生难以预料的错误发生。定义指针并将其初始化为NULL是一个值得提倡的好习惯。

3.5.2 指针赋值

前已述及指针变量是一个特殊的变量,其值是内存的地址,给指针赋值,就是将一个内存地址装入指针变量,这件事一做完就意味着指针指向了该内存地址。请看下例:

```
int a=66;                          // 定义一个整型变量a,并将其初始化为66
int *p=NULL, *q=NULL;              // 定义p,q为指向整型变量的指针变量
                                   // 并初始化为0
p=&a;                              // 将变量a地址赋给p
q=p;                               // 将p的值赋给q
```

图3.5中,当a变量的地址赋给指针p,意味着让指针p指向a。

图3.6中,当执行q=p后,p中所存的a变量的地址值也就被放到q变量中,意味着让指针q也指向a。这里用到了一个取地址运算符&,&a表示取变量a所在的内存的地址。

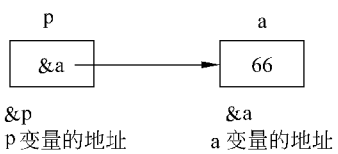

图3.5 指针赋值

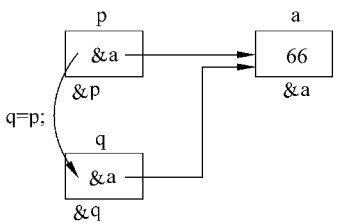

图3.6 赋值

3.5.3 在赋值语句中使用间接访问运算符

下面给出一个程序,在程序中定义两个指针变量p和q,在第14行和第15行,使用取地址运算符将整型变量akey的地址赋给p,b的地址赋给q,即让p和q分别指向akey和b。出现在第16行语句中的*p和*q代表指针所指向的变量单元中的内容。换句话说*p和akey等价,*q与b等价。赋值表达式

```
*q=*p;
```

等价于

b=akey;

```
// ****************************************
// *   程 序 名：3_3.cpp                  *
// *   作     者：wuwh                   *
// *   编制时间：2002 年 11 月 20 日      *
// *   主要功能：指针(使用间接访问运算符) *
// ****************************************
#include <iostream>                    // 预编译命令
using namespace std;
int main()                             // 主函数
{                                      // 函数体开始
    int akey=0,b=0;                    // 定义整型变量
    int *p=NULL,*q=NULL;               // 定义指针变量
    akey=66;                           // 赋值给变量 akey
    p=&akey;                           // 赋值给指针变量 p,让 p 指向变量 akey
    q=&b;                              // 赋值给指针变量 q,让 q 指向变量 b
    *q=*p;                             // 将 p 所指向的 akey 的值赋给 q 所指向的变量 b
    cout<<"b="<<b<<endl;               // 输出 b 的值
    cout<<"*q="<<*q<<endl;             // 输出 b 的值
    return 0;
}                                      // 函数体结束
```

程序操作如图 3.7 所示。

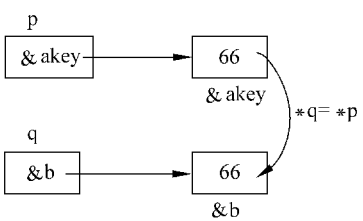

图 3.7　程序 3_3 说明

3.6　小　　结

(1) 掌握变量的概念和变量赋值的 5 个要素，对于程序设计是十分重要的。

(2) 比较第 2 章与第 3 章的内容可见，仅仅能够将计算机当作功能强大的计算器是远远不够的，只是使用了输出流 cout 和一些数学函数，基本上是算术运算。当引入变量