# 7.2 函数参数和按值传递

C++通常按值传递参数，这意味着将数值参数传递给函数，而后者将其赋给一个新的变量，例如，程序清单7.2包含下面的函数调用：

double volume = cube(side);

其中，side是一个变量，在前面的程序运行中，其值为5。cube()的函数头如下：

double cube (double x)

被调用时，该函数将创建一个新的名为x的double变量，并将其切始化为5。这样，cube()执行的操作将不会影响main()中的数据，因为cube()使用的是side的副本，而不是原来的数据。稍后将介绍一个实现这种保护的例子。用于接收传递值的变量放称为**形参**。传递给函数的值被称为**实参**。出于简化的目的，C++标准使用**参数（argument）**来表示实参，使用**参量（parameter）**来表示形参，因此参数传递将参数赋给参量。

在函数中声明的变量（包括参数）是该函数私有的。在函数被调用时，计算机将为这些变量分配内存；在函数结束时，计算机将释放这些变量使用的内存。这样的变量被称为局部变量，因为它们被限制在函数中，这样做有助于确保数据的完整性。还意味着，如果在main()中声明了一个名为x的变量，同时在另一个函数中也声明了一个名为x的变量，则它们将是两个完全不同的、毫无关系的变量。这样的变最也被称为自动变量，因为它们是在程序执行过程中自动被分配和释放的。

## 7.2.1 多个参数

函数可以有多个参数。在调用函数时，只需使用逗号将这些参数分开即可：

n\_chars(‘R’,25);

上述函数调用将两个参数传递给函数n\_chars()，我们将稍后定义该函数。

同样，在定义函数时，也在函数头中使用由逗号分隔的参数声明列表：

void n\_chars(char c,int n) // 两个参数

该函数头指出，函数n\_char()接受一个char参数和一个int参数。传递给函数的值被赋给参数c和n。

如果函数的两个参数的类型相同，则必须分别指定每个参数的类型，而不能像声明常规变量那样，将声明组合在一起：

void fifi(float a,float b) // 分别制定每个参数的类型

void fufu(float a,b) // 不被接受

和其他函数一样，只需添加分号就可以得到该函数的原型：

void n\_chars(char c,int n); // 第一种风格的函数原型

和一个参数的情况一样，原型中的变量名不必与定义中的变量名相同，而且可以省略：

void n\_chars(char,int); // 第二种风格的函数原型

然而，提供变量名将使原型更容易理解，尤其是两个参数的类型相同时。这样，变量名可以提醒参量和参数间的对应关系：

double melon\_density(double weight,double volume);

程序清单7.3演示了一个接受两个参数的函数，它还表明，在函数中修改形参的值不会影响调用程序中的数据。

**程序清单 7.3 twoarg.cpp**

// twoarg.cpp -- 拥有两个参数的函数

#include <iostream>

using namespace std;

void n\_chars(char, int);

int main()

{

 int times;

 char ch;

 cout << "Enter a character: ";

 cin >> ch;

 while (ch != 'q') // 按q退出

 {

 cout << "Enter an integer: ";

 cin >> times;

 n\_chars(ch, times); // 有两个参数的函数

 cout << "\nEnter another character or press the"

 " q-key to quit: ";

 cin >> ch;

 }

 cout << "The value of times is " << times << ".\n";

 cout << "Bye\n";

 cin.get();

 cin.get();

 return 0;

}

void n\_chars(char c, int n) // 输出n个c

{

 while (n-- > 0) // 一直执行直至n到0

 cout << c;

}

在程序清单7.3的程序中，将编译指令using放在函数定义的前面，而不是函数中。下面是该程序的运行情况：

Enter a character:W

Enter an integer:50

WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW

Enter another character or press the q-key to quit:a

Enter an integer: 20

aaaaaaaaaaaaaaaaaaaa

Enter another character or press the q-key to quit: q

The value of times is 20.

Bye

### 程序说明

程序清单7.3中的main()函数使用一个while循环提供重复输入，它使用cin>>ch，而不是cin.gct(ch)或ch = cin.get()来读取一个字符。这样做是有原因的。前面讲过，这两个cin.get()函数读取所有的输入字符，包括空格和换行符，而cin>>跳过空格和换行符。当用户对程序提示作出响应时，必须在每行的最后按Enter键，以生成换行符。cin>>ch方法可以轻松地跳过这些换行符，但当输入的下一个字符为数字时，cin.get()将读取后面的换行符。可以通过编程来避开这种麻烦，但比较简便的方法是像该程序那样使用cin。

n\_char()函数接受两个参数：一个是字符c，另一个是整数n。然后，它使用循环来显示该字符，显示次数为n：

while (n-- > 0) // 一直执行直至n到0

 cout << c;

程序通过将n变量递减来计数，其中n是参数列表的形参，main()中times变量的值被赋给该变量。然后，while循环将n递减到0，但前面的运行情况表明，修改n的值对times没有影响。即使您在函数main()中使用名称n而不是times，在函数n\_chars()中修改n的值时，也不会影响函数main()中n的值。

## 7.2.2 另外一个接受两个参数的函数

下面创建另一个功能更强大的函数，它执行重要的计算任务。另外，该函数将演示局部变量的用法，而不是形参的用法。

目前，美国许多州都采用某种纸牌游戏的形式来发行彩票，让参与者从卡片中选择一定数目的选项。例如，从51个数字中选取6个。随后，彩票管理者将随机抽取6个数。如果参与者选择的数字与这6个完全相同，将赢得大约几百万美元的奖金。我们的函数将计算中奖的几率。

首先，需要一个公式。假设必须从51个数中选取6个，而获奖的概率为1/R，则R的计算公式如下：

R＝

选择6个数时，分母为前6个整数的乘积或6的阶乘。分子也是6个连续整数的乘积，从51开始，依次减１。推而广之，如果从numbers个数中选取picks个数，则分母是picks的阶乘，分子为numbers开始向前的picks个整数的乘积。可以用for循环进行计算：

long double result = 1.0;

for(n = numbers,p = picks;p > 0 ; n--,p--)

 result = result\*n/p;

循环不是首先将所有的分子项相乘，而是首先将1.0与第一个分子项相乘，然后除以第一个分母项。然后下一轮循环乘以第二个分子项，并除以第二个分母项。这样得到的乘积将比先进行乘法运算得到的小。例如，对于(10\*9)/(2\*1)和(10/2)\*(9,1)，前者将计算90/2，得到45，后者将计算为5\*9，得到45。这两种方法得到的结果相同，但前者的中间值（90）大于后者。因子越多，中间值的差别就越大。当数字非常大时，这种交替进行乘除运算的策略可以防止中间结果超出最大的浮点数。

程序清单7.4在probability()函数中使用了这个公式。由于选择的数目和总数同都为正，因此该程序将这些变量声明为unsigned int类型（简称unsigned）。将若干整数相乘可以得到相当大的结果，因此lotto.cpp将该函数的返回值声明为long double类型。另外，如果使用整型，则像49/6这样的运算将出现舍入误差。

**注意**：某些C++实现不支持long 笼类型，如果所用的C++实现是这样的，请使用double类型。

**程序清单7.4 lotto.cpp**

// lotto.cpp -- 中奖概率

#include <iostream>

// 注意：某些C++实现需要用double替代long double

long double probability(unsigned numbers, unsigned picks);

int main()

{

 using namespace std;

 double total, choices;

 cout << "Enter the total number of choices on the game card and\n"

 "the number of picks allowed:\n";

 while ((cin >> total >> choices) && choices <= total)

 {

 cout << "You have one chance in ";

 cout << probability(total, choices); // compute the odds

 cout << " of winning.\n";

 cout << "Next two numbers (q to quit): ";

 }

 cout << "bye\n";

 cin.get();

 cin.get();

 return 0;

}

// 下面的函数计算从number个数中选取picks个数的几率

long double probability(unsigned numbers, unsigned picks)

{

 long double result = 1.0; // 下面是一些局部变量

 long double n;

 unsigned p;

 for (n = numbers, p = picks; p > 0; n--, p--)

 result = result \* n / p ;

 return result;

}

下面是该程序的运行情况

Enter the total number of choices on the game card and the number of picks allowed:

49 6

YOU have one chance in l.39838e+007 of winning.

Next two numbers (q to quit): 51 6

You have one chance in 1.80095e+007 of winning.

Next two numbers （q to quit) : 38 6

You have one chance in 2.76068e+006 of winning.

Next two numbers (q to quit) : q

bye

请注意，增加游戏卡中可供选样的数字数目，获奖的可能性将急剧降低。

### 程序说明

程序清单7.4中的probability()函数演示了可以在函数中使用的两种局部变量。首先是形参（number和picks），这是在左括号前面的函数头中声明的；其次是其他局部变量（result、n和p），它们是在将函数定义括起的括号内声明的。形参与其他局部变最的主要区别是，形参从调用probability()的函数那里获得自己的值，而其他变量是从函数中获得自己的值。