小马加编信息学教案(十五)

C++ 函数(二)

一. 课程内容

1. 函数的参数
2. 函数的参数传递
3. 函数的形参与实参
4. 全局变量与局部变量

二. 知识讲解

1. 函数的参数

函数的参数概念

- 如果把函数比喻成一台机器，那么参数就是原材料，返回值就是最终产品；从一定程度上讲，**函数的作用就是根据不同的参数产生不同的返回值**。

> 参数是函数与函数之间实现通信的数据“接口”。
> 函数调用的过程就是调用者带着实际参数（如果有）执行函数，将实际参数“传递”给形式参数，执行完函数体后再将计算得到的返回值传递给调用者（如果有）。

> 在未调用函数前，函数中的形式参数并不分配内存空间。
> 只有在被调用执行时，才被分配临时存储空间。
> 函数调用结束后，形式参数的内存空间将被操作系统立刻收回。

2. 函数的参数传递

函数的参数传递的"一致要求"

> **实际参数可以是任何符合形式参数类型的常量、变量、表达式**。
> 函数参数传递的过程就是实际参数和形式参数相结合的过程，必须遵守三个一致。即:

> (1) **个数一致**。
> (2) **顺序一致**。
> (3) **类型一致**。

3. 函数的形参与实参

- 形参（形式参数）

> 在函数定义中出现的参数可以看做是一个占位符，它没有数据，只能等到函数被调用时接收传递进来的数据，所以称为形式参数，简称形参。

- 实参（实际参数）

> 函数被调用时给出的参数包含了实实在在的数据，会被函数内部的代码使用，所以称为实际参数，简称实参。

> **形参和实参的功能是传递数据，发生函数调用时，实参的值会传递给形参**

- 形参和实参的区别和联系

> (1) 形参变量只有在函数被调用时才会分配内存，调用结束后，立刻释放内存，所以形参变量只有在函数内部有效，不能在函数外部使用(即**局部变量**)。

> (2) 实参可以是常量、变量、表达式、函数等，无论实参是何种类型的数据，在进行函数调用时，它们都必须有确定的值，以便把这些值传送给形参，所以应该提前用赋值、输入等办法使实参获得确定值。

> (3) 实参和形参在数量上、类型上、顺序上必须严格一致，否则会发生“类型不匹配”的错误。当然，如果能够进行自动类型转换，或者进行了强制类型转换，那么实参类型也可以不同于形参类型。

> (4) **函数调用中发生的数据传递是单向的，只能把实参的值传递给形参，而不能把形参的值反向地传递给实参**；换句话说，一旦完成数据的传递，实参和形参就再也没有瓜葛了，所以，在函数调用过程中，形参的值发生改变并不会影响实参。

```
#include <bits/stdc++.h>

//计算从m加到n的值
int sum(int m, int n) {
    int i;
    for (i = m+1; i <= n; ++i) {
        m += i;
    }
    return m;
}

int main() {
    int a, b, total;
    printf("Input two numbers: ");
    a = 1;  b = 100;
    total = sum(a, b);
    printf("a=%d, b=%d\n", a, b);
    printf("total=%d\n", total);
    return 0;
}
```

> 运行结果

```
a=1, b=100
total=5050
```

> 代码分析
> 在这段代码中，函数定义处的 m、n 是形参，函数调用处的 a、b 是实参。

> 从运行情况看，输入 a 值为 1，即实参 a 的值为 1，把这个值传递给函数 $sum()$ 后，形参 m 的初始值也为 1，在函数执行过程中，形参 m 的值变为 5050。函数运行结束后，输出实参 a 的值仍为 1，可见实参的值不会随形参的变化而变化。

> (5) **形参和实参虽然可以同名，但它们之间是相互独立的，互不影响，因为实参在函数外部有效，而形参在函数内部有效。**

4. 全局变量与局部变量

> 在前面的学习中，我们已经涉及到一些全局变量与局部变量的概念。
> 现在，我们来具体了解一些全局变量与局部变量的定义与内涵。

4.1 作用域

> 所谓作用域，就是变量的有效范围，就是变量可以在哪个范围以内使用。
> 有些变量可以在所有代码文件中使用，有些变量只能在当前的文件中使用，有些变量只能在函数内部使用，有些变量只能在 `for` 循环内部使用。

> 变量的作用域由变量的定义位置决定，在不同位置定义的变量，它的作用域是不一样的。

4.2 全局变量

> **全局变量是指定义在任何函数之外的变量，也就是不被任何“｛函数体｝”所包含，可以被源文件中其他函数所共用，用静态数据区存储，作用域（有效范围）是从定义变量的位置开始到源文件（整个程序）结束。**

> 全局变量在不初始化的情况下，默认为0。

4.3 局部变量

> **局部变量是指在一个函数（包括 main 函数）内部定义的变量，它只在本函数内部有效，其他函数不能使用这些变量，用动态数据区存储，函数的参数也是局部变量。**

> 局部变量在不初始化的情况下，值为随机值。

4.4 全局变量与局部变量的应用

> 指出下面的变量中，哪些是全局变量，哪些是局部变量。以及其对应的作用域。

```
int x, y;
float a, b;
float find(int c, int d) {
    float e, f;
    int i, j;
    …
}

int z;

void doit() {
    …
}
int main() {
    int g, h;
    …
}
```

三. 经典例题

1. 阅读以下程序，写出运行结果，并上机编程检验结果

```
#include <bits/stdc++.h>
using namespace std;

void swap(int x, int y) {
    int temp;
    temp = x;
    x = y;
    y = temp;
    cout << x << " " << y << endl;
 }
int main(){
    int a = 10, b = 50;
    swap(a, b);
    cout << a << " " << b << endl;
    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;

int x = 10, y = 15;

void change(int a, int b, int x){
    int temp;
    x++;    y++;
    temp = a;
    a = b;
    b = temp;
}

int main() {
    int a = 3, b = 5;
    cout << x << “ ” << y << “ ” << a << “ ” << b << endl;
    change(a,b,x);
    cout << x << “ ” << y << “ ” << a << “ ” << b << endl;
    return 0;
}
```

2. 找出程序中的错误。如果去掉错误，程序输出什么?上机检验你的结果

```
#include <bits/stdc++.h>
using namespace std;

int f() {
    int b = 0,c = 1;
    b = b + 1;
    c = c + 1;
    return (b+c);
}

int main() {
    for(int i = 1; i < 4; i++)
         cout << i <<  “ .sum= ” << f() << endl;
    cout <<  “ b= ” << b <<  “ c= ” << c << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;

int x = 233;

int main() {
    int x;
    cin >> x;
    for(int i = 1; i <= x; ++i){
         int x, y;
         cin >> x >> y;
         cout << x + y << endl;
     }
     int x = 5;
     cout << x << endl;
     cout << i <<  “ “ << y << endl;
     return 0;
}
```

3. 打印字符三角形

> 编写一个函数 `print (n,ch)`，表示打印一行 n 个英文字母 ch，并换行。然后，在函数 `main()` 中输入 n 和 ch，调用函数 `print()` 打印一个字符三角形。

> 输入格式
> 一行一个整数 n 和一个英文字母 ch，之间用一个空格隔开。

> 输出格式
> n 行，第 i 行有 i 个字母 ch。

样例输入	样例输出

样例输入	样例输出
3 a	a aa aaa

> 数据范围
> $1 \leq n \leq 20$

四. 提高巩固

1. 轰炸

> 一个大小为$n * m$的城市遭到了X次轰炸，每次都炸了一个每条边都与边界平行的矩形。
> 在轰炸后，有 y 个关键点，指挥官想知道，它们有没有受到过轰炸，如果有，被炸了几次，最后一次是第几轮。

> 输入格式
> 第一行，四个整数：n、m、x、y。
> 以下x行，每行四个整数：x_1、y_1、x_2、y_2，表示被轰炸的矩形的左上角坐标和右下角坐标（比如1 3 7 10就表示被轰炸的地方是从 $(1,3)$ 到 $(7,10)$ 的矩形）。
> 再以下 y 行，每行两个整数，表示这个关键点的坐标。

> 输出格式
> 共 y 行，
> 每行第一个字符为 Y 或 N ，表示是否被轰炸。
> 若为 Y ，在一个空格后为两个整数，表示被炸了几次和最后一次是第几轮。

样例输入	样例输出
10 10 2 3 1 1 5 5 5 5 10 10 3 2 5 5 7 1	Y 1 1 Y 2 2 N

> 数据范围
> $1 \leq N, M \leq 100$

2. 自幂数

> 有一种神奇的数，叫做自幂数”。自幂数是指一个 n 位数，它的每个位上的数字的 n 次幂之和等于它本身。
> 153 就是其中一个自幂数，因为 153 是3位数，且 $1^3+5^3+3^3=153$ 。
> 现在给你 n ，求出所有 n 位的数中的自幂数。

> 输入格式
> 一个正整数 n

> 输出格式
> 按照升序依次输出含 n 位的自幂数

样例输入	样例输出
3	153 370 371 407

3.【Mc生存】插火把

> 话说有一天 $PowderHan$ 在 Mc 开了一个超平坦世界，他把这个世界看成一个 $n*n$ 的方阵，现在他有 m 个火把和 k 个萤石，分别放在 $(x_1,y_1)(x_2,y_2)...(x_m,y_m)$ 和 $(o_1,p_1)(o_2,p_2)...(o_k,p_k)$ 的位置，问在这个方阵中有几个点会生成怪物？（没有光或没放东西的地方会生成怪物）
>
> 火把的照亮范围是：
> |暗|暗| 光 |暗|暗|
> |暗|光| 光 |光|暗|
> |光|光|火把|光|光|
> |暗|光| 光 |光|暗|
> |暗|暗| 光 |暗|暗|
>
> 萤石：
> |光|光| 光 |光|光|
> |光|光| 光 |光|光|
> |光|光|萤石 |光|光|
> |光|光| 光 |光|光|
> |光|光| 光 |光|光|

> 输入格式
> 输入共 $m+k+1$ 行。
> 第一行为n,m,k。
> 第2到第$m+1$行分别是火把的位置(x_i,y_i)。
> 第m+2到第$m+k+1$行分别是萤石的位置(o_i,p_i)。

> 注：可能没有萤石，但一定有火把。
> 所有数据保证在int范围内。

> 输出格式
> 输出一个正整数，表示有几个点会生出怪物。

样例输入	样例输出
5 1 0 3 3	12