12.2-函数的参数

**一.形参和实参：**

函数说明中的参数称为形式参数（形参），函数调用中的参数称为实际参数（实参）

实参表的参数个数、参数顺序、以及参数类型要与被调函数的形参表之间一一对应，才能相互“匹配成功”。

**二.无名参数**

函数定义中，只有类型，没有名称的参数

int f(int a,int b)

{

return 2\*a+b;

}

int f(int a,int b,int) //包含一个无名参数

{

return a+b\*b;

}

int f()

{

return 0;

}

/\*

函数的参数表可以为空

调用参数表为空的函数时，括号不可省略

\*/

int main()

{

int a=1,b=3;

cout<<f()<<endl; //调用第三个函数

cout<<f(a,b)<<endl; //调用第一个函数

cout<<f(a,b,0)<<endl; //调用第二个函数

cout<<f(a,b,1)<<endl; //调用第二个函数

return 0;

}

**三.可缺省参数（为参数设置默认值）**

允许在函数定义处为**连续若干个**参数设置默认值（也称缺省值）

若调用处缺省了某个或某些实参的情况下，系统将自动使用那些在函数定义处给定的参数默认值

void f(int a=1,int b=2,int c=3)

{

//为参数设置了默认值1、2、3

cout<<a<<" "<<b<<" "<<c<<endl;

}

/\*

这是一个没有返回值的函数，类型void为“空型”或“无值型”

这种函数通常用来在不需要返回值时实现某种特定的功能

\*/

int main()

{

int m=6,n=66,k=666;

f(); //调用时省略全部参数，则全部使用默认值

f(m); //调用时缺省（后）两个参数，则后两个数使用默认值

f(m,n); //调用时缺省（后）一个参数，则后一个数使用默认值

f(m,n,k); //调用时未缺省参数，不使用默认值

return 0;

}

注意：只能为函数最后面的连续若干个参数设置默认值，且在调用处也只能缺省后面的连续若干个实参

//void f(int a=1,int b,int c=3) //ERROR！

//void f(int a=1,int b=2,int c) //ERROR！

void f(int a,int b=2,int c=3)

{

//a没有设置默认值

cout<<a<<" "<<b<<" "<<c<<endl;

}

int main()

{

int m=6,n=66,k=666;

//f(); //ERROR! a没有默认值

f(m); //调用时缺省（后）两个参数，则后两个数使用默认值

f(m,n); //调用时缺省（后）一个参数，则后一个数使用默认值

f(m,n,k); //调用时未缺省参数，不使用默认值

//f(m,,k); //ERROR！调用处只能缺省后面的连续若干个实参

return 0;

}

**四.引用变量：**

代码示例：

int x=10,y=20;

int &a=x; //a是一个引用变量，【注意】引用变量必须在声明的同时初始化！

//a变成了x的“别名”，完全可以说，a就是x！这里的“&”不是取地址运算符，只是声明a是一个引用变量

cout<<"a="<<a<<endl;

cout<<"a的地址为"<<&a<<" x的地址为"<<&x<<endl;

a=y;

/\*

等价于x=y;a和x的值的确都改变为y的值，但地址仍然不变且相同

这只是一个简单的赋值操作，引用变量只能在“声明的同时”初始化，赋值并不能使a成为y的别名

\*/

cout<<"x="<<x<<" "<<"a="<<a<<" "<<"y="<<y<<endl;

cout<<"a的地址为"<<&a<<" x的地址为"<<&x<<" y的地址为"<<&y<<endl;

//决定引用变量是谁的别名，就是看声明的同时初始化的时候

代码示例：

void Ex1(int a,int b)

{

//赋值传递

//a=x; b=y;

int tmp;

tmp=a;

a=b;

b=tmp;

cout<<a<<" "<<b<<endl;

}

void Ex1(int x,int y,int)

{

//赋值传递

int tmp;

tmp=x;

x=y;

y=tmp;

cout<<x<<" "<<y<<endl;

}

void Ex2(int &a,int &b)

{

//按引用传递

//&a=x; &b=y;

int tmp;

tmp=a;

a=b;

b=tmp;

cout<<a<<" "<<b<<endl;

}

int main()

{

int x=1,y=2;

Ex1(x,y);

cout<<x<<" "<<y<<endl;

Ex1(x,y,0);

cout<<x<<" "<<y<<endl;

Ex2(x,y);

cout<<x<<" "<<y<<endl;

return 0;

}

**五.函数调用过程中参数的传递**

1.赋值传递

(1)发生函数调用，转到函数体执行

(2)根据函数的形参，分配内存空间

(3)**将实参赋值给形参**，即为形参分配的存储空间赋值，此时实参在函数体内失效,形参有效

(4)函数执行完毕，返回到主调函数，形参所占的空间自动回收，形参失效

//单向传值：只能向被调用的函数传入值，但无法通过形参将值传出

2.按地址传递过程

发生函数调用，转到函数体执行

根据代表地址的参数（数组名、指针、引用等），在以该参数为首地址的空间中进行各种处理

函数执行完毕，无论是否有返回值，函数体对内存空间进行的修改将保留，**对主调函数仍然有效**

//双向传值：既能向被调用的函数传入值，也可以通过形参将值传出

六.当数组作为函数的参数时

1.

（1）一维数组作为函数的参数，数组大小可以指定，也可以不指定；

（2）多维数组作为函数的参数，除了最高维之外，其它维的大小必须指定。

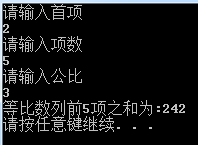
2.数组作为函数参数，是将实参数组的首地址传递给形参，而不是将数组的所有元素传递给形参——这很像一个指针

在函数体中，根据形参数组首地址和下标指示的偏移量访问数组元素

这也就说明，一维数组以及多维数组的第一维大小，形参、实参可以不对应

代码示例：求等比数列前n项的和(假设输入都是整数)

分别输入首项、项数和公比后，输出等比数列前n项的和



#include <iostream>

using namespace std;

int add(int b[],int size) //逐项加和

{ int total=0;

for(int i=0;i<=size-1;i++)

{total+=b[i];}

return total;

}

int sum(int n,int m,int q) //求出每一项并逐项加和

{

int i;

int a[100];

a[0]=n;

for(i=1;i<m;i++)

{

a[i]=a[i-1]\*q;

}

return add(a,m);

}

int main()

{ int n,m,q,Sum;

cout<<"请输入首项"<<endl;

cin>>n;

cout<<"请输入项数"<<endl;

cin>>m;

cout<<"请输入公比"<<endl;

cin>>q;

Sum=sum(n,m,q);

cout<<"等比数列前"<<m<<"项之和为:"<<Sum<<endl;

return 0;

}