C++与C的主要差异
– 函数的引用调用

- 将函数的形参声明为引用主要起到以下两方面的作用。
- 1. 通过引用调用更改实参变量的值
- 前面学习了函数的传值调用，在传值调用方式下，参数的传递为单向传值，即实参值传递给形参后，形参值在函数中的变化对实参值无任何影响。

【例1-12】函数的传值调用。

```
#include <iostream>
using namespace std;
void swap(int a, int b);
int main()
{
    int x=5, y=10;
    cout<<"交换前,x="<<x
        <<",y="<<y<<endl;
    swap(x, y);
    cout<<"交换后,x="<<x
        <<",y="<<y<<endl;
    return 0;
}
```

```
void swap(int a, int b)
{
    int t=a;
    a=b;
    b=t;
}
```

【例1-13】函数的引用调用。

```
#include <iostream>
using namespace std;
void swap(int &a, int &b);
int main()
{
    int x=5, y=10;
    cout<<"交换前,x="<<x
        <<",y="<<y<<endl;
    swap(x, y);
    cout<<"交换后,x="<<x
        <<",y="<<y<<endl;
    return 0;
}
```

```
void swap(int &a, int &b)
{
    int t=a;
    a=b;
    b=t;
}
```

对自定义类型的变量，也可以通过引用方式传递:

```
void StudentInfoInput(Student &stu)
{
    cin>>stu.num>>stu.name>>stu.score;
}
```

在调用时，直接将pStu[i]作为实参传递:

```
StudentInfoInput(pStu[i]);
```

由于形参stu是实参pStu[i]的引用，因此在StudentInfoInput()函数中对形参stu所做的操作就是对实参pStu[i]的操作。

- 2. 通过引用调用提高函数调用效率
- 在调用函数时，需要将实参的值传递给形参。如果一个实参本身的数据量较大，则这个传递过程会消耗较长的时间。为了减少参数传递的时间开销，可以对一些数据量比较大的实参（如结构体变量或对象）采用引用调用方式。
- 当以引用调用方式传递实参，而在函数体中又不需要更改实参的值，则一般在引用形参中加上const关键字，使其成为const引用。使用const引用只能访问所引用对象的值，而不能修改所引用对象的值。const引用有两种声明形式:
- const <数据类型> &<引用名>=<变量名或常量>;
- 或 <数据类型> const &<引用名>=<变量名或常量>;

例如:

```
int a=3;
const int &r=a;
r=10;//错误:不能通过const引用修改所引用对象的值
```

再如：

```
void StudentInfoOutput(const Student &stu)
{
    cout<<stu.num<<','<<stu.name<<',  '<<stu.score<<endl;
}
```

在调用时，直接将pStu[i]作为实参传递:

```
StudentInfoOutput(pStu[i]);
```

- 另外，由于const引用不需要修改所引用对象的值，所以const引用与非const引用还有一个区别: const引用可以使用常量对其进行初始化，而非const引用则不可以。例如:
- const int &r1=3; //正确:const引用可以使用常量对其进行初始化
- int &r2=3; //错误:非const引用不能使用常量对其进行初始化
- 引用调用和传值调用也可以混合使用，例如:
- int fun(int &a, int b);
- 其中，a是引用调用，b是传值调用。