C++与C的主要差异
– 引用和返回引用的函数

- 引用就是别名，变量的引用就是变量的别名，对引用的操作就是对所引用变量的操作。
- 引用的声明形式为:
- <数据类型> &<引用名>=<变量名>;
- 建立引用时，必须用已知变量名为其初始化，表示该引用就是该变量的别名。&是引用运算符，作用于引用名，表示紧随其后的是一个引用。例如，要同时定义两个int型引用r1和r2，必须写成如下形式:
- int a,b;
- int &r1=a, &r2=b;

如果写成:

```
int &r1=a, r2=b;
```

则表示定义了一个引用r1和一个普通变量r2。

一个引用所引用的对象初始化后就不能修改。另外，引用就是一个别名，声明引用不会再为其分配内存空间，而是与所引用对象对应同一片内存空间。

因此，对引用的操作与对所引用对象的操作效果完全一样。例如:

```
int a=5, b=10;
int &r=a;//r是a的引用
r=b;    //将b的值赋给r。因为r是a的引用,所以相当于将b的值赋给a
```

- 也可以为指针变量声明引用，其声明形式为:
- <数据类型> *&<引用名>=<指针变量名>;
- 在实际应用时，引用主要是用在函数中，一方面可以将函数的返回类型声明为引用，另一方面可以将函数的形参声明为引用。

- 返回引用的函数是指函数的返回值是return后变量的引用，返回引用的函数调用可以作为赋值语句的左值。
- 【例1-11】返回引用的函数示例。
- 程序实现

```
#include <iostream>
using namespace std;
int array[5]={1, 2, 3, 4, 5};
int& index(int i);
int main()
{
    cout<<"赋值前,array[3]="
        <<array[3]<<endl;
    index(3)=15;
    cout<<"赋值后,array[3]="
        <<array[3]<<endl;
    return 0;
}
int& index(int i)
{
    return array[i];
}
```

- 提示:
- （1）只有返回引用的函数可以作为赋值语句的左值。返回引用的函数通常用在类中。
- （2）在返回引用的函数中，可以返回全局变量或静态变量的引用，但不能返回局部变量的引用，因为局部变量的生存期只是在定义该局部变量的函数中，当函数调用结束时局部变量的内存空间会被释放，对已释放的内存空间进行引用可能会出现问题。