问题

在函数定义的时候，经常看到使用引用作为函数的参数或者返回值，这样做的好处在哪里？

引用就是某一变量（目标）的别名，对引用的操作与对变量直接操作完全一样。

引用作为函数参数

作为函数参数时引用有种原因：

1. 在函数内部会对此参数进行修改
2. 提高函数调用和运行效率

关于第一点：我们都知道C++里提到函数就会提到形参和实参。函数的参数实质就是形参，不过这个形参的作用域只是在函数体内部，也就是说实参和形参是两个不同的东西，要想形参代替实参，肯定有一个值的传递。函数调用时，值的传递机制是通过“形参=实参”来对形参赋值达到传值目的，产生了一个实参的副本。即使函数内部有对参数的修改，也只是针对形参，也就是那个副本，实参不会有任何更改。函数一旦结束，形参生命也宣告终结，做出的修改一样没对任何变量产生影响。而如果我们将形参定义成引用，那么此时的赋值后形参只是实参的一个别名而已，此时函数体内对形参的任何修改都同样作用与实参。

关于第二点：可以结合第一点分析，形参是实参的引用，不用经过值的传递机制，已经有了实参值的信息。所以没有了传值和生成副本的时间和空间消耗。当程序对效率要求比较高时，这是非常必要的.

引用作为函数返回值

1. 以引用返回函数值，定义函数时需要在函数名前加 `&`
2. 用引用返回一个函数值的最大好处是：在内存中不产生被返回值的副本

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

float temp;              //定义全局变量temp
float fn1(float r);      //声明函数fn1
float &fn2(float r);     //声明函数fn2

float fn1(float r)       //定义函数fn1，它以返回值的方法返回函数值
{
    temp=(float)(r*r*3.14);
    return temp;
}
float &fn2(float r)      //定义函数fn2，它以引用方式返回函数值
{
    temp=(float)(r*r*3.14);
    return temp;
}

int main()
{
    float a=fn1(10.0);       //第1种情况，系统生成要返回值的副本（即临时变量）
    //float &b=fn1(10.0);    //第2种情况，编译不通过，左值引用不能绑定到临时值
    float &d=fn2(10.0);      //第3种情况，系统不生成返回值的副本，可以从被调函数中返回一个全局变量的引用
```

```
    float c=fn2(10.0);       //第4种情况，变量c前面不用加&号，这种也是可以的
    cout<<a<<endl<<c<<endl<<d<<endl;
    return 0;
}
```

引用作为返回值，**必须遵守一下规则**：

1. **不能返回局部变量的引用**。主要原因是局部变量会在函数返回后被销毁，因此被返回的引用就成为了"无所指"的引用，程序会进入未知状态。
2. **不能返回函数内部new分配的内存的引用**。虽然不存在局部变量的被动销毁问题，可对于这种情况（返回函数内部new分配内存的引用），又面临其它尴尬局面。例如，被函数返回的引用只是作为一 个临时变量出现，而没有被赋予一个实际的变量，那么这个引用所指向的空间（由new分配）就无法释放，造成memory leak。
3. **引用与一些操作符的重载。**流操作符<<和>>，这两个操作符常常希望被连续使用，例如：cout << "hello" << endl; 因此这两个操作符的返回值应该是一个仍然支持这两个操作符的流引用。

参考资料

“引用作为函数参数”与 “引用作为函数返回值”