chapter7 函数 - c++的编程模块

（1）基本知识：

> 略之！

（2）函数指针：

> 这里只介绍基础知识，高级知识现在没法理解！
> [1]引入；
>
> 与数据项相似，函数也有地址。函数的地址是存储其机器语言代码的内存的开始地址。通常，这些地址对用户而言，既不重要，也没有什么用处，但对程序而言，却很有用。例如，可以编写将另一个函数的地址作为参数的函数。这样第一个函数将能够找到第二个函数，并运行它。与直接调用另一个函数相比，这种方法很笨拙，但它允许在不同的时间传递不同函数的地址，这意味着可以在不同的时间使用不同的函数。
>
> [2]基础知识：
> （1）要点：获取函数的地址 + 声明一个函数指针 + 使用函数指针调用函数
>
> （2）获取函数的地址：
>
> > 方法：只需要使用函数名（后面不跟参数）即可
> > eg：如果think()是一个函数，则think就是函数地址
> > ***要将函数作为参数进行传递，必须传递函数名！***
>
> 注意区分传递的是 函数的地址 还是 函数的返回值 ！

```
已有函数think():
[1] process( think )    // process()函数能在内部调用think()函数

[2] thought( think() )  // thought()首先调用think()函数，然后将think()返回值传递给thought()函数
```

> （3）声明函数指针：
> 声明应当指定：函数的返回类型 + 函数的参数列表

```
（1）已有函数：  double pam(int);
（2）建立正确的函数指针声明：
        double (*pf)(int);
        //1)  (*pf)与函数名等价，pf是指向pam函数的指针
        //2)  指针pf指向的函数pam的参数类型是int，返回值是double
（3）正确声明pf之后，可以将相应的函数地址赋给它：
```

```
        double pam(int);
        double (*pf)(int);
        pf = pam;          // pf now points to the pam() function
```

（4）使用过程：

```
        void estimate( int lines , double (*pf)(int) ) {...}
        ......
        estimete(666,pam);    // function call telling estimate() to use pam()
```

ps: 1) 类型必须完全一一对应！ 2) 由于运算符优先级，***pf外面的( )不可以省略!***

> （4）使用指针调用函数：

```
double pam(int);
double (*pf)(int);
pf = pam;
...
double x = pam(4);
double y = (*pf)(7);
```

ps:

```
const double * f1(const double ar[], int n);
const double * f2(const double [], int);
const double * f3(const double *, int);
```

这些函数的特征标看似不同，但实际上相同。首先，前面说过，在函数原型中，参数列表 const double ar [ ]与 const double * ar 的含义完全相同。其次，在函数原型中，可以省略标识符。因此，const double ar [ ] 可简化为 const double [ ]，而 const double * ar 可简化为 const double *。因此，上述所有函数特征标的含义都相同。另一方面，函数定义必须提供标识符，因此需要使用 const double ar [ ]或 const double * ar。