面向对象与设计

1. 设计一个多边形 Polygon，其由若干个点连成线。为其实现构造函数，拷贝构造函数，复制操作符，析构函数。

代码：文件夹 Q1 / Polygon.h ；测试例子：Q1 / test_Polygon.cpp

定义了 vector<Point*> points 来存放多边形的顶点

（1）构造函数中，采用深拷贝为 points赋值：

```
for(auto i = 0; i < p.size(); i++)
    {
        if(p[i]!=nullptr)
        {
            Point* point = new Point(*p[i]);
            points.push_back(point);
        }
    }
```

（2）在析构函数中，将vector<Point*> points中的指针元素delete，最后将vector清空：

```
for(auto i = 0; i < points.size(); i++)
{
    delete points[i];
    points[i] = nullptr;
}
points.clear();
```

（3）赋值操作符重载中，先判断是不是自己给自己赋值：

```
if(&polygon == this)
    return *this;
```

然后给父类赋值：

```
this->Shape::operator=(polygon);
```

下一步， 当this的vector不空时，先将vector清空：

```
if(!points.empty())
{
    this->Vector_clc();
}
```

2. 为 Point 类设计一个数据绑定机制，当其坐标 x 或 y 被更改时，可以通知外界其更改的过程（即从旧值改为新值）。将更改过程打印在控制台上。 考虑使用松耦合设计（即未来有可能将更改过程显示在任何界面上）。

代码：Q2/ Notifier.h, Observer.h, Point.h, test_Notifier.cpp

（1）Notifier.h 中是通知者，继承了Point父类；
其中，AddObserver函数添加观察者，RemoveObserver函数删除观察者，Notice
函数发送通知。当点坐标变化时，调用Notice函数

（2）Observer.h中，定义了抽象基类Observer，以及子类SubObserver1和SubObserver2，来实现Response函数，表示收到点坐标变化的通知

（3）test_Notifier.cpp中，定义了两个观察者实例so1，so2，添加到通知者p1的私有队列中；改变p1坐标，为so1和so2发送通知

STL与泛型编程

1. 给定一个 vector：v1 = [0, 0, 30, 20, 0, 0, 0, 0, 10, 0]，希望通过 not_equal_to 算法找到到不为零的元素，并复制到另一个 vector: v2

代码：STL/STL1.cpp

```
while(it2!=v1.end())
{
    it1 =
find_if(it2,v1.end(),(std::bind1st(std::not_equal_to<int>(),0)));
    if(it1==v1.end())
    {
        break;
    }
    v2.push_back(*it1);
    it2 = it1+1;
}
```

2. 为以下 Programmer 对象提供一个基于 Id 并且升序的仿函数 ProgrammerIdGreater，使得 Programmer 对象可以在 set 中以 Id 排序存放

代码：STL/STL2.cpp

算法与系统设计

1. Moving Average

给定一移动个窗口，计算在这个窗口内的平均数，设计这个class，并给出测试用例

代码：movingAverage.cpp
通过deque<double> 双向队列存储窗口内的数据
测试用例： test1: 窗口长度2，数据长度3
test1: 窗口长度0，数据长度3
test1: 窗口长度6，数据长度15

2. Total Difference Strings

给一个string列表，判断有多少个不同的string，返回个数相同的定义：字符串长度相等并从左到右，或从右往左是同样的字符abc和cba为视为相同

代码：strCount.cpp
（1） 遍历已知的vector<string>，将每个string元素正向，反向较大的值存入新的vector<string>

```
for(int i = 0; i < strin.size(); i++)
{
    s1=strin[i];
    reverse(s1.begin(),s1.end());
    if(s1 >= strin[i])
        strList.push_back(s1);
    else
        strList.push_back(strin[i]);
 }
```

（2）将新的vector排序，找出不重复的元素个数
测试用例：test1()vector为空，test2()和test3()都是正常字符串

3. Binary Tree Print

给出一个二叉树，打印所有从root到叶子节点的路径

代码：printTreePath.cpp
利用递归实现先序遍历的思路，当遍历到叶子节点时，打印
测试用例：只有一个分支的二叉树；
正常的二叉树；
空树；
只有根节点的二叉树

vector<int>NodesList 记录路径节点
vector<bool>LeftOrRight 记录该节点时左子节点还是右子节点，用于控制空格，print in beautiful order