Assignment 10

191220022 丁一凡

一、 概念题

1.1 STL主要包含哪些内容？它们的功能分别是什么？

STL主要包含了一些容器模板、算法模板以及迭代器模板。容器用于数据存储，算法用于对容器中的元素进行一些常用操作，迭代器用于访问容器中的元素。

1.2 迭代器有哪几种类型？为什么sort算法不支持对list类型的排序？

迭代器有五种类型：输出迭代器、输入迭代器、前向迭代器、双向迭代器以及随机访问迭代器。

sort算法需要一个随机访问迭代器，而list的迭代器是双向迭代器

二、 编程题

2.1

```
//升序输出计算机专业男生的姓名
vector<Student> result;
vector<Student>::iterator it = result.begin();
it = copy_if(students.begin(), students.end(), [](Student &st){ return
st.get_major() == COMPUTER && st.get_sex() == MALE; });
sort(result.begin(), result.end(), [](Student &s1, Student &s2){ return
s1.get_name() < s2.get_name(); });
for(; it != result.end(); it++)
    cout << result[it].get_name() << "  ";

//升序输出出生地是“南京”、专业为哲学或数学的学生年龄
vector<Student> result;
vector<Student>::iterator it = result.begin();
it = copy_if(students.begin(), students.end(), [](Student &st){ return
st.get_birth_place == "南京" && (st.get_major() == PHILOSOPHY || st.get_major()
== MATH; });
sort(result.begin(), result.end(), [](Student &s1, Student &s2){ return
s1.get_birth_date() > s2.get_birth_date(); });
for(; it != result.end(); it++)
    cout << result[it].get_birth_date() << "    ";

//统计全部女生的平均年龄
double total = 0, size = 0;
```

```
vector<Student>::iterator it = students.begin();
for(; it!= students.end(); it++)                     {
    if(students[it].get_sex() == FEMALE)
        total += students[it].get_age();
}
cout << total / size << endl;

//统计出生地是“南京”的计算机专业学生平均年龄
double total = 0, size = 0;
vector<Student>::iterator it = students.begin();
for(; it!= students.end(); it++)                     {
    if(students[it].get_birth_place() == "南京" && students[it].get_major() ==
COMPUTER)
        total += students[it].get_age();
}
cout << total / size << endl;


//统计非计算机专业年龄小于二十岁的学生的平均年龄
double total = 0, size = 0;
vector<Student>::iterator it = students.begin();
for(; it!= students.end(); it++)                     {
    if(students[it].get_age() < 20 && students[it].get_major() != COMPUTER)
        total += students[it].get_age();
}
cout << total / size << endl;
```

2.2

```
#include <vector>
#include <algorithm>
using namespace std;
class Point
{
    public:
    int x, y;
    Point(int _x, int _y) : x(_x), y(_y) {}
};

int count = 0;
void print(Point &p)
{
    cout << "(" << p.x << "," << p.y << ")" << endl;
}
void count3(Point &p1, Point &p2)
{
    count += (p1.x * p1.x + p1.y * p1.y) + (p2.x * p2.x + p2.y * p2.y);
}
void count4(Point &p)
{
    count += p.x * p.x + p.y * p.y;
}
void count6(Point &p1, Point &p2)
{
    if((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (p1.y - p2.y) == 2)
        count++;
}
int main()
{
    vector<Point> p, q;
    p.push_back(Point(-1, -1));
    p.push_back(Point(2, 2));
    q.push_back(Point(1, 1));
    q.push_back(Point(-2, -2))
    //...
    // 1. 对p、q中顶点"(x, y)" 升序排序并输出(要求按照x大小，若x相等则按y的大小)
    sort(p.begin(), p.end(), [](Point &p1, Point &p2) { return (p1.x < p2.x) ||
(p1.y < p2.y); });
    sort(q.begin(), q.end(), [](Point &p1, Point &p2) { return (p1.x < p2.x) ||
(p1.y < p2.y); });
    for_each(p.begin(), p.end(), print);
    for_each(q.begin(), q.end(), print);

    // 2. 升序输出p中满足x > 0, y > 0的所有顶点与(0, 0)的距离的平方
    vector<Point> result2 = copy_if(p.begin(), p.end(), [](Point &p){ return
p.x>0 && p.y > 0; });
    sort(result2.begin(), result2.end(), [](Point& p1, Point& p2) { return (p1.x
* p1.x + p1.y * p1.y) < (p2.x * p2.x + p2.y * p2.y); });
    for_each(result2.begin(), result2.end(), print);
    // 3. 根据1排序后p中顶点的顺序计算满足x > 0, y > 0相邻两个顶点的距离的平方和

    vector<Point>::iterator it3 = transform(result2.begin(), result.end() - 1,
result.begin() + 1, it3, count3);
    cout << count << endl;

    // 4. 计算p中x > 0, y > 0的顶点与(0, 0)的距离的平方和
    count = 0;
    for_each(result2.begin(), result2.end(), count4);
    cout << count << endl;
    // 5. p、q在每个象限顶点数目相同，统计在1排序后p、q中满足x < 0, y < 0的顶点中按顺序所对
应顶点距离的平方为2的数目
    count = 0;
    vector<Point>::iterator it5 = p.begin();
    vector<Point>::iterator it6 = q.begin();
    for(; it5 != p.end(); it5++, it6++)
    {
        if(p[it5].x < 0 && p[it5].y < 0)
            break;
    }
    transform(it5, p.end(), it6, it3, count6);
    cout << count << endl;

    return 0;
}
```