HW10

181220076 周韧哲

一. 概念题

1. STL主要包含的内容和其功能为：
   - 容器：用于存储序列化的数据，如：向量、队列、栈、集合等。
   - 算法（函数）：用于对容器中的数据元素进行一些常用操作，如：排序、统计等。
   - 迭代器：实现了抽象的指针功能，它们指向容器中的数据元素，用于对容器中的数据元素进行遍历和访问。
2. 迭代器分为：输出迭代器、输入迭代器、前向迭代器、双向迭代器、随机访问迭代器等。因为sort算法中的操作需要随机访问迭代器，而list的内存空间不是连续的。

二. 编程题

1.
```
enum Sex {MALE, FEMALE};
enum Major {MATHEMATICS, PHYSICS, CHEMISTRY, COMPUTER, GEOGRAPHY,
            ASTRONOMY, ENGLISH, CHINESE, PHILOSOPHY};
class Date{
    int year;
    int month;
    int day;
public:
    Date(){}
    Date(int y, int m, int d){
        year = y;
        month = m;
        day = d;
    }
    int get_year() { return year; }
    int get_month() { return month; }
    int get_day() { return day; }
};

class Student{
    int no;
    string name;
    Sex sex;
    Date birth_date;
    string birth_place;
    Major major;
public:
    Student(int no1, char *name1, Sex sex1, Date birth_date1, char
*birth_place1, Major major1){
        birth_place = birth_place1;
        name = name1,
        birth_date = birth_date1,
        no = no1;
        sex = sex1;
        major = major1;
    }
    int get_no() { return no; }
```

```
    string get_name() { return name; }
    string get_birth_place() {return birth_place; }
    int get_sex() { return sex; }
    Major get_major() { return major; }
    int get_age(){ return 2021-birth_date.get_year(); }
};

int main(){
    vector<Student> students;
    students.push_back(Student(2,(char*)"zhang",FEMALE,Date(1990,10,1),
(char*)"nanjing",COMPUTER));
    students.push_back(Student(5,(char*)"li",MALE,Date(1990,10,1),
(char*)"beijing",PHILOSOPHY));
    students.push_back(Student(1,(char*)"wang",MALE,Date(1991,10,1),
(char*)"nanjing",COMPUTER));
    students.push_back(Student(4,(char*)"qian",FEMALE,Date(1991,11,1),
(char*)"shanghai",PHILOSOPHY));
    students.push_back(Student(3,(char*)"zhao",MALE,Date(1993,10,1),
(char*)"nanjing",COMPUTER));
    students.push_back(Student(7,(char*)"shao",MALE,Date(1993,10,1),
(char*)"nanjing",MATHEMATICS));
    students.push_back(Student(6,(char*)"hao",MALE,Date(1993,10,1),
(char*)"nanjing",PHILOSOPHY));
    students.push_back(Student(8,(char*)"hong",MALE,Date(2003,10,1),
(char*)"nanjing",PHILOSOPHY));
    students.push_back(Student(9,(char*)"hou",MALE,Date(2005,10,1),
(char*)"nanjing",PHILOSOPHY));

    //q1
    vector<Student> q1;
    copy_if(students.begin(), students.end(), back_inserter(q1),
            [](Student &st){ return st.get_major()==COMPUTER &&
st.get_sex()==MALE;});
    sort(q1.begin(), q1.end(), [](Student &st1, Student &st2){ return
st1.get_no()<st2.get_no();});
    cout<<"================q1================"<<endl;
    for_each(q1.begin(), q1.end(), [](Student &st){ cout<<st.get_no()<<", "
<<st.get_name()<<endl;});

    //q2
    vector<Student> q2;
    copy_if(students.begin(), students.end(), back_inserter(q2),[](Student
&st){return
        st.get_birth_place()=="nanjing" && (st.get_major()==PHILOSOPHY ||
st.get_major()==MATHEMATICS);});
    sort(q2.begin(), q2.end(), [](Student &st1, Student &st2){ return
st1.get_no()<st2.get_no();});
    cout<<"================q2================"<<endl;
    for_each(q2.begin(), q2.end(), [](Student &st){ cout<<st.get_no()<<", "
<<st.get_name()<<", "<<st.get_age()<<endl;});

    //q3
    vector<Student> q3;
    copy_if(students.begin(), students.end(), back_inserter(q3),[](Student
&st){return st.get_sex()==FEMALE;});
    double mean_age_q3 = (double)accumulate(q3.begin(), q3.end(), 0, [](int
partial, Student &st)->int{ return partial+st.get_age();})/q3.size();
    cout<<"================q3================"<<endl;
```

```
    cout<<mean_age_q3<<endl;

    //q4
    vector<Student> q4;
    copy_if(students.begin(), students.end(), back_inserter(q4),[](Student
&st){return st.get_birth_place()=="nanjing" && st.get_major()==COMPUTER;});
    double mean_age_q4 = (double)accumulate(q4.begin(), q4.end(), 0, [](int
partial, Student &st)->int{ return partial+st.get_age();})/q4.size();
    cout<<"================q4================"<<endl;
    cout<<mean_age_q4<<endl;

    //q5
    vector<Student> q5;
    copy_if(students.begin(), students.end(), back_inserter(q5),[](Student
&st){return st.get_major()!=COMPUTER && st.get_age()<20;});
    double mean_age_q5 = (double)accumulate(q5.begin(), q5.end(), 0, [](int
partial, Student &st)->int{ return partial+st.get_age();})/q5.size();
    cout<<"================q5================"<<endl;
    cout<<mean_age_q5<<endl;
    return 0;
}
```

2.
```
class Point{
    int x, y;
public:
    Point(int _x, int _y) : x(_x), y(_y) {}
    int get_x() { return x; }
    int get_y() { return y; }
};

bool lessthan(Point &pt1, Point &pt2){
    if(pt1.get_x() == pt2.get_x()){
        return pt1.get_y() < pt2.get_y();
    }
    return pt1.get_x() < pt2.get_x();
}

int main(){
    vector<Point> p, q;
    p.push_back(Point(-1, -1));
    p.push_back(Point(-2, -2));
    p.push_back(Point(1, 1));
    p.push_back(Point(6, 9));
    q.push_back(Point(1, 1));
    q.push_back(Point(-3, -3));
    q.push_back(Point(4, 3));
    q.push_back(Point(-2, -2));

    //q1
    sort(p.begin(), p.end(), lessthan);
    sort(q.begin(), q.end(), lessthan);
    cout<<"================q1================"<<endl;
    cout<<"p: ";
    for_each(p.begin(), p.end(), [](Point &pt){ cout<<"("<<pt.get_x()<<","
<<pt.get_y()<<") ";});
    cout<<endl<<"q: ";
```

```
    for_each(q.begin(), q.end(), [](Point &pt){ cout<<"("<<pt.get_x()<<","
<<pt.get_y()<<") ";});
    cout<<endl;

    //q2
    vector<Point> q2;
    copy_if(p.begin(), p.end(), back_inserter(q2), [](Point &pt){ return
pt.get_x()>0 && pt.get_y()>0;});
    cout<<"================q2================"<<endl;
    sort(q2.begin(), q2.end(), lessthan);
    for_each(q2.begin(), q2.end(), [](Point &pt){
cout<<pow(pt.get_x(),2)+pow(pt.get_y(),2)<<" ";});
    cout<<endl;

    //q3
    vector<Point> q3;
    copy_if(p.begin(), p.end(), back_inserter(q3), [](Point &pt){ return
pt.get_x()>0 && pt.get_y()>0;});
    cout<<"================q3================"<<endl;
    int dist_q3 = 0;
    for(vector<Point>::iterator curr=q3.begin(); curr!=q3.end();curr++){
        if(next(curr) != q3.end()){
            dist_q3 += pow(curr->get_x() - next(curr)->get_x(), 2) +
pow(curr->get_y() - next(curr)->get_y(), 2);
        }
    }
    cout<<dist_q3<<endl;

    //q4
    vector<Point> q4;
    copy_if(p.begin(), p.end(), back_inserter(q4), [](Point &pt){ return
pt.get_x()>0 && pt.get_y()>0;});
    cout<<"================q4================"<<endl;
    int q4_sum = accumulate(q4.begin(), q4.end(), 0, [](int partial, Point
&pt){return partial+pow(pt.get_x(),2)+pow(pt.get_y(),2);});
    cout<<q4_sum<<endl;

    //q5
    vector<Point> q5_p, q5_q;
    copy_if(p.begin(), p.end(), back_inserter(q5_p), [](Point &pt){ return
pt.get_x()<0 && pt.get_y()<0;});
    copy_if(q.begin(), q.end(), back_inserter(q5_q), [](Point &pt){ return
pt.get_x()<0 && pt.get_y()<0;});
    cout<<"================q5================"<<endl;
    int cnt = 0;
    vector<Point>::iterator curr_p=q5_p.begin();
    vector<Point>::iterator curr_q=q5_q.begin();
    for(;
        curr_p!=q5_p.end() && curr_q!=q5_q.end(); curr_p++, curr_q++){
        int dist = pow(curr_p->get_x() - curr_q->get_x(), 2) + pow(curr_p-
>get_y() - curr_q->get_y(), 2);
        if(dist == 2) cnt++;
    }
    cout<<cnt<<endl;
    return 0;
}
```