概念题

2. C++的 STL 提供了哪几种模板? 分别简述它们的作用.
   容器类模板: 用于存储序列化的数据, 如: 向量, 队列, 栈, 集合等
   算法函数模板: 用于对容器中的数据元素进行常用操作, 如: 排序, 统计等
   迭代器类模板: 抽象指针指向容器中的数据元素, 用于数据元素遍历和访问
      是容器和算法之间的桥梁: 传给算法的不是容器, 而是指向容器中元素的迭代器
      算法通过迭代器实现对容器中数据元素的访问, 提高算法的通用性

3. 列举 STL 中能快速定位(访问)任意位置的容器, 并说明它们的内部数据结构.
   vector: 用动态数组实现
   deque: 用分段的连续空间结构实现

4. 简述 C++中自定义操作条件(谓词)的概念及作用.
   有些算法可以让使用者提供一个函数或函数对象来作为自定义操作条件(谓词)
   用于对数据元素进行排序和统计时可以自定义排序或统计等规则

编程题

1. 有一个简单的字符串处理框架, 请补全代码完成以下 3 个接口.

```
class StrOperation {
private:
    string str;
public:
    StrOperation(string s) { str = s; }
    bool judgePalindrome() {
        bool flag = true;
        for (int i = 0; i < str.length() / 2; i++) {
            if (str[i] != str[str.length() - i - 1]) {
                flag = false;
                break;
            }
        }
        return flag;
    }
    void insertStr(int i, string s) {
        str.insert(i, s);
    }
    void replaceStr(int be, int en, string s) {
        str.replace(be - 1, en - be + 1, s);
    }
};
```

2. 为方便管理动物园里的动物，有一个系统 System 记录了园内的动物和数量.
   1) 添加新的动物，记录其数量(所有动物按动物单词的首字母从 a 到 z 排序)
   2) 按动物名称删除动物
   3) 按动物名称查找某种动物的数量(如果园内没有该动物，则返回 0)
   4) 输出当前动物种类数和动物个体的总数量
   合理利用所学的容器知识实现该系统类 System，设计程序测试所有管理操作

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

class Animal {
    string name;
    int num;
    friend class System;
public:
    Animal(string n, int m) {
        name = n;
        num = m;
    }
};

class System {
    vector<Animal> Animals;
    int species;
    int total;
public:
    System() {
        vector<Animal> Animals;
        species = 0;
        total = 0;
    }
    void addAnimal(Animal a) {
        if (Animals.size() == 0) {
            Animals.push_back(a);
            species += 1;
            total += a.num;
        }
        else {
            bool exist = false;
            vector<Animal>::iterator it;
            it = Animals.begin();
            for (; it != Animals.end(); it++) {
```

```
            if (a.name.compare(it->name) == 0) {
                exist = true;
                it->num += a.num;
                break;
            }
            else if (a.name.compare(it->name) > 0) {
                it++;
                break;
            }
        }
        if (!exist) {
            Animals.insert(it, a);
            species += 1;
            total += a.num;
        }
    }
}
void deleteAnimal(string name) {
    bool exist = false;
    vector<Animal>::iterator it;
    it = Animals.begin();
    for (; it != Animals.end(); it++) {
        if (name.compare(it->name) == 0) {
            exist = true;
            break;
        }
    }
    if (exist) {
        species -= 1;
        total -= it->num;
        Animals.erase(it);
    }
    else
        cout << "Animal does not exist!" << endl;
}
int searchAnimal(string name) {
    bool exist = false;
    vector<Animal>::iterator it;
    it = Animals.begin();
    for (; it != Animals.end(); it++) {
        if (name.compare(it->name) == 0) {
            exist = true;
            break;
        }
```

```
        }
        if (exist)
            return it->num;
        else
            return 0;
    }
    void countAnimals() {
        cout << "There are " << species << " species, " << total << " animals in total."
<< endl;
    }
    void displayAnimals() {
        vector<Animal>::iterator it;
        it = Animals.begin();
        for (; it != Animals.end(); it++)
            cout << "There are " << it->num << " " << it->name << "(s)." << endl;
    }
};

int main()
{
    System sys;
    Animal cat("cat", 3);
    Animal dog("dog", 5);
    Animal horse("horse", 2);
    Animal fish("fish", 10);
    sys.addAnimal(cat);
    sys.addAnimal(fish);
    sys.addAnimal(horse);
    sys.addAnimal(dog);
    sys.displayAnimals();
    sys.countAnimals();
    sys.deleteAnimal("cat");
    sys.deleteAnimal("horse");
    cout << "Found " << sys.searchAnimal("cat") << " cat(s)."<< endl;
    cout << "Found " << sys.searchAnimal("fish") << " fish(s)." << endl;
    sys.displayAnimals();
    sys.countAnimals();
}
```

3. 书 Book 包含编号，书名，作者，出版年份这些信息(每本书有唯一编号，用数字表示)
图书查询机 Machine 包含所有书籍的信息，并提供如下方式来进行查询等操作
1) 系统自己扫描添加 num 本书并自动编号
2) 删除一本编号为 ID 的书籍(若无此书则无需操作)
3) 可以用书名查询书籍;

4) 按年份由近到远给书籍排序并输出
5) 按作者姓名查询该作者的书籍数量
图书馆书籍流动量大,经常大量地添加和删除书籍信息，添加/删除操作的速度很重要
实现书类 Book，自选两种 STL 容器来分别实现查询机类 Machine
设计程序测试系统大量扫描添加和删除书籍的操作的所需时间，看看哪种容器更适合
设计程序分别测试(3)~(5)功能

```
#include <iostream>
#include <string>
#include <vector>
#include <list>
#include <algorithm>
#include <ctime>
#include <cstdlib>
using namespace std;

class Book {
    int number;
    string name;
    string author;
    int year;
    friend class MachineVector;
    friend class MachineList;
    friend class MachineMap;
public:
    Book(int num, string n, string a, int y) {
        number = num;
        name = n;
        author = a;
        year = y;
    }
};

class MachineVector {
    vector<Book> Books;
    int number;
public:
    MachineVector() {
        vector<Book> Books;
        number = 1000;
    }
    void addBook2(int num, string name, string author, int year) {
        for (int i = 0; i < num; i++)
            Books.push_back(Book(number + i, name, author, year));
```

```
        number += num;
    }
    void deleteBook(int ID) {
        bool exist = false;
        vector<Book>::iterator it;
        it = Books.begin();
        for (; it != Books.end(); it++) {
            if (it->number == ID) {
                exist = true;
                break;
            }
        }
        if (exist) {
            Books.erase(it);
        }
    }
    void searchBook(string name) {
        bool exist = false;
        int count = 0;
        vector<int> id;
        vector<Book>::iterator it;
        it = Books.begin();
        for (; it != Books.end(); it++) {
            if (name.compare(it->name) == 0) {
                exist = true;
                count += 1;
                id.push_back(it->number);
            }
        }
        cout << "Found " << count << " " << name << "(s)." << endl;
        if (exist) {
            cout << "Their ID are: ";
            for_each(id.begin(), id.end(), [](int i) {cout << i << ' '; });
            cout << endl;
        }
    }
    void sortAndDisplay() {
        sort(Books.begin(), Books.end(), [](Book b1, Book b2) { return b1.year <
b2.year; });
        for_each(Books.begin(), Books.end(), [](Book b) {
            cout << b.number << '\t' << b.name << '\t' << b.author << '\t' << b.year << endl;
            }
        );
    }
```

```
    void searchAuthor(string author) {
        bool exist = false;
        int count = 0;
        vector<int> id;
        vector<Book>::iterator it;
        it = Books.begin();
        for (; it != Books.end(); it++) {
            if (author.compare(it->author) == 0) {
                exist = true;
                count += 1;
                id.push_back(it->number);
            }
        }
        cout << "Found " << count << " book(s) by " << author << "." << endl;
        if (exist) {
            cout << "Their ID are: ";
            for_each(id.begin(), id.end(), [](int i) {cout << i << ' '; });
            cout << endl;
        }
    }
};

class MachineList {
    list<Book> Books;
    int number;
public:
    MachineList() {
        list<Book> Books;
        number = 1000;
    }
    void addBook2(int num, string name, string author, int year) {
        for (int i = 0; i < num; i++)
            Books.push_back(Book(number + i, name, author, year));
        number += num;
    }
    void deleteBook(int ID) {
        bool exist = false;
        list<Book>::iterator it;
        it = Books.begin();
        for (; it != Books.end(); it++) {
            if (it->number == ID) {
                exist = true;
                break;
            }
```

```
    }
    if (exist) {
        Books.erase(it);
    }
}
void searchBook(string name) {
    bool exist = false;
    int count = 0;
    vector<int> id;
    list<Book>::iterator it;
    it = Books.begin();
    for (; it != Books.end(); it++) {
        if (name.compare(it->name) == 0) {
            exist = true;
            count += 1;
            id.push_back(it->number);
        }
    }
    cout << "Found " << count << " " << name << "(s)." << endl;
    if (exist) {
        cout << "Their ID are: ";
        for_each(id.begin(), id.end(), [](int i) {cout << i << ' '; });
        cout << endl;
    }
}
void sortAndDisplay() {
    Books.sort([](Book b1, Book b2) { return b1.year < b2.year; });
    for_each(Books.begin(), Books.end(), [](Book b) {
        cout << b.number << '\t' << b.name << '\t' << b.author << '\t' << b.year << endl;
        }
    );
}
void searchAuthor(string author) {
    bool exist = false;
    int count = 0;
    vector<int> id;
    list<Book>::iterator it;
    it = Books.begin();
    for (; it != Books.end(); it++) {
        if (author.compare(it->author) == 0) {
            exist = true;
            count += 1;
            id.push_back(it->number);
        }
```

```
        }
        cout << "Found " << count << " book(s) by " << author << "." << endl;
        if (exist) {
            cout << "Their ID are: ";
            for_each(id.begin(), id.end(), [](int i) {cout << i << ' '; });
            cout << endl;
        }
    }
};

int main()
{
    clock_t a1, b1;
    a1 = clock();
    MachineVector mac_v;
    mac_v.addBook2(100, "Japan & Self Existence", "Mick Karn", 2009);
    mac_v.addBook2(50, "Bit of a Blur", "Alex James", 2005);
    mac_v.addBook2(50, "The Seed and the Sower", "Lawrence Van der Post", 1983);
    mac_v.addBook2(100, "A Far Off Place", "Lawrence Van der Post", 1993);
    mac_v.addBook2(200, "Turquoise Days: the Weird World of Echo & the Bunnymen", "Chris
Adams", 1997);
    mac_v.deleteBook(1050);
    mac_v.deleteBook(1100);
    mac_v.deleteBook(3000);
    mac_v.searchBook("Bit of a Blur");
    mac_v.searchBook("Turquoise Days: the Weird World of Echo & the Bunnymen");
    mac_v.searchBook("Call Me by Your Name");
    mac_v.sortAndDisplay();
    mac_v.searchAuthor("Mick Karn");
    mac_v.searchAuthor("Lawrence Van der Post");
    mac_v.searchAuthor("Philip K. Dick");
    b1 = clock();
    double d1 = (double)(b1 - a1) / CLOCKS_PER_SEC;

    clock_t a2, b2;
    a2 = clock();
    MachineList mac_l;
    mac_l.addBook2(10, "Japan & Self Existence", "Mick Karn", 2009);
    mac_l.addBook2(5, "Bit of a Blur", "Alex James", 2005);
    mac_l.addBook2(50, "The Seed and the Sower", "Lawrence Van der Post", 1983);
    mac_l.addBook2(100, "A Far Off Place", "Lawrence Van der Post", 1993);
    mac_l.addBook2(20, "The Weird World of Echo & the Bunnymen", "Chris Adams", 1997);
    mac_l.deleteBook(1050);
    mac_l.deleteBook(1100);
```

```
    mac_l.deleteBook(3000);
    mac_l.searchBook("Bit of a Blur");
    mac_l.searchBook("The Weird World of Echo & the Bunnymen");
    mac_l.searchBook("Call Me by Your Name");
    mac_l.sortAndDisplay();
    mac_l.searchAuthor("Mick Karn");
    mac_l.searchAuthor("Lawrence Van der Post");
    mac_l.searchAuthor("Philip K. Dick");
    b2 = clock();
    double d2 = (double)(b2 - a2) / CLOCKS_PER_SEC;

    cout << d1 << ' ' << d2 << endl;

    return 0;
}
```

输出结果: …… d1=2.386, d2=1.772
结论: 在用 vector 与 list 的两种实现中, list 更合适.