

# 中山大学移动信息工程学院本科生实验报告

(2015-2016 学年春季学期)

课程名称: Data structures and algorithms

任课教师: 张子臻、黄淦

年级	15	专业 (方向)	软件工程(移动信息工程)
学号	15352408	姓名	张稼伟
电话	13531810182	Email	709075442@qq.com
开始日期	2016. 6. 6	完成日期	2016. 6. 6

## 1. 实验题目

### 1000:

本题练习 STL list 的一些基本操作, 包括 push\_back int, push\_front int, pop\_back, pop\_front 和链表的遍历。注意: 请用迭代器 list<int>::iterator 进行链表的遍历。

### 1001:

有 n 个学生, 每个学生有一个分数。使用一个类或者结构体来管理学生的信息, 使用 vector 来存储 n 个学生的信息, 使用 stl 中的 sort 以分数为关键字从小到大进行排序。有 m 个询问, 每个询问一个整数 x, 输出排第 x 名的学生的相关信息。

### 1002:

实现一个 SetOperation 类, 使其可以求两个集合的交集合并集。  
集

---

## 2. 实验目的

- A. 练习 STL 中的 list 的一些基本操作。
- B. 练习使用模板类 vector 以及使用 STL 中的 sort 函数进行排序。
- C. 练习使用 STL 中的 set。

## 3. 程序设计

**1000:**

push\_back: 向链表末尾插入元素。

push\_front: 向链表头插入元素。

print: 使用迭代器遍历 list, 并输出相应的元素。

pop\_back: 删除链表末尾的元素。

pop\_front: 删除链表头的元素。

```
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
list<int> Q;
int main(){
    int n,x;
    string op;
    while (cin>>n){
        Q.clear();
        rep(i,1,n){
            cin>>op;
            if (op=="push_back"){
                cin>>x;
                Q.push_back(x);
            }
            else if (op=="push_front"){
                cin>>x;
                Q.push_front(x);
            }
            else if (op=="print"){
                list<int>::iterator it;
                cout<<"List:"<<endl;
                for (it=Q.begin();it!=Q.end();it++)
                    cout<<(*it)<<endl;
            }
            else if (op=="pop_back")Q.pop_back();
            else if (op=="pop_front")Q.pop_front();
        }
    }
    return 0;
}
```

### 1001:

- A.采用结构体储存一个学生的姓名和分数。
- B.结构体中以分数为关键字重载 operator <。
- C.使用 push\_back 将代表每个学生的结构体插入 vector 的末尾。
- D.使用 STL 中的 sort 函数对 vector 中的元素进行排序。

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
struct Student{
    string name;
    int score;
    Student(string name ,int score ){
        name=name ;
        score=score ;
    }
    bool operator < (const Student B) const{
        return score<B.score;
    }
};
vector<Student>myVector;
int main(){
    int n,m,x;
    string s;
    cin>>n>>m;
    for (int i=1;i<=n;i++){
        cin>>s>>x;
        myVector.push_back(Student(s,x));
    }
    sort(myVector.begin(),myVector.end());
    for (int i=1;i<=m;i++){
        cin>>x;
        x--;
        cout<<myVector[x].name<<" "<<myVector[x].score<<endl;
    }
    return 0;
}
```

### 1002:

- A.采用 insert 操作将元素插入 set。
- B.求两个集合的并集，因为 set 是集合，会自动去重，所以我们直接把两个集合的元素都插入到一个新的 set 里就能得到并集。
- C.求两个集合的交集时，我们可以遍历其中一个集合 setA,然后对其中的每个元素，在另一个集合 setB 中用 find 函数去查找，若能成功找到，则将其插入到一个新的集合中，若查找失败，find 函数会返回 setB 的 end 地址。

```

class SetOperation{
public:
    SetOperation(int a[],int sizeA,int b[],int sizeB){
        setA.clear();
        setB.clear();
        for (int i=0;i<sizeA;i++) setA.insert(a[i]);
        for (int i=0;i<sizeB;i++) setB.insert(b[i]);
    }
    set<int> Union(){
        set<int> setC;
        setC.clear();
        for (set<int>::iterator it = setA.begin(); it != setA.end(); it++){
            setC.insert(*it);
        }
        for (set<int>::iterator it = setB.begin(); it != setB.end(); it++){
            setC.insert(*it);
        }
        return setC;
    }
    set<int> Intersection(){
        set<int> setC;
        setC.clear();
        for (set<int>::iterator it = setA.begin(); it != setA.end(); it++){
            set<int>::iterator it_ = setB.find(*it);
            if (it_ != setB.end()) setC.insert(*it_);
        }
        return setC;
    }
private:
    set<int> setA;
    set<int> setB;
};

```

## 4. 程序运行与测试

1000:

测试一:

```

8
push_back 1
push_back 2
push_front 3
print
List:
3
1
2
pop_back
print
List:
3
1
pop_front
print
List:
1

```

---

### 测试二:

```
8
push_back 8
push_front 2
print
List:
2
8
pop_front
print
List:
8
push_back 3
pop_back
print
List:
8
-
```

### 测试三:

```
10
push_front 10
push_front 5
push_back 3
print
List:
5
10
3
pop_back
print
List:
5
10
pop_front
print
List:
10
push_back 11
print
List:
10
11
-
```

---

## 1001:

### 测试一:

```
4 4
jimmy 70
alice 50
peter 80
katty 60
1
alice 50
2
katty 60
3
jimmy 70
4
peter 80
请按任意键继续. . .
```

### 测试二:

```
4 4
a 1
b 56
c 7
d 88
2
c 7
3
b 56
1
a 1
4
d 88
请按任意键继续. . .
```

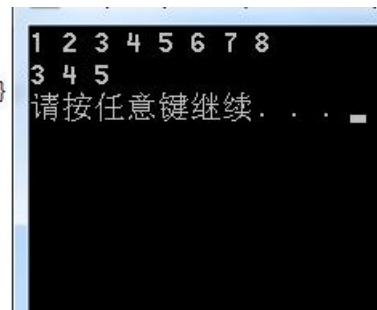
### 测试三:

```
5 5
sdfsdf 12
dsfdf 23
sdf 10
dsfsd 60
df 70
5
df 70
4
dsfsd 60
3
dsfdf 23
2
sdfsdf 12
1
sdf 10
请按任意键继续. . .
```

## 1002:

### 测试一:

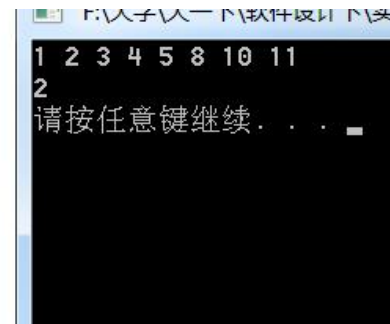
```
5
7 int main(int argc, char *argv[]) {
3   int a[7] = {8, 7, 5, 3, 1, 4, 6};
3   int b[4] = {2, 3, 5, 4};
0   SetOperation s1(a, 7, b, 4);
1   printSet(s1.Union());
2   printSet(s1.Intersection());
3   return 0;
4 }
```



```
1 2 3 4 5 6 7 8
3 4 5
请按任意键继续. . .
```

### 测试二:

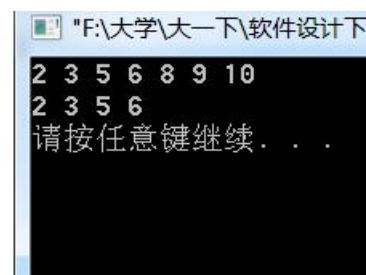
```
int main(int argc, char *argv[]) {
    int a[7] = {1, 2, 3, 10, 11, 2, 8};
    int b[4] = {2, 4, 5, 4};
    SetOperation s1(a, 7, b, 4);
    printSet(s1.Union());
    printSet(s1.Intersection());
    return 0;
}
```



```
1 2 3 4 5 8 10 11
2
请按任意键继续. . .
```

### 测试三:

```
int main(int argc, char *argv[]) {
    int a[7] = {2, 3, 5, 6, 8, 9, 10};
    int b[4] = {2, 3, 5, 6};
    SetOperation s1(a, 7, b, 4);
    printSet(s1.Union());
    printSet(s1.Intersection());
    return 0;
}
```



```
2 3 5 6 8 9 10
2 3 5 6
请按任意键继续. . .
```

## 5. 实验总结与心得

- 1.使用 STL 的容器一定要注意清空，特别是在多组数据或者重复使用同一个容器的时候，如果不想被以前存入容器的元素所影响结果，要先调用 clear()函数清空容器。
- 2.对于可以对容器进行遍历的迭代器，其声明方式为“容器类型<数据类

---

型>::iterator 迭代器名”。一般来说迭代器名取 it。要注意迭代器在这里相当于指针，it 不指代具体的元素，\*it 才是具体元素。

**3.List** 将元素按顺序储存在链表中。与向量(vector)相比，它允许快速的插入和删除，但是随机访问却比较慢。本次实验涉及的操作有 clear()：删除所有元素；begin()：返回指向第一个元素的迭代器；end() 返回末尾的迭代器。pop\_back()：删除最后一个元素；pop\_front()：删除第一个元素；push\_back()：在 list 的末尾添加一个元素；push\_front()：在 list 的头部添加一个元素。

**4.vector** 是一个能够存放任意类型的动态数组，可以使用 push\_back 在尾部添加数据。

**5.sort** 是 STL 中的排序函数，它可以对给定区间所有元素进行排序，实现原理是快速排序。使用方法是 sort(a,b)；其中 a 是指一个区间的左端点，b 是指一个区间的右端点+1，简称“左闭右开”。例如对数组 a[5]的 5 个元素进行排序，则这样使用：sort(a,a+5)；此外，sort 还有第三个参数，是自己写的比较函数，假如自己写了一个比较函数 cmp，则再对数组 a[5]排序的写法变为 sort(a,a+5,cmp)；

**6.set 集合容器**：实现了红黑树的平衡二叉检索树的数据结构，插入元素时，它会自动调整二叉树的排列，把元素放到适当的位置，以保证每个子树根节点键值大于左子树所有节点的键值，小于右子树所有节点的键值；另外，还得保证根节点左子树的高度与右子树高度相等。平衡二叉检索树使用中序遍历算法，检索效率高于 vector、deque 和 list 等容器，另外使用中序遍历可将键值按照从小到大遍历出来。构造 set 集合主要目的是为了快速检索，不可直接去修改键值，同时 set 中的元素是唯一的。

## 附录、提交文件清单

实验报告一份：实验报告.pdf

代码三份：1000.cpp、1001.cpp、1002.cpp