山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

学号：202000130198	姓名： 隋春雨	班级： 20.4
实验题目：排序算法		
实验学时：2	实验日期： 2021-10-14	

实验目的：

掌握各种简单排序算法。如：冒泡、选择、插入等排序，并掌握及时终止的排序。

软件开发环境：

CLION2020

1. 实验内容

1、题目描述：

设通讯录中每一个联系人的内容有：姓名、电话号码、班级、宿舍。由标准输入读入联系人信息，使用线性表中操作实现通讯录管理功能，包括：插入、删除、编辑、查找（按姓名查找）；键盘输入一班级，输出通讯录中该班级中所有人的信息。

每个操作的第一个数为操作数(插入-0，删除-1，编辑-2，查找-3，输出一个班所有人员信息-4)，具体格式如下:

- 0 姓名 电话 班级 宿舍 插入一条记录
- 1 姓名 根据姓名删除一条记录
- 2 姓名 编辑项目 项目新值 根据姓名编辑一条记录(编辑项目为 1 到 3 的整数，1 代表编辑电话，2 代表编辑班级，3 代表编辑宿舍)
- 3 姓名 根据姓名查找，找到输出 1，未找到输出 0
- 4 班级 输出该班级的所有成员的宿舍号的异或值

其中查找操作当找到相应的人时输出 1，未找到输出 0。输出一个班级的人员信息时输出所有成员的宿舍号的异或值。输入数据保证合法。

输入输出格式:

输入:

第一行一个 n(1<=n<=20000)，代表接下来操作的数目。接下来 n 行代表各项操作。

输出:

当遇到查找和输出一个班所有人员信息操作时输出。

2. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）

(1) 结合面向对象的思想，需要创建两个类，一个是 Person 类，一个是 Address 类，其中 Address 类的私有成员有 Person 数组，插入、删除等操作通过调用 Address 类的 Public 函数进行实现。

(2) 插入操作：首先检查 Person 数组是否已经满了，如果满了，则扩容。

```
if (size == capacity)
{
    Person* new_ptr = new Person[size == 0 ? 1 : 2 * size];
```

并且插入操作通过调用 Person 数组成员的 edit 函数来实现。

(3) 删除操作：首先需要根据姓名确定想要删除的人在数组中的位置，然后找到后，将其后面的成员依次向前移动，并且需要注意：一定一定要记得更新私有成员变量的值，否则之后的操作会出错。因为类只有通过自己私有成员才知道自己所处的状态，如果私有成员的值不对，那么操作也很难对。

```
for (int i = pos + 1; i < size; i++)//移动处于被删除的人后面的人
{
    ptr[i - 1] = ptr[i];
}
size--;
```

(4) 编辑操作：首先通过 Address 的共有接口，进行编辑信息的读取，然后通过不同的变量值，进行不同的操作，通过姓名找到那个人，然后调用 Person 类的共有接口函数中的编辑操作，完成编辑。

(5) 查找操作： 遍历一遍 Person 数组，遍历的终止条件为小于数组的 size,通过调用共有函数接口的 name 函数进行判断。最后输出

(6) 输出操作：输出所有成员的宿舍号异或值，因为宿舍号保存的时候是一个 string，首先将其转换为 const char*,然后再调用 atoi 函数进行异或。

3. 测试结果（测试输入，测试输出）

输入：

```
28
0 Evan 57298577609 1 65
0 WINNIE 37367348390 4 1
3 Evan
4 6
3 WINNIE
1 Evan
4 7
1 WINNIE
3 MARYAM
3 CAMERON
3 TZIVIA
0 OMAR 16447001130 6 55
4 8
4 2
3 JADEN
3 ELIZABETH
2 OMAR 1 79409905568
3 JOSHUA
2 OMAR 1 8978214817
1 OMAR
3 Azaan
3 MARIA
0 HANNAH 94060479192 5 98
3 HEIDY
1 HANNAH
0 Axel 92066832927 3 70
1 Axel
3 TIFFANY
```

输出：

```
0 HANNAH 9406047919Z 5 98
3 HEIDY
1 HANNAH
0 Axel 92066832927 3 70
1 Axel
3 TIFFANY
1
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0

进程已结束，退出代码为 0
```

结果：

RANK	PARTICIPANT	SCORE	A
1	♡ **202000130198** 隋春雨	**1** 100	100 95

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

(1) 自己写的时候测的样例都是对的，交到 oj 平台上就 RE 了，怎么办？
解决：RE 常见情况的是数组下标越界，但是经过自己 debug 发现，实际情况是 switch case 条件没有 break 语句，才 RE，在平时，能用 switch case 尽量用 switch case 而不是 If else ，因为 switch case 执行的次数少。

(2) 宿舍号储存的是 string，该怎么将其转换为 int 呢？
解决：首先我们可以一个一个读取来转换，但是我们也可以用简便的方法，首先调用 str()函数，然后再调用 atoi 函数，比较方便。

(3) 在测试样例的时候发现自己的输出值跟预期不同，怎么办？
解决：经过 debug 发现，在删除操作的时候，对于数组的 size 变量没有更新，从而导致错误。以后在写函数的时候，一定需要注意的一点就是调用更新私有变量成员。

```
if (ptr[pos].name() == name)
{
    for (int i = pos + 1; i < size; i++)//移动处于被删除的人后面的人
    {
        ptr[i - 1] = ptr[i];
    }
    size--;//更新私有变量的值
}
```

(4) 一个一个写操作很麻烦怎么办？
解决：运用面向对象的思想，将函数封装为类内函数，以后只需要调用类内函数即可进行操作。

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
#include <iostream>
#include<string>
using namespace std;
class Person {
    string _name;
    string _tel;
    string _class;
    string _dorm;
public:
    Person() {}
    Person(const string& name, const string& tel, const string& Class, const string& dorm) :
        _name(name), _tel(tel), _class(Class), _dorm(dorm) {}

    virtual ~Person() {}

    void change(const string& new_name, const string& new_tel, const string& new_Class, const string& new_dorm)
    {//编辑操作
```

```
            _name = new_name;
            _tel = new_tel;
            _class = new_Class;
            _dorm = new_dorm;
        }
        void edit(int identifier, string new_val)//更改操作
        {
            switch (identifier)
            {
            case 1://要注意这里一定一定要break,否则会re
                _tel = new_val;
                break;
            case 2:
                _class = new_val;
                break;
            case 3:
                _dorm = new_val;
                break;
            default:
                throw "error";

            }
        }
        string name() const
        {
            return _name;
        }

        string Class() const
        {
            return _class;
        }

        string dorm() const
        {
            return _dorm;
        }


};

class Address_book
{
        Person* ptr;
        int size;//数组有效位数
        int capacity;//
```

```
public:
    Address_book() :ptr(new Person[1]), size(0), capacity(1) {}
    Address_book(int capacity) : ptr(new Person[capacity]), size(0), capacity(capacity) {}
    virtual ~Address_book()
    {
        delete[]ptr;
        ptr = nullptr;
    }

    void insert(string name, string tel, string Class, string dorm)//插入操作
    {
        if (size == capacity)
        {
            Person* new_ptr = new Person[size == 0 ? 1 : 2 * size];//特判0的情况
            for (int i = 0; i < size; i++)
            {
                new_ptr[i] = ptr[i];
            }
            delete[]ptr;
            ptr = new_ptr;
            new_ptr = nullptr;
            capacity *= 2;
        }
        ptr[size++].change(name, tel, Class, dorm);
    }

    void edit(string name, int identifier, string new_val)//编辑操作
    {

        for (int pos = 0; pos < size; pos++)
        {
            if (ptr[pos].name() == name)
            {
                ptr[pos].edit(identifier, new_val);//调用Person类的编辑函数

            }
        }

    }

    void erase(string name)//删除操作
    {

        for (int pos = 0; pos < size; pos++)
        {
            if (ptr[pos].name() == name)
```

```
                {
                    for (int i = pos + 1; i < size; i++)//移动处于被删除的人后面的人
                    {
                        ptr[i - 1] = ptr[i];
                    }
                    size--;//更新私有变量的值
                }
            }

        }

        void find(string name) const//查找操作
        {
            bool flag = 0;
            for (int i = 0; i < size; i++)
            {
                if (ptr[i].name() == name)
                {
                    flag = 1;
                    break;
                }
            }
            cout << (flag == 1 ? 1 : 0) << endl;
        }

        void output(string _class) const//输出宿舍号的异或值
        {
            long long    ans = 0;//0和任何数异或都是自身
            for (int i = 0; i < size; i++)
            {
                if (ptr[i].Class() == _class)
                {
                    ans ^= atoi(ptr[i].dorm().c_str());
                }

            }
            cout << ans << endl;//输出异或值

        }
};


int main()
{

    Address_book a;//创建通讯录
```

```
    int n;
    cin >> n;
    string name;
    string tel;
    string Class;
    string dorm;
    string new_val;
    int jud;
    for (int i = 0; i < n; i++) {

        int identifier;
        cin >> identifier;
        switch (identifier)
        {
        case 0://插入操作
            cin >> name >> tel >> Class >> dorm;
            a.insert(name, tel, Class, dorm);
            break;
        case 1://删除操作
            cin >> name;
            a.erase(name);
            break;
        case 2://编辑操作
            cin >> name >> jud >> new_val;
            a.edit(name, jud, new_val);
            break;
        case 3://查找操作
            cin >> name;
            a.find(name);
            break;
        case 4://输出宿舍号的异或值
            cin >> Class;
            a.output(Class);
            break;
        }
    }
    return 0;
}
```