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　　：202200400053	姓名：　王宇涵	班级：　22 级 2 班
实验题目：数组描述线性表		
实验学时：2	实验日期：　　2023-9-27	

实验目的

1.掌握线性表结构、数组描述方法（顺序存储结构）、数组描述线性表的实现。

2.掌握线性表应用。

软件开发环境：

Vscode

1. 实验内容

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）

整体框架

1. 定义数据结构：

- 代码定义了一个名为 `Info` 的结构体，用于存储联系人信息，包括姓名、电话号码、班级和寝室号码。

2. 定义类 `AddressBook`：

-`AddressBook` 类用于管理地址簿。它包含一个动态数组 `Infos` 来存储联系人信息，以及其他私有成员变量用于跟踪数组的长度和大小。

- 构造函数 `AddressBook(int theCapacity)` 用于初始化地址簿，接受一个参数来指定地址簿的容量。

- `push_back()` 方法用于添加新的联系人信息到地址簿。

- `erase(int theIndex)` 方法用于删除指定索引位置的联系人信息。

- `findIndexWithName(string name)` 方法根据姓名查找联系人的索引位置。

- `outputClass(string classNumber)` 方法根据班级号码输出寝室号码的异或值。

- `editMember()` 方法用于编辑联系人信息，可以选择编辑电话号码、班级或寝室号码。

3. `main()` 函数：

- `main()` 函数首先读取输入的整数 `n`，表示接下来会有 `n` 个操作。

- 然后创建一个 `AddressBook` 类的实例 `s`，并根据用户输入的操作执行相应的操作，包括添加、删除、编辑、查找和输出班级信息。

- 使用 `switch` 语句来根据不同的操作类型执行相应的操作。

具体函数:

- `push_back()`: 这个操作用于添加联系人信息。用户输入姓名、电话号码、班级和寝室号码，然后将这些信息存储在 `Info` 结构体中，并将其添加到 `Infos` 数组中。地址簿的 `listSize` 增加，表示联系人数量增加了。

- `erase(int theIndex)`: 通过这个操作，用户可以删除指定索引位置的联系人信息。使用 `copy` 函数将该索引位置后面的联系人信息向前移动，然后减小 `listSize`，表示联系人数量减少了。

- `findIndexWithName(string name)`: 这个方法允许用户根据姓名查找联系人的索引位置。它遍历 `Infos` 数组，逐一比较联系人的姓名，如果找到匹配的姓名，则返回该联系人的索引位置，否则返回 -1 表示未找到。

- `outputClass(string classNumber)`: 用户可以根据班级号码查找联系人，并计算寝室号码的异或值。它创建一个临时数组 `tmp` 来存储匹配班级号码的联系人的寝室号码，然后对这些寝室号码执行异

或操作，并输出结果。

- `editMember()`: 这个操作允许用户编辑联系人信息。用户可以选择编辑电话号码、班级或寝室号码，然后根据姓名查找联系人并更新相应的信息。

- `main()`: 主函数通过读取整数 `n` 来确定接下来会执行多少个操作。然后创建一个 `AddressBook` 类的实例 `s`，根据用户输入的操作类型执行相应的操作，包括：

  - 0：添加联系人信息。

  - 1：删除联系人信息。

  - 2：编辑联系人信息。

  - 3：查找联系人信息。

  - 默认情况下：根据班级号码输出寝室号码的异或值。

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:

1

0

1

0

0

0

0

0

0

0

0

0

0

0

0

0

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

内存管理问题：代码使用动态分配的数组来存储联系人信息，但没有实现析构函数来释放内存。这可能导致内存泄漏。应该在类的析构函数中释放动态分配的内存，以避免资源泄漏。

没有输入验证：代码没有对用户的输入进行任何验证或错误处理。用户可以输入无效的数据，例如非法的电话号码、班级号码或寝室号码，这可能导致程序出现异常行为。应该在输入数据之前添加验证和错误处理机制。

未处理数组越界：在 erase 函数中使用 copy 函数时，没有检查索引范围是否有效。如果 theIndex

大于或等于 listSize，则会导致数组越界。应该添加越界检查以确保操作的安全性。

查找联系人的效率问题：在 findIndexWithName 方法中，采用了线性搜索的方式来查找联系人，这在大型地址簿中可能会效率较低。可以考虑使用更高效的数据结构，如哈希表或二叉搜索树来加速查找操作。

错误处理不充分：在 main 函数中，当用户输入的操作不在 0 到 3 之间时，默认情况下会输出班级信息。然而，如果用户输入了无效的操作，程序应该给出明确的错误提示而不是默认操作。

不支持动态调整容量：代码中的 Infos 数组容量在构造函数中固定，一旦超过容量限制，就无法添加更多的联系人。应该考虑实现动态扩展数组的功能，以适应更多的联系人

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<algorithm>
using namespace std;
typedef long long ll;
int n;
struct Info
{
    string name;
    string teleNumber;
    string classNumber;
    ll domitoryNumber;
};

class AddressBook
{
public:
AddressBook(int theCapacity);
~AddressBook(){delete []Infos;}
void push_back();
void erase(int theIndex);
int   findIndexWithName(string name);
void outputClass(string classNumber);
void editMember();
```

```
private:
	int arrayLength;
	int listSize=0;
	Info* Infos;
};

AddressBook::AddressBook(int theCapacity)
{
	arrayLength=theCapacity;
	Infos=new Info[arrayLength];
}

void AddressBook::push_back()
{
	string name,teleNumber,classNumber;
	ll domitoryNumber;
	cin>>name>>teleNumber>>classNumber>>domitoryNumber;
	Info theInfo={name,teleNumber,classNumber,domitoryNumber};
	Infos[listSize++]=theInfo;
}

void AddressBook::erase(int theIndex)
{
	copy(Infos+theIndex+1,Infos+listSize,Infos+theIndex);
	listSize--;
}

//用姓名查找
int	AddressBook::findIndexWithName(string name)
{
	for(int i=0;i<listSize;i++)
	{
```

```
            auto theInfo=Infos[i];
            if (theInfo.name==name)
            return i;
        }
        return -1;
}

void AddressBook::outputClass(string classNumber)
{
        ll tmp[arrayLength];
        int index=0;
        for(int i=0;i<listSize;i++)
        {
            auto theInfo=Infos[i];
            if(theInfo.classNumber==classNumber)
            tmp[index++]=theInfo.domitoryNumber;
        }
        int value=0;
        for(int i=0;i<index;i++)
        value^=tmp[i];
        cout<<value<<endl;
}

void AddressBook::editMember()
{
        string name;
        int num;
        cin>>name>>num;
        if(num==1)
        {
            string teleNumber;
            cin>>teleNumber;
```

```
            Infos[findIndexWithName(name)].teleNumber=teleNumber;
        }
        else if(num==2)
        {
            string classNumber;
            cin>>classNumber;
            Infos[findIndexWithName(name)].classNumber=classNumber;
        }
        else
        {
            ll domitoryNumber;
            cin>>domitoryNumber;
            Infos[findIndexWithName(name)].domitoryNumber=domitoryNumber;
        }
}
int main()
{
    cin>>n;
    AddressBook s(n);
    int op;
    while(n--)
    {
        cin>>op;
        switch (op)
        {
            case 0:
            {
                s.push_back();
                break;
            }
            case 1:
            {
```

```
                string name;
                cin>>name;
                s.erase(s.findIndexWithName(name));
                break;
            }
            case 2:
            {
                s.editMember();
                break;
            }
            case 3:
            {
                string name;
                cin>>name;
                int theIndex=s.findIndexWithName(name);
                if(theIndex==-1)cout<<0<<endl;
                else cout<<1<<endl;
                break;
            }
            default:
            {
                string classNumber;
                cin>>classNumber;
                s.outputClass(classNumber);
                break;
            }

        }
    }
    return 0;
}
```