**附件：**

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：202000130138 | 姓名：宋璎航 | | 班级： 20.3 |
| 实验题目：数组描述线性表 | | | |
| 实验学时：2 | | 实验日期： 2021/10/14 | |
| 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：  VSC | | | |
| 1. 实验内容   **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。   1. 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）   建立一个数组描述的线性表，内置函数分别实现插入、删除 、查找以及输出相应班级宿舍号异或和的功能，在main函数中用switch语句实现各种操作   1. 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   1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   测试结果均正确   1. 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* "bits/stdc++.h"  using namespace std;  struct Person  {      string name, tel\_num, doom\_id, class\_id;  };  template <class T>  class arrayList  {  protected:      T \*element;      int arrayLength;      int listSize;  public:      arrayList()      {          arrayLength = 20010;          element = new T[arrayLength];          listSize = 0;      }      arrayList(int arraylength);      ~arrayList() { delete[] element; }  *//ADT*      bool empty() const { *return* listSize == 0; }      int size() const { *return* listSize; }      void insert(const T &theElement);      void remove(string &thename);      void edit(string &thename, char &item, string &newitem);      void checkName(string &thename) const;      void outputDoom(string &thedoom) const;  };  template <class T>  arrayList<T>::arrayList(int arraylength)  {  *// if (arraylength < 1)*  *// {*  *//     ostringstream s;*  *//     s << "initial capacity = " << arraylength << ", but must be > 1";*  *//     throw illeagallength(s.str());*  *// }*      arrayLength = 20010;      element = new T[arrayLength];      listSize = 0;  }  template <class T>  void arrayList<T>::insert(const T &theElement)  {  *// if (arrayLength == listSize)*  *// {*  *//     T \*temp = element;*  *//     element = new T[arrayLength + 10];*  *//     arrayLength += 10;*  *//     copy(temp, temp + arrayLength, element);*  *//     delete[] temp;*  *// }*      element[listSize].name = theElement.name;      element[listSize].tel\_num = theElement.tel\_num;      element[listSize].class\_id = theElement.class\_id;      element[listSize].doom\_id = theElement.doom\_id;      ++listSize;  }  template <class T>  void arrayList<T>::remove(string &thename)  {  *for* (int i = 0; i < listSize; ++i)      {  *if* (element[i].name == thename)          {              copy(element + i + 1, element + listSize, element + i);              --listSize;  *return*;          }      }  }  template <class T>  void arrayList<T>::edit(string &thename, char &item, string &newitem)  {  *for* (int i = 0; i < listSize; ++i)      {  *if* (element[i].name == thename)          {  *switch* (item)              {  *case* 1:                  element[i].tel\_num = newitem;  *break*;  *case* 2:                  element[i].class\_id = newitem;  *break*;  *case* 3:                  element[i].doom\_id = newitem;  *break*;  *default*:  *break*;              }  *return*;          }      }  }  template <class T>  void arrayList<T>::checkName(string &thename) const  {  *for* (int i = 0; i < listSize; ++i)      {  *if* (element[i].name == thename)          {              cout << 1 << endl;  *return*;          }      }      cout << 0 << endl;  }  template <class T>  void arrayList<T>::outputDoom(string &theclass) const  {      int ans = 0;  *for* (int i = 0; i < listSize; ++i)      {  *if* (element[i].class\_id == theclass)              ans ^= atoi(element[i].doom\_id.c\_str());      }      cout << ans << endl;  }  int main()  {      arrayList<Person> p;      int n;      cin >> n;  *for* (int i = 0; i < n; ++i)      {          int k;          cin >> k;  *switch* (k)          {  *case* 0:          {              Person newone;              cin >> newone.name >> newone.tel\_num >> newone.class\_id >> newone.doom\_id;              p.insert(newone);          }  *break*;  *case* 1:          {              string thename;              cin >> thename;              p.remove(thename);          }  *break*;  *case* 2:          {              string thename;              char item;              string newitem;              cin >> thename >> item >> newitem;              p.edit(thename, item, newitem);          }  *break*;  *case* 3:          {              string thename;              cin >> thename;              p.checkName(thename);          }  *break*;  *case* 4:          {              string theclass;              cin >> theclass;              p.outputDoom(theclass);          }  *break*;          }      }  } | | | |