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：201900161140 | 姓名： 张文浩 | | 班级： 19级人工智能 |
| 实验题目：数组描述线性表 | | | |
| 实验学时：2 | | 实验日期： 9.25 | |
| 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。 | | | |
| 软件开发工具：  Vscode | | | |
| 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。   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）   利用数组作为线性表结构，自定义结构体student用于记录学生的姓名、电话、班级、宿舍信息内容，然后用这个结构体做一个数组。然后编写相应的函数对这个数组进行操作。  插入insert操作：在数组最后一个结构体中将姓名、电话、班级、宿舍设置为输入的信息，然后将学生总数tot+1.  删除delet操作：从数组头0处开始遍历数组，直到找到与输入名字拥有相同名字的student，利用copy命令，将后面的student向前移动一个位置，直接将这个想要删除的学生覆盖掉，完成删除操作。  编辑edit操作：从数组头0处开始遍历数组，直到找到与输入名字拥有相同名字的student，根据输入的op的数值，将该student的电话/班级/宿舍的值设置为输入的num。  查找search操作：从数组头0处开始遍历数组，如果其中有student的name与输入的\_name相同，就输出1，如果for循环结束，将所有student都遍历了一遍还没找到，返回0.  输出宿舍号的异或值output操作：初始化res为0，作为输出的结果。从数组头0处开始遍历数组，如果遍历到的student的classnum等于输入的\_classnum，就将res与这个student的宿舍号进行异或操作。遍历完所有student后输出res   1. 测试结果（测试输入，测试输出）     n=7，一共七步操作。  前两步插入两个学生，班级都是5班，zwh在2号宿舍，zjh在1号宿舍。  第三步将zwh的宿舍改为1号宿舍。  第四步查找zjh，找到了，输出1  第五步计算5班所有人宿舍号的异或值，即1^1=0  第六步删除zjh同学  第七步再次查找zjh，但没找到，输出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）   在本次实验中，我通过编写一个处理学生信息的线性表的程序，掌握了线性表结构、数组描述方法（顺序存储结构）、数组描述线性表的实现，学会了线性表的应用，并在实验中实现了对线性表操作的各种方法，包括插入、删除、查找、修改、计算等功能。让我对线性表的了解更加深刻。  线性表的构建，最关键的是对其容量的控制。要实现动态更新表的大小，需要在插入时对线性表中数据的个数和储存表的数组的实际大小进行比较。如果存储的数据数快要到达数组实际容量，采用成倍增加容量的方式会是一个比较好的解决方法。调用copy函数实现这个功能会比较简单。在最初的设计中，我采用了依次移动数组内元素向前一位的方式，但这种修改方法较为繁琐，直到查阅到了copy函数，一行代码即可实现更改。在编写程序时采用一些合适的函数，可以使得程序更加简洁明了。   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 <iostream>  #include <string>  using namespace std;  struct student  { //定义结构体存放每个学生的信息      string name;      long long phone;      long long classnum;      long long dorm;  };  class studentlist  {  public:      ~studentlist() { delete[] s; };      void init(student \*stu);                                                           //初始化类内的s数组      void insert(string \_name, long long \_phone, long long \_classnum, long long \_dorm); //插入一个学生信息      void delet(string \_name);                                                          //删除一个学生      void edit(string \_name, long long op, long long num);                              //修改一个学生信息      long long search(string \_name);                                                    //查找学生姓名      long long output(long long \_classnum);                                             //输出一个班学生宿舍号的异或值  private:      int tot; //当前一共有多少个学生      student \*s;  };  void studentlist::init(student \*stu)  {      tot = 0;      s = stu;  }  void studentlist::insert(string \_name, long long \_phone, long long \_classnum, long long \_dorm)  {      s[tot].name = \_name;      s[tot].phone = \_phone;      s[tot].classnum = \_classnum;      s[tot].dorm = \_dorm;      tot++;  }  void studentlist::delet(string \_name)  {      int i;      for (i = 0; i < tot; i++)      {          if (s[i].name == \_name)              break;      }      copy(s + i + 1, s + tot, s + i);      tot--;  }  void studentlist::edit(string \_name, long long op, long long num)  {      for (int i = 0; i < tot; i++)      {          if (s[i].name == \_name)          {              switch (op)              {              case 1:                  s[i].phone = num;                  break;              case 2:                  s[i].classnum = num;                  break;              case 3:                  s[i].dorm = num;              }          }      }  }  long long studentlist::search(string \_name)  {      for (int i = 0; i < tot; i++)      {          if (s[i].name == \_name)              return 1;      }      return 0;  }  long long studentlist::output(long long \_classnum)  {      long long res = 0;      for (int i = 0; i < tot; i++)      {          if (s[i].classnum == \_classnum)              res = res ^ s[i].dorm;      }      return res;  }  int main()  {      int n;      cin >> n;      studentlist l;      student \*stu = new student[n];      l.init(stu);      string name;      long long phone, num;      long long classnum, dorm, op, k;      for (int i = 0; i < n; i++)      {          cin >> k;          switch (k)          {          case 0:              cin >> name >> phone >> classnum >> dorm;              l.insert(name, phone, classnum, dorm);              break;          case 1:              cin >> name;              l.delet(name);              break;          case 2:              cin >> name >> op >> num;              l.edit(name, op, num);              break;          case 3:              cin >> name;              cout << l.search(name) << endl;              break;          case 4:              cin >> classnum;              cout << l.output(classnum) << endl;              break;          }      }      system("pause");      return 0;  } | | | |