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：202000130143	姓名： 郑凯饶	班级： 计科 20.1
实验题目：数组描述线性表		
实验学时：2	实验日期：	

实验目的：

1、掌握线性表的结构、数组描述方法（顺序存储结构）、数组描述线性表的实现。

2、掌握线性表的应用。

软件开发环境：

Windows 10 家庭中文版 64 位（10.0，版本 18363）

Dev-C++ IDE

1. 实验内容

利用线性表结构创建一个通讯录类，储存联系人的姓名、电话号码、班级、宿舍。使用线性表操作实现通讯录管理功能，包括：插入、删除、编辑、查找。

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）

我没有直接继承线性表，而是基于线性表的思想直接设计了通讯录类，设置了如下数据成员及方法满足题目要求。

```
template<class T>
class Alist {
    public:
        Alist(int C = 1000);
        Alist(const Alist<T>&);
        ~Alist() { delete [] ele; }
        void Push(T a);
        // not ADT
        int Find(string name);  // 返回索引即可
        int cal(int tar);       // 并不是 general的
        void Del(string Dtar);
        // set
        void setP(int ind, string np) { ele[ind].phone = np; }
        void setC(int ind, int nc) { ele[ind].clas = nc; }
        void setD(int ind, int nd) { ele[ind].dor = nd; }
        void chk() {
            for (int i = 0; i < Size; i++) {
                cout << ele[i].name << ' ' << ele[i].phone << ' ' << ele[i].clas << ' ' << ele[i].dor;
                cout << '\n';
            }
        }

    private:
        T* ele;
        int Cap, Size;  // 容量及当前大小
};
```

Find()：根据姓名查询联系人返回索引，若未找到返回-1。

cal()：以班级为检索条件查询联系人，并计算他们宿舍号的异或值。

Del()：在 Find()结果的基础上删除联系人，并将其后面的联系人 copy()到前面，填补空缺。

Set*()：设置索引的某个值。

Chk()：输出通讯录的所有内容。

3. 测试结果（测试输入，测试输出）

在 oj 平台上成功提交。

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

其实，这个设计并不好。我应该是先把线性表的数组描述实现了，再在其基础上设计通讯录类。实际上，模板类的作用发挥得并不是很好，大部分函数只适用于 T==stu.

完成代码的过程中我犯了一些错误，包括将变量“op1”写成了“op”，检查了很久。还有

使用 stu 类的析构函数，但是引发了错误，返回异常值“3221226356”。检查后发现是 Del()函数的问题：调用了缺省的~stu()，回收了这片空间，再增加联系人的时候会引发异常。

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<bits/stdc++.h>
using namespace std;

// 7 个输出

struct stu {
 string name, phone;
 int clas, dor;
// stu(string s1, string s2, int a1, int a2) {
//  name = s1; phone = s2; clas = a1; dor = a2;
// }
};

// 根据姓名、宿舍号 Find its index
// index < listSize

template<class T>
void cl(T*& a, int ol, int nl) {
 T* t = new T[nl];
 int num = min(ol, nl);
 copy(a, a + num, t);
 delete [] a;
 a = t;
}

template<class T>
class Alist {
 public:
  Alist(int C = 1000);
  Alist(const Alist<T>&);
  ~Alist() { delete [] ele; }
  void Push(T a);
  // not ADT
  int Find(string name); // 返回索引即可
  int cal(int tar);  // 并不是 general 的
  void Del(string Dtar);
  // set
  void setP(int ind, string np) { ele[ind].phone = np; }
  void setC(int ind, int nc) { ele[ind].clas = nc; }
  void setD(int ind, int nd) { ele[ind].dor = nd; }
//  void chk() {
```

```
47.//   for (int i = 0; i < Size; i++) {
48.//    cout << ele[i].name << ' ' << ele[i].phone << ' ' << ele[i].clas << ' ' <<
    ele[i].dor;
49.//    cout << '\n';
50.//   }
51.//  }
52.
53. private:
54.  T* ele;
55.  int Cap, Size; // 容量及当前大小
56.};
57.
58.// 实现
59.template<class T>
60.Alist<T>::Alist(int C) {
61. Cap = C;
62. ele = new T[Cap];
63. Size = 0;
64.}
65.
66.template<class T>
67.Alist<T>:: Alist(const Alist<T>& a) {
68. Cap = a.Cap;
69. Size = a.Size;
70. ele = new T[Cap];
71. copy(a.ele, a.ele + Size, ele);  // (复制对象，  复制长度，  目标空间)
72.}
73.
74.template<class T>
75.void Alist<T>::Push(T a) {
76. // 判断是否扩容
77. if (Size == Cap) {
78.  cl(ele, Cap, 2*Cap);
79.  Cap *= 2;
80. }
81.
82. ele[Size] = a;  // a: student 's example
83. Size++;
84.}
85.
86.template<class T>
87.int Alist<T>::Find(string name) {
88. for (int i = 0; i < Size; i++) {
89.  if (ele[i].name == name) return i;
90. }
91. return -1;
```

```
}

template<class T>
int Alist<T>::cal(int tar) {
 int res = -1, cnt = 0;  // res = 0
 for (int i = 0; i < Size; i++) {
  if(ele[i].clas == tar){
   if(res == -1) res = ele[i].dor;
       else res = res ^ ele[i].dor;
       cnt ++;
     }
    }
   // if(res == -1) res = 0;
   // if(cnt == 1) res = 0;
   // return res;
    return res == -1 ? 0 : res;

   }

   template<class T>
   void Alist<T>::Del(string Dtar) {
    int ind = Find(Dtar);
    if(ind == -1) return;
    copy(ele + ind + 1, ele + Size, ele + ind);
   // ele[--Size].~T(); // 调用析构函数
    --Size;
   }

   Alist<stu> abk;  //

   void solve() {
    int N; cin >> N; // 操作条数
    string n, p; int c, d;
    for (int i = 1, op, ind; i <= N; i++) {
     cin >> op;
     if (op == 0) {
      cin >> n >> p >> c >> d;
   //   abk.Push(stu(n, p, c, d));
      abk.Push({n, p, c, d});  // 四元组
     }
     else if (op == 1) {
      cin >> n;
      abk.Del(n);
     }
     else if (op == 2) {
      int op1;
```

```
      cin >> n >> op1;
      ind = abk.Find(n);
      if(op1 == 1) {
       cin >> p;
       if(ind == -1) continue;
       abk.setP(ind, p);
      }
      else if (op1 == 2) {
       cin >> c;
       if(ind == -1) continue;
       abk.setC(ind, c);
      }
      else if (op1 == 3) {
       cin >> d;
       if(ind == -1) continue;
       abk.setD(ind, d);
      }
     }
     else if (op == 3) {
      cin >> n;
      ind = abk.Find(n);
      if(ind == -1) cout << 0 << '\n';
      else cout << 1 << '\n';
     }
     else if (op == 4) {
      cin >> c;
      cout << abk.cal(c) << '\n';
     }
    }
   // abk.chk();
   }

   int main(){
   // freopen("out.txt", "w", stdout);

    solve();
    return 0;
   }
```