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：201700130033	姓名： 武学伟	班级： 2017 级 2 班
实验题目：数组描述线性表		
实验学时：4	实验日期： 2018.10.15	

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

软件环境：

Win10home，codeblocks

1. 实验内容（题目内容，输入要求，输出要求）

1、创建线性表类：线性表的存储结构使用数组描述，提供操作：插入、删除、查找等。

2、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。由键盘输入或文件录入的通讯录信息建立通讯录表，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所有人信息。

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）

数据结构：数组描述的线性表

思路描述：

1) 建立一个学生类，包含姓名、电话号码、班级、宿舍。
2) 建立数组描述的线性表，完成以上功能。在数据成员中加入，bool 变量 trueindex 来判断是否进行了有效执行。在成员函数中加入 Edit 函数以及 Output 函数实现编辑于输出功能。
3) 所有的类均采用模板，在定义线性表对象时，数据类型为学生类
4) 因为数据类型为学生类，在线性表的类中为了方便进行比较，重载学生类的输出以及==操作符的重载
5) 因为类的封装的特性，在遍历数组时，不能直接访问对象的数据成员，所以定义一个返回数组地址的成员函数，用以访问数组的地址，方便数组的遍历。

3. 测试结果（测试输入，测试输出，结果分析）

测试输入：插入初始的三组数据

```
All information of students:
Name: wxw
PhoneNumber: 1780000
Class: 2
Dormitary:131

Name: hh1
PhoneNumber: 1780001
Class: 2
Dormitary:131

Name: 1c
PhoneNumber: 1780002
Class: 3
Dormitary:113
```

wxw 1780000 2 131
hhl 1780001 2 131
lc 1780002 3 113

测试功能：

1) 在存在的索引位置和不存在的索引位置插入

Index：3 lxh 1780003 4 124

```
Input theIndex
3
Name:lxh
PhoneNumber:1780003
Class:4
Dormitary:124

请按任意键继续. . .
```

```
All information of students:
Name: wxw
PhoneNumber: 1780000
Class: 2
Dormitary:131

Name: hhl
PhoneNumber: 1780001
Class: 2
Dormitary:131

Name: lc
PhoneNumber: 1780002
Class: 3
Dormitary:113

Name: lxh
PhoneNumber: 1780003
Class: 4
Dormitary:124
```

Index：100 fsy 1780011 2 121

```
Input theIndex
100
Name:fsy
PhoneNumber:1780011
Class:2
Dormitary:121

wrong index
index = 100 size = 4
请按任意键继续. . .
```

2) 在存在的索引位置和不存在的索引位置删除

Index：2 lc 1780002 3 113

```
Please choose
2
Input theIndex
2
```

```
All information of students:
Name: wxw
PhoneNumber: 1780000
Class: 2
Dormitary:131

Name: hhl
PhoneNumber: 1780001
Class: 2
Dormitary:131

Name: lxh
PhoneNumber: 1780003
Class: 4
Dormitary:121
```

Index：100

```
Please choose
2
Input theIndex
100
wrong index
index = 100 size = 3
```

3) 按照存在的和不存在的姓名查找

wxw：

```
Please choose
4
Input name:
wxw

Name: wxw
PhoneNumber: 1780000
Class: 2
Dormitary:131
```

lc：

```
Please choose
4
Input name:
lc

The name of this student has not been found!
```

4) 按照存在的和不存在的班级查找

2 班：

```
Input class:
2

Name: wxw
PhoneNumber: 1780000
Class: 2
Dormitary:131

Name: hhl
PhoneNumber: 1780001
Class: 2
Dormitary:131
```

3 班：

```
Please choose
5
Input class:
3

The name of this class has not been found!
```

5) 在存在的索引位置和不存在的索引位置编辑：

Index：0 name：zjm

```
Please choose
3
Input theIndex
0
Option(s):
1.Name 2.PhoneNumber 3.Class 4.Dormitary
1
New Name: zjm
```

```
All information of students:
Name: zjm
PhoneNumber: 1780000
Class: 2
Dormitary:131

Name: hhl
PhoneNumber: 1780001
Class: 2
Dormitary:131

Name: lxh
PhoneNumber: 1780003
Class: 4
Dormitary:121
```

Index：100 name：yuandiaodiaodiao

```
Please choose
3
Input theIndex
100
wrong index
index = 100 size = 3
请按任意键继续. . .
```

结果分析：所有功能均实现，且异常信息也完成了处理

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

结果符合预期

在程序设计的时候，将选项放入 while（1）循环中，以达到重复操作的目的，但是这样遇到了异常信息就会直接退出程序，所以在异常部分做了处理，定义了 bool 变量 trueindex，初始化为 true，当遇到异常后，输出异常信息，更新变量 trueindex 为 false，在 trueindex 为 true 时执行功能，并在函数结束前再次将变量 trueindex 更新为 true，这样就能实现异常的有效处理。

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>
#include <stdlib.h>
```

```
using namespace std;

/*学生类*/
class Student
{
    private :
        string Name;  //姓名
        string Number;  //电话号码
        string Class;  //班级
        string Dormitary;  //宿舍

    public :
        Student(){};
        Student(string,string,string,string);
        ~Student(){};
        friend ostream& operator << (ostream&out, const Student&st);  //输出学生信息，<<的重载
        void Edit();  //编辑学生信息
        void Insert(); //插入学生信息
        string& returnclass(){return Class;}  //返回学生所在的班级
       string& returnname(){return Name;}  // 返回学生的姓名
       friend bool operator == (const Student&a,const Student&b);   //比较学生信息，==的重载
};

/*改变数组的长度*/
template <class T>
void changeLength1D(T*& a, int oldLength, int newLength)
{
    if (newLength < 0)    //新数组长度小于0，抛出异常
    {
        throw "New length must be >= 0";
        cout << "New length must be >= 0" << endl;
    }
    T* temp = new T[newLength];  //新数组
    int number;                  //需要复制的元素数
    if (oldLength<newLength)
        number = oldLength;
    else
        number = newLength;
    for (int i=0; i<number; i++)
        temp[i] = a[i];
    delete []a;    //删除旧数组
    a = temp;     //复制新的数组
    delete []temp;  //删除 temp 数组
}
```

```
/*数组描述线性表*/
template <class T>
class array_list
{
    protected:
        T* element;
        int arrayLength;
        int listsize;
        bool trueindex;
    public:
        array_list(int initialCapacity = 10);  //构造函数
        array_list(const array_list<T>&);   //复制构造函数
        ~array_list() {delete [] element;}   //析构函数
        bool getbool(){return trueindex;}
        bool Empty() const {return listsize == 0;}  //判断数组是否为空
        int Size() const {return listsize;}   //返回数组的长度
        void Erase(int theIndex);  //删除索引为 theIndex 的元素
        void Insert(int theIndex, const T& theElement);  //在索引为 theIndex 的位置
插入元素，索引后元素右移
        void Edit(int theIndex);  //编辑索引为 theIndex 的元素
        void Output() const;//输出该线性表的所有元素信息
        T *head_element() {return element;}  //返回数组的首地址，用于姓名的的查找
        int capacity() const {return arrayLength;}  //返回数组容量大小
};

/*学生类构造函数*/
Student::Student(string nam,string num,string cla,string dor)
{
    Name = nam;
    Number = num;
    Class = cla;
    Dormitary = dor;
}

/*输出学生信息，<<的重载*/
ostream& operator << (ostream&out,const Student&st)
{
    out<<"Name: "<<st.Name<<endl;
    out<<"PhoneNumber: "<<st.Number<<endl;
    out<<"Class: "<<st.Class<<endl;
    out<<"Dormitary:"<<st.Dormitary<<endl;
    return out;
}

/*比较学生信息，==的重载*/
```

```
bool operator == (const Student&a,const Student&b)
{
    if    (a.Name==b.Name    &&    a.Number==b.Number    &&    a.Class==b.Class    &&
a.Dormitary==b.Dormitary)
        return true;
    else
        return false;
}

/*插入学生信息*/
void Student::Insert()
{
    cout << "Name:";
    cin >> Name;
    cout << "PhoneNumber:";
    cin >> Number;
    cout << "Class:";
    cin >> Class;
    cout << "Dormitary:";
    cin >> Dormitary;
    cout << endl;
}

/*编辑学生信息*/
void Student::Edit()
{
    int op;
    cout << "Option(s):" << endl; //输出选项
    cout << "1.Name" << " " << "2.PhoneNumber" << " " << "3.Class" << " 4.Dormitary"
<< endl;
    cin >> op;  //用数字进行修改内容的选择
    switch(op)
    {
        case 1:                          //修改姓名
            cout << "New Name: ";
            cin >> Name;
            break;
        case 2:                          //修改电话
            cout << "New PhoneNumber: ";
            cin >> Number;
            break;
        case 3:                          //修改班级
            cout << "New Class: ";
            cin >> Class;
            break;
        case 4:                          //修改宿舍
```

```
            cout << "New Dormitary: ";
            cin >> Dormitary;
            break;
        default:break;
    }
    cout << endl;
}

/*线性表构造函数*/
template<class T>
array_list<T>::array_list(int initialCapacity)
{
    if (initialCapacity < 1)
    {
        throw "initialCapacity must be > 0";
        cout << "initialCapacity must be > 0" << endl;
    }
    arrayLength = initialCapacity;
    element = new T[arrayLength];
    listsize = 0;
    trueindex = true;
}

/*线性表复制构造函数*/
template <class T>
array_list<T>::array_list(const array_list<T>& theList)
{
    arrayLength = theList.arrayLength;
    listsize = theList.listsize;
    trueindex = true;
    element = new T[arrayLength];
    for (int i=0; i<theList.listsize; i++)
        element[i] = theList.element[i];
}

/*删除索引为 theIndex 的元素*/
template <class T>
void array_list<T>::Erase (int theIndex)
{
    trueindex = true;
    if (theIndex<0 || theIndex>=listsize)
    {
        cout << "wrong index" << endl;
        cout << "index = " << theIndex << " size = " << listsize << endl;
        trueindex = false;
    }          //检查元素
```

```
    if (trueindex)
    {
        for (int i=theIndex+1; i<listsize; i++)
            element[i-1] = element[i];
        element[--listsize].~T();  //调用析构函数
    }
}

/*在索引为 theIndex 的位置插入元素，索引后元素右移*/
template <class T>
void array_list<T>::Insert(int theIndex, const T& theElement)
{
    trueindex = true;
    if (theIndex<0 || theIndex>listsize)
    {
        cout << "wrong index" << endl;
        cout << "index = " << theIndex << " size = " << listsize << endl;
        trueindex = false;
    }   //检查
    if(trueindex)
    {
        if (listsize == arrayLength)  //数组已满，数组倍增
        {
            changeLength1D(element, arrayLength, 2*arrayLength);
            arrayLength *=2;
        }
        for (int i=listsize-1; i>=theIndex; i--)
            element[i+1] = element[i];
        element[theIndex] = theElement;
        listsize++;
    }

}

/*编辑索引为 theIndex 的元素*/
template <class T>
void array_list<T>::Edit (int theIndex)
{
    trueindex = true;
    if (theIndex<0 || theIndex>=listsize)
    {
        cout << "wrong index" << endl;
        cout << "index = " << theIndex << " size = " << listsize << endl;
        trueindex = false;
    }    //检查元素
    if (trueindex)
```

```
        element[theIndex].Edit();
}

/*输出数组中所有元素信息*/
template<class T>
void array_list<T>::Output() const
{
   for(int i=0; i<listsize; i++)
   cout << element[i] << endl;
}

/*菜单*/
void Menu()
{
    cout << "Student address book management system" << endl;
    cout << "Insert(1)" << endl;      //插入
    cout << "Delete(2)" << endl;     //删除
    cout << "Edit(3)" << endl;       //编辑
    cout << "Find(4)" << endl;       //按姓名查询
    cout << "ClassFind(5)" << endl; //按班级查询
    cout << "Output(6)" << endl;     //输出信息
    cout << "Exit(0)" << endl;
    cout << "Please choose" << endl;
}

int main()
{
    array_list<Student> sarray;    //创建一个学生类型的线性表
    while (1)
    {
        system("cls");
        Menu();
        int c;
        cin >> c;   //输入数字以供选择
        int theIndex;
        if (c==1)    //插入
        {
            cout << "Input theIndex" << endl;
            cin >> theIndex;
            Student stu;
            stu.Insert();
            sarray.Insert(theIndex,stu);
            system("pause");
            continue;
        }
        if (c==2)    //删除
```

```
        {
            cout << "Input theIndex" << endl;
            cin >> theIndex;
            sarray.Erase(theIndex);
            system("pause");
            continue;
        }
        if (c==3)    //编辑
        {
            cout << "Input theIndex" << endl;
            cin >> theIndex;
            sarray.Edit(theIndex);
            system("pause");
            continue;
        }
        if (c==4)    //按姓名查询
        {
            cout << "Input name:" << endl;
            string a;
            cin >> a;
            cout << endl;
            bool f = false;
            for                  (Student                  *p=sarray.head_element();
p!=(sarray.head_element()+sarray.Size()); p++)//遍历线性表的所有元素，查找姓名
            {
                if (a == p->returnname())
                {
                    f = true;
                    cout << *p;
                }
            }
            if(!f)
                cout << "The name of this student has not been found!" << endl;
            system("pause");
            continue;
        }

        if (c==5)    //按班级查询
        {
            cout << "Input class:" << endl;
            string a;
            cin >> a;
            cout << endl;
            bool f = false;
            for                  (Student                  *p=sarray.head_element();
p!=sarray.head_element()+sarray.Size(); p++)//遍历线性表的所有元素，查找班级
```

```
            {
                if (a==p->returnclass())
                {
                    f = true;
                    cout << *p << endl;
                }
            }
            if(!f)
                cout << "The name of this class has not been found!" << endl;
            system("pause");
            continue;
        }
        if (c==6)    //输出信息
        {
            cout << "All information of students:" << endl;
            sarray.Output();
            system("pause");
            continue;
        }
        if (c==0)    //退出
            return 0;
    }
}
```