重 庆 交 通 大 学

《算法与数据结构》课程

实 验 报 告

班　　级：计算机专业 19 级　曙光班

实验项目名称：顺序表实验

实验项目性质：验证性实验

实验所属课程：数据结构与算法

实验室(中心)：B01 409

指 导 教 师 ：鲁云平

实验完成时间：2020 年 10 月 11 日

教师评阅意见： 签名：　　年　月　日
实验成绩：

一、实验目的

理解并熟悉顺序表的操作。

二、实验内容及要求

要求：

实现的顺序表可适用于多种数据类型（简单数据类型和自定义数据类型，如学生类等），并进行测试

三、系统分析

（1）数据方面：

需要存储数据，并且数据个数不定，需要存储能存储的最大数量，现在已经存储的数量。

（2）功能方面：

1、必须要能够扩展数据储存的内存，数据可能会很多，必须要能够动态分配；

2、需要又一个能知道顺序表长度的操作。

3、必须要有一个一个构造和析构操作。

4、必须要可以对数据元素的增加删除操作，那么也必须要有判断顺序表是否已经满了和是否为空的操作。

5、需要一个能够获取顺序表中指定元素值的操作，类是具有封装性的，外部函数是不能访问的。

6、需要一个能遍历整个顺序表的操作。

7、由于数据量比较大，需要文件的读取和存储。

四、系统设计

（1）设计的主要思路

说明整体设计思路

根据题目要求：实现的顺序表可适用于多种数据类型（简单数据类型和自定义数据类型，如学生类等），必须要采用类模板才能实现，先设计一个顺序表模板类。

然后再自定义一个学生类类型进行测试。

（2）数据结构的设计

数据结构设计思路

顺序表类：

```
template <typename T>
class SeqList
{
private:
    T *data;
    int maxsize;
    int num;
    void resetsize(int newsize =2*defaultsize);//每次扩展为原来的两倍
public:
    SeqList(int size= defaultsize);//构造函数
    ~SeqList();//析构函数
    int Length() const;//计算长度
    bool IsFull() const;//判满
    bool IsEmpty() const;//判空
    int Search(T& x) const;//搜索 x
    T& Get(int i);//得到 i 的对象
    bool Push_back(const T& x);//在尾部插入
    bool Insert(int i, const T& x);//在指定位置插入
    bool Remove(T& x);//删除指定元素
    bool RemovePos(int pos);//删除指定位置元素
    void Display(int begin, int end);//显示整个表
    bool SaveFile(const char FileName[]);//保存文件
    bool ReadFile(const char FileName[]);//读取文件
    int Getnum();//获取表中的元素个数
};
```

1、定义好存储数据元素的参数 data，利用指针类型，可以灵活分配空间。

2、顺序表具有增加，删除，查找，遍历等基本功能。

3、由于数据类型可以是解百纳数据类型，也可以是自定义数据类型，所以用模板类实现。

自定义一个学生类测试程序。

学生类：

```
class Student
{
```

```
private:
    string id;
    string name;
public:
    Student()//构造函数
    {

    }
    Student (string i, string n)//含参构造函数
    {
        id=i;name=n;
    }
//    Student( const Student& s);
    ~Student()
    {

    }
    Student operator=(Student s);//重载赋值运算
    bool operator==(Student s);//重载==
    friend ostream& operator<<(ostream &cout, Student s);//重载<<
    friend istream& operator>>(istream &cin, Student &s);//重载>>
    void setIdName(string i,string n);//设置名字和学号
};
```

1、定义名字和学号两个基本数据。

2、重载一些运算符，=、==、<<、>>、这几个运算符，程序中必须要用到的运算符。

3、将<<和>>定义为友元函数，使其获得能够范围跟类中 private 的权限，以改变数据。

（3）基本操作的设计

基本操作的抽象描述，关键算法的设计思路和算法流程图。

基本操作的抽象描述一般为操作名，初始条件，操作结构，参数说明等。

SeqList 类：

```
    void resetsize(int newsize =2*defaultsize);//每次扩展为原来的两倍
/*首先判断传入的 newsize 是否合法
    * 如果说 newsize 大于原来的 maxsize，则重新分配储存空间
    * 如果重新分配的空间等于 NULL 说明分配空间失败
    * 然后再将原来的数据元素赋值给新的顺序表
    * 最后还要改变 maxsize 的值

    int Length() const;//计算长度
 //直接返回现在顺序表中的元素个数久是表的长度
```

```
    bool IsFull() const;//判满
//如果现在顺序表中的元素个数和最大储存元素个数相同表明表已经满了

    bool IsEmpty() const;//判空
//如果顺序表的元素个数为 0，表明表为空

    int Search(T& x) const;//搜索 x
/*如果 data 数组里面有与 x 相同的对象则 flag=1 并且返回是第几个元素
     * 如果没有返回-1

    T& Get(int i);//得到 i 的对象
//判断一个输入的位置是否合法，合法则返回该位置的对象

    bool Push_back(const T& x);//在尾部插入
//在尾部插入对象，首先判断顺序表是否已经满了，如果已经满了，先对顺序表进行扩展，然后再插入对象，数量 num 值增加 1;
    //如果没有满，直接插入末尾，num 值增加 1

    bool Insert(int i, const T& x);//在指定位置插入
//首先判断传入的 i 是否合法
    //再判断顺序表是否已经满了
    //如果满了扩展顺序表,然后将第 i 个位子以后的元素一次向后移动一个位置，然后将 x 插入，num 值增加 1
    //如果没有满，直接将第 i 个位子以后的元素一次向后移动一个位置，然后将 x 插入，num 值增加 1

    bool Remove(T& x);//删除指定元素
//首先判断顺序表是否为空，空表不能删除
    //如果部位空，只需要找到改元素的位置，并且将改元素后面的元素向前移动一个位置，num 值减 1

    bool RemovePos(int pos);//删除指定位置元素
//首先判断传入的位置信息是否合法
    //如果合法，检验表是否为空
    //如果为空不能进行删除操作
    //如果不为空将该位置后面的元素向前移动一个位置，num 值减 1

    void Display(int begin, int end);//显示整个表
//遍历顺序表

    bool SaveFile(const char FileName[]);//保存文件
//首先定义存文件的对象
     //打开文件，如果打开文件失败，提示打开失败
```

```
    bool ReadFile(const char FileName[]);//读取文件
//首先定义读文件的对象
    //打开文件，如果打开文件失败，提示打开失败
```

五、编程环境与实验步骤

（1）编程环境

主要是操作系统、编程工具软件

操作系统：Windows 10

编译工具软件： Qt Creator

（2）实验步骤

只说明程序相关的各种文件创建步骤及文件的作用，不需说明文件的具体内容。

seqlist.h 定义顺序表类以及函数声明

seqlist.cpp 具体操作

student.h 定义学生类以及相关操作声明

student.cpp 具体操作

main.cpp 测试程序

SeqListdata.txt 存放 int 型的数据

SeqListStudent.txt 存放 Student 型的数据

（3）编译参数

若有特殊的编译参数设置，需说明详细步骤。

若无特殊的编译参数设置，则只需简单说明操作步骤。

六、实现代码

主要功能的实现代码

```
#include<iostream>
#include<cstdlib>
#include<fstream>
#include "seqlist.h"
using namespace std;
template <typename T>
void SeqList<T> ::resetsize(int newsize)//扩展数据元素的长度
{
    /*首先判断传入的 newsize 是否合法
```

```
     * 如果说 newsize 大于原来的 maxsize，则重新分配储存空间
     * 如果重新分配的空间等于 NULL 说明分配空间失败
     * 然后再将原来的数据元素赋值给新的顺序表
     * 最后还要改变 maxsize 的值
    */
    if(newsize <=0)
    {
        cerr<<"数组大小无效！！！"<<endl;
        return;
    }
    if(newsize > maxsize)
    {
        T* newarray = new T[newsize];
        if(newarray==NULL)
        {
            cerr<<"内存分配失败！！！"<<endl;
            exit(1);
        }
        int n=num;
        T* str1=data;
        T* str2=newarray;
        while(n--)
            *str2++=*str1++;
        delete [] data;
        data=newarray;
        maxsize=newsize;
    }
}
template <typename T>
SeqList <T> ::SeqList(int sz)
{
    if(sz>0)
    {
        maxsize = sz;num=0;
        data = new T[maxsize];
        if(data==NULL)
        {
            cerr<<"内存分配失败！！！"<<endl;
            exit(1);
        }
    }
}
template <typename T>
SeqList <T> ::~SeqList()
```

```
{
    delete []data;
}
template <typename T>
int SeqList <T> ::Length()const
{
    //直接返回现在顺序表中的元素个数久是表的长度
    return num;
}
template <typename T>
bool SeqList <T> ::IsFull() const
{
    //如果现在顺序表中的元素个数和最大储存元素个数相同表明表已经满了
    return (num==maxsize)?true:false;
}
template <typename T>
bool SeqList <T> ::IsEmpty() const
{
    //如果顺序表的元素个数为 0，表明表为空
    return (num==0)?true:false;
}
template <typename T>
int SeqList<T> ::Search(T& x) const
{
    /*如果 data 数组里面有与 x 相同的对象则 flag=1 并且返回是第几个元素
     * 如果没有返回-1
     */
    int flag=0;
    for(int i = 0;i < num;i++)
    {
        if(data[i]==x)
        {
            flag=1;
            return i+1;
        }
    }
      if(!flag){
            cerr<<"没有该元素"<<endl;
            return -1;
        }
}
template <typename T>
T& SeqList <T>::Get(int i)
{
```

```
    //判断一个输入的位置是否合法，合法则返回该位置的对象
    if(i<num && i>0)
        return data[i-1];
    else
    {
        cerr<<"该位置没有元素！！！"<<endl;
    }
}
template <typename T>
bool SeqList<T>::Push_back(const T &x)
{
    //在尾部插入对象，首先判断顺序表是否已经满了，如果已经满了，先对顺序表进行扩展，然后再插入对象，数量 num 值增加 1；
    //如果没有满，直接插入末尾，num 值增加 1
    if(IsFull())
    {
        resetsize(2*maxsize);
        data[num]=x;
        num++;
        return true;
    }
    else
    {
        data[num]=x;
        num++;
        return true;
    }

}
template <typename T>
bool SeqList<T> ::Insert(int i,const T& x)
{
    //首先判断传入的 i 是否合法
    //再判断顺序表是否已经满了
    //如果满了扩展顺序表，然后将第 i 个位子以后的元素一次向后移动一个位置，然后将 x 插入，num 值增加 1
    //如果没有满，直接将第 i 个位子以后的元素一次向后移动一个位置，然后将 x 插入，num 值增加 1
    if(i<0||i>num)
    {
        cerr<<"输入位置错误！！！"<<endl;
        return false;
    }
    if(IsFull())
```

```
    {
        resetsize(2*maxsize);
        for(int j=num-1;j>=i-1;j--)
            data[j+1]=data[j];
        data[i-1]=x;
        num++;
        return true;
    }
    else
    {
        for(int j=num-1;j>=i-1;j--)
            data[j+1]=data[j];
        data[i-1]=x;
        num++;
        return true;
    }
}
template <typename T>
bool SeqList<T>::Remove(T& x)
{
    //首先判断顺序表是否为空，空表不能删除
    //如果部位空，只需要找到改元素的位置，并且将改元素后面的元素向前移
动一个位置，num 值减 1
    if(IsEmpty())
    {
        cerr<<"顺序表为空，不能进行删除！！！"<<endl;
        return false;
    }
    int i=Search(x);
    if(i==-1)
        return false;
    for(int j=i-1;j<num;j++)
    {
        data[j]=data[j+1];
    }
    num--;
    return true;
}
template <typename T>
bool SeqList<T>::RemovePos(int pos)
{
    //首先判断传入的位置信息是否合法
    //如果合法，检验表是否为空
    //如果为空不能进行删除操作
```

```
    //如果不为空将该位置后面的元素向前移动一个位置，num 值减 1
    if(pos<1||pos>num)
    {
        cerr<<"输入位置错误！！！"<<endl;
        return false;
    }
    if(IsEmpty())
    {
        cerr<<"顺序表为空，不能进行删除！！！"<<endl;
        return false;
    }
    for(int j=pos;j<num;j++)
    {
        data[j-1]=data[j];
    }
    num--;
    return true;
}
template <typename T>
void SeqList<T>::Display(int start,int end)
{
    //遍历顺序表
    for(int i=start;i<=end;i++)
    {
        cout<<data[i]<<endl;
    }
    cout<<"dispaly over"<<endl;
    return;
}
template <typename T>
bool SeqList<T>::ReadFile(const char FileName[])
{
    //首先定义读文件的对象
    //打开文件，如果打开文件失败，提示打开失败
    ifstream fin;
    fin.open(FileName);
    if(fin.fail())
    {
        cerr<<"文件打开失败！！！"<<endl;
        return false;
    }
    fin>>num;
    for(int i=0;i<num;i++)
        fin>>data[i];
```

```
    fin.close();
    cout<<"readover"<<endl;
    return true;
}
template <typename T>
bool SeqList<T>::SaveFile(const char FileName[])
{
    //首先定义存文件的对象
     //打开文件，如果打开文件失败，提示打开失败
    ofstream fout;
    fout.open(FileName);
    if(fout.fail())
    {
        cerr<<"文件打开失败！！！"<<endl;
        return false;
    }
    fout<<num<<endl;
    for(int i=0;i<num;i++)
        fout<<data[i]<<endl;
    fout.close();
    return true;
}
template <typename T>
int SeqList<T>::Getnum()
{
    return num;
}
```

七、测试结果与说明

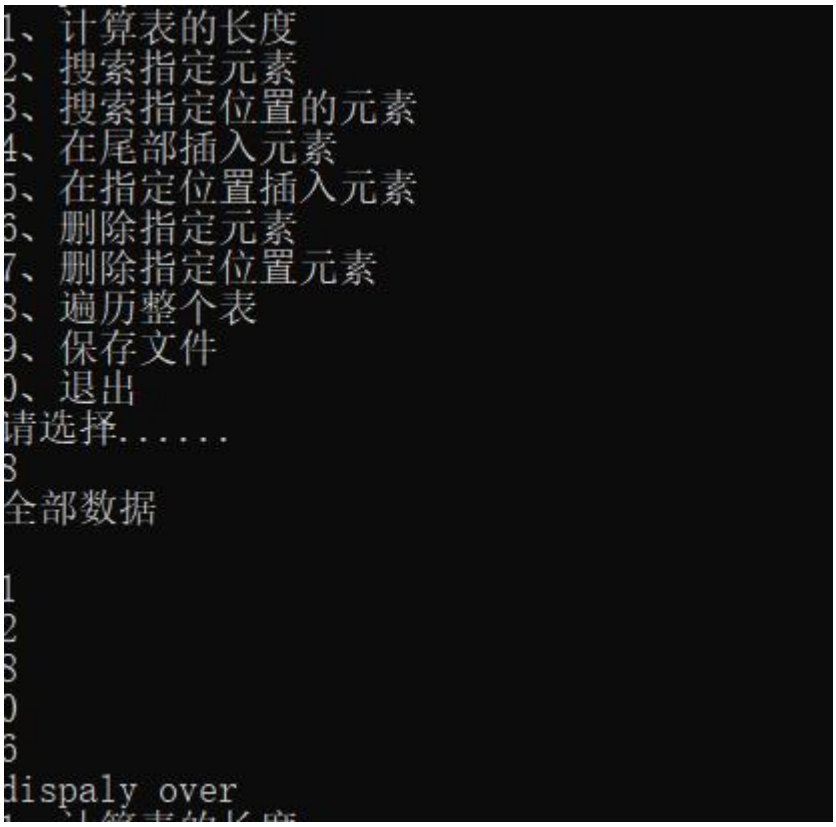
至少完成功能测试，使用测试数据测试相关功能是否符合设计要求。

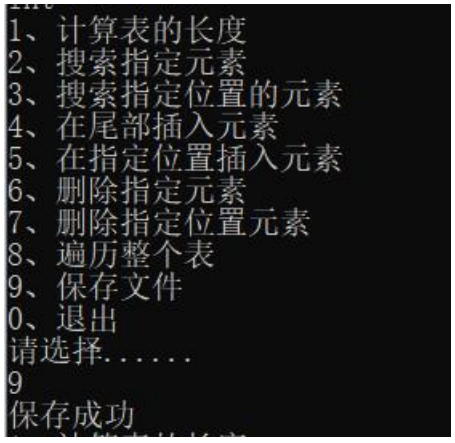
Int　型测试结果

```
输入数据类型int or Student
int
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
1
表的长度为: 5
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
2
输入一个元素
3
3元素是第3个元素
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
3
输入一个位置
4
第4个元素是4
```

```
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
4
输入要插入的整数元素
6

1
2
3
4
0
6
dispaly over
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
5
输入要插入的位置
3
输入要插入的元素
8

1
2
8
3
4
0
6
dispaly over
```

```
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
6
输入要删除的元素
3

1
2
8
4
0
6
dispaly over
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
7
输入要删除的元素位置
4

1
2
8
0
6
dispaly over
```



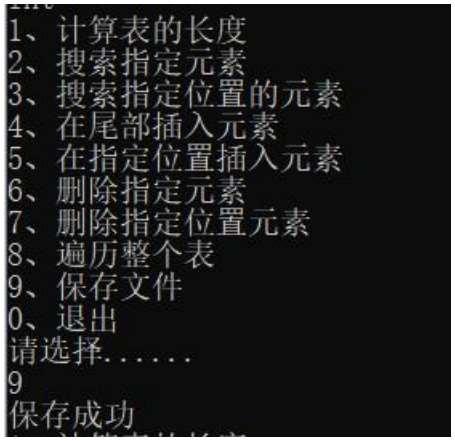


Student 型测试结果

```
输入数据类型int or Student
Student
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
1
表的长度为：5
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
2
输入一个学生的学号
631907060434
输入一个学生的名字
zhouyingchuan
   631907060434        zhouyingchuan学生是第1个学生
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
3
输入一个位置
2
第2个学生是    631907060501                chenhan
```

```
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
4
输入要插入的学生学号
631907060416
输入要插入的学生姓名
fangmin

   631907060434        zhouyingchuan
   631907060501              chenhan
   631707060520           quhongyang
   631907060305         dongjiachneg
   631907060306             dujunjie
   631907060416              fangmin
dispaly over
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
5
输入要插入的位置
3
输入要插入的学生学号
631907060520
输入要插入的学生姓名
zhangsan

   631907060434        zhouyingchuan
   631907060501              chenhan
   631907060520             zhangsan
   631707060520           quhongyang
   631907060305         dongjiachneg
   631907060306             dujunjie
   631907060416              fangmin
dispaly over
```

```
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
6
输入要删除的学生学号
631907060306
输入要删除的学生姓名
dujunjie

   631907060434        zhouyingchuan
   631907060501              chenhan
   631907060520             zhangsan
   631707060520           quhongyang
   631907060305         dongjiachneg
   631907060416              fangmin
dispaly over
1、计算表的长度
2、搜索指定元素
3、搜索指定位置的元素
4、在尾部插入元素
5、在指定位置插入元素
6、删除指定元素
7、删除指定位置元素
8、遍历整个表
9、保存文件
0、退出
请选择......
7
输入要删除的学生位置
2

   631907060434        zhouyingchuan
   631907060520             zhangsan
   631707060520           quhongyang
   631907060305         dongjiachneg
   631907060416              fangmin
dispaly over
```



八、实验分析

（1）算法的性能分析

主要针对增加、删除、搜索等算法。

在尾部增加算法的时间复杂度为 O(1)

在指定位置插入元素的时间复杂度为 O((n）

删除指定元素的时间复杂度为 O(n）

删除指定位置的元素时间复杂度为 O(n)

搜索算法的时间复杂度为 O((n+1)/2)

（2）数据结构的分析

通过性能分析总结此种存储结构的优缺点，并说明其适用场景。

有点：可以随机访问元素，方便去除元素的数据

缺点：删除的时候只是将元素覆盖掉，并没有将该元素的内存释放，有点费空间。

使用场景：适用于需要常用数据，但是整体数据改变不大，且数据总量不大的场景。

九、实验总结

主要针对本实验的分析、设计、实现、测试等环节进行总结，包含收获与不足，此部分的阐述应较为详细。

本实验需要用多种数据类型测试，所以需要用模板。设计的时候，先设计了一个顺序表类，然后再设计了一个学生类进行测试。实现的时候，具体的操作据说再增加删除，搜索需要的时候再去写操作。测试的时候用文件测试，用了多组数据测试，用学生类文件和整型数据测试的。

此次实验让我有进一步熟悉了一下 C++语法，之前很多语法已经忘记了，现在又想了起来。由于对 C++的部分东西已经遗忘，写程序的时候看来很多以前的课件。文件操作不能读中文，我感觉是因为文件格式是 utf-8，而我写代码通常编码是 gb2312，所以读文件会乱码。以后我可能会更多的选择 utf-8 了，这个编码使用的范围更广。

附录

参考文献：

1.
2.
3.