数据结构_基于顺序表的通讯录

顺序表代码

SeqList.h

```h
#pragma once
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
#include"Contact.h"

typedef PF SeqList_Datatype;

typedef struct SeqList
{
    SeqList_Datatype* Data;
    int size;
    int capacity;
}SL;

//顺序表初始化
void SLInit(SL* ps);
//顺序表的销毁
void SLDestroy(SL* ps);
void SLPrint(SL ps);

//头部插入删除  尾部插入删除
void SLPushBack(SL* ps, SeqList_Datatype x);
void SLPushFront(SL* ps, SeqList_Datatype x);
void SLPopBack(SL* ps);
void SLPopFront(SL* ps);

//指定位置之前插入/删除数据
void SLInsert(SL* ps, int pos, SeqList_Datatype x);
void SLErase(SL* ps, int pos);

//查找修改
int SLFind(SL* ps, SeqList_Datatype x);
void SeqListModity(SL* ps, int pos, SeqList_Datatype x);
```

SeqList.c

```c
#define _CRT_SECURE_NO_WARNINGS 1
#include"SeqList.h"

void SLInit(SL* ps)//初始化
{
    ps->Data = NULL;
    ps->capacity = ps->size = 0;
}

void SLDestroy(SL* ps)//销毁
{
    if (ps->Data)
        free(ps->Data);
    ps->Data = NULL;
    ps->size = ps->capacity = 0;
}

void SLPrint(SL ps)//打印
{
    for (int i = 0; i < ps.size; i++)
        printf("%d ",ps.Data[i]);
    printf("\n");
}

void Check_Capacity(SL* ps)//扩容
{
    if (ps->capacity == ps->size)
    {
        int newcapacity = ps->capacity == 0 ? 4 : ps->capacity * 2;
        SeqList_Datatype* tmp = (SeqList_Datatype*)realloc(ps->Data,
sizeof(SeqList_Datatype) * newcapacity);
        if (tmp == NULL)
        {
            perror("realloc is fail!");
            exit(1);
        }
        ps->Data = tmp;
        ps->capacity = newcapacity;
    }
}

void SLPushBack(SL* ps, SeqList_Datatype x)//尾插
{
    assert(ps);
    Check_Capacity(ps);
    ps->Data[ps->size++] = x;
}

void SLPushFront(SL* ps, SeqList_Datatype x)//头插
{
    assert(ps);
    Check_Capacity(ps);
    for (int i = ps->size; i > 0; i--)
```

```
    {
        ps->Data[i] = ps->Data[i - 1];
    }
    ps->Data[0] = x;
    ps->size++;
}

void SLPopBack(SL* ps)//尾删
{
    assert(ps);
    assert(ps->size);
    ps->size--;
}

void SLPopFront(SL* ps)//头删
{
    assert(ps);
    assert(ps->size);
    for (int i = 0; i < ps->size - 1; i++)
    {
        ps->Data[i] = ps->Data[i + 1];
    }
    ps->size--;
}

void SLInsert(SL* ps, int pos, SeqList_Datatype x)//指定插入  pos是下标，不是第几
个!
{
    assert(ps);
    Check_Capacity(ps);
    for (int i = ps->size; i > pos + 1; i--)
    {
        ps->Data[i] = ps->Data[i - 1];
    }
    ps->Data[pos] = x;
    ps->size++;
}

void SLErase(SL* ps, int pos)//指定删除
{
    assert(ps);
    for (int i = pos; i < ps->size; i++)
    {
        ps->Data[i] = ps->Data[i + 1];
    }
    ps->size--;
}

//int SLFind(SL* ps, SeqList_Datatype x)//查找
//{
//  assert(ps);
//  for (int i = 0; i < ps->size; i++)
//  {
//      if (x == ps->Data[i])
```

```
//          return i;
//  }
//  return -1;
//}

void SeqListModity(SL* ps, int pos, SeqList_Datatype x)//修改
{
    assert(ps);
    assert(pos < ps->size);
    ps->Data[pos] = x;
}
```

通讯录代码（以顺序表为底层逻辑）

在顺序表的基础上，新定义一个（personinfo）结构体，将这个结构体作为顺序表的数据类型，存储在顺序表中。通过顺序表的基本接口实现personinfo的增删查改。

顺序表部分让我觉得很痛苦的是，在对尾部进行操作的时候需要用while进行找尾，这里我经常搞错循环进行的结束条件。需要注意。

Contact.h

```
#pragma once
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>

#define NAME_MAX 20
#define TEL_MAX 11

typedef struct personinfo
{
    char name[NAME_MAX];
    char tel[TEL_MAX];
}PF;

typedef struct SeqList Contact;


//通讯录的初始化
void ContactInit(Contact* con);
//通讯录的销毁
void ContactDesTroy(Contact* con);
//通讯录添加数据
void ContactAdd(Contact* con);
//通讯录删除数据
void ContactDel(Contact* con);
//通讯录的修改
```

```
void ContactModify(Contact* con);
//通讯录查找
void ContactFind(Contact* con);
//展示通讯录数据
void ContactShow(Contact* con);
```

Conatct.c

```
#define _CRT_SECURE_NO_WARNINGS 1

#include"Contact.h"
#include"SeqList.h"
#include<string.h>

void ContactInit(Contact* con)
{
    SLInit(con);
}

void ContactDesTroy(Contact* con)
{
    void SLDestroy(con);
}

void ContactAdd(Contact* con)
{
    PF info;
    printf("请输入姓名：");
    scanf("%s", info.name);
    printf("请输入电话号码：");
    scanf("%s", info.tel);
    SLPushBack(con, info);
}

int FindbyName(Contact* con, char* name)
{
    for (int i = 0; i < con->size; i++)
        if (0 == strcmp(con->Data[i].name, name))
        {
            return i;
        }
    return -1;
}

void ContactDel(Contact* con)
{
    char name[NAME_MAX];
    printf("请输入要删除的人的名字：");
    scanf("%s",name);
```

```
    int find = FindbyName(con, name);
    if (find < 0)
    {
        printf("查无此人");
        return;
    }
    void SLErase(con, find);
    printf("删除成功! \n");
}

void ContactShow(Contact* con)
{
    int i = 0;
    for (i = 0; i < con->size; i++)
    {
        printf("\n第%d位: ", i);
        printf("姓名: %s ", con->Data[i].name);
        printf("电话: %s\n", con->Data[i].tel);
    }
}

void ContactModify(Contact* con)
{
    char name[NAME_MAX];
    printf("请输入要修改的人的名字: ");
    scanf("%s",name);
    int find = FindbyName(con, name);
    if (find < 0)
    {
        printf("您输入的人不存在! ");
        return;
    }
    printf("请重新输入名字: ");
    scanf("%s",con->Data[find].name);
    printf("请重新输入电话: ");
    scanf("%s", con->Data[find].tel);
    printf("信息修改成功! \n");
}

void ContactFind(Contact* con)
{
    char name[NAME_MAX];
    printf("请输入要查找的人的名字: ");
    scanf("%s", name);
    int find = FindbyName(con, name);
    if (find < 0)
    {
        printf("您要查找的人不存在! ");
        return;
    }
    printf("姓名: %s\n",con->Data[find].name);
    printf("电话: %s\n", con->Data[find].tel);

}
```

本博客旨在记录学习过程，以后忘了随时来看。