《数据结构》上机报告

2018 年 11 月 28 日

姓名： 赵得泽 学号： 1753642 班级： 电子二班 得分：

<table>
<tr><td>实验题目</td><td colspan="2">静态链表</td></tr>
<tr><td>问题描述</td><td colspan="2">静态链表（或称为数组链表），用一个一维数组 S 存储链表的所有结点。每个结点包含 data 和 cur 两个域，data 存储数据,cur(称作游标，int 类型)代替指针，存储链表的下一个结点在数组中的相对位置。 数组前两个 s[0]存放链表头指针,s[1]存放闲置链表头指针;S[0].cur 初始为-1,表示空表。S[1].cur 初始为 2，表示第 1 个可分配的结点位置。S[MaxSize-1]=-1 表示无后继结点。初始 S[i].cur=i+1， 表示所有未分配结点已形成一个备用链表，每当进行插入时，从备用链表中取得第 1 个结点作为待插入的新结点，删除时则将从链表中删除的结点链接到备用链表中。

静态链表实现线性表的操作与动态链表相似，执行插入和删除操作时不需移动元素，只需修改指针，因此具有链表存储结构的主要优点。</td></tr>
<tr><td rowspan="2">基本要求</td><td colspan="2"></td></tr>
<tr><td>已完成基本内容（序号）</td><td></td></tr>
<tr><td rowspan="2">选做要求</td><td colspan="2">1. 本题练习用静态链表实现线性表的基本操作，包括表的初始化，尾部追加一个元素、第 i 个元素位置前插入一个新的元素、删除第 i 个元素、查找某元素、判表空、判表满、表的遍历。 元素数据类型是字符串。</td></tr>
<tr><td>已完成选做内容（序号）</td><td>1</td></tr>
<tr><td>数据结构设计</td><td colspan="2">

```
typedef struct SListNode
{
    string data;
    int cur;
}SLinkList[MaxSize];
 SLinkList s;
```

本次实验是对静态链表的使用，数据结构为线性表的顺序存储结构。通过设置两个数据域 data 和 cur 来完成各项基本操作，比如插入删除查找等等。但是它和顺序表的操作不一样，静态链表完成的实际上还是动态链表的功能，通过对 cur 域的操作来达到移动指针的效果。所以静态链表依旧可以通过改变 cur 的值来进行随意访问表中的数据，达到和动态链表同样的效果。只不过存储方式不一样罢了。</td></tr>
</table>

功能（函数）说明	/********** 函数功能：创建静态链表 函数说明：首先按照题目所给的要求进行初始化，输入的元素依次插入以第一个节点为头结点的链表；最后一个元素的cur域置-1；若有空余的结点，则是以第二个节点为头结点进行空结点的存储。除此之外还要进行特殊情况的考虑，即n=0,m=2,m=1等特殊情况下静态链表的输出情况。 **********/

```
void CreatSL(SLinkList s, int n, string str[], int m)
{
    int i;
    s[0].cur = -1;
    s[1].cur = 2;
    s[0].data = "N/A";
    s[1].data = "N/A";
    int k = 0;
    for (int i = 2; i < m; i++)
    {
        if (m - n >= 2)
        {
            if (k < n)
                s[i].data = str[k++];
            else
            {
                s[i].data = "N/A";
                k++;
            }
        }
        else
        {
            s[i].data = str[k++];
        }
        s[i].cur = i + 1;
    }
    if (m != 2)
        s[m - 1].cur = -1;
    if (m - n >= 2)
    {
        if (n == 0 && m == 2)
        {
            s[0].cur = -1;
            s[1].cur = 2;
        }
        if (n != 0)
```

```
        {
            s[1].cur = s[n + 1].cur;
            s[0].cur = 2;
            s[n + 1].cur = -1;
        }
    }
    else if (m != 2)
    {
        s[0].cur = 2;
        s[1].cur = -1;
    }
    if (m == 2 || m == 1)
    {
        s[0].cur = -1;
        s[1].cur = -1;
    }
}
/*************************************************************
函数功能：静态链表的打印
说明：静态链表的打印则利用for循环进行打印。
*************************************************************
/
void PrintSL(SLinkList &s, int m)
{
    int  k = 0;
    for (int i = 0; i < m; i++)
    {
        cout << i << " : " << s[i].data << " : " << s[i].cur <<
'\t' << '\t';
        k++;
        if (k == 3)
        {
            cout << endl;
            k = 0;
        }
    }
    if (k != 0)
        cout << endl;
}
/*************************************************************
*
函数功能：打印数据
函数说明：通过i=s[i].cur进行指针的移动，从而输出静态链表的元素。
*************************************************************
```

```
**/
void PrintData(SLinkList &s)
{
    int i = 0;
    while (s[i].cur != -1)
    {
        i = s[i].cur;
        cout << s[i].data << " ";
    }
    cout << endl;
}
/*************************************************************
函数功能：计算静态链表的长度
函数说明：同样是通过i = s[i].cur进行指针的移动，直到移到最后
一位，同时进行计数，从而得到长度。
*************************************************************/
int Len(SLinkList &s)
{
    int i = 0;
    int len = 0;
    while (s[i].cur != -1)
    {
        i = s[i].cur;
        len++;
    }
    return len;
}
/*************************************************************************
*****
函数功能：向静态链表中插入元素
函数说明：若是空链表的头结点的cur域是-1则说明静态链表已经满了，故此时不
能向其中插入元素；否则就在元素链表的头结点后面插入元素x，然后空结点链表
的一个结点被占用，再将剩下的空结点进行链接。若是在len+1的位置进行插入，
即在表尾插入，就直接进行插入。
*************************************************************************
*****/
void InsertSL(SLinkList &s, int i, string x, int &n, int m)
{
    int l, j = 0;
    int k = 1;
    if (s[k].cur == -1 || (i == 1 && s[j].cur == -1))
```

```
    {
        cout << "FULL" << endl;
        return;
    }
    if (IsEmpty(s) && i == 1)
    {
        s[2].data = x;
        int flag = s[s[k].cur].cur;
        s[s[k].cur].cur = -1;
        s[k].cur = flag;
        s[j].cur = 2;
        PrintData(s);
    }
    else if (i >= 1 && i <= Len(s) + 1)
    {
        s[j].cur = 2;
        for (l = 1; l < i; l++)
            j = s[j].cur;
        s[s[k].cur].data = x;
        int flag = s[s[k].cur].cur;
        s[s[k].cur].cur = s[j].cur;
        s[j].cur = s[k].cur;
        s[k].cur = flag;
        PrintData(s);
    }
    else
        cout << -1 << endl;
}
/*************************************************************************
*****
函数功能：从静态链表中删除元素
函数说明：若元素链表的头结点等于-1则证明链表为空，则不能进行删除操
作；
否则进行删除操作。再删除的时候要注意就是：如果空结点链表除了头节点
之外
还有空节点就将删除元素之后的结点连接到空结点链表，否则就直接将它的
cur=-1;
表示该结点是空结点链表的最后一个空节点。最后再进行删除元素之后空结
点的赋空和cur域的操作。
*************************************************************************
*****/
void DelSL(SLinkList &s, int j, int n, int m)
{
    int i, k = 0, l = 1;
```

```
    if (s[k].cur == -1)
        cout << "EMPTY" << endl;
    else
    {
        if (j < 1 || j>Len(s))
            cout << -1 << endl;
        else
        {
            for (i = 1; i < j; i++)
                k = s[k].cur;
            cout << s[s[k].cur].data << endl;
            int flag = s[s[k].cur].cur;
            if (s[s[1].cur].data == "N/A"&&s[1].cur != -1)
                s[s[k].cur].cur = s[1].cur;
            else
                s[s[k].cur].cur = -1;
            s[1].cur = s[k].cur;
            s[s[k].cur].data = "N/A";
            s[k].cur = flag;
        }
    }
}
/*************************************************************************
*****
函数功能：在静态链表中查找元素
函数说明：利用循环从头到尾遍历，找到就返回位置值，否则返回-1；
*************************************************************************
*****/
void SearchSL(SLinkList &s, string z)
{
    int i = 0, count = 0, pos = 0, j = 0;
    while (j++ <= Len(s))
    {

        count++;
        if (s[i].data == z)
        {
            pos = count - 1;
            break;
        }
        i = s[i].cur;
    }
    if (pos == 0)
        cout << -1 << endl;
```

	`` else cout << pos << endl; } /*** ***** 函数功能：在静态链表尾部追加元素 函数说明：如果空节点链表的cur=-1则表明空节点链表中已经没有空结点可 供添加 说明此时链表已满；否则，先找到空节点链表的第一个空节点（空头节点除 外）将 cur置-1，然后再让空节点链表的第一个头结点cur等于第一个空节点cur， 即将第 一个空节点从空节点链表中删除，在连接到元素链表的最后一个元素的后面 即可。 *** *****/ void AddSL(SLinkList &s, string z, int m) { int k = 1; if (s[k].cur == -1 || m == 2) cout << "FULL" << endl; else { k = s[k].cur; s[1].cur = s[k].cur; s[k].cur = -1; int j = 0; while (s[j].cur != -1)j = s[j].cur; s[j].cur = k; s[k].data = z; PrintData(s); } } ```
开发环境	Win10，C++高级程序语言设计，vs2017

<table>
<tr><td>调试分析</td><td>
<pre>
9 5
Jan Feb Mar Apr May
2 Jun
1
Mar
Oct
0 : N/A : 2   1 : N/A : 7   2 : Jan : 3
3 : Feb : 4   4 : Mar : 5   5 : Apr : 6
6 : May : -1   7 : N/A : 8   8 : N/A : -1
Jan Jun Feb Mar Apr May
Jan
3
Jun Feb Mar Apr May Oct
0 : N/A : 7   1 : N/A : 8   2 : Oct : -1
3 : Feb : 4   4 : Mar : 5   5 : Apr : 6
6 : May : 2   7 : Jun : 3   8 : N/A : -1
</pre>
<pre>
2 5
Jan Feb Mar Apr May
2 Jun
1
Mar
Oct
0 : N/A : -1   1 : N/A : -1
FULL
EMPTY
-1
FULL
0 : N/A : -1   1 : N/A : -1
</pre>
</td></tr>
<tr><td>心得体会</td><td>
通过本次对静态链表的运用，让我认识到了静态链表通过顺序存储结构来实现动态链表功能的灵活性。从而对两者之间（动态链表的静态链表）的区别和联系也有了更加深刻地体会。总之，用非指针数据实现指针的功能就是静态链表的最大特点。

还有就是根据本题目的要求，通过对空结点链表和元素链表的分开又配合的使用，我也学会了如何更加熟练的对静态链表进行插入、删除、查找等操作。
</td></tr>
</table>