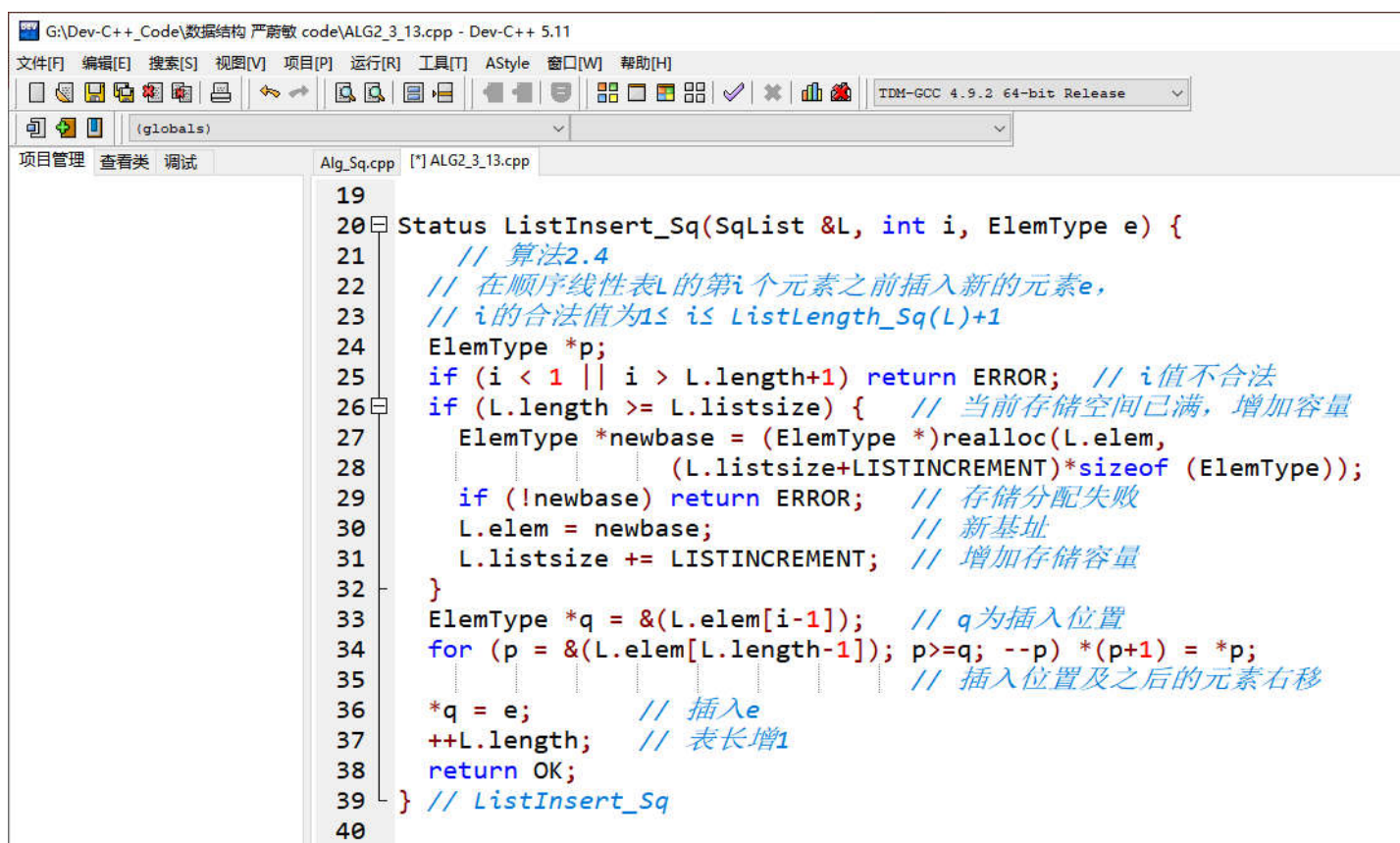


顺序表的算法设计 示例

◆ 关于编译器的选择

- Dev-C++ 5.11 TDM-GCC 4.9.2 Setup.exe
- codeblocks-17.12mingw-setup.exe



```
G:\Dev-C++_Code\数据结构 严蔚敏 code\ALG2_3_13.cpp - Dev-C++ 5.11
文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] AStyle 窗口[W] 帮助[H]
(globals)
项目管理 查看类 调试
Alg_Sq.cpp [*] ALG2_3_13.cpp
19
20 Status ListInsert_Sq(SqList &L, int i, ElemType e) {
21     // 算法2.4
22     // 在顺序线性表L的第i个元素之前插入新的元素e,
23     // i的合法值为1≤ i≤ ListLength_Sq(L)+1
24     ElemType *p;
25     if (i < 1 || i > L.length+1) return ERROR; // i值不合法
26     if (L.length >= L.listsize) { // 当前存储空间已满, 增加容量
27         ElemType *newbase = (ElemType *)realloc(L.elem,
28             (L.listsize+LISTINCREMENT)*sizeof (ElemType));
29         if (!newbase) return ERROR; // 存储分配失败
30         L.elem = newbase; // 新基址
31         L.listsize += LISTINCREMENT; // 增加存储容量
32     }
33     ElemType *q = &(L.elem[i-1]); // q为插入位置
34     for (p = &(L.elem[L.length-1]); p>=q; --p) *(p+1) = *p;
35     // 插入位置及之后的元素右移
36     *q = e; // 插入e
37     ++L.length; // 表长增1
38     return OK;
39 } // ListInsert_Sq
40
```

【示例-1】课本中插入算法与删除算法的演示。

```
#include <bits/stdc++.h>
using namespace std;
/* 添加其他算法代码 */
void dispList(SqList L)
{
    for(int i = 1; i <= L.length; i++)
        printf("%d ", L.elem[i - 1]);
}
int main()
{
    int A[8] = {1, 2, 3, 4, 5, 100, 200, 300};
    int i, j, e = 586;
    SqList List;
    InitList_Sq(List);
```

【示例-1】课本中插入算法与删除算法的演示。

```
for(i = 1, j = 0; i <= 8; i++, j++)  
    ListInsert_Sq(List, i, A[j]);  
printf("\n插入之前的元素序列为: \n");  
dispList(List);  
i = 3;    //插入位置  
ListInsert_Sq(List, i, e);  
printf("\n\n插入后的元素序列为: \n");  
dispList(List);  
i = 6;    //删除位置  
printf("\n\n删除后的元素序列为: \n");  
ListDelete_Sq(List, i, e);  
dispList(List);  
}
```

运行结果

```
插入之前的元素序列为：
1 2 3 4 5 100 200 300

插入后的元素序列为：
1 2 586 3 4 5 100 200 300

删除后的元素序列为：
1 2 586 3 4 100 200 300

Process returned 0 (0x0)   execution time : 0.377 s
Press any key to continue.
```

【示例-2】 假设有一个顺序表L，其中元素为整数且所有元素值均不相同。设计一个算法将最大值元素与最小值元素交换。

解：算法思路

- 用maxi和mini记录顺序表L中最大值元素和最小值元素的下标，初始时maxi=mini=0。
- i从1开始扫描所有元素：当
L.elem[i]>L.elem[maxi]时置maxi=i；否则若
L.elem[i]<L.elem[mini]时置mini=i。
- 扫描完毕时，L.elem[maxi]为最大值元素，
L.elem[mini]为最小值元素，将它们交换。

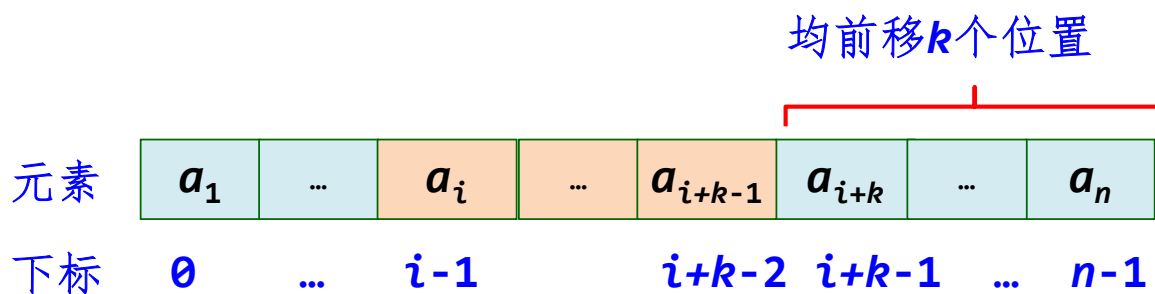
```
void swap(ElemType &x, ElemType &y) //交换x和y
{
    ElemType tmp=x;
    x=y; y=tmp;
}

void SwapMaxMin(SqList &L) //交换L中最大值元素与最小值元素
{
    int i,maxi,mini;
    maxi=mini=0;
    for (i=1;i<L.length;i++)
        if (L.elem[i]>L.elem[maxi])
            maxi=i;
        else if (L.elem[i]<L.elem[mini])
            mini=i;
    swap(L.elem[maxi],L.elem[mini]);
}
```

【示例-3】 设计一个算法，从线性表中删除自第 i 个元素开始的 k 个元素，其中线性表用顺序表 L 存储。

解：算法思路

将线性表中 $a_i \sim a_{i+k-1}$ 元素（对应 $L.elem[i-1..i+k-2]$ 的元素）删除，即将 $a_{i+k} \sim a_n$ （对应 $L.elem[i+k-1..n-1]$ ）的所有元素依次前移 k 个位置。

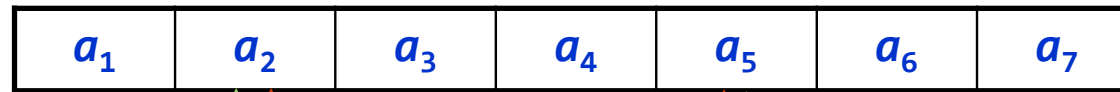



```
int Deletek(SqList &L,int i,int k)
{
    int j;
    if (i<1 || k<1 || i+k-1>L.length)
        return 0;                //判断i和k是否合法
    for (j=i+k-1;j<L.length;j++)  //将元素前移k个位置
        L.elem[j-k]=L.elem[j];
    L.length-=k;                  //L的长度减k
    return 1;
}
```

【示例-4】 已知线性表 (a_1, a_2, \dots, a_n) 采用顺序表 L 存储，且每个元素都是互不相等的整数。设计一个将所有奇数移到所有的偶数前边的算法（要求时间最少，辅助空间最少）。

解： 算法思路

算法设计思路：置 $i=0$ ， $j=n-1$ ，在顺序表 L 中从左向右找到偶数 $L.elem[i]$ ，从右向左找到奇数 $L.elem[j]$ ，将两者交换；循环这个过程直到 i 等于 j 为止。



指向偶数: i 交换 j : 指向奇数

```
void Move(SqList &L)
{ int i=0,j=L.length-1;
  while (i<j)
  { while (L.elem[i]%2==1) i++;    //从前向后找偶数
    while (L.elem[j]%2==0) j--;    //从后向前找奇数
    if (i<j)
      swap(L.elem[i],L.elem[j]); //交换这两元素
  }
}
```

【示例-5】 已知一个整数线性表采用顺序表L存储。
设计一个尽可能高效的算法删除其中所有值为负整数的
元素（假设L中值为负整数的元素可能有多）。

解：算法思路

采用整体重建顺序表的算法思路, 仅仅将插入元素
的条件设置为“元素值 ≥ 0 ”即可。

```
void DeleteMinus(SqList &L)
{
    int i, k=0;
    for (i=0;i<L.length;i++)
        if (L.elem[i]>=0) //将不为负数的元素插入到L中
        {
            L.elem[k]=L.elem[i];
            k++;
        }
    L.length=k; //重置L的长度
}
```

【示例-6】 设将 n ($n > 1$) 个整数存放于一维数组 R 中。试设计一个时间和空间两方面尽可能高效的算法，将 R 中整数序列循环左移 p ($0 < p < n$) 个位置，即将 R 中的数据序列 $(X_0, X_1, \dots, X_{n-1})$ 变换为 $(X_p, X_{p+1}, \dots, X_{n-1}, X_0, X_1, \dots, X_{p-1})$ ，要求：

- (1) 给出算法的基本设计思想。
- (2) 根据设计思想，采用C、C++或Java语言描述算法，关键之处给出注释。
- (3) 说明你所设计算法的时间复杂度和空间复杂度。

本题为2010年全国考研题。

解：算法思路

(1) 设 $R=(X_0, X_1, \dots, X_p, X_{p+1}, \dots, X_{n-1})$ ，记 $A=(X_0, X_1, \dots, X_{p-1})$ （共有 p 个元素）， $B=(X_p, \dots, X_{n-1})$ （共有 $n-p$ 个元素），设计逆置算法 $\text{reverse}(R)$ ，用于原地逆置数组 R ，则 A 原地逆置后 A' 变为 $(X_{p-1}, \dots, X_1, X_0)$ ， B 原地逆置后 B' 变为 $(X_{n-1}, \dots, X_{p+1}, X_p)$ ，也就是说 $A'B'=(X_{p-1}, \dots, X_1, X_0, X_{n-1}, \dots, X_{p+1}, X_p)$ ，再将 $A'B'$ 原地逆置变为 $(X_p, X_{p-1}, \dots, X_{n-1}, X_0, X_1, \dots, X_{p-1})$ 即为所求，即进行3次逆置操作：

$$\text{reverse}(R) = \text{reverse}(\text{reverse}(A), \text{reverse}(B))$$

(2) 对应的算法如下:

```
void reverse(int R[],int m,int n)           //将R[m..n]逆置
{
    int i;
    int tmp;
    for (i=0;i<(n-m+1)/2;i++)
    {
        tmp=R[m+i];           //将R[m+i]与R[n-i]进行交换
        R[m+i]=R[n-i];
        R[n-i]=tmp;
    }
}

int ListReverse(int R[],int n,int p)        //循环左移
{
    if (p<=0 || p>=n)
        return 0;
    else
    {
        reverse(R,0,p-1);
        reverse(R,p,n-1);
        reverse(R,0,n-1);
        return 1;
    }
}
```


(3) $\text{reverse}(\mathbf{R}, m, n)$ 算法的时间复杂度为 $O(n-m)$, 所以
 $\text{ListReverse}(\mathbf{R}, n, p)$ 算法的时间复杂度 $= O(p) + O(n-p) + O(n) = O(n)$ 。

另外, $\text{ListReverse}(\mathbf{R}, n, p)$ 算法中只定义几个变量, 所以空间复杂度为 $O(1)$ 。



— END —