```cpp
#include <stdio.h>
#include <stdlib.h>

#define EQ(a, b) ((a) == (b))
#define LT(a, b) ((a) <  (b))
#define LQ(a, b) ((a) <= (b))

enum Status
{
    OVERFLOW = -1,
    ERROR    =  0,
    OK       =  1
} ;

////////////////////////////////////////////////////
//线性表顺序存储结构
typedef int KeyType;
typedef struct
{
    KeyType key;
}RecType;

typedef struct
{
    RecType* elem;
    int length;
    int first;
    int last;
}STable;

////////////////////////////////////////////////////
//函数声明
Status CreatSTable(STable& ST, bool nul);
void   TRInsertionSort(STable& ST);
int    FindInsertPos(STable& ST, KeyType key, int low, int high, bool kind);
void   PrintSTable(STable& ST);

////////////////////////////////////////////////////
//主函数
int main()
{
    STable ST;
    ST.elem = NULL;

    char chos;

    while(1)
    {
        printf("1.创建查找表\n2.2-路插入排序\n3.打印查找表\n4.退出\n");
```

```
50          printf("请选择: ");
51          scanf_s("%c", &chos);
52          while ((chos == 10) || (chos == 32))
53          {
54              scanf_s("%c", &chos);
55          }
56
57          switch(chos)
58          {
59          case '1':
60              {
61                  if (ST.elem != NULL)
62                  {
63                      free(ST.elem);
64                  }
65
66                  printf("\n请输入查找表长度: ");
67                  scanf_s("%d" ,&ST.length);
68                  CreatSTable(ST, false);
69              }break;
70          case '2':
71              {
72                  TRInsertionSort(ST);
73              }break;
74          case '3':
75              {
76                  PrintSTable(ST);
77              }break;
78          case '4':
79              {
80                  if (ST.elem != NULL) free(ST.elem);
81                  exit(0);
82              }break;
83          default:
84              {
85                  printf("CHOS ERROR!\n");
86              };
87          }
88
89          putchar(10);
90      }
91
92      return 0;
93  }
94
95  ////////////////////////////////////////////////
96  //函数定义
97  Status CreatSTable(STable& ST, bool nul)
98  {
```

```
99      ST.elem = (RecType*) malloc ((ST.length + 1) * sizeof(RecType));    //零号单元另作他用
100
101     if (ST.elem == NULL) return OVERFLOW;
102
103     if (!nul)   //不创建空表
104     {
105         printf ("请输入%d个关键字: ", ST.length);
106         for (int i=1; i<=ST.length; ++i)
107         {
108             scanf_s("%d", &ST.elem[i].key);
109         }
110
111         ST.first = 1;
112         ST.last  = ST.length;
113     }
114     else
115     {
116         ST.first = ST.last = 1;
117     }
118     return OK;
119 }
120
121 void   TRInsertionSort(STable& ST)
122 {
123     int insertPos;
124     STable AT;      //辅助空间
125     AT.length = ST.length;
126     CreatSTable(AT, true);
127     AT.elem[1] = ST.elem[1];
128
129     for (int i=2; i<=ST.length; ++i)
130     {
131         if (LT(ST.elem[i].key, ST.elem[1].key))     //插入位在后半部分
132         {
133             if (AT.first == 1)
134             {
135                 insertPos = AT.length;
136                 AT.first = insertPos;
137             }
138             else
139             {
140                 insertPos = FindInsertPos(AT, ST.elem[i].key, AT.first,
                      AT.length, true);
141
142                 for (int i = AT.first; i <= insertPos; ++i)     //记录前移,空出插入位
143                 {
144                     AT.elem[i - 1] = AT.elem[i];
```

```
145                 }
146                 --AT.first;
147             }
148         }
149         else    //插入位在前半部分
150         {
151             insertPos = FindInsertPos(AT, ST.elem[i].key, 1, AT.last, false);
152             for (int i = AT.last; i >= insertPos; --i)      //记录后移,空出插入位
153             {
154                 AT.elem[i + 1] = AT.elem[i];
155             }
156             ++AT.last;
157         }
158         printf ("frist: %d  last: %d  插入位：%d \n", AT.first, AT.last,
              insertPos);
159
160         AT.elem[insertPos] = ST.elem[i];
161     }
162
163     //拷贝
164     for (int i=0; i<=AT.length; ++i)
165     {
166         ST.elem[i] = AT.elem[i];
167     }
168     ST.first = AT.first;
169     ST.last = AT.last;
170 }
171
172 int FindInsertPos(STable& ST, KeyType key, int low, int high, bool kind)
      //折半查找插入位
173 {
174     int insertPos;
175     int mid;
176
177     while (low <= high)
178     {
179         mid = (low + high) / 2;
180         if (LT (key, ST.elem[mid].key)) high = mid - 1;
181         else                            low  = mid + 1;
182     }
183
184     if (kind == true)   insertPos = high;
185     else                insertPos = high + 1;
186
187     return insertPos;
188 }
189
190 void   PrintSTable(STable& ST)
191 {
```

```
192     for (int i=ST.first; i!=ST.last; ++i)
193     {
194         printf("%d ", ST.elem[i]);
195         if (i >= ST.length)
196         {
197             i = 0;
198         }
199     }
200     printf("%d ", ST.elem[ST.last]);
201     putchar (10);
202 }
```