实验二　顺序表实验

【实验目的】

（1）掌握顺序表的存储结构及应用。

（2）掌握用顺序表表示数据、并进行有关算法设计的方法。

【实验背景】

顺序表中，逻辑上相邻的节点其物理存储位置也相邻，节点间的逻辑关系由存储单元的邻接关系来体现。我们可以用数组存储表中元素，并同时记录顺序表中的最后一个元素在数组中的下标。用 C 语言描述顺序表的数据类型如下：

```
＃define maxlen 100
typedef struct{
      Datatype data[maxlen]；
      int last；
}Sequenlist；
```

这样，顺序表的基本运算分别描述为：

1　置空表运算

```
 void SqLsetnull (Sequenlist *L){
      L－>last = -1；
 }
```

2　建空表运算

```
Sequenlist *SqLsetnull(){
      Sequenlist *L；
      L=malloc( sizeof(Sequenlist))；
      L－>last = -1；
      return ( L)；
}
```

3　求表长运算

```
int   SqLlength (Sequenlist *L )
{      return ( L－>last＋1)；
}
```

4　按序号取元素运算

```
Datatype SqLget(Sequenlist *L ， int i){
      Datatype x；
      if(i<1 || i>SqLlength(L) ) printf(“超出范围”)；
      else x=L－>data[i –1]；
      return (x)；
}
```

5　按值查找运算

```
void SqLlocate (Sequenlist *L ， Datatype x){
      int i，z=0；
      for( i=0；i< SqLlength (L)；i++)
            if( L－>data[i]= =x ){
```

```
                printf("%d  ", i+1);
                z =1;
            }
        if( z= =0 ) printf("%d  ", -1);
    }
```

6 判表满运算

```
int SqLempty(Sequenlist *L){
        if( L—>last+1> = maxlen ) return (1);
        else return ( 0);
     }
```

7 插入数据元素运算

```
int SqLinsert (Sequenlist *L, int i, Datatype x ){
        int j ;
        if(SqLempty(L)= = 1){
            printf("overflow");
            return (0);
        }
        else if ( (i<1)||(i >L->last +2)) {
            printf ("error");
            return (0);
        }
        else{
            for (j=L->last; j>= i –1; j--)
                L->data[ j+1] = L->data[ j];
            L->data[i –1] = x;
            L->last = L->last+1;
            return (1);
        }
    }
```

在该算法中，i 是顺序表中数据元素的序号，而 L->last 表示表中最后一个数据元素在数组中的下标。

8 删除数据元素运算

```
int SqLdelete(Sequenlist *L, int i){
        int j;
        if(L->last <0){                    //表空
            printf ( "顺序表空！"); return (0);
        }
        else if ( (i<1)||(i >L->last +1) ){
            printf ("i 参数出错！"); return (0);
        }
        else {
            for( j=i; j<=L->last+1; j++)
```

```
                L->data[ j-1] =L->data[ j];
            L->last - - ；return (1)；
        }
    }
```

该算法中 i 是顺序表中数据元素的序号，而 L->last 表示表中最后一个数据元素在数组中的下标，表长 n = L->last+1。

【实验任务】

1、程序验证

（1）建立含有若干个元素的顺序表，并实现顺序表的插入、删除、查找等操作。

（2）阅读下列程序，指出算法的功能，写出其运行结果，并通过运行来验证。

```
#include  “malloc.h”
#define  maxlen 50
typedef struct{
      int data[maxlen]；
      int last；
}Sequenlist；
Sequenlist  *ABC(Sequenlist  *A ,  Sequenlist  *B){
   int  i,  j;
   Sequenlist  *C;
   C = malloc(sizeof(Sequenlist));
   C->last = -1;
   for(i=0;  i<=A->last;  i++)
      for(j=0;  j<=B->last;  j++) {
         if(A->data[i]= = B->data[j]) {
            C->last++;
            C->data[C->last] = A->data[i];
            break;
         }
   return  C;
}
Sequenlist  *SqLset( ){
      Sequenlist *L；
      int  i;
      L=malloc( sizeof(Sequenlist))；
      L->last = -1；
      scanf(“%d”, &i);     //输入表长
      if( i>0) {
          for(L->last=0;  L->last <i;  L->last ++)
             scanf(“%d”, & L->data[L->last]);
          L->last--；
      }
      return ( L)；
}
main( ) {
  Sequenlist  *A , *B, *C;
  int  i;
```

```
    A = SqLset( );
    B = SqLset( );
    C = ABC(A, B);
    for(i=0;  i<=C->last;  i++)
      printf("%4d", C->data[i] );
  }
```

（3）下面算法的预定功能是实现顺序表的倒置，试检查其中是否有错；若有错，指出错误所在，并修改之，然后通过运行来验证。

```
    #include  "malloc.h"
    #define  maxlen 50
    typedef struct{
        int data[maxlen];
        int last;
    }Sequenlist;
    Sequenlist  *SqLset( ){
        Sequenlist *L;
        int  i;
        L=malloc( sizeof(Sequenlist));
        L->last = -1;
        scanf("%d", &i);     //输入表长
        if( i>0) {
            for(L->last=0;  L->last <i;  L->last ++)
              scanf("%d", & L->data[L->last]);
        }
        return ( L);
    }
    Sequenlist  *reverse(Sequenlist  *L) {
        int  i,  j,  x;
        for(i=0, j=L->last-i+1;  i<= j;  i ++ ) {
          x = L->data[i];  L->data[i]= L->data[j]; L->data[j]=x;
        }
        return ( L);
    }
    main( ) {
      Sequenlist  *A;
      int  i;
      A = SqLset( );
      for(i=0;  i<=A->last;  i++)
        printf("%4d", A->data[i] );
      printf("\n");
      A = Sequenlist ( );
      for(i=0;  i<=A->last;  i++)
        printf("%4d", A->data[i] );
      printf("\n");
}
```

2、算法填空

请在下面算法的空格处填入适当内容，以使算法能求出顺序表中的最大和最小值，并通过运行来验证。

```
#include  “malloc.h”
#define  maxlen 50
typedef struct{
      int data[maxlen];
      int last;
}Sequenlist;
Sequenlist  *SqLset( ){
      Sequenlist *L;
      int  i;
      L=malloc( sizeof(Sequenlist));
      L->last = -1;
      scanf(“%d”, &i);     //输入表长
      if( i>0) {
          for(L->last=0;  L->last <i;  L->last ++)
             scanf(“%d”, & L->data[L->last]);
      }
      return ( L);
}
void  maxmin(Sequenlist  *L) {
     int  min,  max,  i;
     if( _____ ) {
          max = min = L->data[0];
          for(i =1;  i <= _____ ;  i ++) {
               if( max < L->data[i])
                    max = L->data[i];
               if( min > L->data[i])
                    min = L->data[i];
          }
     printf(“max=%d, min= %d\n”, max, min );
}
main( ) {
   Sequenlist  *A;
   A = SqLset( );
   maxmin(A);
}
```

3、算法设计

（1）设计算法实现删除顺序表中多余重复元素。如：对于顺序表（1，2，3，1，3，4，3，5），删除第四个元素 1 及第五、第七个元素 3。

（2）设计算法，实现在一个递增有序的顺序表的适当位置插入元素 x，使得该顺序表仍然递增有序。分析算法的时间复杂度。

4、实例演练：通讯录管理程序

[问题描述]

设计一个通讯录管理程序，使其具备通讯者的插入、删除、简单查询，以及通讯录表的输出等功能。

[基本要求]

①将通讯录表设计为一个顺序表，记录的信息包括编号、姓名、性别、电话和地址；在该顺序表中实现通讯者信息的插入、删除、简单查询，以及通讯录表的输出等功能。

② 设计算法完成问题求解；

③ 分析算法的时间复杂度和空间复杂度。

[实现提示]

为实现通讯录管理的几种操作功能，首先设计一个含有多个菜单项的主控菜单程序，然后再为这些菜单项配上相应的功能。

①主控菜单设计

菜单内容包括：1 通讯录表的建立
2 通讯者的插入
3 通讯者的删除
4 通讯者的查询
5 通讯录表的输出
0 退出管理程序

使用数字 0－5 来选择菜单项，其它输入则不起作用。输入的数字假设用变量 sn 存储，使用函数 menu_select( )接受数字输入，该函数的返回值提供给主函数；则主函数用应使用 for 循环实现重复选择，以实现不同的菜单功能。

```
/*菜单选择函数*/
int   menu_select( ) {
      int   sn;
      printf(“     通讯录管理系统\n”);
      printf(“==================\n”);
      printf(“   1  通讯录表的建立\n”);
      printf(“   2  通讯者的插入\n”);
      printf(“   3  通讯者的删除\n”);
      printf(“   4  通讯者的查询\n”);
      printf(“   5  通讯录表的输出\n”);
      printf(“   0  退出管理系统\n”);
      printf(“==================\n”);
      printf(“     请   选   择 0－5：\n”);
      for(   ;   ; )
      {   scanf( “%d”, &sn);
          if( sn<0|| sn>5 )
             printf(“\n\t 输入错误，重选 0－5：\n”);
          else
                 break;
       }
       return   sn;
   }
/*主控菜单处理函数（主函数）*/
void     main(   )
{        for(   ;   ; )
           {   switch( menu_select( ) )
               {case   1:   printf(“   通讯录表的建立\n”);
                            调用建立通讯录表函数；
                            break;
```

```
                case  2:  printf("  通讯者的插入\n");
                          调用通讯者的插入函数;
                          break;
                case  3:  printf("  通讯者的删除\n");
                          调用通讯者的删除函数;
                          break;
                case  4:  printf("  通讯者的查询\n");
                          调用通讯者的查询函数;
                          break;
                case  5:  printf("  通讯录表的输出\n");
                          调用通讯录表的输出函数;
                          break;
                case  0:  printf("  再见\n");
                          exit(0);
            }
        }
    }
```

②功能函数设计

功能函数包括建立通讯录表、通讯者的插入、通讯者的删除、通讯者的查询以及通讯录表的输出等 5 个函数。

可以先定义通讯录表的结点类型和通讯录表类型：

```
#include  SIZE  50
typedef  struct {
    char  num[5];        //编号
    char  name[10];      // 姓名
    char  sex[3];        // 性别
    char  phone[15];     // 电话
    char  addr[30];      // 地址
} Datatype;
typedef struct{
     Datatype  tx[SIZE];;
     int last;
}Tx_Seqlist;
Tx_Seqlist  *L;
```

接下来，分别定义对顺序表 L 的建立、插入、删除、查询及输出函数，请读者自行设计完成。

[思维扩展]

对该顺序表的查询扩展为按姓名查询、按电话号码查询、按地址查询等不同的查询方法。