实验指导书 1 顺序表操作

一、实验目的：

1．掌握线性表的基本操作函数

2．掌握顺序存储的概念,学会对顺序存储数据结构进行操作

二、实验学时： 2 学时

三、实验内容和步骤：

1． 编写线性表的基本操作函数

(1)初始化线性表 InitList

(2)向线性表制定位置插入元素

(3)删除指定元素值的线性表记录 DeleteList1

(4)删除指定位置的线性表记录 DeleteList2

(5)查找线性表中的元素 FindList

(6)输出线性表元素 OutputList

2．调用上述函数实现下列操作,步骤如下

(1)初始化线性表

(2)调用插入函数建立一个线性表

(3)在线性表中寻找指定的元素

(4)在线性表中删除指定值的元素

(5)在线性表中删除指定位置的元素

(6)遍历并输出线性表

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

1、定义顺序表的存储结构

```
struct LinearList        /*定义线性表结构*/
{
    int *list;           /* 存线性表元素 */
    int size;            /* 存线性表长度 */
    int MaxSize;           /* 存 list 数组元素个数 */
};
typedef struct LinearList LIST;
```

注：以上定义与 p.17 的定义其实是一样的

2、构建一个空的顺序表

```
void InitList( LIST *L, int ms )    /* 初始化线性表 */
{
    if( (L->list =        1              ) == NULL ) {
        printf( "内存申请错误!\n" );
        exit( 1 );
    }
        2
    L->MaxSize = ms;
}
```

3、调用 main 函数构建一个空顺序表，并调试和运行程序

```
void main()
{
    LIST LL;
    int i, r;
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize);
    InitList( &LL, 100 );
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize);
}
```

4、顺序表的插入函数

```
int InsertList( LIST *L, int item, int rc )
/* item:记录值  rc:插入位置  */
{
        int i;
    if(       3          ) /* 线性表已满 */
        return -1;
    if( rc < 0 )                /* 插入位置为 0 --> L->size */
        rc = 0;
    if(        4        )
        rc = L->size;
    for( i = L->size - 1; i >= rc; i-- )    /* 将线性表元素后移 */
            5
    L->list[rc] = item;
    L->size ++;
    return 0;
}
```

5、输出顺序表中元素

```
void OutputList( LIST *L )      /* 输出线性表元素 */
{
    int i;
    for( i = 0;      6              i++ )
        printf( "%d ", L->list[i] );
        printf( "\n" );
}
```

6、调用 main 函数试运行插入函数 InsertList

```
  void main()
{
    LIST LL;
        int i, r;
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize);
    InitList( &LL, 100 );
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize);
    while( 1 )
```

```
    {
        printf( "请输入元素值，输入 0 结束插入操作:" );
        fflush( stdin );    /* 清空标准输入缓冲区 */
        scanf( "%d", &i );
        if(      1          )
                    break;
        printf( "请输入插入位置：" );
        scanf( "%d", &r );
        InsertList(      2        );
        printf( "线性表为: " );
        _____3_____
    }
}
```

7、**顺序表的删除算法**

```
int DeleteList1( LIST *L, int item )
/* 删除指定元素值的线性表记录，返回>=0：删除成功 */
{
    int i, n;
    for( i = 0; i < L->size; i++ )
        if( item == L->list[i] )    /* 找到相同的元素 */
            break;
    if( i < L->size ) {
        for( n = i; n < L->size - 1; n++ )
            L->list[n] = L->list[n+1];
        L->size --;
        return i;
    }
        return -1;
}
```

8、**main 函数试调用删除函数**

```
void main()
{
    LIST LL;
     int i, r;
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize );
    InitList( &LL, 100 );
    printf("list addr=%p\tsize=%d\tMaxSize=%d\n", LL.list, LL.size, LL.MaxSize );
    while( 1 )
    {
        printf( "请输入元素值，输入 0 结束插入操作:" );
        fflush( stdin );    /* 清空标准输入缓冲区 */
        scanf( "%d", &i );
        if(      1          )
                    break;
```

```
        printf( "请输入插入位置：" );
        scanf( "%d", &r );
        InsertList(      2        );
        printf( "线性表为:  " );
              3
    }
    while( 1 )
    {
        printf( "请输入删除元素值，输入 0 结束查找操作:" );
        fflush( stdin );    /* 清空标准输入缓冲区 */
        scanf( "%d", &i );
        if( i == 0 )
                    break;
        r =      5
                if( r < 0 )
            printf( "没找到\n" );
        else {
            printf( "有符合条件的元素，位置为：%d\n 线性表为：", r+1 );
            OutputList( &LL );
                }
    }
```

}

四、实验类型：验证

五、实验要求：

1、C++/C 完成算法设计和程序设计并上机调试通过。

2、撰写实验报告，提供实验结果和数据。

3、分析算法，要求给出具体的算法分析结果，并简要给出算法设计小结和心得。

实验指导书 2 链表操作

一、实验目的：

1．掌握链表的基本操作函数

2．掌握链式存储的概念,学会对链式存储数据结构进行操作

二、实验学时： 2 学时

三、实验内容和步骤：

1． 编写链表的基本操作函数

(1) 建立一个带头节点的单链表

(2)向链表指定位置插入元素

(3) 删除指定位置的链表记录

(5)查找链表中的元素 FindList

(6)输出线性表元素 OutputList

2．调用上述函数实现下列操作,步骤如下

(1)初始化线性表调用建表函数建立一个链表

(3)在链表中寻找指定的元素

(4)在链表的指定位置中插入指定值的元素

(5)在链表中删除指定位置的元素

(6)遍历并输出链表

一、定义单链表的数据类型

```
#include<stdio.h>
#include<alloc.h>
typedef struct Node / *  结点类型定义  * /
{int    data;
 struct Node    * next;
}LNode, *LinkList;
```

二、建立一个带头节点的单链表

```
LinkList Create_LinkList3()      /*  建立一个带头节点单链表  */
{


}
Void    main（）{
      LinkList L=Create_LinkList3();
}
```

三、顺序输出单链表中所有元素的值

```
void OutputList( LinkList p)            /*  输出链表结点的键值  */
{
      while(_________) {
            printf( "%d", p->data );
            p = p->next;          /*  遍历下一个结点  */
```

```
            }
    }
    Void    main（）{
        LinkList L=Create_LinkList3();
        OutputList(L);
    }
```

四、按序号查找第ｉ个元素

```
LNode *Get_LinkList(LinkList L,int i)
{
    LNode *p=L;
    int    j=0;
    while(p->next!=NULL&&j<i)
    { p=p->next;
    j++;            }
    if( j==i)
        return      p;
    else
        return       NULL;
}
```

五、单链表的插入操作，向链表Ｌ的第ｉ个位置上插入元素ｘ

```
void InsertList1(LinkList L, int i, int x )
 {

}
```

六、 单链表的删除操作，删除单链表Ｌ上的第ｉ个数据节点/

```
int DeleteList(LinkList L, int i )
{
}
```

七、主函数调用插入和删除操作

```
void main(){
        LinkList L=Create_LinkList3();
        OutputList(L);

            //LinkList p;
            int i,r,op,rc;
            while( 1 )
            {
        printf( "请选择操作   1：插入操作   2: 删除操作  -1：退出  \n" );
        fflush( stdin );      /*  清空标准输入缓冲区  */
        scanf( "%d", &op );
```

```
        switch( op ) {
            case -1:        /*  退出  */
                return;
            case 1:         /*  指定位置追加结点  */
                printf( "请输入插入的元素值：" );
                scanf( "%d", &r);
                printf("请输入插入元素的位置");
                scanf( "%d", &i );
                Insert_LinkList(L,i,r);
                OutputList(L);
                break;
            case 2:         /*  删除结点  */
                printf( "请输入要删除结点的键值：" );
                scanf( "%d", &i );
                rc =Del_LinkList(L,i);
                if( rc == 0 )
                    printf( "删除成功\n", rc );
                else
                    printf( "没找到\n" );
                OutputList(L);
                break;


    }
}

}
```

四、实验类型：验证

五、实验要求：

1、C++/C 完成算法设计和程序设计并上机调试通过。

2、撰写实验报告，提供实验结果和数据。

3、分析算法，要求给出具体的算法分析结果，并简要给出算法设计小结和心得。

实验指导书 3　　约瑟夫问题

一、实验环境

Windows xp 操作系统，visual c++软件

一、实验目的

1、掌握线性表的特点。

2、掌握线性表存储结构特点。

3、掌握线性表链式存储结构的基本运算。

二、实验类型：设计性实验

三、实验学时：2

四、实验内容和步骤

约瑟夫问题：已知 n 个人围坐在一张圆桌周围，现在从序号为 k 的人开始报数，数到 m 的那个人出列；他的下一个人又从 1 开始报数，数到 m 的那个人又出列；依次规则重复下去，直到圆桌周围的人全部出列为止。

请应用所学的(循环)链表相关知识写出 c 程序,能根据输入的 n,k,m 输出 n 个人的出列顺序。

五、实验要求：

1、C++/C 完成算法设计和程序设计并上机调试通过。

2、撰写实验报告，提供实验结果和数据。

3、分析算法，要求给出具体的算法分析结果，包括时间复杂度和空间复杂度，并简要给出算法设计小结和心得。

实验指导书 4 栈及栈的应用

一、实验目的：

1．掌握用 C 语言调试程序的基本方法。

2．掌握栈的顺序表示表示的实现

3、掌握递归函数的实现过程

二、实验学时： 2 学时

三、实验内容和步骤：

1．栈在顺序存储结构上的插入元素，删除元素运算

（1）定义栈的数据类型

（2）写出初始化栈和出栈、入栈的程序

2．在步骤 1 的基础上实现课本 P.48 的数制转换程序

3．写出汉诺塔程序，并利用调试理解递归的入栈出栈的过程。

四、实验类型：验证

五、实验要求：

1、C++/C 完成算法设计和程序设计并上机调试通过。

2、撰写实验报告，提供实验结果和数据。

3、分析算法，要求给出具体的算法分析结果，并简要给出算法设计小结和心得。

实验指导书 5 树的操作

一、实验目的：

1．进一步掌握树的结构及非线性特点，递归特点和动态性。

2．进一步巩固对指针的使用和二叉树的三种遍历方法、建立方法。

二、实验内容：

1．二叉树的实现(按先序遍历建立二叉链表)

2．二叉树的遍历(递归)

三、实验要求：

1．用 C++/C 完成算法设计和程序设计并上机调试通过。

2．撰写实验报告，提供实验结果和数据。

3．分析算法，要求给出具体的算法分析结果，并简要给出算法设计小结和心得。

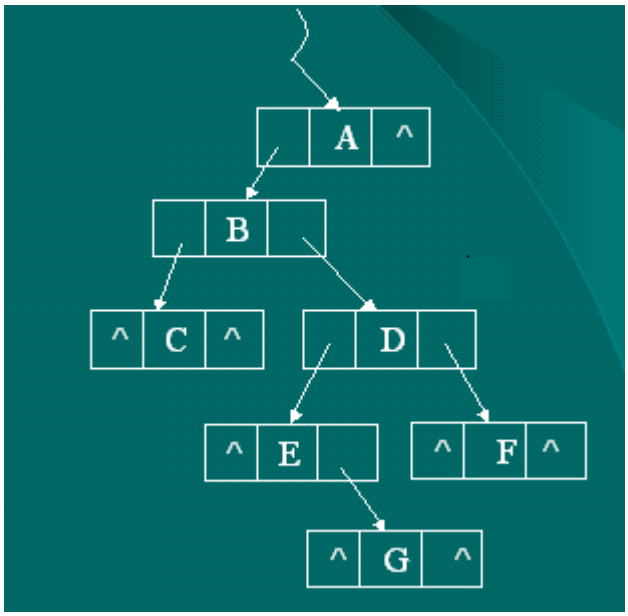
四、实验类型：验证

五、实验学时：2

六、程序实现：

写出每个操作的算法（操作过程）

程序运行情况，写出输入数据及运行结果。



实验指导书 6 图的操作

一、实验目的：

1．进一步掌握图的结构及非线性特点，递归特点和动态性。

2．进一步巩固对指针的使用和图的两种遍历方法、建立方法。

二、实验内容：

1．图实现

2．图遍历

三、实验要求：

1．用 C++/C 完成算法设计和程序设计并上机调试通过。

2．撰写实验报告，提供实验结果和数据。

3．分析算法，要求给出具体的算法分析结果，包括时间复杂度和空间复杂度，并简要给出算法设计小结和心得。

三、实验学时：2

四、实验类型：验证

五、程序实现：

写出每个操作的算法（操作过程）

程序运行情况，写出输入数据及运行结果。

实验指导书 7 查找

实验目的:

1、掌握查找的不同方法，并能用高级语言实现查找算法。

2、熟练掌握顺序表和有序表的查找方法以及静态查找树的构造方法和查找算法，理解静态查找树的折半查找的方法。

3、熟练掌握二叉树的构造和查找方法。

实验内容:

设计一个读入一串整数构成一棵二叉排序树算法。

实现提示：二叉排序树的构成，可从空的二叉树开始，每输入一个结点数据，就建立一个新结点插入到当前已生成的二叉排序树中，所以它的主要操作是二叉排序树插入运算。在二叉排序树中插入新结点，只要保证插入后仍符合二叉排序树的定义即可。插入是这样进行的：若二叉排序树为空，则待插入结点*s 作为根结点插入到空树中；当二叉排序树非空，将待插结点的关键字 s->key 与树根的关键字 t->key 比较，若 s->key=t->key，则说明树中已有此结点，无须插入；若 s->key<t->key，则将待插结点*s 插入到根的左子树中，否则将*s 插入到根的右子树中。而子树中的插入过程又和在树中的插入过程相同，如此进行下去，直到把结点*s 作为一个新的树叶插入到二叉排序树中，或者直到发现树中已有结点*s 为止。

实验学时:2

关键代码:

```
typedef int keyType;
typedef int ElemType;
typedef struct{
   ElemType *elem;
   int length;
}SSTable;
int Create(SSTable &ST,int n){
   int i;
   int x=0;
   ST.elem=(ElemType*)malloc((n+1)*sizeof(ElemType));
   printf("请输入 9 个数:");
   for(i=1;i<=n;i++){
         scanf("%d",&x);
      ST.elem[i]=x;
   }
   ST.length=n;
   return 1;
}
int Search_Seq(SSTable ST,keyType key){
     int i;
     ST.elem[0]=key;
     for(i=ST.length;ST.elem[i]!=key;--i);
     return i;
}
```

```
int Search_Bin(SSTable ST,keyType key){
    ElemType low,high,mid;
    low=1;high=ST.length;
    while(low<=high){
        mid=(low+high)/2;
        if (key==ST.elem[mid]) return mid;
        else if (key<ST.elem[mid]) high=mid-1;
        else low=mid+1;
    }
    return 0;
}
```

实验指导书 8 排序

实验目的:

1、掌握常用的排序方法，并掌握用高级语言实现排序算法的方法。

2、深刻理解排序的定义和各种排序方法的特点，并能加以灵活应用。

3、了解各种方法的排序过程及其依据的原则，并掌握各种排序方法的时间复杂度的分析方法。

实验学时: 2

实验内容:

在学生成绩管理中，经常会遇到求平均成绩，统计不及格学生成绩，统计优秀学生人数，以及按成绩对学生进行排名等。现假设有某个班级的若干名学生，每个学生都考试完成了 4 门课程，试对所有学生的成绩完成以下工作：

（1）求每门课程的平均成绩。

（2）输出所有有不及格课程的学生的学号、姓名、全部课程的成绩、平均成绩。

（3）输出所有平均分在 90 分以上（含 90 分）的学生学号、姓名。

（4）对 4 门课程中的任何一门，可随意抽取 1 门按学生成绩采用任意一种排序方法进行排序

关键代码:

```
void QuickSort ( SqList &L, int low, int high ){
//在序列 low~high  中递归地进行快速排序
      if ( low < high) {
           int pivotloc= Partition (L, low, high);      //划分
           QuickSort ( L, low, pivotloc-1); //对左序列同样处理
           QuickSort ( L, pivotloc+1, high); //对右序列同样处理
      }
}
int Partition ( SqList &L, int low, int high ) {
L.r[0]=L.r[low];//子表的第一个记录作基准对象
pivotkey = L.r[low].key; //基准对象关键字
While(low<high){
       While(low<high && L.r[high].key >= pivotkey) --high;
         L.r[low] = L.r[high]; //小于基准对象的移到区间的左侧
         While(low<high&& L.r[low].key <= pivotkey) ++low;
           L.r[high] = L.r[low] ; //大于基准对象的移到区间的右侧
}
L.r[low] = L.r[0];
return low;
}
```