安徽工程大学

《数据结构与算法》
实验指导书

专　　业：电子信息工程

电气工程学院（集成电路学院）　汤红霞

2025 年 8 月

实 验 说 明

《数据结构与算法》是电子信息工程专业的一门专业基础课。本课程较系统地介绍了软件设计中常用的数据结构以及相应的存储结构和实现算法，介绍了常用的多种查找和排序技术，并做了性能分析和比较。本课程的学习将为后续课程的学习以及软件设计水平的提高打下良好的基础。

该课程内容丰富，学习量大，所用到的技术多，学习和消化有一定的难度。根据课程本身的技术特性，设置课程实验环节十分重要。通过实验实践内容的训练，突出创新性思维训练的特征，可以达到以下几方面目标：

（1）加深对课堂讲授内容的理解

实验是对学生的一种全面综合训练，是与课堂听讲、自学和练习相辅相成的必不可少的一个教学环节。通常，实验题中的问题比平时的习题复杂，也更接近实际。实验着眼于原理与应用的结合点，使学生学会如何把书上学到的知识用于解决实际问题；另一方面，能使书上的知识变"活"，起到深化理解和灵活掌握教学内容的目的。

不少学生在解答习题尤其是算法设计题时，觉得无从下手，做起来特别费劲。实验中的内容和教科书的内容是密切相关的，解决题目要求所需的各种技术大多可从教科书中找到，只不过其出现的形式呈多样化，因此需要仔细体会，在反复实践的过程中才能掌握。

（2）培养学生软件设计的综合能力

平时的练习较偏重于如何编写功能单一的"小"算法，而实验题是软件设计的综合训练，包括问题分析、总体结构设计、用户界面设计、程序设计基本技能和技巧，多人合作，以至一整套软件工作规范的训练和科学作风的培养。

实验着眼于原理与应用的结合点，使学生学会如何把书本上和课堂上学到的知识用于解决实际问题，通过实验使学生不仅能够深化理解教学内容，进一步提高灵活运用数据结构、算法和程序设计技术的能力，有利于培养和提高学生的实践创新能力。

目　录

实验 1 线性表.....1
一、实验目的.....1
二、实验内容.....1
三、实验报告.....13

实验 2 栈和队列.....15
一、实验目的.....15
二、实验内容.....15
三、实验报告.....21

实验 3 二叉树.....23
一、实验目的.....23
二、实验内容.....23
三、实验报告.....34

实验 4 图.....36
一、　实验目的.....36
二、实验内容.....36
三、实验报告.....44

附录 实验环境配置.....46

实验 1 线性表

一、实验目的

(1) 掌握线性表的顺序存储
(2) 掌握线性表的链式存储
(3) 掌握基本算法（建表、插入、删除）的实现
(4) 理解各种算法的时间复杂度

二、实验内容

（一）线性表的顺序存储：掌握线性表的顺序存储结构及其基本操作、合并、逆置等算法

设顺序表的存储结构定义如下：（同学们可扩展考虑其他形式的存储结构定义）

```
#define LIST_INIT_SIZE 100        // 线性表存储空间的初始分配量
#define LISTINCREMENT 10          // 线性表存储空间的分配增量
typedef struct
{
   int *elem;                     // 存储空间基址
   int length;                    // 当前长度
   int listsize;                  // 当前分配的存储容量（以 sizeof(int)为单位）
}SqList;
```

[题目 1：编写算法，创建初始化容量为 LIST_INIT_SIZE 的顺序表 L，并实现插入、删除、遍历操作。]

测试样例格式说明：

根据菜单操作：

1、输入 1，表示要实现插入操作，紧跟着要输入插入的位置和元素，用空格分开
2、输入 2，表示要实现删除操作，紧跟着要输入删除的位置
3、输入 3，表示要输出顺序表的所有元素
4、输入 0，表示程序结束

请补全代码：

```
#include<stdio.h>
#include<malloc.h>
#define OK 1
#define ERROR 0
#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10
#define ElemType int

typedef struct
{
 int *elem;
 int length;
 int listsize;
}SqList;
```

```
int InitList_Sq(SqList &L)
{// 构造一个空的线性表 L，该线性表预定义大小为 LIST_INIT_SIZE
// 请补全代码
}

int Load_Sq(SqList &L)
{// 输出顺序表中的所有元素
 int i;
 if(_____) printf("The List is empty!");   // 请填空
 else
 {
      printf("The List is: ");
      for(_____) printf("%d ",_____);   // 请填空
 }
 printf("\n");
 return OK;
}

int ListInsert_Sq(SqList &L,int i,int e)
{// 在顺序线性表 L 中第 i 个位置之前插入新的元素 e
// i 的合法值为 1≤i≤L.length +1
// 请补全代码
}

int ListDelete_Sq(SqList &L,int i, int &e)
{
// 算法 2.5,在顺序线性表 L 中删除第 i 个位置的元素，并用 e 返回其值
// i 的合法值为 1≤i≤L.length
// 请补全代码
}

int main()
{
 SqList T;
 int a, i;
 ElemType e, x;
 if(_____)      //请填空
 {
      printf("A Sequence List Has Created.\n");
 }
 while(1)
 {
      printf("1:Insert element\n2:Delete element\n3:Load all elements\n0:Exit\nPlease choose:\n");
      scanf("%d",&a);
      switch(a)
      {
```

```
            case 1: scanf("%d%d",&i,&x);
                    if(_____) printf("Insert Error!\n"); // 请填空
                    else printf("The Element %d is Successfully Inserted!\n", x);
                    break;
            case 2: scanf("%d",&i);
                    if(_____) printf("Delete Error!\n"); // 请填空
                    else printf("The Element %d is Successfully Deleted!\n", e);
                    break;
            case 3: Load_Sq(T);
                    break;
            case 0: return 1;
        }
    }
}
```

测试样例：

```
A Sequence List Has Created.
1:Insert element
2:Delete element
3:Load all elements
0:Exit
Please choose:
1
1 10
The Element 10 is Successfully Inserted!
1:Insert element
2:Delete element
3:Load all elements
0:Exit
Please choose:
1
2 20
The Element 20 is Successfully Inserted!
1:Insert element
2:Delete element
3:Load all elements
0:Exit
Please choose:
1
3 33
The Element 33 is Successfully Inserted!
1:Insert element
2:Delete element
3:Load all elements
0:Exit
Please choose:
3
The List is: 10 20 33
1:Insert element
2:Delete element
3:Load all elements
0:Exit
Please choose:
```

[题目 2：设有一顺序表 A=（$a_0,a_1,...,a_i,...a_{n-1}$），其逆顺序表定义为 A'=($a_{n-1},...,a_i,...,a_1,a_0$)。设计一个算法，将顺序表逆置，要求顺序表仍占用原顺序表的空间。请同学们独立完成，所需子函数参考前面题目 1 完成的内容。]

备注：

(1) 程序要添加适当的注释，程序的书写应采用缩进格式。

(2) 程序要具在一定的健壮性，即当输入数据非法时，程序也能适当地做出反应，如插入删除时指定的位置不对等等。

(3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。

测试样例格式说明：

[键盘输入]

第一行：输入顺序表的元素个数

第二行：输入顺序表的各元素，用空格分开

[正确输出]

第一行：逆置前的顺序表元素列表

第二行：逆置后的顺序表元素列表

测试样例：	
[第一组自测数据] [键盘输入] 10↙ 1 2 3 4 5 6 7 8 9 10↙ [正确输出] The List is:1 2 3 4 5 6 7 8 9 10 The turned List is:10 9 8 7 6 5 4 3 2 1	[第二组自测数据] [键盘输入] 8↙ 32 97 54 65 35 84 61 75↙ [正确输出] The List is:32 97 54 65 35 84 61 75 The turned List is:75 61 84 35 65 54 97 32

（二）线性表的链式存储：掌握线性表的链式存储结构及其基本操作、合并、逆置等算法。本实验以单链表为例，在完成题目的过程中，同学们可扩展考虑双链表及循环链表等结构的操作。

设单链表的存储结构定义如下：（同学们可扩展考虑其他形式的存储结构定义）

```
#include<stdio.h>
typedef struct LNode
{
  int data;                    // 存储在结点中的数据
  struct LNode *next;          // 指向下一结点的指针
}LNode,*LinkList;
```

[题目 3：编写算法，创建一个含有 n 个元素的带头结点的单链表 L 并实现插入、删除、遍历操作。]

测试样例格式说明：

根据菜单操作：

1、输入 1，表示要实现插入操作，紧跟着要输入插入的位置和元素，用空格分开

2、输入 2，表示要实现删除操作，紧跟着要输入删除的位置

3、输入 3，表示要输出顺序表的所有元素

4、输入 0，表示程序结束

请补全代码：

```
#include<stdio.h>
#include<malloc.h>
#define ERROR 0
#define OK 1
#define ElemType int

typedef struct LNode
{
  int data;
  struct LNode *next;
}LNode,*LinkList;

int CreateLink_L(LinkList &L,int n){
// 创建含有 n 个元素的单链表
    LinkList p,q;
    int i;
    ElemType e;
    L = (LinkList)malloc(sizeof(LNode));
    L->next = NULL;                    // 先建立一个带头结点的单链表
    q = (LinkList)malloc(sizeof(LNode));
    q = L;
    for (i=0; i<n; i++) {
       scanf("%d", &e);
        p = (LinkList)malloc(sizeof(LNode));   // 生成新结点
       // 请补全代码
    }
    return OK;
}

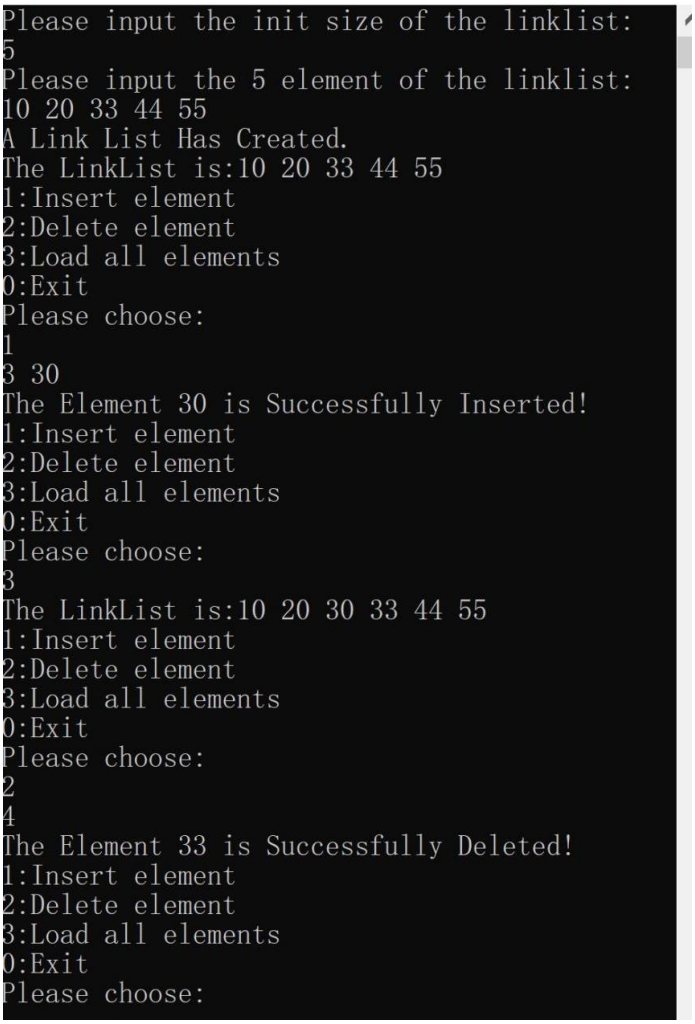
int LoadLink_L(LinkList &L){// 单链表遍历
  LinkList p = L->next;
  if(_____)printf("The List is empty!"); // 请填空
  else
  {
      printf("The LinkList is:");
      while(_____)       // 请填空
      {
          printf("%d ",p->data);
          _____      // 请填空
      }
  }
  printf("\n");
  return OK;
}
```

```
int LinkInsert_L(LinkList &L,int i,ElemType e){
// 在带头结点的单链线性表 L 中第 i 个位置之前插入元素 e
// 请补全代码
}

int LinkDelete_L(LinkList &L,int i, ElemType &e){
// 在带头结点的单链线性表 L 中，删除第 i 个元素，并用 e 返回其值
// 请补全代码
}

int main()
{
 LinkList T;
 int a,n,i;
 ElemType x, e;
 printf("Please input the init size of the linklist:\n");
 scanf("%d",&n);
 printf("Please input the %d element of the linklist:\n", n);
 if(_____)        //请填空
 {
    printf("A Link List Has Created.\n");
    LoadLink_L(T);
 }
 while(1)
   {
        printf("1:Insert element\n2:Delete element\n3:Load all elements\n0:Exit\nPlease choose:\n");
        scanf("%d",&a);
        switch(a)
        {
             case 1: scanf("%d%d",&i,&x);
                     if(_____) printf("Insert Error!\n"); // 请填空
                     else printf("The Element %d is Successfully Inserted!\n", x);
                     break;
             case 2: scanf("%d",&i);
                     if(_____) printf("Delete Error!\n"); // 请填空
                     else printf("The Element %d is Successfully Deleted!\n", e);
                     break;
             case 3: LoadLink_L(T);
                     break;
             case 0: return 1;
        }
   }
}
```

测试样例：



[题目 4 设有一线性表 A=（a_0,a_1,..., a_i,...a_{n-1})，其逆线性表定义为 A'=(a_{n-1},..., a_i,...,a_1, a_0)，设计一个算法，将链式线性表逆置，要求线性表仍占用原线性表的空间。请同学们独立完成，所需子函数参考题目 3 完成的内容。]

测试样例格式说明：

[键盘输入]

第一行：输入 n，表示单链表的元素个数

第二行：输入单链表的各元素，用空格分开

[正确输出]

第一行：输出单链表逆置前的元素列表

第二行：输出单链表逆置后的元素列表

测试样例：

[键盘输入]

8↙

32 97 54 65 35 84 61 75↙

[正确输出]

The List is:32 97 54 65 35 84 61 75

The turned List is:75 62 84 35 65 54 97 32

（三）线性表的合并、有序表的合并（选做）

[题目 5：编写算法，将两个非递减有序顺序表 A 和 B 合并成一个新的非递减有序顺序表 C。请同学们独立完成，所需子函数参考前面题目 1 完成的内容。]

测试样例格式说明：
[键盘输入]
第一行：顺序表 A 的元素个数
第二行：顺序表 A 的各元素（非递减），用空格分开
第三行：顺序表 B 的元素个数
第四行：顺序表 B 的各元素（非递减），用空格分开
[正确输出]
第一行：顺序表 A 的元素列表
第二行：顺序表 B 的元素列表
第三行：合并后顺序表 C 的元素列表

测试样例：	
[第一组自测数据] [键盘输入] 5↙ 1 3 5 7 9↙ 5↙ 2 4 6 8 10↙ [正确输出] List A:1 3 5 7 9 List B:2 4 6 8 10 List C:1 2 3 4 5 6 7 8 9 10	[第二组自测数据] [键盘输入] 6↙ 12 24 45 62 84 96↙ 4↙ 15 31 75 86↙ [正确输出] List A:12 24 45 62 84 96 List B:15 31 75 86 List C:12 15 24 31 45 62 75 84 86 96

请补全代码：

```
#include<stdio.h>
#include<malloc.h>
#define OK 1
#define ERROR 0
#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10
#define ElemType int

typedef struct
{
      int *elem,length,listsize;
 }SqList;

 int InitList_Sq(SqList &L)
 { // 请补全代码
```

```
        ____________________
        ....................................
}

int Load_Sq(SqList &L)
{// 请补全代码
        ____________________
        ....................................
}

int ListLength(SqList L)
{// 请补全代码
        ____________________
        ....................................
}

int GetElem(SqList L,int i,ElemType &e)
{// 请补全代码
        ____________________
        ....................................
}

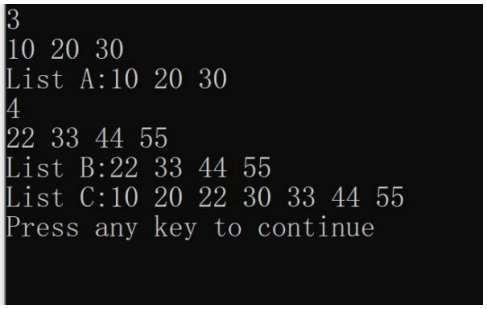
int ListInsert_Sq(SqList &L,int i,int e)
{// 请补全代码
        ____________________
        ....................................
}

void MergeList(SqList La,SqList Lb,SqList &Lc)
{
        int i,j,k,La_len,Lb_len,ai,bj;
        i=j=1;
        k=0;
        InitList_Sq(Lc);
        La_len=ListLength(La);
        Lb_len=ListLength(Lb);
        while((i<=La_len)&&(j<=Lb_len))
        {
                GetElem(La,i,ai);
                GetElem(Lb,j,bj);
                if(ai<=bj)
                {// 请补全代码
                        ____________________
                        ....................................
                }
                else
```

```
            {// 请补全代码
                ____________________
                ....................................
            }
        }
        while(i<=La_len)
        {// 请补全代码
            ____________________
            ....................................
        }
        while(j<=Lb_len)
        {// 请补全代码
            ____________________
            ....................................
        }
        Load_Sq(Lc);
}

int main()
{
        int an,bn,i,e;
        SqList La,Lb,Lc;
        InitList_Sq(La);
        scanf("%d",&an);
        for(i=1;i<=an;i++)
        {
            scanf("%d",&e);
            ListInsert_Sq(La,i,e);
        }
        printf("List A:");
        Load_Sq(La);
        InitList_Sq(Lb);
        scanf("%d",&bn);
        for(i=1;i<=an;i++)
        {
            scanf("%d",&e);
            ListInsert_Sq(Lb,i,e);
        }
        printf("List B:");
        Load_Sq(Lb);
        printf("List C:");
        MergeList(La,Lb,Lc);
        return 0;
}
```

测试样例：



[题目 6：设计一个算法将两个非递减有序链表 A 和 B 合并成一个新的非递减有序链表 C。请同学们独立完成，所需子函数参考题目 3 完成的内容。]

测试样例格式说明：

[键盘输入]

第一行：单链表 A 的元素个数

第二行：单链表 A 的各元素（非递减），用空格分开

第三行：单链表 B 的元素个数

第四行：单链表 B 的各元素（非递减），用空格分开

[正确输出]

第一行：单链表 A 的元素列表

第二行：单链表 B 的元素列表

第三行：合并后单链表 C 的元素列表

测试样例：

[键盘输入]

6↙

12 24 45 62 84 96↙

4↙

15 31 75 86↙

[正确输出]

List A:12 24 45 62 84 96

List B:15 31 75 86

List C:12 15 24 31 45 62 75 84 86 96

请补全代码：

```
#include<stdio.h>
#include<malloc.h>
#define ERROR 0
#define OK 1
#define ElemType int

typedef struct LNode
{
    int data;
    struct LNode *next;
}LNode,*LinkList;

int CreateLink_L(LinkList &L,int n)
```

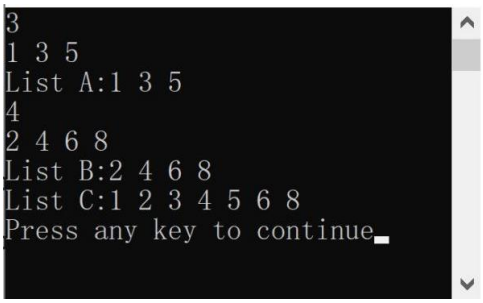
```
{
    LinkList p,q;
    int i;
    ElemType e;
    // 请补全代码
    _____
    .....
    q=L;
    for(i=0;i<n;i++)
    {
        scanf("%d",&e);
        p=(LinkList)malloc(sizeof(LNode));
    // 请补全代码
    _____
    .....
    }
    return OK;
}

int LoadLink_L(LinkList &L)
{
    LinkList p=L->next;
    if(!p)
        printf("The List is empty!");
    else
    {
    // 请补全代码
    _____
    .....
    }
    printf("\n");
    return OK;
}

void MergeList_L(LinkList &La,LinkList &Lb,LinkList &Lc)
{
    LinkList pa,pb,pc;
    pa=La->next;
    pb=Lb->next;
    Lc=pc=La;
    while(pa&&pb)
    {
        if(pa->data<=pb->data)
        {
        // 请补全代码
        _____
        .....
        }
        else
        {
        // 请补全代码
        _____
        .....
        }
    }
    pc->next=pa?pa:pb;
    free(Lb);
}
```

```
int main()
{
    LinkList La,Lb,Lc;
    int n;
    scanf("%d",&n);
    CreateLink_L(La,n);
    printf("List A:");
    LoadLink_L(La);
    scanf("%d",&n);
    CreateLink_L(Lb,n);
    printf("List B:");
    LoadLink_L(Lb);
    MergeList_L(La,Lb,Lc);
    printf("List C:");
    LoadLink_L(Lc);
    return 0;
}
```

测试样例：



三、实验报告

（1）本实验各题目，要求自测通过样例。
（2）实验报告针对以上实验内容。
（3）实验总结主要是对实验内容的掌握及理解程序作一个归纳性的叙述。
（4）对于思考题进行分析和思考，并做出相应的结论。

实验报告格式

实验 1：线性表

班级_____ 学号_____ 姓名_____ 实验时间_____

一、实验目的

(1) 掌握线性表的顺序存储
(2) 掌握线性表的链式存储
(3) 掌握基本算法（建表、插入、删除）的实现
(4) 理解各种算法的时间复杂度

二、实验内容

题目 1：编写算法，创建初始化容量为 LIST_INIT_SIZE 的顺序表 T，并实现插入、删除、遍历操作。
题目 2：设有一顺序表 A=（a_0,a_1,..., a_i,...a_{n-1})，其逆顺序表定义为 A'=(a_{n-1},..., a_i,...,a_1, a_0)。设计一个算法，将顺序表逆置，要求顺序表仍占用原顺序表的空间。请同学们独立完成，所需子函数参考前面题目 1 完成的内容。

题目 3：编写算法，创建一个含有 n 个元素的带头结点的单链表 L 并实现插入、删除、遍历操作。
题目 4 设有一线性表 A=（a_0,a_1,..., a_i,...a_{n-1})，其逆线性表定义为 A'=(a_{n-1},..., a_i,...,a_1, a_0)，设计一个算法，将链式线性表逆置，要求线性表仍占用原线性表的空间。请同学们独立完成，所需子函数参考题目 3 完成的内容。

三、实验步骤和结果（各题调试正确后，将运行截图提交到学习通对应答题项下，并在实验报告纸上撰写以下函数的代码）

1、题目 1：（1）顺序表 T 初始化；（2）插入元素；（3）删除元素。
2、题目 3：（1）单链表 L 初始化；（2）插入元素；（3）删除元素。

四、实验总结（写出实验过程中遇到的问题、解决办法和心得体会）

五、思考题（在实验报告纸上回答以下思考题）

1. 确定线性表的存储结构时，什么情况下应选用顺序表？什么情况下应选用链表？
2. 在顺序表中插入和删除一个结点需平均移动多少个结点？具体的移动次数取决于哪两个因素？
3. 求所实现的单链表逆置算法的时间复杂度。

成绩_____ 教师签名_____ 日期_____