# 《计算机实践》—数据结构实验报告要求及模板

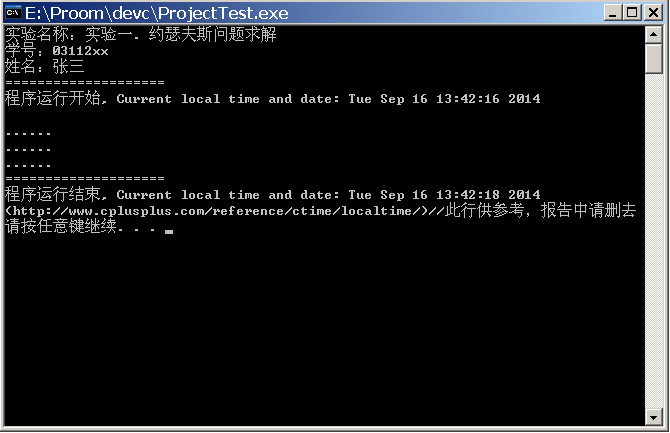
南京航空航天大学

自动化学院

2022年10月

## 第1部分. 实验报告要求

* 简述每一部分的对象、目的和要求；
* 画出算法的程序流程图；
* 说明程序的数据输入要求；
* 调试结果：以截屏图像的方式给出，包括程序运行开始和结束时的系统时间、姓名、学号、实验名称等信息；



* 实验遇到的问题和解决办法：程序调试日志；
* **实验的收获和感想：**个人感想，方法的优缺点、对本实验的要求和建议；忌空、大话。
* 附源程序清单（双栏排版）；对源程序的功能块做适当注释。
* 其他未尽事项按写报告的一般要求进行。
* 提交内容包括①实验报告的WORD版本，②源程序（包括可以直接运行演示的可执行文件）。每个同学提交的内容放在一个以“DS实验\*\_学号\_姓名”(如DS实验一032110101某某某)方式命名的压缩文件中提交到学习通。

## 实验题求解的构思、程序实现可以适当参考文献资料，但应避免直接照搬。每人独立完成实验，如发现实验程序或报告的内容有雷同，雷同双方（或多方）报告视为无效。

## 实验报告是评分的重要依据，务必做到内容清晰易读，格式符合要求。

* 实验报告封面及参考模板如下。

## 

# 

# 《计算机实践》

# 实验报告I—数据结构

班号： 学号：

姓名： Email：

签名：

南京航空航天大学

## 实验\*\*\*：顺序表与链表

### 一、实验目的

　　熟练掌握动态链表结构及有关算法的设计。

掌握用链表表示特定形式的数据的方法，并能编写出有关运算的算法。

### 二、实验内容

链式存储结构及应用

1．键盘输入一组元素，尾部插入法建立一个单向链表。

2．在单向链表中删除一个结点。

3．在链表中插入一个元素。

### 三、算法分析与程序设计

1．输入的形式和输入值的范围

建立链表时输入的都是整数,输入0代表输入的结束，插入元素时需要输入插入的位置和元素的值，删除元素时输入删除元素的值。

2．输出形式

所有操作在出现错误时都会有提示，并且在任意一个操作结束后都会输出操作后的链表。

3．算法描述

给出算法描述的程序流程图

4．程序设计

给出实现程序功能的函数及说明，示例如下：

(1) 本程序包含的函数

主函数 main()

显示操作菜单函数 mainmenu()

显示单链表内容函数 print()

尾部插入建立链表函数 creatb()

插入元素函数 insert()

其中main函数负责操作其他的函数。

(2)各函数的功能设计为：

struct student \*creatb()：建立一个链表A(尾部插入建立的链表)，并且返回这个链表的头指针.

struct student \*insert(int a,int b,struct student \*head)：接受2个int类型的数据a和b(其中a为插入的位置，b为插入的数据)，head为需要操作的链表的头指针.

struct student \*creath()：建立链表B(头部插入建立的链表)，返回头指针.

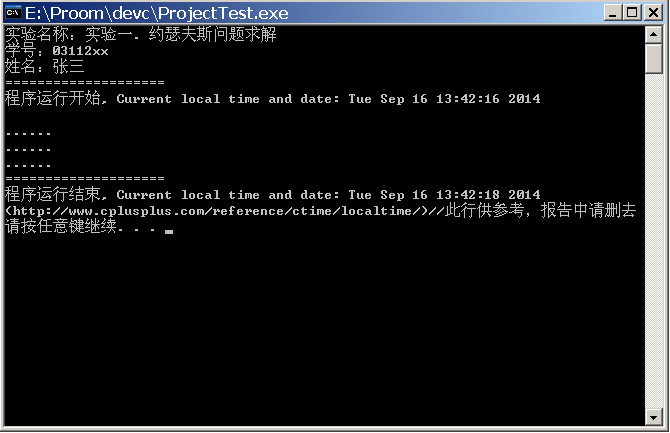
void compare(struct student \*head1,struct student \*head2,int flag)：比较两链表是否相等.其中flag为判断用户是否进行了两链表的排序操作，当flag为1时，表示进行了排序操作.为0时表示没用进行排序操作(这样可以节省运算的时间).

mainMenu()：打印出菜单

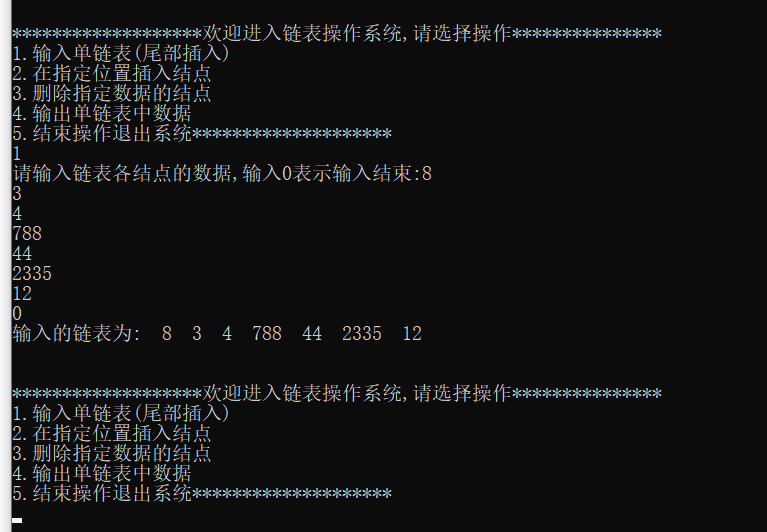
### 四、调试分析

#### 1、调试过程及说明

* 以截屏图像的方式给出，包括程序运行开始和结束时的系统时间、姓名、学号、实验名称等信息。如：



* 中间的调试结果……

****

#### 2、调试分析---只是示例

（1）刚开始时曾忽略了一些变量参数的标识“&”，使调试程序时费时不少。今后应重视确定参数的变量和赋值属性的区分和标识。

（2）本程序的模块划分比较合理，且尽可能将指针的操作封装在结点和链表的两个模块中，致使集合模块的调试比较顺利。反之，如此划分的模块并非完全合理，因为在实现集合操作的编码中仍然需要判别指针是否为空。按理，两个链表的并、交和差的操作也应封装在链表的模块中，而在集合的模块中，只要进行相应的应用即可。

### 五、使用说明

给出程序使用过程中的注意事项，如输入要求等。

**六、测试结果**

内容：给出测试用例测试程序。写出测试的情况，使用的输入以及得到的结果。

要求：测试的情况越全面越好。

示例：

**1 、建立单链表：**

选择 1 ，分别输入15 14 13 12 11。得到单链表（ 15，14，13，12，11 ）

**2 、插入：**

（1）测试插入位置不合法

选择 2 输入（ -1，2 ），显示输入错误

（2）测试插入位置在头部

选择 2 输入（ 1，100 ），得到单链表（100，15，14，13，12，11 ）

（3）测试插入位置在尾部

选择 2 输入（ 7，2 ），得到单链表（100，15，14，13，12，11, 2）

（4）测试插入位置在中间

选择 2 输入（ 3，8 ），得到单链表（100，15，8, 14，13，12，11, 2）

**3、删除：**

（1）测试删除位置不合法

选择 3 ，输入 10 。返回输入错误

（2）测试删除位置在头部

选择 3 ，输入 1 。返回 e=100 ，得到单链表（15，8, 14，13，12，11, 2 ）

（3）测试删除位置在中间

选择 3 ，输入 4 。返回 e=13 ，得到单链表（15，8, 14，12，11, 2 ）

（4）测试删除位置在尾部

选择 3 ，输入 6 。返回 e=2 ，得到单链表（15，8, 14，12，11 ）

继续删除头部的节点,直到链表只剩下一个元素11。

（5）测试在空链表中删除元素

选择 3 ，输入 1 。提示链表为空，不能删除

**七、实验的收获和感想**

个人感想，方法的优缺点、对本实验的建议；忌空、大话。

**八、附录：源代码（双栏排版，加必要的注释）**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct student

{

int num;

struct student \*next;

};

void print(struct student \*head)

{

struct student \*p;

p=head;

char s=' ';

if(head==NULL)

{

printf("该链表为空");

}

if(head!=NULL)

{

do

{

printf("%c%c%d",s,s,p->num);

p=p->next;

}

while(p!=NULL);

printf("\n");

}

}

struct student \*creatb()

{

struct student \*head;

struct student \*p1,\*p2;

int n=0;

p1=p2=(struct student\*)malloc(sizeof(struct student));

scanf("%d",&p1->num);

head=NULL;

while(p1->num!=0)

{

n=n+1;

if(n==1)

head=p1;

else

p2->next=p1;

p2=p1;

p1=(struct student\*)malloc(sizeof(struct student));

scanf("%d",&p1->num);

}

p2->next=NULL;

return head;

}

struct student \*insert(int a,int b,struct student \*head)

{

int position;

position=a;

struct student \*p,\*q,\*p1;

q=(struct student\*)malloc(sizeof(struct student));

q->num=b;

p=head;

p1=head;

if(position==1)

{

q->next=p;

p=q;

}

if(position==2)

{

if(p->next!=NULL)

{

q->next=p1->next;

p1->next=q;

}

else

{

p->next=q;

q->next=NULL;

}

}

if(position>=3)

{

for(int i=1;i<(position-1);i++)

p1=p1->next;

if(p1->next!=NULL)

{

q->next=p1->next;

p1->next=q;

}

else

{

p1->next=q;

q->next=NULL;

}

}

return p;

}