数据结构实验报告——实验 X

学号：__20201050331___姓名：__黄珀芝___得分：_____

一、实验目的

1、复习线性表的逻辑结构、存储结构及基本操作；
2、掌握顺序表和（带头结点）单链表；
3、了解有序表

二、实验内容

1、(必做题)假设有序表中数据元素类型是整型,请采用顺序表或(带头结点)单链表实现:
(1) Orderlnsert(&, e. int (*compare)(a，b)
//根据有序判定函数 compare,在有序表 L 的适当位置插入元素 c
(2) OrderInput(&, int (*compare)(a, b) 根据有序判定函数 compare, 并利用有序插入函数 OrderInsert,构造有序表 L;
(3)OrderMerge(&La, &Lb, &le, int (compare)())//根据有序判定函数 compare,将两个有序表 La 和 Lb 归并为一个有序表 Lc。

2、(1)升幂多项式的构造,升幂多项式是指多项式的各项按指数升序有序,约定系数不能等于 0,指数不能小于 0;
(2)两个升幂多项式的相加；

3、问题描述:将数字 1,2,,n 环形排列;按顺时针方向从 1 开始计数,计满 k 时输出该位置上的数字,同时从环中删除该数字;然后从下一位置开始重新开始计数,直到环中所有数均被输出为止。请使用顺序表或链表实现:对输入的任意 n 和 k,输出相应的出列序。

三、数据结构及算法描述

实验一：
数据结构： 构造两个有序表 La 和 Lb。
操作：1、键盘输入链表长度，确定链表能存储的数据空间。2、键盘输入数据通过 for 循环分别存储在链表 La 和 Lb 里。3、通过 orderOutput 函数输出 La 和 Lb。4、选择是否需要在 La 或者 Lb 中插入元素，选择要插入的长度让链表扩张相应长度。5、for 循环输入要插入的数组。6、合并 La 和 Lb。

算法：（1）定义带头结点的单链表；
（2）从键盘输入数据，排序存入顺序表中；
（3）从键盘中输入需要操作对应的序号；
（4）调用函数和打印函数显示结果；
（5）最后用归并函数归并 La 和 Lb 输出 Lc。

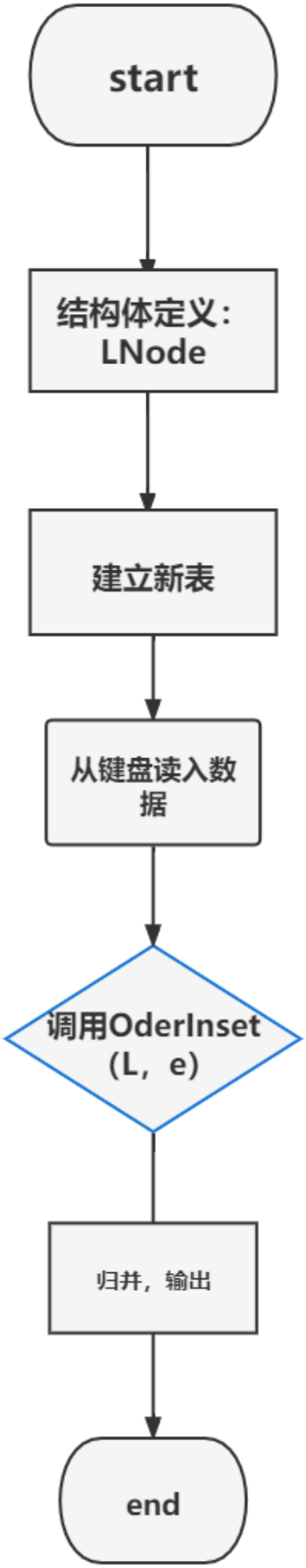
实验二：
数据结构：创建单链表两个：or 存取序数，in 存取指数。链表 x 存储 or，链表 z 存储 in。

操作：1、从头结点开始存储。2、用动态数组分配存储空间。3、比较两个多项式里的 in 是否有相同的，若有相同则链表所指向的对应的 or 相加。4、不相同的则直接归并一起输出。

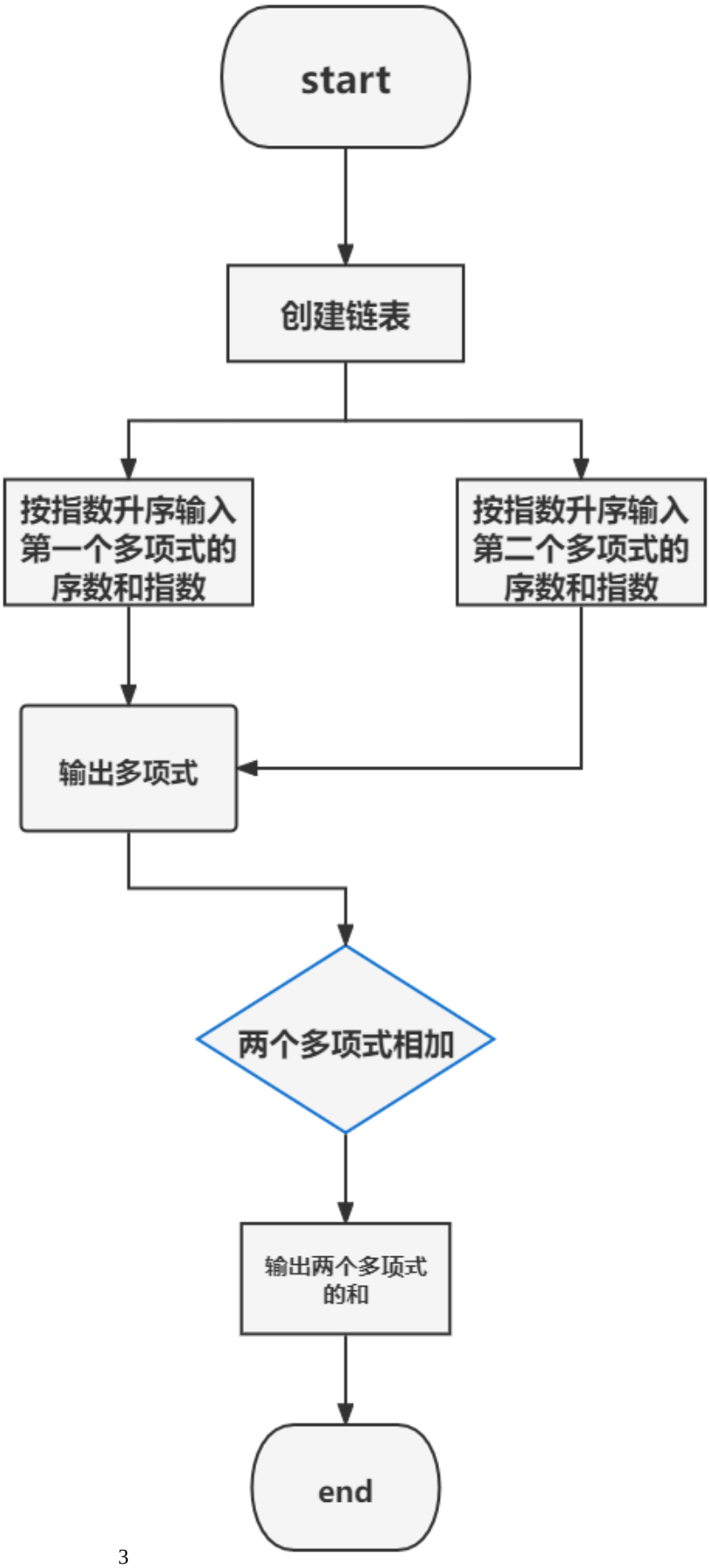
算法：（1）创建链表，按指数升序输入多项式的序数和指数；
（2）输出多项式；
（3）按指数升序输入第二个多项式的序数和指数；
（4）两个多项式相加；
（5）输出第二个多项式和两个多项式的和。

（流程图在下一张，此张放不下）

四、实验设计：实验一流程图：



实验二流程图：



五、程序代码

实验一：

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

typedef struct LNode//定义结构体，带头结点的单链表
{
    int data;//数据域
    struct LNode *next;//指针域
}Linklist;

int OrderInsert(Linklist *L,int e)//在有序表 L 的适当位置插入元素 e
{
    Linklist *p,*s,*q;
    s=(Linklist *)malloc(sizeof(Linklist));
    s->data = e;
    p = L->next;
    q = L;
    while(p!=NULL)
    {
        if(e<=p->data)
        {
            q->next = s;
            s->next = p;
            break;
        }
        else
        {
            q = p;
            p = p->next;
        }
    }
    if(p==NULL)
    {
        q->next = s;
        s->next = NULL;
    }
    return 1;
}

Linklist* OrderMerge(Linklist*La,Linklist*Lb,Linklist*Lc)//将两个有序表 La 和 Lb 合并为一个有序表 Lc
{
    Linklist*Pa,*Pb,*Pc;
```

```
    Pa = La->next;
    Pb = Lb->next;
    Pc = Lc = La;
    while(Pa && Pb)
    {
        if(Pa->data<=Pb->data)
        {
            Pc->next=Pa;
            Pc=Pa;
            Pa=Pa->next;
        }
        else
        {
            Pc->next=Pb;
            Pc=Pb;
            Pb=Pb->next;
        }
    }
    Pc->next = Pa?Pa:Pb;
    return Lc;
}


void OrderOutput(Linklist *p)//输出链表
{
    Linklist *q;
    q=p->next;
    while(q)
    {
        printf("%d\t",q->data);
        q=q->next;
    }
     printf("\n");
}


void Lfree(Linklist *head)//释放结点
{
    while(head!=NULL)
    {
        free(head);
        head=head->next;
    }
}
```

```
int main()
{
    int e,i,n,k,length_a,length_b;

    Linklist *La,*Lb,*Lc;
    La=(Linklist*)malloc(sizeof(Linklist));
    Lb=(Linklist*)malloc(sizeof(Linklist));
    La->next=NULL;
    Lb->next=NULL;
    printf("请输入链表 La 的长度：\n");
    scanf("%d,&length_a");
    printf("输入%d 个数(数据间用空格隔开，输入最后一个数字后按回车键。)\n",length_a);
    for(i=1;i<=length_a;i++)
    {
        scanf("%d",&e);
        OrderInsert(La,e);
    }
    printf("请输入链表 b 的长度：\n");
    scanf("%d",&length_b);
    printf("输入%d 个数(数据间用空格隔开，输入最后一个数字后按回车键。)\n",length_b);
    for(i=1;i<=length_b;i++)
        {scanf("%d",&e);
        OrderInsert(Lb,e);
    }
    printf("是否向链表中插入数？\n\n");
    printf("向链表 La 中插入则按 1 并回车进入输入。\n 向链表 Lb 中插入则按 2 回车进入
输入。\n 若不插入请输入 0。\n\n");
    printf("现在，请输入你的操作对应的序号：\n");
    scanf("%d",&n);
    while(n!=0)
    {
        {
            if(n==1)
            {
                printf("请输入要插入的个数\n");
                scanf("%d",&k);
                printf("请输入%d 个数（数与数之间用空格隔开，回车进入插入操作）\n",k);
                for(i=0;i<k;i++)
                {
                    scanf("%d",&e);
                    OrderInsert(La,e);
                }
```

```
                break;
            }
            else
            {
                printf("请输入要插入的个数\n");
                scanf("%d",&k);
                printf("请输入%d 个数（数与数之间用空格隔开，回车进入插入操作）",k);
                for(i=0;i<k;i++)
                {
                    scanf("%d",&e);
                    OrderInsert(Lb,e);
                }
                break;
            }
        }
    }
    printf("La:");
    OrderOutput(La);
    printf("Lb:");
    OrderOutput(Lb);
    Lc=OrderMerge(La,Lb,Lc);
    printf("合并后的链表 Lc 为：\n");
    OrderOutput(Lc);
    Lfree(La);
    Lfree(Lb);
    return 0;
}
```

实验二：

```
#include <stdio.h>
#include <malloc.h>

typedef struct node
{
    int x;
    int z;
    struct node *next;

}Node;

Node *Creat()
{
    Node *head,*p,*q;
```

```
    int or,in;
    head = (Node*)malloc(sizeof(Node));
    head->next=NULL;
    q=head;
    printf("请输入该多项式的序数与指数：\n");
    printf("请按照指数升序输入，系数不能等于 0 且指数不能小于 0，序数与指数用空格隔
开，并以 0 0 结束输入\n");
    scanf("%d %d",&or,&in);
    while(or)
    {
        p=(Node *)malloc(sizeof(Node));
        p->x=or;
        p->z=in;
        p->next=q->next;
        q->next=p;
        q=p;
        scanf("%d %d",&or,&in);
    }
    return head;
}
void visit(Node *head)
{
    Node *p=head->next;
    while(p)
    {
        printf("%dX^%d+",p->x,p->z);
        p=p->next;
    }
    printf("NULL\n\n");
}
    Node *Add(Node *head1,Node *head2)
    {
        Node *p,*head,*p1,*p2;
        int sum;
        head = (Node *)malloc(sizeof(Node));
        p=head;
        p1=head1->next;
        p2=head2->next;
        while(p1&&p2)
        {
            if(p1->z==p2->z)
            {
                sum=p1->x+p2->x;
                if(sum)
```

```
                {
                    p1->x=sum;
                    p->next=p1;
                    p=p1;
                }
                p1=p1->next;
                p2=p2->next;
            }
            else
            {
                if(p1->z<p2->z)
                {
                    p->next=p1;
                    p=p1;
                    p1=p1->next;
                }
                else
                {
                    p->next=p2;
                    p=p2;
                    p2=p2->next;
                }
            }
        }
        if(p1)
            p->next=p1;
        else
            p->next=p2;
        return head;
}
int main()
{
        printf("请输入第一个多项式\n");
        Node *head,*p1,*p2;
        p1=Creat();
        printf("多项式为：\n");
        visit(p1);
        printf("请输入第二个多项式：\n");
        p2=Creat();
        printf("多项式为：\n");
        visit(p2);
        head=Add(p1,p2);
        printf("\n 多项式相加后的公式为：\n");
        visit(head);
```

```
    return 0;
}
```

六、测试和结果

实验一：

（1） 根据提示进入测试，向 La 中插入（67 9）：

```
请输入链表La的长度:
4
输入4个数，数据间以空格隔开，输入最后一个数后按回车键。
1 4 5 6
请输入链表Lb的长度:
6
输入6个数，数据间以空格隔开，输入最后一个数后按回车键。
5 8 9 7 14 23
是否向链表中插入数?

向链表La中插入则按1并回车进入输入。
向链表Lb中插入则按2并回车进入输入。
若不插入请输入0。

现在，请输入你的操作对应的序号。
1
请输入要插入的个数
2
请输入2个数，数与数之间用空格隔开，回车进入插入操作。
67 9
La:1    4       5       6       9       67
Lb:5    7       8       9       14      23
合并后的链表Lc为:
1       4       5       5       6       7       8       9       9       14      23      67

Process returned -1073740940 (0xC0000374)   execution time : 203.254 s
Press any key to continue.
```

（2）提示进入测试，向 Lb 中插入（34 67 0 33）：

```
请输入链表La的长度:
5
输入5个数，数据间以空格隔开，输入最后一个数后按回车键。
45 1 6 58 79
请输入链表Lb的长度:
4
输入4个数，数据间以空格隔开，输入最后一个数后按回车键。
12 67 44 99
是否向链表中插入数?

向链表La中插入则按1并回车进入输入。
向链表Lb中插入则按2并回车进入输入。
若不插入请输入0。

现在，请输入你的操作对应的序号。
2
请输入要插入的个数
4
请输入4个数，数与数之间用空格隔开，回车进入插入操作。
34 67 0 33
La:1    6       45      58      79
Lb:0    12      33      34      44      67      67      99
合并后的链表Lc为:
0       1       6       12      33      34      44      45      58      67      67      79      99

Process returned -1073740940 (0xC0000374)   execution time : 36.145 s
Press any key to continue.
```

（3）根据提示进入测试，不插入测试用例为：

```
请输入链表La的长度:
5
输入5个数，数据间以空格隔开，输入最后一个数后按回车键。
78 566 34 9 2
请输入链表Lb的长度:
6
输入6个数，数据间以空格隔开，输入最后一个数后按回车键。
66 56 33 22 11 88
是否向链表中插入数?

向链表La中插入则按1并回车进入输入。
向链表Lb中插入则按2并回车进入输入。
若不插入请输入0。

现在，请输入你的操作对应的序号。
0
La:2    9       34      78      566
Lb:11   22      33      56      66      88
合并后的链表Lc为:
2       9       11      22      33      34      56      66      78      88      566

Process returned -1073740940 (0xC0000374)   execution time : 27.742 s
Press any key to continue.
```

实验二：

根据提示进入测试，测试用例为（第一个多项式：3X+5X^2+6X^4;第二个多项式为：2X^2+4X^3+7X^4;相加后为：3X^1+7X^2+13X^4）：

```
请输入第一个多项式
请输入该多项式的序数与指数:
请按照指数升序输入，系数不能等于0且指数不能小于0，序数与指数用空格隔开，并以0 0结束输入
3 1 5 2 6 4 0 0
多项式为:
3X^1+5X^2+6X^4+NULL

请输入第二个多项式:
请输入该多项式的序数与指数:
请按照指数升序输入，系数不能等于0且指数不能小于0，序数与指数用空格隔开，并以0 0结束输入
2 2 4 3 7 4 0 0
多项式为:
2X^2+4X^3+7X^4+NULL


多项式相加后的公式为:
3X^1+7X^2+4X^3+13X^4+NULL


Process returned 0 (0x0)   execution time : 61.712 s
Press any key to continue.
```

七、用户手册

实验一：

（1）、打开并运行，按提示进行操作。

（2）、输入多个数据时，数据与数据间应该用空格隔开，输入最后一个数据时按回车进行下一步操作。

（3）、注意：未按回车键之前可以用键盘的 backspa 删除或者修改之前输入的数据，一旦按下回车键之后不能再进行删除修改操作。

实验二：

（1）、打开并运行，按文字提示进行下一步操作。

（2）、输入的多项式的序数和指数不能小于 0（输入序数为 0\指数小于 0 程序将直接输出 NULL，在本程序中设定为不允许的操作）

（3）输入序数与指数完成后输入 0 0 结束（是两个数字 0 之间有空格），按回车下一步。