

1. P45 第 3 题//单链表逆置

```
#include<malloc.h>
#include<stdio.h>
//单链表结构类型定义
typedef char datatype;
typedef struct node
{
    datatype data;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
void invert(linklist*);
int main(void)
{
    Linklist * head;
    printf("*****请创建链表*****\n");
    head=create( );
    printf("*****输出原链表*****\n");
    print(head);
    invert(head);//调用单链表逆置的函数
    printf("*****输出逆置后的链表*****\n");
    print(head);
    return 0;
}
```

2. P45 第 3 题//顺序表逆置

```
#include<stdio.h>
#include<malloc.h>
typedef char datatype;
#define  maxsize  1024
typedef struct
{ datatype data[maxsize];
  int last;
}sequenlist;
sequenlist* create( );
void print(sequenlist*);
void invert(sequenlist*);

int main(void)
{
    sequenlist*L;
    printf("*****请创建顺序表*****\n");
    L=create( );//建立顺序表
    printf("*****输出顺序表*****\n");
    print(L);//输出顺序表
    invert(L);//调用顺序表逆值的函数
    printf("*****输出逆置后的顺序表*****\n");
    print(L);//输出顺序表
    return 0;
}
```

3. P45 第 6 题// 归并递减

```
#include<malloc.h>
#include<stdio.h>
//单链表结构类型定义
typedef int datatype;
typedef struct node
{
    datatype data;
    struct node *next;
}linklist;

linklist* create();
void print(linklist *);
linklist* mergelist(linklist*, linklist *);
void insert(linklist*,linklist*);
int main(void)
{
    linklist*La,*Lb,*Lc;
    printf("*****请创建链表 La*****\n");
    La=create();
    printf("*****请创建链表 Lb*****\n");
    Lb=create();
    printf("*****输出显示链表 La*****\n");
    print(La);
    printf("*****输出显示链表 Lb*****\n");
    print(Lb);
    Lc=mergelist(La,Lb);
    printf("*****输出显示 La 表和 Lb 表归并递减后的链表 Lc*****\n");
    print(Lc);
    return 0;
}
```

4. //多项式求和运算：设单链表 A 和 B 分别存储不同的多项式，要求完成多项式的求和运算，求和结果存放在 A 表中（备注：B 表清空，测试用例如下）。

提示：

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//多项式单链表结构类型定义
typedef struct node
{
    int coef;
    int exp;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
void SumofPoly(linklist *, linklist *);
void main( )
{
    linklist*A, *B;
    printf("*****请输入原多项式A链表为*****\n");
    A = create( );
    printf("*****请输入原多项式B链表为*****\n");
    B = create( );
    printf("*****显示原多项式A链表为*****\n");
    print(A);
    printf("*****显示原多项式B链表为*****\n");
    print(B);
    SumofPoly(A, B); //调用多项式求和的函数
    printf("*****显示求和运算之后的多项式A链表为*****\n");
    print(A);
    printf("*****显示求和运算之后的多项式B链表为*****\n");
    print(B);
}
```

测试用例：

- (1) $A(x)=7+3x+9x^8+5x^{17}+2x^{20}$;
 $B(x)=8x+22x^7-9x^8-4x^{18}+30x^{25}+10x^{35}+19x^{55}$;

运行结果：

$A(x)= 7+11x+22x^7+5x^{17}-4x^{18}+2x^{20}+30x^{25}+10x^{35}+19x^{55}$;
B 表空

测试用例：

- (2) $A(x)=19+3x+72x^7+6x^{17}+2x^{28}+10x^{35}+19x^{55}$;
 $B(x)=8x^4+22x^7-6x^{17}-2x^{28}$

运行结果：

$A(x)= 19+3x+8x^4+94x^7+10x^{35}+19x^{55}$;
B 表空

测试用例：

(3) $A(x)=23+3x+7x^6+16x^{18}+2x^{23}+10x^{32};$
 $B(x)=-23-3x+17x^6-16x^{18}-2x^{23}-10x^{32};$

运行结果：

$A(x)=24x^6;$

B 表空

测试用例：

(4) $A(x)=23+3x+7x^6+16x^{18}+2x^{23}+10x^{32};$
 $B(x)=6x^{12}+16x^{38}-2x^{42}-10x^{62};$

运行结果：

$A(x)=23+3x+7x^6+6x^{12}+16x^{18}+2x^{23}+10x^{32}+16x^{38}-2x^{42}-10x^{62};$

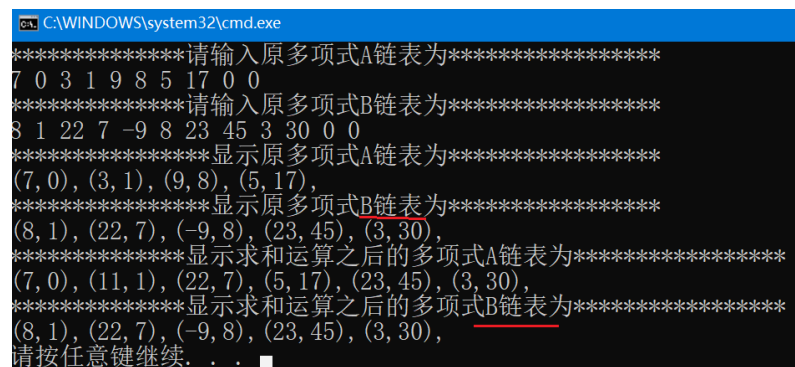
B 表空

5. //多项式求和运算：设单链表 A 和 B 分别存储不同的多项式，要求完成多项式的求和运算，求和结果存放在 A 表中（备注：B 表不变，测试用例至少测 3 组不同情况）。

提示：

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//多项式单链表结构类型定义
typedef struct node
{
    int coef;
    int exp;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
void SumofPoly(linklist *, linklist *);
void main( )
{
    linklist*A, *B;
    printf("*****请输入原多项式A链表为*****\n");
    A = create( );
    printf("*****请输入原多项式B链表为*****\n");
    B = create( );
    printf("\n*****显示原多项式A链表为*****\n");
    print(A);
    printf("\n*****显示原多项式B链表为*****\n");
    print(B);
    SumofPoly(A, B); //调用多项式求和的函数
    printf("\n*****显示求和运算之后的多项式A链表为*****\n");
    print(A);
    printf("\n*****显示求和运算之后的多项式B链表为*****\n");
    print(B);
}
```

参考类似截屏。



```
C:\WINDOWS\system32\cmd.exe
***** 请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
***** 请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0
***** 显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
***** 显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
***** 显示求和运算之后的多项式A链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
***** 显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
请按任意键继续. . .
```

测试用例：

(1) $A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;

$B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55}$;

运行结果：

$$A(x) = 7 + 11x + 22x^7 + 5x^{17} - 4x^{18} + 2x^{20} + 30x^{25} + 10x^{35} + 19x^{55};$$

$$B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55};$$

测试用例:

$$(2) A(x) = 19 + 3x + 72x^7 + 6x^{17} + 2x^{28} + 10x^{35} + 19x^{55};$$

$$B(x) = 8x^4 + 22x^7 - 6x^{17} - 2x^{28}$$

运行结果:

$$A(x) = 19 + 3x + 8x^4 + 94x^7 + 10x^{35} + 19x^{55};$$

$$B(x) = 8x^4 + 22x^7 - 6x^{17} - 2x^{28}$$

测试用例:

$$(3) A(x) = 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32};$$

$$B(x) = -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32};$$

运行结果:

$$A(x) = 24x^6;$$

$$B(x) = -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32};$$

测试用例:

$$(4) A(x) = 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32};$$

$$B(x) = 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62};$$

运行结果:

$$A(x) = 23 + 3x + 7x^6 + 6x^{12} + 16x^{18} + 2x^{23} + 10x^{32} + 16x^{38} - 2x^{42} - 10x^{62};$$

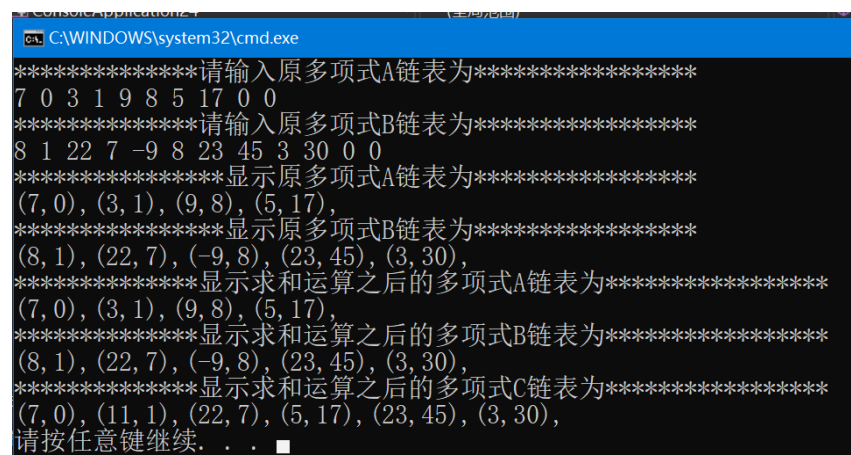
$$B(x) = 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62};$$

6. //多项式求和运算：设单链表 A 和 B 分别存储不同的多项式，要求完成多项式的求和运算，求和结果存放在 C 表中（备注：A 表、B 表均不变，生成新的多项式和链表 C；测试用例至少测 3 组不同情况）。

提示：

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//多项式单链表结构类型定义
typedef struct node
{
    int coef;
    int exp;
    struct node *next;
}linklist;
linklist* create();
void print(linklist *);
linklist * SumofPoly(linklist *, linklist *);
void main()
{
    linklist*A, *B,*C;
    printf("*****请输入原多项式A链表为*****\n");
    A = create();
    printf("*****请输入原多项式B链表为*****\n");
    B = create();
    printf("\n*****显示原多项式A链表为*****\n");
    print(A);
    printf("\n*****显示原多项式B链表为*****\n");
    print(B);
    C=SumofPoly(A, B);//调用多项式求和的函数
    printf("\n*****显示求和运算之后的多项式A链表为*****\n");
    print(A);
    printf("\n*****显示求和运算之后的多项式B链表为*****\n");
    print(B);
    printf("\n*****显示求和运算之后的多项式之和C链表为*****\n");
    print(C);
}
```

参考类似截屏。



```
C:\WINDOWS\system32\cmd.exe
*****请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
*****请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0
*****显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
*****显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
*****显示求和运算之后的多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
*****显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
*****显示求和运算之后的多项式C链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
请按任意键继续. . .
```



```
C:\WINDOWS\system32\cmd.exe
*****请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
*****请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0

*****显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),

*****显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),

*****显示求和运算之后的多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),

*****显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),

*****显示求和运算之后的多项式C链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
请按任意键继续. . .
```

```
C:\WINDOWS\system32\cmd.exe
*****请输入原多项式A链表为*****
3 56 23 60 34 70 -3 80 4 90 0 0
*****请输入原多项式B链表为*****
3 80 5 82 0 0

*****显示原多项式A链表为*****
(3, 56), (23, 60), (34, 70), (-3, 80), (4, 90),

*****显示原多项式B链表为*****
(3, 80), (5, 82),

*****显示求和运算之后的多项式A链表为*****
(3, 56), (23, 60), (34, 70), (-3, 80), (4, 90),

*****显示求和运算之后的多项式B链表为*****
(3, 80), (5, 82),

*****显示求和运算之后的多项式C链表为*****
(3, 56), (23, 60), (34, 70), (5, 82), (4, 90),
请按任意键继续. . .
```

测试用例：

(1) $A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;
 $B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55}$;

运行结果：

$A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;
 $B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55}$;
 $C(x) = 7 + 11x + 22x^7 + 5x^{17} - 4x^{18} + 2x^{20} + 30x^{25} + 10x^{35} + 19x^{55}$;

测试用例：

(2) $A(x) = 19 + 3x + 72x^7 + 6x^{17} + 2x^{28} + 10x^{35} + 19x^{55}$;
 $B(x) = 8x^4 + 22x^7 - 6x^{17} - 2x^{28}$;

运行结果：

$A(x) = 19 + 3x + 72x^7 + 6x^{17} + 2x^{28} + 10x^{35} + 19x^{55}$;
 $B(x) = 8x^4 + 22x^7 - 6x^{17} - 2x^{28}$;
 $C(x) = 19 + 3x + 8x^4 + 94x^7 + 10x^{35} + 19x^{55}$;

测试用例：

(3) $A(x) = 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}$;
 $B(x) = -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32}$;

运行结果：

$$\begin{aligned}A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\B(x) &= -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32}; \\C(x) &= 24x^6;\end{aligned}$$

测试用例：

(4)
$$\begin{aligned}A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\B(x) &= 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62};\end{aligned}$$

运行结果：

$$\begin{aligned}A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\B(x) &= 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62}; \\C(x) &= 23 + 3x + 7x^6 + 6x^{12} + 16x^{18} + 2x^{23} + 10x^{32} + 16x^{38} - 2x^{42} - 10x^{62};\end{aligned}$$