

作业一

主讲教师：金蓓弘

张远航 2015K8009929045

第 1 章 绪论

- 1.8 (1) $n - 1$; (2) $n - 1$; (3) $n - 1$; (4) $\frac{n(n+1)}{2}$;
(5) $1 + (1 + 2) + \cdots + (1 + 2 + \cdots + n) = \frac{1}{12}n(n+1)(2n+3)$;
(6) n ; (7) $\lfloor \sqrt{n} \rfloor$; (8) 1100.

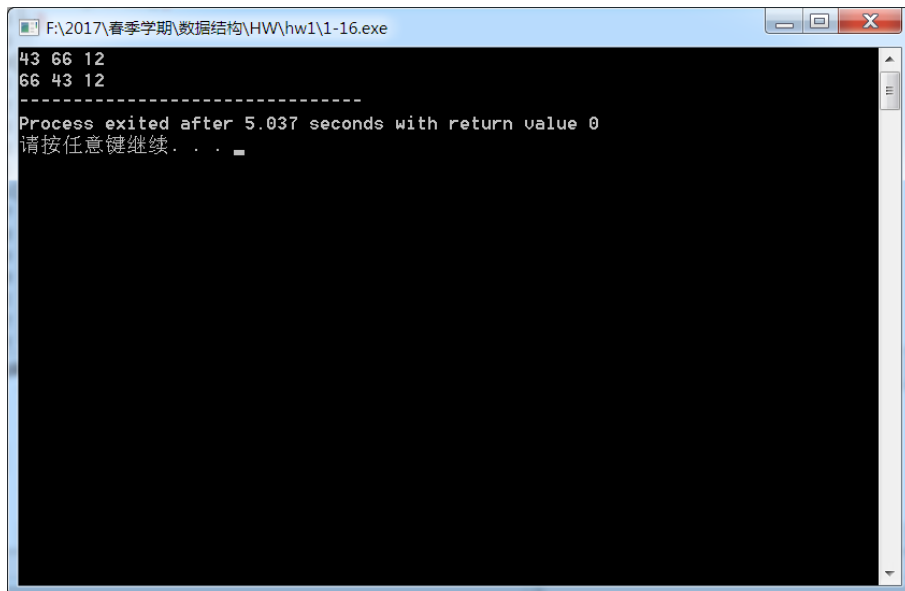
1.9 时间复杂度为 $O(\log_2 n)$, $\text{count} = \log_2 n - 2$.

1.12 (1) $\sqrt{\quad}$; (2) \times ; (3) \times ; (4) $\sqrt{\quad}$; (5) \times .

1.16 代码如下.

```
1  #include <stdio.h>
2
3  void swap(int *x, int *y) {
4      int temp;
5      temp = *x;
6      *x = *y;
7      *y = temp;
8  }
9
10 void sort(int *x, int *y, int *z) {
11     if (*x < *y) swap(x, y);
12     if (*x < *z) swap(x, z);
13     if (*y < *z) swap(y, z);
14 }
15
16 int main() {
17     int x, y, z;
18     scanf("%d%d%d", &x, &y, &z);
19     sort(&x, &y, &z);
20     printf("%d %d %d", x, y, z);
21
22     return 0;
23 }
```

输入:



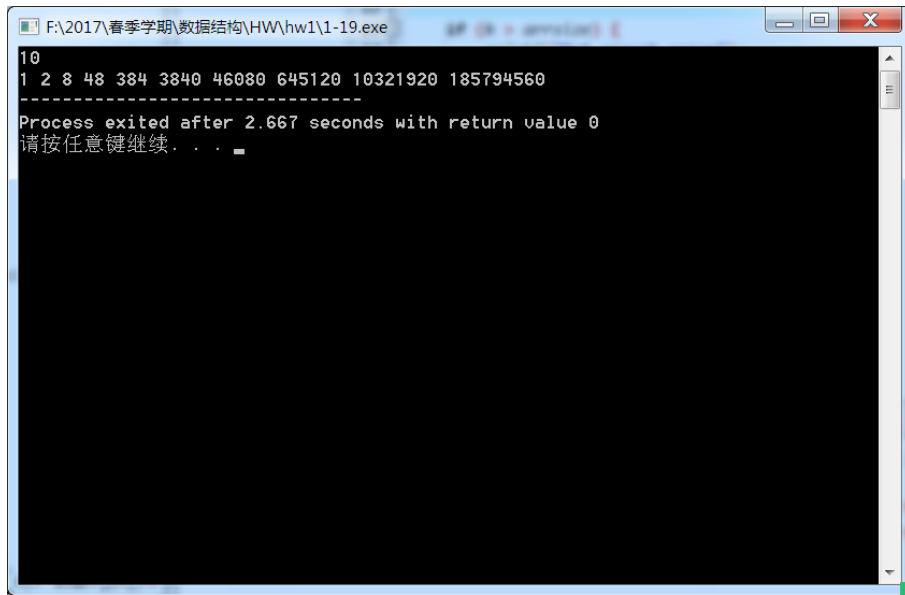
```
F:\2017\春季学期\数据结构\HW\hw1\1-16.exe
43 66 12
66 43 12
-----
Process exited after 5.037 seconds with return value 0
请按任意键继续...
```

1.19 代码如下.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <limits.h>
4  #define MAXINT INT_MAX
5
6  const int arrsize = 100;
7
8  int main(){
9      int i, k, a[arrsize];
10     scanf("%d", &k);
11
12     if (k > arrsize) {
13         printf("Not enough space");
14         exit(0);
15     }
16     for (i = 0; i < k; i++) {
17         if (!i) a[i] = 1;
18         else {
19             if (a[i-1] > MAXINT / (2*i)) {
20                 printf("Overflow");
21                 exit(0);
22             }
23             a[i] = 2 * i * a[i-1];
24         }
25     }
26
27     for (i = 0; i < k; i++)
28         printf("%d ", a[i]);
29
```

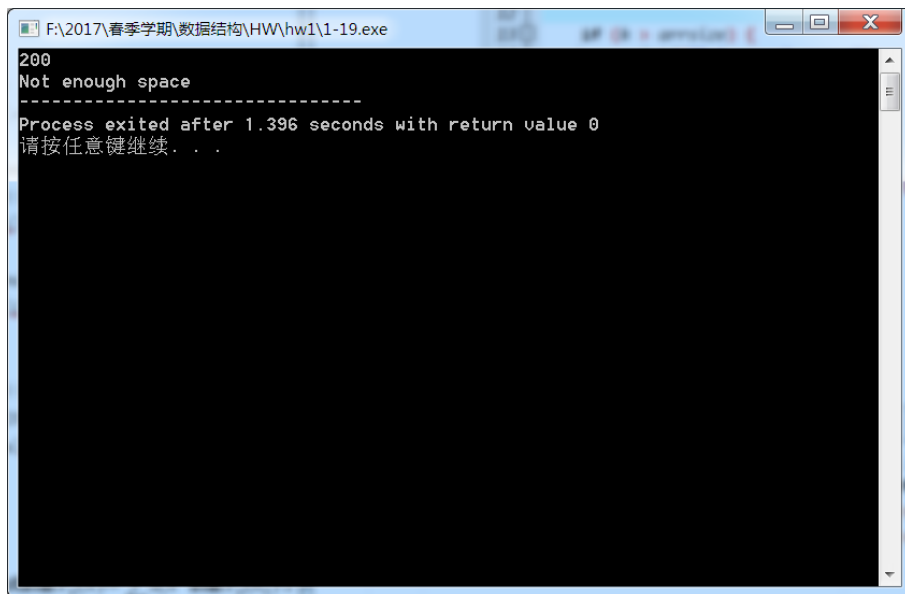
```
30     return 0;
31 }
```

输入: 10



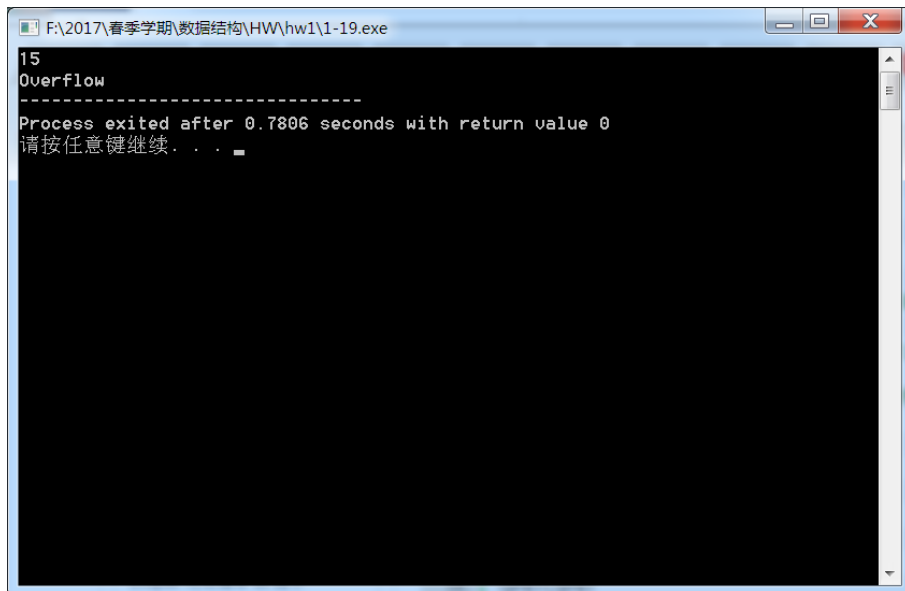
```
F:\2017\春季学期\数据结构\HW\hw1\1-19.exe
10
1 2 8 48 384 3840 46080 645120 10321920 185794560
-----
Process exited after 2.667 seconds with return value 0
请按任意键继续...
```

输入: 200



```
F:\2017\春季学期\数据结构\HW\hw1\1-19.exe
200
Not enough space
-----
Process exited after 1.396 seconds with return value 0
请按任意键继续...
```

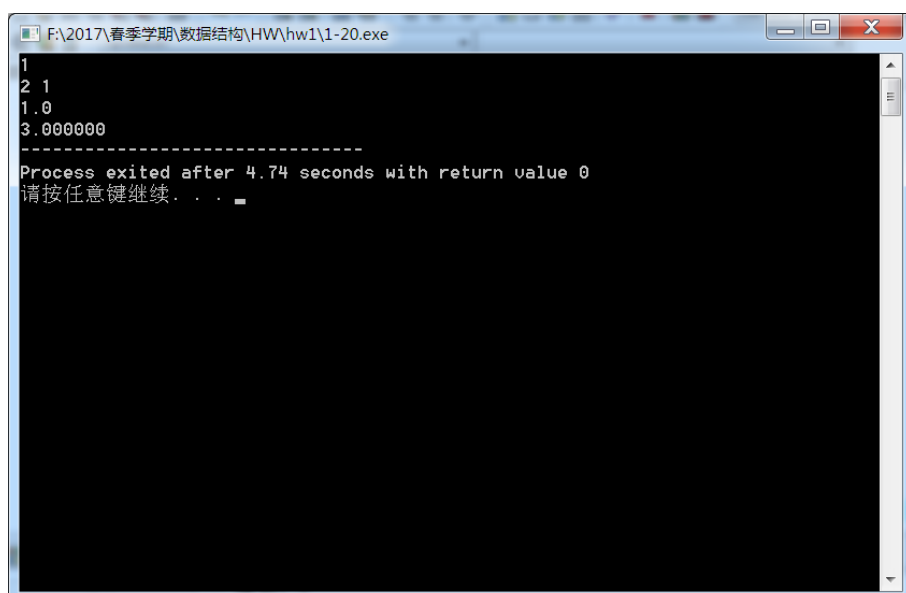
输入: 15



1.20 代码如下.

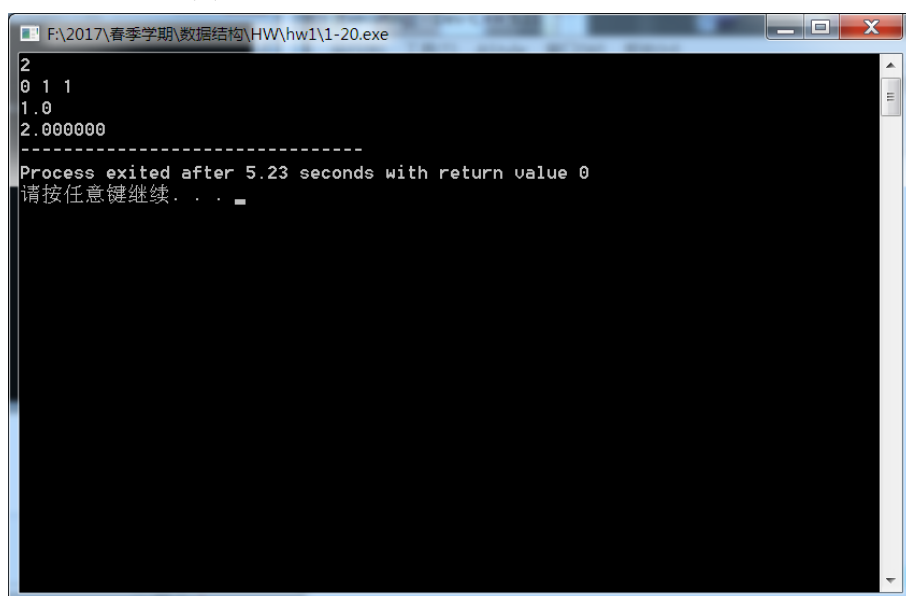
```
1  #include <stdio.h>
2  #define DEGREE 100
3
4  double eval(int *a, int n, double x0) {
5      int i;
6      double ans = (double)*a, power = 1.0;
7
8      for (i = 1; i <= n; i++) {
9          power *= x0;
10         ans += *(a+i) * power;
11     }
12
13     return ans;
14 }
15
16 int main() {
17     int n, a[DEGREE];
18     double x0;
19
20     int i;
21     scanf("%d", &n);
22     for (i = 0; i <= n; i++)
23         scanf("%d", &a[i]);
24     scanf("%lf", &x0);
25
26     printf("%lf", eval(&a[0], n, x0));
27
28     return 0;
29 }
```

输入函数为 $P_1(x) = x + 2$ ，取值 $x_0 = 1.0$ ：



```
F:\2017\春季学期\数据结构\HW\hw1\1-20.exe
1
2 1
1.0
3.000000
-----
Process exited after 4.74 seconds with return value 0
请按任意键继续...
```

输入函数为 $P_2(x) = x^2 + x$ ，取值 $x_0 = 1.0$ ：

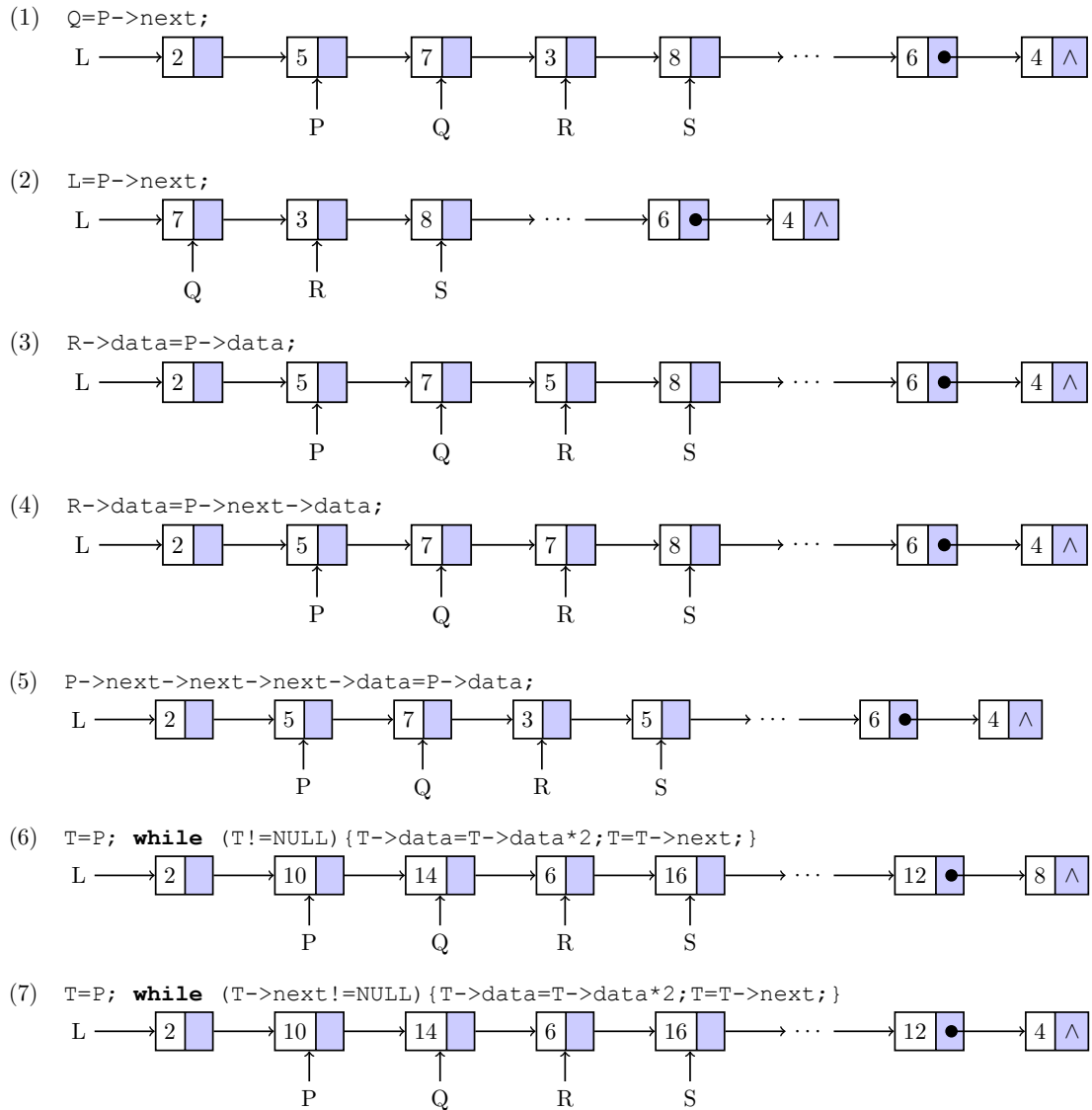


```
F:\2017\春季学期\数据结构\HW\hw1\1-20.exe
2
0 1 1
1.0
2.000000
-----
Process exited after 5.23 seconds with return value 0
请按任意键继续...
```

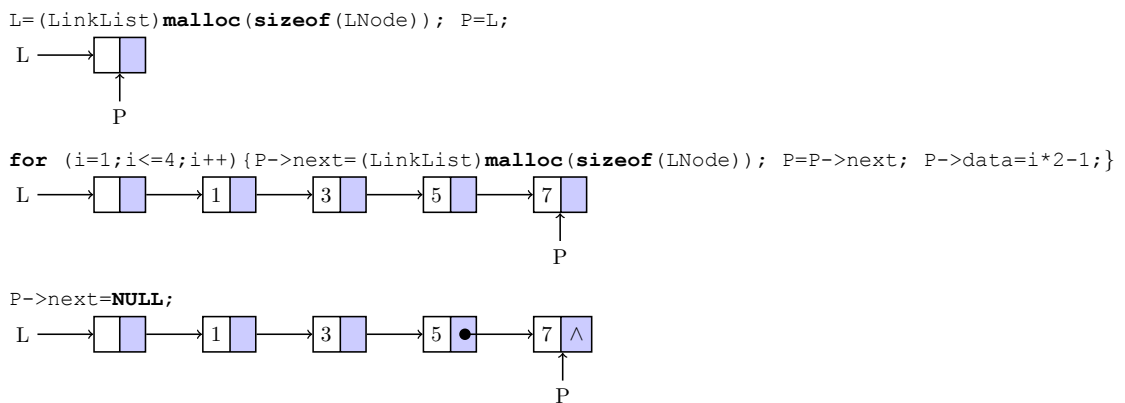
程序中，`power *= x0;` 和 `ans += *(a+i) * power;` 的执行频度均为 n 次，整个算法的时间复杂度为 $O(n)$ 。

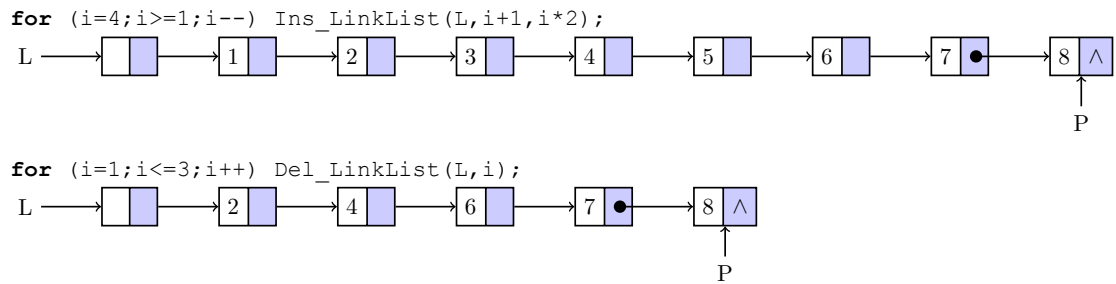
第 2 章 线性表

2.4 示意图如下:



2.5 示意图如下:





2.9 (1) 若表 L 的长度不小于 2，将 L 的首结点变成尾结点；

(2) BB 把单循环链表中的两个结点直接连接起来，AA 利用 BB 将一个单循环链表拆成两个单循环链表。

2.11 代码如下。

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define OK 1
5  #define ERROR 0
6  #define OVERFLOW -2
7
8  #define LIST_INIT_SIZE 100
9  #define LISTINCREMENT 10
10
11 // assume the elements are of type int
12 typedef int ElemType;
13 typedef int Status;
14
15 typedef struct {
16     ElemType *elem;
17     int length;
18     int listsize;
19 } SqList;
20
21 int precede(ElemType x, ElemType y) {
22     return x < y;
23 }
24
25 Status InitList_Sq(SqList *L) {
26     L->elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
27     if (!L->elem) exit(OVERFLOW);
28     L->length = 0;
29     L->listsize = LIST_INIT_SIZE;
30
31     return OK;
32 }

```

```

33
34 // insert x into the non-decreasing list va while maintaining order
35 Status InsertElem_Sq(SqList *va, ElemType x) {
36     if (va -> length == va -> listsize) {
37         ElemType *newbase = (ElemType*)realloc(va -> elem, (va -> listsize +
38             LISTINCREMENT) * sizeof(ElemType));
39         if (!newbase) return ERROR;
40         va -> elem = newbase;
41         va -> listsize += LISTINCREMENT;
42     }
43     int i;
44     for (i = va -> length; (i > 0) && precede(x, va -> elem[i-1]); i--)
45         va -> elem[i] = va -> elem[i-1];
46     va -> elem[i] = x;
47     ++va -> length;
48
49     return OK;
50 }
51
52 void PrintList_Sq(SqList *L) {
53     int i;
54
55     for(i = 0; i < L -> length; i++)
56         printf("%d ", L -> elem[i]);
57     printf("\n");
58 }
59
60 void DestroyList_Sq(SqList *L) {
61     free(L);
62 }
63
64 // test case
65 int main(){
66     SqList va;
67     int a[] = {1, 2, 3, 5, 6, 7, 8};
68
69     if (InitList_Sq(&va) != ERROR) {
70         int i;
71         for (i = 0; i < 7; i++)
72             va.elem[i] = a[i];
73         va.length = 7;
74     }
75     else {
76         printf("Error");
77         exit(ERROR);
78     }

```



```

79     // before insertion
80     PrintList_Sq(&va);
81     // after insertion
82     if (InsertElem_Sq(&va, 4) != ERROR)
83         PrintList_Sq(&va);
84     else printf("Insert failed");
85     if (InsertElem_Sq(&va, 9) != ERROR)
86         PrintList_Sq(&va);
87     else printf("Insert failed");
88     if (InsertElem_Sq(&va, 0) != ERROR)
89         PrintList_Sq(&va);
90     else printf("Insert failed");
91     if (InsertElem_Sq(&va, -5) != ERROR)
92         PrintList_Sq(&va);
93     else printf("Insert failed");
94
95     DestroyList_Sq(&va);
96
97     return 0;
98 }

```

向有序表 $va = \{1, 2, 3, 5, 6, 7, 8\}$ 中依次插入元素 4、9、0 和 -5（检查在中间和两边插入）：

```

F:\2017\春季学期\数据结构\HW\hw1\2-11.exe
1 2 3 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
-5 0 1 2 3 4 5 6 7 8 9
-----
Process exited after 0.08039 seconds with return value 3221226356
请按任意键继续...

```

2.12 代码如下.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define OK 1
5 #define ERROR 0

```

```

6  #define OVERFLOW -2
7
8  #define GREATER 1
9  #define EQUAL 0
10 #define LESS -1
11
12 #define LIST_INIT_SIZE 100
13 #define LISTINCREMENT 10
14
15 // assume the elements are of type int
16 typedef int ElemType;
17 typedef int Status;
18
19 typedef struct {
20     ElemType *elem;
21     int length;
22     int listsize;
23 } SqList;
24
25 int precede(ElemType x, ElemType y) {
26     return x < y;
27 }
28
29 Status InitList_Sq(SqList *L) {
30     L -> elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
31     if (!L -> elem) exit(OVERFLOW);
32     L -> length = 0;
33     L -> listsize = LIST_INIT_SIZE;
34
35     return OK;
36 }
37
38 // compare two sequential lists
39 Status CompareList_Sq(SqList *A, SqList *B) {
40     int p = 0;
41
42     while (A -> elem[p] == B -> elem[p]) {
43         ++p;
44         if (p == A -> length)
45             return (p == B -> length)? EQUAL: LESS;
46     }
47     if (precede(A -> elem[p], B -> elem[p]))
48         return LESS;
49     return GREATER;
50 }
51
52 void DestroyList_Sq(SqList *L) {

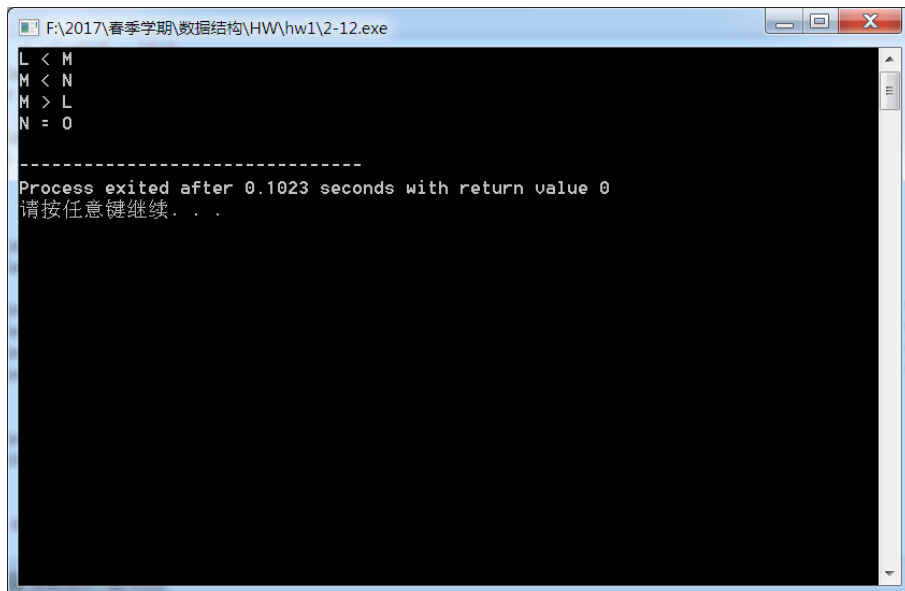
```

```

53     free(L);
54 }
55
56 int main() {
57     SqList L, M, N, O;
58     int l[] = {1, 2, 3, 4, 5, 6, 7};
59     int m[] = {1, 2, 3, 5, 6, 7, 8};
60     int n[] = {1, 2, 3, 5, 6, 7, 8, 9};
61
62     if (InitList_Sq(&L) && InitList_Sq(&M) && InitList_Sq(&N) && InitList_Sq(&
        O)) {
63         int i;
64         Status result;
65
66         for (i = 0; i < 7; i++)
67             L.elem[i] = l[i];
68         L.length = 7;
69         for (i = 0; i < 7; i++)
70             M.elem[i] = m[i];
71         M.length = 7;
72         for (i = 0; i < 8; i++) {
73             N.elem[i] = n[i];
74             O.elem[i] = n[i];
75         }
76         N.length = 8;
77         O.length = 8;
78
79         printf("L %c M\n", ((result = CompareList_Sq(&L, &M)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
80         printf("M %c N\n", ((result = CompareList_Sq(&M, &N)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
81         printf("M %c L\n", ((result = CompareList_Sq(&M, &L)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
82         printf("N %c O\n", ((result = CompareList_Sq(&N, &O)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
83     }
84     else {
85         printf("Error");
86         exit(ERROR);
87     }
88
89     return 0;
90 }

```

创建四个线性表 $L = \{1, 2, 3, 4, 5, 6, 7\}$, $M = \{1, 2, 3, 5, 6, 7, 8\}$,
 $N = O = \{1, 2, 3, 5, 6, 7, 8, 9\}$, 比较结果如下:



```
F:\2017\春季学期\数据结构\HW\hw1\2-12.exe
L < M
M < N
M > L
N = 0

-----
Process exited after 0.1023 seconds with return value 0
请按任意键继续. . .
```

2.19 代码如下.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define OK 1
5  #define ERROR 0
6
7  // assume the elements are of type int
8  typedef int ElemType;
9  typedef int Status;
10
11 typedef struct LNode {
12     ElemType data;
13     struct LNode *next;
14 } LNode, *LinkList;
15
16 int precede(ElemType x, ElemType y) {
17     return x < y;
18 }
19
20 int leq(ElemType x, ElemType y) {
21     return x <= y;
22 }
23
24 Status InitList_L(LinkList *L) {
25     *L = (LinkList)malloc(sizeof(LNode));
26     if (*L == NULL) return ERROR;
27     (*L) -> next = NULL;
28
29     return OK;
```

```

30 }
31
32 Status CreateList_L(LinkList *L, int i) {
33     LinkList p;
34
35     p = (LinkList)malloc(sizeof(LNode));
36     if (!p) return ERROR;
37     p -> data = i;
38     p -> next = (*L) -> next;
39     (*L) -> next = p;
40
41     return OK;
42 }
43
44 // delete all elements greater than mink and less than maxk
45 Status ListDeletebyBounds_L(LinkList *L, int mink, int maxk) {
46     LinkList p = (*L) -> next, left, t;
47
48     if (mink > maxk) return ERROR;
49     if (precede(maxk, p -> data)) return OK;
50
51     while (p && leq(p -> data, mink)) {
52         left = p;
53         p = p -> next;
54     }
55     if (p == NULL) return OK;
56
57     while (p && precede(p -> data, maxk)) {
58         t = p -> next;
59         free(p);
60         p = t;
61     }
62     left -> next = p;
63
64     return OK;
65 }
66
67 Status PrintList_L(LinkList *L) {
68     LinkList p = *L;
69
70     while (p -> next != NULL) {
71         p = p -> next;
72         printf("%d ", p -> data);
73     }
74     printf("\n");
75
76     return OK;

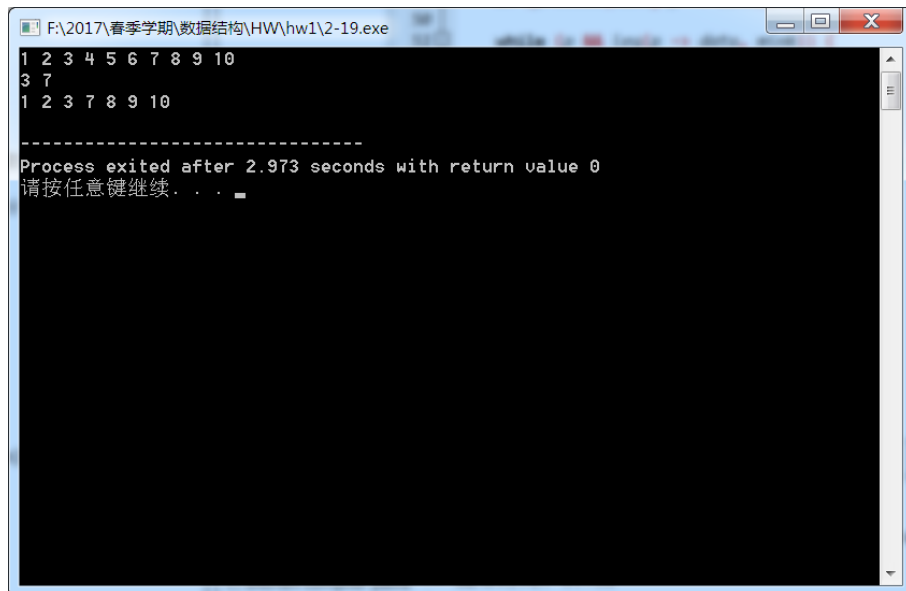
```

```

77 }
78
79 void DestroyList_L(LinkList *L) {
80     free(*L);
81 }
82
83 int main() {
84     LinkList L;
85     int i;
86
87     InitList_L(&L);
88     for (i = 10; i > 0; i--) {
89         if (CreateList_L(&L, i) != ERROR) continue;
90         else {
91             printf("Error");
92             exit(ERROR);
93         }
94     }
95     PrintList_L(&L);
96
97     int mink, maxk;
98     scanf("%d%d", &mink, &maxk);
99     if (ListDeletebyBounds_L(&L, mink, maxk) != ERROR)
100         PrintList_L(&L);
101     else {
102         printf("Error");
103         exit(ERROR);
104     }
105     DestroyList_L(&L);
106
107     return 0;
108 }

```

创建单链表 $L = \{1, 2, 3, 5, 6, 7, 8, 9, 10\}$, 输入 $mink=3$, $maxk=7$:



时间复杂度为 $O(\max k)$.

2.22 代码如下.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define OK 1
5  #define ERROR 0
6
7  // assume the elements are of type int
8  typedef int ElemType;
9  typedef int Status;
10
11 typedef struct LNode {
12     ElemType data;
13     struct LNode *next;
14 } LNode, *LinkList;
15
16 Status InitList_L(LinkList *L) {
17     *L = (LinkList)malloc(sizeof(LNode));
18     if (*L == NULL) return ERROR;
19     (*L) -> next = NULL;
20
21     return OK;
22 }
23
24 Status CreateList_L(LinkList *L, int i) {
25     LinkList p;
26
27     p = (LinkList)malloc(sizeof(LNode));
```

```

28     if (!p) return ERROR;
29     p -> data = i;
30     p -> next = (*L) -> next;
31     (*L) -> next = p;
32
33     return OK;
34 }
35
36 Status PrintList_L(LinkList *L) {
37     LinkList p = *L;
38
39     while (p -> next != NULL) {
40         p = p -> next;
41         printf("%d ", p -> data);
42     }
43     printf("\n");
44
45     return OK;
46 }
47
48 void DestroyList_L(LinkList *L) {
49     free(*L);
50 }
51
52 // in-place reversal
53 Status ReverseList_L(LinkList *L) {
54     LinkList p = (*L) -> next, q;
55
56     (*L) -> next = NULL;
57     while (p) {
58         q = p;
59         p = p -> next;
60         q -> next = (*L) -> next;
61         (*L) -> next = q;
62     }
63
64     return OK;
65 }
66
67 int main() {
68     LinkList L;
69     int i;
70
71     InitList_L(&L);
72     for (i = 10; i > 0; i--) {
73         if (CreateList_L(&L, i) != ERROR) continue;
74         else {

```

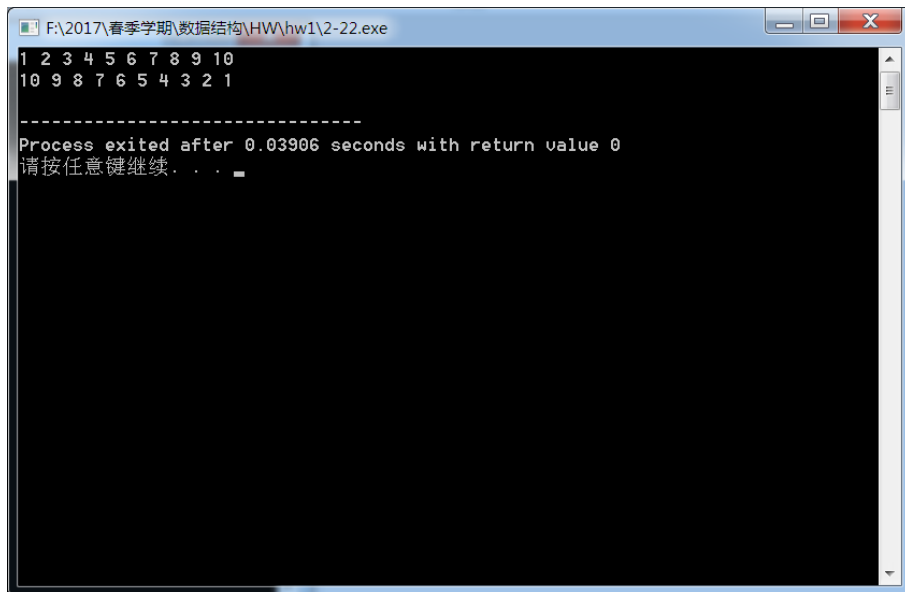


```

75         printf("Error");
76         exit(ERROR);
77     }
78 }
79 PrintList_L(&L);
80
81 if (ReverseList_L(&L) != ERROR)
82     PrintList_L(&L);
83 else {
84     printf("Error");
85     exit(ERROR);
86 }
87 DestroyList_L(&L);
88
89 return 0;
90 }

```

原地逆置单链表 $L = \{1, 2, 3, 5, 6, 7, 8, 9, 10\}$:



```

F:\2017\春季学期\数据结构\HW\hw1\2-22.exe
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
-----
Process exited after 0.03906 seconds with return value 0
请按任意键继续. . .

```

2.29 代码如下.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define OK 1
5  #define ERROR 0
6  #define OVERFLOW -2
7
8  #define FOUND 1
9  #define NONE 0
10

```

```

11 #define LIST_INIT_SIZE 100
12 #define LISTINCREMENT 10
13
14 // assume the elements are of type int
15 typedef int ElemType;
16 typedef int Status;
17
18 typedef struct {
19     ElemType *elem;
20     int length;
21     int listsize;
22 } SqList;
23
24 int precede(ElemType x, ElemType y) {
25     return x < y;
26 }
27
28 Status InitList_Sq(SqList *L) {
29     L -> elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
30     if (!L -> elem) exit(OVERFLOW);
31     L -> length = 0;
32     L -> listsize = LIST_INIT_SIZE;
33
34     return OK;
35 }
36
37 void PrintList_Sq(SqList *L) {
38     int i;
39
40     for(i = 0; i < L -> length; i++)
41         printf("%d ", L -> elem[i]);
42     printf("\n");
43 }
44
45 Status LocateElem_Sq(SqList *L, ElemType e) {
46     int i;
47
48     if (precede(e, L -> elem[0]) || precede(L -> elem[L -> length-1], e))
49         return NONE;
50     for (i = 0; i < L -> length && e != L -> elem[i]; i++)
51         ;
52     return (i == L -> length)? NONE: FOUND;
53 }
54 // note that the list is only non-decreasing so there may be multiple
55 // occurrences
56 Status DeleteElem_Sq(SqList *L, ElemType e) {

```

```

56     int i = 0, l, r;
57
58     if (precede(e, L -> elem[0]) || precede(L -> elem[L -> length-1], e))
        return OK;
59     for (i = 0; (i < L -> length) && (e != L -> elem[i]); i++)
60         ;
61     if (i == L -> length) return OK;
62     l = i;
63     while (e == (L -> elem[i]) && i < L -> length) i++;
64     r = i;
65     L -> length = L -> length - r + l;
66     for (i = l; i < L -> length; i++)
67         L -> elem[i] = L -> elem[i+(r-l)];
68
69     return OK;
70 }
71
72 Status RemoveMutual_Sq(SqList *A, SqList *B, SqList *C) {
73     int i;
74
75     for(i = 0; i < B -> length; i++)
76         if (LocateElem_Sq(C, B -> elem[i]) == FOUND) DeleteElem_Sq(A, B ->
            elem[i]);
77     return OK;
78 }
79
80 void DestroyList_Sq(SqList *L) {
81     free(L);
82 }
83
84 int main() {
85     SqList A, B, C;
86     int a[] = {1, 2, 3, 4, 5, 6, 7, 7};
87     int b[] = {0, 2, 3, 4, 6, 7, 9, 9};
88     int c[] = {1, 2, 3, 6, 7, 8, 9, 9};
89
90     if (InitList_Sq(&A) && InitList_Sq(&B) && InitList_Sq(&C)) {
91         int i;
92
93         for (i = 0; i < 7; i++) {
94             A.elem[i] = a[i];
95             B.elem[i] = b[i];
96             C.elem[i] = c[i];
97         }
98         A.length = 7;
99         B.length = 7;
100        C.length = 7;

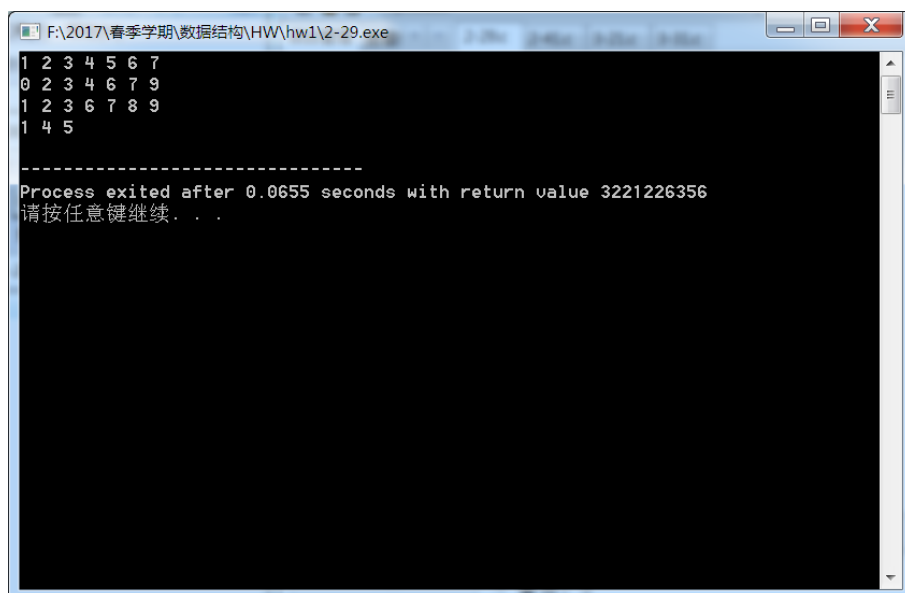
```

```

101         PrintList_Sq(&A);
102         PrintList_Sq(&B);
103         PrintList_Sq(&C);
104     }
105     else {
106         printf("Error");
107         exit(ERROR);
108     }
109
110     RemoveMutual_Sq(&A, &B, &C);
111     PrintList_Sq(&A);
112     DestroyList_Sq(&A);
113     DestroyList_Sq(&B);
114     DestroyList_Sq(&C);
115
116     return 0;
117 }

```

从表 $A = \{1, 2, 3, 4, 5, 6, 7, 7\}$ 中删除表 $B = \{0, 2, 3, 4, 6, 7, 9, 9\}$ 和 $C = \{1, 2, 3, 6, 7, 8, 9, 9\}$ 的共同元素:



```

F:\2017\春季学期\数据结构\HW\hw1\2-29.exe
1 2 3 4 5 6 7
0 2 3 4 6 7 9
1 2 3 6 7 8 9
1 4 5
-----
Process exited after 0.0655 seconds with return value 3221226356
请按任意键继续...

```

2.41 代码如下.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define OK 1
5  #define ERROR 0
6
7  typedef int Status;
8

```

```

9  typedef struct {
10      int coef;
11      int exp;
12  } PolyTerm;
13
14  typedef struct PolyNode {
15      PolyTerm data;
16      struct PolyNode *next;
17  } PolyNode, *PolyLink;
18
19  typedef PolyLink LinkedPoly;
20
21  Status CreatePoly(LinkedPoly *L, int n) {
22      *L = (LinkedPoly)malloc(sizeof(PolyNode));
23      if (*L == NULL) return ERROR;
24      (*L) -> next = NULL;
25
26      int i;
27      LinkedPoly p, prev, q;
28
29      for (i = 0; i <= n; i++) {
30          p = (LinkedPoly)malloc(sizeof(PolyNode));
31          if (!p) return ERROR;
32          scanf("%d", &p -> data.coef);
33          p -> data.exp = i;
34          prev = *L;
35          q = (*L) -> next;
36          while (q && (q -> data.exp < p -> data.exp)) {
37              prev = q;
38              q = q -> next;
39          }
40          p -> next = q;
41          prev -> next = p;
42      }
43
44      return OK;
45  }
46
47  Status PrintPoly(LinkedPoly *L) {
48      LinkedPoly p;
49      p = *L;
50      printf("f(x) = ");
51      while (p -> next) {
52          if (p -> next -> data.coef)
53              if (p -> next -> data.exp != 0)
54                  printf("%dx^%d%c", p -> next -> data.coef, p -> next -> data.
                    exp, (p -> next -> next)? '+': ' ');

```

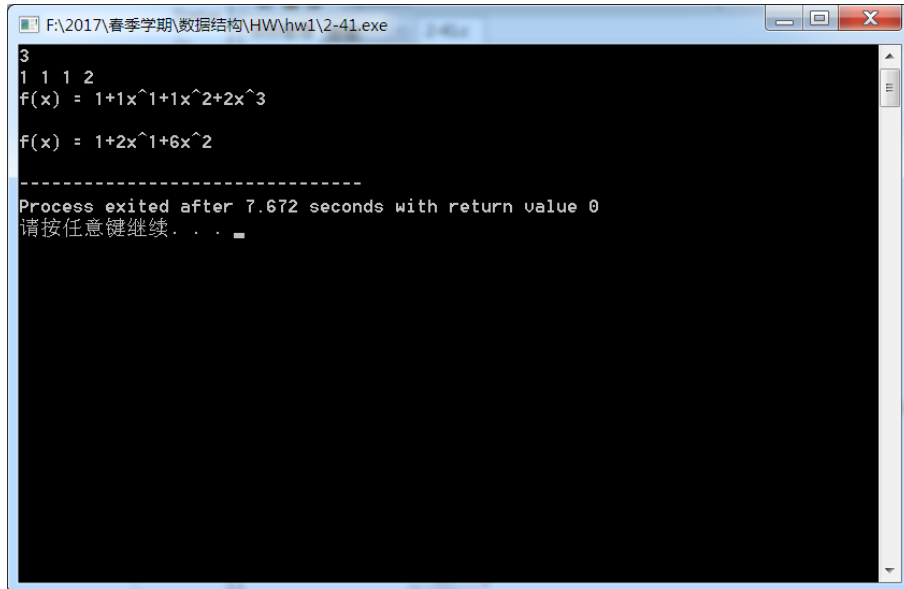
```

55         else
56             printf("%d%c", p -> next -> data.coef, (p -> next -> next)? '+'
                    ': ' ');
57         p = p -> next;
58     }
59     printf("\n");
60
61     return OK;
62 }
63
64 Status DifferentiatePoly(LinkedPoly *L) {
65     LinkedPoly p, q, t;
66
67     q = *L;
68     p = (*L) -> next;
69     while (p) {
70         if (p -> data.exp == 0) {
71             t = p;
72             p = p -> next;
73             q -> next = p;
74             free(t);
75         }
76         else {
77             p -> data.coef *= p -> data.exp;
78             --p -> data.exp;
79             q = p;
80             p = p -> next;
81         }
82     }
83
84     return OK;
85 }
86
87 int main() {
88     LinkedPoly P;
89     int n;
90
91     scanf("%d", &n);
92     if (CreatePoly(&P, n) != ERROR) {
93         PrintPoly(&P);
94         DifferentiatePoly(&P);
95         printf("\n");
96         PrintPoly(&P);
97     }
98     else {
99         printf("ERROR");
100        exit(ERROR);

```

```
101     }
102
103     return 0;
104 }
```

首先读入多项式的最高项次数 $n = 3$ ，然后次数由低到高依次读入各项系数 $1, 1, 1, 2$ ，对多项式 $2x^3 + x^2 + x + 1$ 求导：



```
F:\2017\春季学期\数据结构\HW\hw1\2-41.exe
3
1 1 1 2
f(x) = 1+1x^1+1x^2+2x^3
f(x) = 1+2x^1+6x^2
-----
Process exited after 7.672 seconds with return value 0
请按任意键继续. . .
```