

【例 9-2】 输入两个数，并使其从大到小输出。

```
1 #include <stdio.h>
2
3 void main ( )
4 {
5     int *p1, *p2, *p, a, b;
6
7     scanf ("%d, %d", &a, &b);
8     p1=&a; p2=&b;           //p1 指向变量 a, p2 指向变量 b
9     if ( a<b )
10    { p=p1; p1=p2; p2=p; } //将 p1、p2 所指向的变量地址交换
11    printf ("a=%d, b=%d\n", a, b);
12    printf ("max=%d, min=%d\n", *p1, *p2);
13 }
```

【例 9-3】 利用一般指针变量对二维数组的引用。

```
1 #include <stdio.h>
2
3 void main ( )
4 {
5     short int a[2][3]={{1, 2, 3}, {4, 5, 6}};
6     short int i, j, *p;
7     p=&a[0][0];           //也可写成 p=a[0]; p 指向二维数组 a 的第一个单元
8     for (i=0; i<2; i++) //变量 i 控制行数
9     {
10        for (j=0; j<3; j++) //变量 j 控制列数
11            printf("a[%d] [%d]=%d", i, j, *(p+i*3+j)); //显示每个数组元素的值
12        printf ("\n"); //数组的每一行元素输出后换行
13    }
14 }
```

【例 9-4】利用二维数组的行指针对二维数组的引用。

```
1 #include <stdio.h>
2
3 void main ( )
4 {
5     short int a[2][3]={{1, 2, 3}, {4, 5, 6}};
6     short int (*p)[3];
7     short int i, j; 行指针
8
9     p=a;
10    for (i=0; i<2; i++) // p 指向二维数组 a 的第一行行地址
11    {
12        for (j=0; j<3; j++) // 变量 i 控制行数 // 变量 j 控制列数
13        printf ("a[%d] [%d]=%d ", i, j, p[i][j]); // 显示每个数组元素的值
14        printf ("\n"); // 数组的每一行元素输出后换行
15    }
```

```
16     p++; // p 指向二维数组 a 的第二行行地址
17     for (j=0; j<3; j++)
18         printf ("%d ", p[0][j]); // 显示数组中第二行的元素值
19 }
```

运行结果：

```
a[0][0]=1 a[0][1]=2 a[0][2]=3
a[1][0]=4 a[1][1]=5 a[1][2]=6
4 5 6
```

【例 9-5】利用指针数组对键盘输入的 5 个整数进行从小到大排序。

```
1 #include <stdio.h>
2 void main ( )
3 {
4     int i, j, t;
5     int a, b, c, d, e;
6     int *p[5]={&a,&b,&c,&d,&e}; //将 a,b,c,d,e 的内存地址分别赋给 p[0]…p[4]
7
8     scanf ("%d,%d,%d,%d,%d",p[0],p[1],p[2],p[3],p[4]); //对 a,b,c,
9     for (i=0; i<4; i++)          //利用冒泡法排序
10    for (j=i+1; j<5; j++)
11        if (*p[i]>*p[j])           //交换 p[i]、p[j] 所指向的变量值
12        {
13            t=*p[i];
14            *p[i]=*p[j];
15            *p[j]=t;
16        }
17    for (i=0; i<5; i++)          //显示排序后的结果
18        printf("%d ", *p[i]);
19 }
```

运行结果 (假设输入为: 3, 8, 7, 6, 4↙):

```
3 4 6 7 8
```

【例 9.9】

利用字符指针数组对一组城市名进行升序排列。

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void main ()
5 {
6     int i, j, k;
7     char *pcity[]={"Wuhan", "Beijing", "Shanghai", "Tianjin", "Guangzhou", ""};
8     char *ptemp;
9
10    for (i=0; strcmp(pcity[i], "") !=0; i++)
11    {
12        k=i; //k 为当前最小字符串的字符指针数组的下标, 初始假设为 i
13        for (j=i+1; strcmp(pcity[j], "")!=0; j++) //找比 pcity[k] 小的字
14            if (strcmp(pcity[k], pcity[j])>0)
15                k=j;
16
17        if (k !=i) //将最小字符串的地址 pcity[k] 与 pcity[i] 交换
18        {
19            ptemp=pcity[i];
20            pcity[i]=pcity[k];
21            pcity[k]=ptemp;
22        }
23    }
24
25    for (i=0; strcmp(pcity[i], "") !=0; i++) //显示排序后的结果
26    printf ("%s ", pcity[i]);
27    printf ("\n");
28 }
```

运行结果:

```
Beijing Guangzhou Shanghai Tianjin Wuhan
```

【例 9-11】 利用二级指针来处理字符串。

```
1 #include <stdio.h>
2 #define NULL 0
3
4 void main ( )
5 {
6     char **p;
7     char *name[ ] = { "hello", "good", "world", "bye", "" };
8
9     p=name+1;
10    printf ("%x: %s ", *p, *p);
11    p +=2;
12    while (**p !=NULL)
13        printf ("%s\n", *p++);
14 }
```

运行结果：

```
42003C: good bye
```

【例 9-13】求某矩阵中各行元素之和的最大值。

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <malloc.h>
4
5 int GetSumRow (int *p, int num); //求某行某列元素之和
6 int GetMaxRow (int **p, int row, int col); //求各行最大值
7
8 void main ( )
9 {
10     int row, col;
11     int i, j, **p, maxrow;
12
13     printf ("input row=");
14     scanf ("%d", &row);
15     printf ("input col=");
16     scanf ("%d", &col);
17
18     //根据输入的矩阵的行数和列数来动态建立一个二维数组 p
19     p=(int **) malloc(row * sizeof(int *));
20     for (i=0 ; i<row; i++)
21         p[i]=(int *) malloc(col * sizeof(int));
22
23     //通过输入值对二维数组的元素赋值
24     printf ("input the number:\n");
25     for (i=0 ; i<row; i++)
26         for (j=0; j<col; j++)
27             scanf ("%d", p[i]+j);
28
29     //调用函数来计算矩阵中所有行元素之和的最大值
30     maxrow=GetMaxRow (p, row, col);
31     printf ("-----\n");
32     printf ("maxrow=%d\n", maxrow);
33
34     //释放动态分配的内存空间

```

```

35     for (i=0 ; i<row; i++)
36         free (p[i]);
37     free (p);
38 }
39 //计算矩阵中所有行元素之和的最大值
40 int GetMaxRow (int **p, int row, int col)
41 {
42     int i, max, t;
43
44     max=GetSumRow (p[0], col);
45     for (i=1 ; i<row; i++)
46     {
47
48         t=GetSumRow (p[i], col);
49         if (t>max)
50             max=t;
51     }
52     return (max);
53 }
54 pstart=psstr;
55 //计算矩阵中某行元素之和
56 int GetSumRow (int *p, int num)
57 {
58     int i, sum=0;
59     while (pstart <=pend) //指向左边第一个非空格字符的位置
60     for (i=0; i<num; i++)
61         sum +=p[i];
62     return (sum);
63 }

```

运行结果 (假设输入如下):

```

input row=3
input col=4
input the number:
3 -4 6 8
4 6 -2 7
0 9 8 6

```

maxrow=23

【例 11-1】计算学生 5 门课的平均成绩，最高分和最低分。

```
1 #include <stdio.h>
2
3 struct score
4 {
5     float grade[5];
6     float avegrade, maxgrade, mingrade;
7 }
8
9 void main ()
10 {
11     int i;
12     struct score m;
13
14     printf ("input the grade of five course:\n");
15     for (i=0; i<5; i++) //输入 5 门课的成绩
```

```
16     scanf ("%f", &m.grade[i]);
17
18     m.avegrade=0;
19     m.maxgrade=m.grade[0];
20     m.mingrade=m.grade[0];
21     for (i=0; i<5; i++) //求平均分、最高分、最低分
22     {
23         m.avegrade +=m.grade[i];
24         m.maxgrade=(m.grade[i]>m.maxgrade) ? m.grade[i] : m.maxgrade;
25         m.mingrade=(m.grade[i]<m.mingrade) ? m.grade[i] : m.mingrade;
26     }
27     m.avegrade /=5;
28     printf ("avegrade=%5.1f maxgrade=%5.1f mingrade=%5.1f\n",
29             m.avegrade, m.maxgrade, m.mingrade);
30 }
```

运行结果（假设输入 5 门课的成绩为： 75 80 86 90 68）：

```
avegrade=79.8 maxgrade=90.0 mingrade=68.0
```

【题 6.125】从键盘输入若干整数(数据个数应少于 50),其值在 0~4 的范围内,用 -1 作为输入结束的标志;统计同一整数的个数。试编程。

【题 6.126】若有说明: int a[2][3]={{1,2,3},{4,5,6}}; 现要将行和列的元素互换后存到另一个二维数组 b 中。试编程。

【题 6.127】定义一个含有 30 个整型元素的数组,按顺序分别赋予从 2 开始的偶数;然后按顺序每 5 个数求出一个平均值,放在另一个数组中并输出。试编程。

【题 6.128】通过赋初值按行顺序给 2×3 的二维数组赋予 2、4、6、…等偶数,然后按列的顺序输出该数组。试编程。

【题 6.129】通过循环按行顺序为一个 5×5 的二维数组 a 赋 1~25 的自然数,然后输出该数组的左下三角。试编程。

【题 6.130】下面是一个 5 阶的螺旋方阵。试编程打印出此形式的 $n(n \leq 10)$ 阶的方阵(顺时针方向旋转)。

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

【题 6.131】数组 a 包括 10 个整数,把 a 中所有的后项除以前项之商取整后存入数组 b,并按每行 3 个元素的格式输出数组 b。试编程。

【题 6.132】从键盘输入一个字符,用折半查找法找出该字符在已排序的字符串 a 中的位置。若该字符不在 a 中,则打印出 **。试编程。

【题 6.133】从键盘输入两个字符串 a 和 b,要求不用库函数 strcat 把串 b 的前 5 个字符连接到串 a 中;如果 b 的长度小于 5,则把 b 的所有元素都连接到 a 中。试编程。

【题 6.134】从键盘输入一个字符串 a,并在 a 串中的最大元素后边插入字符串 b ($b[] = "ab"$)。试编程。

```
【題 6.125】 #include <stdio.h>
#define M 50
int main()
{ int a[M], c[5], i, n=0, x;
printf("Enter 0 or 1 or 2 or 3 or 4, to end with -1\n");
scanf("%d", &x);
while( x!= -1)
{ if( x>=0 && x<=4)
{ a[n]=x; n++; }
scanf("%d", &x);
}
for( i=0;i<5;i++) c[i]=0;
for( i=0;i<n;i++) c[a[i]]++;
printf("The result is :\n");
for( i=0;i<=4;i++) printf("%d: %d\n", i, c[i]);
return 0;
}
```

```
【題 6.126】 #include <stdio.h>
int main()
{ int a[5][4]={3,6,2,1,3,9,0,8,2,1,5,6,7,2,7,4,0,0,0,0};
int i,j;
for(i=0;i<4;i++)
{ for(j=0;j<4;j++)
a[4][j] += a[i][j];
}
printf("The result is :\n");
for(i=0;i<4;i++) printf("%3d", a[4][i]);
return 0;
}
```

```
【題 6.127】 #include <stdio.h>
#define SIZE 30
```

```

int main( )
{
    float b[SIZE/5], sum; int a[SIZE], i,j,k;
    for( k=2,i=0;i<SIZE; i++)
    {
        a[i]=k; k+=2;
    }
    sum=0.0;
    for( k=0,i=0;i<SIZE;i++)
    {
        sum+=a[i];
        j=i+1;
        if((i+1)%5==0)
        {
            b[k]=sum/5;
            sum=0;
            k++;
        }
    }
    printf("The result is:\n");
    for(i=0;i<SIZE/5;i++) printf("%5.2f",b[i]);
    return 0;
}

```

【題 6.128】 # include <stdio.h>

```

int main( )
{
    int i,j,a[2][3]={{2,4,6},{8,10,12}};
    printf("The original array is:\n");
    for( i=0; i<2; i++)
    {
        for ( j=0; j<3; j++) printf("%4d",a[i][j]);
        printf("\n");
    }
    printf("\nthe result is :\n");
    for( i=0; i<3; i++)
    {
        for( j=0; j<2; j++) printf("%4d",a[j][i]);
        printf("\n");
    }
    return 0;
}

```

【題 6.129】 # include <stdio.h>

```

int main ( )
{
    int a[5][5], i, j, n=1;
    for( i=0; i<5; i++)
        for( j=0; j<5; j++)
            a[i][j]=n++;
    printf ("The result is :\n");
    for( i=0; i<5; i++)
        {
            for( j=0; j<=i; j++)
                printf ("%4d",a[i][j]);
            printf ("\n");
        }
    return 0;
}

```

```

        printf("\n");
    }
    if((i+j)>=m) return 0;
}

```

【题 6.130】 #include <stdio.h>

```

int main( )
{
    int a[10][10], i, j, k=0, m, n;
    printf("Enter n (n<10) : \n");
    scanf("%d", &n);
    if(n%2==0) m=n/2;
    else m=n/2+1;
    for(i=0; i<m; i++)
    {
        for(j=i; j < n-i; j++)
        {
            k++;
            a[i][j]=k;
        }
        for(j=i+1; j < n-i; j++)
        {
            k++;
            a[j][n-i-1]=k;
        }
        for(j=n-i-2; j >= i; j--)
        {
            k++;
            a[n-i-1][j]=k;
        }
        for(j=n-i-2; j >= i+1; j--)
        {
            k++;
            a[j][i]=k;
        }
    }
    for(i=0; i < n; i++)
    {
        for(j=0; j < n; j++)
            printf("%5d", a[i][j]);
        printf("\n");
    }
    return 0;
}

```

【题 6.131】 #include <stdio.h>

```

int main( )
{
    int a[10], b[10], i;
    for(i=0; i<10; i++) scanf("%d", &a[i]);
    for(i=1; i<10; i++) b[i]=a[i]/a[i-1];
    for(i=1; i<10; i++)
    {
        printf("%3d", b[i]);
        if(i%3==0) printf("\n");
    }
    return 0;
}

```

【题 6.132】 #include <stdio.h>

```

int main( )
{
    char a[12] = "adfgikmnprs", c; int i, top, bot, mid;
    printf("Input a character\n");
}

```

```

scanf("%c", &c);
printf("c=%'c\n", c);
for(top=0, bot=10; top<=bot; )
{ mid=(top+bot)/2;
  if(c==a[mid])
    { printf("The position is %d\n", mid+1);
      break;
    }
  else if(c>a[mid]) top=mid+1;
  else bot=mid-1;
}
if(top>bot) printf("* *\n");
return 0;
}

```

【题 6.133】 #include <stdio.h>
 #include <string.h>
 int main()
 { char a[80],b[80]; int i=0,j;
 printf("Input two strings.\n");
 gets(a); gets(b);
 while(a[i++]!='\0');
 for(j=0,i--; j<5 && b[j]!='\0'; j++)
 a[i++]=b[j];
 a[i]='\0';
 puts(a);
 return 0;
 }

【题 6.134】 #include <stdio.h>
 #include <string.h>
 int main()
 { char a[80],b[]="ab",max; int i=1,j;
 printf("Input a string\n");
 gets(a);
 puts(a);
 max=a[0];
 while(a[i]!='\0')
 { if(a[i]>max)
 { max=a[i]; j=i; }
 i++;
 }
 for(i=strlen(a)+2; i>j; i--)
 a[i]=a[i-2];
 a[i+1]='a'; a[i+2]='b';
 }

```
    puts(a);
    return 0;
}
```

7.3 编 程 题

题 7.68】函数 fun 的功能是：判断输入的 3 个整型值能否组成三角形，组成的是等边三角形，还是等腰三角形。请在函数中填写正确的内容。

```
#include <stdio.h>
void fun(int a, int b, int c);
int main()
{
    int a,b,c;
    printf("\ninput a,b,c:\n");
    scanf("%d%d%d", &a, &b, &c);
    fun(a,b,c);
    return(0);
}
void fun(int a,int b,int c)
{
    if(a+b>c&&b+c>a&&a+c>b)
        (请在此处填写正确的内容)
    else
        printf("不能组成三角形");
}
```

题 7.69】已有变量定义和函数调用语句 int x=57; isprime(x); 函数 isprime() 用来判断一个整型数 a 是否为素数；若是素数，则函数返回 1，否则返回 0。请编写 isprime 函数。

```
int isprime(int a)
{
}
```

题 7.70】已有变量定义和函数调用语句 int a,b; b=sum(a); 函数 sum() 用以求 $\sum k$ ，和数作为函数值返回。若 a 的值为 10，则经函数 sum 的计算后，b 的值是 55。请编写 sum 函数。

```
int sum(int n)
{
}
```

题 7.71】已有变量定义语句 double a=5.0,p;int n=5; 和函数调用语句 p=mypow(a, n)；用以求 a 的 n 次方。请编写 double mypow(double x,int y) 函数。

7.3 编程题

```
【题 7.68】 { if(a==b&&b==c) printf("这是等边三角形");  
    else if(a==b||b==c||a==c) printf("这是等腰三角形");  
    else  
        printf("组成一般三角形");  
}
```

• 292 •

【题 7.69】 int isprime(int a)

```
{ int i;  
    for(i=2;i<sqrt((double)a);i++)  
        if(a%i==0) return 0;  
    return 1;  
}
```

【题 7.70】 int sum(int n)

```
{ int i,k=0;  
    for(i=0;i<=n;i++) k+=i;  
    return k;  
}
```

【题 7.71】 double mypow(double x,int y)

```
{ int i; double p;  
    p=1.0;  
    for(i=1;i<=y;++) p=p*x;  
    return p;  
}
```

【题 7.72】 double f(float x0)

```
{ double x1;  
    x1=(cos(x0)-x0)/(sin(x0)+1);  
    x1=x1+x0;  
    return x1;  
}
```

【题 7.73】 for(i=1;i<=x;i++)

【题 9.186】 编写程序, 将字符串 computer 赋给一个字符数组, 然后从第一个字母开始间隔地输出该串, 请用指针完成。

【题 9.187】 编写程序, 将字符串中的第 m 个字符开始的全部字符复制成另一个字符串。要求在主函数中输入字符串及 m 的值并输出复制结果, 在被调用函数中完成复制。

【题 9.188】 从键盘输入一个字符串, 然后按照下面要求输出一个新字符串。新串是在原串中每两个字符之间插入一个空格, 如原串为 abcd, 则新串为 a□b□c□d(□代表空格)。要求在函数 insert 中完成新串的产生; 并在函数中完成所有相应的输入和输出。

【题 9.189】 设有一个数列, 包含 10 个数, 已按升序排好。现要求编写程序, 把从指定位置开始的 n 个数按逆序重新排列并输出新的完整数列。进行逆序处理时要求使用指针方法。试编程。(例如: 原数列为 2、4、6、8、10、12、14、16、18、20, 若要求把从第 4 个数开始的 5 个数按逆序重新排列, 则得到新数列为 2、4、6、16、14、12、10、8、18、20。)

【题 9.190】 编写程序, 统计 d 输入的命令行中第二个参数所包含的英文字符个数。

【题 9.191】 通过指针数组 p 和一维数组 a 构成一个 3×2 的二维数组, 并为 a 数组赋初值 2、4、6、8...。要求先按行的顺序输出此二维数组, 然后再按列的顺序输出它。试编程。

9.3 编程题

【题 9.186】 #include <stdio.h>

```
int main()
{
    static char x[] = "computer";
    char * p;
    for(p=x; p<x+7; p+=2)
        putchar(*p);
    printf("\n");
    return 0;
}
```

【题 9.187】 #include <stdio.h>

```
#include <string.h>
void cpystr(char * p1, char * p2, int m);
int main()
{
    int m, l;
    char str1[80], str2[80];
    printf("Input a string:\n");
    gets(str2);
    printf("Input m:\n");
    scanf("%d", &m);
    l = strlen(str2);
    if (m > l)
        m = l;
    cpystr(str1, str2, m);
    printf("The string after copy is %s\n", str1);
}
```

```

187】 scanf("%d", &m); l = pointer((char*)m, '0'); if(m>=0) l--;
l = strlen(str2);
if(l<m) printf("Err input!\n");
else { copystr(str1,str2,m);
printf("Result is : %s\n",str1);
}
return 0;
}

void copystr(char * p1,char * p2,int m)
{ int n=0;
while(n<m-1)
{ p2++,n++; }
while(*p2!='0')
{ *p1= *p2; p1++; p2++; }
*p1='0';
}

188】 #include<stdio.h>
#include <string.h>
void insert(char * p);
int main( )
{ char str[80];
printf("Input a string:\n");
gets(str);
insert(str);
printf("Result is: %s\n",str);
return(0);
}

void insert(char * p)
{ int i;
for(i=strlen(p); i>0; i--)
{ *(p+2*i)= *(p+i);
*(p+2*i-1)=' ';
}
}

189】 #include <stdio.h>
int main( )
{ int b[10], position, num, k, * q1, * q2, temp;
printf("Input 10 sorted numbers\n");
for (k=0; k<10; k++)
scanf("%d",&b[k]);
printf("\nInput the position:\n");

```

```

scanf("%d", &position);
printf("Input the number of data that be sorted again:\n");
scanf("%d", &num);
printf("The old array b is:\n");
for(k=0; k<10; k++)
    printf("%4d", b[k]);
printf("\n");
q1 = &b[position-1];
q2 = &b[position-2 + num];
for(; q1<&b[position-1]+num/2; q1++, q2--)
{
    temp = *q1; *q1 = *q2; *q2 = temp;
}
printf("The new array b is:\n");
for(k=0; k<10; k++)
    printf("%4d", b[k]);
return(0);
}

```

9. 190】

```

#include <stdio.h>
#include <ctype.h>
int main (int argc, char * argv[ ])
{
    char * str;
    int num=0;
    if(argc==2)
    {
        str=argv[1];
        while(*str)
            if(isalpha(*str++)) num++;
        printf ("\nThe count is :%d.\n", num);
    }
    return 0;
}

```

10.87】试利用指向结构体的指针编制一个程序,实现输入3个学生的学号、数学期中和期末成绩,然后计算其平均成绩并输出成绩表。

10.88】请编程序建立一个带有头结点的单向链表,链表结点中的数据通过键盘输入,当输入数据为-1时,表示输入结束。(链表头结点的数据域不放数据,表空的条件是 $ph->next==NULL$ 。)

10.89】已知 head 指向双向链表的第一个结点。链表中每个结点包含数据域(info)、后继元素指针域(next)和前趋元素指针域(pre)。请编写函数 print1 用来从头至尾输出这一双向链表。

10.90】已知 head 指向一个带头结点的单向链表,链表中每个结点包含字符型数据域(data)和指针域(next)。请编写函数实现在值为 a 的结点前插入值为 key 的结点,若没有值为 a 的结点,则插在链表最后。

10.91】已知 head 指向一个带头结点的单向链表,链表中每个结点包含数据域(data)和指针域(next)。请编写函数实现如图 10-8 所示链表的逆置。
若原链表为:

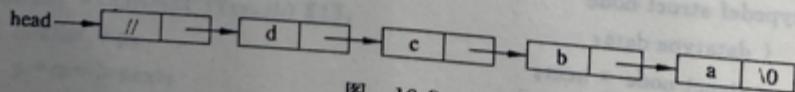
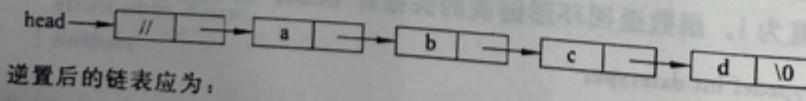


图 10-8

86】 # include <stdio. h>

```
int main( )
{
    struct study
    {
        int mid;
        int end;
        int average;
    } math;
    scanf("%d %d", &math. mid, &math. end);
    math. average=(math. mid+math. end)/2;
    printf("average=%d\n", math. average);
    return 0;
}
```

87】 # include <stdio. h>

```
struct stu
{
    int num;
    int mid;
    int end;
```

```

int ave;
} s[3];
int main()
{ int i;
  struct stu * p;
  for(p=s;p<s+3;p++)
  { scanf("%d %d %d", &(p->num), &(p->mid), &(p->end));
    p->ave=(p->mid+p->end)/2;
  }
  for(p=s;p<s+3;p++)
  printf("%d %d %d %d\n", p->num, p->mid, p->end, p->ave);
  return 0;
}

```

2. 分析以下算法的时间复杂度。

```

void fun(int n)
{
  int i, x = 0;
  for (i = 1; i < n; i++)
    for (j = i + 1; j <= n; j++)
      x++;
}

```

答：设基本运算 $x++$ 的执行次数为 $T(n)$ ，则有以下关系。

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n 1 = \sum_{i=1}^{n-1} (n-i) = n(n-1)/2 = O(n^2)$$

该算法的时间复杂度为 $O(n^2)$ 。

3. 分析以下算法的时间复杂度。

```

void fun(int n)
{
  int s = 0, i, j, k;
  for (i = 0; i <= n; i++)
    for (j = 0; j <= i; j++)
      for (k = 0; k < j; k++)
        s++;
}

```

答：该算法的基本运算是语句 $s++$ ，其频度如下。

时间复杂度表示为 $O(n^2)$ 。

4. 设 n 是偶数, 试计算执行以下算法后 m 的值, 并给出其时间复杂度。

```
void fun(int n)
{
    int m = 0, i, j;
    for (i = 1; i <= n; i++)
        for (j = 2 * i; j <= n; j++)
            m++;
}
```

答: 算法的基本运算为 $m++$, 由于内循环为 $2 * i \sim n$, 即 i 的最大值满足 $2i \leq n, i \leq n/2$, 所以基本运算的频度如下。

$$T(n) = \sum_{i=1}^{n/2} \sum_{j=2i}^{n/2} 1 = \sum_{i=1}^{n/2} (n - 2i + 1) = n \times \frac{n}{2} - 2 \sum_{i=1}^{n/2} i + \frac{n}{2} = \frac{n^2}{4}$$

而 m 是从 0 开始的, 所以算法执行后 $m = n^2/4$ 。该算法的时间复杂度为 $O(n^2)$ 。

5. 设 n 为正整数, 分析以下算法的时间复杂度。

```
    return false;
}
```

11. 【单链表算法】设计一个算法判定单链表 L (带头结点)是否为递增的。
解：从链表 L 的第 2 个结点开始，判定每个结点的值是否比其前驱结点的值大。若有一个不成立，则整个链表便不是递增的，否则是递增的。对应的算法如下：

```
bool increase(LinkNode * L)
{
    LinkNode * p = L->next, * post;
    post = p->next;
    while (post != NULL)
    {
        if (post->data > p->data)           //p 指向首结点
        {
            p = post;                      //post 指向 p 结点的后继结点
            post = post->next;             //若正序则继续判断下一个结点
            //p,post 同步后移
        }
        else
            return false;
    }
    return true;
}
```

12. 【单链表算法】假定采用带头结点的单链表保存单词，当两个单词有相同的后缀时可共享相同的后缀存储空间，例如“loading”和“being”，如图 2.3 所示。设 $str1$ 和 $str2$ 分别指向两个单词所在单链表的头结点，链表结点结构为 $(data, next)$ 。请设计一个时间上尽可能高效的算法，找出由 $str1$ 和 $str2$ 所指向两个链表共同后缀的起始位置(如图中字符‘i’所在结点的位置 p)。

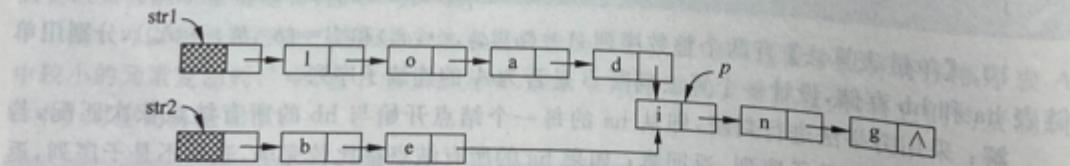


图 2.3 两个单词的后缀共享存储结构

- 解：分别求出 $str1$ 和 $str2$ 所指的两个链表的长度 m 和 n 。将两个链表以表尾对齐，令指针 p, q 分别指向 $str1$ 和 $str2$ 的头结点，若 $m \geq n$ ，则让 p 指向 $str1$ 链表中的第 $m-n+1$ 个结点；若 $m < n$ ，则让 q 指向 $str2$ 链表中的第 $n-m+1$ 个结点，即让指针 p, q 所指结点到表尾的长度相符。反复将指针 p, q 同步后移，并判断它们是否指向同一结点。若 p, q 指向同一结点，则该结点即为所求的共同后缀的起始位置。对应的算法如下：

```
LinkNode * Findlist(LinkNode * str1, LinkNode * str2)
{
    int m, n;
    LinkNode * p, * q;
```

```

m = ListLength(str1);           //求单链表 str1 的长度 m
n = ListLength(str2);           //求单链表 str2 的长度 n
for (p = str1; m > n; m--)
    p = p->next;
for (q = str2; m < n; n--)
    q = q->next;
while (p->next != NULL && p->next != q->next)
{   p = p->next;           //p,q 两步后移找第一个指针值相等的结点
    q = q->next;
}
return p->next;
}

```

13. 【单链表算法】某非空单链表 L 中的所有元素为整数,设计一个算法将所有小于零的结点移到所有大于等于零的结点的前面。

解: 用 p 指针扫描单链表 L , pre 指向其前驱结点。当 p 不空时循环,若 p 所指结点的数据值小于 0,通过 pre 指针将其从链表中移去,然后将 p 结点插入到表头,并置 $p=pre->next$;否则, pre,p 同步后移一个结点。对应的算法如下:

```

void Move(LinkNode * &L)
{
    LinkNode * p = L->next, * pre = p;
    while (p != NULL)
    {
        if (p->data < 0)           //将结点 p 从链表中移去
        {
            pre->next = p->next;   //将结点 p 插入到表头
            p->next = L->next;
            L->next = p;
            p = pre->next;
        }
        else                        //pre,p 同步后移
        {
            pre = p;
            p = p->next;
        }
    }
}

```

14. 【单链表算法】设计一个算法删除带头结点的非空单链表 L 中第一个值为 x 的结点(值为 x 的结点可能有多个),成功操作返回 true,否则返回 false。

解: 用 pre,p 遍历整个单链表, pre 指向结点 p 的前驱结点, p 用于查找第一个值为 x 的结点,当找到后通过 pre 结点将 p 结点删除,返回真,否则返回假。对应的算法如下:

```

bool DelFirstx(LinkNode * &L, ELEMTYPE x)
{
    LinkNode * pre = L, * p = pre->next;           //pre 指向结点 p 的前驱结点
    while (p != NULL && p->data != x)
    {
        pre = p;
        p = p->next;           //pre,p 同步后移
    }
    if (p == NULL)           //找到值为 x 的 p 结点

```

```
{     pre->next = p->next;
    free(p);
    return true;
}
else return false;                                //未找到值为 x 的结点
}
```

本算法的时间复杂度为 $O(n)$ 。

22.【双链表算法】有一个非空双链表 L , 设计一个算法在第 i 个结点之后插入一个值为 x 的结点。

解: 先在 L 中查找第 i 个结点 p , 若不存在这样的结点返回 false; 否则新建一个值为 x 的结点 s , 在结点 p 之后插入 s 结点, 返回 true。对应的算法如下:

```
bool InsertAfteri(DLinkNode * &L, int i, ElemenType x)
{
    DLinkNode * p = L->next, * s;
    int j = 1;
    if (i < 1) return false;
    while (p != NULL && j < i)           //查找第 i 个结点 p
    {
        j++;
        p = p->next;
    }
}
```

//没有找到第 i 个结点返回假

```

if (p == NULL)
    return false;
s = (DLinkNode *)malloc(sizeof(DLinkNode));
//在 p 结点之后插入新结点
s->data = x;
if (p->next != NULL) p->next->prior = s;
s->next = p->next;
p->next = s;
s->prior = p;
return true;
}

```

6. 23. 【双链表算法】有一个非空双链表 L, 设计一个算法删除所有值为 x 的结点。

解: 先让 p 指向首结点。p 不为空时循环: 若 p 所指结点值为 x, 删去 p 结点, 让 p 指向下一个结点; 否则 p 后移一个结点。对应的算法如下:

```

void DeleteAllx(DLinkNode * &L, ElemtType x)
{
    DLinkNode * p = L->next, * q;
    while (p != NULL)
    {
        if (p->data == x)
        {
            q = p->next;           //临时保存 p 结点的后继结点
            p->prior->next = p->next;
            if (p->next != NULL)
                p->next->prior = p->prior;
            p = q;
        }
        else p = p->next;
    }
}

```

24. 【双链表算法】有一个带头结点的双链表 L, 其所有元素均为整数, 设计一个算法删除所有奇数元素的结点。

解: 先让 p 指向首结点。p 不为空时循环: 若 p 所指结点值为奇数, 删去 p 结点, 置 p 为 q; 否则 p 后移一个结点。对应的算法如下:

```

void DelOdd(DLinkNode * &L)
{
    DLinkNode * p = L->next, * q;
    while (p != NULL)
    {
        if (p->data % 2 == 1)           //p 指向奇数结点
        {
            q = p->next;
            p->prior->next = q;
            q->prior = p->prior;
            free(p);
            p = q;
        }
        else p = p->next;           //p 指向偶数结点
    }
}

```

27. 【循环双链表算法】有一个非空循环双链表 L , 设计一个算法删除第一个值为 x 的结点。

解: 让 p 指向首结点。 p 不为 L 且 p 结点值不为 x 时循环: p 后移一个结点。循环结束时, 若 p 为 L 表示未找到值为 x 的结点, 返回 false, 否则删除 p 结点并返回 true。对应的

59

算法如下:

```
bool DeleteFirstx(DLinkNode * &L, ElemType x)
{
    DLinkNode * p = L -> next;
    while (p != L && p -> data != x)
        p = p -> next;
    if (p == L) //未找到值为 x 的结点返回 false
        return false;
    else
    {
        p -> prior -> next = p -> next; //删除 p 结点
        p -> next -> prior = p -> prior;
        free(p);
        return true;
    }
}
```

13. 【二叉链存储结构+先序递归遍历算法】假设二叉树中每个结点值为单个字符, 采用二叉链存储结构存储。设计一个算法求二叉树 b 的最小枝长。所谓最小枝长是指根结点到最近叶子结点的路径长度。

解: 采用基于先序遍历的递归方法, 先判断根结点不空, 再求出左、右子树的最小枝长 \min_1 和 \min_2 , 并返回 $\min(\min_1, \min_2) + 1$ 。对应的算法如下:

```

int MinBranch(BTNode * b)
{
    int min1, min2, min;
    if (b == NULL)
        return 0;
    else
    {
        min1 = MinBranch(b->lchild);
        min2 = MinBranch(b->rchild);
        if (min1 < min2) min = min1 + 1;
        else min = min2 + 1;
        return min;
    }
}

```

14. 【二叉链存储结构+先序递归遍历算法】假设二叉树中每个结点值为单个字符,采用二叉链存储结构存储。设计一个算法,求二叉树 b 中第 k 层上的叶子结点个数。

解: 采用基于先序遍历的递归方法, num 置初值 0, 若当前访问结点 b 是第 k 层上的叶子结点, 则 num++ , 再求出左、右子树第 k 层上的叶子结点个数 min1 和 min2, 最后返回 $\text{min1} + \text{min2}$ 。对应的算法如下:

```

int LevelLeafCount(BTNode * b, int h, int k)
{
    //h 的初值为 1
    int num1, num2, num = 0;
    if (b != NULL)
    {
        if (h == k && b->lchild == NULL && b->rchild == NULL)
            num++;
        num1 = LevelLeafCount(b->lchild, h + 1, k);
        num2 = LevelLeafCount(b->rchild, h + 1, k);
        num += num1 + num2;
        return num;
    }
    else return 0;
}
int LevelLeafk(BTNode * b, int k)           //求 b 中第 k 层上的叶子结点个数
{
    return LevelLeafCount(b, 1, k);
}

```

16. 【二叉链存储结构+后序递归遍历算法】假设二叉树中每个结点值为单个字符,采用二叉链存储结构存储。试设计一个算法,采用后序遍历方式求一棵给定二叉树 b 中的所有小于 x 的结点个数。

解: 对应的算法如下。

```
int LessNodes(BTNode * b, char x)
{
    int num1, num2, num = 0;
    if (b == NULL)
        return 0;
    else
    {
        num1 = LessNodes(b->lchild, x);
        num2 = LessNodes(b->rchild, x);
        num += num1 + num2;
        if (b->data < x) num++;
        return num;
    }
}
```

8. 【二叉链存储结构+先序递归遍历算法】假设二叉树采用二叉链存储结构存储,试设计一个算法,输出从每个叶子结点到根结点的逆路径。

解: 采用基于先序遍历的递归方法,用 path 数组存放查找的路径, pathlen 存放路径长度,当找到叶子结点 b 时,由于 b 叶子结点尚未添加到 path 中,因此在输出路径时还需输出 $b->data$ 值;若 b 不为叶子结点,将 $b->data$ 放入 path 中,然后在左、右子树中递归查找。递归算法如下:

```
void AllPath1(BTNode * b, ElemType path[], int pathlen)
{
    //b 为根结点时, pathlen 初始值为 0
    int i;
    if (b != NULL)
    {
        if (b->lchild == NULL && b->rchild == NULL)           //b 为叶子结点
        {
            printf(" %c 到根结点逆路径: %c ", b->data, b->data);
            for (i = pathlen - 1; i >= 0; i--)
                printf(" %c ", path[i]);
            printf("\n");
        }
    }
}
```

```
else
{
    path[pathlen] = b->data;           //将当前结点放入路径中
    pathlen++;
    AllPath1(b->lchild, path, pathlen); //递归扫描左子树
    AllPath1(b->rchild, path, pathlen); //递归扫描右子树
    pathlen--;
}
```

9. 【二叉链存储结构+先序递归遍历算法】假设二叉树采用二叉链存储结构存储,设