

C语言与程序设计

The C Programming Language



第4章 流程控制

华中科技大学计算机学院

黄宏



主要内容

if语句

switch语句

while语句

for语句

do-while语句

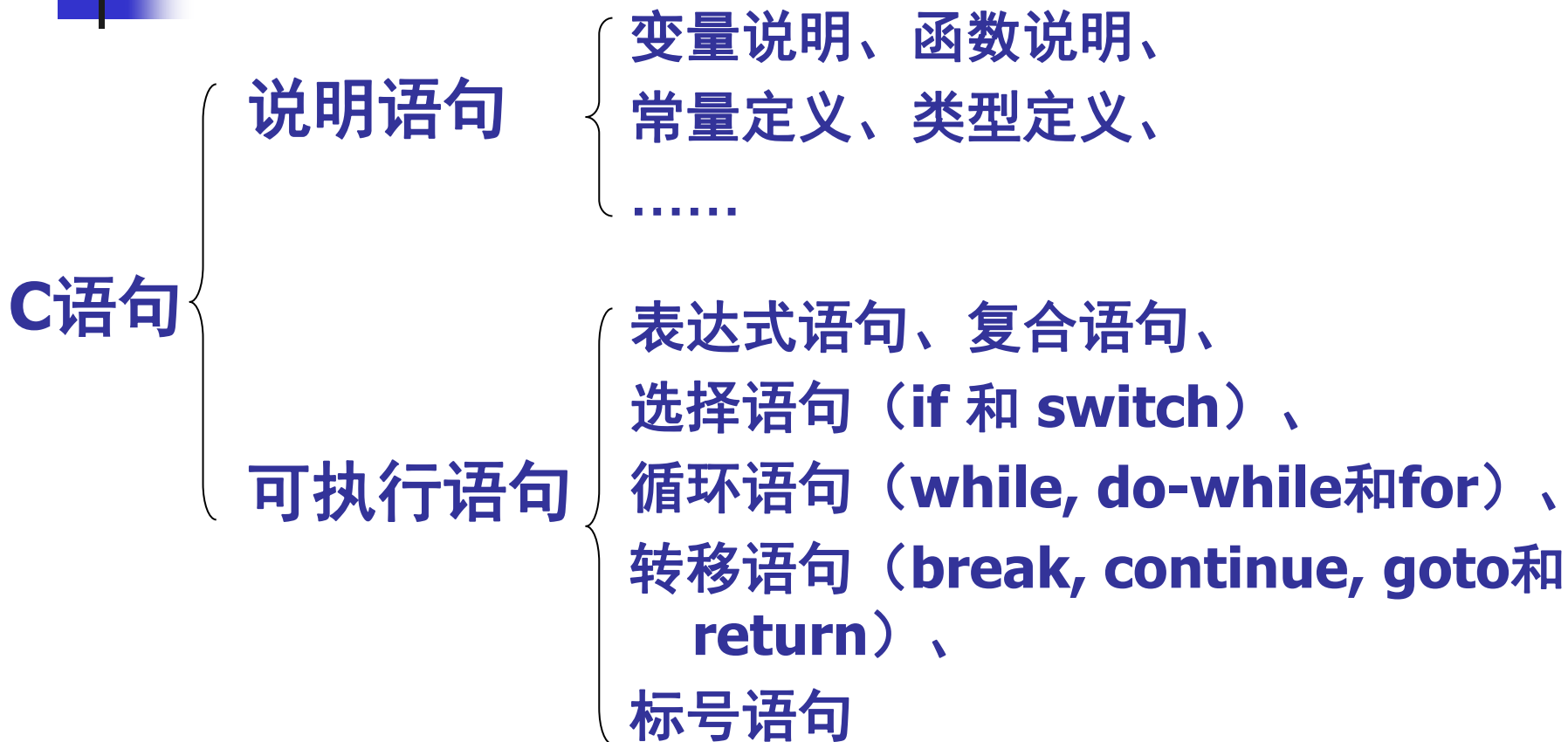
goto语句和标号语句

break语句

continue语句

常用算法：枚举、筛法、递推

4.1 C语句分类





4.2 表达式语句

表达式;

例: $x = y + 1$

$x = y + 1$;

`printf("hello")`

`printf("hello");`

;
/* 空语句 */



4.3 复合语句

{

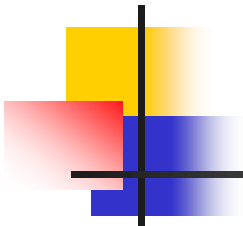
说明部分

语句部分

}

复合语句又称块。函数体是一个块。

注意变量的作用域。



x=2,y=2
x=1,y=2

函数体

```
int main(void)
{
    int x=1,y=2;
    {
        int x=2;
        printf ("x=%d, y=%d\n", x,y);
    }
    printf ("x=%d, y=%d\n", x,y);
    return 0;
}
```

复合语句



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

```
void main( void )
```

```
{
```

```
    int x=5,y=1,t;
```

```
    if(x>y) {
```

```
        t=x;
```

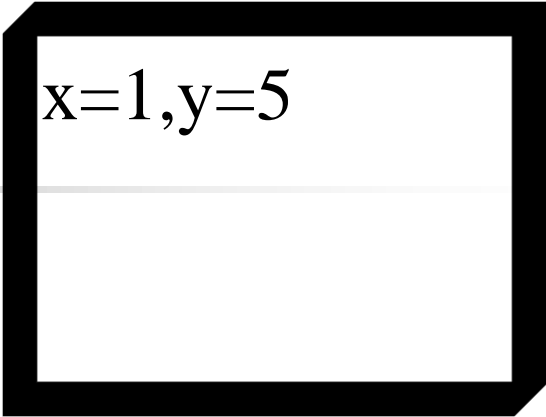
```
        x=y;
```

```
        y=t;
```

```
    }
```

```
    printf ("x=%d,y=%d\n", x,y);
```

```
}
```



x=1,y=5

Ex4_3. c



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

```
void main( void )
```

```
{
```

```
    int x=5,y=1;
```

```
    if(x>y) {
```

```
        int t;
```

```
        t=x;
```

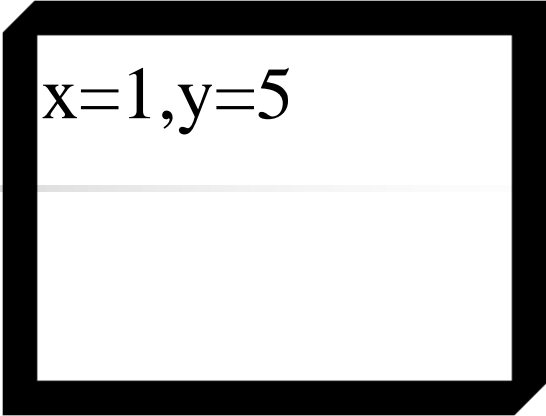
```
        x=y;
```

```
        y=t;
```

```
    }
```

```
    printf ( “x=%d,y=%d\n”, x,y);
```

```
}
```



x=1,y=5

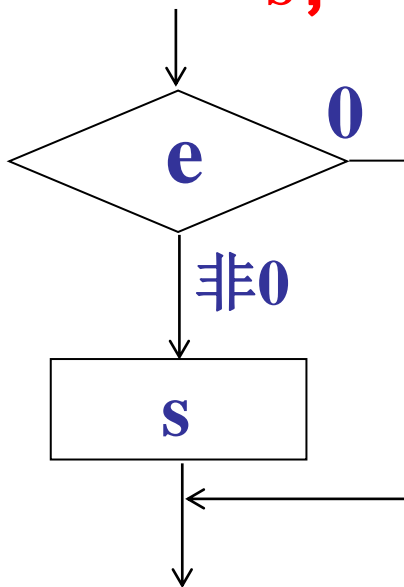
4.4 if语句

1. if 语句的一般形式

形式1：（单分支）

if (e)

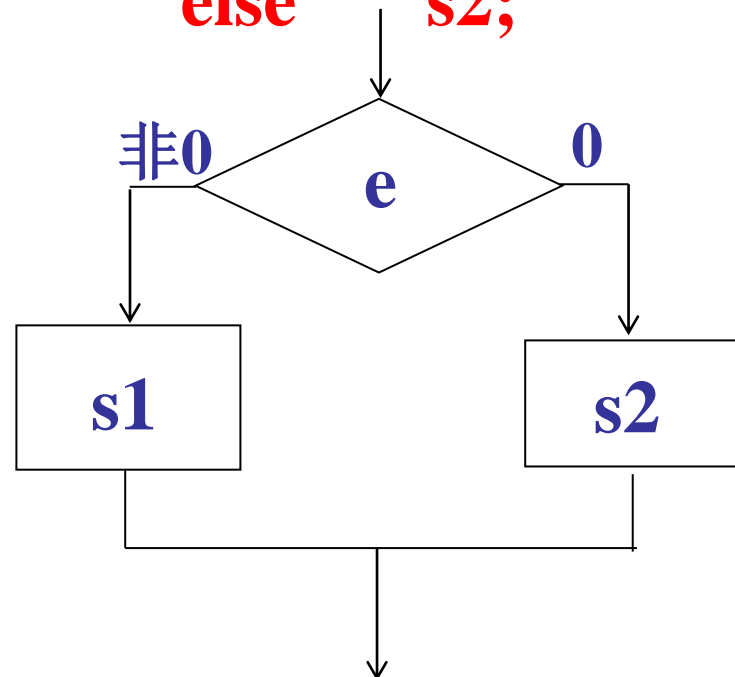
s;



形式2：（双分支）

if (e) s1;

else s2;



如何用 if 语句实现多分支结构？

//统计正数、负数、零的个数

positive=negative=zero=0;

for(k=1;k<11;k++) {

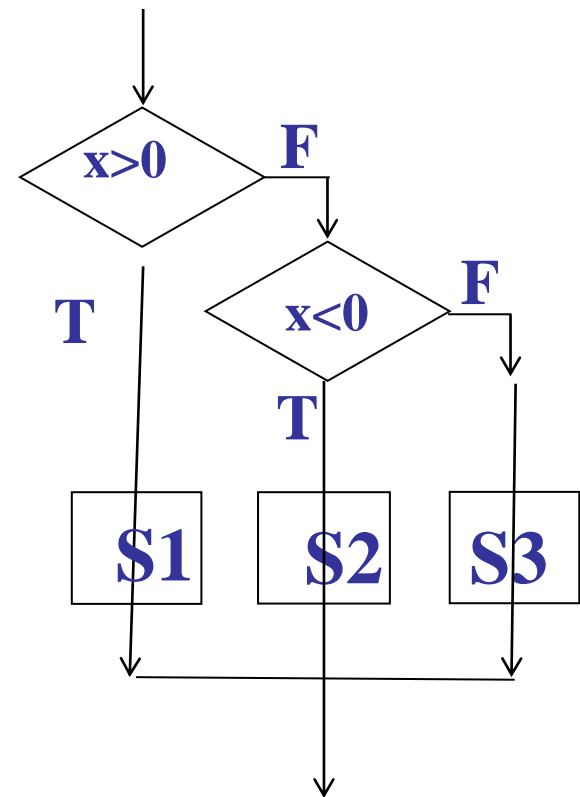
scanf("%d",&x);

if(x>0) positive++;

else if(x<0) negative++;

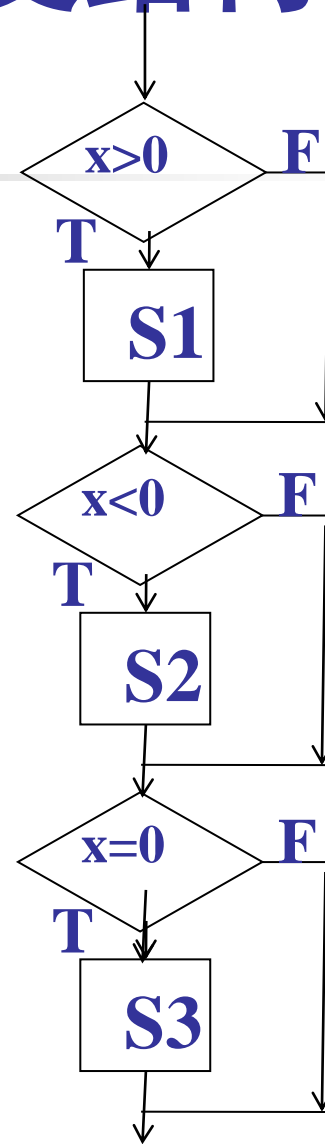
else zero++;

}

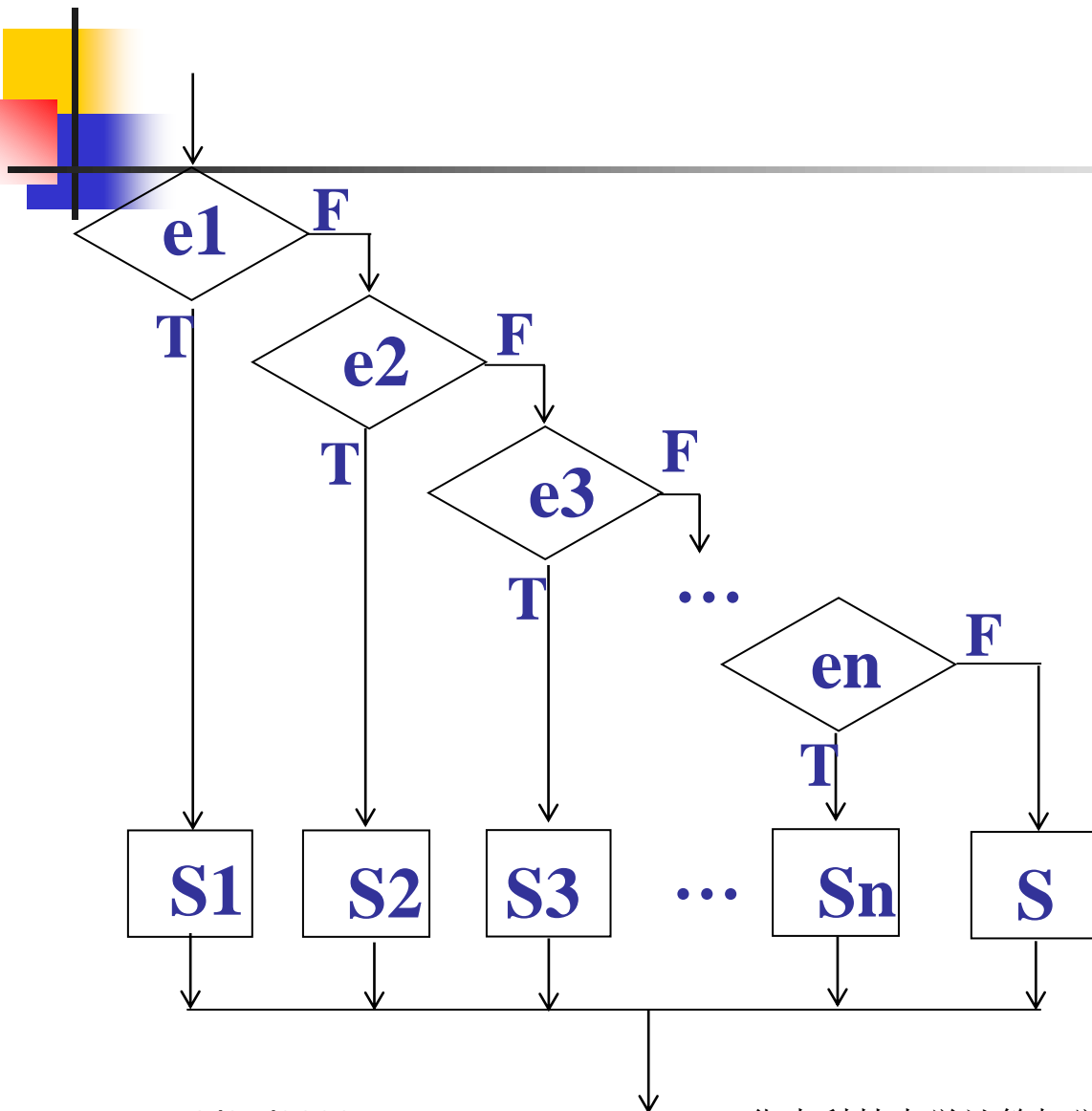


如何用 if 语句实现多分支结构？

```
//统计正数、负数、零的个数
positive=negative=zero=0;
for(k=1;k<11;k++) {
    scanf("%d",&x);
    if(x>0) positive++;
    if(x<0) negative++;
    if(x==0) zero++;
}
```



形式3: if-else-if结构 (多分支)



if(e1)
 S1
else if (e2)
 S2
else if (e3)
 S3
 ...
else if(en)
 Sn
else S

2. 嵌套的if语句

1) 一般形式

if (e)

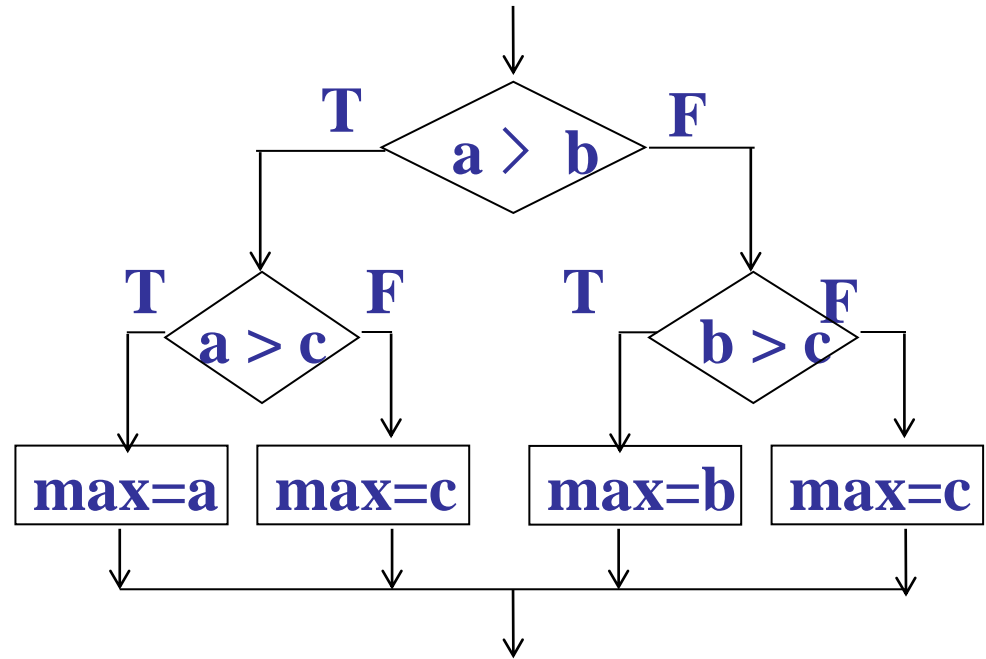
if 语句

else

if 语句

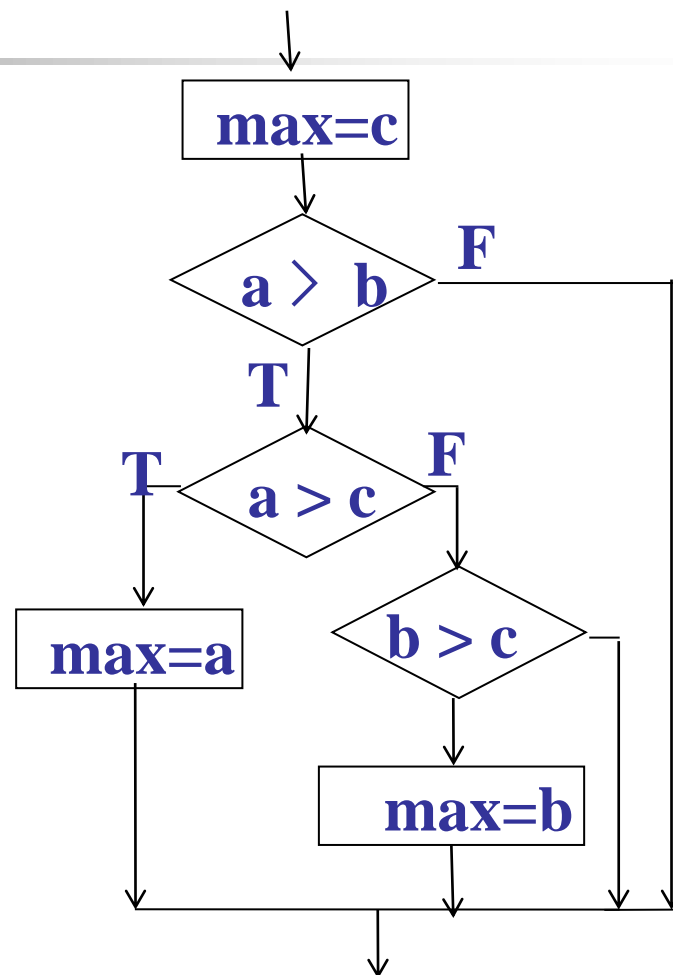
用嵌套的if语句求a, b, c三个数中最大值。

```
if ( a > b )  
    if ( a > c ) max = a;  
    else max = c;  
else  
    if ( b > c ) max = b;  
    else max = c;
```



下面代码能实现求a, b, c中最大值吗?

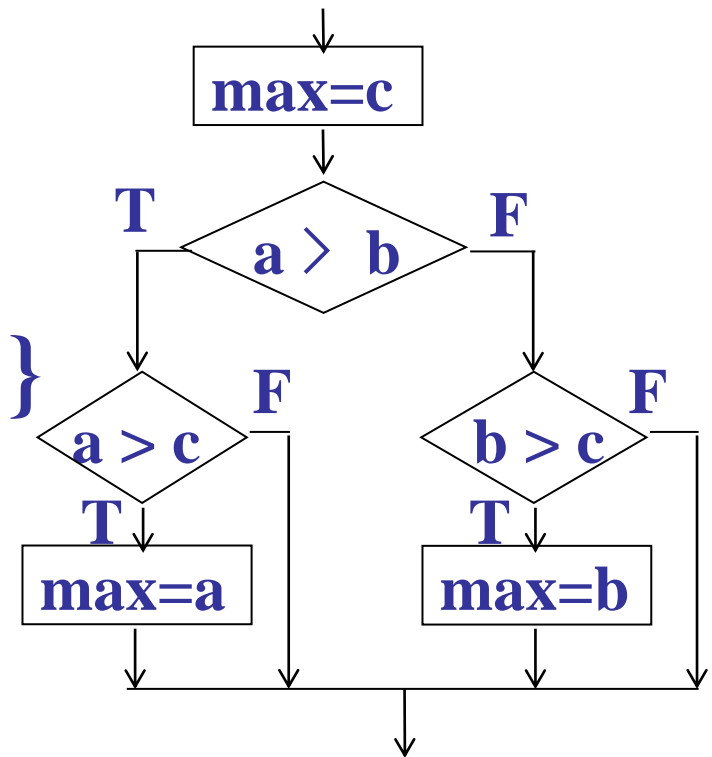
```
max = c;  
if ( a > b )  
    if ( a > c ) max = a;  
else  
    if ( b > c ) max = b;
```



2) 嵌套if 语句中else的配对规则

else与其同一程序块中最靠近的还未配对的**if**配对。

```
max = c;  
if ( a > b )  
    { if ( a > c ) max = a; }  
else  
    if ( b > c ) max = b;
```





举例：输入百分制成绩，输出英文等级。

分数范围

$90 \leq x \leq 100$

$80 \leq x < 90$

$60 \leq x < 80$

$x < 60$

等级英文名

excellent（优）

good（良）

middle（中）

bad（差）

方案一：用多路选择结构 if-else-if

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

```
int main (void)
```

```
{
```

```
    float x;
```

```
    printf("input the score : ");
```

```
    scanf("%f", &x);           // 输入double: %lf
```

```
    if ( x > 100 || x < 0)
```

```
        printf("input error!\n");
```

```
    else {
```

```
        if (x>=90) printf(" excellent! \n");
```

```
        else if (x>=80) printf(" good! \n");
```

```
        else if (x>=60) printf(" middle! \n");
```

```
        else printf(" bad \n");
```

```
    }
```

```
    return 0;
```

```
}
```

可无

Ex4_4. c

方案二：用多条 if 语句

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

```
int main (void)
```

```
{
```

```
    float x;
```

```
    printf("input the score : ");
```

```
    scanf("%f", &x);
```

```
    if ( x > 100 || x < 0)
```

```
        printf("input error!\n");
```

```
    else {
```

```
        if (x>=90) printf(" excellent! \n");
```

```
        if (x<90&&x>=80) printf(" good! \n");
```

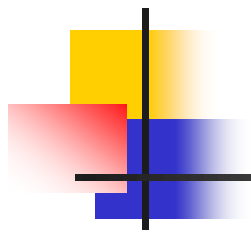
```
        if (x<80&&x>=60) printf(" middle! \n");
```

```
        if(x<60) printf(" bad \n");
```

```
    }
```

```
    return 0;
```

```
}10/25/2023
```



方案三：用*switch*语句

4.5 switch 语句——开关语句

功能:

一般形式:

```
switch (e) {  
    case 常量1: 语句序列1  
                break;  
    case 常量2: 语句序列2  
                break;  
    ...  
    case 常量n: 语句序列n  
                break;  
    default    : 语句序列  
                break;  
}
```

计算 e,

如 $e = \text{常量}i$, 则执行其后的语句,直到 **break** 语句止

若与所有常量值不相等,则从 **default** 后的语句开始执行

方案三：用switch语句

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

```
int main (void)
```

```
{
```

```
float x;
```

```
printf("input the score x( 0 <= x <= 100):\n");
```

```
scanf("%f ", &x);
```

```
if ( x > 100 || x < 0)   printf("input error!\n");
```

```
else
```

```
switch ((int)(x / 10))
```

```
{
```

```
case 10:
```

```
case 9: printf(" excellent! \n"); break;
```

```
case 8: printf(" good! \n"); break;
```

```
case 7:
```

```
case 6: printf(" middle! \n"); break;
```

```
default: printf(" bad! \n"); break;
```

```
}
```

```
return 0;
```

```
}  
10/25/2023
```

空的case语句，用于
几种情况合并执行一
组语句。

Ex4_5. c



注意：当多个case执行相同的语句时，这些case可以写成一行。

case 7: case 6: printf(" middle! \n"); break;

case 7, case 6: printf(" middle! \n"); break; ❌

case 7, 6: printf(" middle! \n"); break; ❌



下面语句执行时的输出？

```
i = 1;
```

```
switch ( i ){
```

```
    case 0: printf("%d\t", i);
```

```
    case 1: printf("%d\t", i++);
```

```
    case 2: printf("%d\t", i++);
```

```
    case 3: printf("%d", i++);
```

```
    default: printf("\n");
```

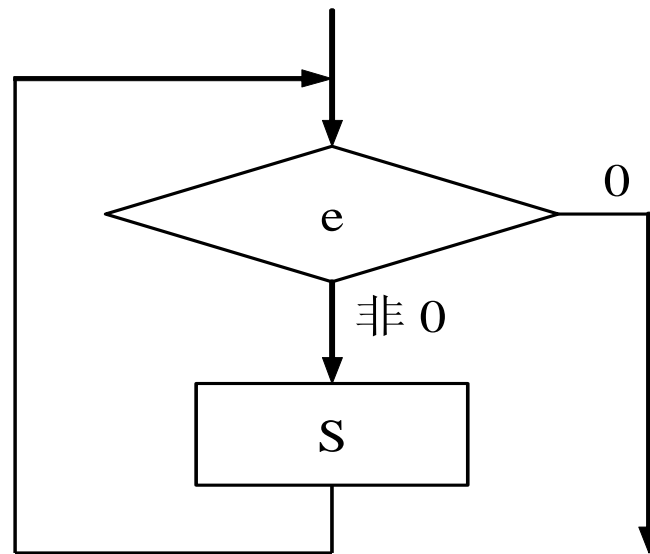
```
}
```

```
printf("%d\n", i);
```

1	2	3
4		

4.6 while语句

while (e)
S





```
i = 1 ; sum=0;
```

```
while ( i<=100) {
```

```
    sum+=i;
```

```
    i++;
```

```
}
```

```
printf(“sum=%d\n”,sum);
```



举例

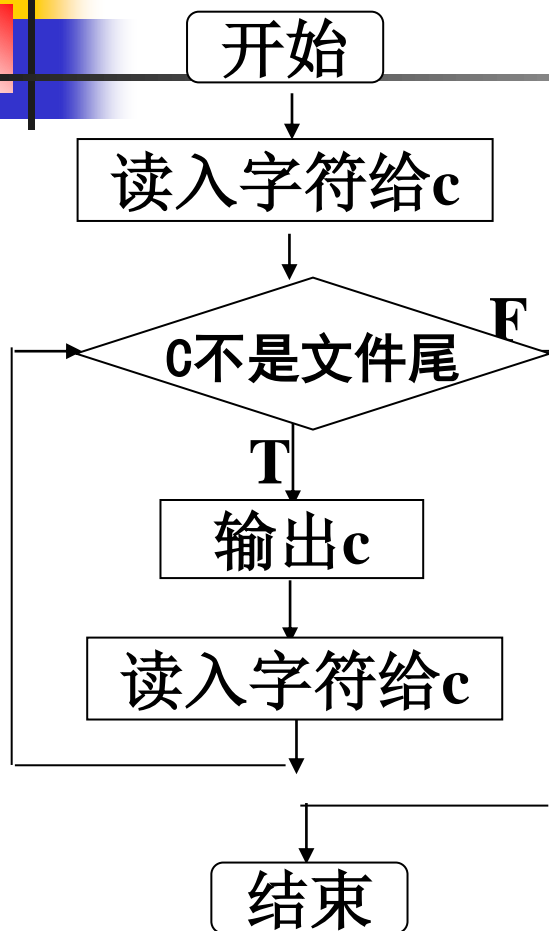
将来自标准输入文件的字符复制

到标准输出文件

键盘

显示器

算法



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

```
int main( void )
```

```
{
```

```
char c;
```

```
printf( “ Input characters:\n” );
```

```
c=getchar( );
```

```
while(c != EOF) {
```

```
    putchar(c) ;
```

```
    c=getchar( );
```

```
}
```

```
return 0;
```

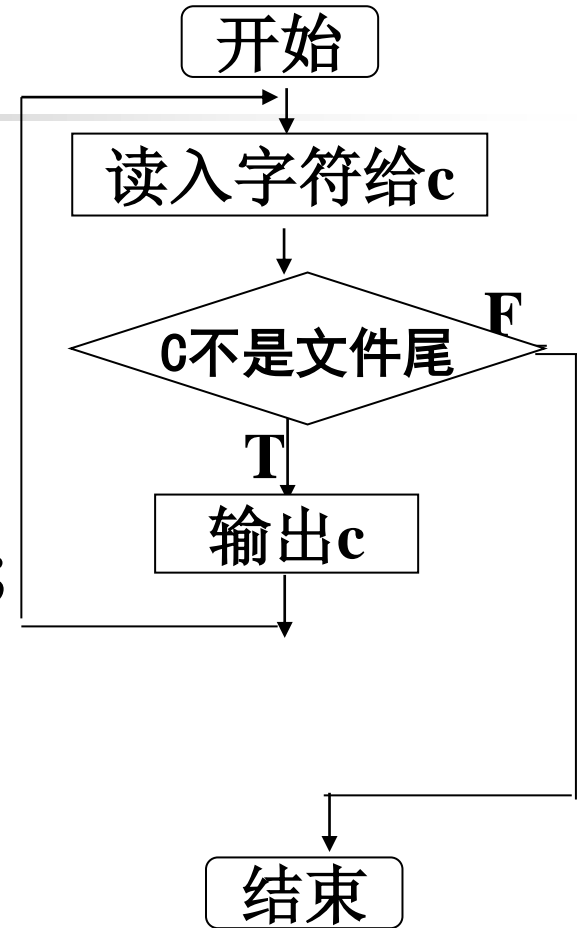
```
}
```

Ex4_10.c

算法

将程序简化为：

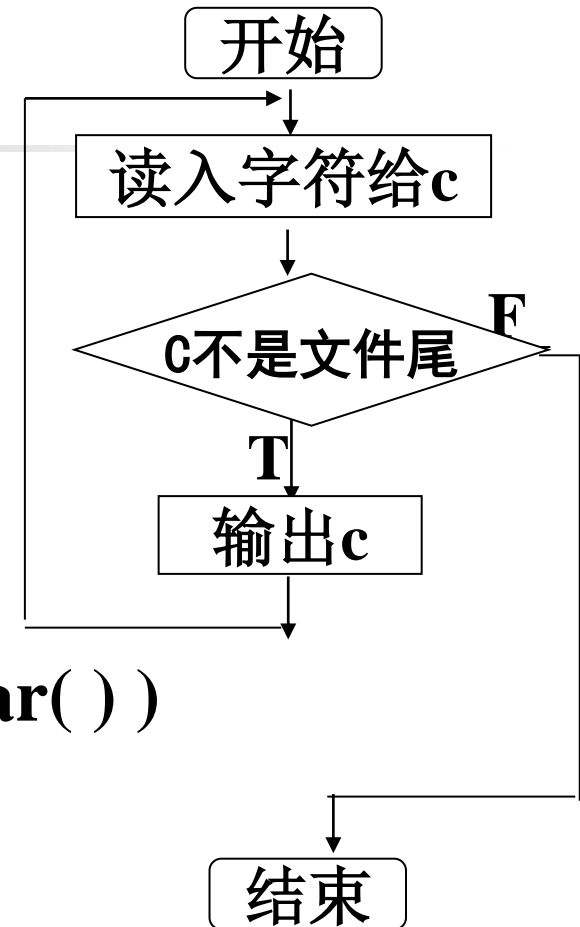
```
#include<stdio.h>
int main(void)
{
    char c;
    printf( “ Input characters :\n” );
    while( (c=getchar( )) !=EOF)
        putchar(c);
    return 0;
}
```



算法

将程序简化为：

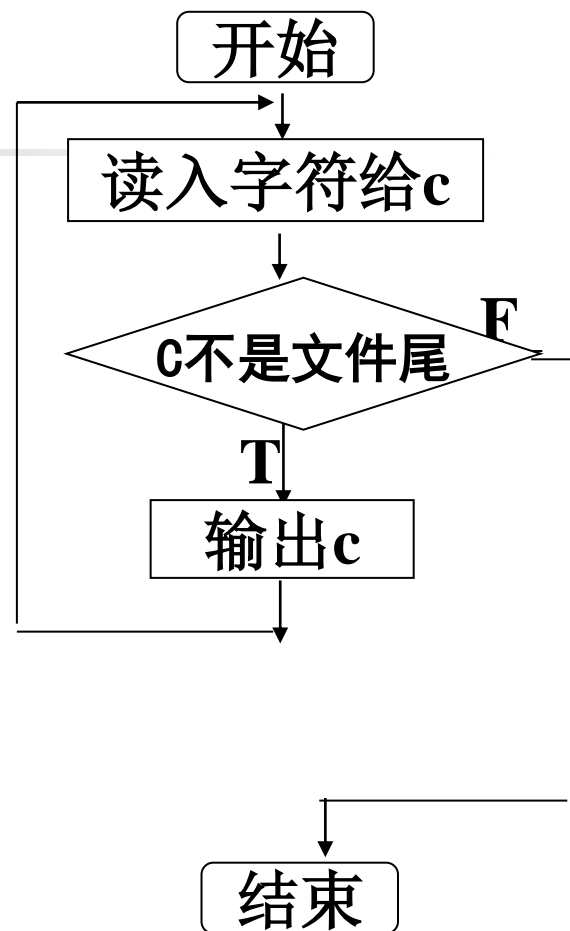
```
#include<stdio.h>
int main(void)
{
    char c;
    printf( “ Input characters :\n” );
    for( c=getchar( );c !=EOF; c=getchar( ) )
        putchar(c);
    return 0;
}
```



将程序简化为：

```
#include<stdio.h>
int main(void)
{
    char c;
    printf( “ Input characters :\n” );
    for( ; (c=getchar( )) !=EOF; )
        putchar(c);
    return 0;
}
```

算法





例：将输入正文每一行的前导空格删除后，将后面部分进行输出

分析：用有限状态机（FSM—Finite State Machine）方法。
状态机的4个要素：

一句话描述：当系统处于某状态时，如果发生了什么事件，就执行某动作，然后系统变成新状态

```
while ( 事件 )  
    switch (当前状态) {  
        case 状态i:  
            if (某条件/事件)  
                执行动作, 迁移至状态j  
            break;  
        case 状态j:  
            .....  
            .....  
    }
```




使用状态机解决问题，主要有两个步骤：

(1) 确定系统总共有几个状态

(2) 确定状态之间的迁移过程

状态迁移图：是一种描述系统的状态、以及相互转化关系的图形方式

```
while ((ch = getchar()) != EOF)
```

```
{
```

```
    switch(当前状态) {
```

```
        case 前导空格:
```

```
            if (ch不为空白符)
```

```
                输出ch;
```

```
                状态迁移至“拷贝态”
```

```
            break;
```

```
        case 拷贝 :
```

```
            输出ch;
```

```
            if (ch为换行符)
```

```
                状态迁移至“前导空格态”
```

```
            break;
```

```
    }
```

```
}
```

```
enum { HeadSpace, COPY };
int main()
{
    int ch, state;      /* state存当前状态 */

    state=HeadSpace;    // 初始前导空格状态
    printf("Input text,    end of Ctrl+z\n") ;
    while ((ch = getchar()) != EOF)
    {
        switch(state) {
            case HeadSpace:
                if (ch!=' ' && ch!='\t')    /* 不为空格时 */
                { putchar(ch); state=COPY; }
                break;
            case COPY :
                putchar(ch);
                if (ch=='\n')    state=HeadSpace;
                break;
        }
    }
    return 0;
}
```

例4.11 输入一个C程序(一段正文)，按原来格式复制输出，复制过程中删去所有的注释。

分析：用有限状态机（FSM--Finite State Machine）方法。

状态机的4个要素：现态、条件(事件)、动作、次态

一句话描述：当系统处于某状态时，如果发生了什么事件，就执行某动作，然后系统变成新状态

while (事件)

switch (当前状态)

{ case 状态*i*: if (某条件/事件)

 执行动作，迁移至状态*j*

 break;

 case 状态*j*:

.....

}

例4.11 输入一个C程序(一段正文)，按原来格式复制输出，复制过程中删去所有的注释。

分析：用有限状态机（FSM--Finite State Machine）方法。

状态机的4个要素：

现态：当前所处的状态

条件（事件）：当一个条件被满足，将会触发一个动作，或者执行一次状态的迁移。

动作：条件满足后执行的动作，动作不是必需的，当条件满足后，也可以不执行任何动作，直接迁移到新状态。也可以仍旧保持原状态。

次态：条件满足后要迁往的新状态，“次态”一旦被激活，就转变成新的“现态”了。

一句话描述：当系统处于某**状态**时，如果发生了什么**事件**，就执行某**动作**，然后系统变成**新状态**

例4.11 输入一个C程序(一段正文)，按原来格式复制输出，复制过程中删去所有的注释(标准C的注释)。

为了删去C程序中所有的注释，关键在于如何区分注释部分和需要复制的部分。为此，可将复制过程划分为4种状态：

- 0--复制状态(COPY)
- 1--开始注释状态 (START)
- 2--注释状态 (COMMENT)
- 3--结束注释状态 (END);

初始状态为**COPY**。

Ex4_11.c

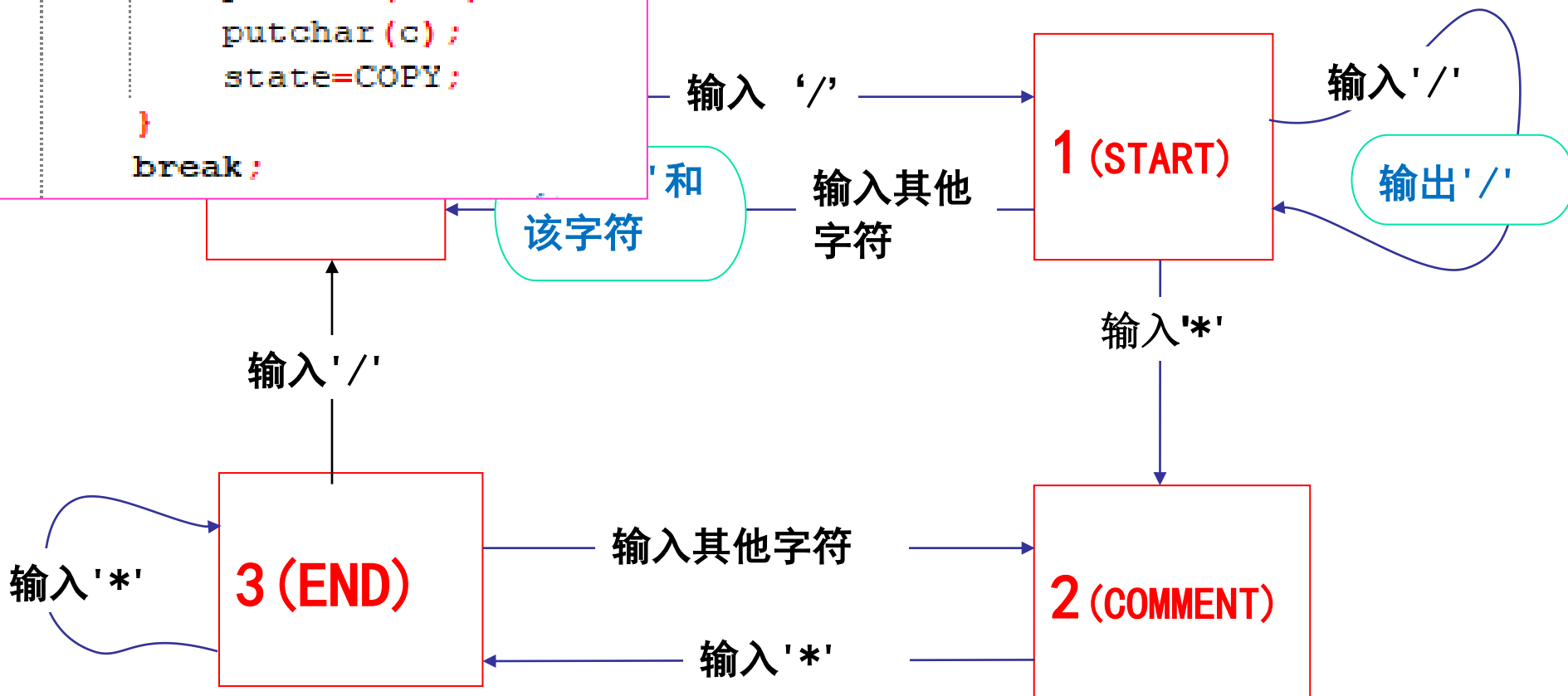
```

case START:      /* 在START处
    if (c == '*') /* 如果
        state = COMMENT;
    else if (c == '/') { /*
        putchar('/');
    }
    else {        /* 如果
        putchar('/');
        putchar(c);
        state=COPY;
    }
    break;

```

状态迁移图

ex4-11





自主练习

1、修改例4.11，使之能处理字符串常量。

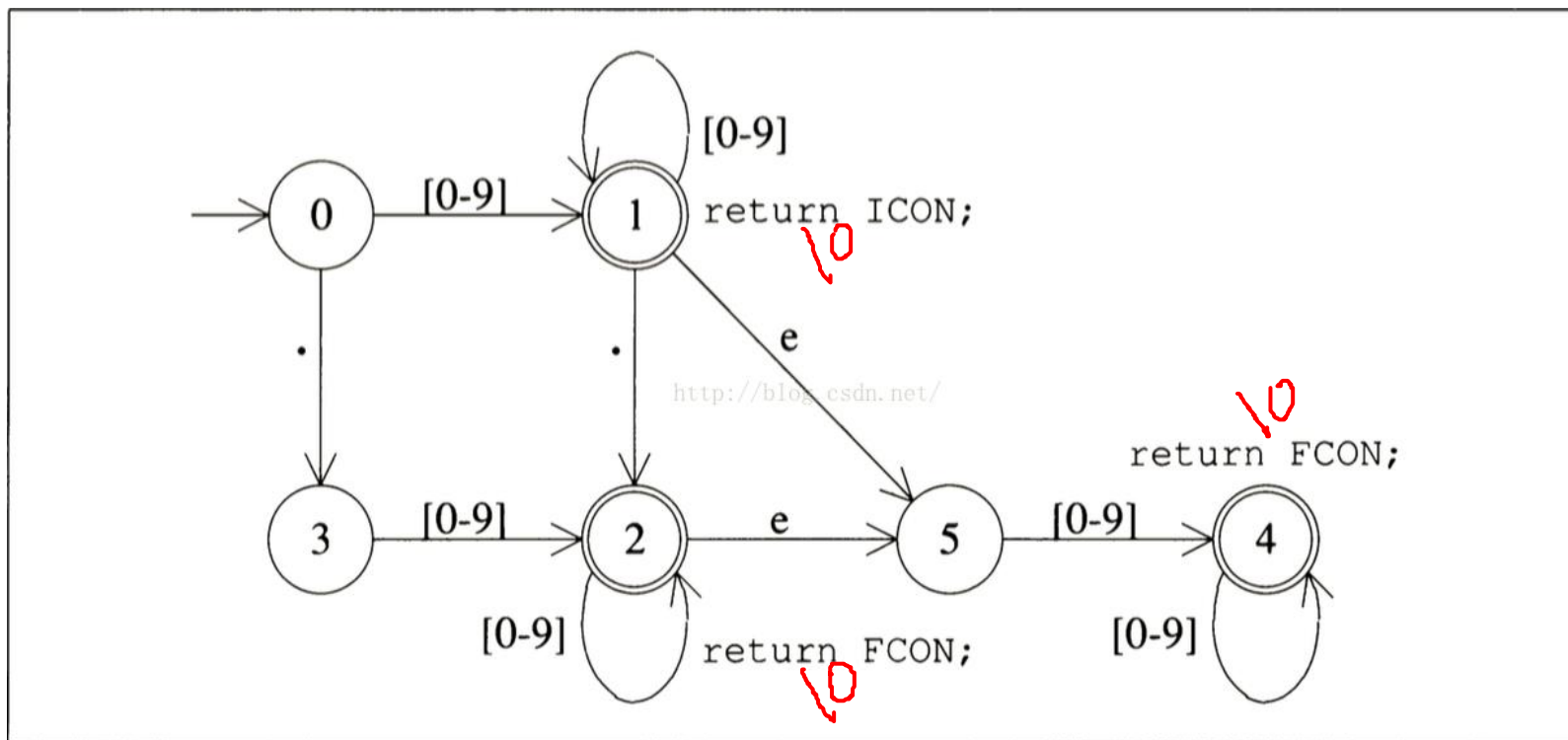
```
“/* comment in\” string */”
```

2、状态机统计单词个数

输入一段英文，统计单词数。单词仅包含大写字母和小写字母，单词之间用空白符或标点符号隔开。

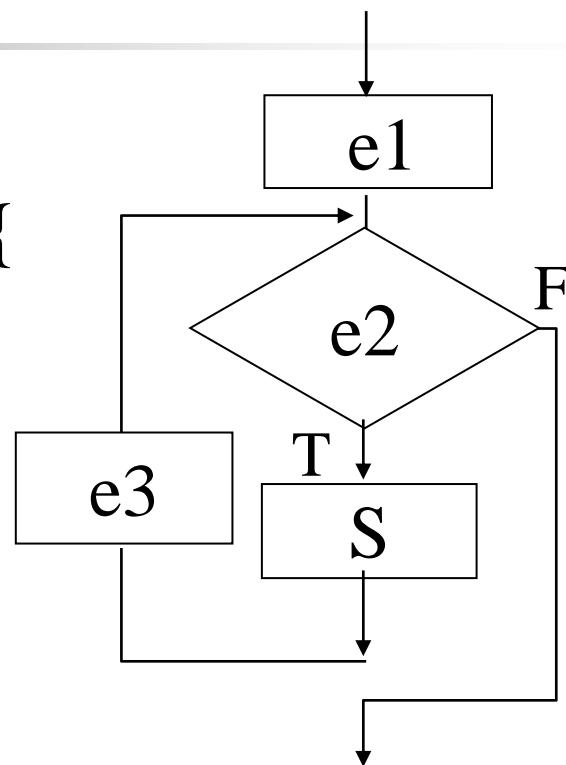
自主练习

3、整形/浮点型数值识别器：判断给定的字符串是否是十进制整数和实数（含科学记数法）表示的数值。



4.7 for 语句

for(e1; e2; e3) \leftrightarrow **while**(e2) {
 S
 e3;
}



for语句的几种特例

★ e2为空

$\text{for}(e1; ; e3) \ S \longleftrightarrow \text{for}(e1; 1 ; e3) \ s$

★ 只有e2

$\text{for}(; e2 ;) \ S \longleftrightarrow \text{while} (e2) \ s$

★ e1,e2,e3全为空

$\text{for}(; ;) \ S \longleftrightarrow \text{while}(1) \ s$

举例：输入一批正整数，以0为结束。输出其中最大的一个值。

```
#include<stdio.h>
void main(void)
{
    int x, max;
    printf("input a group integer end of 0: \n");
    scanf("%d", &x);
    max = x;
    for( ; x ; ) { /* x 等价于 x!=0 */
        scanf("%d", &x);
        if (max < x) max = x;
    }
    printf("max=%d\n", max);
}
```

ex4_15.c



例：求n!，n从终端输入。

```
/* 求n! */  
#include<stdio.h>  
int main(void)  
{ int n, i;  
  unsigned long fac; // unsigned long long fac; C99标准  
  printf("input n:\n");  
  scanf("%d", &n);  
  for (fac = 1, i = 1; i <= n; i++)  
    fac *= i;  
  printf("%d! = %lu\n", n, fac); // %llu 或者 %l64u  
}
```

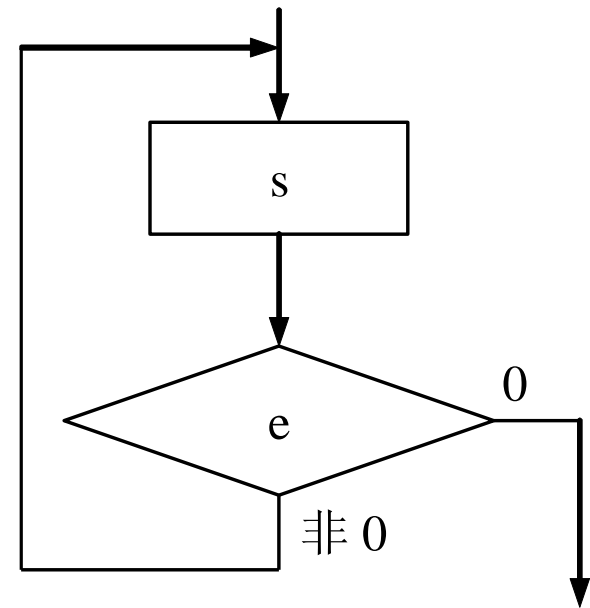
Ex4_16.c

4.8 do-while语句

do
S
while (e);

等价于

S
while (e)
S



举例:把输入的正整数按反方向输出。例如, 输入的数是12345, 要求输出结果是54321。

分析: 输入: 从高位到低位。

输出: 从低位到高位

关键: 如何获取一个整数 (x) 的个位数字。

算法: (1) $x \% 10$

(2) $x /= 10$, 不停循环语句直至 $x = 0$

Ex4_17.c

/*把输入的正整数按反方向输出*/

#include<stdio.h>

int main(void)

{

int x, digit;

printf("input an integer x>0:\n");

scanf("%d", &x);

do {

digit = x % 10; /* 求个位数字 */

printf("%d", digit);

x /= 10; /* 十位数字变个位数字 */

}while (x);

return 0;

}

/*把输入的整数按反方向输出*/

#include<stdio.h>

int main(void)

{

int x, digit;

printf("input an integer:\n");

scanf("%d", &x);

while (x) {

digit = x % 10; /* 求个位数字 */

printf("%d", digit);

x /= 10; /* 十位数字变个位数字 */

}

return 0;

}

4.9 goto语句和标号语句

goto 标号;

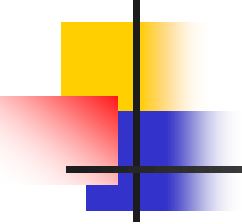
标号: 语句

功能: 无条件转向标号处。

用途: (1) 与if语句一起构成循环语句

```
loop: if(e) {  
    语句序列  
    goto loop;  
}
```

(2) 从循环体中跳转到循环体外



```
for(e1; e2; e3) {  
    .....  
    for(e1; e2; e3) {  
        .....  
        if(e) goto stop;  
    }  
    .....  
}  
stop: .....
```

为了使程序的结构化功能强，应尽量少用goto语句。

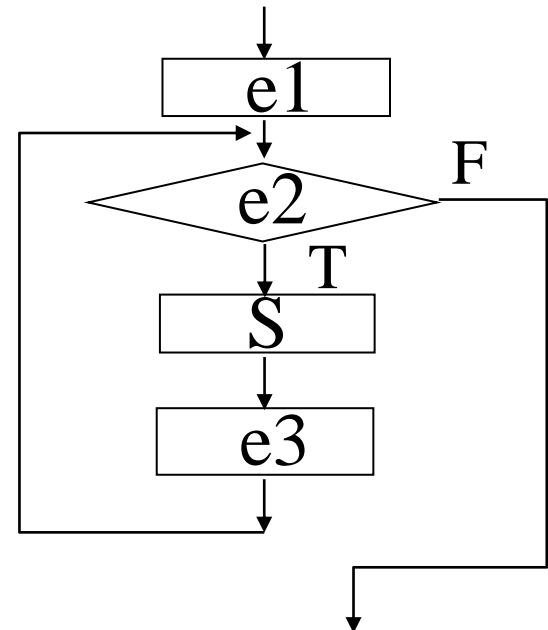
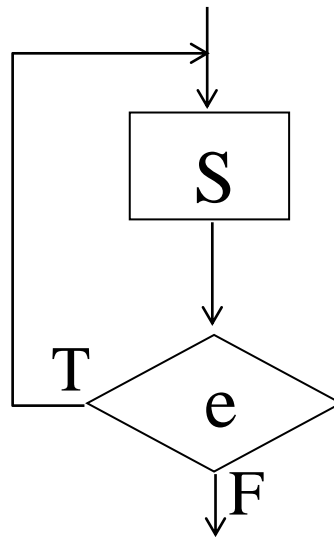
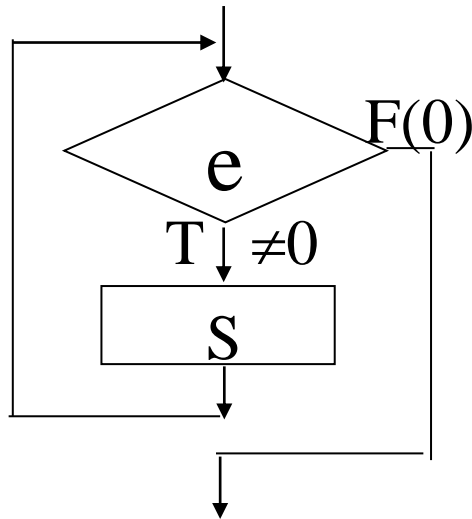
4. 10 break、continue语句

一、break语句

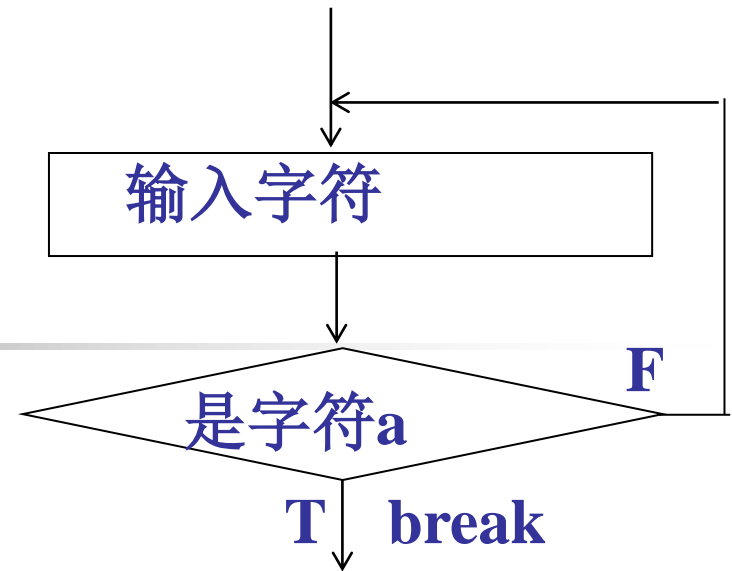
break;

用途:

- 在switch语句中，用于终止case子句的执行
- 用在循环体内部，强迫立即终止循环，跳过循环条件检测



```
for(;;) {
    ch = getchar ( ) ;
    if (ch == 'a') break ;
}
```



1、

do

ch = getchar () ;

while (ch != 'a') ;

2、

while ((ch = getchar ()) != 'a') ;

3、

for(ch = getchar () ; ch != 'a' ; ch = getchar ()) ;

例 判断 m 是否素数.



凡不能被自身及1以外的整数整除的自然数

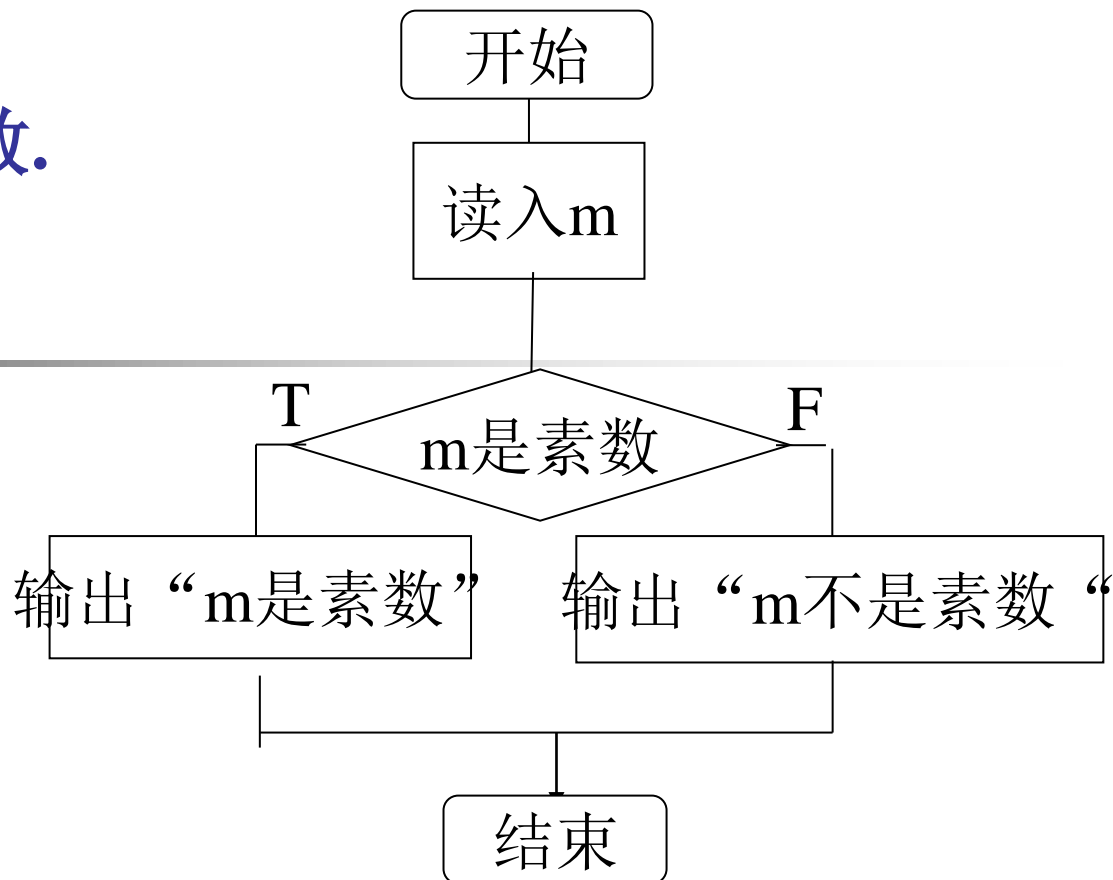
判一个数 m 是否为素数的方法:

当用 $2, 3, \dots, (m-1)$ 的整数去除它时均不能除尽,
则为素数。

$2, 3, \dots, (m/2)$

$2, 3, \dots, \sqrt{m}$

例 判断m是否素数.



将判断一个整数是否素数定义成函数，
是素数函数返回1，否则，返回0。

```
int isprime(int m);
```



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

```
int isprime(int m); /*函数原型 */
```

```
int main(void)
```

```
{ int m;
```

```
do{
```

```
    printf( “input m:” );
```

```
    scanf( “%d” ,&m);
```

```
} while(m<2); // 输入一个大于等于2的整数
```

```
if(isprime(m)) /*函数调用 */
```

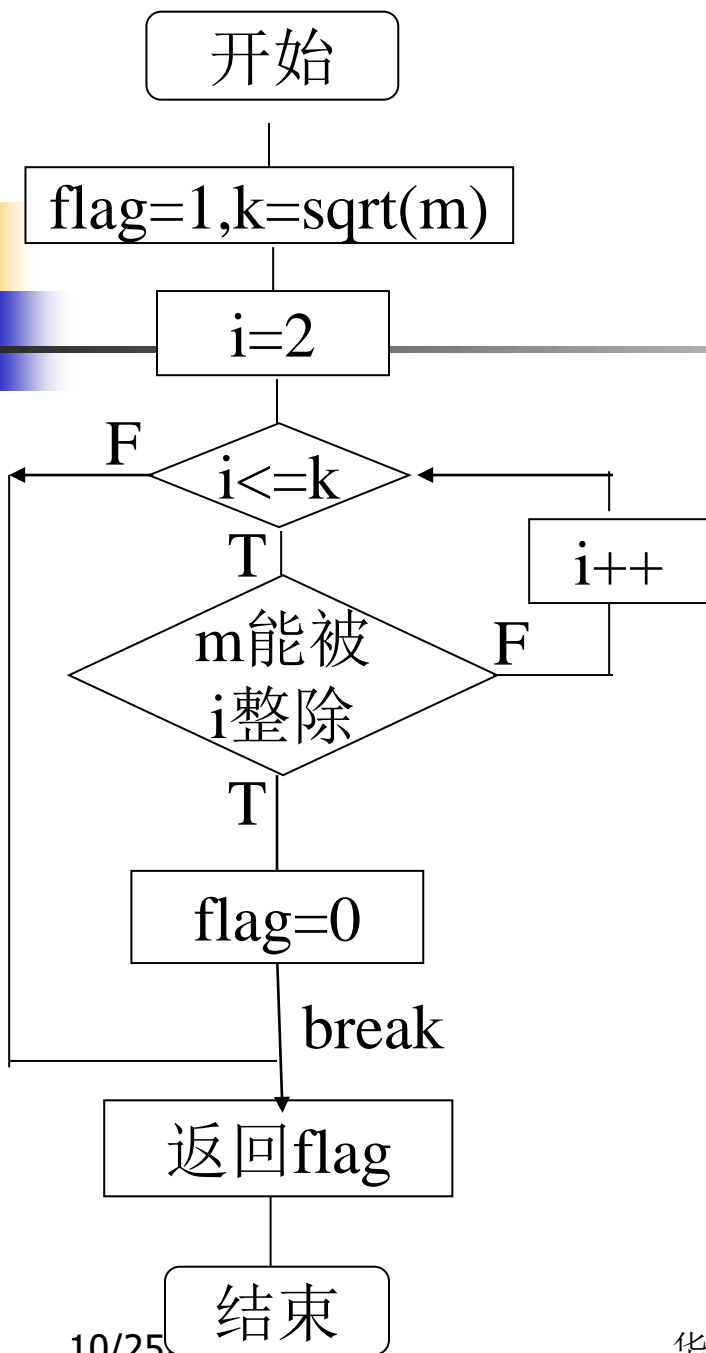
```
    printf( “%d is a prime\n” ,m);
```

```
else
```

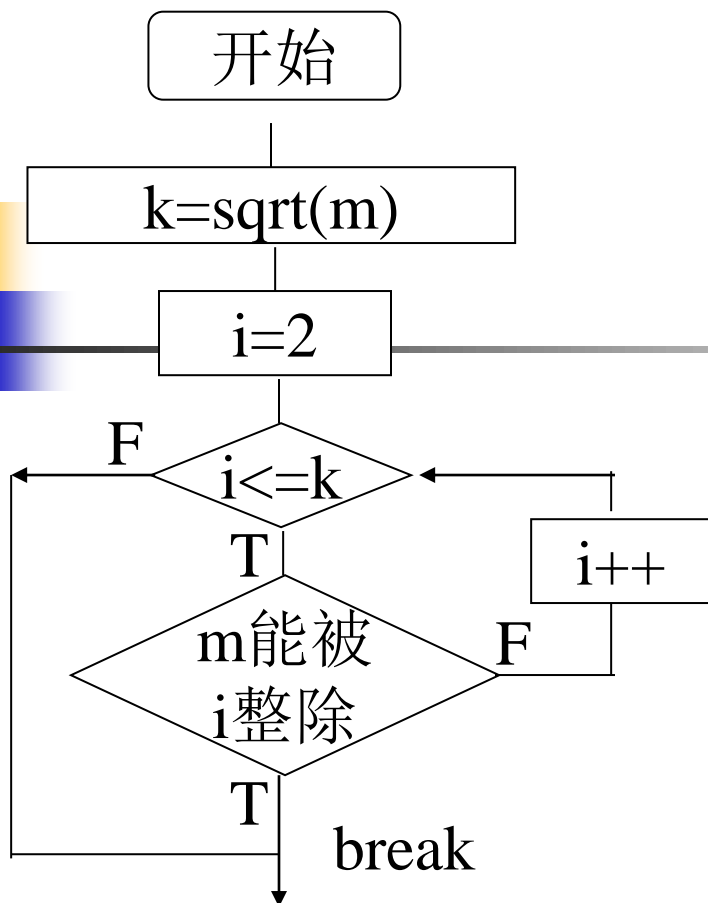
```
    printf( “%d isn’ t a prime\n” ,m);
```

```
return 0;
```

```
}
```

```
#include<math.h>
int isprime(int m)
{
    int flag,k,i;
    flag=1; // 初始是素数
    k=sqrt(m);
    for(i=2;i<=k;i++)
        if (!(m%i)) {
            flag=0;
            break;
        }
    return flag;
}
```

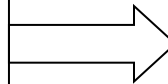


```
#include<math.h>
int isprime(int m)
{
    int k,i;
    k=sqrt(m);
    for(i=2;i<=k;i++)
        if (!(m% i)) {
            break;
        }
    if(i>k) return 1;
    else return 0;
}
```

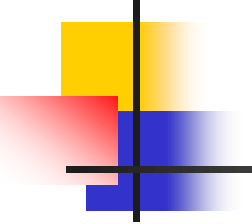


```
#include<math.h>
int isprime(int m)
{
    int k,i;
    k=sqrt(m);
    for(i=2;i<=k;i++)
        if (!(m% i)) {
            break;
        }
    if(i>k) return 1;
    else return 0;
}
```

```
for (i=2;i<=k&&m%i;i++)
    ;
```



```
for(i=2;i<=k;i++)
    if (!(m% i)) {
        break;
    }
```



```
#include<math.h>
int  isprime(int m)
{
    int k,i;
    if(m==2) return 1;
    if(!(m%2)) return 0;
    k=sqrt(m);
    for(i=2;i<=k;i++)
        if (!(m% i)) return 0;
    return 1;
}
```



```
/*#include<math.h>*/
```

```
int isprime(int m)
```

```
{
```

```
    int k,i;
```

```
    if(m==2) return 1;
```

```
    if(!(m%2)) return 0;
```

```
    /*k=sqrt(m);*/
```

```
    for(i=2;i*i<=m;i++)
```

```
        if (!(m% i)) return 0;
```

```
    return 1;
```

```
}
```

例4.22：输出2~100的所有素数，每行输出10个数。

每输出10个数，换一次行。

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

```
#include N 10
```

```
int isprime(int m); /* 函数原型 */
```

```
int main( )
```

```
{
```

```
    int n,k=1;
```

```
    printf(“%5d”,2);
```

```
    for (n=3; n<100; n+=2 ) {
```

```
        if( isprime(n) ) /* 函数调用 */
```

```
        { printf(“%5d”,n);
```

```
            if(!(++k/N)) putchar('\n');
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

二、*continue*语句

continue;

用在循环体内，跳过其后代码，继续下一次循环。

/ 输入10个数，求其中正数的个数及平均值 */*

```
for (sum=num=i=0; i<10; i++) {
```

```
    scanf ( “%d” , &x);
```

```
    if (x<=0) continue;
```

```
    num++;
```

```
    sum+=x;
```

```
}
```

```
if(num)
```

```
    printf( “numbers:%d, average:%f ” ,num,1.0*sum/num);
```

```
if (x>0) {  
    num++;  
    sum+=x;  
}
```

4.11 return语句

return语句可以是如下两种形式之一：

(1) **return ;** **/* void函数 */**

导致从当前函数立即退出，返回到调用函数处

void函数也可以不包含return语句。如果没有return语句，当执行到函数结束的右花括号时，控制返回到调用处，把这种情况称为离开结束。



(2) return e ; /* 非void函数 */

使流程返回到调用函数处，并送表达式的值到调用处。

表达式值的类型应该与函数定义的返回值类型一致，如果不相同，就把表达式值的类型自动转换为函数的类型。

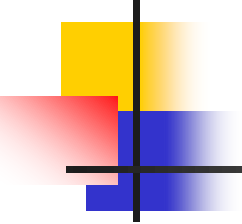


4.12 嵌套循环程序设计

嵌套循环指循环体是一个循环语句，或循环体包含循环语句。

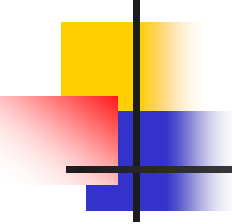
嵌套循环又称为多重循环，其中二重循环应用最为普遍。

```
for(.....) {  
    .....  
    循环语句  
    .....  
}
```

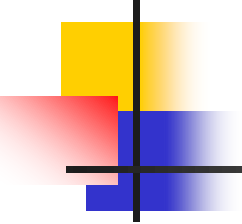


```
for (i=1;i<5;i++) {  
    for(j=1;j<=i;j++)  
        putchar( '*' );  
    putchar( '\\n' );  
}
```

```
*  
**  
***  
****
```




```
for (i=1; i<5; i++) {  
    for( i=1; i<=5; i++ )  
        putchar('*');  
    putchar('\n');  
}
```



```
for (i=1; i<8; i++) {  
    for( i=1; i<=5; i++ )  
        putchar('*');  
    putchar('\n');  
}
```

```
*****  
*****  
*****  
*****  
.  
.  
.
```

◆ 内外循环变量不应同名，
否则将导致死循环或结果出错，



```
for (i=1; i<5; i++)  
    for( j=1; j<=i; j++ )  
        putchar('*');  
    putchar('\n');
```

- ◆ 不要忘记内、外循环体中必须的 { } ,
否则将导致死循环或结果出错,

[例4.26] 编制程序:

求 $s = 1^1 + 2^2 + 3^3 + \cdots + n^n$, **n**由终端输入。

分析:

(1) 输入**n**, 求和变量**s**置初值**0**;

(2) **for(i=1;i<=n;i++) {**

term= i^i ;

s+=term;

}

(3) 输出**s**

[例4.26] 编制程序:

求 $s = 1^1 + 2^2 + 3^3 + \cdots + n^n$, n 由终端输入。

分析:

(1) 输入 n , 求和变量 s 置初值0;

(2) `for(i=1;i<=n;i++) {`

`for(term=1, j=1; j<=i; j++) term*= i;`

`s+=term;`

`}`

(3) 输出 s


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

```
/* compute  $s = 1^1 + 2^2 + \dots + n^n$  */
```

```
int main( )
```

```
{ int i, j, n;
```

```
    long s, term;
```

```
    printf ( “Input n:” );
```

```
    scanf ( “%d” , &n);
```

```
    s=0;
```

```
    for ( i=1; i<=n; i++) {
```

```
        for(term=1, j=1; j<=i; j++) /*  $term = i^i$  */
```

```
            term*=i;
```

```
            s+=term;
```

```
        }
```

```
        printf( “s=%ld\n” , s);
```

```
}
```

◆ 注意给与循环有关的变量赋初值的位置。只需赋一次初值的操作应放在最外层循环开始执行之前，给内循环的有关变量赋初值应放在外循环体内、内循环开始执行之前

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

```
/* compute  $s = 1^1 + 2^2 + \dots + n^n$  */
```

```
int main( )
```

```
{ int i, j, n;
```

```
    long s, term;
```

```
    printf ( “Input n:” );
```

```
    scanf ( “%d” , &n);
```

```
    s=0;  term=1;
```

```
    for ( i=1; i<=n; i++) {
```

```
        for(j=1; j<=i; j++)
```

```
            term*=i;
```

```
        s+=term;
```

```
    }
```

```
    printf( “s=%ld\n” , s);
```

```
}
```

◆ 会怎样？

```
// term= $1^1 * 2^2 * (i-1)^{i-1} * i^i$ 
```

```

#include<stdio.h>
/* compute  $s = 1^1 + 2^2 + \dots + n^n$  */
long power(int x,int n);
int main( )
{ int i, n;
  long s;
  printf ( “Input n:” );
  scanf ( “%d” , &n);
  s=0;
  for ( i=1; i<=n; i++) {
    s+=power(i,i) ;
  }
  printf( “s=%ld\n” , s);
}

```

```

//计算  $x^n$  ( $n>0$ )
long power(int x, int n)
{
  int i;
  long p=1;
  for (i=1; i<=n; i++)
    p*=x;
  return p
}

```

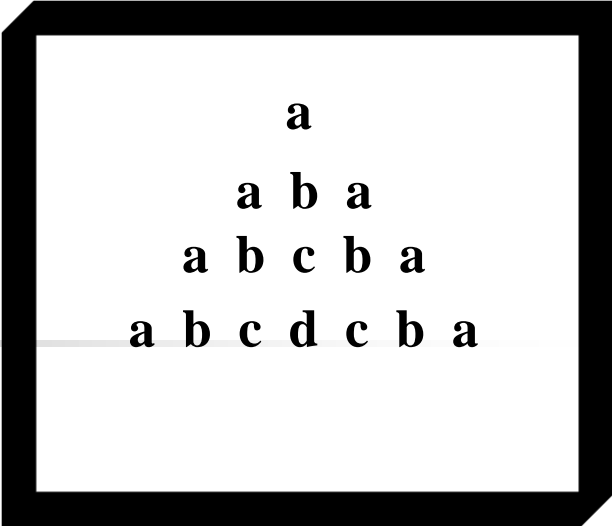
			a							A					
		a	b	a						A	B	A			
	a	b	c	b	a				A	B	C	B	A		
a	b	c	d	c	b	a		A	B	C	D	C	B	A	

- }

```

#define AXIS 40
#include<ctype.h>
int main ( )
{ char c,c1,c2,top;
  int j;
  c=getchar();
  top=isupper(c)?'A':(islower(c)? 'a' :'\0') ;
  if(top) { /* 如 top 非零，输出图形 */
    for(c1= top ; c1<=c; c1++){
      for(j=1;j<=AXIS-2*(c1-top);j++) /* 输出左侧空格 */
        putchar(' ');
      for(c2=top;c2<=c1;c2++) /* 输出左半段 */
        printf("%2c",c2);
      for(c2=c1-1;c2>=top;c2--) /* 输出右半段 */
        printf("%2c",c2);
      printf("\n"); /* 换行 */
    }
  }
  reurn 0;
}

```



```

      a
    a b a
  a b c b a
a b c d c b a

```

```

#define AXIS 40
#include<ctype.h>
int main ()
{ char c,c1,c2,top;
  int i;
  c=getchar();
  top=isupper(c)?'A':(islower(c)? 'a' :'\0');
  if(top) { /* 如 top 非零，输出图形 */
    for(c1= top ; c1<=c; c1++) {
      printf("%*c", AXIS-2*(c1-top), ' '); /* 输出左侧空格 */

      for(c2=top;c2<=c1;c2++) /* 输出左半段 */
        printf("%2c",c2);
      for(c2=c1-1;c2>=top;c2--) /* 输出右半段 */
        printf("%2c",c2);
      printf("\n"); /* 换行 */
    }
  }
  reurn 0;
}

```

可变域宽

```

      a
    a b a
  a b c b a
a b c d c b a

```

枚举（穷举）法

穷举算法是程序设计中用得最为普遍、必须熟练掌握和正确运用的一种算法。它利用计算机运算速度快、精确度高的特点，**对问题的所有可能情况，一个不漏地进行检查，从中找出符合要求的答案。**

用穷举算法解决问题，通常可从两方面进行分析：

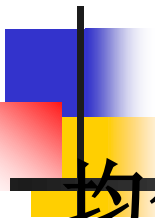
一、问题所涉及的情况：问题所涉及的情况有哪些，情况的种数可不可以确定。把它描述出来。

二、答案需要满足的条件：分析出来的这些情况，需要满足什么条件，才成为问题的答案。把这些条件描述出来。

[例2.24]: 捕鱼和分鱼问题

A, B, C, D, E五人在某天夜里合伙捕鱼，到第二天凌晨时都疲惫不堪，于是各自找地方睡觉。**A**第一个醒来，他将鱼分为五份，把多余的一条扔掉，拿走自己的一份。**B**第二个醒来，也将鱼分为五份，把多余的一条扔掉，拿走自己的一份。**C**、**D**、**E**依次醒来，也按同样的方法拿鱼。问他们合伙捕了多少条鱼？

问题分析




根据题意，总计将所有的鱼进行了**5**次平均分配，每次分配的策略是相同的，即扔掉一条后剩下的分为五份，然后拿走自己的一份，余下其他的四份。

假定鱼的总数为 **X** ，则 **X** 可以按照题目的要求进行五次分配： **$X-1$** 后可被**5**整除，余下的鱼为 **$4*(X-1)/5$** 。若 **X** 满足上述要求，则 **X** 就是题目的解。

```

#include<stdio.h>
int main(void)
{
    int n,i,x,flag=1;    /*flag: 控制标记, 1-能分配, 0-不能分配*/
    for(n=6;flag;n++)    /*采用试探的方法, 令试探值n逐步加大*/
    { /*判断是否可以按照题目要求进行5次分配*/
        for(x=n,i=1;flag&& i<=5;i++)
            if((x-1)%5==0) x=4*(x-1)/5;
            else flag=0;    /*若不能分配则设置标记flag=0, 退出分配*/
        if(flag) break;    /*若分配过程正常结束则找到结果退出试探过程*/
        else flag=1;    /*否则继续试探下一个数*/
    }
    printf("Total number of fish caught=%d\n",n);
    return 0;
}

```



[例4.27] 下列乘法算式中，每个汉字代表1个0~9的数字，不同的汉字代表不同的数字。试编程确定使得整个算式成立的数字组合，如有多种情况，请给出所有可能的答案。

赛软件 * 比赛 = 软件比拼

赛软件 * 比赛 = 软件比拼

a b c d a b c d e

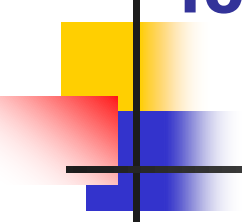
分析：这个算式中有5个不同的汉字，分别用变量a、b、c、d、e来代表，此算式可表示为：

$$(a*100+b*10+c) * (d*10+a) \\ == b*1000+c*100+d*10+e。$$

a: 0~9 b: 0~9 c: 0~9 d: 0~9 e: 0~9

且取值不得重复，共有 $10*9*8*7*6=30240$ 。

可采用枚举算法，依次测试每一种取值情况，如果满足上式，则得到一个解，直到测试完毕，得到所有解。



```
for (a=0; a<10; a++) { /* 枚举 a */
    for (b=0; b<10; b++) { /* 枚举 b */
        for (c=0; c<10; c++) { /* 枚举 c */
            for (d=0; d<10; d++) { /* 枚举 d */
                for (e=0; e<10; e++) { /* 枚举 e */
                    if (a~e互不相等, 且算式成立)
                        输出算式
                }
            }
        }
    }
}
```

[\源程序\ex4_27_1.c](#)

```

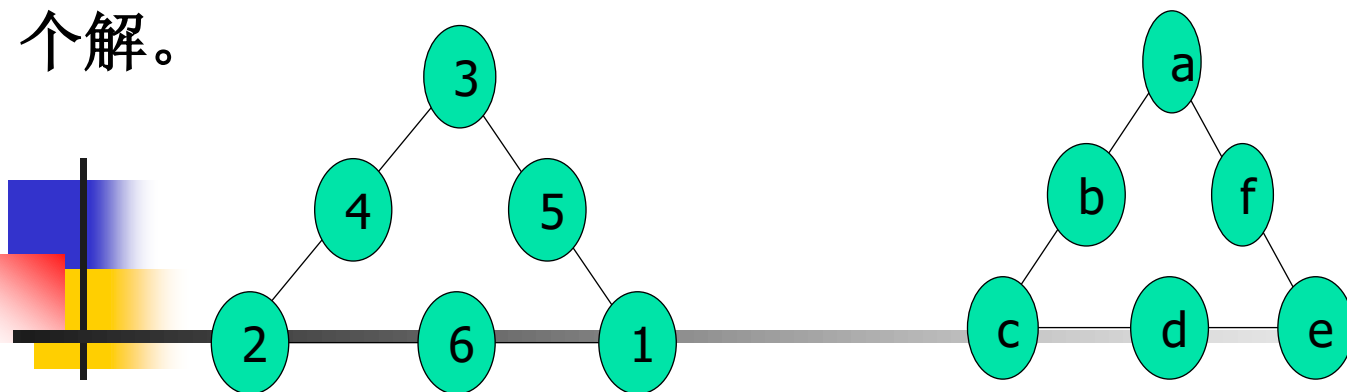
for(a=0;a<=9;a++){ /* 枚举 a, 循环10次 */
    for(b=0;b<=9;b++){ /*枚举 b,循环9次*/
        if(b和a相等) continue;
        for (c=0; c<10; c++) { /*枚举 c,循环8*/
            if(c和a相等或者c和b相等) continue;
            for (d=0; d<10; d++) { /*枚举 d,循环7*/
                if(d和a~c中之一相等) continue;
                for (e=0;e<10; e++) { /*枚举e,循环6
                    if(e和a~d中之一相等) continue;
                    if (算式成立) 输出算式
                }
            }
        }
    }
}

```

源程序\ex4_27.c

枚举法练习

1、将1~6这六个整数排成如图所示的三角形，使得三角形三条边上的数字之和相等，求满足条件的全部解。如图就是一个解。



2、输入正整数 x ($2 \leq x \leq 79$)，输出所有形如 $abcde/fghij=x$ 的表达式，其中 $a \sim j$ 由不同的0~9九个数组成的。例如， $x=32$ 时，输出为：

$$75168/02349=32$$

$$a/b=x$$

穷举 $b=1234 \sim 98765$ ，算出 $a=b*x$

if ($a \leq 98765$ 且 a 和 b 没有相同的数字)

输出 算式

筛法

【例4. 28】 找出10000以内的所有素数，建立素数表。

```
int a[100001];    // 以空间换时间
```

标记x是否素数？ x作为下标值，

 a[x]:1是素数

 :0不是素数

筛法找素数

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

(1) 2是素数，将 $(2*2 \sim N)$ 之间所有2的倍数都划掉

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

(2) 3是素数，将 $(3*3 \sim N)$ 之间所有3的倍数都划掉

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

(3) 4不是素数（不在筛子里），不处理

(4) 5是素数，将 $(5*5 \sim N)$ 之间所有5的倍数都划掉

....

直至 $N/2$ 或 \sqrt{N}

```
for (i=2; i*i<=N; i++)
    if ( a[i] ) {
        for (j=i*i; j<=N; j+=i)
            a[j]=0;
    }
```

【例4. 28】 用筛法构造2~10000之间的素数表，并输出该素数中的素数。

```
#define N 10000
int main()
{
    short i, j, a[N+1]={0, 0};

    for (i=2; i<=N; i++) /* 数组初始化，先假定2~N都是素数 */
        a[i] = 1; /* 数组下标值是被测数，用1标记素数 */
    for (i=2; i*i<=N; i++) /* 从2开始，“筛掉”每个素数的倍数 */
        if (a[i]) { /* 如果a[i]值为1，则i为素数 */
            for (j=i*i; j<=N; j+=i) /* (可以是2*i)，从i倍的i开始“筛”，因之前的倍数已被“筛掉” */
                a[j] = 0; /* 元素值为0表示该元素下标值不是素数 */
        }
    for (i=2, j=0; i<=N; i++) /* 输出2~N间的素数 */
        if (a[i]) { /* “筛”过之后，值为1的元素对应下标为素数 */
            printf("%9d", i);
            if (++j == 8) { /* 为了输出整齐，每行输出8个素数 */
                j = 0;
                printf("\n");
            }
        }

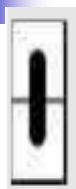
    return 0;
}
```



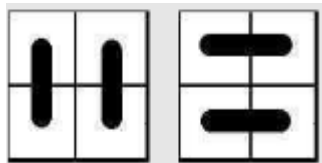
练习

- 1、给出 m ($4 \leq m \leq 10000$), 找出 m 之前的所有孪生素数。孪生素数就是指距离为2的相邻素数。例如, $(3, 5)$, $(5, 7)$ 就是。
- 步骤:
 - (1) 筛法建立素数表
 - (2) 找孪生素数并输出

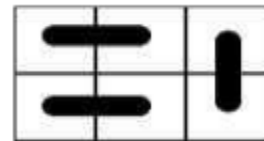
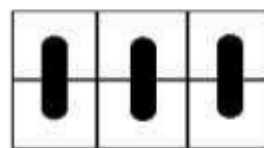
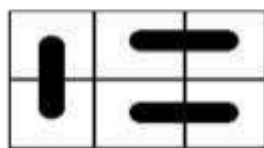
2、用 1×2 的骨牌去铺满一个 $2 \times n$ 的长方形方格, 共有几种铺法. 输入 n 值, 输出铺法种数.



$n=1$
1种



$n=2$
2种



$n=3$
 $2+1=3$ 种

思路: 设 $2 \times n$ 的长方形方格共有 $f(n)$ 种铺法, 显然

$$f(1)=1, f(2)=2, f(3)=3,$$

对于 $f(n)$, 考察最后一列: 如果最后一列是竖着放的, 那么方案数为 $f(n-1)$; 如果最后一列是横着放的, 那么必是两个横着放, 此时方案数为 $f(n-2)$. 因此, $f(n)=f(n-1)+f(n-2)$, $n>2$, 即为Fibonacci数。



递推

递推算法是一种简单的算法，不断通过已知某个条件或结果，逐步推出下一个结果或条件，直至得到问题的解。

- **例4.29** 盒子里有 n ($n < 10000$) 个球，A、B 两人轮流从盒中取球，游戏规则为：

(1) 每次从盒中取出的球的数目必须是 1, 3, 7 或者 8 个；

(2) A 先取球，B 接着取球，双方交替；

(3) 取走盒中最后一个球的一方为输方。

假设取球过程中A和B都不存在失误，即不会将赢的局面变为输。

- 编程实现，输入小球的数目 n ，按规则取球，如果A为赢方则输出1，A为输方则输出0。

- 只有1个球--- A输

递推出 2,4,8,9个球----A赢

($1+1=2, 1+3=4, 1+7=8, 1+8=9$)

- 只有3个球--- A输

递推出 4,6,10,11个球----A赢

($3+1=4, 3+3=6, 3+7=10, 3+8=11$)

- 只有5个球--- A输

递推出 6,8,12,13个球----A赢

...

在递推中需要记录得出的结论？

定义数组tab[i]：下标i代表球数，元素值代表输赢。

即tab[i]表示球数为i个时，谁赢，1--A赢；0-A输

(1) 输入小球的数目n

(2) 初始化数组tab, 设置为0 (A输)

(3) for (i=1; i<=n; i++) { //从只有1个球开始递推
if (i个球时, A输)

则 i+1, i+3, i+7, i+8个球, A赢

推出n个球A赢, 退出整个递推过程, 输出结果

}

(4) 输出tab[n]

将每次能取出的球数保存在一个数组中。

```
int ins[]={1, 3, 7, 8};
```

```
for (j=0; j<4; j++){  
    num=i+ins[j];  
    tab[num] = 1;
```

```
}
```

数组tab
不能越
界!!!

源程序\ex4_29.c