福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

高级语言程序设计（C语言与数据结构）

杨雄

83789047@qq.com

第五章 循环结构

5.1 while语句

5.2 do-while语句

5.3 for语句

5.4 循环嵌套

5.5 break、continue语句

学习目的

- 掌握循环结构程序设计
- 熟练应用循环结构编写程序
- 深刻理解循环结构for语句、while语句、do-while语句的异同

Part.1

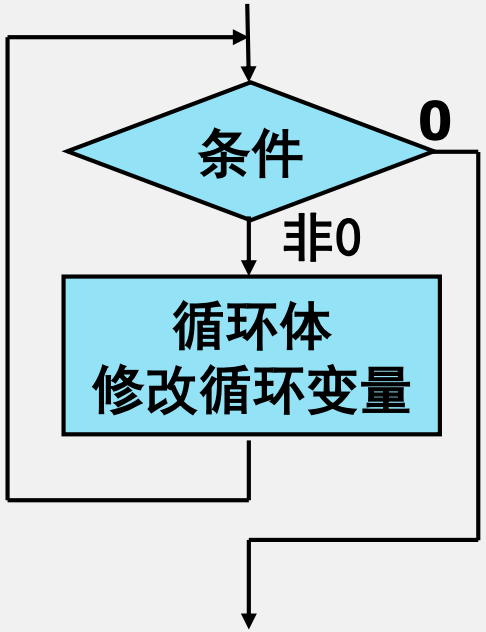
5.1 while语句

5.1 while语句

- **while语句用来实现“当型”循环结构** 循环变量初始化

  - **格式:**

```
while(<表达式>)
{
    语句块; (循环体)
}
```



  - **特点: 先判断, 再执行, 有可能一次也不执行。**

例1： 求1+2+3+ …+100的和

分析:

求和的过程为:

1）sum=1

2）sum =1+2

3）sum =1+2+3

……

100) sum =1+2+ …+100

求和表达式:

sum = sum + i

初值: sum = 0

循环次数 i : 1~100

5.1 while语句

```
#include <stdio.h>
void main( )
{ int  i, sum ;
  i=1;   sum=0 ;
  while( i<=100 )
  {
    sum=sum+i ;
    i++;
  }
  printf( “sum=%d\n” , sum);
}
```

循环前
定义变量
初始化

执行循环
累加求和
修改循环变量

结束
输出结果

输出结果　　sum=5050

5.1 while语句的三种形式

说明:

1. **表达式必须用一对圆括号括起来, 如果循环体包含一条以上的语句应用花括号括起来。**

2. **在循环体内必须修改循环变量的值, 否则死循环。**

3. **在循环体中, 循环变量的值可以被引用、修改, 但不能另赋新值。**

4. **若表达式只用来表示等于零或不等于零的关系时, 条件表达式可以简化成如下形式:**

while (x!=0)　可写成 while (x)

while (x==0) 可写成 while (!x)

例: 指出下列语句的错误

1)
```
int  x=1，y=0；
while( x<10){y=y+x；x++; }
```

可改为:
```
y+=x++;
```

2)
```
int  x=10，y=0;
while(x-->=0)y=y+x；
```

3)
```
int  x=1，y=0  ；
while (x<10)
{ x=3；  y= y+x;  x++;}
```

说明:

5. **循环体通常是一个复合语句，但也可以是一个空语句或单个语句。如:**

空语句:

```
while (x++<10000) ; //分号不能省
```

单语句:

```
x=10 ;
while(x-->0)printf( “%d”, x);
```

输出: **9 8 7 6 5 4 3 2 1 0**

例2： 求n!

```
#include <stdio.h>
void  main( )
{ int  n, i=1 ;          // 控制变量i初值为1,从1～n
  int  s=1 ;
  scanf( “%d” , &n) ;
  while (i<=n )
   { s *=i ;             // 累乘i, 放入s
     i++ ;
   }
  printf( “%d!=%d\n” ,n, s);
}
```

程序运行如下：

5 ↙

5!= 120

①定义变量
②累乘变量赋初值
③读n
④执行循环,求阶乘。
⑤输出

例: 若k为整型变量，则while循环

①
```
k=10;
while(k=0) k=k-1;
```

a)执行10次　b)无限循环　c)一次不执行 √　d)执行一次

②
```
k=2;
while(k!=0) printf(“%d”, k),
k-- ; printf(“\n”);
```

a)执行无限次　b)执行0次　c)执行1次　d) 执行2次 √

③下列程序的输出为:

a) y=0　　b) while构成无限循环

c) y=1　　d) y=-1 √

```
#include <stdio.h>
void main( )
{ int  y=10;
   while(y--) ;
   printf( “y=%d\n” ,y );
}
```

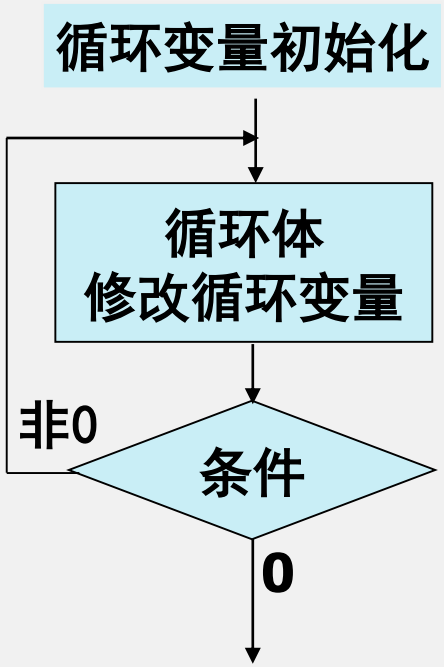
Part.2

5.2 do-while语句

5.2 do-while语句

- do-while语句用来实现“直到型”循环结构

```
do
    循环体
while(<表达式>) ;
```



直到表达式为假时才退出循环。

- 特点：先执行，再判断，至少执行一次循环体。

例:

```
int  t=10 ;
do
{
    t-- ;
 } while ( t>=0 ) ;
 printf( “t=%d\n” , t ) ;
```

输出: t=-1

例：while 与 do-while区别

```
#include <stdio.h>
 voin main( )
{ int  i=65;
   while (i<‘A’)
   { putchar(i);   i++; }
}
```

无输出

```
#include <stdio.h>
 voin main( )
{ int  i=65;
   do
   { putchar(i);   i++;
   } while(i<‘A’);
}
```

输出A

例: 说明下列语句的循环次数及结果

1）
```
a = 1; b = 10 ;
do
{ b-=a ; a++ ;
} while( b-- < 0);
printf( “%d %d\n”, a, b);
```
2 8

2）
```
a = 5 ;
do
{ printf( “%2d\n”, a-- );
} while(!a);
```
5

Part.3

5.3 for语句

5.3 for语句

- **格式:**

循环变量赋初值　循环条件　修改循环变量

for (表达式1 ; 表达式2 ; 表达式3)

没有分号

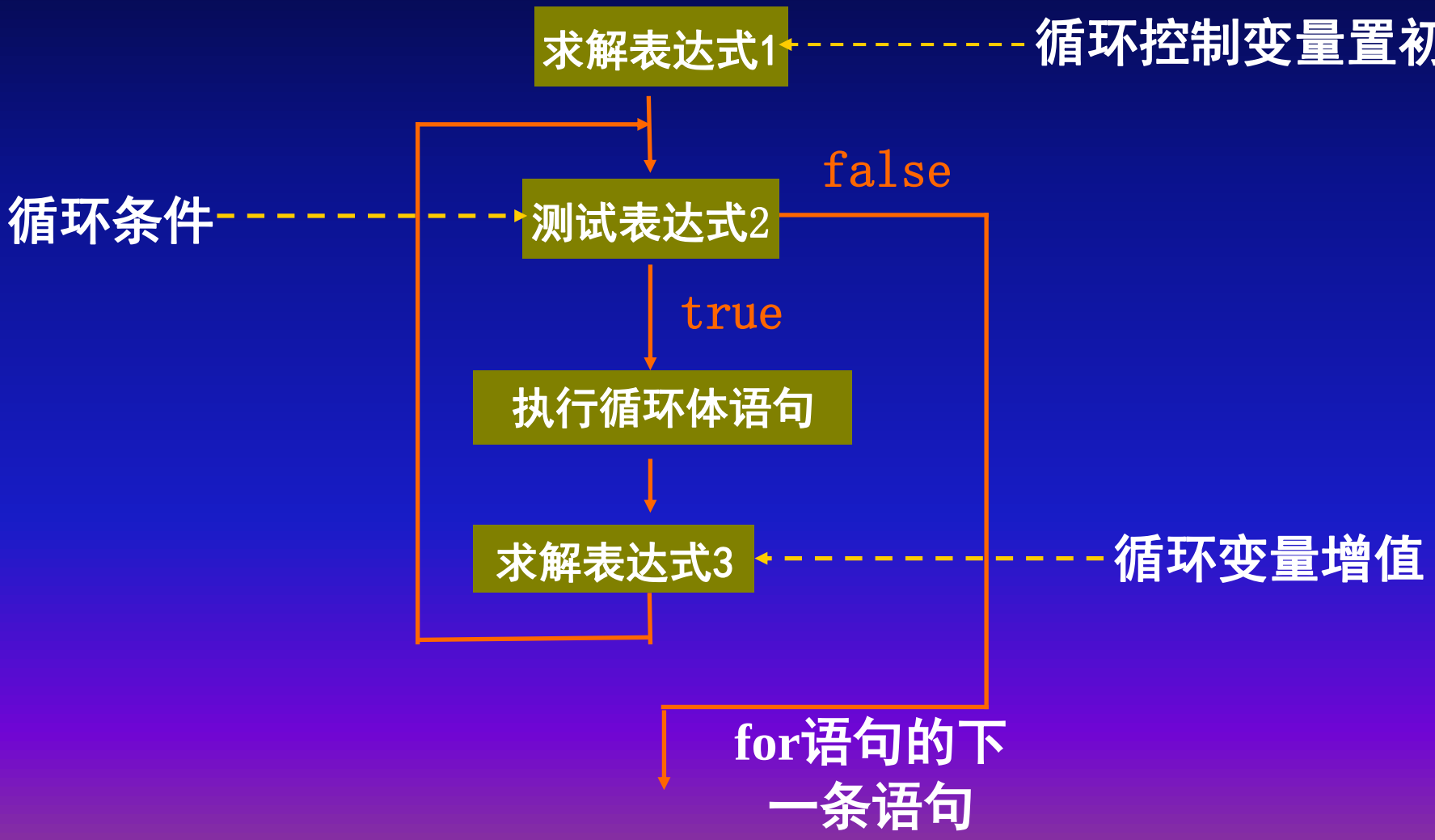
- **例: 求1~100的和**

```
for( i=1; i<=100; i++)
        sum+=i ;
```

⇒

```
i=1;
while( i<=100 )
  sum+=i++;
```

5.3 for语句


求解表达式1
循环控制变量置初值
false
循环条件
测试表达式2
true
执行循环体语句
求解表达式3
循环变量增值
for语句的下
一条语句

5.3 for语句

```
#include <stdio.h>
void  main( )
{  int  i ;
    for ( i = 1;  i <= 10 ;  i++ )
      printf( "%d " , i );
}
```

对控制变量进行初始化

循环条件

控制变量增1

```
1 2 3 4 5 6 7 8 9 10
```

5.3 for语句

- 特点：先判断，再执行，有可能一次也不执行。

如：　x=10；

```
for( y=10; y != x; ++y )
    printf(“%d”, y);
```

for语句的几种格式:

(1)表达式1可以省略，此时应在for语句之前给循环变量赋初值，并且分号不能省略。

如求和运算:

```
i=1 ;
for ( ; i<=100 ; i++)      /*分号不能省略*/
    sum+=i ;
```

for语句的几种格式:

(2)省略表达式2, 则认为循环条件始终为真, 程序将陷入死循环。如求和运算:

```
for( i=1 ; ; i++ )
{  sum+=i ;
   if (i>=100) break;
}
```

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

for语句的几种格式:

(3)表达式3可省略, 此时应在循环体内对循环变量进行修改, 以保证循环能正常结束。

如, 求和运算:

```
for ( i=1 ; i<=100; )    //分号不能省略
    sum+=i++ ;           //修改循环变量
```

for语句的几种格式:

(4)三个表达式都省略，此时应在循环体内对循环变量进行修改，用break语句终止循环。

如, 求和运算:

```
i=1 ;
for ( ; ; )
{ sum+=i++ ;
  if ( i > 100 ) break ;
}
```

for语句的几种格式:

(5)表达式1、3都可以有一项或多项，若有多项则使用逗号表达式。

如, 求和运算:

```
for ( s=0, i=1 ; i<=100; i++)
        s+=i ;

for (s=0, i=1;  i<=100; s+=i, i++) ;
```

Part.4

5.4 循环嵌套

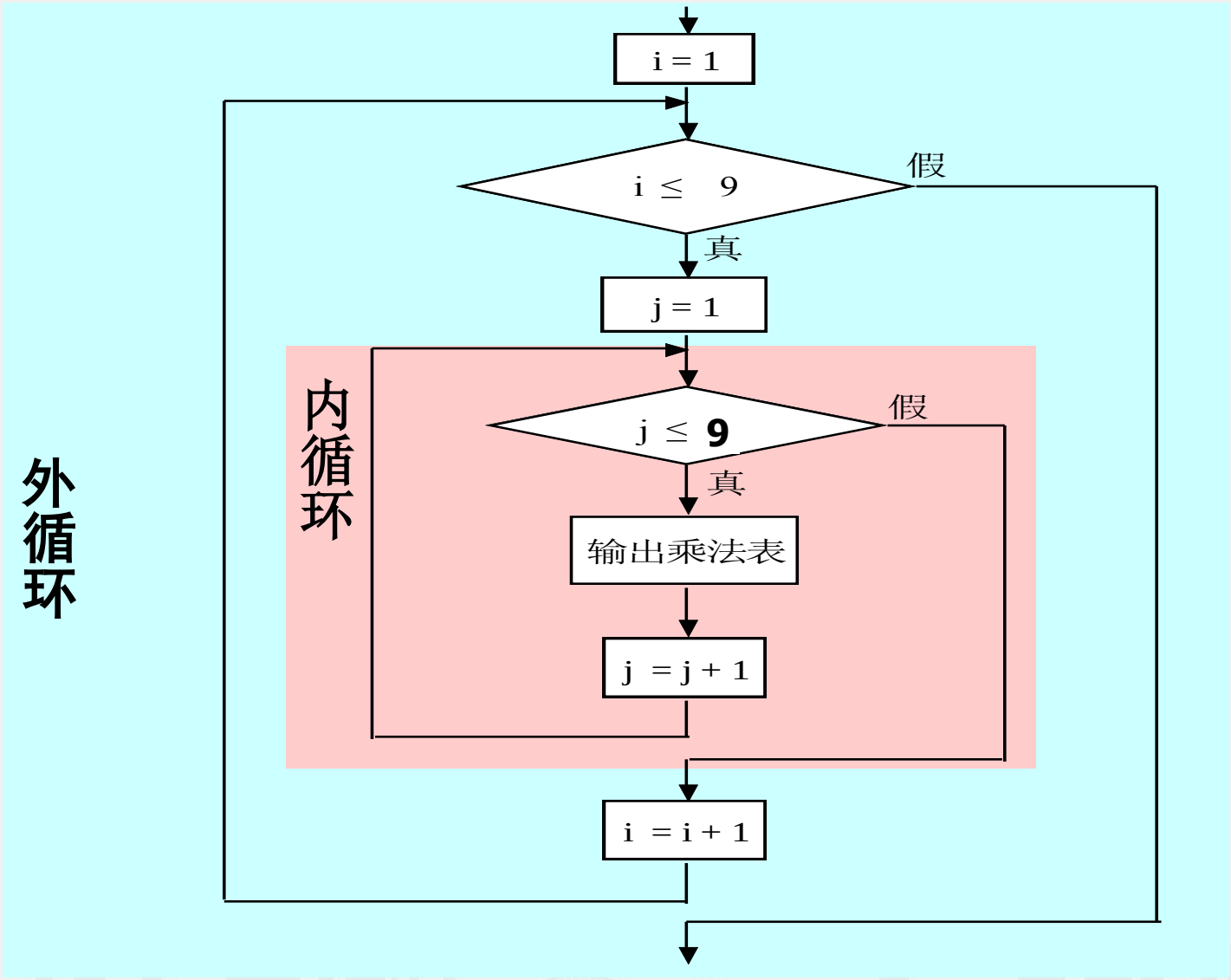
5.4 循环嵌套

在一个循环体内又包含有一个或多个完整的循环结构，称为**循环的嵌套**。内嵌的循环中还可以嵌套循环即为多重循环。

形式:（以双重循环为例）

```
for ( ;  ; )
{      ┆
    for ( ; ; )
}
```

例6　打印“九九表”



i * j

1*1=1

1*2=2

⋮

1*9=9

2*1=2

2*2=4

⋮

2*9=18

⋮

9*1=9

9*2=18

⋮

9*9=81

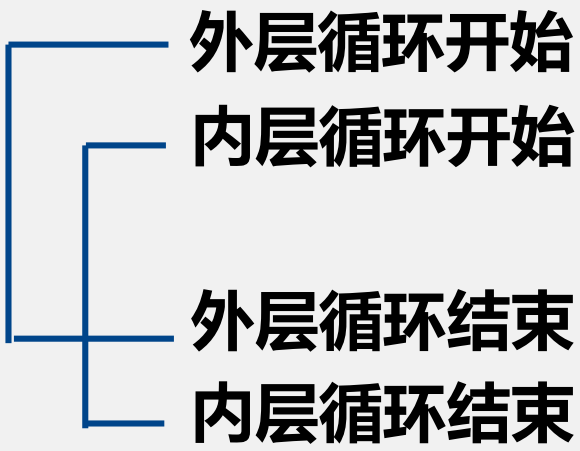
例6 打印“九九表”

```
#include <stdio.h>
void main( )
{ int  i , j ;
  for( i=1 ; i<=9 ; i++)
  { for( j=1; j<= i ; j++)
    printf( “%d*%d=%2d  ” , j, i , i*j );
    printf(” \n” );
  }
}
```

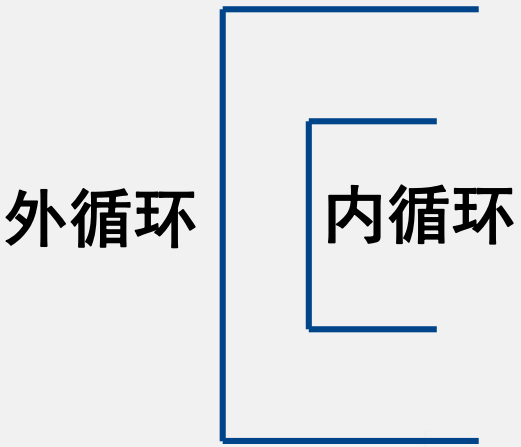
```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
        ……
1*9=9
2*9=18   ……    9*9=81
```

说明:

1) 嵌套要完整，不能交叉。



非法嵌套



合法嵌套

说明:

2) 并列的循环变量可以同名，但嵌套的循环变量不允许同名。

```
for( i=0 ; i<10 ; i++)
{  for ( i =0 ; i <=5 ; i++ )
        printf("%d\n", i );

   for ( j=0 ; j<=5 ; j++ )
        printf( "%d,%d\n" , i, j );
}
```

内循环1

内循环2

说明:

3) 三种循环语句可以相互嵌套，但不允许交叉。

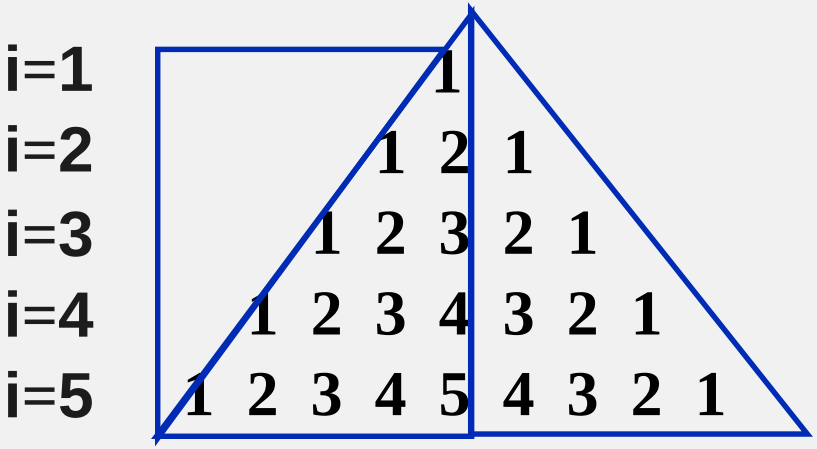
4) 循环与分支可以相互嵌套但不允许交叉。

例7: 输入一个整数n, 输出如下回文塔。

n＝3时

```
    1
  1 2 1
1 2 3 2 1
```

n＝5时




```
for( i=1; i<=n; i++)
{ for( j=1; j<=n-i; j++) printf(“  ”);
  for( j=1; j<=i; j++) printf(“%d ”, j );
  for( j= i-1; j>=1; j--) printf(“%d ”, j ); }
```

Part.5

5.5 continue、break语句

5.5.1 break语句

- **格式:** **break ;**
- **作用:** **提前退出某个循环或跳出switch结构。**
  - **退出当前switch语句**
  - **退出当前循环**(在循环体中break语句与if语句搭配使用)
- **用法: 只能在switch语句和循环体中使用**

- **break只能跳出一层循环 (或一层switch语句)**

5.5.1 break语句

例:

```
for( ; ; )
{  for( ; ; )
     {  …
        if(i==1) break ;
          …
     }
      a=1 ;      /*break跳至此处*/
       …
 }
```

5.5.2 continue语句

- 格式：　continue；
- 作用：结束本次循环，不在执行continue语句之后的循环体语句，使程序回到循环条件，接着进行下一次是否执行循环的判定。
- 说明：只能用于循环结构中，常与if语句联合起来使用，以便在满足条件时提前结束本次循环。
- 注意：在while和do-while结构中，在continue语句被执行之后立即进行循环条件的测试；在for结构中，表达式3被执行之后，然后进行循环条件的测试。

5.5.2 continue语句

例5.9统计1～20之间不能被3整除的数的个数，并输出这些数，要求每行输出十个数。

```
#include <stdio.h>
 void   main( )
 { int   i, s=0;
    for ( i=1 ; i<=20 ; i++)
    {  if ( i%3==0 ) continue ;
        printf( “%d ” , i ) ;
        s++;
        if( s%10==0) printf ( “\n” );
    }
   printf(“\ntotal:%d\n”, s );
 }
```

i—循环变量，从1～20

s—输出数据的个数

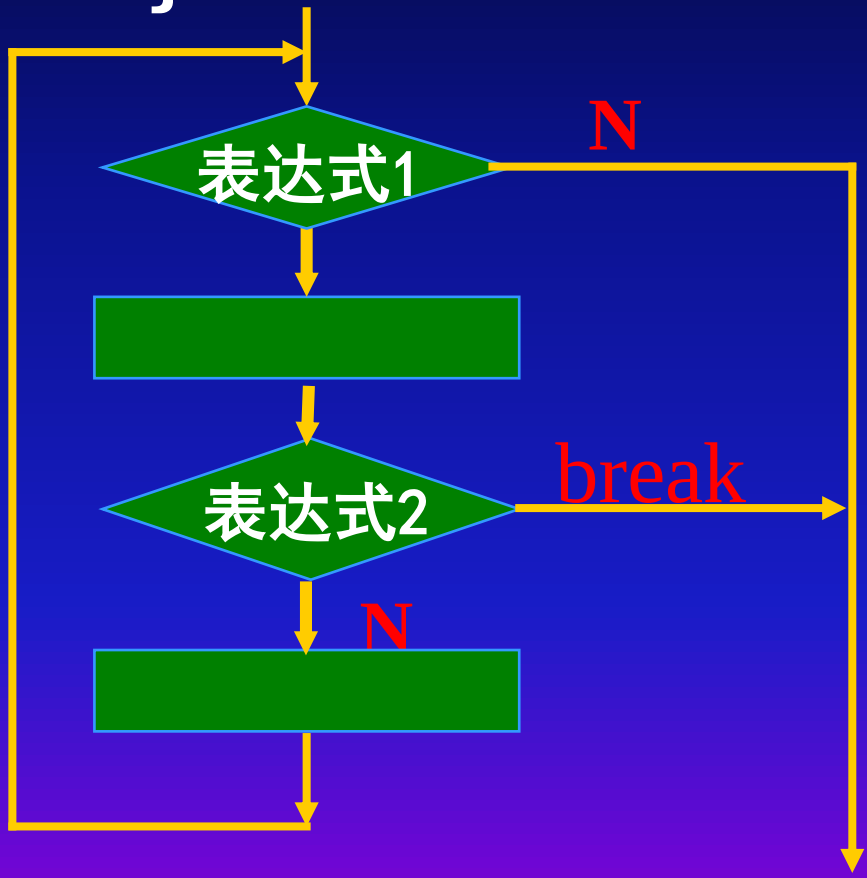
(1)while (表达式1)

{ …

if (表达式2) break ;

…

}



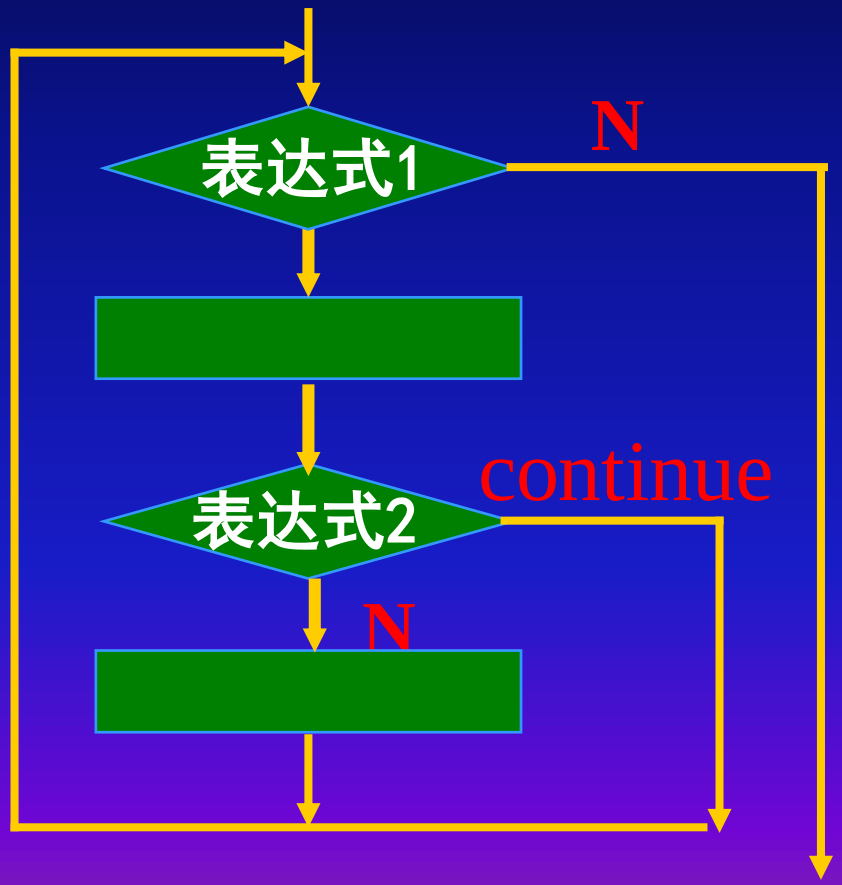
(2) while (表达式1)

{ …

if (表达式2) continue ;

…

}



5.5.2 continue语句

例: break 与 continue 的区别

```
#include <stdio.h>
void main( )
{ int  x;
   for( x=1; x<=10; x++ )
  { if (x==5) break;
     printf( "%d " , x);
   }
   printf( "\n" );
}
```

输出结果: 1 2 3 4

```
#include <stdio.h>
void main( )
{ int  x;
   for( x=1; x<=10; x++)
  { if (x==5) continue;
     printf( "%d " , x);
  }
   printf( "\n" );
}
```

输出结果:1 2 3 4 6 7 8 9 10

5.5.2 continue语句

例11 阅读程序段:

```
int  x, y ;
for (y=x=1; y<= 50; y++)
{ if (x >= 10 ) break  ;
   if ( x%2 == 1 )
      { x += 5; continue; }
   x -= 3 ;  printf( "%d  " , x );
}
```

3 5

① y=1 x=6
② y=2 x=3
③ y=3 x=8
④ y=4 x=5
⑤ y=5 x=10

Part.6

总结

总 结

几种循环的比较：

1) 三种循环可以互相代替。

- 如循环次数已知，一般用for语句；如果循环次数是由循环体的执行情况确定的，则用while或do-while语句。
- 当循环体至少执行一次时，用do-while语句，反之，如果循环体可能一次也不执行，选用while语句或for语句。

2) for、while 属当型循环（先判断后执行）

do-while 循环属直到型循环（先执行后判断）

3) for 循环中的循环体无需对循环条件进行修改，其他循环则必须在循环体中对循环条件进行修改。

习 题

1. 下面选项中，与 if(a) 等价的是 ____。

a) if(a==0)　　√b) if(a!=0)

c) if(a=0)　　d) if(a==1)

2. 在if语句中的 !a等价于 _____ 。

```
int a ; scanf(" %d", &a) ;
if( !a) printf( "continue" ) ;
```

a) a!=0　√b) a==0　c) a>0　d) a>=0

习 题

3. 若有int x, y ; 且x=20, 则以下关于for 循环语句的正确判断为 ____ 。

for(y=20; x!=y; ++x, y++) printf(”----\n”);

√ a) 循环体一次也不执行　　b) 循环体只执行一次

c) 死循环　　d) 输出----

习 题

4. 以下程序段的输出结果是

```
int  x=3 ;
do
 {
    printf(" %3d" , x-=2 ) ;
 } while( !(--x)) ;
```

a) 1　　b) 30　　c) √ 1 –2　　d) 死循环

习 题

5. 以下程序段中, 不是无限循环的是:

a)
```
i=100;
while(1)
{  i=i%100; i++;
   if(i>100)break;
 }
```

b)
```
for( ; ; );
```

√c)
```
short  k=32764;
do
{ k++; k++;
} while(k>0);
```

d)
```
s=32764;
while((s++%2)||s%2)
   s++;
```

习 题

6.下面程序的运行结果是

```
void  main( )
{ int  i;
   for( i=0;  i<=5;  i++)
   { if( i%2 ) printf( “*” );
      else  continue ;
      printf( “#” );  }
   printf( “$\n” );
}
```

√ A) *#*#*#$　　B) #*#*#*$

C) *#*#$　　D) #*#*$>>>

习 题

7. 下面程序的运行结果是

```
void  main( )
{ int  i, j, a=0;
   for( i=0; i<2; i++)
  { for( j=0; j<4; j++)
     { if ( j%2 ) break;
          a++;        }
      a++;
  }
  printf( "%d\n" ,a);  }
```

√ A) 4　　B) 5　　C) 6　　D) 7