高级语言程序设计

主讲：王老师

尚德机构

学习是一种信仰

第四章　结构化程序设计

第一节 结构化程序设计方法

结构化程序设计是以模块功能和处理过程设计为主的详细设计的基本原则。

具体的，采取以下方法可以保证获得结构化程序：

1、自顶向下，逐步求精

2、模块化设计

3、结构化编码：经模块化设计后进入编码阶段，任何程序都由顺序、选择和循环三种基本结构组成。

第二节 结构化程序三种基本结构

- 顺序结构是程序的一个基本结构，它是按照书写顺序依次执行操作。

- 选择结构又称分支结构，是根据某个或某些条件，从若干个操作中选择某个操作执行的一种控制结构。具体的，选择结构分为单分支选择结构、双分支选择结构和多分支选择结构三种。

- 循环结构是由循环控制条件控制循环体是否重复执行的一种控制结构。循环结构分为当型循环结构、直到型循环结构和次数型循环结构三种。

❖控制语句：共9种

- 完成一定的控制功能，这些语句能够根据一定的测试条件决定某些语句是否被执行，如分支、循环、跳转等语句。

if( )~else~	(条件语句)
for( )~	
while( )~	(循环语句)
do~while( )	
continue	(结束本次循环语句)
switch	(多分支选择语句)
break	(终止switch或循环语句)
goto	(转向语句)
return	(从函数返回语句)

❖连续赋值语句

```
int a , b , c ;
a=b=c=1 ;
```

```
int   a=b=c=1 ;
```

步骤：

连续赋值语句应“从右向左”计算　a=b=c=1 ;

等价于：

a=(b=(c=1));

1. 把常数 1 赋给变量c，表达式(c=1) 的值为1;
2. 把表达式(c=1)赋给变量b，表达式(b=(c=1))的值为1;
3. 将右侧表达式的值1赋给a , a =1 。

第三节　顺序结构程序设计

赋值语句的格式包括两种：

1、第一种

【格式】变量=表达式;

【功能】计算表达式的值，然后赋予变量。

2、第二种

【格式】变量 符合赋值运算符 表达式;

【功能】将变量和表达式进行制定的算术或位运算后，再将运算结果赋予变量。

本质上，赋值语句是赋值表达式后面跟一个分号。

顺序结构程序举例

$$s = \frac{1}{2}(a+b+c)$$

$$area = \sqrt{s \times (s-a) \times (s-b) \times (s-c)}$$

例 输入三角形边长，求面积

```
#include <math.h>
#include <stdio.h>
void main( )
{   float a,b,c,s,area;
    scanf("%f,%f,%f",&a,&b,&c);
    s=1.0/2*(a+b+c);
    area=sqrt(s*(s-a)*(s-b)*(s-c));
    printf("a=%7.2f, b=%7.2f, c=%7.2f, s=%7.2f\n",a,b,c,s);
    printf("area=%7.2f\n",area);
}
```

文件包含预处理命令

变量定义

输入数据

输出数据

输入：3,4,6 ↵

输出：a= 3.00, b= 4.00, c= 6.00 s= 6.50
area= 5.33

例 从键盘输入大写字母，用小写字母输出

```
#include "stdio.h"
void main()
{
    char c1,c2;
    c1=getchar( );
    printf("%c,%d\n",c1,c1);
    c2=c1+32;
    printf("%c,%d\n",c2,c2);
}
```

输入：A↵
输出：A,65
a,97

函数调用语句

函数调用语句的格式和功能：

【格式】函数名（参数1，参数2，…，参数n）；

【功能】调用指定的库函数或用户自定义函数，对圆括号中的参数进行该函数指定的运算，运算结果作为函数的返回值返回。

本质上，函数调用语句是函数调用后面跟一个分号。

表达式语句

表达式语句的格式和功能如下：

【格式】表达式；

【功能】计算表达式的值。

本质上，表达式语句的表达式后面跟一个分号。需要注意的是，任何表达式都可以构成语句。

复合语句

复合语句的格式和功能如下：

【格式】{

语句1;

语句2;

...

语句n;

}

```
int i=0,sum=0;
while(i<=100)
{
    sum=sum+i;
    i=i+1;
}
```

【功能】依次执行语句1，语句2，...，语句n。

【说明】1、复合语句可以包含多条语句，但整体上是作为一条语句看待。

2、复合语句中若有数据定义语句，则应放在复合语句中其他语句的前面。

字符输出函数putchar( )

【调用格式】putchar(ch)

【参数】“ch”可以是一个整型变量、字符型变量、整型常量或字符型常量，也可以是一个转义字符或整型表达式，但不能是字符串。

【功能】把一个字符输出到标准输出设备（显示器）。此外，就功能而言“%c”格式说明符的printf()函数可以完全代替putchar()函数的功能。

【返回值】输出ch对应的单个字符。

【说明】

1、putchar()函数只能用于单个字符的输出，一次输出一个字符。

2、C源程序中若使用了putchar()函数，则必须在C源程序（或文件）的开头加上编译预处理命令#include<stdio.h>或#include"stdio.h"。

★putchar 函数（单字符输出函数）

格式： putchar(‘字符’);
或 putchar(字符变量);

强调：被输出的单个字符必须被‘’括起来

❖输出一个字符：

运行结果：BOY

结果当然还是一样的！

```
/* 例 putchar(字符变量) */
#include <stdio.h>
void main()
{
    char a,b,c;
    a='B'; b='O'; c='Y';
    putchar(a); putchar(b); putchar(c);
}
```

❖输出控制字符

```
/* 例   putchar(‘控制字符’ ) */
#include <stdio.h>
void main()
{  char a,b;
    a=‘O'; b=‘k';
    putchar(a); putchar(‘\n’);
putchar(b);
}
```

运行结果：O
k

❖输出转义字符

运行结果：AB

```
/*例   putchar( ) 含有转义符* /
#include <stdio.h>
void main()
{  char a;
    a='B';
    putchar('\101'); putchar(a);
}
```

字符输入函数getchar( )

【调用格式】getchar()

【参数】无参数

【功能】从标准输入设备（键盘）输入一个字符。另外，就功能而言带“%c”格式说明符的scanf( )函数可以完全替代getchar( )函数的功能。

【返回值】输出的单个字符。

【说明】

1、getchar()函数只能用于单个字符的输入，一次输入一个字符。输入的字符也是按字符处理。当输入多于一个字符时，只接收第一个字符。

2、C源程序()函数输入的字符可以赋予一个字符变量或整型变量，也可以不赋给任何变量，而作为表达式的一部分。

3、C源程序中若使用了getchar()函数，则必须在C源程序（或文件）的开头加上编译预处理命令#include<stdio.h>或#include"stdio.h"。

★getchar 函数（单字符输入函数）

格式： getchar( ) ;

强调：输入单个字符后，必须按一次回车，
计算机才接受输入的字符。

```
/*例4.2 */
#include <stdio.h>
void main()
{  char  c;
   c=getchar( );
   putchar(c);
}
```

运行结果：
a↵
a

★ # include <stdio.h>不能少！

格式输出函数printf( )

【调用格式】printf(格式控制字符串,输出项表)

【参数】“格式控制字符串”是由控制输出格式的字符组成的字符串。

“输出项表”是用逗号分隔的若干个表达式。

【功能】按照用户指定的格式把指定的数据输出到标准输出设备（显示器）。

【返回值】返回值为整型，等于正常输出的数据个数。

【说明】

“格式控制字符串”又称“转换控制字符串”，是用一对双引号括起来的字符串，其作用是将输出项表中的数据从内存的二进制形式转换成指定的格式输出。格式控制字符串由格式说明符、附加格式说明符、转义字符和普通字符组成。

1、格式说明符

格式说明符由“%”和“格式字符”组成，其作用是将要输出的数据按格式说明符指定的格式输出。

★printf函数（格式输出函数）

❖printf函数的一般格式

格式： printf(格式控制， 输出表列);

- 格式控制：用双引号括起来的字符串，包含两种信息
  - ◆格式说明：%[修饰符]格式字符，指定输出格式
  - ◆普通字符：原样输出
- 输出表列：要输出的数据，可以是变量或表达式，可以没有，多个时以“,”分隔）

例： printf(“%d %d”， a,b);

printf(“a=%d b= %d\n”， a,b);

★格式字符

❖d格式符：输出十进制整数，有3种用法

- %d格式：按数据实际长度输出，数据范围 -32768~32767。

```
/* %d 格式的 printf( ) 函数使用 */
#include <stdio.h>
void main()
    {
        int a=3 ， b=4;
        printf(“%d  %d\n ”,a,b);
        printf(“a=%d , b=%d\n”,a,b);
     }
```

运行结果：

```
3  4
a=3, b=4
```

★

格式说明决定最终输出的格式

格式说明应与输出列表项个数相同，顺序一致

格式说明通常用小写字母表示

- %md格式：m指定输出字段的宽度
  - ◆数据位数小于m，左端补空格，反之按实际输出。

```
int a=123 ， b=12345 ;
printf(“%4d  %4d ”, a , b);
```

```
_123  12345
```

- %ld格式：输出长整型数据
  - ◆可以用%mld格式指定输出列宽

```
long c=135790
printf(“%ld   \n”, c);
printf(“%8ld    ”, c);
```

```
135790
__135790
```

135790 > 32767 (int 型数据的最大值)

❖o格式符：八进制输出整数

- 是将内存中的二进制位整个按八进制输出，所以输出值没有符号。
- 可以指定输出宽度%mo，长整型可以用%lo格式输出。

❖x格式符：十六进制输出整数

- 同o格式符，无符号，即无负十六进制数。
- 可以指定输出宽度%mx，长整型可以用%lx格式输出。

```
int a= -1;
printf("%d , %o , %8o , %x ", a , a , a , a);
```

输出：

-1，177777，_ _177777，ffff

11	11	11	11	11	11	11	11

-1在内存的存放形式（补码）

❖u格式符：十进制输出unsigned型数据

- int型可以用%u格式输出，unsigned型也可以用%d、%o和%x格式输出。

例 无符号数据的输出

```
#include <stdio.h>
void main()
{
    unsigned int a=65535;
    int b= -2;
    printf("a=%d , %o , %x , %u\n ",a,a,a,a);
    printf("b=%d , %o , %x , %u\n" ,b,b,b,b);
}
```

运行结果：

```
a= - 1 , 177777 , ffff , 65535
b= - 2 , 177776 , fffe , 65534
```

❖c格式符：输出一个字符

- 值在0~255的整数，可以用%c形式输出为字符

例 字符数据的输出

```
#include <stdio.h>
void main()
{
    char c='a';
    int i=97;
    printf("%c , %d\n ",c,c);
    printf("%c , %d\n" ,i ,i);
}
```

运行结果：

```
a , 97
a , 97
```

❖s格式符：输出一个字符串

- 有%s，%ms，% -ms，%m.ns，% -m.ns五种用法

```
例 字符串的输出
 #include <stdio.h>
 void main()
 {
   printf("%3s , %7.2s , %.4s , %-5.3d\n ", "CHINA",
                          "CHINA", "CHINA", "CHINA");
 }
```

m自动=n

运行结果：

CHINA , _ _ _ _ _ _CH , CHIN , CHI _ _

❖f格式符：输出实数

- %f格式：整数部分全部输出，小数6位。可以有非有效数字输出，因为单精度有效位7位，双精度16位。
- %m.nf格式：占m列，其中n位小数，左补空格。
- % -m.nf格式：右补空格

例 %f格式输出实数时的有效位数

```
#include <stdio.h>
void main()
{
    float x,y;
    x=111111.111;y=222222.222;
    printf("%f\n",x+y);
}
```

运行结果：

333333.328125

❖f格式符：输出实数

- %f格式：整数部分全部输出，小数6位。可以有非有效数字输出，因为单精度有效位7位，双精度16位。
- %m.nf格式：占m列，其中n位小数，左补空格。
- % -m.nf格式：右补空格

例 %f格式输出双精度实数时的有效位数

```
#include <stdio.h>
void main()
{
    double x,y;
     a=111111111111.111111111;
     b=222222222222.222222222;
     printf(“%f\n”,x+y);
}
```

运行结果：

333333333333.333010

❖f格式符：输出实数

- %f格式：整数部分全部输出，小数6位。可以有非有效数字输出，因为单精度有效位7位，双精度16位。
- %m.nf格式：占m列，其中n位小数，左补空格。
- % -m.nf格式：右补空格

例 %f格式输出实数时指定小数位数

```
#include <stdio.h>
void main()
{
    float f=123.456;
    printf("%f  %10f  %10.2f  %.2f  %-10.2f\n",f,f,f,f,f);
}
```

运行结果：

```
123.456001_ _123.456001_ _ _ _ _ _123.46_ _123.46_ _123.46_ _ _ _
```

❖e格式符：指数形式输出实数

- %e格式：不指定m和n，小数6位，指数部分共5位，
  其中e和指数符号各1位，指数值3位。
- %m.ne和% -m.ne格式：m、n、-的含义同前面。
  没有n时，自动=6

❖g格式符：输出实数

- 可以自动根据数值大小选择 f 或 e 格式（选列少的）
- 不输出无意义的零

❖ 格式字符表

d	十进制整数	int a=100;printf (“%d”,a);	100
x,X	十六进制无符号整数	int a=255;printf(“%x”,a);	ff
o	八进制无符号整数	int a=8;printf(“%o”,a);	10
u	不带符号十进制整数	int a=100;printf(“%u”,a);	100
c	单一字符	char a=65;printf(“%c”,a);	A
s	字符串	printf(“%s”,“ABC”);	ABC
e,E	指数形式浮点小数	float a=567.789;printf(“%e”,a);	5.677890e+002
f	小数形式浮点小数	float a=567.789;printf(“%f”,a);	567.789000
g,G	e和f中较短一种	float a=567.789;printf(“%g”,a);	567.789
%%	百分号本身	printf(“%%”);	%

表 4-1　printf( )函数常用的格式字符及其含义

格式字符	含　义
d	以十进制有符号形式输出整型数据（正整数不输出符号+）
o	以八进制无符号形式输出整型数据（不输出前缀 0）
X、x	以十六进制无符号形式输出整型数据（不输出前缀 0x），若 x 则输出小写十六进制数符 a～f，若 X 则输出大写十六进制数符 A～F
u	以十进制无符号形式输出整型数据
f	以小数形式输出实型数据（包括单精度实型和双精度实型），小数部分为 6 位
G、g	以 f 或 e 格式中宽度较短的一种格式输出实型数据，不输出无意义的 0。在用 G 时，若以指数形式输出，则指数以大写 E 表示
E、e	以指数形式输出实型数，小数部分为 6 位
c	以字符形式输出，只输出一个字符
s	输出字符串

输出不同类型的数据，需要使用不同的格式字符，其中：
格式字符d——以有符号的十进制整数形式输出。
格式字符f——以小数形式、按系统默认的宽度，输出单精度和双精度实数。
格式字符c——输出一个字符（只占一列宽度）。
格式字符s——输出一个字符串。

2、附加格式说明符

在“%”和“格式说明符”之间还可以加入附加格式说明符，以说明输出数据的标志、域宽（所占列数）、精度和长度修饰符。

表 4-2　printf( )函数常用的附加格式说明字符及其含义

附加格式说明字符	含　义
l	在格式字符 d、o、x、u 之前，输出长整型数据；在格式字符 e、f、g 之前，输出双精度实型数据
h	输出短整型数据，可用在格式字符 d、o、x、u 之前
m（整数）	m 表示输出数据所占的域宽。 当输出数据宽度小于 m 且 m 为正时，输出的数据或字符右对齐，左边补空格；当 m 为负时，输出的数据或字符左对齐，右边补空格。 当输出数据宽度大于 m 时，按实际宽度输出
n（正整数）	当输出实型数据时，表示输出 n 位小数；当输出字符串时，表示截取前 n 个字符输出
-	输出的数据或字符在域内左对齐
+	指定在有符号数的正数前面显示正号 +
0	输出数值时指定左侧的空位自动置 0
#	在八进制和十六进制数前显示前缀 0 和 0x

表 4-3　常用的输出格式

数据类型	输出格式	数据类型	输出格式
有符号整型数据	%d	无符号整型数据	%u
有符号长整型数据	%ld	无符号长整型数据	%lu
实型数据	%m.nf	字符型数据	%c
字符串数据	%s		

3、转义字符

转义字符由一个反斜杠和一个字符组成。

4、普通字符

普通字符是指除格式说明符、附加格式说明字符和转义字符之外的其他字符，输出时按原样输出，起到提示的作用。

5、输出项表

输出项表中给出各个输出项，输出项可以是变量、表达式、常量或函数调用等。

格式输入函数scanf( )

【调用格式】scanf(格式控制字符串,输入项首地址表)

【参数】“格式控制字符串”是由控制输入格式的字符组成的字符串。

“输入项首地址表”由若干个输入项首地址组成，相邻两个输入项首地址之间用逗号分隔。

【功能】从键盘按照“格式控制字符串”中规定的格式读取若干个数据，然后按照“输入项首地址表”中的顺序，依次将数据存入相应的变量对应的地址。

【返回值】输入的数据个数

格式输入函数scanf( )

【说明】

1、格式控制字符串

格式控制字符串可以包含三种类型的字符，即格式说明符、空白字符（空格、回车和<Tab>键）和非空白字符（又称普通字符）

scanf()函数的格式说明符的一般形式：

%[*][宽度][F|N][h|l]格式字符

（1）宽度。宽度是指定该项输入数据所占列数。

（2）赋值抑制字符*。赋值抑制字符表示本输入项对应的数据读入后，不赋予相应的变量（该变量由下一个格式字符输入）

（3）类型修饰符——F、N、h、l分别表示远指针、近指针、短整型和长整型。

★scanf 函数（格式输入函数）

❖一般形式

格式： scanf(格式控制， 地址表列)；

- 功能：按指定格式从键盘读入数据，存入地址表指定的存储单元中,并按回车键结束
- 格式控制：含义同printf函数
- 地址表列：变量地址或字符串地址，地址间“，”分隔。
- 强调：地址列表中每一项必须以取地址运算符&开头。

```
例 用scanf函数输入数据
#include <stdio.h>
voidmain()
{
    int a , b , c ;
    scanf(“%d%d%d”,&a,&b,&c) ;
    printf(“%d, %d, %d\n”,a,b,c) ;
}
```

输入：3_4_5↵
输出：3,4,5

例 scanf(“%4d%2d%2d”,&yy,&mm,&dd);

输入 19991015 ↵

则 1999⇒yy, 10 ⇒mm, 15 ⇒dd

例 scanf(“%3d%*4d%f”,&k,&f);

输入 12345678765.43↵

则 123⇒k, 8765.43⇒f

例 scanf(“%2d %*3d %2d”,&a,&b);

输入 12 345 67↵

则 12⇒a, 67⇒b

例 scanf(“%3c%2c”,&c1,&c2);

输入 abcde ↵

则 ‘a’⇒c1, ‘d’⇒c2

❖输入分隔符的指定

- 一般以空格、TAB或回车键作为分隔符
- 输入数据时，遇非法输入则认为数据结束
- 其它字符做分隔符：格式串中两个格式符间有其它字符，则输入时对应位置也要有相同的字符。

例 scanf("%d:%d:%d",&h,&m,&s);
输入 12:30:45↵
则12 ⇒h, 30 ⇒m, 45 ⇒s

例 scanf("%d%c%f",&a,&b,&c);
输入1234a123o.26 ↵
则 1234 ⇒a, 'a' ⇒b, 123 ⇒c

非法字符

表 4-4　scanf( )函数常用的格式字符及其含义

格 式 字 符	含　义
d	以十进制有符号形式输入整型数据
o	以八进制无符号形式输入整型数据
x	以十六进制无符号形式输入整型数据
c	输入一个字符（空格、回车和制表符都作为有效字符输入）
s	将一个字符串输入到字符数组中，当遇到第一个空白字符（空格、回车和制表符）时结束输入
f、e	输入实型数据

表 4-5　scanf( )函数常用的附加格式说明字符及其含义

附加格式说明字符	含　义
l	在格式字符 d、o、x、u 之前，输入长整型数据；在格式字符 e、f、g 之前，输入双精度实型数据
h	输入短整型数据，可用在格式字符 d、o、x 之前
m（正整数）	指定输入数据所占的域宽，可用在格式字符 d、o、x、f、e 之前
*	表示本输入项在读入后不赋予相应的变量

表 4-6　常用的输入格式

数 据 类 型	输 入 格 式	数 据 类 型	输 入 格 式
有符号整型数据	%d	无符号整型数据	%u
有符号长整型数据	%ld	无符号长整型数据	%lu
单精度实型数据	%f	双精度实型数据	%lf
字符型数据	%c	字符串数据	%s

输入项首地址表

输入项首地址表是由接收输入数据的变量地址组成的，地址之间用逗号“，”分隔。

例：

1、执行语句printf(“%s\n,”“ World\0Wide\0Web”);后的输出结果是_____。

答案：World

解析：执行语句printf(“%s\n,”“ World\0Wide\0Web”);后的输出结果是World，是因为%s\n在输出的时候遇到\0就结束输出。

2、设int a；，从键盘输入数据给变量a的输入语句是_____。

答案：scanf（“%d”，&a）；

解析：格式输入函数scanf（）的格式是scanf（格式控制字符串，输入项首地址表），故从键盘输入数据给变量a的输入语句是scanf（“%d”，&a）；。

第四节 选择结构程序设计

单分支选择结构

C语言中实现单分支选择结构的语句是if语句。if语句的一般形式：

【格式】if（表达式）语句;

【功能】计算表达式的值，若为“真”（非0），则执行语句；否则不执行语句。

【说明】

1、if语句中的“表达式”必须用一对圆括号括起来。

2、“表达式”可以是任何类型，除了常用的关系表达式或逻辑表达式外，也允许是其他类型的数据，如整型、实型或字符型等。

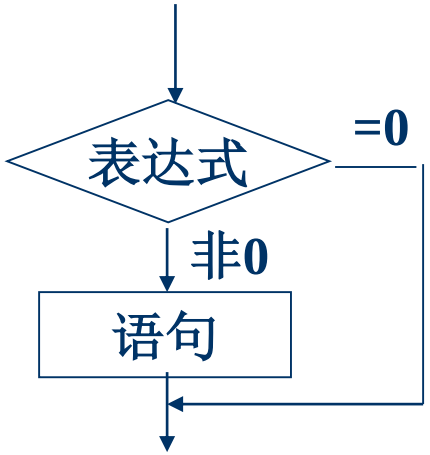
3、语句可以是任何语句（除了数据定义语句），也可以是另一个if语句（称为嵌套的if语句）。

if 语句（条件选择语句）

★If语句的三种形式

❖形式一：

- 格式：if (表达式) 语句
- 执行过程：




```
main()
{
   int x , y ;
   x=20 ;
   y=10 ;
   if (x>y)
       printf("%d",x);
}
```

没有“；”

双分支选择语句

C语言中实现双分支选择语句的是if-else语句。if-else语句的一般形式：

【格式】if（表达式）语句1;

else 语句2;

【功能】计算表达式的值，若为“真”（非0），则执行语句1；否则执行语句2.

【说明】

1、if语句中的“表达式”必须用一对圆括号括起来。

2、“表达式”可以是任何类型，除了常用的关系表达式或逻辑表达式外，也允许是其他类型的数据，如整型、实型或字符型等。

双分支选择语句

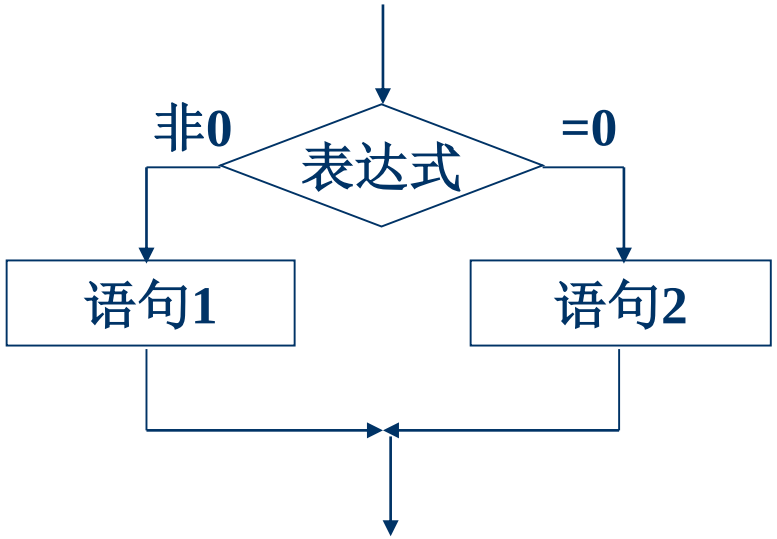
3、语句1和语句2可以是任何语句（除了数据定义语句），也可以是另一个if-else语句（称为嵌套的if-else语句）。

4、“语句1”和“语句2”可以是一条简单的语句，也可以是复合语句。不管是简单语句还是复合语句中的各个语句，每条语句后面的分号必不可少。

5、else子句是if语句的一部分，必须与if配对使用，不能单独使用。

❖形式二：

- 格式：if (表达式)
  语句1
  else
  语句2
- 执行过程：




```
main()
{
   int x , y ;
   x=20 ;
   y=10 ;
   if (x>y)
       printf("%d",x);
  else
       printf("%d",y);
}
```

if-else-if语句的嵌套（分支嵌套的特殊形式）

这种嵌套是在每一层else分支下嵌套一个if-else语句，即：

```
if（表达式1） 语句1
  else if（表达式2） 语句2
        else if（表达式3） 语句3
                …
                  else if （表达式n） 语句n
                        else  语句n+1
```

if-else-if语句的嵌套（分支嵌套的特殊形式）

需要注意以下几点：

1、从上至下逐一对表达式进行判断，只有遇到某一表达式值为“真”（非0）时，则执行与之相对应的语句，执行完毕后跳出整个选择结构；如果没有任何一个表达式值为“真”，则执行最后一个else后面的语句n+1.

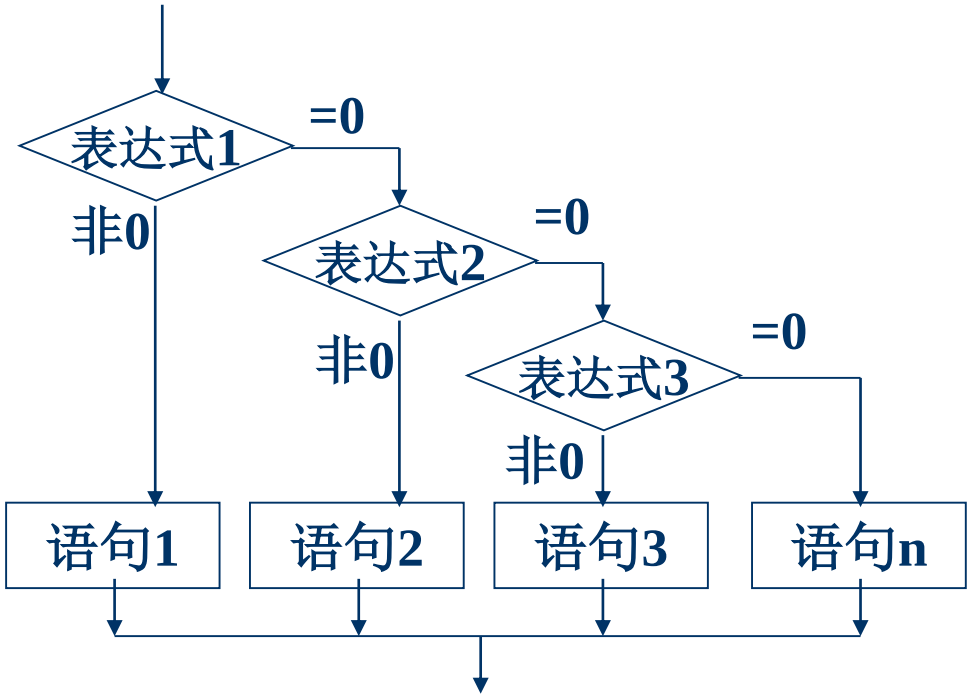
2、不管有多少分支，程序在执行完一个分支后其余的分支将不再执行。

3、else if不能写成elseif，即else和if之间需要有空格分隔。

4、if语句的嵌套层数没有限制，可以多重嵌套。多重嵌套可以扩展选择的分支数量，但嵌套的层数太多会增加变成的难度。因此，嵌套的层数不宜过多。

❖形式三：

- 格式：if (表达式1) 语句1
  else if (表达式2)　语句2
  else if (表达式3)　语句3
  ……
  else if (表达式m)　语句m
  else 语句n
- 执行过程：



例：
```
if (number>500)          cost=0.15 ;
else if (number>300)     cost=0.1 ;
else if (number>100)     cost=0.075 ;
else if (number>50)      cost=0.05 ;
else                     cost=0 ;
```

必须有“；”

❖几点说明：

- if后面的表达式类型任意

```
if(a = =b&&x= =y)    printf("a=b,x=y");
if(3)    printf("OK");
if('a')   printf("%d",a);
```

- if 后面的语句可以是复合语句，必须要加{ }

考虑下面程序的输出结果：

```
main()
   {   int x,y;
       scanf("%d,%d",&x,&y);
       if(x>y)
         {x=y;   y=x;  }
       else
         { x++; y++;  }
       printf("%d,%d\n",x,y);
    }
```

{ }后没有“ ; ”

Compile Error!

例：输入两个数并判断两数是否相等

```
#include <stdio.h>
void main()
{
   int a,b;
   printf("Enter integer a:");
   scanf("%d",&a);
   printf("Enter integer b:");
   scanf("%d",&b);
   if(a= =b)
      printf("a==b\n");
   else
      printf("a!=b\n");
}
```

运行：**Enter integer a:12↵**
Enter integer b:12↵
a==b

运行：**Enter integer a:12↵**
Enter integer b:9↵
a!=b

例：判断输入字符种类

```
#include<stdio.h>
void main()
 {  char c;
    printf("Enter a character:");
    c=getchar();
    if(c<32)  printf("The character is a control character\n");
    else if(c>='0'&&c<='9')  printf("The character is a digit\n");
    else if(c>='A'&&c<='Z')  printf("The character is a capital letter\n");
    else if(c>='a'&&c<='z')  printf("The character is a lower letter\n");
    else printf("The character is other character\n");
 }
```

运行： **Enter a character： F1 ↵**

The character is other character

例 输入两个实数，按由小到大的次序输出两数

```
#include <stdio.h>
void main()
{
   float a,b,t ;
   scanf("%f,%f",&a,&b);
   if(a>b)
      {t=a;a=b;b=t;}
   printf("%5.2f,%5.2f",a,b);
}
```

输入：3.6, -3.2 ↵
输出：-3.20, 3.60

例：输入三个数，按由小到大的次序输出

```
#include <stdio.h>
void main()
{
   float a,b,c,t ;
   scanf("%f,%f,%f",&a,&b,&c);
   if(a>b)
      {t=a;a=b;b=t;}
   if(a>c)
      {t=a;a=c;c=t;}
   if(b>c)
      {t=b;b=c;c=t;}
   printf(“%5.2f,%5.2f,%5.2f”,a,b,c);
}
```

输入：3,7,1 ↵
输出：1.00,3.00,7.00

if语句的嵌套（分支嵌套的一般形式）

这种嵌套是在if-else语句的任一分支上嵌套单分支的if选择结构或双分支的if选择结构。

例如：if（表达式1）

```
        if(表达式2) 语句1;
        else 语句2;
    else
        if(表达式3) 语句3;
        else 语句4;
```

if后面和else后面的语句可以再包含if语句。需要注意的是，else总是与前面最近的并且没有与其他else匹配的if相匹配。

★If 语句的嵌套

- If语句中又包含一个或多个if语句称为if语句的嵌套。
- 实际上只要将前述if语句的形式1和2中的内嵌语句用一个if语句代替，即成为if语句的嵌套。
- 嵌套的if语句还可以嵌套另一个if语句，形成多重嵌套。

❖一般形式：

```
if (条件1)
        if (条件2)  语句1
        else        语句2
else
        if(条件3)   语句3
        else        语句4
```

内嵌if

内嵌if

注意：else 总是与前面最近的if 配对。

例：输入两个数并判断其大小关系

```
#include <stdio.h>
void main()
{
   int x,y;
   printf("Enter integer x,y:");
   scanf("%d,%d",&x,&y);
   if(x!=y)
      if(x>y)  printf("X>Y\n");
      else       printf("X<Y\n");
   else
      printf("X==Y\n");
}
```

Enter integer x,y:12,23↵
X<Y
Enter integer x,y:12,6↵
X>Y
Enter integer x,y:12,12↵
X==Y

★条件运算符

if语句中，当表达式为“真”和“假”时，都只执行一个赋值语句给同一个变量赋值时，可以用条件运算符处理。

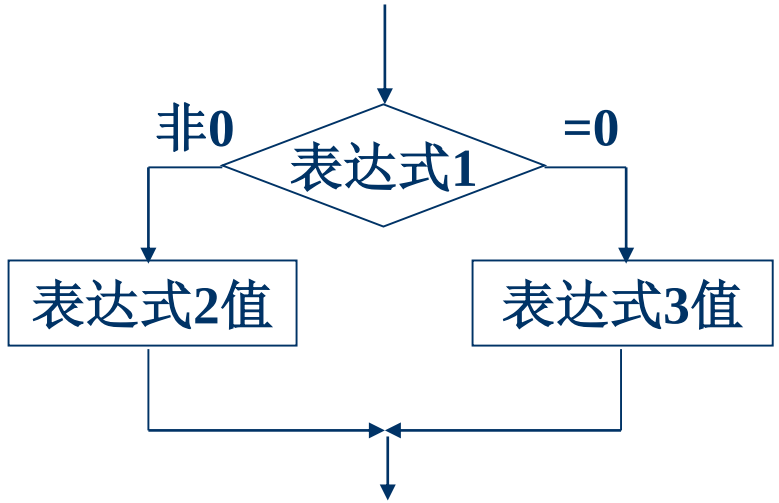
```
if (a>b) max=a;
else max=b;
```

⟺

```
max=(a>b)? a:b;
```

表达式1？ 表达式2：表达式3

条件运算符是 C 语言中唯一的三目运算符



例　输入一个字母，大写转小写，然后输出字母

```
#include <stdio.h>
void main()
{
    char ch;
    scanf("%c",&ch);
    ch=(ch>='A' && ch<='Z')? (ch+32) : ch;
    printf("%c",ch);
}
```

输入：A
输出：a

多分支选择语句

除了if语句的嵌套可以实现多分支选择结构外，C语言还提供了一种直接实现多分支选择结构的switch语句。

switch语句的一般形式：

【说明】switch(表达式)

```
{
   case 常量表达式1:语句组1;break;
   case 常量表达式2:语句组2;break;
   …
   case 常量表达式n:语句组n;break;
   [default:语句组n+1;[break;]]
}
```

多分支选择语句

【功能】

1、switch后面的“表达式”常用的是字符型或整型表达式。

2、“常量表达式”是由常量或符号常量组成的表达式，所有常量表达式的值必须互不相同。

3、当switch后面的“表达式”的值，与某个case后面的“常量表达式”的值相等时，则执行case后面的语句（组），当执行到语句break;时，跳出该switch语句，转而执行switch语句的下一条语句。

4、如果switch后面“表达式”的值没有与任何一个case后面的“常量表达式”的值相等，则执行default后面的语句（组）。之后，执行switch语句的下一条语句。

多分支选择语句

【说明】

1、每个case后面的“常量表达式”的值，必须各不相同，否则会出现相互矛盾的现象。

2、case后面的“常量表达式”仅起到语句标号的作用，并不进行条件判断。系统一旦知道入口标号，即从此标号开始执行，不再进行标号判断，所以必须加上语句break;，以便结束switch语句。

3、程序的执行结果与各case和default子句的先后次序无关。

4、多个case子句，可共用同一个语句（组）。

多分支选择语句

5、switch语句实现的多分支选择结构，完全可以用if语句或if语句的嵌套实现。

6、break语句是C语言的一种语句，其功能是中断正在执行的语句。在switch语句中其作用是执行完某一语句（组）后，跳出该switch语句，无条件转至switch语句的下一条语句。若省略break语句，则执行完某一语句（组）后，将继续执行switch语句中其后的所有语句（组）。

7、default及其后面的语句（组）可以省略。省略时，若表达式的值和n个常量表达式的值均不同，则不执行任何操作就跳出该switch语句。

8、语句组不需要用一对花括号{}括起来。

9、switch语句允许嵌套。

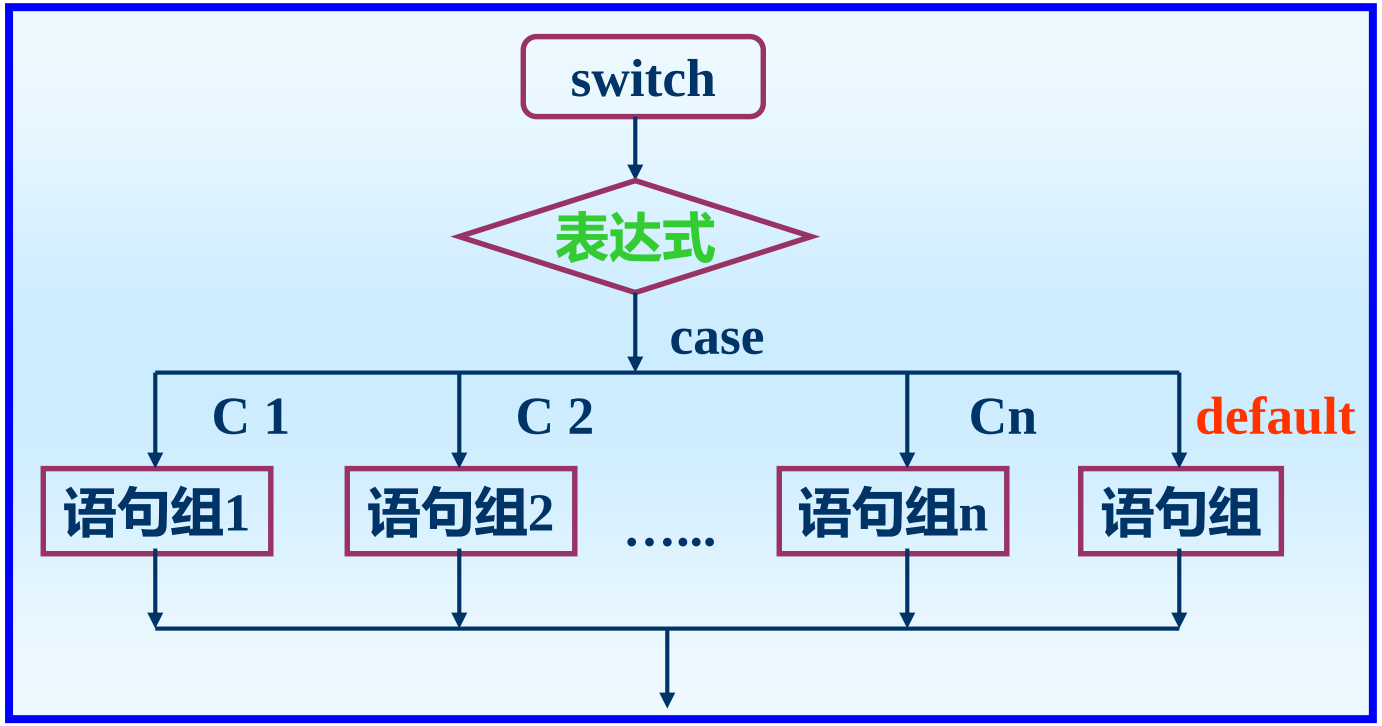
switch语句（多分支选择语句）

❖一般形式：

```
switch(表达式e)
{ case  C1:
        语句1; break;
  case  C2:
        语句2; break;
       …….
  case  Cn:
        语句n; break;
  [default:语句n+1; break;]
}
```

需要跳出switch语句时使用

❖执行过程



❖switch几点说明

- C1,C2,…Cn是常量表达式,且值必须互不相同
- 常量表达式起语句标号作用，必须用break跳出
- case后可包含多个可执行语句，且不必加{ }
- switch可嵌套
- 多个case可共用一组执行语句

```
#include <stdio.h>
main()
{ char  grade;
   grade =getchar();
   switch(grade)
      { case 'A' : printf("85~100 \n"); break;
        case 'B' : printf("70~84   \n"); break;
        case 'C' : printf("60~69   \n"); break;
        case 'D' : printf("<60      \n"); break;
        default   :  printf("Error  \n");
      }
}
```

```
A↵
85~100
```

eak;

课堂练习

1、if（表达式）和while（表达式）中的“表达式”（　）。

A. 只能是逻辑的

B. 只能是关系的

C. 只能是算术的

D. 可以是任意的

课堂练习

1、if（表达式）和while（表达式）中的“表达式”（　）。

A. 只能是逻辑的

B. 只能是关系的

C. 只能是算术的

D. 可以是任意的

答案：D

解析：if（表达式）可以是任何类型，除了常用的关系表达式或逻辑表达式外，也允许是其他类型的数据，如整型、实型或字符型等。

while（表达式）还被称为“循环控制条件”，可以是任何类型的表达式，常用关系或逻辑表达式。

课堂练习

2、下列叙述正确的是（）

A. for语句中的循环体至少被执行一次

B. while语句中的循环体至少被执行一次

C. continue与break在循环语句中的作用相同

D. switch语句中case后可以不出现break语句

课堂练习

2、下列叙述正确的是（）

A. for语句中的循环体至少被执行一次

B. while语句中的循环体至少被执行一次

C. continue与break在循环语句中的作用相同

D. switch语句中case后可以不出现break语句

答案：D

解析：break语句是C语言的一种语句，其功能是中断正在执行的语句，在switch语句中其作用是执行完某一语句后，跳出该switch语句，无条件转至switch语句的下一条语句。若省略break语句，则执行完某一语句后，将继续执行switch语句中其后的所有语句。

第五节 循环结构程序设计

while语句

while语句主要用于实现当型循环结构，其一般形式：

【格式】while(表达式)

语句;

【功能】计算表达式的值，当其值为“真”（非0），则执行语句；重复执行上述操作，直至表达式的值为“假”（0）时为止。

循环结构程序设计

【说明】

1、语句又称为“循环体”，可以是任何语句（除了数据定义语句外），通常是复合语句。

2、表达式称为“循环控制条件”，可以是任何类型的表达式，常用关系或逻辑表达式。

3、先判断后执行循环体。即表达式不成立时，循环体最少执行0次。

4、当循环体内无改变表达式的语句（例如i++）时，while(1)将是死循环。

5、若循环体中又含有“循环语句”，则称为循环的嵌套，即多重循环。

6、在书写格式上建议语句（循环体）比while缩进若干格，以便识别重复执行的的操作。

while语句

while语句实现“当型”循环结构。

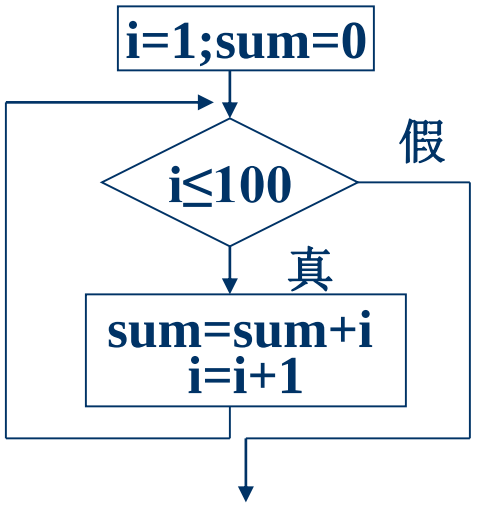
❖一般形式：

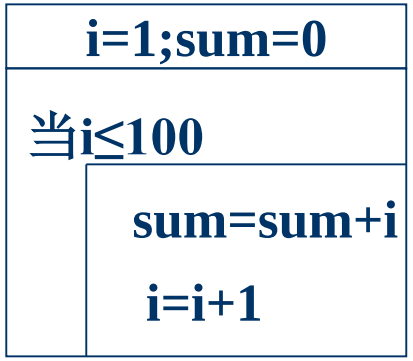
```
while(表达式)
    循环体语句;
```

没有“;”

❖功能：先判断表达式，若为真则执行循环体，再判断表达式，重复上述过程，直到表达式为假时退出循环。

❖求 $\sum_{n=1}^{100} n$





例 用while语句构成循环，求 $\sum_{n=1}^{100} n$

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

循环初值

循环终值

循环条件

循环体

循环变量增值

❖关于while循环语句：

- while循环先判断表达式，后执行循环体。循环体有可能一次也不执行。
- 循环体若包含一个以上语句，应该用{}括起来。
- 循环体应包含有使循环趋向结束的语句；
- 下列情况，退出while循环
  - 条件表达式不成立（为零）
  - 循环体内遇 break , goto
- 无限循环

```
while(1)
    循环体；
```

例 用while语句显示1~10的平方

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

```
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
10*10=100
```

do-while语句

do-while语句主要用于实现直到型循环结构，其一般形式：

【格式】do 语句;

while(表达式);　　/*本行的分号不能省略*/

【功能】

1、执行语句。

2、计算表达式，如果其值为“真”（非0），则转置1继续循环；否则，转至3。

3、执行do-while语句的下一条语句。

do-while语句

【说明】

1、语句又称为“循环体”，可以是任何语句（除了数据定义语句外），通常是复合语句。

2、表达式称为“循环控制条件”，可以是任何类型的表达式，常用关系或逻辑表达式。需要注意的是，表达式必须用圆括号括起来，其后有一个分号。

3、先执行循环体，在判断循环控制条件。适用于无论循环控制条件是否成立，先执行一次循环体的情形。此外，do-while语句能够实现的功能for语句也能实现，而且更简洁。

4、若循环体中又含有“循环语句”，则称为循环的嵌套，即多重循环。

5、在书写格式上建议do和while对其，以便识别重复执行的操作。

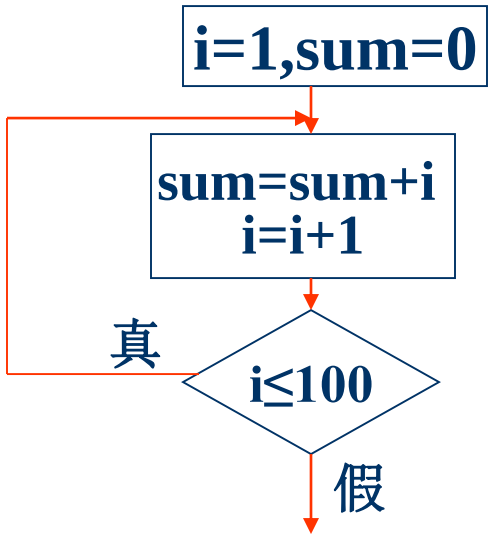
do-while语句

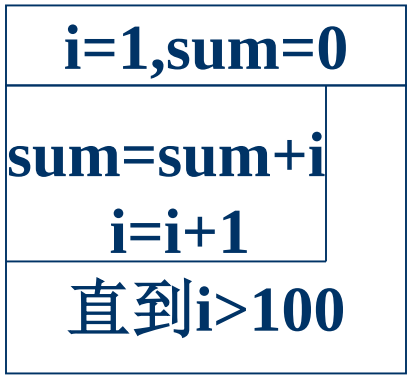
- do-while语句实现“当型”循环结构。
- 一般形式：

```
do
循环体语句;
while(表达式) ;
```

有“;”

- 功能：先执行循环体，然后判断表达式。若为真，则再次执行循环体，否则退出循环。
- 求 $\sum_{n=1}^{100} n$





例 用do-while语句构成循环，求 $\sum_{n=i}^{100} n$

输入：1
输出：
5050　　5050

输入：101
输出：
101　　0

```
#include <stdio.h>
void main( )
{ int i,sum=0;
   scanf("%d",&i);
   do
    {  sum+=i;
       i++;
    }
   while(i<=100);
 printf("%d",sum);
}
```

循环体

循环条件

```
#include <stdio.h>
void main()
{int i,sum=0;
  scanf("%d",&i);
  while(i<=100)
    { sum+=i;
      i++;
    }
 printf("%d",sum);
}
```

比较

- do-while 循环，循环体至少执行一次；
- while和do-while可以解决同一问题，两者可以互换。
- while后的表达式一开始就为假时，两种循环结果不同。

for语句

for语句主要用于实现次数型循环结构。for语句的一般形式：

【格式】for(表达式1;表达式2;表达式3)

语句;

【功能】

1、计算表达式1，实现变量赋初值。

2、计算表达式2，如果其值为非0，则执行3；否则转至4。

3、执行语句，并计算表达式3， 然后转至2。

4、不执行语句，退出for循环，执行for语句的下一条语句。

for语句

【说明】

1、“表达式1”的作用是对循环控制变量赋初值；“表达式2”的作用是控制循环的条件；“表达式3”的作用是循环后修正变量，使循环趋于结束。

2、“表达式1”可以是任何类型，其后有一个分号，“表达式1”可以省略。

3、“表达式2”可以是任何类型，其后有一个分号，“表达式2”可以省略。省略时相当于“表达式2”的值永远为“真”（非0），所以循环将一直执行，这种情况称为“死循环”，一般应避免。

4、“表达式3”可以是任何类型，其后没有分号，“表达式3”可以省略。

for语句

5、“语句”又称为“循环体”，可以是任何语句（除数据定义语句外），通常为复合语句。

6、“表达式1”“表达式2”和“表达式3”均可以省略，甚至三者全部省略，但它们之间的分号不能省略。

7、书写格式上建议以for为准，语句缩进若干格，以便识别重复执行的操作。

8、当“表达式1”和“表达式3”省略时，相当于while循环，即先判断后执行循环体，循环体至少执行0次；当“表达式3”省略时，则循环体内应有改变“表达式2”的语句；当“表达式2”省略时，无终止条件，则循环体内应有跳出该循环的语句，如break、goto、exit()或return等。

for语句

for 语句是 C 语言中**最为灵活，使用最广泛**的循环语句，可完全替代while，do-while语句。

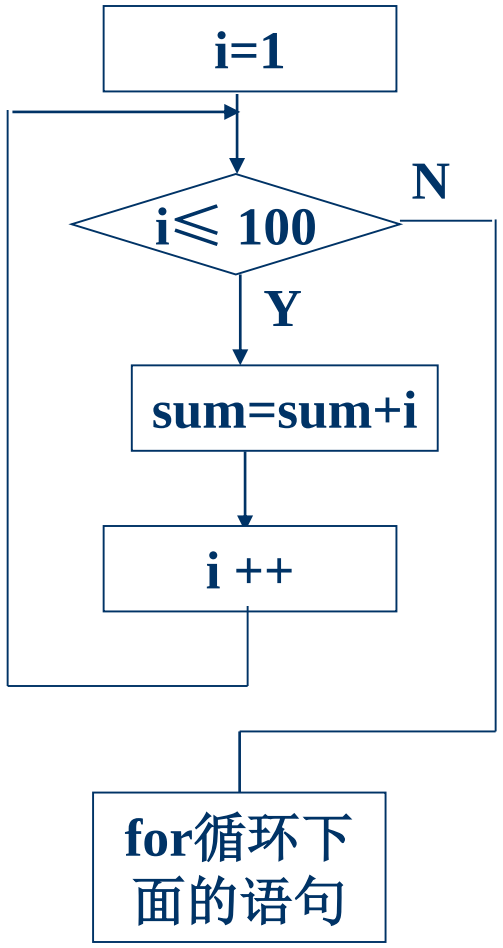
❖一般形式

```
for(表达式1；表达式2；表达式3)
        循环体语句；
```

❖常用形式

```
for(循环变量赋初值；循环条件；循环变量增值)
        循环体语句；
```

例 用for语句构成循环，求 $\sum_{n=1}^{100} n$




```
#include <stdio.h>
void main( )
{
  int i,sum=0;
  for(i=1;i<=100;i++)
     sum+=i;
  printf("%d",sum);
}
```

循环步长

★几点说明：

- for语句中表达式1、2、3类型任意，都可省略，但分号“；”不能省。
- 无限循环：for(;;)不断执行循环体，循环不终止。
- For语句可以转换成while结构

```
表达式1;
while(表达式2)
{
        循环体语句;
        表达式3;
}
```

❖几种形式：

- 省略表达式1：应在for之前为变量赋初值。
- 省略表达式2：循环条件始终为“真”，循环不终止。
- 省略表达式3：应另外设法使程序能够结束。
- 省略表达式1、3：完全等同于while语句。
- 三个表达式都省略：无初值，不判断条件，循环变量不增值，死循环。

```
/*正常形式*/
main( )
{ int i ,sum=0;
  for(i=1;i<=100;i++)
     sum=sum+i;
  printf("%d",sum);
}
```

```
/*省略表达式1*/
main( )
{ int i,sum=0;
   i=1;
   for(;i<=100;i++)
      sum=sum+i
   printf("%d",sum);
}
```

```
/*省略表达式3*/
main( )
{ int i,sum=0;
  for(i=1;i<=100;)
      {sum=sum+i
       i++;}
   printf("%d",sum);
}
```

```
/*省略表达式1、3*/
main( )
{ int i,sum=0;
   i=1;
  for(;i<=100;)
      {sum=sum+i
       i++;}
   printf("%d",sum);
}
```

- **表达式1、3可以是与循环无关的表达式，也可以是逗号表达式。**

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

- **表达式2可以是关系、逻辑、算术、字符表达式，非0时，执行循环体，为0时退出循环。**

```
/*表达式是逗号表达式*/
main()
{ int  i,j,k;
   for(i=0,j=100;i<=j;i++,j--)
   { k=i+j;
       printf("%d+%d=%d\n",i,j,k);
   }
}
```

```
#include <stdio.h>
main()
{  int  i,c;
   for(i=0;(c=getchar())!='\n';i+=c)
          printf("%d ",i+c);
}
```

```
#include<stdio.h>
main()
{  char  c;
    for(;(c=getchar())!='\n';)
       printf("%c ",c);
}
```

break语句

【格式】break;

【功能】强制结束当前的循环，不在执行循环体中break语句后面的语句。

【说明】

1、break语句能用在三种循环语句和switch语句中，其作用是跳出本层循环或跳出switch语句。

2、通常，break语句与if语句联合使用，表明程序在满足一定条件下提前结束循环，在循环体中单独使用break语句是没有意义的。

★Break语句

- ❖功能：在循环语句和switch语句中,终止并跳出循环体或开关体。
- ❖说明：
  - break只能终止并跳出最近一层的结构。
  - break不能用于循环语句和switch语句之外的任何其它语句之中。
- ❖一般形式：break；

例 break举例：小写字母转换成大写字母,直至输入非字母字符

```
#include <stdio.h>
void main()
{char  c;
  while(1)
    { c=getchar();
       if(c>='a' && c<='z') putchar(c-32);
       else
          if(c>='A' && c<='Z') putchar(c);
          else break;
    }
}
```

break 使无限循环结束

continue语句

【格式】continue;

【功能】立即结束本次循环，转去判断循环控制条件是否成立，即跳过continue之后的语句，重新判断循环控制条件，决定是否继续执行循环。

【说明】

1、continue语句只能用于三种循环语句的循环体中。

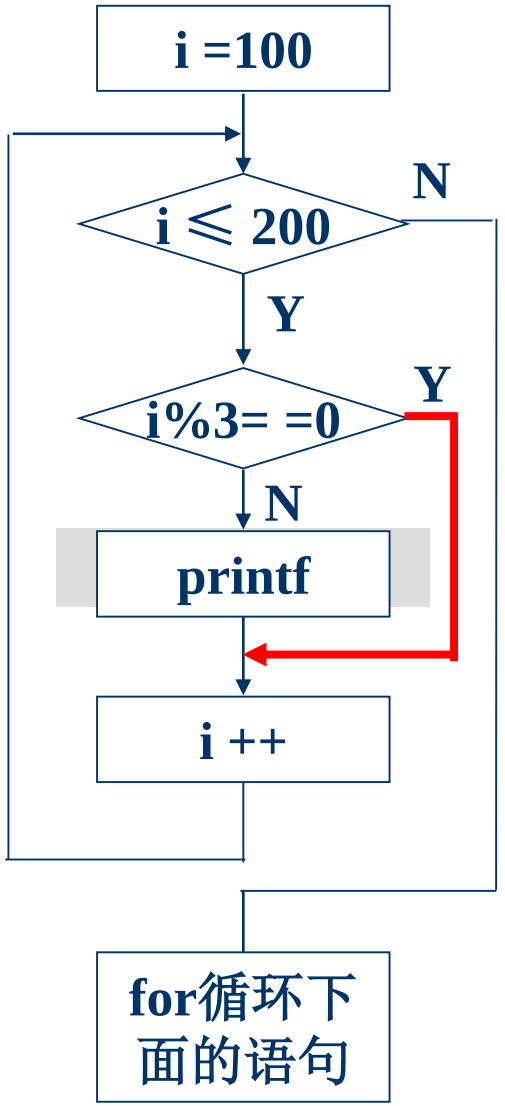
2、通常，continue语句与if语句联合使用。在循环体中单独使用continue语句是没有意义的。

例 把100~200之间不能被3整除的数输出

```
#include <stdio.h>
void main()
{ int i;
   for(i=100;i<=200;i++)
     { if(i%3= =0)
          continue;
       printf("%d",i);
     }
}
```

运行结果：
100，101，
103，104，
106，107，
109，110，
112，113，
115，116，
118，119，

使用continue
结束本次循环



★Continue语句

- ❖功能：结束本次循环，跳过循环体中尚未执行的语句，进行下一次是否执行循环体的判断。
- ❖continue 语句仅用于循环语句中。

break 和 continue 语句的区别

- ❖continue 语句只结束本次循环，break语句则是结束整个循环。
- ❖continue 语句只用于while,do-while,for循环语句中，break语句还可以用于switch语句中。

break语句和continue语句的区别

break语句和continue语句的区别主要体现在：

1、break语句能用于三种循环语句和switch语句，continue语句只能用于三种循环语句。

2、continue语句的作用是强制结束本次循环，而不是终止整个循环过程；break语句的作用是终止整个循环过程，不再判断循环控制条件是否成立。

需要注意的是，循环嵌套时break语句和continue语句均只影响包含它们的最内层循环，与外层循环无关。

循环的嵌套

循环语句的循环体内又包含另一个完整的循环结构称为循环的嵌套。多重循环是嵌套的循环结构，外面的循环语句称为“外层循环”，“外层循环”的循环体中的循环称为“内层循环”。原则上，循环嵌套的层数是任意的。for语句、while语句和do-while语句都允许循环的嵌套。

多重循环中，内层循环必须完整地包含在外层循环的循环体中，不能出现内、外层循环体交叉的情况，但是允许在外层循环体中包含多个并列的内层循环。

课堂练习

3、循环体至少被执行一次的循环语句是_____。

课堂练习

3、循环体至少被执行一次的循环语句是_____。

答案：do-while

解析：do-while语句是先执行循环体，再判断循环控制条件，所以是至少要执行一次循环体。

课堂练习

4、设int x=1，y=2，z=3；，下列语句错误的是（）

A. if（x<y）；

B. for（z=0；z<y；z++）；

C. for（z=0，z<y，z++）；

D. if（x<y）；else z=x+y;

课堂练习

4、设int x=1，y=2，z=3；，下列语句错误的是（）

A. if（x<y）；

B. for（z=0；z<y；z++）；

C. for（z=0，z<y，z++）；

D. if（x<y）；else z=x+y;

答案：C

解析：for语句的一般形式：for（表达式1；表达式2；表达式3），“表达式”与“表达式”之间是用分号进行隔开，而不是逗号。

课堂练习

4、设int x=1，y=2，z=3；，下列语句错误的是（）

A. if（x<y）；

B. for（z=0；z<y；z++）；

C. for（z=0，z<y，z++）；

D. if（x<y）；else z=x+y;

答案：C

解析：for语句的一般形式：for（表达式1；表达式2；表达式3），“表达式”与“表达式”之间是用分号进行隔开，而不是逗号。

祝大家顺利通过考试！