Alan J. Perlis (1922-1990)

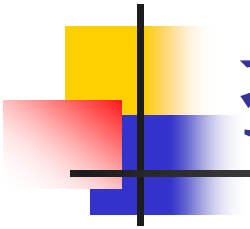
Alan J. Perlis简介

- 出生于美国宾州匹茨堡。1943 年 获 得 CMU的化学学士学位；1949 年和1950年，分别获得MIT 的数学硕士与博士学位
- 第一届图灵奖获得者（1966），为Algol语言和编译器方面作出特殊贡献。ALGOL语言直接导致了PASCAL语言的产生
- 使计算机科学成为独立学科的奠基人。最早在大学开设程序设计课程；CMU计算机科学系的首位系主任（1965-1971）；1971年加入耶鲁大 学计算机科学系 ， 并在1976年到1980年出任计算机系系主任
- 在1962-1964年期间任ACM主席；倡导创办Communications of ACM，并任第一任主编
- 1977年当选美国工程院院士

第4讲
结构化程序设计（Part I）

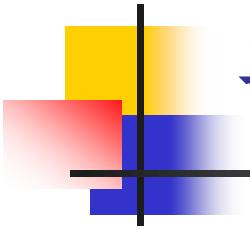
周水庚

2016年10月8日



提要

- 顺序结构
- 选择结构
- 循环结构



语句分类

- C语言的语句可分两类
  - 简单语句
  - 结构语句
- 简单语句：表达式语句、空语句和控制转移语句
  - goto，continue，break，return等
- 结构语句：实现对成分语句的顺序、选择和循环等控制

什么是顺序结构？

- 顺序结构描述一个计算操作序列，表示从序列的第一个计算开始，顺序执行序列中的每个计算，直至计算操作序列的最后一个计算操作
- 在C程序中，顺序执行的语句序列，用花括号括住，就构成C语言的复合语句
- 在逻辑上，复合语句是单个语句，常用作其它结构的成分语句

例子

- 为交换变量x、y的值，可表示为以下顺序执行的三个计算：
  - temp = x;　　/* 将x的值保护到变量temp */
  - x　= y;　　/* 变量x置y值 */
  - y　= temp;　/* 变量y置temp的值 */
- 若把交换变量x、y的值作为一个不可分割的整体来考虑，应把它们写成复合语句：
  - { /* 本复合语句要求外面为它定义temp变量 */
  - 　temp = x ;
  - 　x　= y ;
  - 　y　= temp ;
  - }

关于复合语句

- 在构造复合语句时，为完成复合语句要做的工作，可能需要临时工作变量。如前例中的temp变量
- 在C语言的复合语句中，在语句序列之前可以插入变量定义，引入只有复合语句内的语句可使用的临时变量。如上面的例子改写成以下形式，从逻辑上，它的独立性更强，它不再要求外面为它定义专用变量

```
{ int temp ;  /* 定义自己专用的临时变量 */
    temp = x ;
    x    = y ;
    y    = temp ;
}
```

- 在很多场合，复合语句内会包含其它结构

选择结构与if语句

- 选择结构可分为两路选择结构和多路选择结构
- if语句是一种选择结构，它根据给定的作为选择条件的表达式值为非0或为0两种情况，从两个供选择的成分语句中自动选取一个成分语句执行
- if语句用于描述按条件作两路选择的程序结构

if语句

- if语句的一般形式为

  if（表达式）

  语句$_1$

  else

  语句$_2$

- if语句的执行过程是
  - 计算表达式的值
  - 测试表达式的值并选择执行语句。若表达式的值非0，则执行语句$_1$，并结束if语句；否则执行语句$_2$，并结束if语句

if语句（续）

- 无论条件表达式的值为何值，只能执行语句$_1$或语句$_2$中的一个
- 当if语句中的else后的语句$_2$为空语句时，可简写成：

```
if（表达式）
    语句
```

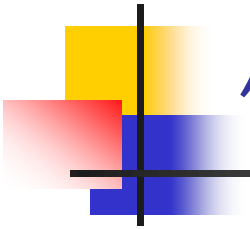
- 这种形式的if语句的执行过程是
  - 计算表达式的值
  - 测试表达式的值。若表达式的值非0，则执行它的成分语句，并结束if语句。否则立即结束if语句

例子

- if (a > b)
  printf("MAX = %f\n", a);
  else
  printf("MAX = %f\n", b);
- if(a != 0 && x/a > 0.5)
  printf("a != 0 && x/a > 0.5\n");
- if (x+y) printf("x + y != 0\n");
- if (!x) printf("x = 0\n");
- if (ch >= 'A' && ch <= 'Z')
  ch = ch + 'a' - 'A';
- if (a < b) a = b ; /* 求a = max(a, b) */

if语句（续）

- if语句中的语句、语句$_1$和语句$_2$可以是任何语句
- 当它们中的某一个需用语句序列描述时，必须将这语句序列写成复合语句
- 当它们中的某一个又是if语句时，就呈现嵌套的if语句形式



例子

- 已知三角形的三条边长a、b、c,求三角形面积

```
if ( a + b > c && b + c > a && c + a > b ) {
  float s ;  /* 必须构成复合语句 */
  s = ( a + b + c ) / 2.0 ;
  area = sqrt(s*(s-a)*(s-b)*(s-c));
}
else  area = 0.0 ;
```

例子

```
if (score >= 90) /* 按得分输出适当信息 */
    printf("Excellent.\n") ;
else if( score < 60 ) /* 嵌套的if语句 */
        printf("Dismal.\n") ;
      else printf("Typical.\n");
```

if语句（续）

- 因if语句中供选择的语句也可以又是if语句，这时，if语句呈嵌套的结构形式，应注意else与if的对应关系

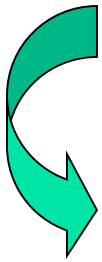
```
if ( 表达式1 )
    if ( 表达式2 )
        语句1
    else
        语句2
```

```
if ( 表达式1 )
        if ( 表达式2 )
                语句1
    else
            语句2
```

if语句（续）

- C语言约定:else总是与它前面最接近的if对应

```
if(表达式1) {/* 将分支语句变成一个复合语句 */
    if (表达式2)  语句1
}
else  语句2
```



```
if (表达式1)
    if (表达式2) 语句1
    else ;   /* 这里有1个空语句 */
else  语句2
```

if语句（续）

- 为了正确书写if语句，需要说明以下几点
  - 若if语句中的语句、语句$_1$、语句$_2$是一个简单语句，则这些简单语句之后会有一个分号，这是C语言对这些简单语句的要求
  - 若if语句中的语句、语句$_1$、语句$_2$要用语句序列（即为顺序结构）来实现，则必须将它们改写成复合语句，即逻辑上把它们变成一个语句
  - 在if语句中，每个else总要与它前面的if对应，不可出现没有对应if的else

例子

- 输入三个整数，输出其中的最大数

```
#include  <stdio.h>
void main()
{ int a, b, c, max;
  printf("输入三个整数\n");
  scanf("%d%d%d", &a, &b, &c);
  max = a;
  if (max < b) max = b;
  if (max < c) max = c;
  printf("最大数是 %d\n", max);
}
```

例子

- 输入三个整数，按值从大到小的顺序输出它们

输入：X Y Z

第一步：X1=Max(X,Y) Y1=Min(X, Y) Z1=Z

第二步：X2=Max(X1,Z1) Y2=Y1 Z2=Min(X1, Z1)

第三步：X3=X2 Y3=Max(Y2, Z2) Z3=Min(Y2,Z2)

```
#include  <stdio.h>
void main()
{ int x, y, z, temp;
   printf("Enter x, y, z.\n");
   scanf("%d%d%d", &x, &y, &z);
   if (x < y) { temp = x; x = y; y = temp;}
   if (x < z) { temp = x; x = z; z = temp;}
   if (y < z) { temp = y; y = z; z = temp;}
   printf("%d\t%d\t%d\n", x, y, z);
}  /* 排序 */
```

```
#include  <stdio.h>
void main( )
{ int x, y, z, *big = &x, *mid = &y, *sma = &z, *temp;
 printf("输入x,y,z.\n"); scanf("%d%d%d",big,mid,sma);

 if (*big<*mid){ temp = big; big = mid; mid = temp; }
 if (*big<*sma){ temp = big; big = sma; sma = temp; }
 if (*mid<*sma){ temp = mid; mid = sma; sma = temp; }
 printf("%d\t%d\t%d\n", x, y, z);
 printf("%d\t%d\t%d\n", *big, *mid, *sma);
}
```

- 求一元二次方程的根

设一元二次方程为$ax^2+bx+c=0$，方程系数a、b和c从键盘输入。对任意的系数a、b、c，有以下几种情况需要考虑：

- a≠0。方程有两个根
- a=0，b≠0。方程退化为一次方程b*x+c=0。方程有根 -c/b
- a=0，b=0。方程或为同义反复(c=0)，或矛盾(c≠0)。

由以上分析得到以下程序结构：

```
{ (1) 输入方程系数a, b, c;
  if (a != 0.0) (2)求两个根;
  else  if (b != 0.0)  (3) 输出方程根:-c/b;
  else  if (c == 0)  (4)输出:"方程同义反复" 字样;
  else  (5)输出 "方程矛盾" 字样;
}
```

用C代码描述其中计算步骤 (1)、(3)、(4)、（5）是简单的事情。对于计算步骤(2)，由代数知识，方程根据判别式 △=b*b-4*a*c大于等于0或小于0，分别有两个实根或复根：

$root_{1,2} = (-b \pm \sqrt{\triangle})/2a, \quad \triangle \geq 0;$

或

$root_{1,2} = (-b \pm \sqrt{-\triangle}i)/2a, \quad \triangle < 0。$

对于两个复根情况，可分别计算它们的实部和虚部。对于实根，也可根据上面给出的公式计算两个根。但是考虑到b*b >> 4*a*c时，有一个根就非常接近零。数值计算中，两个非常接近的数执行减法求出的值的精度很低。为此，先求出一个绝对值大的根root1。然后，利用根与系数的关系

root2 = c/(a*root1)

求出root2

```
#include <stdio.h>
#include <math.h>     /* 使用数学库函数 */
void main()
{ double  a, b, c, delta, re, im, root1, root2;
  printf("输入方程系数a,b,c\n");
  scanf("%lf%lf%lf", &a, &b, &c);
  if (a != 0.0){  /* 有两个根 */
    delta = b*b-4.0*a*c;     re     = -b/(2.0*a);
    im     = sqrt(fabs(delta))/(2.0*a);
    if (delta >= 0.0){/* 两个实根，先求绝对值大的根*/
     root1 = re + (b < 0.0 ? im : -im);
     root2 = c/(a*root1);
     printf("两实根是: %7.5f, %7.5f\n", root1, root2);
    }
```

```
  else /* delta < 0.0  */
    printf("两复根 %7.5f+%7.5fi，%7.5f-%7.5fi\n",
                    re, fabs(im), re, fabs(im));
 }
 else  /* a = 0.0 */
 if (b != 0.0)
       printf("单根 %7.5f\n", -c/b);
 else if (!c)
         printf("方程同义反复.\n");
       else printf("方程矛盾。\n");
}
```

switch语句

- C语言提供switch语句用于描述多路选择情况，其形式是：

```
switch(表达式) {
        case 常量表达式1: 语句序列1 ;
        case 常量表达式2: 语句序列2 ;
                    ...
        case 常量表达式n: 语句序列n ;
        default          : 语句序列n+1 ;
 }
```

switch语句（续）

- switch语句的执行过程
  - 先计算表达式值，以该值与各常量比较，如果表达式的值等于某个常量，switch语句就从该常量后的语句序列开始执行，如没有执行转出该switch语句的语句(如break语句)，自动进入下一个常量后的语句序列继续执行。或执行完最后一个语句序列，或执行了转出该switch语句的语句，就结束switch语句执行
  - 如果没有相匹配的常量，就从以default为情况前缀的语句序列开始执行。如果也没有default情况前缀，则立即结束switch语句

switch语句（续）

- 在C中，“case 常量表达式”只是起语句序列入口位置的作用
- 在执行switch语句时，根据switch后的表达式值找到与该值匹配的入口位置，就从此入口处开始执行，只要未遇到转出该switch语句的break语句或goto语句，就一直向下执行，也不再理会与经过的别的case后的常量值不匹配的情况

switch语句（续）

- 对switch语句需说明以下几点
  - switch后面括号内的表达式只限于是整型表达式或字符型表达式或枚举型表达式
  - case后的常量表达式称为情况前缀，要求所有常量表达式的值互不相同，并与switch后面括号内的表达式值的类型相一致
  - 语句序列由任意条合法的C语句构成，也可以没有语句
  - 情况前缀default可以缺省，但至多出现一次，习惯总是将它写在全部情况前缀之后，如有必要也可写在某case之前

例子

```
char choice;
printf("Enter choice !(A, B, C, ...\n");
scanf("%c", &choice);
switch (choice)  {
   case 'A' : printf(" A chosen!\n"); break;
   case 'B' : printf(" B chosen!\n"); break;
   case 'C' : printf(" C chosen!\n"); break;
   default  : printf("default chosen!\n");
}
```

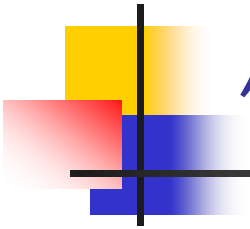
例子

```
int w_con;  /* 天气情况变量定义 */
printf("天气如何? [1:晴天，2:多云，3:下雨] ");
scanf("%d",&w_con);
switch (w_con)
{  case 1 : printf("上街购物!\n");  break;
   case 2 : printf("去游泳!\n");    break;
   case 3 : printf("在家看电视!\n");break;
   default: printf("错误选择!\n");
}
```

例子

- 按照考试成绩的等级打印出百分制分数

```
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");
}
```



例子

- 由于switch语句的表达式不允许是实型的，当应用于实型值选择情况时，通常需作以下处理：将实表达式乘上一个适当的比例因子，将实表达式的值映照到一个较小的范围上，然后再将它转换到整型
- 譬如：收入与赋税的影射关系用(income-1)/200转换

```
switch ((int)((income-1.0)/200.0)) {
    case  0:
    case  1:
    case  2:  tax = income*0.15;      break;
    case  3:
    case  4:  tax = income*0.19-24;   break;
    case  5:
    case  6:  tax = income*0.23-64;   break;
    case  7:
    case  8:  tax = income*0.28-134;  break;
    case  9:  tax = income*0.35-260;  break;
    default:  tax = income*0.43-420;
}
```

(0, 600]

(600, 1000]

(1000, 1400]

(1400, 1800]

(1800, 2000] →

(2000, +∞] →

循环结构

- 循环结构主要由一个称为循环条件的表达式和一个称为循环体的要循环执行的语句组成
- C语言提供描述三种不同的循环结构的语句
  - while语句
  - do_while语句
  - for语句

while语句

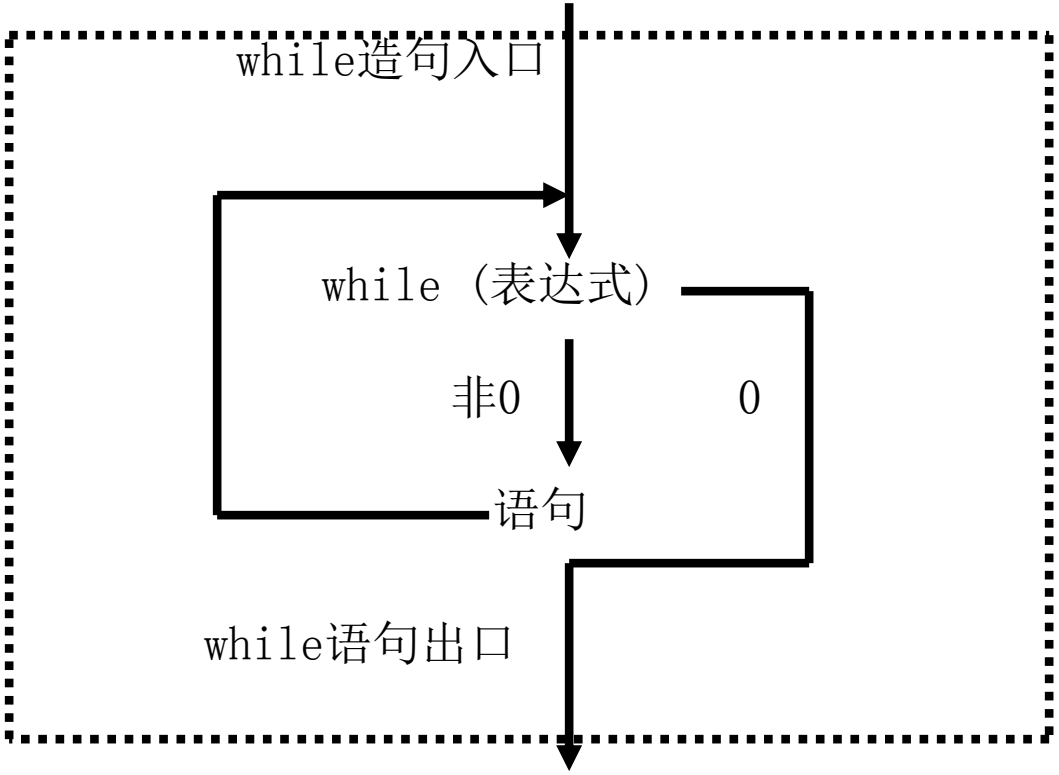
- while语句用来描述while型循环结构
- while语句的一般形式为：

```
while (表达式)
    语句;
    or
    复合语句;
```

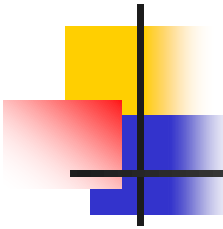
while语句

- while语句的执行过程是
  - (1)计算while之后的表达式的值
  - (2)测试表达式的值，当值为非0时，转步骤（3）；如值为0，则结束while语句
  - (3)执行while语句的循环体，并转步骤（1）

while语句



while语句执行过程示意图



例子

- 求s = 1 + 2 + 3 + … + 100

```
s = 0; i = 1;
while (i <= 100) {
   s += i;
   i++;
}
```

```
s = 0;
i = 0;
while (i < 100)
   s += ++i;
```

while语句

- 如用以下代码实现跳过输入的空白类字符：

```
while((c = getchar())==' '||c=='\n'||c=='\t') ;
```

代码说明当循环体为空时，用空语句代替

- 为使while语句的执行能正常结束，控制循环条件的变量应在循环体内被更新，使表达式的值会变为0
- 有时，很难写出循环条件，可用1代之,而在循环体中有当条件满足时，执行break语句，跳出循环。如：

```
while (1) {
    ...
    if (表达式) break;
    ...
}
```

例子

- **问题**:输入学生成绩，求平均成绩。约定当输入负数时，表示输入结束
- **方法**:采用成绩逐个输入、累计总分和计数学生人数的方法，直到输入成绩是负数时循环结束，然后求出平均成绩，并输出

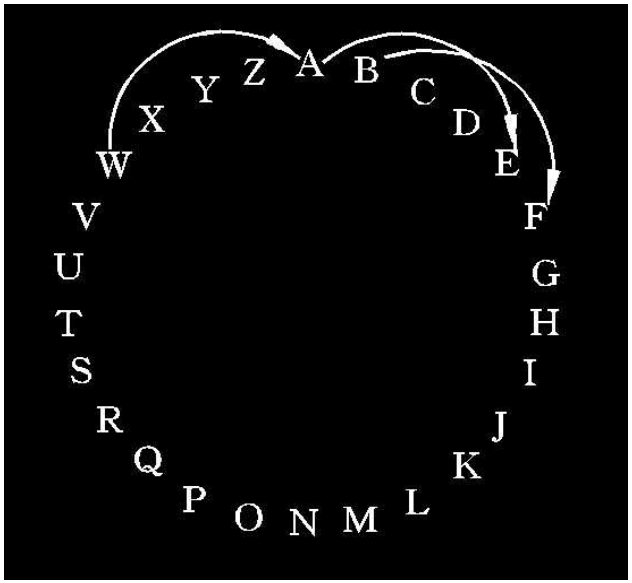
```c
#include  <stdio.h>
void main()
{ int sum, count, mark;
  sum   = 0;  count = 0;
  while (1) { /* 循环条件永远为真 */
    printf("输入成绩(小于0结束)\n"); scanf("%d", &mark);
    if (mark < 0) break; /* 跳出while循环 */
    sum += mark;   count++;
  }
 if(count) printf("平均成绩为 %.2f\n", ((float)sum)/count);
  else printf("没有数据输入.\n");
}
```

例子：统计输入字符行中，空白类字符、数字符和其它字符的个数。

```
#include <stdio.h>
void main()
{ int c, nwhite, nother, ndigit;
  nwhite=nother=ndigit=0; printf("输入字符行\n");
  while ((c = getchar()) != '\n')
   switch ( c ) {
    case '0': case '1': case '2': case '3': case '4':
    case '5': case '6': case '7': case '8': case '9':
              ndigit++; break;
    case ' ': case '\t': nwhite++; break;
    default: nother++; break;
   }
   printf("digit = %d\twhite space = %d\tother = %d\n",
                  ndigit, nwhite, nother);
}
```

例子

- **问题**：为使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文



```
#include <stdio.h>
main() {
    char c;
    while((c=getchar())!='\n') {
      if((c>='a' && c<='z') || (c>='A' && c<='Z')) {
        c=c+4;
        if(c>'Z' && c<='Z'+4 || c>'z') c=c-26;
      }
      printf("%c", c);
    }
}
```

do-while语句

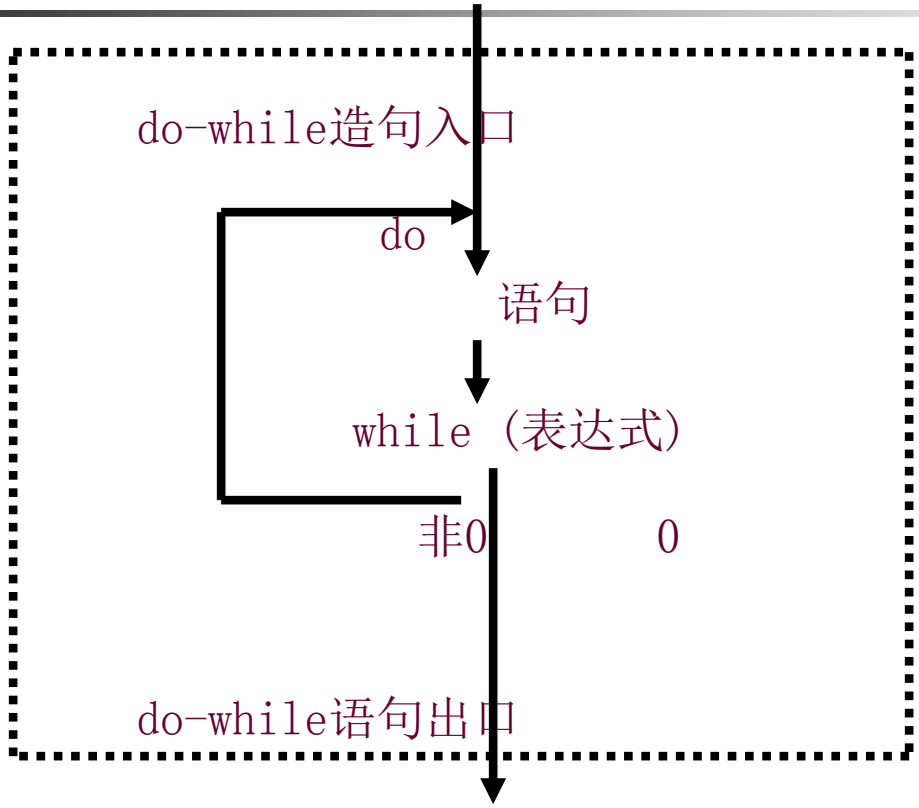
- do-while语句的一般形式为

```
do
    语句
while (表达式);
```

循环体

分号是do-while语句的结束符
决不能省略!

- do-while语句的执行过程是
  - (1)执行do-while语句的循环体
  - (2)求while之后的表达式的值
  - (3)测试表达式的值，当值为非0时，转步骤（1）（从而构成循环）；如值为0，则结束while语句



do-while语句执行过程示意图

例子

- 求 s = 1 + 2 + 3 + … + 100

```
s = 0;   i = 1;
do {
   s += i;     i++ ;
} while (i <= 100);
```

```
s = 0;     i = 1;
do
   s += i++;
while (i <= 100);
```

do-while语句(续)

- 与while语句一样，当循环体由多个语句组成时，必须把它们书写成复合语句
- while语句与用do-while语句的区别是在于执行循环时，对循环条件表达式的求值和测试的时间不同
- while语句对循环条件表达式求值和测试在执行循环体之前，而do-while语句对循环条件的表达式求值和测试在执行循环体之后

do-while语句(续)

- 对于do-while语句，它的循环体至少被执行一次，而while 语句的循环体在循环条件表达式值一开始就为0的情况下，就一次也未被执行
- 如能保证while语句中的循环条件表达式在第一次求值总是非0，则把该循环条件移至循环体执行之后求值和测试，能起同样的控制作用。在这种情况下，while语句就能改写成do-while语句

例子

- **问题**：求方程$f(x)=3x^3+4x^2-2x+5$的实根
- **方法**：
  - 用牛顿迭代方法求方程$f(x)=0$的根的近似解：
  - $x_{k+1}=x_k-f(x_k)/f'(x_k)$，　$k=0，1，\ldots$
  - 当修正量$d_k=f(x_k)/f'(x_k)$ 的绝对值小于某个很小数ε时，x_{k+1}就作为方程的近似解

```
#include <stdio.h>
#include <math.h>   /* 引用数学函数 */
#define Epsilon    1.0e-6
void main()
{ double x, d;
  x = -2.0;
  do {
   d = (((3.0*x+4.0)*x-2.0)*x+5.0)/((9.0*x+8.0)*x-2.0);
   x = x-d;
  } while (fabs(d) > Epsilon);
  printf("The root is %.6f\n", x);
}
```

$f(x)=3x^3+4x^2-2x+5$

$f'(x)=9x^2+8x-2$

$x_{k+1}=x_k-f(x_k)/f'(x_k)$

例子

- **问题**：寻找一个最小整数，要求该整数满足以下条件
  - 被3除余2，被5除余3，被7除余4
- **求解方法**
  - 若采用变量从初值2开始，循环执行变量的值增长1和测试该变量是否不满足问题要求的条件，直至变量的值满足条件结束

```
#include  <stdio.h>
void main()
{ int i = 2;
   do
    i++;
   while(!(i%3 == 2 && i%5 == 3 && i%7 == 4));

   printf("被3除余2，5除余3，7除余4的最小数是
%d\n", i);
}
```

如采用分阶段的办法，先让变量在保证满足条件被3除余2情况下，寻找被5除余3的解；接着在保证被3除余2和被5除余3的条件下，寻找被7除余4的解，这样做能简化循环条件，并能加快找解的速度。

```
#include  <stdio.h>
void main()
{ int i = 2; /* 初值i被3除余2 */
  do  i += 3;
  while (i % 5 != 3);

  while (i % 7 != 4) i += 15;
  printf("被3除余2,5除余3,7除余4的最小数是%d\n\n",i);
}
```

保证i被3除余2

保证i被3除余2并被5除余3

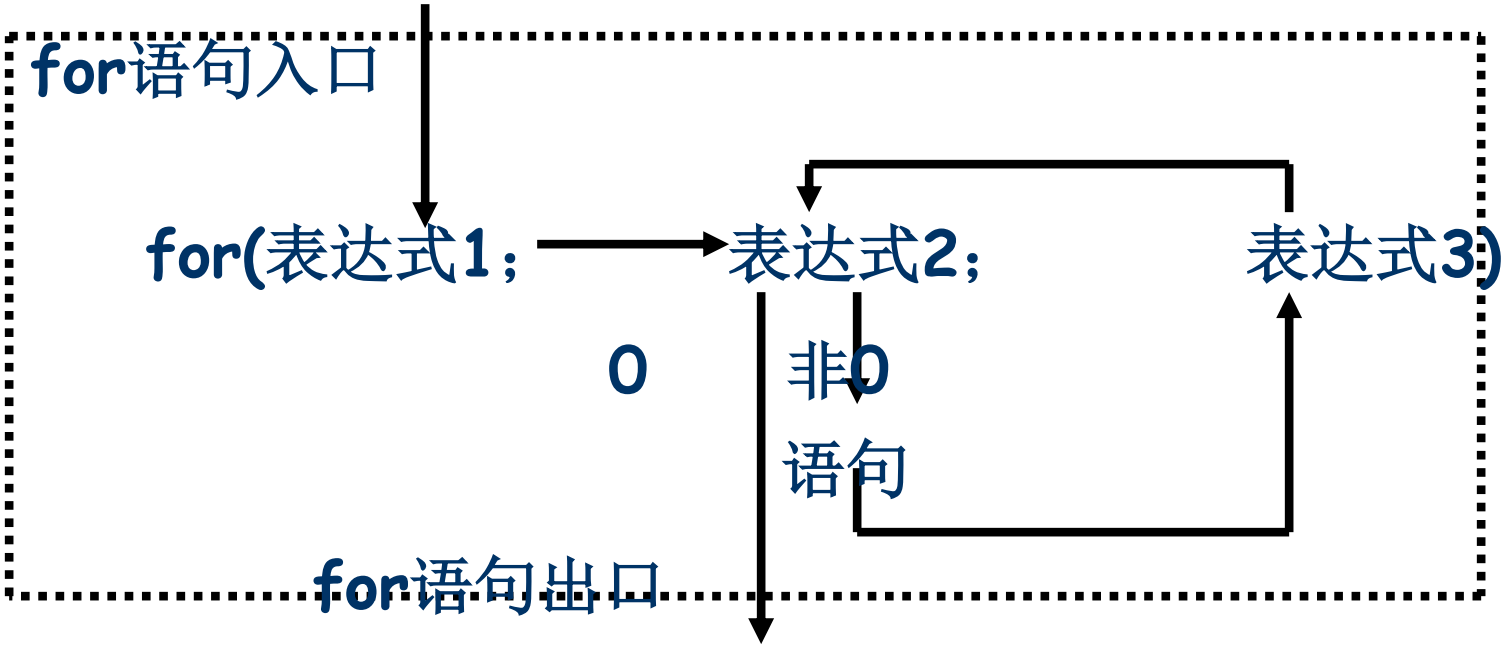
for语句

- for语句的一般形式为

  for(表达式$_1$；表达式$_2$；表达式$_3$)

  语句

- 其中,语句是for语句的循环体，for语句执行过程：
  - (1) 计算表达式$_1$；
  - (2) 计算表达式$_2$的值，并测试其值为0或非0。若值为非0，转步骤（3）；否则结束for语句；
  - (3) 执行循环体；
  - (4) 计算表达式$_3$；
  - (5) 转向步骤（2）。



for语句的执行流程图

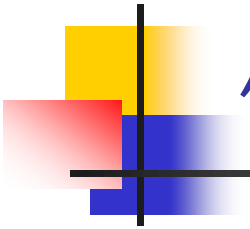
for语句 vs. While语句

- for语句的一般形式也可等价地用以下形式的while语句来表达：

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

- for语句的表达式1的作用是对控制循环的有关变量赋初值；表达式2是控制循环的条件；表达式3用于修正有关变量；语句是循环体
- for语句按各部分的功能，可以形象地写成以下形式：

```
for(初值表达式 ; 控制循环条件表达式 ; 修正变量表达式)
        完成循环计算语句
```



例子

- 求s = 1 + 2 + 3 + … + 100
- 用for语句可以写成：

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

for语句(续)

- 使用for语句时，需注意的几种情况：
  - for语句的一般形式中，表达式$_1$、表达式$_2$和表达式$_3$都可以省略
  - 如表达式$_1$省略，表示该for语句没有赋初值部分，或前面的程序段已为有关变量赋了初值，或确实没有特别的初值；
  - 如表达式$_2$省略，表示循环条件永远为真，可能循环体内有控制转移语句转出for语句；
  - 表达式$_3$省略，表示没有修正部分，对变量的修正已在循环体内一起完成。

例子

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

```
for(s = 0, i = 1; ; i++) {
   s += i;
   if(i == 100) break; }
```

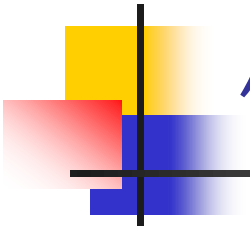
```
for(s = 0, i = 1; i <= 100;)
    s += i++;
```

for语句(续)

- 不管表达式$_1$、表达式$_2$和表达式$_3$省略情况如何，其中两个分号都不能省略。对于三个表达式都省略情况for语句呈以下形式：

```
for( ; ; )
    语句
```

  - 表达式$_1$、表达式$_2$和表达式$_3$都可包含逗号运算符由多个表达式组成。如上面的例子中，表达式1为s=0，i=1。对于s = 1+2+3+…+100的计算，下面都是合理的for语句描述
  - for(s = 0, i = 1; i <= 100; s += i, i++);
  - for(s = 0, i = 1; s += i, i < 100; i++);
  - for(s = 0, i = 0; i < 100; ++i, s += i);



例子

- **问题:**编制一个程序，实现输入n个整数，输出其中的最大数，并指出其是第几个数

```
#include <stdio.h>
void main()
{  int n, i, max, x, index;
   printf("输入 n!\n");    scanf("%d", &n);
   for(i = 1; i <= n; i++) {
     printf("输入第%d个数据.", i);  scanf("%d", &x);
     if (i == 1) {
       max = x;  index = 1;   continue; }
     if ( x > max) { /* 输入了一个更大的数 */
            max = x;
            index = i;
     }
   }
   printf("最大数是 %d，它是第 %d 个数。\n\n", max, index);
}
```

三种循环语句

- while语句、do_while语句和for语句对应于三种不同的构造循环计算的思路
  - 当某条件成立时循环执行某个计算，直至条件不成立结束循环
    - while
  - 循环执行某个计算，直至条件不成立时结束循环
    - do-while
  - 某个（或某些）变量从初值开始，顺序变化，对其中的每一个（或组）值，当条件成立时，循环执行某计算，直至条件不成立结束循环
    - for(;;)

循环语句比较：例子

- 求和式：

  p = 1-1/3+1/5-1/7+……

  - 要求程序按公式逐项累计求和，直到某项的绝对值小于0.000001时结束
  - 记公式中项的分母为d，d的初值为1；引入存储当前项值的变量t
  - 用三种循环实现代码

while

- 以当前项t的绝对值不小于0.000001时循环求和，并修正变量d和t，有代码：

```
p = 0.0; d = 1; t = 1.0;
while (fabs(t) >= 1.0e-6) {
  p += t;
  d += 2;
  t = ((d-1)%4）? -1.0/d : 1.0/d;
}
```

do-while

- 循环求和，并修正变量d和t，直至新的t的绝对值小于0.000001时结束，有代码：

```
p = 0.0; d = 1; t = 1.0;
do {
    p += t;
    d += 2;
    t = ((d-1)%4）? -1.0/d : 1.0/d;
} while (fabs(t) >= 1.0e-6);
```

for(;;)

- 从变量d为1.0、t为1.0开始，当t的绝对值不小于0.000001时循环求和，每次循环后相应地修正变量d和t，有代码：

```
for(  p = 0.0, d = 1, t = 1.0;
      fabs(t) >= 1.0e-6;
      d += 2, t = (d-1)%4）? -1.0/d : 1.0/d)
          p += t;
```

循环语句比较(续)

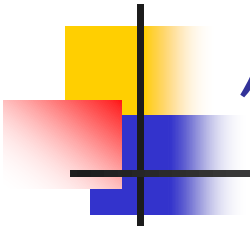
- 三种循环语句在代码编写时有一些差别
  - while语句和for语句循环条件表达式求值和测试在前，执行循环体在后。极端情况，循环体可能一次也没有执行。do_while语句执行循环体在前，循环条件表达式的求值和测试在后，因此，循环体至少执行一次
  - do_while语句的最后要接上一个分号，这是其句法的要求。while语句和for语句如果循环体是一个表达式语句,最后也会以分号结束，但这个分号是该表达式语句的要求
  - while语句或do_while语句，通常在它们之前会有赋值语句给有关变量赋初值，在循环体中有对有关变量值的修正
  - 而for语句的赋初值部分应出现在表达式$_1$中，变量修正出现在表达式$_3$中。所以for语句更加灵活，应用也最多。

Break和continue

- 前面，break语句可用于使流程跳出switch结构，继续执行Switch语句下面的一个语句。这里，break语句还可以用来从循环体内跳出循环体，即提前结束循环，接着执行循环语句下面的语句
- continue一般形式为：continue;
  - 其作用为结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判定
- continue语句和break语句的区别
  - continue语句只结束本次循环，而不是终止整个循环的执行
  - break语句则是结束整个循环过程，不再判断执行循环的条件是否成立

嵌套的循环结构

- 循环结构的嵌套：一个循环结构的循环体中又包含一个完整的循环结构
- 在实际应用中，三种循环语句可以相互嵌套，会呈现多种复杂形式
- 在编写有循环嵌套的程序时，要注意各层次上的控制循环的变量的变化规律



例子

- **问题：**已知直角三角形每边长为25以内的整数，求出所有这样的直角三角形三边长
- **思路：**设三边长为a、b、c，且c > b >= a。用三重嵌套的for语句实现
  - 设最外层的c的变化范围是3至25
  - 对于确定的c、内层的b的取值范围是1至c-1；最内层的a的取值范围是1至b

```
#include <stdio.h>
void main()
{ int a, b, c;
  for(c = 3; c <= 25; c++)
    for(b = 1; b < c; b++)
      for(a = 1; a <= b; a++)
        if(a*a + b*b == c*c) {
          printf("A=%d\t B=%d\t C=%d\n",a,b,c);
          scanf("%*c"); /* 等待输入回车 */
        }
}
```

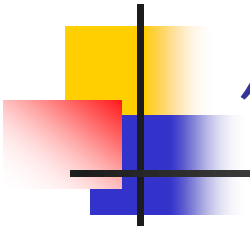
例子

- 问题：甲，乙，丙三位球迷分别预测已进入半决赛的四队A、B、C、D 的名次如下：
  - 甲预测：A第一名、B第二名；
  - 乙预测：C第一名、D第三名；
  - 丙预测：D第二名、A第三名。
  - 设比赛结果，四队名次互不相同，且甲、乙、丙的预测各对一半，求A、B、C、D四队的名次

例子

- 分析
  - 令变量a、b、c、d分别表记四队的名次，对四队所有可能的名次组合作循环测试，就能找到解。为了表达预测的条件，以甲的预测条件为例。因其预测对了一半，所以若A是第一名，则B不会是第二名；或者A不是第一名，则B必须是第二名
  - 按此说法，此条件可用逻辑运算与关系运算描述如下：
    - a == 1 && b != 2 || a != 1 && b == 2
  - 条件“A是第一名”和条件“B是第二名”不允许同时成立，即条件a==1和b==2不允许相等。这样预测对一半的条件可写成
    - （a == 1) != (b == 2)
  - 对乙、丙也可写出类似的表达式

```
#include <stdio.h>
void main()
{ int a, b, c, d, t;
  for(a = 1; a <= 4; a++)
    for(b = 1; b <= 4; b++) {
      if(b == a) continue;
      for(c = 1; c <= 4; c++) {
        if(c == a || c == b) continue;
        d = 10-a-b-c;  /* 四队名次之和为10 */
        t = ((a==1)!=(b==2)) && ((c==1)!=(d==3))
                  && ((d==2)!=(a==3));
        if(t) printf("A=%d,B=%d,C=%d,D=%d\n",a,b,c,d);
      }
    }
}
```



作业

- 习题三
  - 1，5，13， 18， 20