疑难讨论

- 练习题5. 1−3. d

```
i=1;j=1;k=1;
printf("%d",++i||++j&&++k);
printf("%d %d %d",i,j,k);
```

- 结果短路计算2, 1, 1
- 原因短路规则 高于 运算符优先级规则
- 节省计算资源

第六章

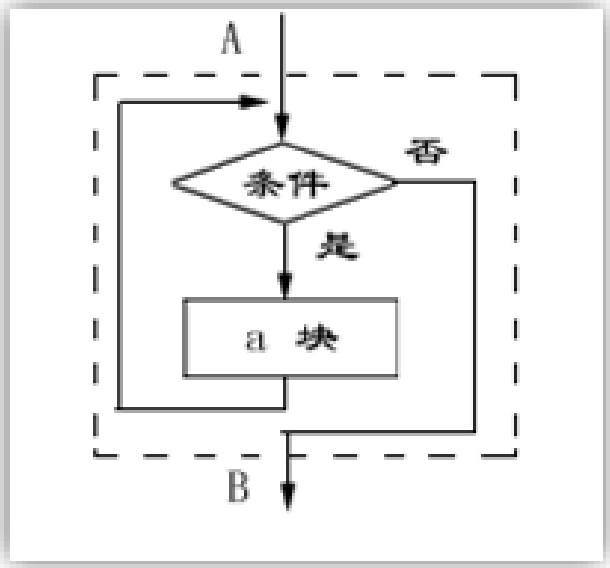
循环

内容提要

- 三种循环语句：
- while
- do-while
- for

循环结构与循环语句

- 循环结构：
  - **重复执行某任务的程序结构**
- 循环语句：
  - **设置循环，实现循环结构。**
- 循环体：
  - **重复执行的语句**
- 循环条件：
  - **控制表达式，每次执行循环体时测试控制表达式**
    - 真——继续循环
    - 假——中止循环



循环语句类比示例

- eg.　食堂打饭程序（师傅执行）
- /*伪代码，算法*/

只要(当)（有同学排队）

{

打菜;

打饭;

刷卡;

}

循环语句类比示例

- c程序：

```
while(p > 0)
{
    sell_meat();
    sell_rice();
    swipe_card();
} //复合语句
```

循环条件

循环体

循环语句

- C提供三种循环语句:
- while语句：先测试，后循环（先谈条件，再做事）
  - **先看是否有人等待**
- do语句(do while)：先循环（1次），后测试（先做事，再谈条件）
  - **先打饭菜，后测试**
- for语句：计数循环
  - **定量打饭**

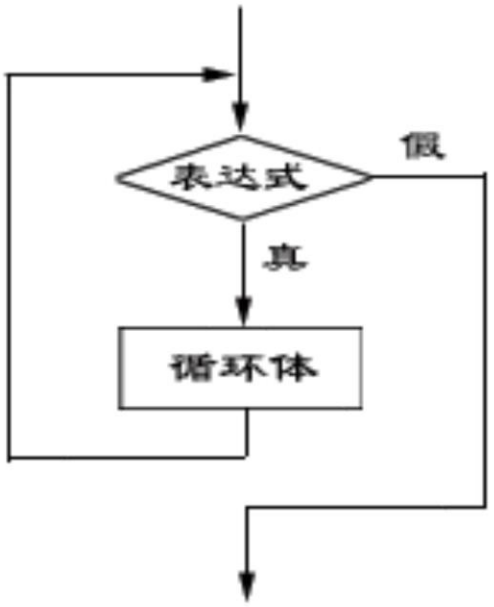
while语句——当型循环

- 最简单、最基本的循环
- 格式：

  while (表达式)

  语句

- 表达式：控制表达式；
- 语句：循环体。



(1) 计算表达式的值；

(2)真(非0)，执行循环体，

重复上面步骤，

(3)直到为假(0)，结束循环。

while语句——示例：

- 大于或等于n的最小的2的幂（等于或刚大于n的2^k）
  - **$2^{k-1} < n \&\& 2^k >= n$//无法数学求解**
- 思路：不断测试逼近，i < n && i*2 >=n
  - **i = 1（2^0） < n？**
  - **i *= 2（2^1） < n？**
  - **i *= 2（2^2） < n?**
  - **……**
  - **i *= 2（2^{k-1}）< n**
  - **i *= 2（2^k）>= n**
- 表达
  - **i跟踪（表示）2^x，从1（2^0）开始，**
  - **当i<n时循环，i *=2（求下一个2的幂），**
  - **当i<n不成立结束循环，当前i为结果**

while语句——示例：

- while语句：

```
int i = 1; // 2^0循环初始化
while (i < n)
    i = i * 2; // i *=2;
```

- 说明
  - 控制变量：i
  - 控制表达式：i < n
  - 循环体：i = i * 2
- 执行过程
  - 先计算控制表达式
  - true：执行循环体，再次判定表达式。
  - false：不执行循环，循环终止

while语句

- 示例“倒数计数”

```
i = 10;//循环初始化
while (i > 0) {//循环条件i>0
        printf("T minus %d and counting\n", i);
        i--;}
```

- i减为0时，循环终止
- 复合语句循环体(原子操作)，思考没有大括号后果；eg. 食堂师傅打饭

```
while(p > 0)
        sell_meat();
        sell_rice();
        swipe_card();
```

- 始终大括号括住循环体更保险：

循环结构编写攻略

- 1）构造循环体
  - **动作、变量等，常复合语句**
- 2）控制循环
  - **开始（初始化）：循环什么时候或条件开始;**
  - **条件（继续，循环结束）：什么条件下循环执行;**
  - **变化（效果）：每次循环执行后应体现循环效果或变化**
- 循环开始、条件和变化一般通过一个变量来设置——循环控制变量

```
开始(初始化);
while(条件)
{
    循环体(含控制变量处理，变化);
}
```

关于while语句的一些讨论

- 循环条件：逻辑表达式、算数表达式
  - **一般应含循环控制变量，如i < n，i>0**
  - **值通常从真变假，真假切换时终止**
- 先判定控制表达式
  - **可能根本不执行while循环体**
- 多种写法，更简明递减计数循环：

```
while (i > 0)
    printf("T minus %d and counting\n", i--);
```

无限循环

- while语句无法终止：
  - **控制表达式始终非零，eg.循环体忘了复合语句**
- 常量可作为控制表达式，如：
  - **while (3)语句1**
  - **while (0)语句2**
- 合法，不合理
- C语句程序员有时故意构造无限循环：
  - **while (1) …**
  - **eg，ATM**
- 循环体含有跳出循环控制的语句(break, goto, return)或调用了导致程序终止的函数

程序：显示平方值的表格

- while语句显示平方值表格。
- 用户指定平方值表格的行数：

```
This program prints a table of squares.
Enter number of entries in table: 5
        1         1
```

开始（从哪里开始）：	i=1;
条件（继续，到哪里结束）：	while(i<=5){
循环体（控制变量）：	printf("%10d%10d\n",i,i*i);
变化（控制变量更新）：	i++;}

循环初始化(while前)

square.c

```
int i, n;

printf(“This program prints a table\
 of squares.\n");
printf("Enter number of entries\
 in table: ");
scanf("%d", &n);

i = 1;
while (i <= n) {
    printf("%10d%10d\n", i, i * i);
    i++;
}

return 0;
```

程序：数列求和

- 对用户输入的整数数列求和：

  输入：8 23 71 5 0

  计算：8+23+71+5

- 数列项数不确定
  - **不能事先定义变量记录**
- 思路：循环
  - **输入一项，累加一项，直到输入0**
  - **循环scanf读取用户输入数**

程序：数列求和

- 循环体：
  - **输入数项：**
    - scanf(“%d”,&n);//int n;
  - **累加：**
    - sum=sum+n;//int sum;
- 循环开始：
  - **scanf(“%d”,&n);**
- 循环条件：
  - **输入0表示数列结束**
  - **while(n!=0)**

```
int n, sum=0;

Printf……//提示
scanf("%d", &n);
while (n != 0)
{
    sum += n;
    scanf("%d", &n);
}
```

首项在循环前获取，所以循环体先累加

程序练习——数列求均值

- 计算平均成绩
  - **输入数列**
  - **输入0结束**
  - **求数列均值**

程序练习——数列求均值(伪代码)

输入数(考试成绩)

while（输入不为0）{

累加

累计输入项数

输入下一个成绩

}

计算均值

程序练习——数列求均值

```
int n, c=0, sum = 0, average;//初始化
printf("Enter integers (0 to terminate):
\n");
scanf("%d",&n);
while(n !=0)
{
    sum += n;
    scanf("%d", &n);
    c++;
}
average = sum/c;
printf("the average is %d\n", average);
```

先录入第一个数
构造循环条件

累加

累计项数

程序练习——求1到n的累加和与阶乘

- 输入n
- 累加
  - 循环体:sum += i;
  - 初始：i=1;
  - 条件：i <= n;
  - 更新：i++;
- 累积
  - 循环体：product *= i;
  - 初始：i=1;
  - 条件：i <= n;
  - 更新：i++;

程序练习——求1到n的累加和与阶乘

- 累加

```
sum = 0;
i = 1;
while(i<=n) sum += i;
```

- 累积

```
product =1;
i = 1;
while(i<=n) product *= i;
```

程序练习——求1到n的累加和与阶乘

```
sum = 0;
product = 1;

scanf("%d",&n);
i = 1;
while(i<=n){
    sum += i;
    i++;
}

i = 1;
while(i<=n){
    product *= i;
    i++;
}
```

开始相同
继续相同
变化相同
—循环次数/控制相同
可合并

程序练习——求1到n的累加和与阶乘

```
sum = 0;
product = 1;
scanf("%d",&n);
i = 1;
while(i<=n){
    sum += i;
    product *= i;
    i++;
}
```

while型-数列求和

```
int n, sum=0;
scanf("%d", &n);
while (n != 0)
{
    sum += n;
    scanf("%d", &n);
}
```

1. 循环体：scanf("%d",&n);
2. 初始：scanf("%d",&n);

循环体至少执行一次，do-while

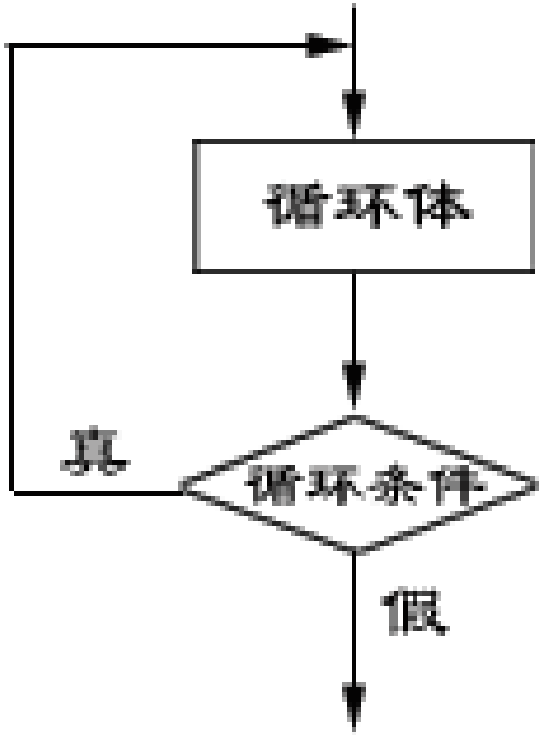
do语句do-while（直到型）

一般形式：

```
do
   语句
while(表达式）;
```

分号不能少

执行过程：



1. 执行循环体中的语句；
2. 计算表达式，测试循环条件，真(非0)则重复上面步骤，假(0)结束循环。

数列求和

- While版

```
int n, sum=0;
scanf("%d", &n);
while (n != 0)
{
    sum += n;
    scanf("%d", &n);
}
```

- Do while版

```
int n, sum=0;
do
{
  scanf("%d", &n);
  sum += n;
}while (n != 0);
```

do语句

- do语句和while语句没本质区别。
  - **do语句的循环体至少会被执行一次。**
- do语句很容易被误认为是while语句

```
do
  printf("T minus %d and counting\n", i--);
while (i > 0);
```

- 最好使用大括号包括循环体

程序：计算整数的位数

- 计算用户输入的整数的位数：

  Enter a nonnegative integer: 60

  The number has 2 digit(s).

- 方法：
  - **将输入整数反复除以10，直到结果变为0，除的次数就是整数的位数。**
  - **eg. 534**
- do语句比while语句更适合
  - **控制条件：n>0**
  - **即便输入0，也是一位整数，需执行一次循环（除一次）。**

numdigits.c

```
#include <stdio.h>
int main(void)
{
  int digits = 0, n;
   printf("Enter a nonnegative integer: ");
   scanf("%d", &n);
  do {
      n /= 10;
      digits++;
   } while (n > 0);
   printf("The number has %d digit(s).\n",
  digits);
   return 0;
}
```

温故而知新——编写正确的循环结构

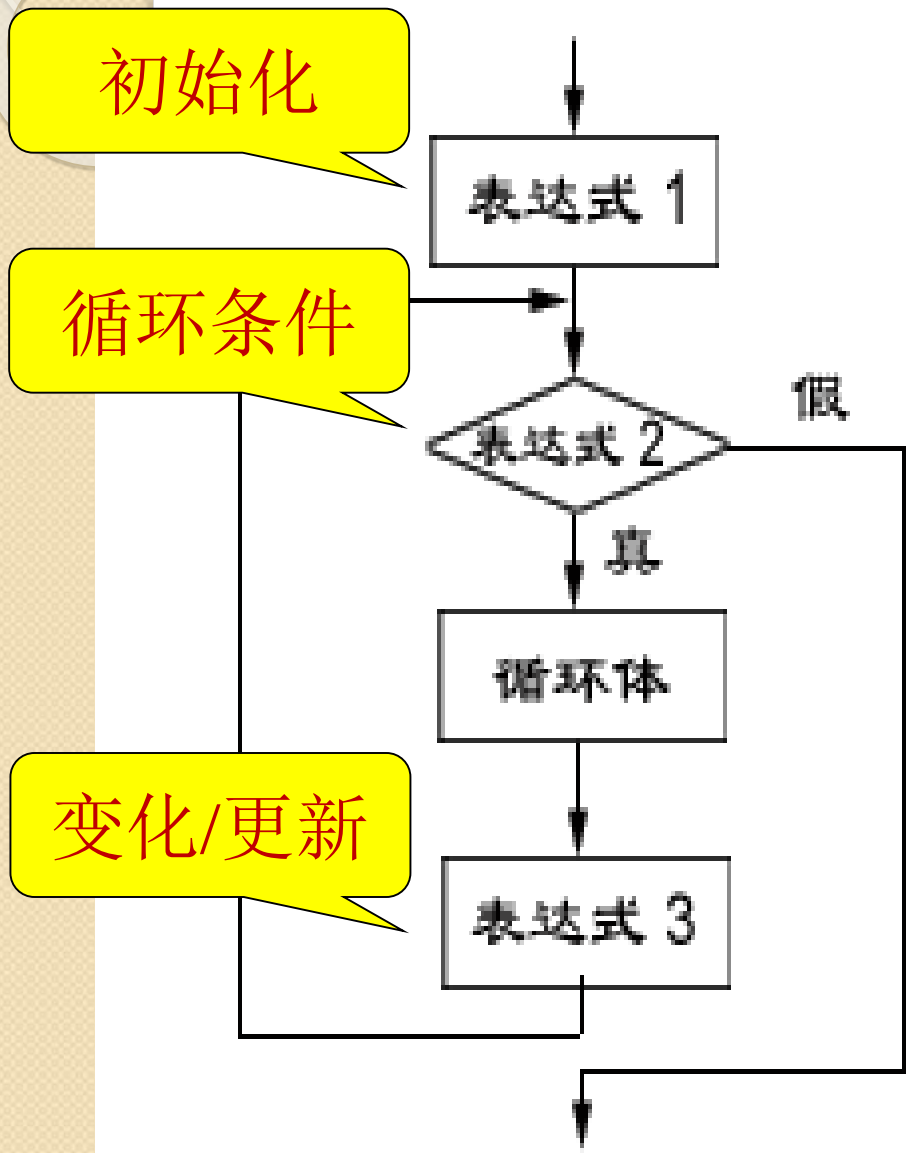
- 1、确定循环体，eg数列求和
  - **scanf（新数列项）**
  - **sum+=新数列项**
- 2、控制循环，操作控制变量：
  - **开始：控制变量赋初值;**
    - i =l ;
  - **继续：把控制变量写入正确的循环条件(expr);**
    - i<n(平方值表) , i>0 (倒计数), n!= 0(数列)
  - **变化：更新控制变量体现循环效果**
    - i++； i--； scanf(“%d”, n);

for语句

- 计数循环：循环次数确定
- 也灵活用于其他类型的循环中。
- 一般格式如下：
  - **for(表达式1;表达式2;表达式3）语句**

不能省略

for执行过程：



- (1)求表达式1的值
- (2)判断表达式2
  - **为假退出循环，**
  - **否则转(3)；**
- (3)执行循环体中语句
- (4)执行表达式3
- (5)转向(2)。

for语句

for （表达式1；表达式2；表达式3）
　　语句

- 表达式2
  - **循环继续条件**
  - **逻辑表达式，真or假**
- 表达式1和表达式3以语句的方式执行
  - **表达式1，执行1次，初始化循环，常为赋值表达式**
  - **表达式3，每次循环反映变化，常为自增/自减表达式。**

for语句

- for语句和while语句关系紧密。
- 除了极少数情况，for循环总可以等价替换为while循环：

while循环：

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

for循环：

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

for语句“倒数计数”程序

- While版

```
i = 10;
while (i > 0) {
    printf("T minus %d and counting\n", i);
    i--;
}
```

表达式1

表达式2

表达式3

- For版

```
for (i = 10; i > 0; i--)
  printf("T minus %d and counting\n", i);
```

循环互换

- 除极少数情况，三种循环可相互替换

while循环：

```
i=10;
while(i>0)
{
        printf…;
        i--;
}
```

三个表达式：

i=10：初始化循环；

i>0：控制循环条件

i--：控制变量更新

for循环：

```
for(i=10;i>0;i--)
{
        printf…;
}
```

do while循环：

```
i=10;
do
{
        printf…;
        i--;
}while(i>0)
```

for语句惯用法

- for是“向上加”（自增）或“向下减”（自减）循环的最好选择
- 共有n次的情况：
  - **从0向上加到n–1：for (i = 0; i < n; i++) …**
  - **从1向上加到n：for (i = 1; i <= n; i++) …**
  - **从n-1向下减到0：for (i = n - 1; i >= 0; i--) …**
  - **从n向下减到1：for (i = n; i > 0; i--) …**

for语句-常见语句错误(控制表达式)

- <，>用反
  - **“向上加”——设上限：使用<或<=**
  - **“向下减”——设下限：使用>或>=**
- 谨慎用==代替<，<=，>，或>=
  - **循环条件初始要为真，**
  - **while（i==n）无意义**
- 注意边界，避免循环次数偏差
  - **用i<=n代替i<n，循环次数差一次，反之多一次**

在for语句中省略表达式

- C语言允许省略for任意或全部的表达式
- 省略表达式1，没有初始化，需前置

```
i = 10;
for (; i > 0; --i)
  printf("T minus %d ……\n", i);
```

初始化前置

在for语句中省略表达式

- 省略表达式3，需由循环体完成循环控制变量更新：

```
for(i = 10; i > 0;)
  printf("T minus %d ……\n", i--);
```

- 或

```
{
    printf("T minus %d ……\n", i);
    i--;
}
```

循环更新下移

在for语句中省略表达式

- 省略表达式1, 3
- for循环与while循环没有任何区别：

```
for (; i > 0;)
        printf("T minus %d ……\n", i--);
```

- 等价于

```
while (i > 0)
        printf("T minus %d ……\n", i--);
```

- while更清楚，更可取。

在for语句中省略表达式

- 省略表达式2
- 默认为真：for无限循环（除非以某种其他形式停止）
- 例如：

  for (;;) …

C99中的for语句

- 表达式1可为一个声明（定义），允许在for循环中定义变量：

```
for (int i = 0; i < n; i++)
        {…}
printf("%d", i);
```

illgal

- 专用循环控制变量，循环外不可见
- 通常是个好的做法，方便且让程序易于理解。

逗号运算符（任意元）

- 连接任意表达式构成逗号表达式：
  - **表达式1，表达式2，……**
- for中可多个初始化表达式，或多个变量进行更新（自增或减）。
  - **for（逗号表达式1;表达式2;逗号表达式3），Eg:**

```
for (sum = 0, i = 1; i <= N; i++)
    sum += i;
for (sum = 0, i = 1; i <= N; sum += i,i++);
```

- 为增强程序的可读性
  - **一般不把循环无关的东西放到for语句中**

逗号运算符

- 表达式1，表达式2
- 左结合，以最右表达式值作为整个逗号表达式结果。eg，
  - **i = l, j = 2, k = i + j**
  - **l）i = l**
  - **2）j = 2**
  - **3）k = i + j**

程序：显示平方值表格（改进版）

```
#include <stdio.h>

int main(void)
{
  int i, n;

  printf("prints a table of squares.\n");
  printf("Enter number of : ");
  scanf("%d", &n);

  for (i = 1; i <= n; i++)
    printf("%10d%10d\n", i, i * i);

  return 0;
}
```

显示平方值表格（改进版）

- for语句的灵活性：
  - **for三个表达式通常对同一个变量进行初始化、判定和更新**
  - **while、do亦然**
- 但C语言没有任何限制，不要求它们之间以任何方式进行相互关联。
- 程序square3.c 等价于square2.c，

```
#include <stdio.h>
int main(void)
{
  int i, n, odd, square;

  printf("This program prints a tabl
  squares.\n");
  printf("Enter number of entries in
  scanf("%d", &n);
  i = 1;
  odd = 3;
  for (square = 1; i <= n; odd += 2) {
    printf("%10d%10d\n", i, square);
    ++i;
    square += odd;
  }

  return 0;
}
```

1	1	+3
2	4	+5
3	9	+7
4	16	+9

难于理解

初始化一个变量(square)，
判定另一个变量(i)，
对第三个变量(odd)进行自增操作。

退出循环

- 常规终止（循环不再继续）：
  - 由（循环控制表达式）控制，while或for循环体之前，或do循环体之后。
- 非常规终止：
  - break语句，return，exit，goto等，
  - 在循环体中间设置退出点，甚至设置多个退出点,eg.

```
while（队列不为空）{
    打卡;
    打饭;
    打菜;
    if(菜卖完||机器故障||师傅故障）结束循环;
}
```

break语句

- 从switch中转移程序控制出来，还可用于跳出循环。
- eg.保研基本条件：没有挂科

```
n = 20;//20门课
for (i = 0; i < n; i++)
   if (s[i] < 60) break;//挂科出局
//循环结束，正常或非正常
if (i < n)//循环非正常终止，有挂科
   printf("你有挂科，出局\n");
else
   printf("%你有基本资格\n");
```

break语句

- 检测数n是否素数
  - 从2开始循环测试是否是n的约数，只要发现一个约数就用break终止循环
  - 2（3更好）到n-1（或n/2,sqrt(n)）去除n，余数为0是约数

d*d<=n;

```
for (d = 3; d < = sqrt(n); d++)
   if (0 == n % d) break;//合数
if (d <= sqrt(n))
   printf("%d is divisible by %d\n", n, d);
else
   printf("%d is prime\n", n);
```

break语句

- 适合构造在循环体中的退出
- eg，读入用户输入并且在遇到特殊输入值时终止的循环，如0，某字符’x’
- 读入数据，计算立方值，并重复该过程

```
for (;;) {
    printf("Enter a number (enter 0 to stop): ");
    scanf("%d", &n);
    if (n == 0) break;
    printf("%d cubed is %d\n", n, n * n * n);
}
```

break语句

- 当出现嵌套时，break语句只能跳出一层嵌套：

```
while (…) {
  switch (…) {
    …
    break;
    …
  }
}
```

- 从switch中转出程序控制，但没有从while循环中转移出来。

continue语句

- break结束循环（跳出循环）
- continue结束当前循环体，进入下一次循环（程序控制留在循环体内）。
- eg，食堂打饭
  - **break：菜卖完||师傅故障**
  - **continue：当前同学故障（卡没钱了），跳过（结束）当前循环（当前同学打饭），进入下一个同学的循环**

break PK continue

```
while（有人排队）{
    语句1
    if(饭菜买完) break;//??
    if(同学卡没钱) continue;//??
    语句2
}
语句3
```

continue语句

- 示例：统计10项的数列之和，0不计入

```
n = 0;
sum = 0;
while (n < 10) {
        scanf("%d", &i);
        if (i == 0)
          continue;
        sum += i;
        n++;
        /* continue jumps to here */
}
```

break;

跳到这里
跳出循环;

continue语句

- 不用continue语句的相同循环示例：

```
n = 0;
sum = 0;
while (n < 10) {
        scanf("%d", &i);
        if (i != 0) {
          sum += i;
          n++;
        }
}
```

goto语句

- goto语句格式如下：
  - **goto 标识符（L）；**
- 跳转到放置“标识符”的语句处。
  - **标识符（L）：语句**
- 一条语句可以有多个标号。
  - **一个单位多个牌子**
- 执行语句 goto L，把程序控制转移到标号L后的语句上
- 转移目标语句必须与goto语句在同一个函数中。

goto语句

- goto Lable;//可以跳转到任意位置，跳出多重嵌套

```
{
    ……
    Lable: 语句
    ……
    if（expr）goto Lable;
    ……
}//同一函数中
```

goto语句

- 如果没有break语句，goto语句可以用于退出循环：

```
for (d = 2; d < =sqrt(n); d++)
  if (n % d == 0) break;//合数
if (d <= sqrt(n))
  printf("%d is divisible by %d\n", n, d);
else
  printf("%d is prime\n", n);
```

```
for (d = 2; d < =sqrt(n); d++)
    if (n % d == 0) goto done;
done: if (d < n)
   printf("%d is divisible by %d\n", n, d);
else
    printf("%d is prime\n", n);
```

goto语句

- break、continue和return 等语句和exit函数，本质上是受限制的goto语句，足够应付程序控制跳转
- 尽量少用，避免程序控制跳跃

goto语句

- 从switch中跳出循环：

```
while (…) {
  switch (…) {
    …
    goto loop_done; //can break work
    …
  }
}
loop_done: …
```

- goto语句在需要从嵌套的多层循环中转出时还是很有用。

程序：账本结算

- 基于菜单的交互式程序：（ATM操作）
- 显示命令列表。
- 读入用户命令，执行相应的操作**（对号入座）**，然后提示用户输入下一条命令**（循环）**。
- 持续到用户选择“退出”或“停止”命令。

```
显示命令列表
for（;;）{//用户多次执行操作
   提示用户输入命令;//0清空、1存、2取、3查、4退
   读入命令;
   switch（命令）{
   case 命令：执行命令;
   }
}
```

空语句

- 什么都不做，只有“；”结尾符
- 编写空循环体。
- 凉拌鸡：

```
cook（）；
while（chicken is hot）；
mix（）；
```

空语句

- 寻找素数的循环：

```
for (d = 2; d < =sqrt(n); d++)
  if (n % d == 0)break;
```

- 常规退出和非常退出循环条件合并
  - **非常规退出条件取反后与表达式2合并**
- 示例：条件n % d == 0取反，并入循环控制表达式2，那么循环体将会为空：

```
for (d = 2; d <=sqrt(n)&& n % d != 0; d++);
```

空语句

- if、while或for语句提前放置分号，会形成空语句会造成上述语句的提前结束。
- 例1：

```
if (d == 0);
    printf(“除数为0\n”);//与if无关
```

- 例2：

```
i = 10;
while (i > 0); //无限空循环
{
  printf("T minus %d and counting\n", i);
  --i;
}
```

三种循环语句的总结

- 共同点：
  - 循环控制条件非零，执行循环体，否则终止循环。
  - 语句（循环体）可以是任何语句，简单语句、复合语句、空语句。
  - 在循环体内或循环条件中必须有使循环趋于结束的语句，否则死循环
- 不同点：
  - while和for语句先判断再执行，可能一次也不执行；do语句先执行后判断，至少也要执行一次。

实验——求出100~200以内的所有素数

实验——求出100~200以内的所有素数

```
for(m=101;m<=199;m+=2 )//逐个考查
 {

  for(i=3;i<=sqrt(m);i++)
  //从2到m平方根测试m是否被整除
  if(m % i==0) break;
        //被整除退出循环
  if(i>sqrt(m))printf( “m是素数");
}
```

程序练习——求出100~200以内的所有素数

```
int m, i, n=0;
for(m=101;m<=199;m+=2)
{
    for(i=3;i<=sqrt(m);i++)
        if(m % i==0) break;

    if(i>sqrt(m)){
        printf("%d is prime\n",m);
        n++;
    }
}
```