EEM103
Computer Programming

Week6

- Iteration (loop) statements
  - *for*
  - *while*
  - *do - while*
- *break* and *continue* statements

for Iteration Statement

General Format of a for Statement

- The general format of the for statement is

```
                 1                2           4
 for  (initialization;  condition;  increment)
{
   statement 3
 }
```

where
the *initialization* expression initializes the loop-control variable,
the *condition* expression is the loop-continuation condition
the *increment* expression increments the control variable.

for Iteration Statement Properties

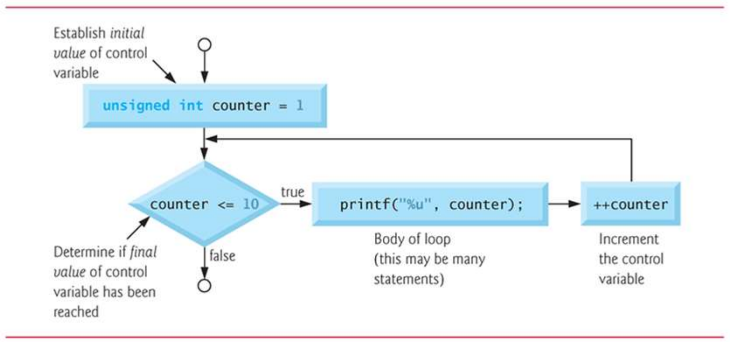
- The two semicolons in the for statement are required
- Expressions in the for Statement's Header Are Optional

  for (; ;)

- The increment expression in the for statement acts like a stand-alone C statement at the end of the body of the for.
- Comma-Separated Lists of Expressions

  for (i=0 , j=1 ; i<10 ; i++ , j*=10)

3


Establish *initial value* of control variable
unsigned int counter = 1
counter <= 10
true
printf("%u", counter);
++counter
Determine if *final value* of control variable has been reached
false
Body of loop (this may be many statements)
Increment the control variable

```c
// Fig. 4.5: fig04_05.c
// Summation with for.
#include <stdio.h>

int main(void)
{
   unsigned int sum = 0; // initialize sum

   for (unsigned int number = 2; number <= 100; number += 2) {
      sum += number; // add number to sum
   }

   printf("Sum is %u\n", sum);
}
```

4

while and *do..while* Iteration Statements

- In the `while` statement, the loop-continuation condition is tested at the beginning of the loop **before** the body of the loop is performed.
- The do…`while` statement tests the loop-continuation condition ***after*** the loop body is performed.
- Therefore, the loop body will be executed at least once.

```
while (condition)              do
    statement                      statement
                               while (condition);
```

break & continue

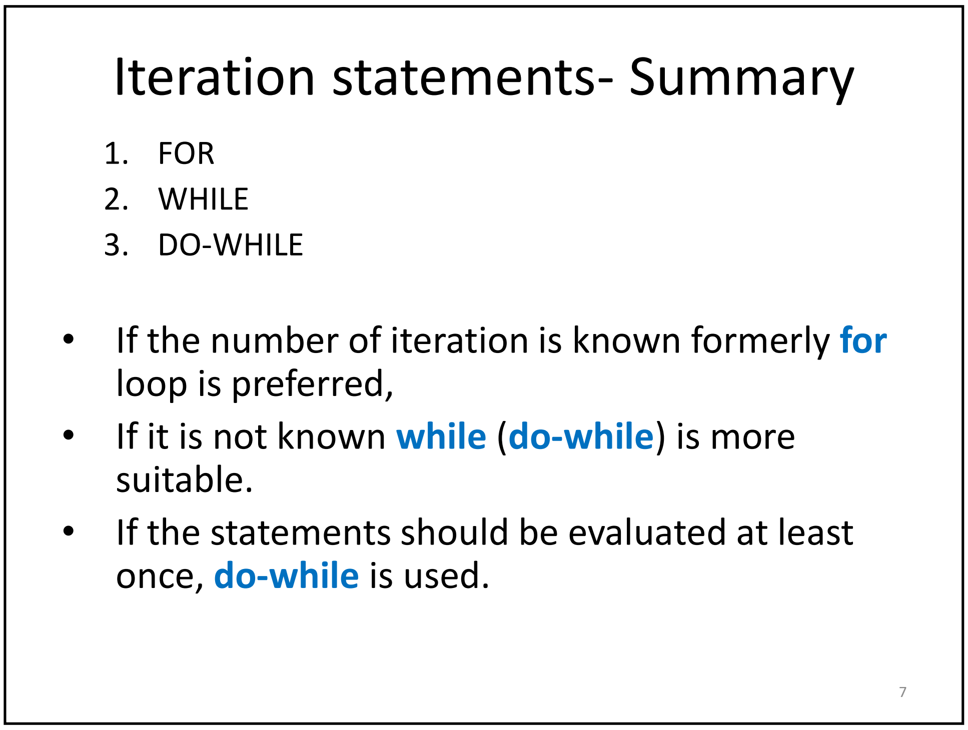
- The ***break*** statement, when executed in a *while*, *for*, *do…while* or *switch* statement, causes an immediate exit from that statement.
- The ***continue*** statement, when executed in a *while*, *for* or *do…while* statement, skips the remaining statements in the body of that control statement and performs the next iteration of the loop.

```
for (i=0 ; i<9; i++)
{
   if(i==4)
       break;
   printf(“%d ”,i)
}
```

```
0 1 2 3
```

```
for (i=0 ; i<9; i++)
{
   if(i==4)
       continue;
   printf(“%d”,i)
}
```

```
0 1 2 3 5 6 7 8
```



Iteration statements- Summary

1. FOR
2. WHILE
3. DO-WHILE

- If the number of iteration is known formerly **for** loop is preferred,
- If it is not known **while** (**do-while**) is more suitable.
- If the statements should be evaluated at least once, **do-while** is used.

7