Iteration

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  - Identify the constructs in algorithms that are associated with ***for, while*** and ***do-while*** control flow construct
  - Express ***for, while*** and ***do-while*** control flow constructs in C programming language

Contents

- *for* Looping Construct
- *while* Looping Construct
- *do-while* Looping Constructs

A Question

Can you write logic to calculate average of n numbers?

Loop Control Statements

- There are always problems whose solution requires doing same steps for a given number of steps

- Looping is also called a repetitive or an iterative control mechanism

- For any looping mechanism, following steps are included
  - Initialization
  - Decision
  - Updating

Types of Loop Control Statements

- C provides 3 types of loop control structures
  1. for statement
  2. while statement
  3. do-while statement

for Statement

```
for (expression1; expression2; expression3)
{
   //statements
}
```

- Initialisation of loop variable
- Condition that must be true to continue loop
- Increment/decrememt Operation to do at end of loop

- Note that no semi colon after the for statement

for Statement contd.

- for(j=0;j<25;j++); //loop without no body
- The three expressions in the for statement are optional
  - One may omit *expression1* if the control variable is initialized elsewhere in the program

    for(;j<25;j++)
  - If *expression2 is omitted, C* assumes that the condition is true, thus creating an infinite loop

    for(j=0;;j++)
  - *expression3 may be omitted* if the increment is calculated by statements in the body of the for statement or if no increment is needed

    for(j=0;j<25;)

for Statement - Examples

- Vary the control variable from 1 to 100 in increments of 1

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

- Vary the control variable from 100 to 1 in increments of -1 (decrements of 1)

  ```
  for ( i = 100; i >= 1; i-- )
  ```

- Vary the control variable from 7 to 77 in steps of 7

  ```
  for ( i = 7; i <= 77; i += 7 )
  ```

- Multiple initialisation and multiple updates

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

for Statement - Algorithms

for <identifier> *in* <initial value> *to* <final value>, *step* <increment> *do*

begin

…

end

- Examples

  for i *in* 0 *to* n, *step* 1 *do*

  begin

  …

  end

while and *do-while* Statements

- To loop based on conditions, ***while*** and ***do-while*** control flow statements are used

- **While** loops are used to loop only when the condition is true

- **Do-While** loops are used to loop at least once and then only when the condition is true

while Statement

- In C programming language, ***while statement*** can be programmed as follows:

```
printf("Please enter the number of numbers(n): ");
scanf("%d",&n);
i = 0;
while (i < n){
    printf("%d  ", i );
    i++;
} //note that no semi colon
```

do-while Statement

- Tests the loop-continuation condition after the loop body is performed
- The loop body will be executed at least once

- Format

```
do {
        statement
} while ( condition );
//note the semi colon after the while statement
```

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

Correct, but confusing

do-while - Example

- In C programming language, ***do-while statement*** can be programmed as follows:

```
printf("Please enter the number of numbers(n): " );
scanf("%d",&n);
i = 0;
do{
      printf("%d  ", i);
      i++;
}while (i < n);
```

Algorithm for *while* and *do-while*

while Statement

while <condition>
begin
…
end

- Example

while (a[i] != n)
begin
…
end

do-while statement

do
…
while <condition>

- Example

do
…
while (a[i] != n)

for, while and *do-while*

for statement

```
for (expression1; exprssion2; expression3)
    statement
```

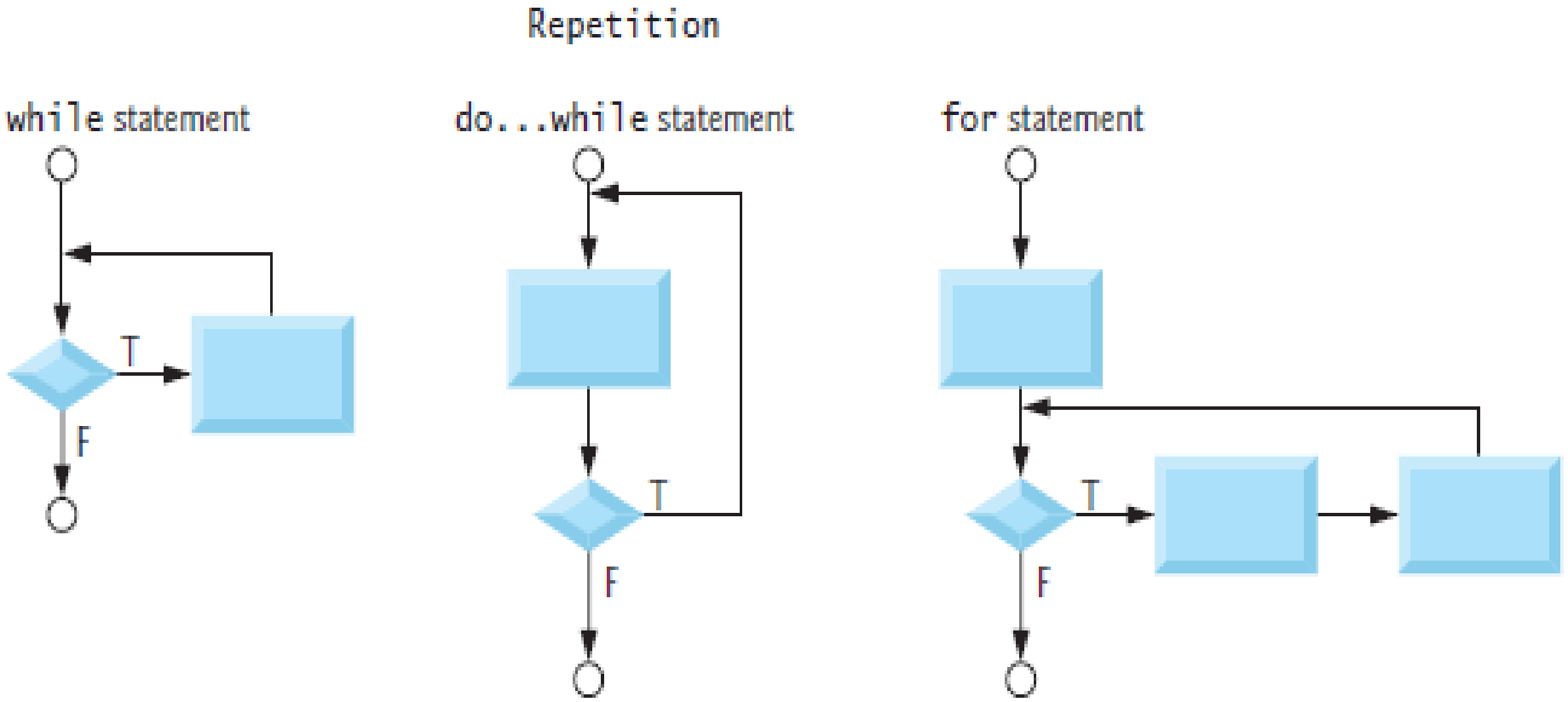
while statement

```
expression1;
while(exprssion2){
    statement
    expression3;
}
```

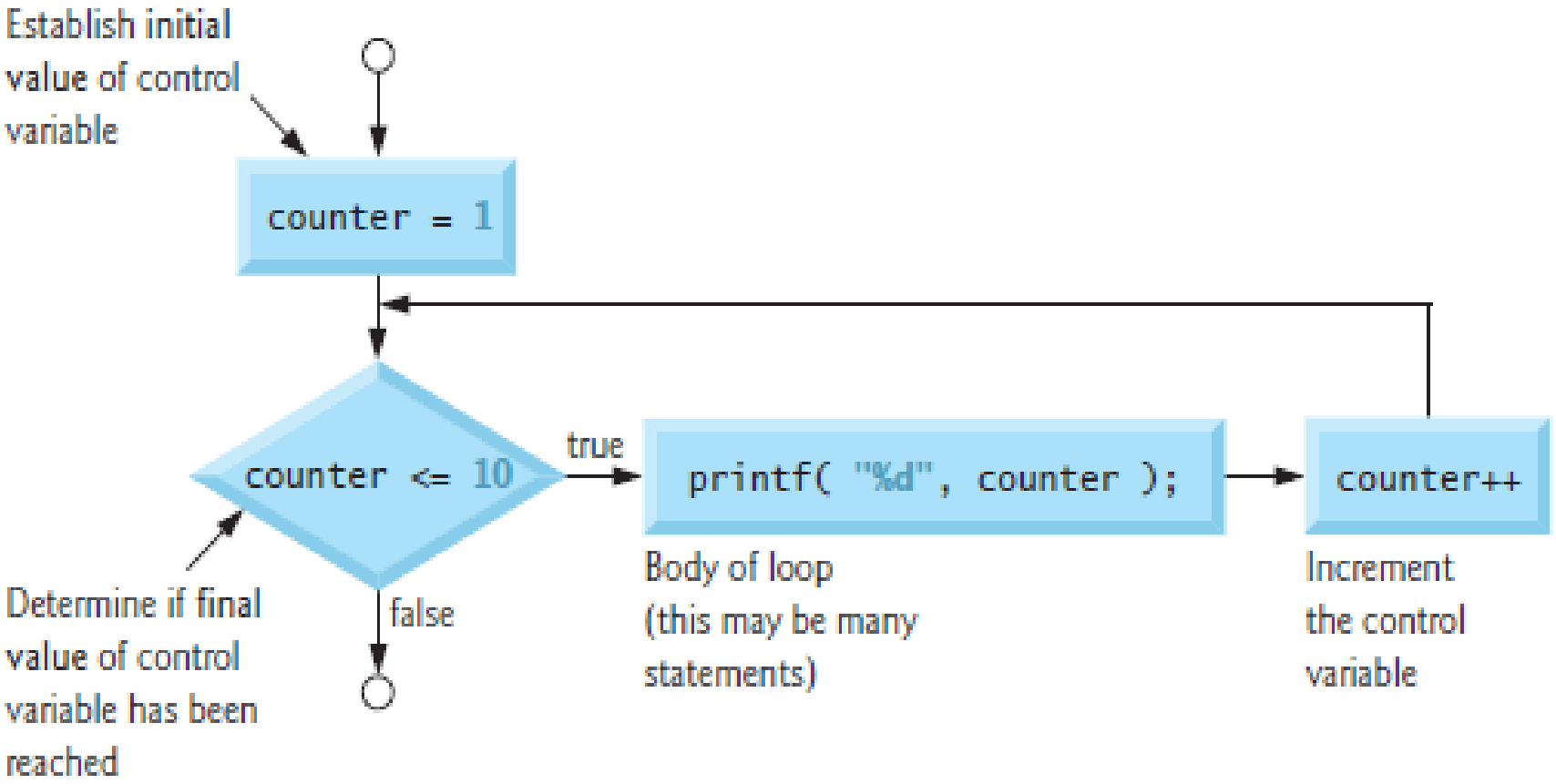
do-while statement

```
expression1;
do{
    Statement
    expression3;
} while(exprssion2);
```

for, while and *do-while* - Flowcharts



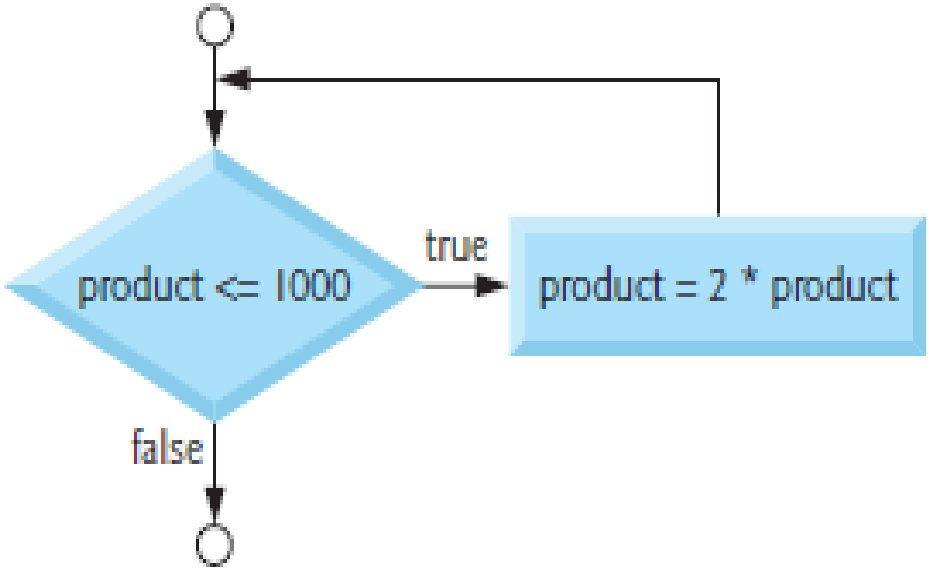
for – Flow chart - Example



while - Flow Chart- Example

```
product= 2;
while ( product <= 100 ) {
    product = 2 * product;
} /* end while */
```

What is the final value of product?



Jumps in Loops

- C permits a jump from one statement to another within a loop as well as the jump out of a loop

- **Jumping out of a Loop**
- An early exit from a loop can be accomplished by using the break statement or the goto statement
  - When the break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop
  - break will exit only a single loop

Jumps in Loops contd.

- **Skipping a part of a Loop**

  continue;

- skip the following statements and continue with the next iteration

- Example

```
while (i<=10){
    if(i==8){
        continue;
    }
    printf("%d",i);
}
```

Break and Continue

```
int main( void ){
    int x;
    for ( x = 1; x <= 10; x++ ) {
        if ( x == 5 )
            break;
        printf( "%d ", x );
    }
    printf( "\nBroke out of loop at x
    == %d\n", x );
    return 0;
}
```

```
int main( void ){
    int x;
    for ( x = 1; x <= 10; x++ ) {
        if ( x == 5 )
            continue;
        printf( "%d ", x );
    }
     printf( "\nUsed continue to
    skip printing the value 5\n" );
    return 0;
}
```

Summary

- Looping constructs alter control flow to repeat a set of instructions
- A loop repeats a block of statements for a predefined number of times
- Loop control structures in C are
  - for
  - while
  - do-while
- All such control structures operate on logical or comparison operators that give true or false values

Further Reading

Dromey, R. (1982) *How To Solve it By Computer.* Noida: Pearson Education Inc.

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.