CONTROL STATEMENTS

while, do-while, for

Designed by Parul Khurana, LIECA.

Control Statements

- Suppose we want to display hello on output screen five times in five different lines,
- We might think of writing either five printf statements or one printf statement consisting of constant string “hello\n” five times.

Control Statements

- *What if we want to display hello 500 times*?
  - Should we write 500 printf statements or equivalent ?
  - Obviously not.
    - It means that we need some programming facility to repeat certain works.
    - Such facility is available in form of ***looping statements***.

Control Statements

- These statements allow us to perform the action of iteration (there may be the situations when we need to *repeat* certain group of statements over either fixed number of times or till certain criteria is met).
- C language provides the following statements:
  - **LOOPING: THE while STATEMENT**
  - **MORE LOOPING: THE do - while STATEMENT**
  - **STILL MORE LOOPING: THE for STATEMENT**

Control Statements

- LOOPING: THE while STATEMENT
  - **The while statement is used to carry out looping operations, in which a group of statements is executed repeatedly, until some condition has been satisfied.**
  - **The while statement is suited for problems where it is known in advance that *how many times a statement or a statement-block* will be executed.**

Control Statements

- LOOPING: THE while STATEMENT
  - **The general form of the while statement is**
    - **while (expression) statement**
      - » **The statement will be executed repeatedly, as long as the expression is true (i.e., as long expression has a nonzero value).**

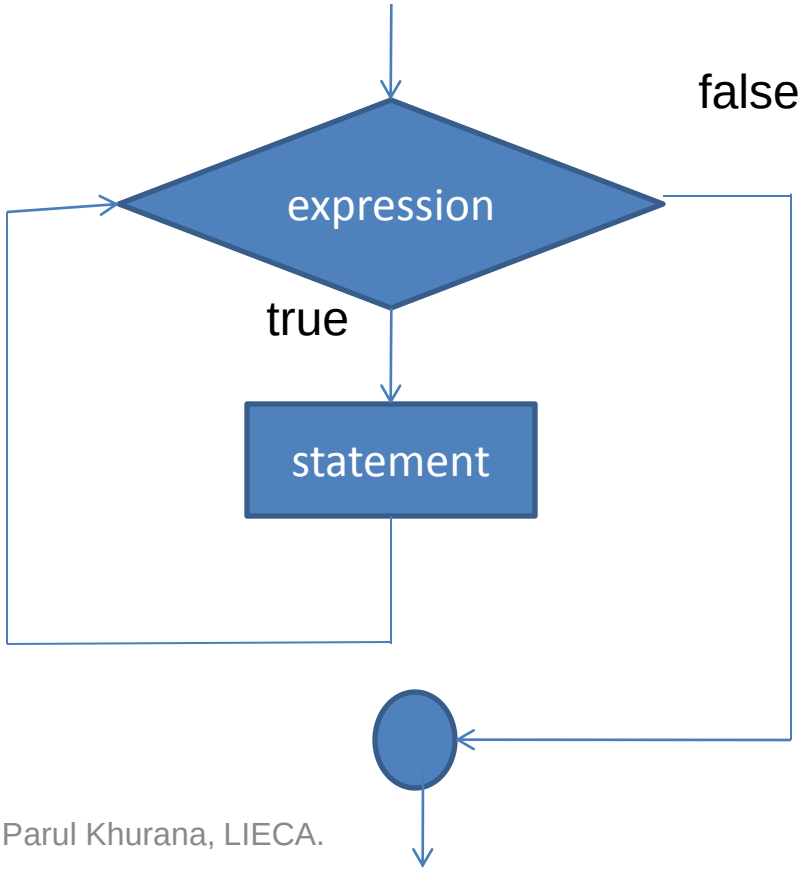
Control Statements

- LOOPING: THE while STATEMENT
  - **The general form of the while statement is**
    - **while (expression) statement**
      - » **This statement can be simple or compound, though it is usually a compound statement.**
      - » **It must include some feature that eventually alters the value of the expression, thus providing a stopping condition for the loop.**

Control Statements

- LOOPING: THE while STATEMENT - iterative structure: (Code and Logic Flow Control)

```
:
:
while ( expression )
    {
     statement
    }
:
:
```



Control Statements

- LOOPING: THE while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
 while ( digit <= 9)   {
        printf ( "%d\n", digit);
         ++digit;          }
}
```

  - **Initially digit is *assigned* a value of 0.**
  - **The while loop then displays the current value of digit, increases its value by 1 and then repeats the cycle, until the value of digit exceeds 9.**

Control Statements

- LOOPING: THE while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
 while ( digit <= 9)   {
        printf ( "%d\n", digit);
         ++digit;          }
}
```

  - **The net effect is that the *body of the loop will be repeated 10 times*, resulting in 10 consecutive lines of output.**
  - **Each line will contain a successive integer value, beginning with 0 and ending with 9.**

Control Statements

- LOOPING: THE while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
 while ( digit <= 9)   {
        printf ( "%d\n", digit);
         ++digit;          }
}
```

  - **Thus when program is executed, the following output *will be generated:***

```
0
1
2
3
4
5
6
7
8
9
```

Control Statements

- MORE LOOPING: THE do - while STATEMENT
  - **When a loop is constructed using the while statement described in previous slides, the test for continuation of the loop is carried out at the beginning of each pass.**
  - **Sometimes, however, it is desirable to have a loop with the test for continuation at the end of each pass. This can be accomplished by *means of the do while statement*.**

Control Statements

- MORE LOOPING: THE do - while STATEMENT
  - **The general form of the do - while statement is**
    - **do { statement } while (expression);**
      - **» The statement will be executed repeatedly, as, long as the value of expression is true (i.e., is nonzero).**
      - **» Notice that statement will always be executed at least once, since the test for repetition does not occur until the end of the first pass through the loop.**

Control Statements

- MORE LOOPING: THE do - while STATEMENT
  - **The general form of the do - while statement is**
    - **do { statement } while (expression);**
      - **» The statement can be either simple or compound, though most applications will require it to be a compound statement.**
      - **» It must include some feature that eventually alters the value of expression so the looping action can terminate.**

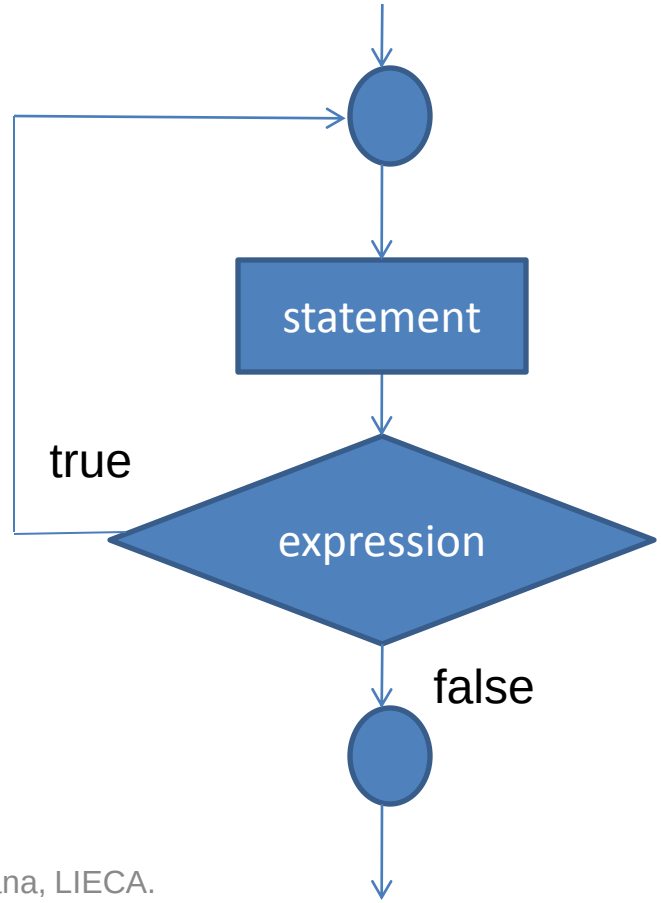
Control Statements

- MORE LOOPING: THE do - while STATEMENT
  - **The general form of the do - while statement is**
    - **do { statement} while (expression);**
      - » **For many applications it is more natural to test for continuation of a loop at the beginning rather than at the end of the loop.**
      - » **For this reason, the do-while statement is used less frequently than the while statement.**

Control Statements

- MORE LOOPING: THE do while STATEMENT - iterative structure: (Code and Logic Flow Control)

```
:
:
do
{
      statement
}while ( expression );
:
:
```



Control Statements

- MORE LOOPING: THE do while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
do {
 printf ( "%d\n", digit++);
      } while ( digit <= 9);
}
```

  - **As in the earlier example, digit is initially *assigned* a value of 0.**
  - **The do - while loop displays the current value of digit, increases its value by 1 and then tests to see if the current value of digit exceeds 9.**

Control Statements

- MORE LOOPING: THE do while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
do {
 printf ( "%d\n", digit++);
      } while ( digit <= 9);
}
```

  - **If so, the loop terminates; otherwise, the loop continues, using the new value of digit.**
  - **Note that the test is carried out at the end of each pass through the loop.**

Control Statements

- MORE LOOPING: THE do while STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
{
int digit = 0;
do {
 printf ( "%d\n", digit++);
      } while ( digit <= 9);
}
```

  - **The net effect is that the loop will be repeated 10 times, resulting in 10 successive lines of output.**
  - **Each line will appear exactly as shown in the previous example of while loop.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **The for statement is the third and perhaps the most commonly used looping statement in C.**
  - **This statement includes an expression that specifies an initial value for an index, another expression that determines whether or not the loop is continued, and a third expression that allows the index to be modified at the end of each pass.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **The general form of the for statement is**
    - **for (expression 1; expression 2; expression 3) statement**
      - » **where expression 1 is used to initialize some parameter (called an index) that controls the looping action, expression 2 represents a condition that must be true for the loop to continue execution, and expression 3 is used to alter the value of the parameter initially assigned by expression 1.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT

  - **The general form of the for statement is**

    - **for (expression 1; expression 2; expression 3) statement**

      - **Typically, expression 1 is an assignment expression, expression 2 is a logical expression and expression 3 is a unary expression or an assignment expression.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **The general form of the for statement is**
    - **for (expression 1; expression 2; expression 3) statement**
      - » **When the for statement is executed, expression 2 is evaluated and tested at the beginning of each pass through the loop, and expression 3 is evaluated at the end of each pass.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **The general form of the for statement is**
    - **for (expression 1; expression 2; expression 3) statement**
      - » **Thus, the for statement is equivalent to:**

```
expression 1;
    while ( expression 2)   {
            statement
             expression 3;
    }
```

- **The looping action will continue as long as the value of expression 2 is not zero, that is, as long as the logical condition represented by expression 2 is true.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **The for statement, like the while and do - while statements, can be used to carry out looping actions where the number of passes through the loop is not known in advance.**
  - **Because of the features that are built into the for statement, however, it is particularly well suited for loops in which the number of passes is known in advance.**

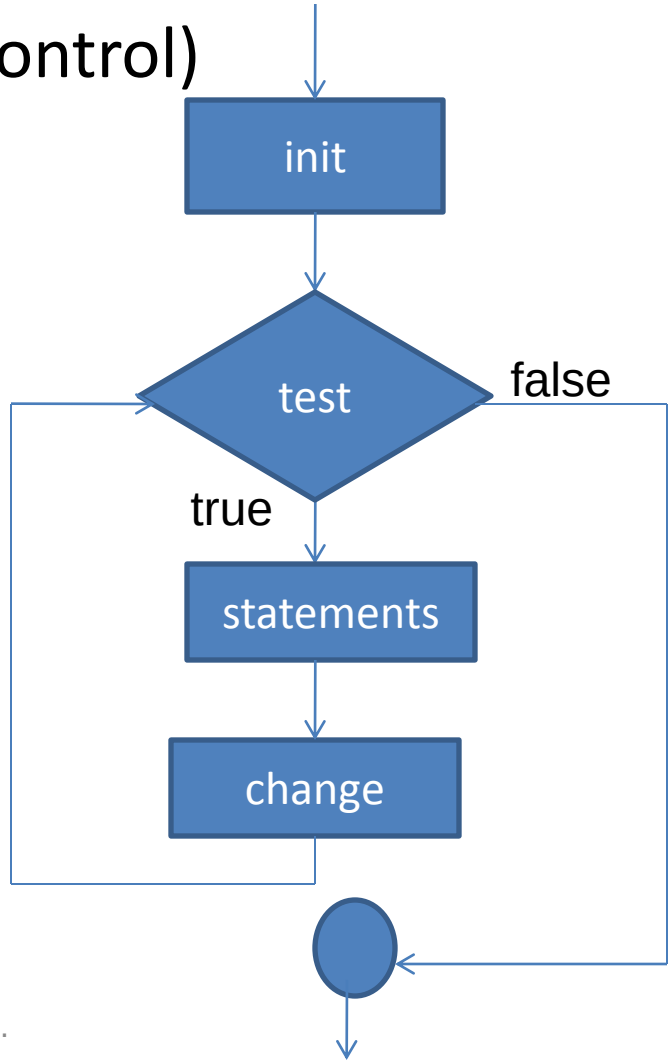
Control Statements

- STILL MORE LOOPING: THE for STATEMENT
  - **As a rough rule of thumb, while loops are generally used when the number of passes is not known in advance, and for loops are generally used when the number of passes is known in advance.**

Control Statements

- STILL MORE LOOPING: THE for while STATEMENT - iterative structure: (Code and Logic Flow Control)

```
:
for ( init; test; change )
        {
        statements
        }
```



where init is an expression to initialize the counter, the test is an expression to see when to stop iterating, and change is an expression to change the counter for each pass of the loop.

Control Statements

- STILL MORE LOOPING: THE for STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
 {
int digit ;
for ( digit = 0; digit <=9; ++digit)
      {
       printf ( "%d\n", digit);
       }
 }
```

  - **The first line of the for statement contains three expressions, enclosed in parentheses.**
  - **The first expression assigns an initial value 0 to the integer variable digit;**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
 {
int digit ;
for ( digit = 0; digit <=9; ++digit)
      {
       printf ( "%d\n", digit);
       }
 }
```

  - **The second expression continues the looping action as long as the current value of digit does not exceed 9 at the beginning of each pass; and the third expression increases the value of digit by 1 at the end of each pass through the loop.**

Control Statements

- STILL MORE LOOPING: THE for STATEMENT - Example
  - **Suppose we want to display the consecutive digits 0, 1, 2, 3,. . . . . 9, with one digit on each line. This can be accomplished with the following program.**

```
#include <stdio.h>
void main ( void )
 {
int digit ;
for ( digit = 0; digit <=9; ++digit)
      {
       printf ( "%d\n", digit);
       }
 }
```

  - **The printf function, which is included in the for loop, produces the desired output.**
  - **Each line will appear exactly as shown in the previous example of while loop.**

Practice Questions

- Predict the output of the following code segment:

```
int n = 0;
        if ( n >= 0 )
                for ( i = 0 ; I < n ; i++ )
                        if (s[i] > 0 ) {
                                printf (" ... ");
                                return(i);
                        }
        else
                printf ("Error");
```

Practice Questions

- Predict the output of the following code segment:

```
int x = 0;
for ( ; ; )
{
 if (x++ == 4)
  break;
 continue;
}
     printf("x = %d\n", x);
```

```
int x = 0;
for (x = 1; x < 4; x++);
          printf("x = %d\n", x);
```

Practice Questions

- Predict the output of the following programs:

```
#include <stdio.h>
int i;
int main()
{
  for(i = 0; i < 10; i++)
  printf("i = %d\n", i);
  return 0;
}
```

```
#include <stdio.h>
int i;
int main()
{
  for(i = 0; i < 10; i++);
  printf("i = %d\n", i);
  return 0;
}
```