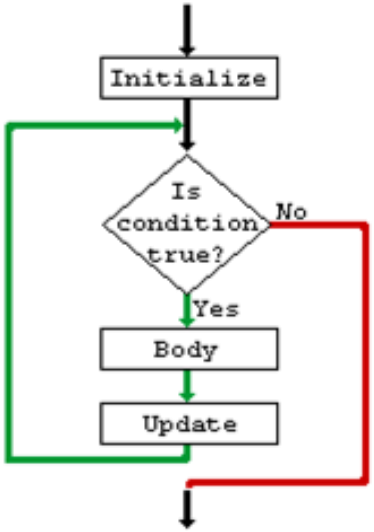
Loop Control Structures

L9



Learning Objectives

- To learn and appreciate the following concepts
  - The `while` Statement
  - Programs

Learning Outcome

- At the end of session student will be able to learn and understand
    - The while Statement
    - Sample programs

Controlling the program flow

- Forms of controlling the program flow:
  - Executing a sequence of statements
  - Using a test to decide between alternative sequences (**branching**)
  - Repeating a sequence of statements (until some condition is met) (**looping**)

```
Statement1
Statement2
Statement3
Statement4
Statement5
Statement6
Statement7
Statement8
```

Program Looping

- **A set of statements that executes repetitively for a number of times.**
- Simple example: displaying a message 100 times:

```
printf(hello !\n”);
printf(hello !\n”)
printf(hello !\n”)
…
printf(hello !\n”)
printf(hello !\n”)
```

```
Repeat 100 times
        printf(hello !\n”)
```

***Program looping**:* enables you to develop concise programs containing repetitive processes that could otherwise require many lines of code !

The need for program looping

Example problem: computing triangular numbers.
(The n-th triangular number is the sum of the integers from 1 through *n*)

```
#include <stdio.h>
int main (void)
{
    int triangularNumber;
    triangularNumber = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8;
    printf("The eighth triangular number is
    %d",triangularNumber);
    return 0;
}
```

What if we have to compute the 200-th (1000-th, etc) triangular number ?

We have 3 different statements for looping.

Iterative (loop) control structures

- Three kinds of loop control structures:
  - ✓while
  - ✓do while
  - ✓for

Iterative (loop) control structures

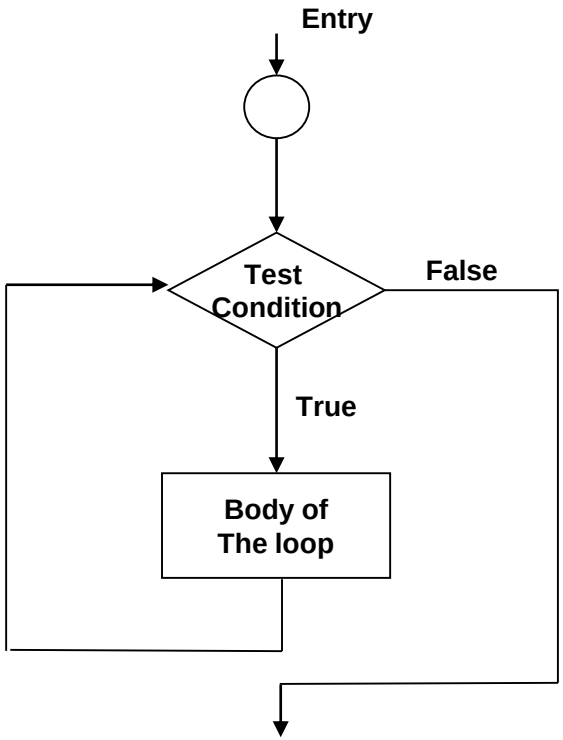
- Each loop control structure will have
  - ✓ **Program loop:** body of loop.
  - ✓ **control statement** → tests certain conditions & then directs repeated execution of statements within the body of loop.

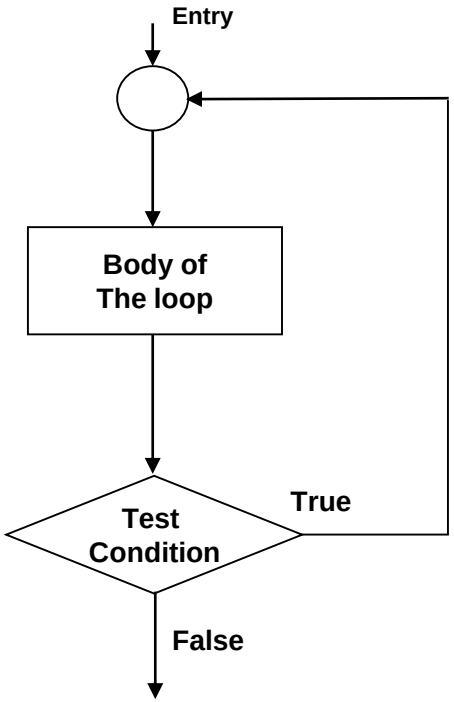
- **Two types:** Based on position of control statement.

1) **Entry controlled loop**: control is tested before the start of the loop. If false, body will not be executed.

2) **Exit controlled loop**: test is performed at the end of the body. i.e. body of loop executed at least once.

Entry Controlled & Exit controlled loops





while-statement

General format:

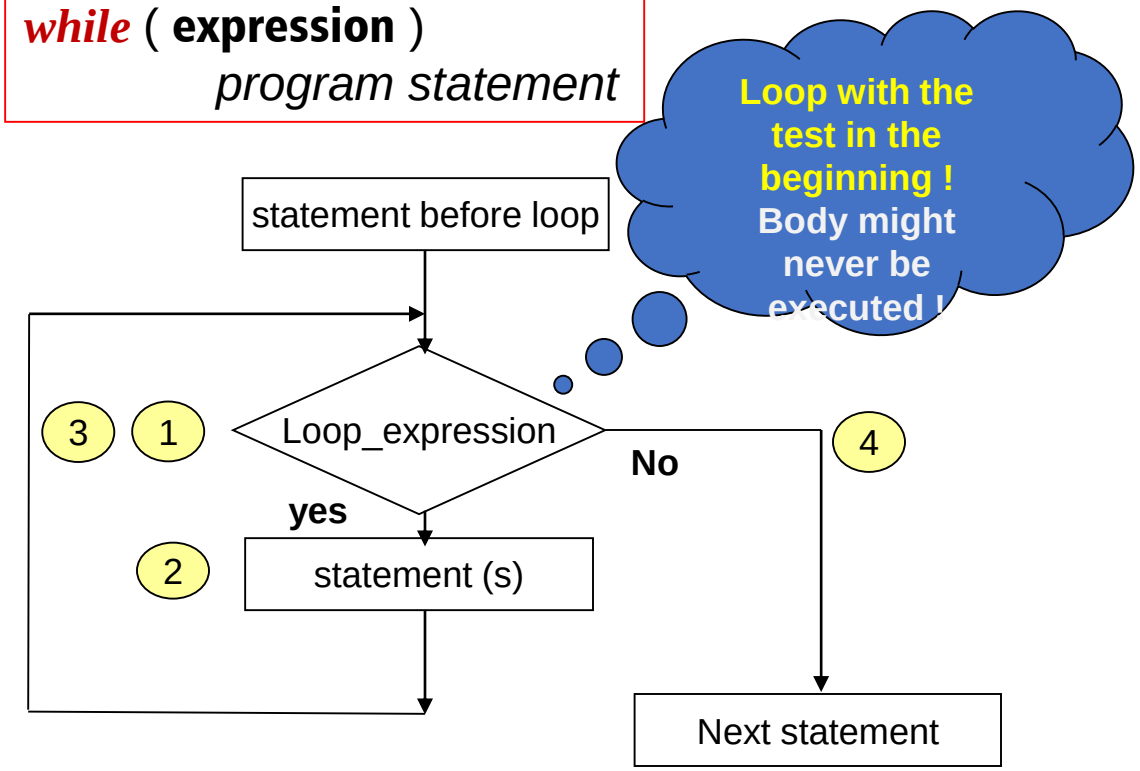
```
while (test expression)
    {
       body of the loop
    }
```

Note: braces optional if only one statement.

- ✓ **Entry controlled** loop statement.
- ✓ **Test condition** is evaluated & if it is true, then body of the loop is executed.
- ✓ This is **repeated until the test condition becomes false**, & control transferred out of the loop.
- ✓ **Body of loop is not executed if the condition is false at the very first attempt.**
- ✓ **While loop can be nested.**

The `while` statement

while (**expression**)
program statement



Sum and Mean of first N natural numbers

Name of the algorithm: Sum and Mean of natural numbers.

Step 1: Start

Step 2: [Read the maximum value of N]

Input N

Step 3: [Set sum equal to 0]

Sum ← 0

Step 4: [Compute the sum of all first N natural numbers]

```
i=1
While(i<=N)
begin
        Sum ← Sum + i
        i++;
end
```

Sum and Mean of first N natural numbers

Step 5: [Compute mean value of N natural numbers]

Mean ← Sum / N

Step 6: [Print Sum and Mean]

Print 'Sum of N natural numbers=',Sum

Print 'Mean of N natural numbers =',Mean

Step 7: [End of algorithm]

Stop

Finding sum of natural numbers up to 100

```
#include <stdio.h>
int main()
{
   int n;
   int sum;
   sum=0; //initialize
    sum
   n=1;
   while (n < 100)
   {
       sum= sum + n;
       n = n + 1;
   }
   printf("%d",sum);
   return 0;
}
```

Program to reverse the digits of a number

```
#include <stdio.h>
int main()
{
        int number, rev=0, right_digit;

        printf("Enter your number.\n");
        scanf("%d",&number);

        while ( number != 0 )
        {
                right_digit = number % 10;
                rev=rev*10 + right_digit;
                number = number / 10;
        }
        printf("The reversed number is %d", rev);
        return 0;
}
```

Check for palindrome

```
n = num;
 while(num>0)
            {
              dig = num % 10;
              rev = rev * 10 + dig;
              num = num / 10;
            }
if (n == rev)
 printf("\n\t GIVEN NO IS A PALINDROME");
else
 printf("\n\t GIVEN NO NOT A PALINDROME");
```

Palindrome (number)
e.g.- 121

Session 7 Summary

- **Switch statement**
- **Looping Concepts**
- **While loop**

Poll Question

Go to chat box/posts for the link to the Poll question

Submit your solution in next 2 minutes

Click the result button to view your score