Lab 08: THE DO-WHILE LOOP

*Further exploring the loop control structures with emphasis on the ‘do-while’ loop.*

1. **The DO-WHILE loop**

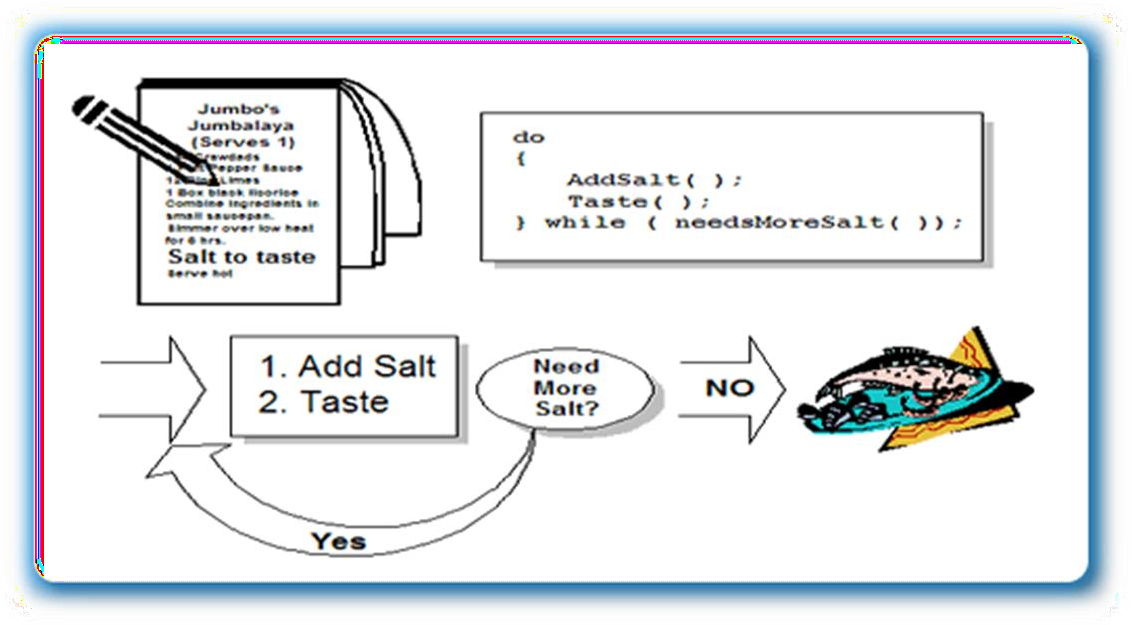
Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop. The *expression* in a **do-while** statement is evaluated after the body of the loop is executed. Therefore, the body of the loop is always executed at least once.

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Execution proceeds as follows:

* + 1. The statement body is executed.
    2. Next, *expression* is evaluated. If *expression* is false, the **do-while** statement terminates and control passes to the next statement in the program. If *expression* is true (nonzero), the process is repeated, beginning with step 1.

The **do-while** statement can also terminate when a **break**, **goto**, or **return** statement is executed within the statement body.



One application of a do-while loop is an ATM-type situation. You've got your money or made your deposit, and, instead of giving you your card back, the machine asks you "Do you want another transaction?.

**1.1 Syntax of the ‘do-while’ loop**

do{

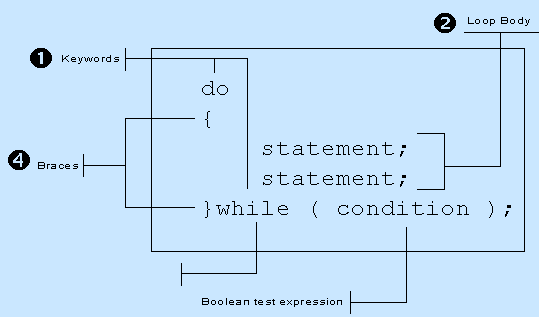
//body of the loop

}

while(condition);

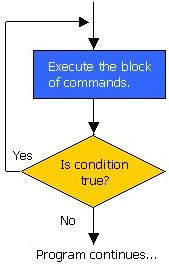
Notice that the condition is tested at the end of the block instead of the beginning, so the block will be executed at least once. If the condition is true, we jump back to the beginning of the block and execute it again .A do while loop is almost the same as a while loop except that the loop body is guaranteed to execute at least once.

A while loop says "Loop while the condition is true, and execute this block of code", a do while loop says “Execute this block of code, and then continue to loop while the condition is true.”



**Flow chart:**

.



# EXAMPLE 1

#include <stdio.h> int main()

{

int x;

x = 0;

do {

/\* "Hello, world!" is printed at least one time even though the condition is false \*/ printf( "Hello, world!\n" );

} while ( x != 0 ); getch();

}

Keep in mind that you must include semi-colon after the while in the above example. A common error is to forget that a do-while loop must be terminated with a semicolon (the other loops should not be terminated with a semicolon, adding to the confusion). Notice how the loop will execute once, because it automatically executes before checking the condition.

# EXAMPLE 2

#include <stdio.h> int main () {

int a = 10;

/\* do loop execution \*/ do {

printf("value of a: %d\n", a); a = a + 1;

}while( a < 20 );

getch();

}

value of a: 10 value of a: 11 value of a: 12 value of a: 13 value of a: 14 value of a: 15 value of a: 16 value of a: 17 value of a: 18

value of a: 19

**Output:**

.

**Student Tasks:**

## Task 1:

An electric power distribution company charges its domestic consumers as follows.

## Consumption Units Rate of Charge

* 0-200 Rs.0.50 per unit
* 201-400 Rs.100 plus Rs.0.65 per unit excess 200
* 401-600 Rs.230 plus Rs.0.80 per unit excess of 400.

Write a C program that reads the customer number and power consumed and prints the amount to be paid by the customer. The program runs at least once.

## Task 2:

Write a C program that allows the user to enter in 5 grades, ie, marks between 0 - 100. The program must calculate the average mark, and state the number of marks less than 65.

## Task 3:

Write a program that will generate even numbers and calculate sum and average of those numbers.

## Task 4:

Using do while loop, program to show a below multiplication table:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
| 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| 6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 |
| 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 |
| 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |
| **Task 5:** |  |  |  |  |  |  |  |  |  |

Write a program that will generate years from 1950 to 2015 and mention if the year is leap year**. Task 6:**

Write a program to output a table of values of the integers starting at 1 and their squares. Label the table at the top of the columns. For example, your output might look like this:

|  |  |
| --- | --- |
| Number | Square |
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |

## Task 7:

Write a program to print all Prime numbers between 1 to n.

A **Prime Number** can be divided evenly only by 1, or itself. And it must be a whole number greater than 1.

**Example:**

5 can only be divided evenly by 1 or 5, so it is a prime number.

But 6 can be divided evenly by 1, 2, 3 and 6 so it is NOT a prime number (it is a composite number).

## Task 8:

Write a program to enter any number and calculate its factorial using do-while.

How Factorial is calculated?

**Factorial of 5:**

= 5!

= 5 \* 4!

= 5 \* 4 \* 3!

= 5 \* 4 \* 3 \* 2!

= 5 \* 4 \* 3 \* 2 \* 1!

= 5 \* 4 \* 3 \* 2 \* 1 \* 0!

= 5 \* 4 \* 3 \* 2 \* 1 \* 1

=120

## Task 9:

Write a C program to enter any number from user and find the reverse of number, the program executes once and if user wants to repeat the program he will press Y.

***Example:***

Input:1234 Output:4321

## Task 10:

Write a loop which will produce the following output.

1

22

333

4444

55555