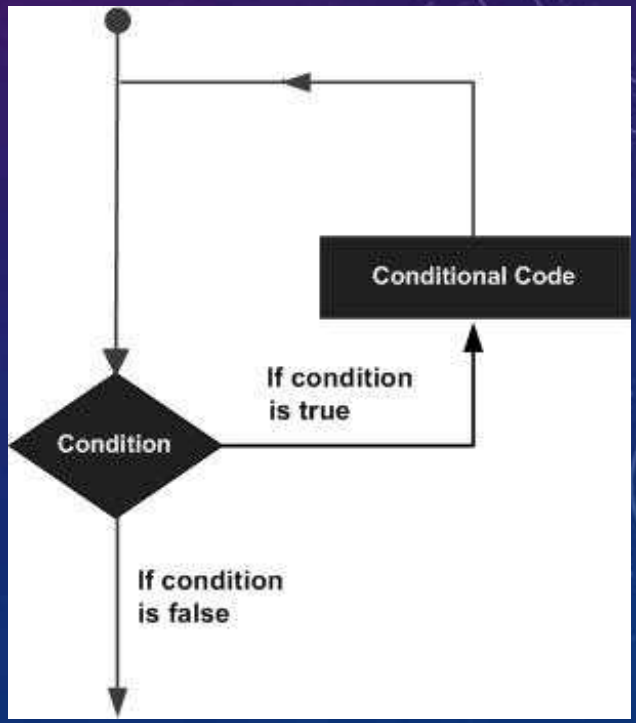
STUDY C LANGUAGE II

WITH

ARAFAT BIN REZA

LOOPS

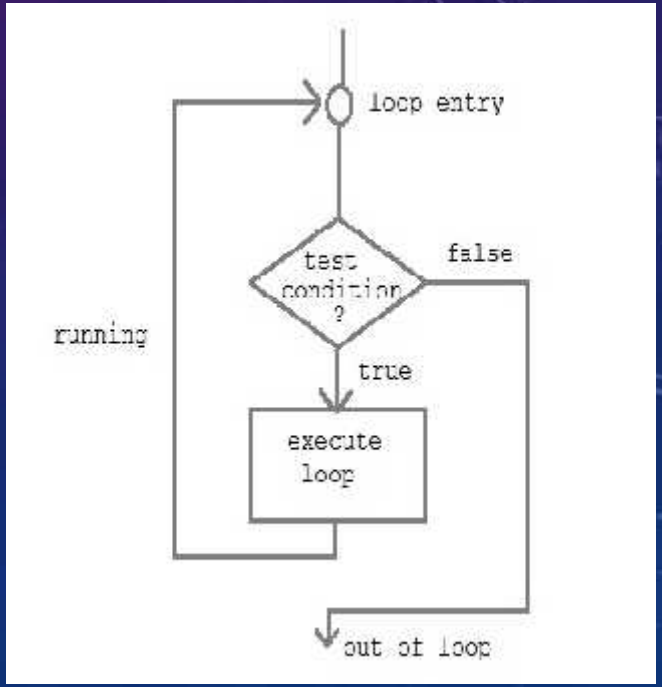
- You may encounter situations, when a block of code needs to be executed several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- A loop statement allows us to execute a statement or group of statements multiple times. Given below is the general form of a loop statement in most of the programming languages −



HOW TO USE LOOPS

How it Works:

A sequence of statements are executed until a specified condition is true. This sequence of statements to be executed is kept inside the curly braces { } known as the Loop body. After every execution of loop body, condition is verified, and if it is found to be true the loop body is executed again. When the condition check returns false, the loop body is not executed.



LOOP TYPE

- There are 3 type of Loops in C language

1. while loop
2. for loop
3. do-while loop

WHILE LOOP

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
while loop can be addressed as an entry control loop. It is completed in 3 steps.

Variable initialization.(e.g int x=0;)
condition(e.g while(x<=10))
Variable increment or decrement (x++ or x-- or x=x+2)

Syntax :

```
variable initialization ;
while (condition)
{
 statements ;
 variable increment or decrement ;
}
```

FOR LOOP

for loop is used to execute a set of statements repeatedly until a particular condition is satisfied. we can say it an open ended loop. General format is,

```
for(initialization; condition ; increment/decrement)
{
  statement-block;
}
```

In for loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. for loop can have only one condition.

NESTED FOR LOOP

We can also have nested for loops, i.e one for loop inside another for loop. Basic syntax is,

```
for(initialization; condition; increment/decrement)
{
  for(initialization; condition; increment/decrement)
     {
        statement ;
     }
}
```

DO WHILE LOOP

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement.

General format of do-while loop is,

```
do
{
 ....
 .....
}
```

THE INFINITE LOOP

A loop becomes an infinite loop if a condition never becomes false. The for loop is traditionally used for this purpose. Since none of the three expressions that form the 'for' loop are required, you can make an endless loop by leaving the conditional expression empty.

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.

You can terminate an infinite loop by pressing Ctrl + C keys.

EXAMPLE :

```
#include <stdio.h>

int main () {

  for( ; ; ) {
    printf("This loop will run
forever.\n");
  }

  return 0;
}
```

JUMPING OUT OF LOOPS

Sometimes, while executing a loop, it becomes necessary to skip a part of the loop or to leave the loop as soon as certain condition becomes true, that is called jumping out of loop. C language allows jumping from one statement to another within a loop as well as jumping out of the loop.

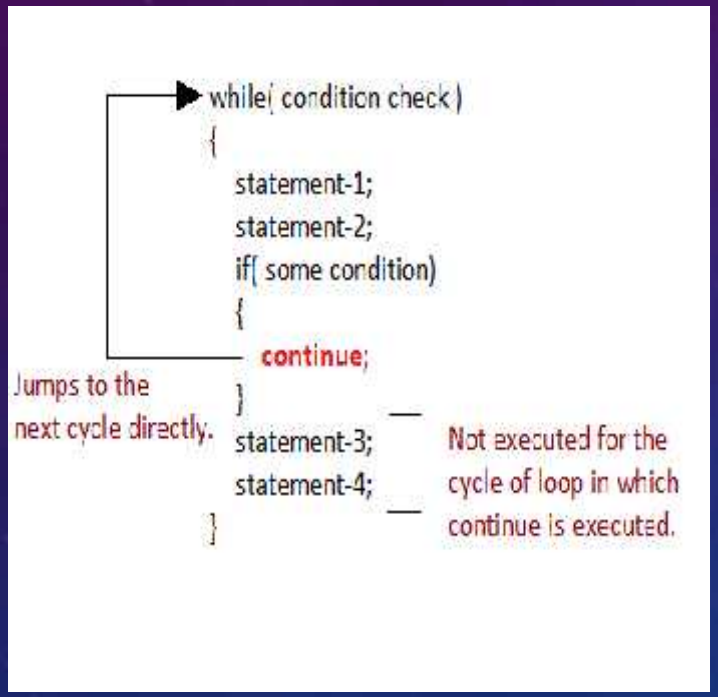
1. break statement :

When break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.

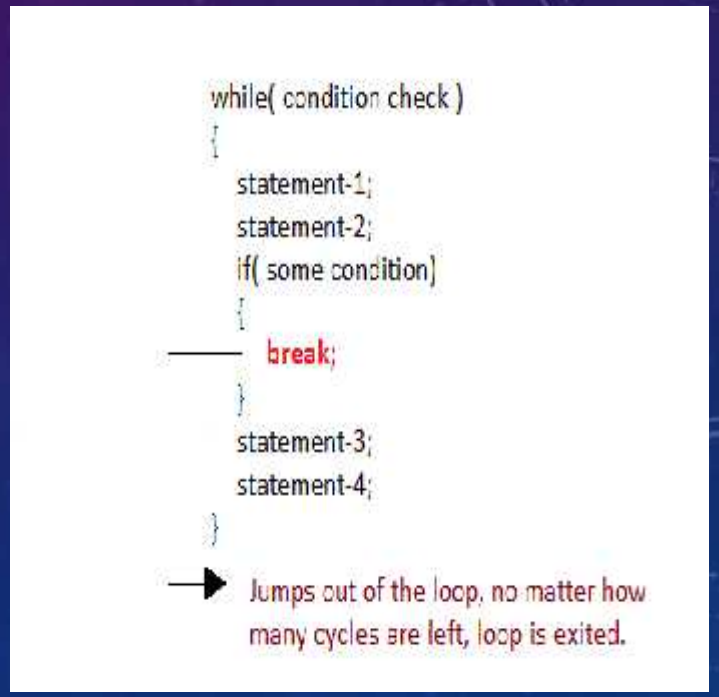
2. continue statement:

It causes the control to go directly to the test-condition and then continue the loop process. On encountering continue, cursor leave the current cycle of loop, and starts with the next cycle.

FLOW DIAGRAM:



break statement



continue statement

GOTO STATEMENT

A goto statement in C programming provides an unconditional jump from the 'goto' to a labeled statement in the same function.

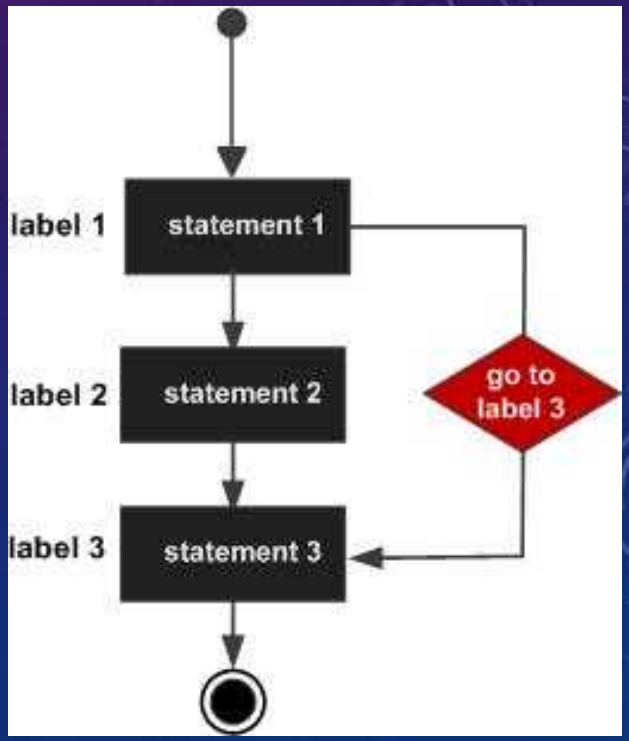
Syntax:

goto label;

..

.

label: statement;



SWITCH STATEMENT

Switch statement is used to solve multiple option type problems for menu like program, where one value is associated with each option. The expression in switch case evaluates to return an integral value, which is then compared to the values in different cases, where it matches that block of code is executed, if there is no match, then default block is executed. The general form of switch statement is,

```
switch(expression)
{
 case value-1:
        block-1;
        break;
 case value-2:
        block-2;
        break;
 case value-3:
        block-3;
        break;
 case value-4:
        block-4;
        break;
 default:
        default-block;
        break;
}
```

POINTS TO REMEMBER

We don't use those expressions to evaluate switch case, which may return floating point values or strings.
It isn't necessary to use break after each block, but if you do not use it, all the consecutive block of codes will get executed after the matching block.

The output was supposed to be only A because only the first case matches, but as there is no break statement after the block, the next blocks are executed, until the cursor encounters a break.

default case can be placed anywhere in the switch case. Even if we don't include the default case switch statement works.

```
int i = 1;
switch(i)
{
 case 1:
   printf("A");       // No break
 case 2:
   printf("B");       // No break
 case 3:
   printf("C");
   break;
}
```

Output : A B C