Math Series

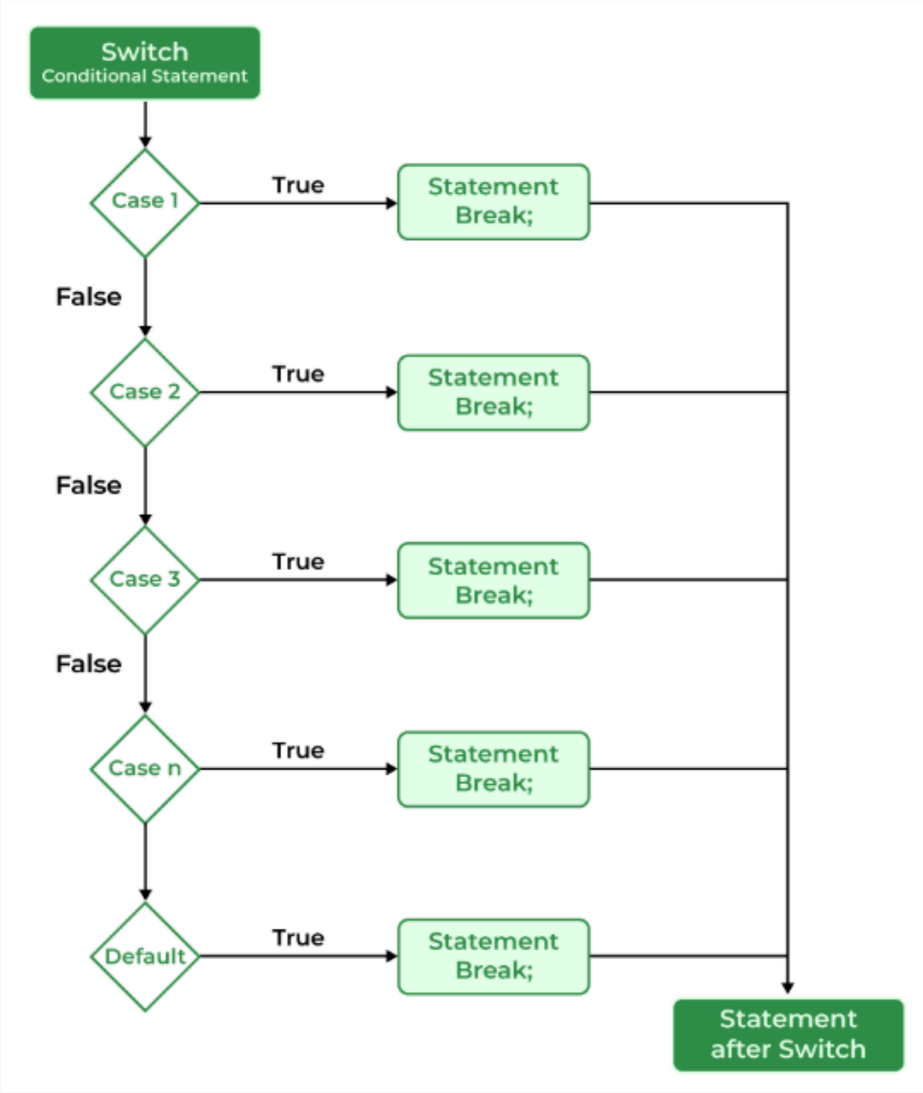
C Programming

UG Sem-3 Major (Kalyani University)

Day - 04

Switch Statement

when there are several options and we have to choose only one option from the available ones, we can use switch statement. Depending on the selected option, a particular task can be performed. A task represents one or more statements.



Switch Statement

Syntax

```
switch (expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Rules for Switch statements

- *The test expression of Switch must necessarily be an int or char.*
- *The value of the case should be an integer or character.*
- *Cases should only be inside the switch statement.*
- *Using the break keyword in the switch statement is not necessary.*
- *The case label values inside the switch should be unique.*

Switch Statement

Syntax

```
switch (expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Rules for Switch statements

- *The expression in switch statement must be an integer value or a character constant.*
- *No real numbers are used in an expression.*
- *The default is optional and can be placed anywhere, but usually placed at end.*
- *The case keyword must terminate with colon (:).*
- *No two case constants are identical.*
- *The case labels must be constants.*

Switch Statement

Valid Switch	Invalid Switch	Valid Case	Invalid Case
switch(x)	switch(f)	case 3;	case 2.5;
switch(x>y)	switch(x+2.5)	case 'a';	case x;
switch(a+b-2)		case 1+2;	case x+2;
switch(func(x,y))		case 'x'>'y';	case 1,2,3;

Iteration Statements/ Loop Control Statements

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. The loops in C language are used to execute a block of code or a part of the program several times. In other words, it iterates/repeat a code or group of code many times.

Or Looping means a group of statements are executed repeatedly, until some logical condition is satisfied.

Loops in C Language

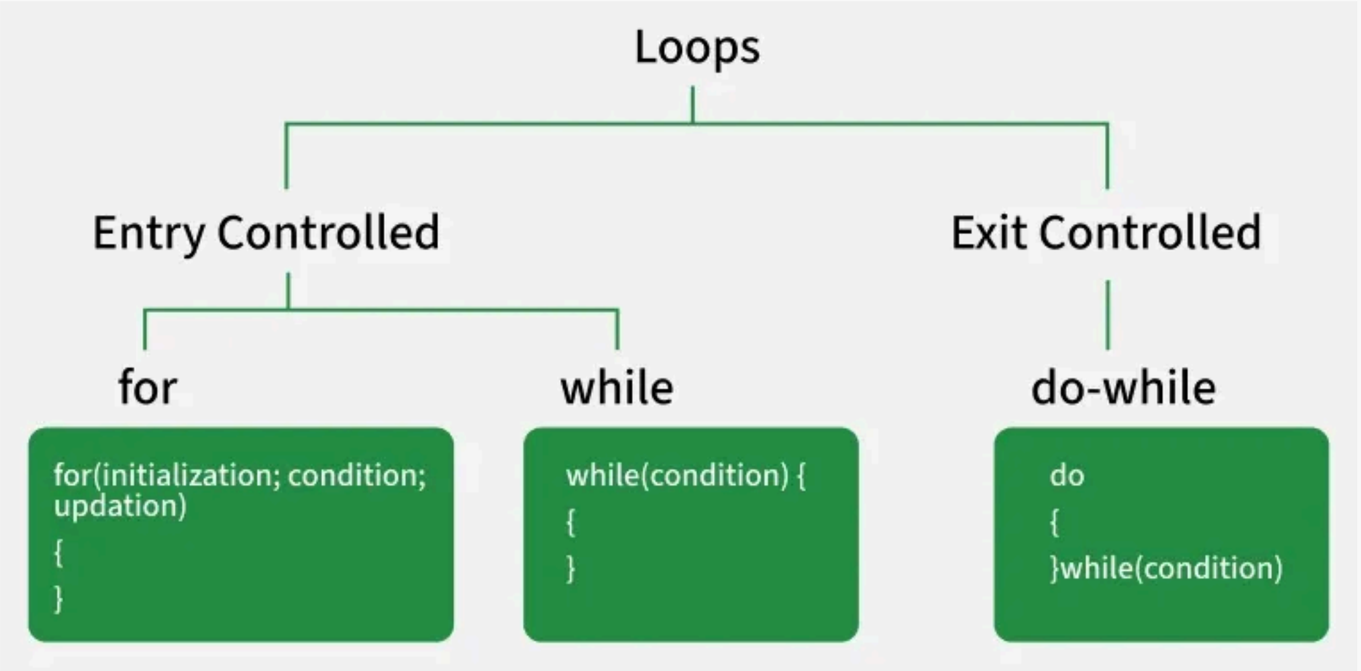
A looping process would include the following four steps.

1 : Initialization of a condition variable.

2 : Test the condition.

3 : Executing the body of the loop depending on the condition.

4 : Updating the condition variable.



Entry Controlled Loop

Entry controlled loops are loop structures where the condition is checked before entering the loop body. If the condition evaluates to true, the loop body is executed, otherwise, the loop is terminated.

Entry Controlled Loops are executed in the following order:

Condition Check: *The loop condition is checked before executing the loop body.*

Execution: *If the condition is true, the loop body is executed; otherwise, the loop terminates without executing the loop body.*

Initialization and Increment/Decrement: *For for loops, initialization, condition checking, and increment/decrement are all handled in the loop's syntax. For while loops, the initialization and increment/decrement must be handled separately within the loop's body or before and after the loop.*

For Loop

A for loop is a repetition control structure that allows us to efficiently write a loop that will execute a specific number of times.

The for loop working is as follows:

- *The initialization statement is executed only once; in this statement, we initialize a variable to some value.*
- *In the second step, the test expression is evaluated. Suppose the test expression is evaluated to be true. In that case, the for loop keeps running, and the test expression is re-evaluated, but if the test expression is evaluated to false, then the for loop terminates.*
- *The loop keeps executing until the test expression is false. When the test expression is false, then the loop terminates.*

For Loop

Syntax :

for(initialization; condition; increment/decrement)

{

Statements;

}

```c
#include <stdio.h>

int main() {
  int i;

  for (i = 0; i < 5; i++) {
    printf("%d\n", i);
  }

  return 0;
}
```

Exit Controlled Loop

In exit-controlled loops, the test condition is tested at the end of the loop. Regardless of whether the test condition is true or false, the loop body will execute at least once. The do-while loop is an example of an exit-controlled loop.