B A S I C O F C L A N G U A G E F R O M S E L F L E A R N I N G .

*Basics with exampled repository file.*

C LANGUAGE

B A S I C

Book Written By

Kaushal Bhatol

## Links

All documents, pictures and articles root links for preventing **copyright** issue.

YouTube playlist:

Site: [www.geeksforgeeks.org](http://www.geeksforgeeks.org)

## License

This Book is created for personal use only, **we don’t give any copyright security**.

This document is collection of data from many other sites.

All contents are not created by Author, So please don’t try to publish or shell this Book.

This book is not registered in any License.

## Operators [Week 1, 08]:

An Operators is a symbol used to perform operations in given programming language.

==> **types of operators**

Arithmetic operators

Relation Operators

Logical Operators

Bit-wise operators

Assignment operators

Miscellaneous

1. Arithmetic operator [week 1 : 08.1]

This operators are basic operators for day to day life.

|  |  |
| --- | --- |
| Operators | Description |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | modules |

1. Relation operator [week 1 : 08.1]

|  |  |
| --- | --- |
| Operators | Description |
| == | Is equal to |
| != | Is not equal to |
| > | Greater than |
| < | Less than |
| >= | Greater than |
| <= | Less than or equal to |

Note : mostly this operators are used in if, else statement.

1. Logical operator [week 2 : 08.2]

|  |  |  |
| --- | --- | --- |
| Operators | Description | Example |
| && | Logical “AND” operator, both the operands are non-zero then condition is true. | a&&b |
| || | Logical “OR” operator, if any of these two operands is none-zero, Then condition become true. | a||b |
| ! | Logical “NOT” operator, It is used to reverse the logical state of its operand, if condition is true.. | !a |

1. Bit-wise operator [week 1, 08.2]

*bit-wise operator are functioning on bit.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Value A | Value B | A&B | A|B | A^B |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Symbols Names

& AND

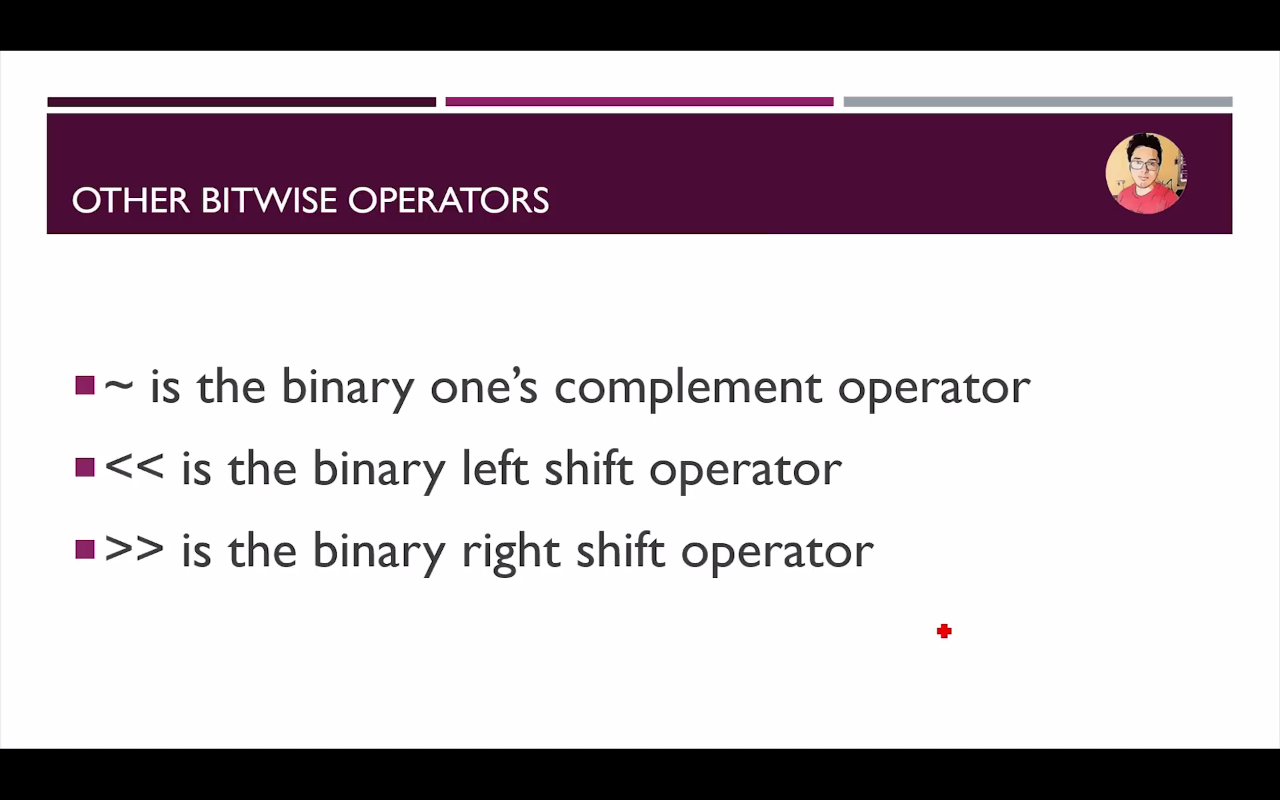
| OR

^ Exclusive (XOR)

**XOR Operator:**

If one value is true and second one is false, Then XOR give true value.

**other Bit-wise operators**



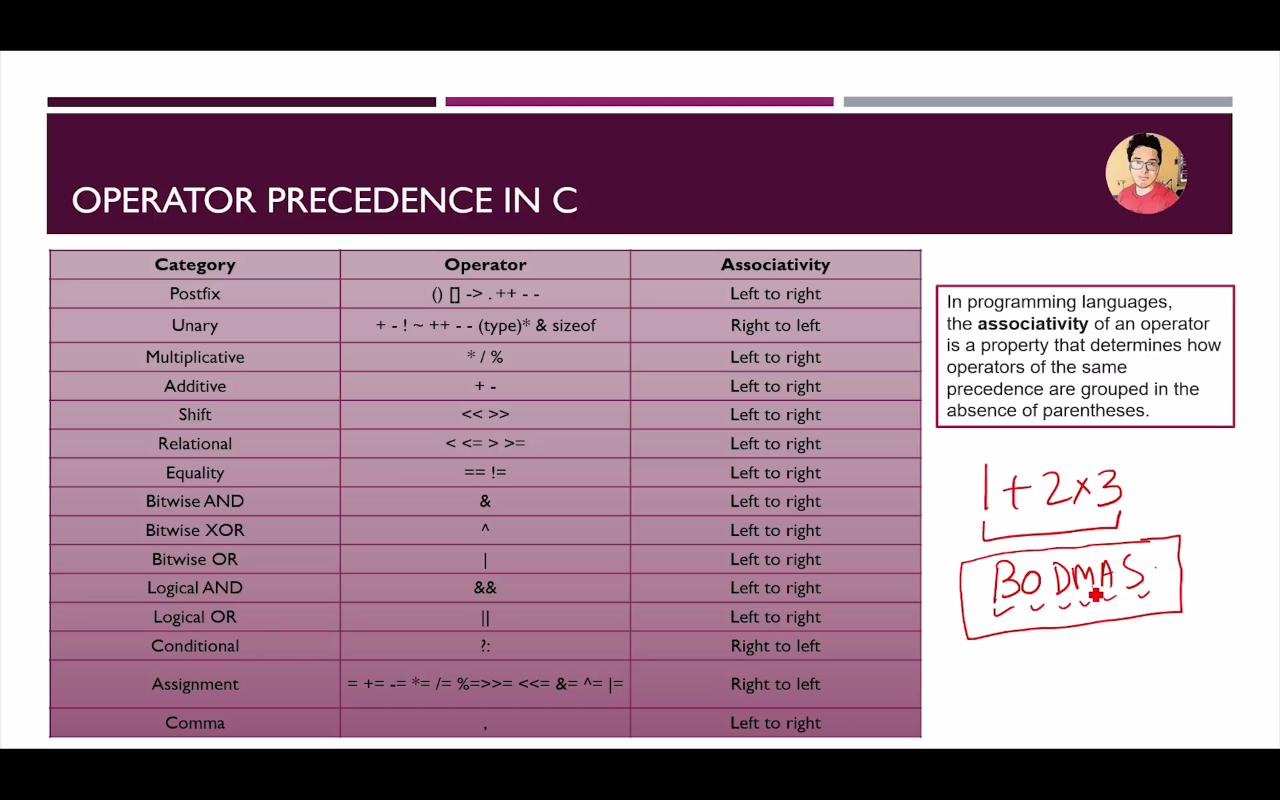
1. Assignment operator

|  |  |
| --- | --- |
| Operators | Description |
| = | Simple assignment operator assign value from right side operands |
| += | Add AND assignment operator. It’s adds the right operand to the left operands and assign the result to the left operands |
| -= | Subtract AND assignment operator. It subtracts the right operands from the left to the result is assignment to the left operand. |
| \*= | Multiply AND assignment operator. It multiples the right operands from the left to the result is assignment to the left operand. |
| /= | Divide AND assignment operator. It divide the right operands from the left to the result is assignment to the left operand. |

1. Miscellaneous

|  |  |  |
| --- | --- | --- |
| Operators | Description | Example |
| sizeof() | Returns the size of variable. | sizeof(a), where a is a integer, will return int’s size on that architecture . |
| & | Returns the address of a variable. | &a; returns the actual address of the variable. |
| \* | Pointer to variable | \*a; |
| ?: | Conditional Expression | If condition is true? The value X: otherwise value Y |

Operators PRECEDENCE IN C



## Format specifiers [week 1, 09]

|  |  |
| --- | --- |
| Format | Work |
| %c | Character print |
| %d | Integer print |
| %f | Float print |
| %l | Long print |
| %lf | Double print |
| %LF | Long Double print |

-> Format specifiers is a way to tell the compiler what type of data I sin a variable during taking input and displaying output to the user.

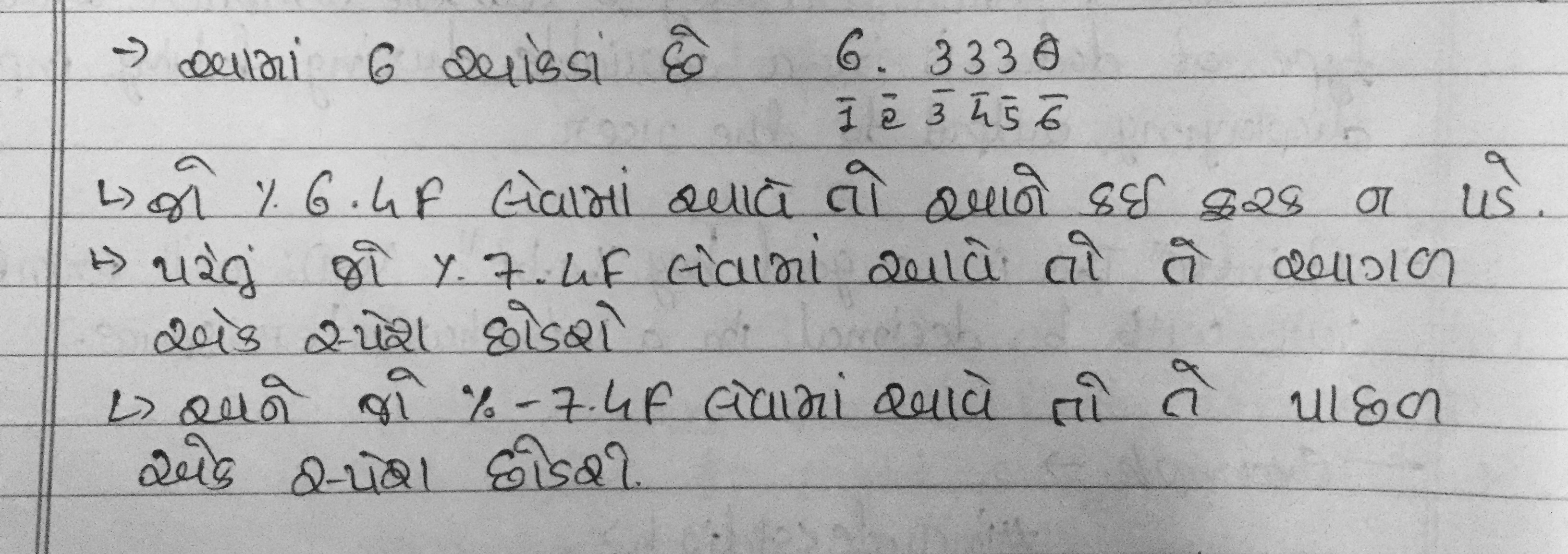
-> Pritf(“This is a good boy %a.bf”, var); will print vvar with b decimale in a “a” character space

-Exp->

Week 1, 09 .c file

float c= 6.333

Printf(“%7.4f”, c);



(Avoid above picture if you don’t know Guajarati language.)

-> Note: 0 is always included in format

-> It work with - also.. exp: %-2.2f

->It only work if value is less then applied specifier. Else it show real values.

## Constant [Week 1, 10]

*-> A constant is a value or variable that can’t be changed in the program.*

Example: 15, ‘a’, 3.4, “Name, etc..

-> There are Two ways to define constant in C Program.

1. Constant Keyword

Exp: const float b = 7.499;

1. #define preprocessor

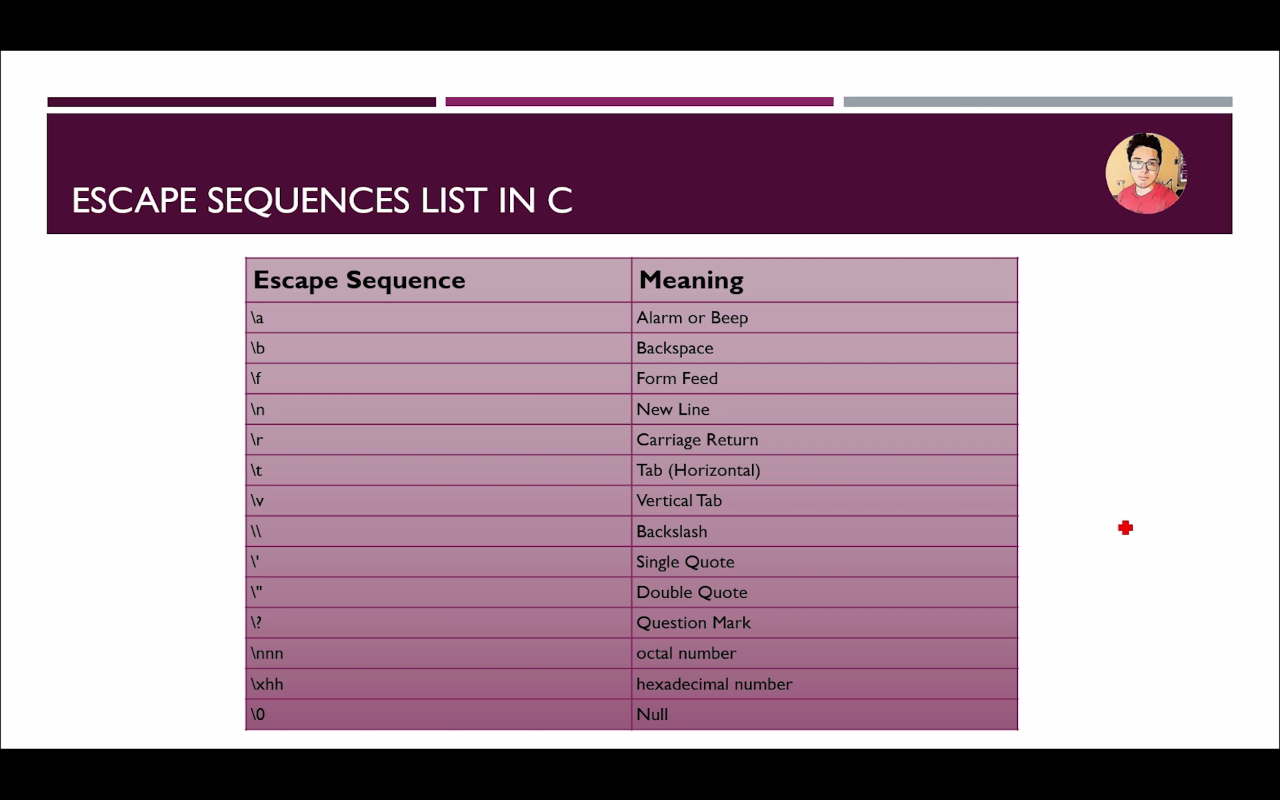
Exp: #define PI 3.14

## Escape sequence [Week 1]

**->** *an escape sequeance inn C programming, languaage is a sequance of characters.*

-> It dosen’t represent itself when used inside string literal or character.

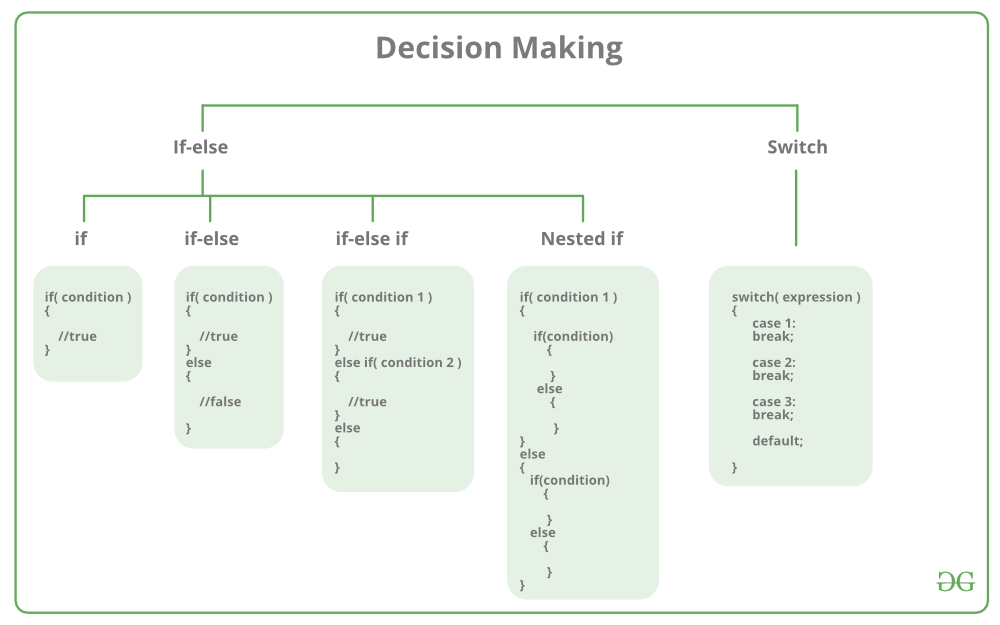
-> It is composed of two or more characters string with backslash \.



## if else [week 1, 11]

**if else [11]**

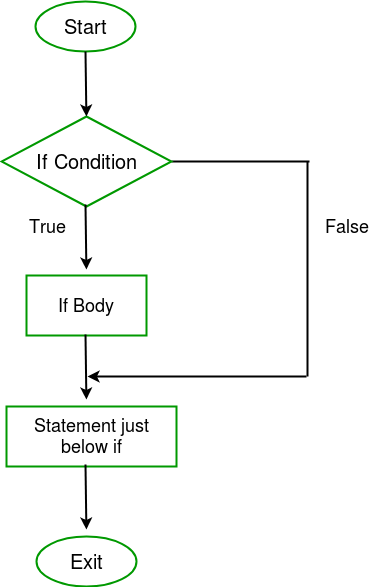
* *It is used to perform operation based on some condition*
* *Types of if statement:*
  + If statement
  + If else statement
  + if-else-if ladder
  + Nested if statement

Source: https://www.geeksforgeeks.org/

1. If statement [11.1]:

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

**Syntax**: 

if(condition)

{

// Statements to execute if

// condition is true

}

Here, **condition**after evaluation will be either true or false. C if statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not. If we do not provide the curly braces ‘{‘ and ‘}’ after if(condition) then by default if statement will consider the first immediately below statement to be inside its block.   
**Example**: 

if(condition)

statement1;

statement2;

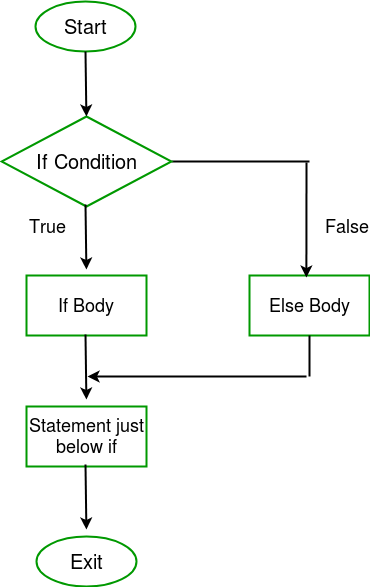
// Here if the condition is true, if block

// will consider only statement1 to be inside

// its block.

2. If else [11.1]:

The *if*statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won’t. But what if we want to do something else if the condition is false. Here comes the C *else*statement. We can use the *else*statement with *if*statement to execute a block of code when the condition is false.   
**Syntax**: 

if (condition)

{

// executes this block if

// condition is true

}

else

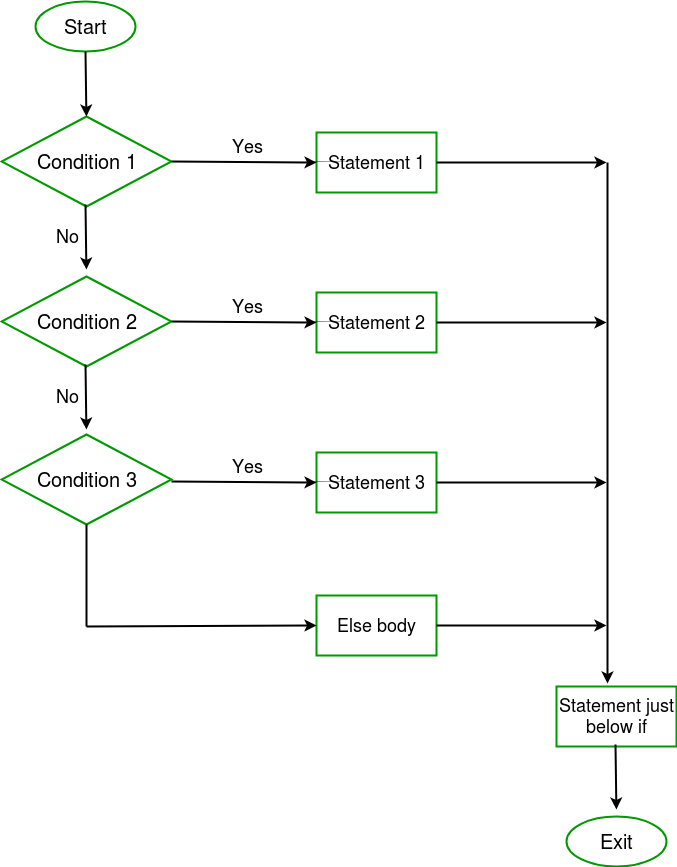
{

// executes this block if

// condition is false

}

3. if-else-if ladder [11.1]:

The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions are true, then the final else statement will be executed. 

**Syntax:** 

if (condition)

statement;

else if (condition)

statement;

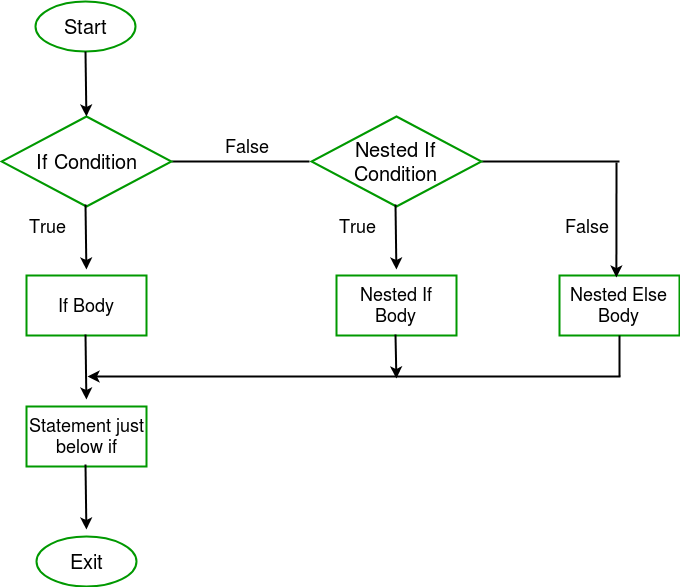
.

.

else

4. Nested if statement [11.2]:

A nested if in C is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement. Yes, both C and C++ allows us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

  
**Syntax:** 

if (condition1)

{

// executes when condition1 is true

if (condition2)

{

// executes when condition2 is true

}

}

**Note**: when you want a is equal to some value use “==” instead of “=”, else it don’t go on next steps and print the”=” condition in all situation.

## Switch Case [week 1, 12]

switch-case-in-java**Rules:**

1. Switch expression must be int or char.

2. Case value must be int or char.

3. Case must come inside switch.

4. Break is not a must.

**Note**: If break is not added, code will execute below cases also. And Stop when it get break in any case.

**Types:**

1. Switch case without break.

2. Switch case.

3. Nested switch case.

**Syntax:**

switch (n)

{

case 1: // code to be executed if n = 1;

break;

case 2: // code to be executed if n = 2;

break;

default: // code to be executed if n doesn't match any cases

}

## Loops [week 1, 13]

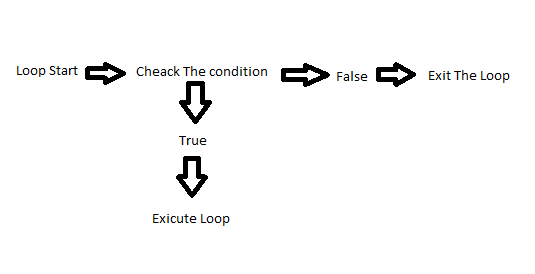
**Advantage of Loops**

* Code reusability
* Saves time
* Traversing

**Types of Loops**

* Do while loop
* While loop
* For loop

**Basic syntax:**

****

1. Do While Loop [13.1]

**Syntax:**

do{

// Code to be Executed

} while(condition);

**Note:**

* Do while loop execute once time without checking condition.

**Example:**

* Week 1, 13.1 do while loop.c

**Exercise:**

* Write Multiplication table for users input number.

*a. Exercise / 05 multiplication do while loop.c*

2. While Loop

**Syntax:**

while(condition){

// code to be executed

}

**Note:**

While loop check condition first then execute.

**Example & Exercise:**

* Week 1, 13.2 file
* Print numbers 0 to user input. (A. Exercise /06 file)
  + Maximum 50 allowed
  + Validate for positive values only.

3. for Loop