Math Series

C Programming

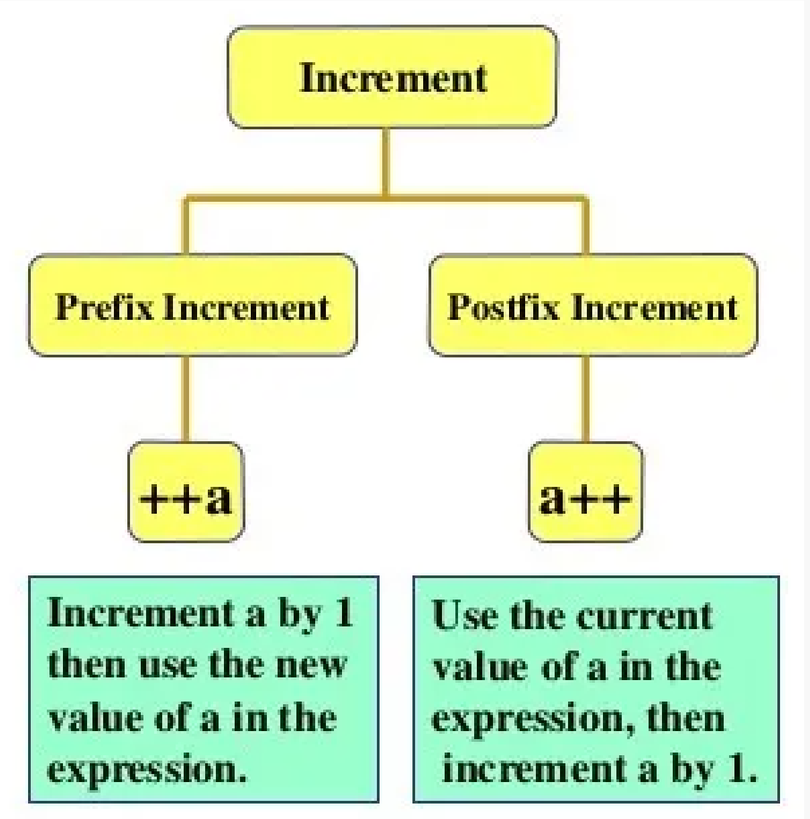
UG Sem-3 Major (Kalyani University)

Day - 03

Increment Operator

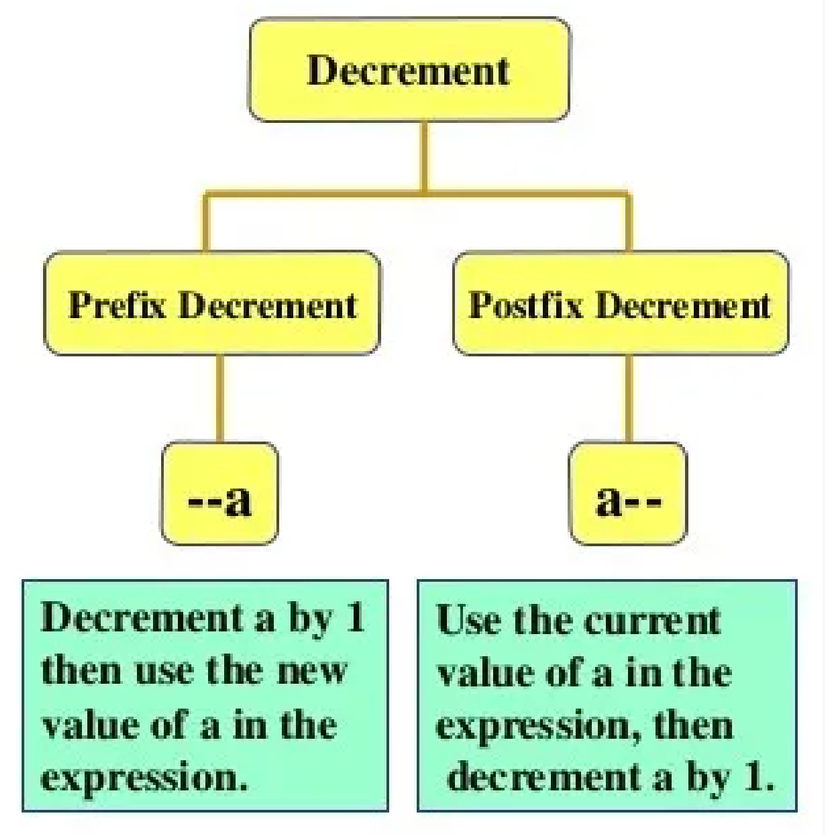
1. Increment operator is used to increment the current value of variable by adding integer 1.
2. Increment operator can be applied to only variables.
3. Increment operator is denoted by ++.

We have two types of increment operator i.e Pre-Increment and Post-Increment Operator.



Decrement Operator

Decrement operators in C language are used to reduce the value of a variable by one. Represented by the symbol --, they have two forms: prefix (--a) and postfix (a--).



Increment and Decrement Operators in C

Operator	Type	Description
++a	Prefix Increment	Increments the value of a by 1, then uses it in the expression.
a++	Postfix Increment	Uses the current value of a in the expression, then increments a by 1.
--a	Prefix Decrement	Decrements the value of a by 1, then uses it in the expression.
a--	Postfix Decrement	Uses the current value of a in the expression, then decrements a by 1

Examples of Increment and Decrement Operators

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = ++a;  // Increments a, then assigns it to b
    printf("Value of a: %d\n", a);  // Output: 6
    printf("Value of b: %d\n", b);  // Output: 6
    return 0;
}
```

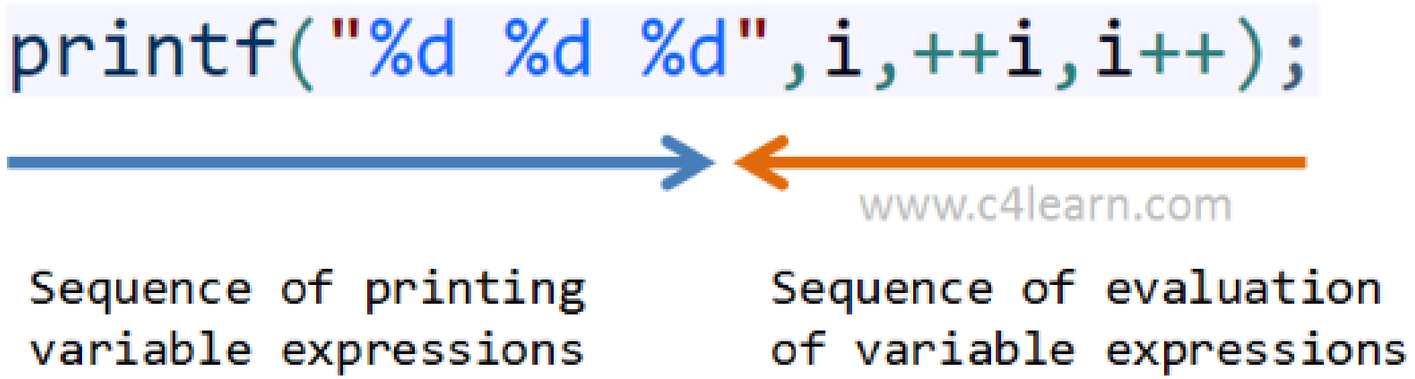
Examples of Increment and Decrement Operators

Multiple increment operators inside printf

```
#include<stdio.h>
void main() {
  int i = 1;
  printf("%d %d %d", i, ++i, i++);
}
```

Output : 3 3 1

Pictorial representation



Explanation

1. Whenever more than one format specifiers (i.e %d) are directly or indirectly related with same variable (i,++i,i++) then we need to evaluate each individual expression from right to left.

2. As shown in the above image evaluation sequence of expressions written inside printf will be – i++,++i,i

3. After execution we need to replace the output of expression at appropriate place

No	Step	Explanation
1	Evaluate i++	At the time of execution we will be using older value of i = 1
2	Evaluate ++i	At the time of execution we will be increment value already modified after step 1 i.e i = 3
2	Evaluate i	At the time of execution we will be using value of i modified in step 2

Conditional Operator/ Ternary operator:

conditional operator checks the condition and executes the statement depending of the condition. A conditional operator is a ternary operator, that is, it works on 3 operands.
Conditional operator consist of two symbols.
1 : question mark (?).
2 : colon (:).

Syntax : condition ? exp1 : exp2;

```c
#include <stdio.h>

int main() {
    int a = 10, b = 20;
    int max;

    // Use ternary operator to find the maximum value
    max = (a > b) ? a : b;

    printf("The maximum value is: %d\n", max); // Output: The maximum value is: 20

    return 0;
}
```

Operator Precedence :

Arithmetic Operators are evaluvated left to right using the precedence of operator when the expression is written without the paranthesis.They are two levels of arithmetic operators in C.

*1 : High Priority * / %*

2 : Low Priority + -.

Arithmetic Expression evaluation is carried out using the two phases from left to right.

1 : First phase : The highest priority operator are evaluated in the 1st phase.

2 : Second Phase : The lowest priority operator are evaluated in the 2nd phase.

*Ex : a=x-y/3+z*2+p/4.*
x=7, y=9, z=11, p=8.
*a= 7-9/3+11*2+8/4.*

1st phase :
*1 : a = 7-3+11*2+8/4*
2 : a = 7-3+22+8/4
3 : a = 7-3+22+2

2nd phase :
1 : a = 4+22+2
2 : a = 26+2
3 : a = 28

Operator Precedence and Associativity :

Operator	Description	Precedence	Associativity
( ) [ ]	Function call Square brackets.	1	L-R (left to right)
+ - ++ -- ! ~ * & sizeof	Unary plus Unary minus Increment Decrement Not operator Complement Pointer operator Address operator Sizeof operator	2	R-L (right to left)
* / %	Multiplication Division Modulo division	3	L-R (left to right)
+ -	Addition Subtraction	4	L-R (left to right)
<< >>	Left shift Right shift	5	L-R (left to right)

Operator Precedence and Associativity :

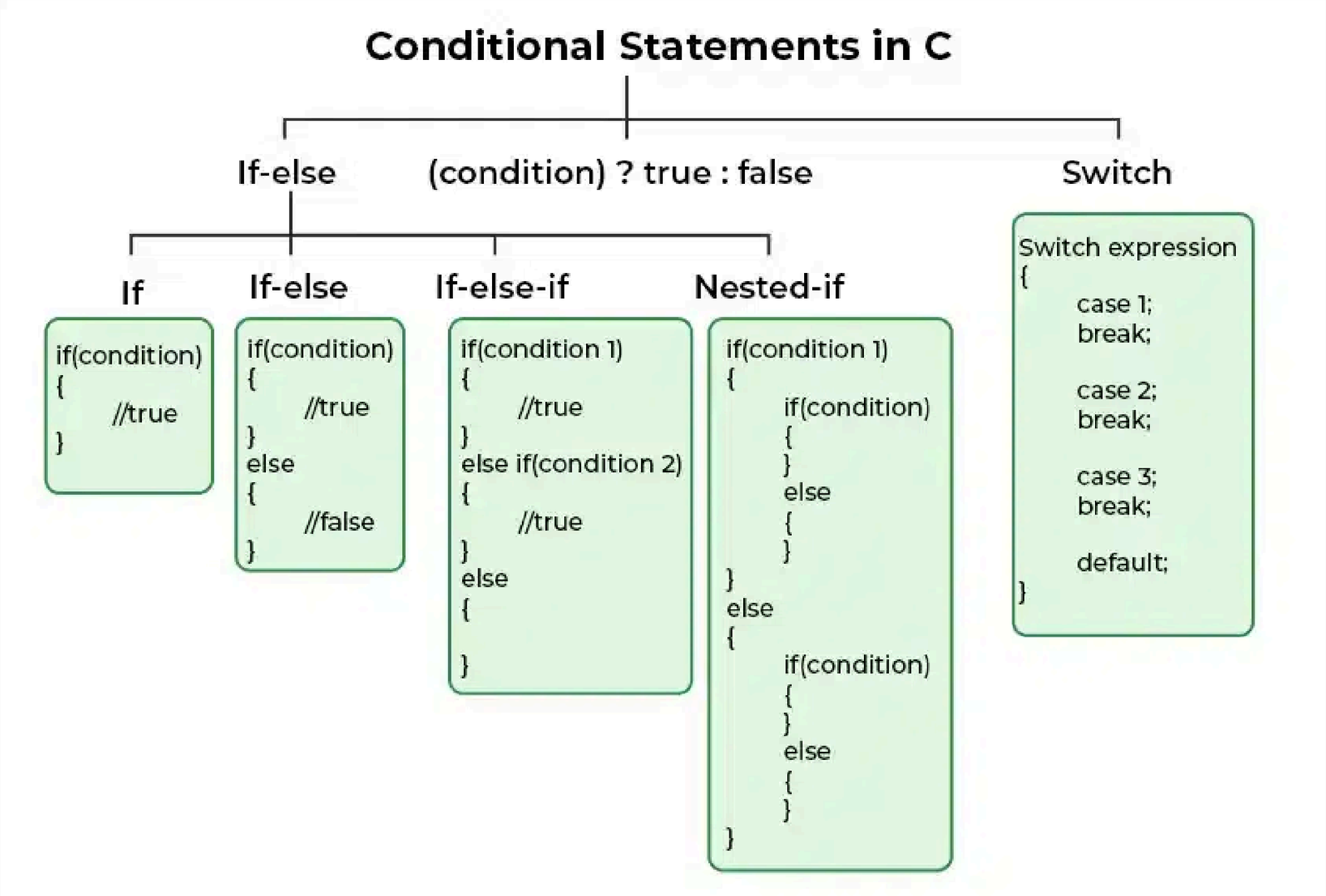
< <= > >=	Relational Operator	6	L-R (left to right)
= = !=	Equality Inequality	7	L-R (left to right)
&	Bitwise AND	8	L-R (left to right)
^	Bitwise XOR	9	L-R (left to right)
|	Bitwise OR	10	L-R (left to right)
&&	Logical AND	11	L-R (left to right)
||	Logical OR	12	L-R (left to right)
?:	Conditional	13	R-L (right to left)
= *= /= %= += -= &= ^= <<= >>=	Assignment operator	14	R-L (right to left)
,	Comma operator	15	L-R (left to right)

Decision Making Statements

A selection statement selects among a set of statements depending on the value of a controlling
expression. Or, Moving execution control from one place/line to another line based on condition
Or, Conditional statements control the sequence of statement execution, depending on the value of a integer expression

C‟ language supports two conditional statements.
1: if
2: switch.

Decision Making Statements



If Statement

The if statement is the simplest 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 statements is executed otherwise not. A condition is any expression that evaluates to either a true or false (or values convertible to true or false).

Syntax :

```
if(condition/expression)
{
true statement;
}
 statement-x;
```

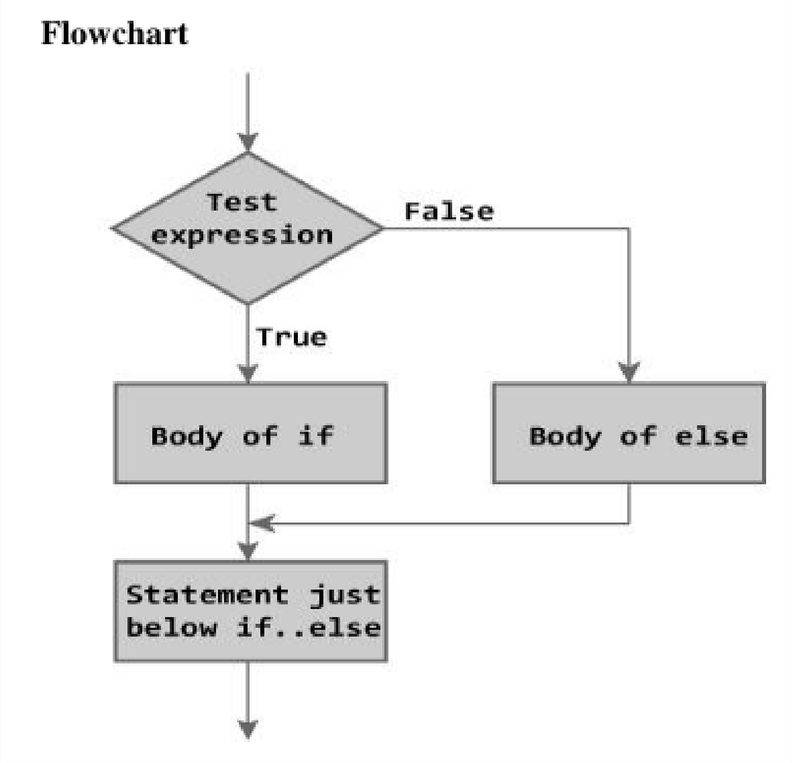
```c
#include <stdio.h>

int main() {

    int age = 20;

    // If statement
    if (age >= 18) {
        printf("Eligible for vote");
    }
}
```

If-else Statement

The if-else statement is an extension of the simple if statement. The general form is. The if...else statement executes some code if the test expression is true (nonzero) and some other code if the test expression is false (0).

Syntax :

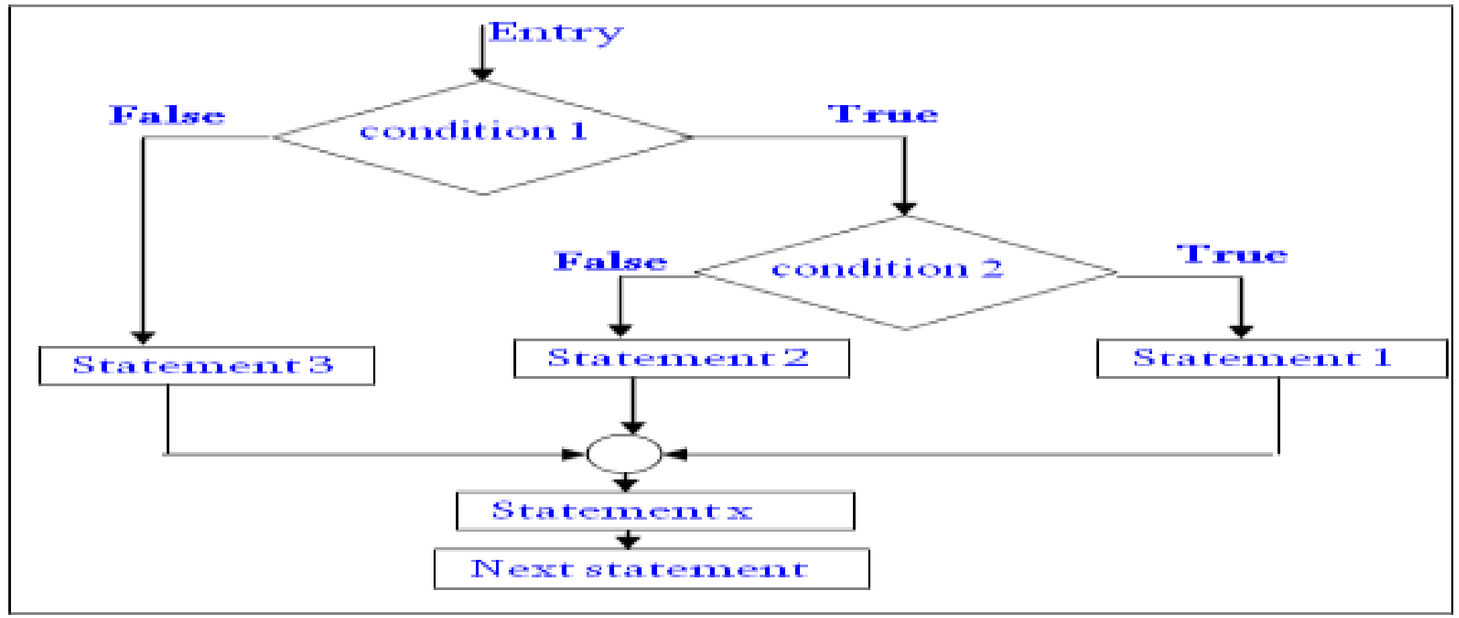
```
if (condition)
{
true statement;
}
else
{
false statement;
}
statement-x;
```



Nested if-else statement

When a series of decisions are involved, we may have to use more than on if-else statement in nested form. If –else statements can also be nested inside another if block or else block or both.

Flowchart



Syntax

```
if ( condition1 )
{
   if ( condition2 )
   {
      ....
      True block of statements 1;
   }
   ....
}
else
{
    False block of condition1;
}
```