CS -01

PROBLEM SOLVING METHODOLOGIES AND PROGRAMMING IN C

By Rachel

UNIT 1- PART 3

TOPIC 5

C CHARACTER SET

C Character Set is divided into 4 parts.

1. Letters 2. Digits 3. Special Characters 4. White spaces

Letters	*Digits*
Uppercase A.....Z	All decimal digits 0 9
Lowercase a.....z	

Special Characters

, comma	& ampersand
. period	^ caret
; semicolon	* asterisk
: colon	– minus sign
? question mark	+ plus sign
' apostrophe	< opening angle bracket
" quotation mark	(or less than sign)
! exclamation mark	> closing angle bracket
| vertical bar	(or greater than sign)
/ slash	(left parenthesis
\ backslash	) right parenthesis
~ tilde	[left bracket
_ under score	] right bracket
$ dollar sign	{ left brace
% percent sign	} right brace
	# number sign

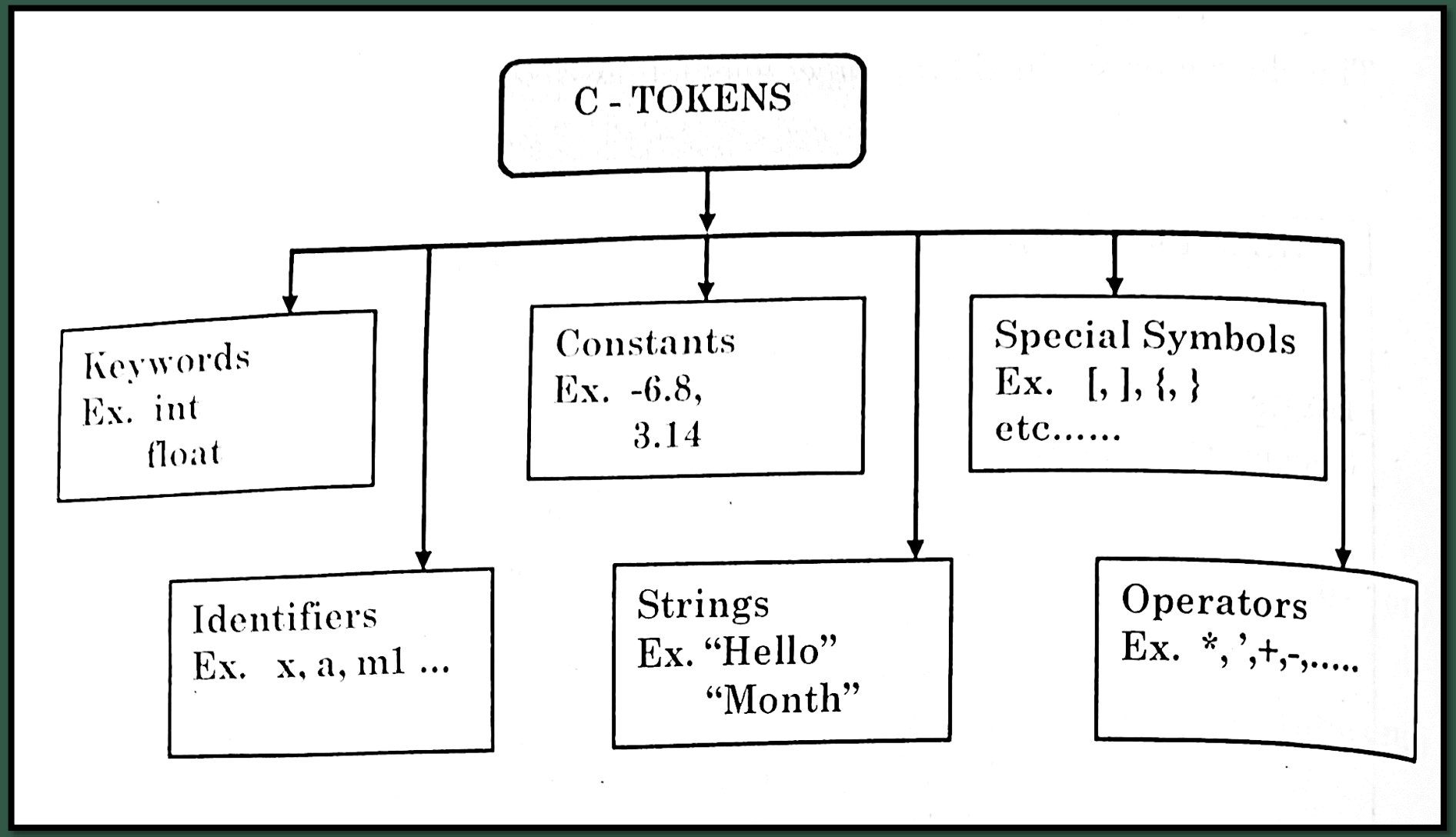
White Spaces

Blank space
Horizontal tab
Carriage return
New line
Form feed

TOPIC 6

C TOKENS

C programs are made of various keywords, operators and special symbols. They are called tokens. They are the building blocks for C language.



1. Keywords

- There are 32 keywords in C. They are the special words which have specific meaning.
- They are the first building block of writing C programs.
- All keywords are in lowercase.
- No keyword contain space, underscore or number.
- Maximum length is 8.
- Examples :

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

2. Identifiers

- Identifier is the name of a variable, constant, structure, union, arrays or function given by the programmer.
- Rules for naming identifiers:
  - It can contain letters, digits and underscore.
  - It should start with a letter, not a digit.
  - It should not be the name of any keyword.
  - It can have maximum 31 characters without space.
  - It should not have space, comma or any other special symbol.
  - It is case sensitive. Upper case and lower case is different.

Examples : a1, x_y, Abc, Name

3. Variable

- Variable is an identifier for data name that can be used to store values during the program. It takes different values at different time during the execution of the program. It should be meaningful.
- It is used in place of actual memory location.
- Declaring a variable - Declaration contains Name and Type.

  Syntax : Data type v1,v2,v3,….,vn
- Assigning values to variable - To assign a value to the variable = (assignment operator) is used.

  Syntax : variable name = constant (any value)

Example:

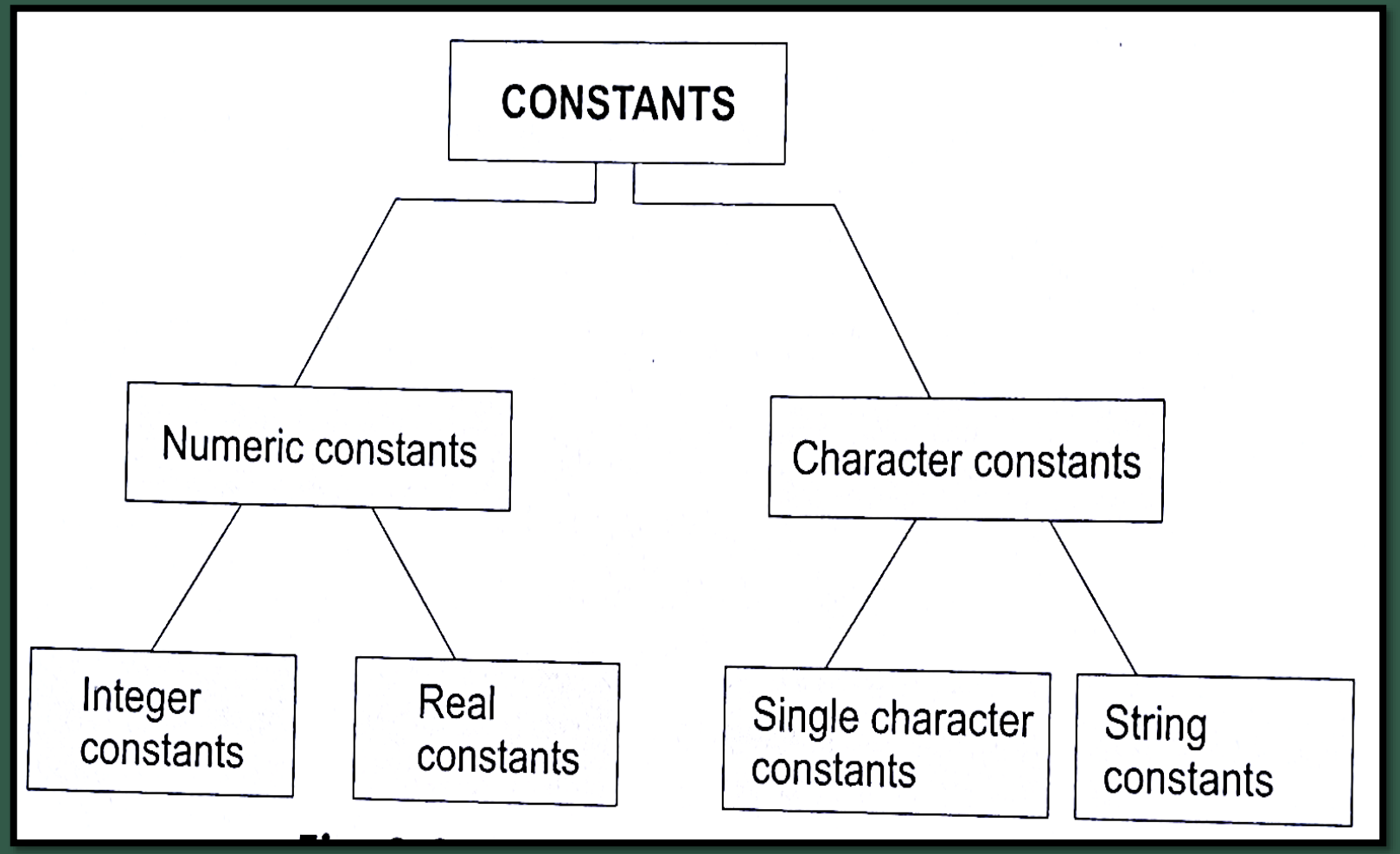
```
main ()
{
    int a ;
    char b;
    float c;          } Declaration Statements.

    int d=20;
    long int e=21;    } Declaration & Assignment Statements.
    char f='A';

    a=78;
    b='$'             } Assignment Statements.
    c=29.678;
}
```

4. Constants

Constants in C refer to fixed values that do not change during the execution of a program.



Constants	Examples
Integer Constant	Decimal Integer Constant: 1, 3, 7, 8, 65, 543676664 Octal Integer Constant: 037, 0320, 0456, 0552, 0432 Hexadecimal Integer Constant: 0x4, 0X456, 0x552, 0x43
Real Constant	`-2.0, 2.15, -.71, +.5, 0.0000234, -0.22E-5` `6.65e4, 1.5e+5`
Character Constant	'a' , 'm' , 'F' , 'A' , 'B', 'C' , 'Z'
String Constant	**"ABCD" , "Hi ","hello world",** **"i love java programming“**

5. Operators

An operator is a symbol that causes specific mathematical or logical manipulations to be performed. They are of 8 types:

1. Arithmetic operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increments and Decrement Operators
6. Conditional Operators
7. Bitwise Operators
8. Special Operators

1. Arithmetic operator

Integers, single precision floating point numbers(32 bits) and double precision floating point numbers(64 bits) can be added, subtracted, divided or multiplied using arithmetic operators.

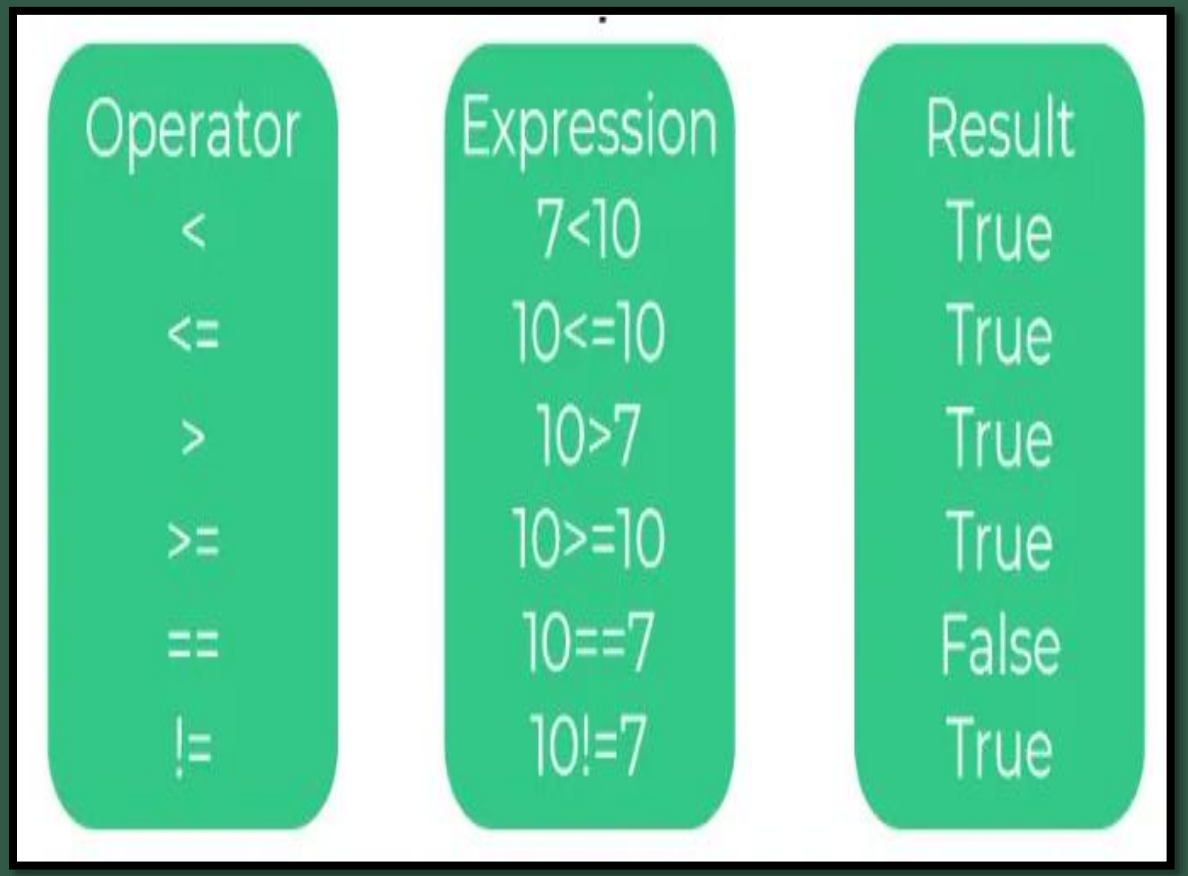
If value of x = 100 and y = 20

Operator	Operation	Result
+ (Addition)	z = x + y	z = 120
- (Subtraction)	z = x - y	z = 80
* (Multiplication)	z = x * y	z = 2000
/ (Division)	z = x / y	z = 5
% (Modulus)	z = x % y	z = 0

2. Relational operator

Relational operators are used to test or compare the values between two operands. If the condition is false, then the integer result is 0 and if the condition is true, then the integer result is 1.

Operator Symbol	Operator Name
==	Equal To
!=	Not Equal To
<	Less Than
>	Greater Than
<=	Less Than Equal To
>=	Greater Than Equal To



3. Logical operator

Logical operators are used to combine two or more relations. They are called Boolean operators because the test between values are reduced to either true (1) or false (0).

Operator	Description
&&	AND
||	OR
!	NOT

Truth Table			
Expr 1	Expr 2	Expr 1 && Expr 2	Expr 1 || Expr 2
1(T)	1(T)	1(T)	1(T)
1(T)	0(F)	0(F)	1(T)
0(T)	1(T)	0(F)	1(T)
0(T)	0(F)	0(F)	0(F)

Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
||	Logical or	(5<2)||(5>3)	True
!	Logical not	!(5<2)	True

4. Assignment operator

- It has the form : variable = expression;
- C has a set of shorthand assignment operators.
  It has the form : variable op = expression;

Ex.	x += 3	means	x = x + 3
	x * = 3	means	x = x * 3
	x /= 5	means	x = x / 5
	x -= 8	means	x = x – 8
	x %= 2	means	x = x % 2

5. Increment and Decrement operator

- The increment and decrement operators are add one and subtract one.
- They can increment and decrement the value either before or after the value of the variable.
- They can be either prefix or postfix.

++ = Increment Operator (it means + 1)
-- = Decrement Operator (it means – 1)

Ex. m = 5;
y = ++m;
Then, y = 6 (m+1) and m=6(++m)
Again :
m = 5;
y = m++;
Then, y = 5 and m = 6

6. Conditional operator

- The conditional operator is ? : and it is used for checking the condition and gives output according to the condition as true or false.
- It is also called Turnery operator.
- It has the form : condition ? True value : False value
- Example :

```
a = 10;
b = 8;
x = (a>b) ? a+2 : b-2
```

Here, the x value will be 12.

7. Bitwise operator

- There are 6 bitwise manipulators which are used for the manipulation of individual bits.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise EXOR
>>	Bitwise Shift Right
<<	Bitwise Shift Left
~	Bitwise NOT

8. Special operators

- Special operators are for some special processes in C programming. Some of special operators are

Comma operator (,)
Size of operator
Pointer operator (& and *)
Member selection (. and →)

Program

```c
// Working of arithmetic operators
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);
    c = a-b;
    printf("a-b = %d \n",c);
    c = a*b;
    printf("a*b = %d \n",c);
    c = a/b;
    printf("a/b = %d \n",c);
    c = a%b;
    printf("Remainder when a divided by b = %d \n",c);
}
```

Output

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b = 1
```

Program

```c
// Working of increment and decrement operators
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

    printf("++a = %d \n", ++a);
    printf("--b = %d \n", --b);
    printf("++c = %f \n", ++c);
    printf("--d = %f \n", --d);
}
```

Output

```
++a = 11
--b = 99
++c = 11.500000
--d = 99.500000
```

Program

```c
// Working of assignment operators
#include <stdio.h>
int main()
{
    int a = 5, c;

    c = a;          // c is now 5
    printf("c = %d\n", c);
    c += a;         // c is now 10
    printf("c = %d\n", c);
    c -= a;         // c is now 5
    printf("c = %d\n", c);
    c *= a;         // c is now 25
    printf("c = %d\n", c);
    c /= a;         // c is now 5
    printf("c = %d\n", c);
    c %= a;         // c is now 0
    printf("c = %d\n", c);
}
```

Output

```
c = 5
c = 10
c = 5
c = 25
c = 5
c = 0
```

Program

```c
// Working of relational operators
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;

    printf("%d == %d is %d \n", a, b, a == b);
    printf("%d == %d is %d \n", a, c, a == c);
    printf("%d > %d is %d \n", a, b, a > b);
    printf("%d > %d is %d \n", a, c, a > c);
    printf("%d < %d is %d \n", a, b, a < b);
    printf("%d < %d is %d \n", a, c, a < c);
    printf("%d != %d is %d \n", a, b, a != b);
    printf("%d != %d is %d \n", a, c, a != c);
    printf("%d >= %d is %d \n", a, b, a >= b);
    printf("%d >= %d is %d \n", a, c, a >= c);
    printf("%d <= %d is %d \n", a, b, a <= b);
    printf("%d <= %d is %d \n", a, c, a <= c);
}
```

Output

```
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
5 >= 10 is 0
5 <= 5 is 1
5 <= 10 is 1
```

Program

```c
// Working of logical operators
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) is %d \n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) is %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) is %d \n", result);

    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) is %d \n", result);

    result = !(a != b);
    printf("!(a != b) is %d \n", result);

    result = !(a == b);
    printf("!(a == b) is %d \n", result);
}
```

Output

```
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
```

TOPIC 7

HIERARCHY OF OPERATORS

Operator	*Description*	*Associativity*	*Rank*
( ) [ ]	Function call Aray element reference	Left to right	1
+ – ++ –– ! ~ * & sizeof (type)	Unary plus Unary minus Increment Decrement Logical negation Ones complement Pointer reference (indirection) Address Size of an object Type cast (conversion)	Right to left	2
* / %	Multiplication Division Modulus	Left to right	3

+ −	Addition Subtraction	Left to right	4
<< >>	Left shift Right shift	Left to right	5
< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	Left to right	6
== |=	Equality Inequality	Left to right	7
&	Bitwise AND	Left to right	8
^	Bitwise XOR	Left to right	9
|	Bitwise OR	Left to right	10
&&	Logical AND	Left to right	11
||	Logical OR	Left to right	12
?:	Conditional expression	Right to left	13
= * = /= %= += –= &= ^= |= <<= >>=	Assignment operators	Right to left	14
,	Comma operator	Left to right	15