BASIC CONSTRUCTS OF C

EXPRESSIONS

- An expression is a combination of variables, constants (also called as operands) and operators.
- In C every expression evaluates to a value i.e., every expression results in some value of a certain type.
- If want to store the value of the expression we can assign it to a variable.
- Depending upon the number of operands, an operator can be classified as unary, binary or ternary.

OPERATORS

Operators in C

Operation Type	Operator's Type	Operators
Unary Operators	increment, Decrement Operators	++, --
Binary Operators	Arithmatic Operators	+, -, *, /, %
	Logical Operators	&&, ||, !
	Relational Operators	<, <=, >, >=, ==, !=
	Bit-wise Operators	&, |, <<, >>, ~, ^
	Assignment Operators	=, +=, -=, *=, /=, %=
Ternary Operator	Ternary or Conditional Operator	? :

ARITHMETIC OPERATORS

- Used to perform arithmetic operations
- The following symbols are used to perform basic arithmetic operations:

  + : Addition

  - : Subtraction

  * : Multiplication

  / : Division (Gives the quotient)

  % : Modulus (Gives the remainder)

ARITHMETIC OPERATORS

- Arithmetic operators have the following order of precedence:
  1) %, *, /
  2) +,-
- However, if there are more than one operators of equal precedence, it will be evaluated from left to right.
- Example:

a+b+c-d ⟶ (((a+b) +c)-d)

ASSIGNMENT OPERATORS

- Syntax:

L.H.S = R.H.S

- The R.H.S can be a :
  - Variable
  - Constant
  - Expression
- The L.H.S of an assignment operator can only be a variable.
- This is because only variables have memory capacity to store a value.

ASSIGNMENT OPERATOR

- Assignment operators can also be complex assignment operator.
- Example:
  - L.H.S += R.H.S actually means, L.H.S = L.H.S+R.H.S
- All assignment operators have equal precedence
- If there are more than one assignment operator, it is executed from right to left

QUICK EXERCISE

- Predict the output of the following code:

```
int x=5, y=6,z=8;
x=y=z;
printf("%d %d %d ", x,y,z);
```

  Output: 8 8 8

- Write a program to input a single two digit number and print the digits separately.

  Example:

  Enter a single two-digit number

  47<------- input by the user

  The two digits are : 4 and 7

RELATIONAL OPERATORS

- Sometimes we might want to compare two different things in the program
- Relational operators are:

  >,<,>=,<=,==,!=
- The relational expression generates the output as true/false i.e a Logical 1 or a Logical 0
- What if we want to do multiple comparisons in the same expression?
  - Whether a number lies between 10 and 20

LOGICAL OPERATORS

- Consider the previous example

  $10<x<20$ ✘
- When we have multiple comparisons in a single expression, we combine them using a logical operator (&&, ||)
- The operators follow the truth table of a logical AND and Logical OR
- Thus like relational expressions, the logical expressions also generate a Logical 0 or 1.
- Any non-zero value is considered as a Logical 1 and a 0 is considered as a Logical 0.
- If the value of the first expression generates a definite value, the second expression is not evaluated.

RELATIONAL AND LOGICAL OPERATORS

- The order of precedence of relational operators are:
  - <, <=, >,>=
  - ==,!=
- The order precedence of both logical operators are the same.
- When both relational operators are used in the same expression,
  - Firstly the relational operators are evaluated from left to right
  - Secondly, the logical operators are evaluated from left to right.

QUICK EXERCISE

- Find the output of the following code:

```
int x=8,y=4,z,c=-4,d;
d = (x>y)&& (c);
z = ( x!=y) || ( c=0);
printf("%d %d %d", d,z,c);
```

Output:

1 1 -4

TERNARY OPERATOR

- Also called as conditional operator.
- Syntax:

  condition ? expression1 : expression2;
- The condition is an expression that yields a true/false (Logical 0/Logical 1).
- If it yields a true, expression 1 will be executed,
- If it yields a false, expression 2 will be executed.
- Can be combined with assignment operator a follows:

variable = (condition ? expression1: expression2);

QUICK EXERCISE

- Find the output:

```
int x=3, y=10, z,sum;
z= x>y ? x+4 : y*2;
sum = z+=6;
printf("%d %d", z, sum);
```

Output:

26 26

- Program to input a two digit number and divide the larger digit by the smaller digit.
- Program to find if a number is divisible by 6 or not.