UNIT-02

Operators

Operator:-

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. The operators operate on the available operants in a program.

C language is rich in built-in operators & provides the following type of operators:-

- ⊙ Unary Operator
- ⊙ Binary Operator
- ⊙ Terenary Operator

⇒ Unary Operator:

Unary operator are operators that act upon a single operant to produce a new value.

* Types of Unary Operator:

↳ **Unary Minus:**

The minus (-) operator changes

the sign of its argument. A positive number becomes negative & a negative number becomes positive.

Example:-
```
int a = 10;
int b = -9;
// b = -10;
```

↳ Increment:

It is used to increment the value of the variable by 1. The increment can be done in 2 ways:-

1) Prefix increment:

In this method the operator precedes the operant (eg. ++a) the value of the operant will be altered before it is used.

Example:-
```
int a = 1;
int b = ++a;
// b = 2;
```

2) Postfix increment:

In this method the operator follows the operants (eg. a++) the value operant will be altered after it is used.

Example:-
```
int a = 1;
int b = a++;
// c = a;        // b = 1
                 // c = 2.
```

↳ Decrement:
It is used to decrement the value of the variable by 1. The decrement can be done in 2 ways:-

1) Prefix decrement:
In this method the operator precedes the operant (--a). The value of the operant will be altered before it is used.

Example:-
```
int a = 1;
int b = --a;
// b = 0
```

2) Postfix decrement:
In this method the operator follows the operant (a--). The value of the operant will be altered after it is used

Example:-
```
int a = 1;
int b = a--;      // b = 1
int c = a;        // c = 0
```

↳ NOT (!) :-

Not is unary.

It is used to reverse the logical state of its operand. If a condition is true then the logical NOT operator will make it false

Example:- if x is true, then
! x is false.
if x is false, then
! x is true.

```
int a=10;
int b=5;
if (!a>6)
{
printf("b is greater than a");
}
else
printf("a is greater than b");
getch();
}
```

↳ Address of operator (&) :

It gives an address of a variable. It is used to return the memory address of a variable. These addresses return by the address of operator are known as pointers because they point to the variable in memory.

Example :- int a ;
int * ptr ;
ptr = &a ; // address of a is copied to ptr location ptr.

6) Sizeof() :
This operator returned the sizeof() its operant in bytes. The sizeof() operator precedes its operant. The operant is an expression or it may be a cast.

Example : int main()
{
float n = 0 ;
printf("Size of (n)") ;
return 1 ;
}

⇒ Binary Operator

↳ Arithmatic Operator :
These operators are used to perform arithmatic / mathematical operations on operants.
Such as addition (+), subtraction (-), multiplication (*), division (/) modules etc. on the given operants.

Example:- 5+3 = 8, 5-3 = 2,
2*4 = 8 etc.

↳ **Relational Operators:**
Relational operators are used for comparison of the values of two operands
Example:- checking if one operand is equal to the other operand or not, whether an operand is greater than the other operand or not.
Some of the relational operators are (==, >=, <=).

```
int a = 3
int b = 5
(a>b)   // operator to check
           if a is greater than b.
```

↳ **Logical Operators:**
They perform logical operations on a given expressions by joining 2 or more expressions (conditions) it can be used in various relational & conditional expressions. This operator is based on boolean values to logically check the conditions & if the condition are true it returns 1, otherwise it returns 0 (false). In C programming logical operator

are classified into 3 types such as:-

* The logical "AND" operator (&&),
* The logical "OR" operator (||),
* The logical "NOT" operator (!).

→ **Logical "AND" Operator:**

The logical "AND" operator is represented as the double ampheresant (&&) symbol. It checks the condition of 2 or more operants by combining in an expression & if all the conditions are true the logical "AND" operator returns the boolean value true or 1, else it returns false or 0.

Syntax: (condition 1 && condition 2)

* Truth Table for "AND".

A	B	A && B
1	1	1
1	0	0
0	1	0
0	0	0

→ **Logical "OR" Operator:**

In logical 'OR' operator is ~~dep~~ denoted by double pipe character (||) it is used to check the combinations of more than one

condition. It is a binary operator which requires 2 operants. If any of the operant value is non-zero (true), logical "OR" operator returns 1 otherwise it returns zero (0). if all operant values are 0 (false).

Syntax :- (condition 1 || condition 2)

* Truth Table for "OR".

A	B	A || B
1	1	1
1	0	1
0	1	1
0	0	0

→ Logical "NOT" Operator:

The logical "NOT" operator is represented by (!) symbol. which is used to reverse the result of any given expression or condition. If the result of an expression is non-zero or true, then result will be reversed as zero or false value. Similarly if the condition result is false or zero, the 'NOT' operator reverses the result & returns 1 or true.

Example:- The user enter the non-zero value is 5 the logical 'NOT' operator returns the zero or false boolean value. & if the user enters 0 value the operator returns the true boolean value or 1.

* Truth Table for 'NOT'

Condition	! Condition
0	1
1	0

↳ Bitwise Operator :
It cannot be directly applied to primitive datatypes such as integer, float, double etc. always remember one thing that bitwise operator are mostly used with the integer datatypes because of its compatibility.

The bitwise logical operator work on the data bit-by-bit.
Starting from the least significant bit (LSB) which is the right most bit. Working towards the most significant bit (MSB) which is the left most bit.

The result of the computation of Bitwise logical operator is shown in the table given below:-

x	y	x & y	x | y	x ^ y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

→ **Bitwise "AND" operator:**
Bitwise "AND" operator is denoted by (&) single ampersand sign. 2 integer operands are written on both sides of the (&) operator. If the corresponding bits of both the operands are 1, then output is 1, otherwise 0.

→ **Bitwise "OR" operator:**
It is represented by single pipe symbol (|). If the bit value of any operand is 1 then output is 1, otherwise 0.

→ **Bitwise "XOR" operator:**
It is represented by (^) symbol. If the corresponding bit of any operand is 1 then output is 1, otherwise 0.

Assignment Operator:

They are used to assign value to a variable. The left side operand of the assignment operator is a variable & the right side operand of the assignment operator is a value. The value on the right side must be of the same datatype as the variable on the left side, otherwise the compiler will raise an error.

↳ Different types of assignment operator are shown as:-

Example:-
```
a = 10;
b = 20;
ch = 'y';
```

**NOTE:- '=' this is the simplest assignment operator because this operator is used to assign the value on the right to the variable on the left.

'+=' this operator is combination of + & = operator. This operator first adds the current value of the variable on left to the value on the right & then assign the result to the variable on the left.

Ternary Operator:

The ternary operator is a condition operator in C. It takes three operand in this the symbol (?) is used as ternary value.

Syntax:- <expression> ? <value1> : <value2>

↓

Relational

Example:- c = a<b>?<a>:<b>;

here, c will be assign the value of a if a is less than b otherwise it will be assign the value of b.

⇒ Character Input / Output Functions:

1. getch() function:

getch() function is used to read a character from the keyboard. It does not expect the enter key press. It has the following syntax:-

```
ch = getch()        // where ch is
                    a char variable.
```

Example:

```
char ch;
ch = getch();
```

2. getchar():
getchar() function is used to read one character at a time from the standard input device such as keyboard.
It has the following syntax:
ch = getchar() // where ch is a char variable

3. putchar() function:
putchar() function is used to display one character at a time on the monitor screen.
It has the following syntax:
putchar(ch);

Example:
```
char 'M'   char c = 'M';
           putchar(c);
```

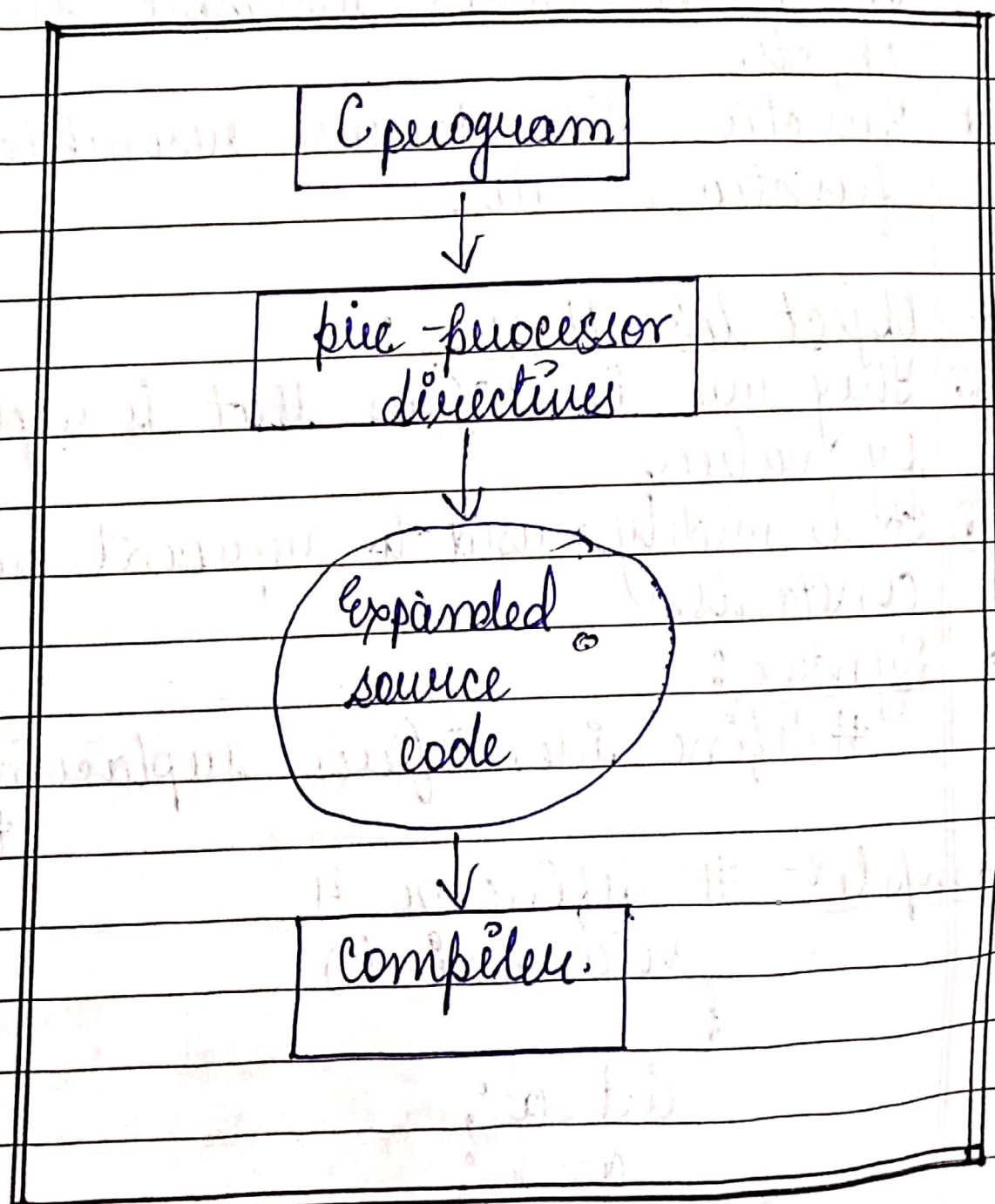
Pre-processors & Directives in C.

→ A pre-processor is a program that processes our source program before it is passed to the compiler.
→ The pre-processor works on the source code & creates an expanded source code.

→ The pre-processor offers several features called pre-processor directives

→ Pre-processor directives began with as (#) symbol.

→ The directives can be placed anywhere in a program but most often placed at the beginning of the program before the function definition.



→ The various pre-processor directives are as follows:

1) Macros Expansion: (# define)
A Macros is a fragment of code which has given a name in C. The name is used, it is replaced by the contents of the macros.
There are 2 kinds of Macros:-
* Object like Macros resemble data objects.
* Function like Macros resemble function calls.

* Object like Macros:
① They are identifiers that is replaced by values.
② It is widely used to represent numeric constants.
③ Syntax:
#define identifiers replacement text.

Example:- # define x 4
void main()
{
int a;
a = x;
}
where x = identifiers
4 = replacement text

```
#define pi 3.14
void main()
{
float r = 6.39, area;
area = pi * r * r;
printf("the area is = %f", area);
}
```

* Function like Macros:

① It looks like a function call & can be defined with #define directives but with a pair of braces or parenthesis ().

Example :- #define area(x) pi * x * x;

⇒ Macros like arguments:-

```
#define <stdio.h>
#define Min(a, b)((a) < (b) ? (a) : (b))
void main()
{
printf("Minimum between 10 & 20 = %d
, Min(10, 20));
}
```

#2) File Inclusion:

For file inclusion we can use the #include.

Syntax:

#include < Type file name >

- We can replace the content that is comprised in the file name at the point where a certain directive is returned written.
- We can use a file inclusion directive & include the header file in the programs.
- We can integrate function declaration, macros & declations of the external variables in the top header files rather than repeating each logic in the program.
- The <stdio.h> header file includes the function declarations & can provide the details about the input & output.

→ Examples of the file inclusion statements:

→ #include "Define your file name" -

* In this example we can search a file in the current working directory through easy search.

→ # include < Define file name > -
* in this example we can define a certain directory & we search a file within it.

Conditional compilation :
We can use conditional compilation on a certain logic where we need to define a condition logics. It utilizes directives like :- # if ; # else ; # ~~else~~if end if.

Syntax : # if test your condition
{
statement 1;
statement 2;
}
# else
{
statement 3;
statement 4;
}
# end if.

↳ # if :
The # if preprocessor directive evaluates the expression & condition. *

If condition is true it executes the code.

Syntax: # if test your condition
{
statement 1;
statement 2;
}
# end if

↳ # else:

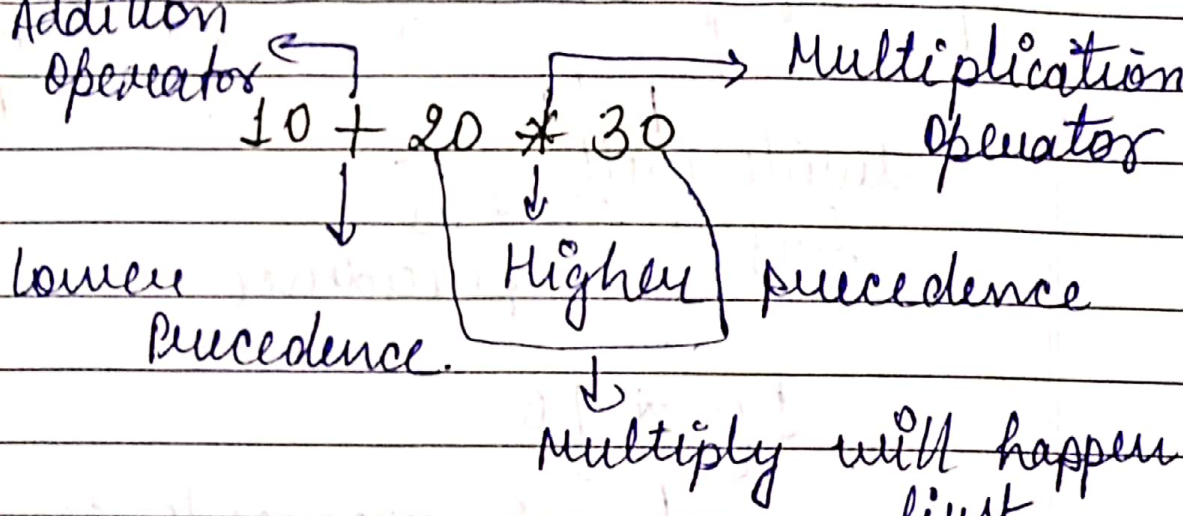
The # else pre-processor directives evaluates the expression or condition. if condition of # if is false it can be used # if, # else, # else if directives.

Syntax: # if test your condition
{
statement 1;
statement 2;
}
# else if
{
statement 3;
statement 4;
}
# end if.

Operator Precedence and Associativity in C

Operator precedence determine which operation is perform first in an expression with more than one operators with different precedence.

For Example :- 10 + 20 * 30
= 610 Ans.



10 + 20 * 30 is calculated as 10 + (20 * 30).

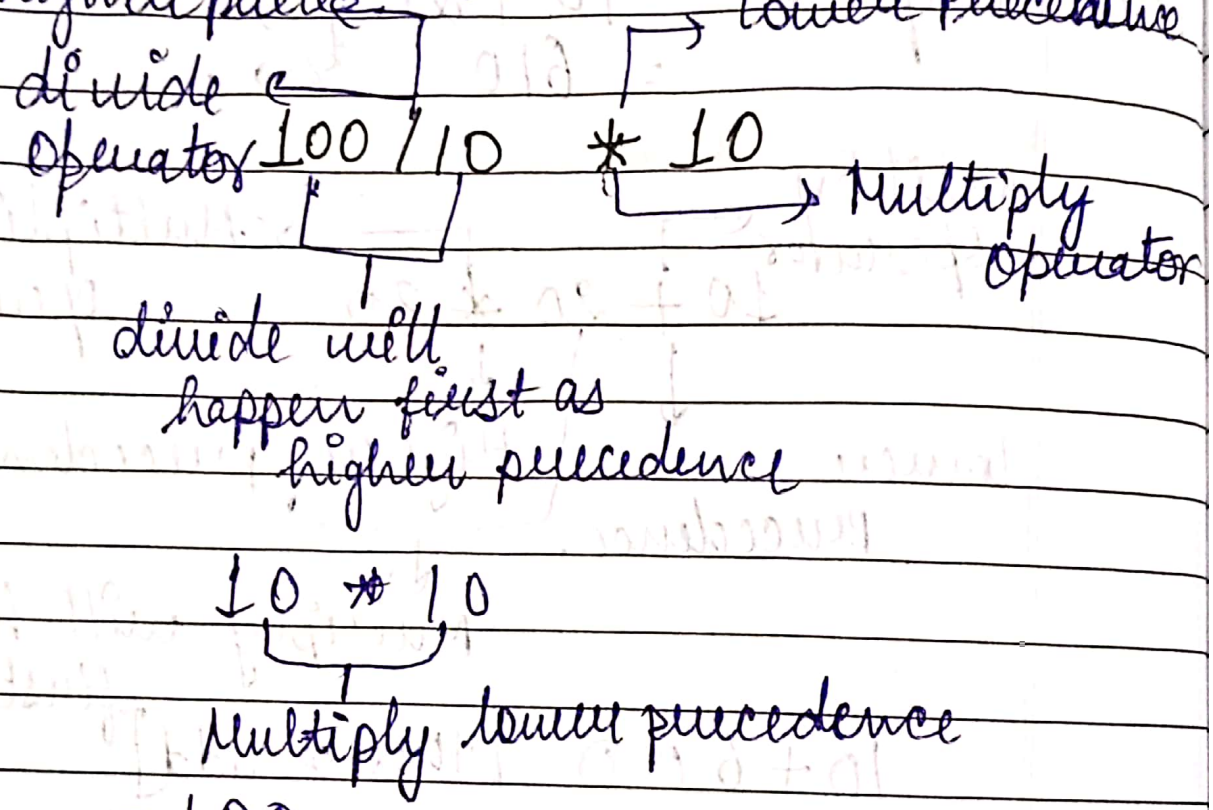
↳ Operator associativity

It is used when two operators of same precedence appear in an expression. Associativity can be

either left to right or right to left.

For example: * and / have same precedence & there associativity left to right so the expression "100/10*10" = 100

= 100



NOTE: Divide & Multiply both have the same precedence but left to right associativity

Ques. What would be the output / result obtained from this programme.?

```
int main()
{
 int x=5, y=10;
 int z;
 z = x+2 * y;
 printf("%d", z);
}
```

(i) 0
(ii) 5
(iii) 25 ✓ output
(iv) 20

Ques2. What would be the output/result obtained from this programme?

```
int main()
{
 int x=50, y=5;
 int z;
 z = x/2 * y;
 printf("%d", z);
}
```

⇒ output = 125.