UNIT 2

CHAPTER 1

OPERATORS AND EXPRESSIONS

C supports a rich set of operators. An operator is a symbol that tells the computer to perform an arithmetic or logical function. Operators are used in programs to manipulate data and variables. Operators in C can be classified into a number of categories. They are

- ✓ ArithmeticOperators
- ✓ RelationalOperators
- ✓ LogicalOperators
- ✓ AssignmentOperators
- ✓ Increment and DecrementOperators
- ✓ ConditionalOperators
- ✓ BitwiseOperator
- ✓ SpecialOperators

q)List and explain arithmetic and relational operators,assignment operators in C with example.(8m)

3.1 ArithmeticOperators:

All the basic arithmetic operators are supported by C. These operators can manipulate with any built in data types supported by C. The various operators are tabulated as follows:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

3.1.1 Integer Arithmetic:

When both the operands in the expression are integers, then the operation is known as integer arithmetic.
For example :if the values of a=14 and b=4 then
a+b=18 ,
a-b=10,
a*b=56 ,

a/b=3,

During modulo division, the sign of the result is the sign of the first operand.
E.g. -14%3=-2 ,14%-3=2

3.1.2 Real Arithmetic:

An arithmetic operation involving only real operands is called real arithmetic. A real value can assume value either in decimal or exponential notation. % operator cannot be used with realoperands
E.g. If a=2.4 and b=1.2 then
a+b=3.6 ,
a-b=1.2,
a*b=2.88a/b=2.0

Mixed mode Arithmetic:

When one of the operands is real and the other is an integer, the expression is called mixed mode arithmetic expression. The result is always a real number
E.g. : 15/10.0=1.5 ,
15/10=1.5

3.2 Relational Operators:

Comparisons can be done using relational operators. A relational expression contains a combination of arithmetic expressions, variables or constants along with relational operators. A relational expression can contain only two values i.e. true or false. When the expression is evaluated as true then the compiler assigns a non zero(1) value and 0 otherwise. These expressions are used in decision statements to decide the course of action of a running program.
Syntax: ae1 relational operator ae2
where ae1 and ae2 are arithmetic expressions

Operators	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
y>=	Greater than or equal to
==	Equal to
!=	Not equal to

Example:4.5<= 10 true
a+b == c+d

10< 7+5

-35 >= 0

Logical Operators:

An expression that combines two or more relational expressions is called a logical expression and the operators used to combine them are called logical operators.

Syntax: R1 operator R2

where R1 and R2 are relational expressions

Operator	Meaning
&&	Logical And
||	Logical Or
!	Logical Not

Example :

if(age > 55 && salary < 1000)

if(number < 0 || number > 100)

Relational Expression R1	Relational Expression R2	R1&&R2	R1||R2
0	0	0	0
0	Non zero	0	Non zero
Non zero	0	0	Non zero
Non zero	Non zero	Non zero	Non zero

Assignment Operators:

Assignment operators are used to assign the result of an expression to variable. The operator used for assignment in C is '='. C also supports short hand assignment operators like

v op = expression where v- variable and op- operator. Here op= is known as a short had assignment operator.

v op=expression is equivalent to **v=v op expression**

E.g. The statement x+=y+1; is equivalent to x=x+y+1;

Statement with simple Assignment operator

a=a+1 a+=a

a=a-1 a-=a

a=a*(n+1) a*=(n+1)

The advantage of short hand assignment operator is

a) Easy to read and write
b) Statements are more concise and efficient

3.3 Increment/Decrement Operators:

C has 2 very powerful operators that are not found in any other language. They are increment/decrement operators i.e. ++ and --. The ++ operator is used to increment value of a variable by 1. The -- operator is used to decrement value of a variable by 1.

Both are unary operators and can be written as follows: ++m, m++ , m—and –m. Both m++ and ++m mean the same thing when they form statements independently. But, they behave differently when used in expression on the right hand side of assignment operator.

E.g. Let a=5; x=a++; Here, this statement can be broken into 3 statements as follows a=5; x=a; a=a+1;

In the above example, the value of a is assigned to x and then its value is incremented by 1**. A prefix operator first adds 1 to the operand and then the result is assigned to the variable on the left. A postfix operator first assigns its value to the variable and then increments its value by 1.**

E.g. m=10;
y=--m

Here the value of m is decremented by 1 and then assigned to y. Hence the value of y is 9.

Example:Y= ++m In this case value of y and m would be 6
Y= m++ In this case vale of y would be 5 and m would be 6.

Example:#include<stdio.h>

```
#include<conio.h>
void main()
    {
intn,k,m;
clrscr();
     k=1,m=2;
      n= k++ + m;
printf("%d",k);
printf("%d",n);
getch();

}
```

Output:k=2, n=3

q)Differnentiate i++ and ++I with example

They are unary operators

- When postfix++(or--)is used with a variable in an expression ,the

expression is evaluated using the original value of the variable and then the variable is increment.ex m=a++; m=a--

- When prefix++(or--)is used in a expression, the variable is incremented and then the expression is evaluated using the new value of the variable.ex:m=++a m=--a

q)How does ternary operator works?give example(2).

3.4 Conditional Operator:

Conditional operators are also called ternary operators. Conditional expressions can be constructed in C using the operator pair '?:'.

Syntax: expr?expr1:expr2;

Here expr is evaluated first. If the value is true, then expr1 is evaluated and becomes the value of the expression. If the expression is false then, expr2 is evaluated and becomes the value of the expression.

E.g. a=5;b=10;

x=(a>b)?a:b;

The output of the above example is as follows- The value of b i.e. 10 is assigned to x.

Bitwise Operators:

These operators are used to manipulate data at bit level. These operators are used for testing the bits or shifting the bits either to the left or right.

Operator	Meaning
&	Bitwise And
|	Bitwise Or
^	Bitwise exclusive Or
<<	Shift left
>>	Shift Right
~	One's complement

Special Operators:

C supports other special operators like pointer operator (&,*) and member selection operator(.). Comma is also an operator use to link related expressions together. The sizeof operator is a compile time operator. It returns the number of bytes occupied by the operand depending on the data type. The operand could be a variable, constant or a data type qualifier.

q)explain comma operator

Comma Operator :Comma Operator can be used to link the related expression together.expressions are evaluated *to left to right* and the value *of right-most* expression is the value of the combined expression.

Ex:Value = (x=10, y=5, x+y); value=15

```
#include<stdio.h>
void main()
{
inta,b,c;
c=(a=15,b=10);
printf("The value of c is = %d",c);
getch();
}
Output
The value of c is = 10
```

Sizeof operator is a compile time operator. It returns number of bytes occupied by the operand. The operand may be a variable, constant or a data type.

Ex:m=sizeof(sum);
n=sizeof(long int)

```
Example: #include<stdio.h>
void main()
{
float x;
printf("The size of variable x is %d bytes",sizeof(x));
getch();
}
Output
The size of variable x is 4 bytes
```

Arithmetic Expressions:

An arithmetic expression is a combination of variables, constants and operators arranged as per the syntax of the language. Expressions are evaluated using assignment statements of the form

variable=expression;

Some of the examples of C expressions are shown below.

Alygebraic expression	**C expression**
• ab-c	a*b-c

- (m+n) (x+y) (m+n)*(x+y)
- Ab/c a*b/c
- $3x^2$ +2x+1 3 * x * x +2 * x+ 1

q)Explain evaluation of expression and precedence of arithmetic operators with examples

Expressions are always evaluated from left to right, using the rules of precedence of operators, if parenthesis are missing

High Precedence- * / %

Low Precedence - + -

Basically the evaluation procedure involves 2 left to right passes. During the first pass the high priority operators are applied as they are encountered and during the second pass, lower priority operators are applied as they are encountered.

Example: Consider the following expression

x= a-b/3+c*2-1 If the values of a=9, b=12,c=3

Now x= 9-12/3+3*2-1

Pass 1: x=9-4+6-1

Pass 2: x=5+6-1=10

Example 2:9-12/(3+3)(2-1)*

Whenever parentheses are used, the expressions within parentheses assume highest priority. If two or more sets of parentheses appear one after another as shown above, the expression contained in the left-most set is evaluated first and the right-most in the last. Given below are the new steps.

First pass

Step 1: 9-12/6 * (2-1)

Step 2 : 9-12/6 * 1

Second pass

Step 3 : 9-2* 1

Step 4 : 9-2

Third pass

Step 5 :7

q)Explain precedence and associativity of arithmetic operators in C

Precedence of operators:

If more than one operators are involved in an expression then, C language has predefined rule of priority of operators. This rule of priority of operators is called operator precedence.

In C, precedence of arithmetic operators(*,%,/,+,-) is higher than relational operators(==,!=,>,<,>=,<=) and precedence of relational operator is higher than logical operators(&&, || and !).

Suppose an expression:

(a>b+c&&d)

This expression is equivalent to ((a>(b+c))&&d)

i.e. (b+c) executes first then,

(a>(b+c)) executesthen, (a> (b=c)) && d) executes

Associativity of operators

Associativity indicates in which order two operators of same precedence(priority) executes. Let us suppose an expression:

a==b!=c

Here, operators == and != have same precedence. The associativity of both == and != is left to right, i.e, the expression in left is executed first and execution take place towards right. Thus, a==b!=c equivalent to :(a==b)!=c

The table below shows all the operators in C with precedence and associativity.

Note: Precedence of operators decreases from top to bottom in the given table.

Summary of C operators with precedence and associativity

Operator	**Meaning of operator**	**Associativity**
() [] -> .	Functional call Array element reference Indirect member selection Direct member selection	Left to right
! ~ + - ++ -- & * Sizeof (type)	Logical negation Bitwise(1 's) complement Unary plus Unary minus Increment Decrement Dereference Operator(Address) Pointer reference Returns the size of an object Type cast(conversion)	Right to left
* /	Multiply Divide	Left to right

%	Remainder	
+ -	Binary plus(Addition) Binary minus(subtraction)	Left to right
<< >>	Left shift Right shift	Left to right
< <= > >=	Less than Less than or equal Greater than Greater than or equal	Left to right
== !=	Equal to Not equal to	Left to right
&	Bitwise AND	Left to right
^	Bitwise exclusive OR	Left to right
|	Bitwise OR	Left to right
&&	Logical AND	Left to right
||	Logical OR	Left to right
?:	Conditional Operator	Left to right
=	Simple assignment	Right to left

q)What are the rules for evaluation of expressions?

1. First parenthesized sub expression from left to right are evaluated.
2. If parentheses are nested, the evaluation begins with the innermost sub expression
3. The precedence rule is applied in determining the order of application of operators in evaluating sub expressions
4. The associatively rule is applied when 2 or more operators of the same precedence level appear in a sub expression.
5. Arithmetic expressions are evaluated from left to right using the rules of precedence
6. When parentheses are used, the expressions within parentheses assume highest priority

Evaluate x=--c/2+b*8/b-b+a++/3

When a=4,b=3,c=25

X=--25/2+3*8/3-3+4++ /3

=--25/2+3*8/3-3+4/3

=24/2+3*8/3-3+4/3

=12+3*8/3-3+4/3

=12+24/3-3+4/3

=12+8-3+4/3

=12+8-3+1

=20-3+1

=17+1=18

Q)Define type conversion?explain different type ?

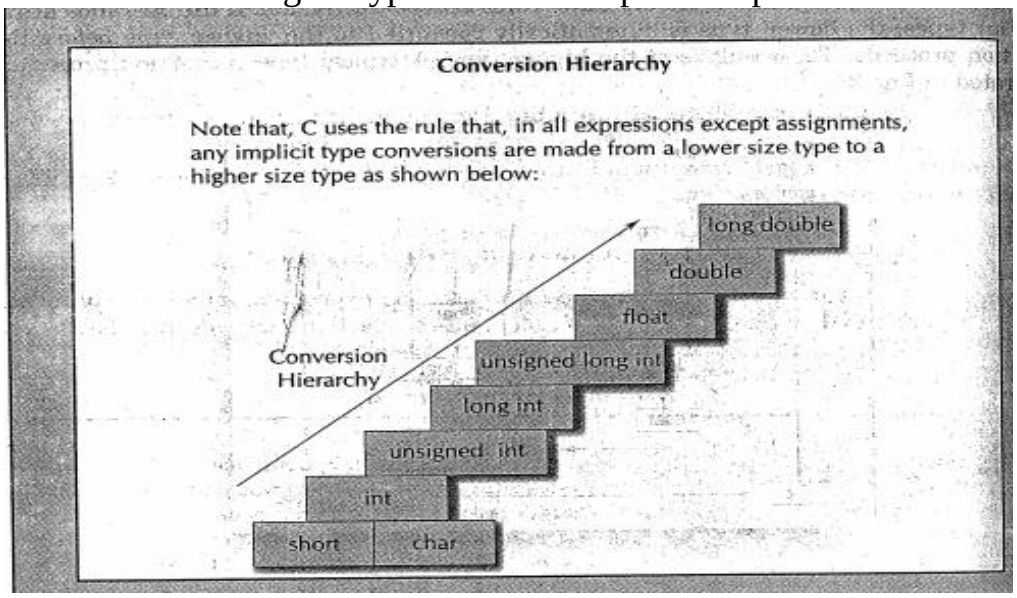
Type casting is a way to convert a variable from one data **type** to another data **type**.

There are two types of the type conversions:

- Implicit Type Conversion
- Explicit Type Conversion

Implicit Type Conversion :

- When data value automatically convert from one type to another type is called the Implicit type conversion.
- If the operands are of different types, the lower type is automatically converted to the higher type before the operation proceeds.



1.float to int causes truncation of the fractional part.
2.double to float cause rouding of digits.
3.longint to int cause dropping of the excess higher order bits.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
float c;
a= 12,b=13;
c= a+b;    //automatically conversion   done
printf("%f",c);
getch();
```

}

q) what is explicit type conversion? Give example.

Explicit Type Conversion:

When a process of the conversion done manually of locally then this process is called the explicit type conversion process or casting operation.

Syntax :(type_name) expression

Ex:

Example	*Action*
x =(int)7.5	7.5 is converted to integer bytruncation.
a =(int)21.3/(int)4.5	Evaluated as 21/4 and the result would be5.
b=(double)sum/n	Division is done in floating pointmode.
y =(int)(a+b)	The result of a+b is converted tointeger
z=(int)a+b	a is converted to integer and then added tob.

example:

```
#include<stdio.h>
 #include<conio.h>
void main()
{
floata,b;
int c;
clrscr();
 a=12.3;
b=13.3;
c= (int) (a+b);
printf("%f",c);
getch();
}
/* Output 25 */
```

Note : whenever you try to convert higher data type to lower type your result will be loss.

Built-in Mathematical functions:

Mathematical functions such as cos, tan and sqrt are widely used in problem solving. All these functions are available in the <math.h> header file. It is necessary to include this header file in the program in order to use these functions. The list of mathematical functions available is given below:

Function	Meaning
ceil(x)	x rounded up to the nearest integer
exp(x)	e to the power of x
fabs(x)	Absolute value of x

floor(x)	x rounded down to the nearest integer
log(x)	Natural log of x, x>0
pow(x,y)	x raised to y (x^y)
sqrt(x)	Square root of x ; x>=0

Chapter 2
Input and Output Statements in C

Input means to provide the program with some data to be used in the program and **Output** means to display data on screen or write the data to a printer or a file.

C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.

All these built-in functions are present in C header files;

The getchar() and putchar() Functions [Unformatted I/O]

The **getchar()** function reads the character from the keyboard. This function reads only single character at a time. To continuously read the characters use getchar() within looping statement.

Syntax : c=getchar() where c is character variable.

The **putchar(int c)** function printsthe given character on the screen . This function puts only single character at a time. You can use this method in the loop in case you want to display more than one character.

```
#include <stdio.h>
main( )
{
  int c;
printf( "Enter a value :");
  c = getchar( );
printf( "\nYou entered: ");
putchar(c);
}
```

When the above code is compiled and executed, it waits for you to input some text. When you enter a text and press enter, then the program proceeds and reads only a single character and displays it

Formatted input/output - scanf() and printf() functions

The standard input-output header file, named stdio.h contains the definition of the functions printf() and scanf()

scanf() function

Syntax :

scanf("format string", arg1,arg2…argn);

- This function is usually used as an input statement.

- The format string must be a text enclosed in double quotes. It contains type of data to input. Example: integer (%d) , float (%f) , character (%c) or string (%s).
- The arg1,arg2..argn contains a list of variables each preceded by & (to get address of variable) and separated by comma.

Commonly used format specifiers are as follows

%c character	%f floating point
%d integer	%lf double floating point
%i integer	%e exponential notation
%u unsigned integer	%s string
%ld signed long	%x hexadecimal
%lu unsigned long	%o octal

Examples :-

```
int i, d ;
char c ;
float f ;
scanf( "%d", &i ) ;  /* input integer data*/
scanf( "%d %c %f", &d, &c, &f ) ; /* input int , char and float */
```

The & character is the *address of* operandin C, it returns the address in memory of the variable it acts on.

printf() function

Syntax :
printf("format string", v1,v2…vn);

The printf() function is used for formatted output and uses a format string which is made up of a series of format specifiers to govern how it prints out the values of the variables or constants required. The more common format specifiers are given below

%c character	%f floating point
%d integer	%lf double floating point
%i integer	%e exponential notation
%u unsigned integer	%s string
%ld signed long	%x hexadecimal
%lu unsigned long	%o octal

```
#include<stdio.h>
void main()
{
  int i;
printf("Please enter a value...");
scanf("%d", &i);
```

```
printf( "\nYou entered: %d", i);
}
```
When you will compile the above code, it will ask you to enter a value. When you will enter the value, it will display the value you have entered on screen.

Some further examples :-

```
int i = 10, j = 20 ;
char ch = 'a' ;
double f = 23421.2345 ;
printf( "%d + %d", i, j ) ;  /* values are substituted from the variable list  in order as required */
printf( "%c", ch ) ;
printf( "%s", "Hello World\n" ) ;
printf( "The value of f is : %lf", f ) ;          /*Output as : 23421.2345 */
printf( "f in exponential form : %e", f ) ;          /* Output as : 2.34212345e+4*/
```

Field Width Specifiers

Field width specifiers are used in the control string to format the numbers or characters output appropriately .

Syntax :- **%[total width][.decimal places]format specifiers**

where square braces indicate optional arguments.

For Example:-

```
int i = 15 ;
float f = 13.3576 ;
printf( "%3d", i ) ;   /* prints "_15 " where _ indicates a space  character */
printf( "%6.2f", f ) ; /* prints "_13.36" which has a total width of 6 and displays 2 decimal places  */
```

gets() & puts() functions

gets() functions reads a line of string and store it in a string variable.

Syntax : gets(str) where str is string variable

puts() functions prints string .

Syntax :puts(str) where str is string variable

Example:

```
#include<stdio.h>
void main()
{
   /* character array of length 100 */
   char str[100];
printf("Enter a string");
gets( str );
puts( str );
getch();
```

}

Difference between scanf() and gets()

The main difference between these two functions is that scanf() stops reading characters when it encounters a space, but gets() reads space as character too.

If you enter name as **Hello World** using scanf() it will only read and store **Hello** and will leave the part after space. But gets() function will read it completely.

CHAPTER 3
Conditional and Loop Control Structures

C conditional statements allow you to make a decision, based upon the result of a condition. These statements are called as **Decision Making Statements** or **Conditional Statements**.

If statement(s)

If statements in C is used to control the program flow based on some condition, it's used to execute some statement code block if the expression is evaluated to true. Otherwise, it will get skipped. This is the simplest way to modify the control flow of the program.

There are four different types of if statement in C. These are:

- Simple if Statement
- if-else Statement
- Nested if-else Statement
- else-if Ladder

Simple if Statement

The basic format of if statement is:

```
if(test_expression)
{
   statement 1;
   statement 2;
...
}
```

Here if the test expression is evaluated to true, the statement block will get executed, or it will get skipped.

Example :

```
#include<stdio.h>

main()
{
  int a = 15, b = 20;

  if (b > a) {
printf("b is greater");
  }
}
```

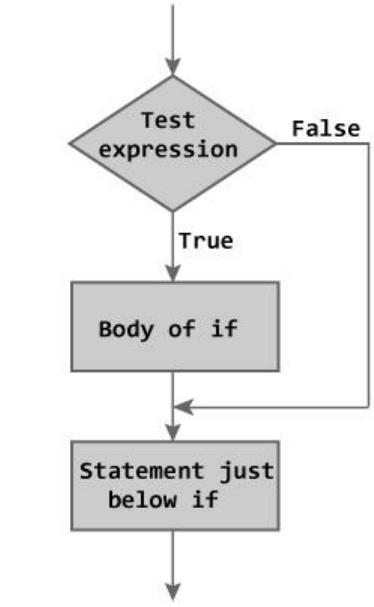


Figure: Flowchart of if Statement

If ..else Statement

The syntax of an **if...else** statement in C programming language is −

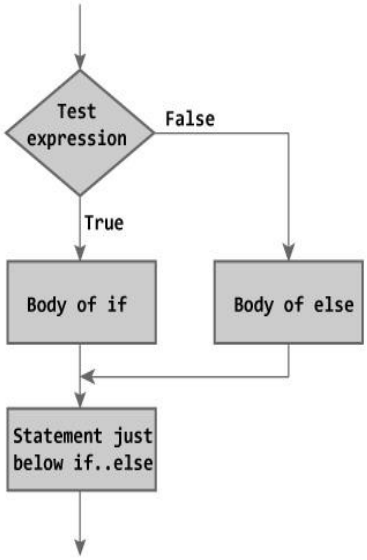


Figure: Flowchart of if...else Statement

```
if(test expression)
{
  Statement block-1
}
 else
{
  Statement block-2
}
```

If the test expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed.

Example:

```
#include<stdio.h>

main( )
  {
   int a;
printf("n Enter a number:");
scanf("%d", &a);
   if(a>0)
printf( "n The number %d is positive.",a);
   else
printf("n The number %d is negative.",a);
  }
```

nested-if Statement

A nested if is an if statement that is inside another if statement .

Syntax:

```
if (condition1)
{
/* Executes when condition1 is true*/
  if (condition2)
  {
/* Executes when condition2 is true*/
  }
}
```

Example

```
#include <stdio.h>
int main()
{
  int var1, var2;
printf("Input the value of var1:");
scanf("%d", &var1);
printf("Input the value of var2:");
```

```
scanf("%d",&var2);
  if (var1 != var2)
  {
        printf("var1 is not equal to var2\n");
        /*Nested if else*/
        if (var1 > var2)
        {
              printf("var1 is greater than var2\n");
        }
        else
        {
              printf("var2 is greater than var1\n");
        }
  }
  else
  {
        printf("var1 is equal to var2\n");
  }
  return 0;
}
```

if-else-if ladder

Here, a user can decide among multiple options. The 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 ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:

```
if (condition-1)
{
   Statement block-1;
}
else if (condition-2)
{
   Statement block-2;
}
.
.
Else
{
Default block of  statement(s);
}
```

The if else ladder statement in C is used to test set of conditions in sequence. An if condition is tested only when all previous if conditions in if-else ladder is false. If any of the conditional expression evaluates to true, then it will execute the corresponding code block and exits whole if-else ladder.

- First of all condition1 is tested and if it is true then statement block-1 will be executed and control comes out of whole if else ladder.

- If condition1 is false then only condition2 is tested. Control will keep on flowing downward, If none of the conditional expression is true.

- The last else is the default block of code which will gets executed if none of the conditional expression is true.

Example

```
#include <stdio.h>
int main()
{
   int marks;
   printf("Enter your marks?");
   scanf("%d",&marks);
   if(marks > 85 && marks <= 100)
   {
      printf("Congrats ! you scored grade A ...");
   }
   else if (marks > 60 && marks <= 85)
   {
      printf("You scored grade B + ...");
   }
   else if (marks > 40 && marks <= 60)
   {
      printf("You scored grade B ...");
   }
   else if (marks > 30 && marks <= 40)
   {
      printf("You scored grade C ...");
   }
   else
   {
      printf("Sorry you are fail ...");
   }
}
```

Switch Statement

Syntax :

```
switch (variable or integer expression)
{
   case constant1:
    Statement(s) ;
break;

   case constant2:
         Statement(s) ;
 break;
       .
       .
       .
   default:
```

default statement(s);
}

Switch statement is a control statement that allows us to choose only one choice among the many given choices. The expression in `switch` evaluates to return an integral value, which is then compared to the values present in different cases. It executes that block of code which matches the case value. If there is no match, then **default** block is executed(if present).

Rules for using switch statement

1. The expression (after switch keyword) must yield an **integer** value i.e the expression should be an integer or a variable or an expression that evaluates to an integer.
2. The case **label** values must be unique.
3. The case label must end with a colon(:)
4. The next line, after the **case** statement, can be any valid C statement.

Difference between switch and if

- if statements can evaluate float conditions. switch statements cannot evaluate float conditions.
- if statement can evaluate relational operators. switch statement cannot evaluate relational operators i.e they are not allowed in switch statement.

Example :

```
#include <stdio.h>
main()
{
  int x = 2;
  switch (x)
  {
     case 1:
printf("Choice is 1");
break;
     case 2:
printf("Choice is 2");
 break;
     case 3: printf("Choice is 3");
           break;
     default: printf("Choice other than 1, 2 and 3");
            break;
  }
  return 0;
}
```

Goto Statement

The goto statement is a jump statement which is sometimes also referred to as unconditional jump statement. The goto statement can be used to jump to specified label within a function.

Syntax:

goto label;

….

….

label: statement(s);

example:

```
#include<stdio.h>
main()
{
int sum=0;
for(int i = 0; i<=10; i++){
            sum = sum+i;
            if(i==5){
            goto addition;
            }
  }

  addition:
printf("%d", sum);

}
```

Multiple choice questions

1. Among unary operation which operator represents increment?

A) --

B) ++

C) –

D) +

2.When one of the operands is real and the other is an integer, the expression is called _____ arithmetic expression.

A) mixed mode

B) real mode

C) integer mode

D) expression mode

3.Comparisons can be done using_____.

A) Arithmetic operator

B) Logical operators

C) Conditional operator

D) relational operators

4._____are used to assign the result of an expression to variable

A) Assignment operators
B) Conditional operator
C) Arithmetic operator
D) None of the above

5. Bitwise operators are used to manipulate data at _____

A) byte level
B) bit level
C) number level
D) bitwise level

6. The function scanf is used to ___

A) To take logical decisions
B) Input a set of values
C) Print a set of values
D) Do mathematical manipulations

7. The _____ function reads the character from the keyboard.

A)getchar()
B) putchar()
C) scan()
D) none of the above

8. The _____ function is used for formatted output.

A) prints()
B) put()
C) printf()
D) scanf()

9. Operator % in C Language is called.?

A) Percentage Operator
B) Quotient Operator
C) Modulus
D) Division

10. Choose a right statement.

int a = 10 + 4.867;
A) a = 10
B) a = 14.867
C) a = 14
D) compiler error.

11. What is the priority of operators *, / and % in C language.?

A) * > / > %
B) % > * > /
C) Both % = / , * are same
D) All three operators *, / and % are same.

12. Choose a C Conditional Operator from the list.
A) ?:
B) :?
C) :<
D) <:

13. What is the other name for C Language ?: Question Mark Colon Operator.?
A) Comparison Operator
B) If-Else Operator
C) Binary Operator
D) Ternary Operator

14. Choose a syntax for C Ternary Operator from the list.
A) condition ? expression1 : expression2
B) condition : expression1 ? expression2
C) condition ? expression1 < expression2
D) condition < expression1 ? expression2

15. which of the following operators takes only integer operands?
A) +
B) *
C) /
D) %

16. What is the output of the C statement.?

```
int main()
{
   int a=0;
   a = 5<2 ?4 : 3;
printf("%d",a);
   return 0;
}
```

A) 4
B) 3
C) 5
D) 2

17. If you have to make decision based on multiple choices, which of the following is best suited?
a) if
b) if-else
c) if-else-if
d) All of the above

18. Which of the following cannot be checked in a switch - case statement?
A) Character
B) Integer
C) Float

D) enum

19. Which of the following is branching statement of C language?
A) if statement
B) if…else statement
C) switch statement
D) All of these

20. If the Boolean expression of if statement evaluates to _____, then the block of code inside the if statement will be executed.
A. True
B. False
C. True and False
D. None of these

21. What will be the output of the following C code?

```
#include <stdio.h>
   void main()
   {
      int x = 5;
      if (x < 1)
printf("hello");
      if (x == 5)
printf("hi");
      else
printf("no");
   }
```

A)hi
b) hello
c) no
d) error

22. What is the output of C Program with switch statement.?

```
int main()
{
   int a=3;

   switch(a)
   {
      case 2: printf("ZERO "); break;
      case default: printf("RABBIT ");
   }
   }
```

A) RABBIT
B) ZERO RABBIT
C) No output
D) Compiler error

23. The _____statement terminates the loop immediately when it is encountered.
A) break

B) goto
C) continue
D) none

24. The _____statement skips some statements inside the loop and goes back to beginning of loop for executing next iteration.
A. if
B. continue
C. goto
D. break

25. A loop becomes an infinite loop if a condition never becomes_____
A. false
B. true
C. either true or false
D. none of the above

Descriptive questions

1) List and explain arithmetic, logical and relational operators available in C
2) List and explain a) conditional operator, b) increment & decrement operators available in C
3) Write a note on bitwise operators available in C
4) Explain with example evaluation of arithmetic expression in C
5) Explain precedence of arithmetic operators with the help of an example expression
6) Explain the concept of type conversion in C, Give examples
7) Explain in brief implicit and explicit type conversion with example
8)Explain with example formatted Input in C
9) Explain with example formatted Output in C
10) Explain unformatted input/ output in C.
11) Explain simple if statement and else..if with syntax and example.
12) Explain nested if statement with the help of a program example
13) Explain else..if ladder with the help of a program example
14) Explain switch statement with syntax and a program example