Simplified

SARVA EDUCATION

e- Book

C & C++ LANGUAGE

SARVA EDUCATION ^{SM} *- An I.T & Skill Advancement Training Programme, Initiated by* ***SITED***$^{®}$***-India***

An ISO 9001:2015 Certified Organization

UNITS AT A GLANCE

Unit-I

C LANGUAGE

CHAPTER-1

INTRODUCTORY CONCEPTS

Computer Languages

The functioning of a computer is controlled by a set of instructions (called a computer program). These instructions are written to tell the computer.

What operation to perform? Where to locate data?
How to present result? When to make certain decisions?

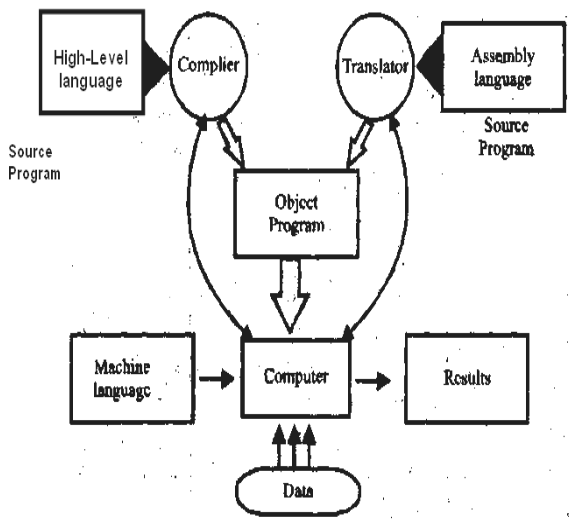
The communication always needs a common language or terminology. The language used in the communication of computer instructions is known as the programming language into this language. The computer has its own language and any communication with the computer must be in its language or translated into this language.
A language is a system of communication. We communicate to one another our ideas and emotions by means of language. Similarly, a computer language is a –mean of Communication, which is used to communicate between people and the (Computer). Using some computer language a programmer can still the computer what he wants to do. Computer languages divide in three parts**:-**

1. **Machine languages (low level languages)**
2. **Assembly (or symbolic) languages, and**
3. **High level languages**

1. Machine Language
Computers are made of two-state electronic components which can understand only pulse and no-pulse (or '1' and '0') conditions. Therefore, al instructions 'and data should be written using binary -codes 1 and 0. The binary code is called the machine code or machine language
Computer does not understand English, Hindi or Tamil. They respond only to machine language. Added to this, computers are not identical in design. Therefore, each-computer has its own machine language. (However, the script, 1 and 0, is the same for all computers.) This poses two problems for the user. **First,** it is a traumatic experience to understand and remember the various combinations of 1 'sand O's representing numerous data and instructions. Also, writing error-free instructions is a slow process. **Secondly,** since every machine has its own machine language, the user cannot communicate with other computers (if he does not know its language). Imagine a Tamilian making his first trip to Delhi. He would face enormous obstacles as soon as he moved out for shopping. A language barrier would prevent him from communicating.

2. Assembly Language
An assembly language uses machine codes rather than numeric codes (as used in machine language). For example, ADD or A is used as a symbolic operation code to represent addition and SUB or S is used for subtraction. Memory locations containing data are given names .such as TOTAL, MARKS, TIME, MONTH, etc. As the computer understands only machine-code instructions, a program written in assembly language must be translated into machine language before the program is executed. , This translation is done by a computer program referred to as an assembler.

3. High Level Language
The assembly language is again a machine-oriented language and hence the program has to be different for different machines. The programmer should remember machine characteristics when he prepares a program. Writing a program in assembly language is still a slow and tedious task. High label languages are BASIC, COBOL, and FORTRAN etc.
These languages consist of a set of words and symbols and one can write programs using these in conjunction with certain rules like 'English' languages. These languages are oriented towards



the problem to be solved or procedures for solution rather than mere computer instructions. These are more user- centered than the machine-centered languages. They are better known as high level languages.

The most important characteristic of a high level language is that it is machine independent and a program written in a high- level language can be run on computers of different makes with little or no modification. The programmer need not know the characteristic of that machine. However, such programs are to be translated into equivalent machine-code instructions before actual implementations.

A program written in a high-level language is known as the source-program and can be run on different machines using different translators. The translated program is called the object program. The major disadvantage of high-level language is that they take extra time for conversion and thus they are less efficient compared to the machine-code languages. Figure shows the system of implementing the three levels of languages.

SUMMARY OF COMMON HIGH LEVEL LANGUAGE

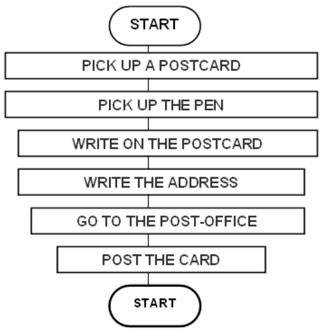
Year	Language Version	Name Derived from	Developed by	Application	Latest
1957	FORTRAN	Formula	IBM	Science, Engineering	FORTRAN77
1958	ALGOL	Translation Algorithmic	International Language	Engineering	ALGOL68
1959	LISP	List Processing	MIT, USA	Artificial Intelligence	LISP 1.6
1960	APL	A Programming Languages	IBM	Science, Engineering	APLSV
1961	COBOL	Common Business Oriented Language	Defense Dept. (USA)	Business	Cobol 85
1964	BASIC	Beginner All Purpose Symbolic Instruction Code	Dartmouth College	Engineering, Science, Business, Education	Standard Basic
1965	PL/1	Programming Language	IBM	General	ANSI PL/1
1970	Pascal	Blasie Pascal	Federal Institute Of Technology,	General Switzerland'	Standard Pascal
1972	PROLOG	Programming In Logic	University of Marselie	Artificial Intelligence	Standard PROLOG
1973	C	Earlier language	Bell Laboratory	General	ANSI C
1975	Ada	Augusta Ada	U.S. Defense	General	Ada
1983	C++	C	Bajaarne Stroustrup	OOP's	VC++
1995	Java	oak	Sun Microsystem	Internet	Java 1.4

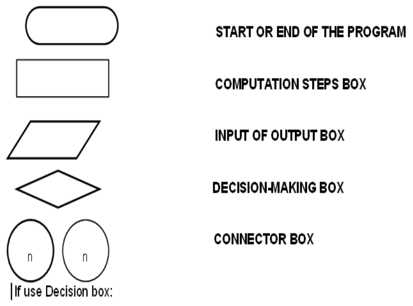
Algorithm and Flowcharts:

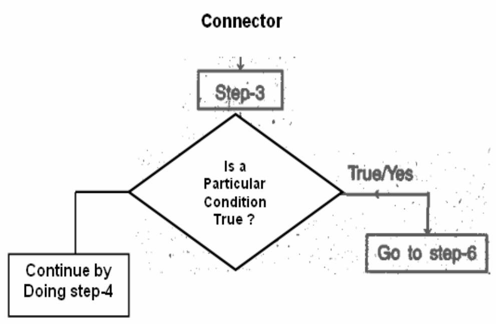
Algorithm: The set of instruction in a sequence manner is called algorithm.
Flowcharts: The diagrammatic representation of program is called Flow chart.

ALGORITHM AND FLOW CHART TO POST A CARD:

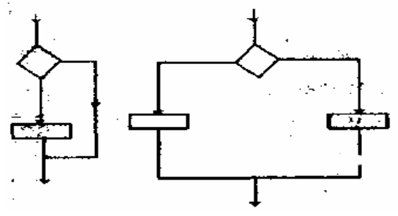
STEP-1.	**START**
STEP-2.	**PICK UP A POSTCARD**
STEP-3.	**PICK UP THE PEN**
STEP-4.	**WRITE ON THE POSTCARD**
STEP-5.	**WRITE THE ADDRESS**
STEP-6.	**GO TO THE POST-OFFICE**
STEP-7.	**POST THE CARD**
STEP-8	**STOP**



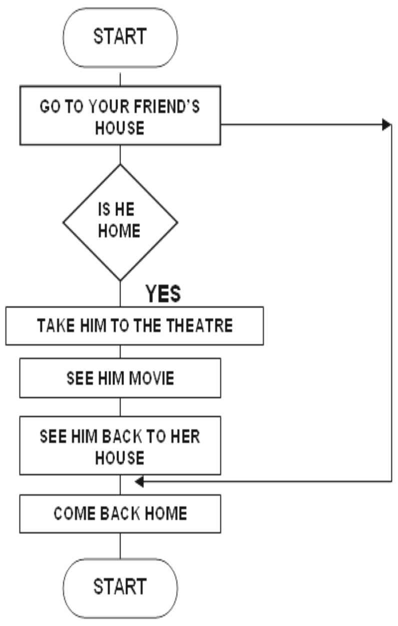




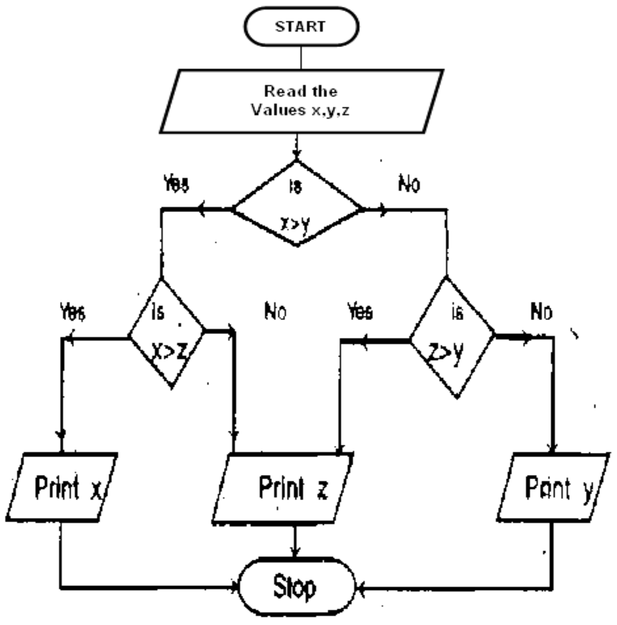
Branching: Branching refers to the process of the following one of two more optional paths of computations.



Example: Going to movies with your friend.



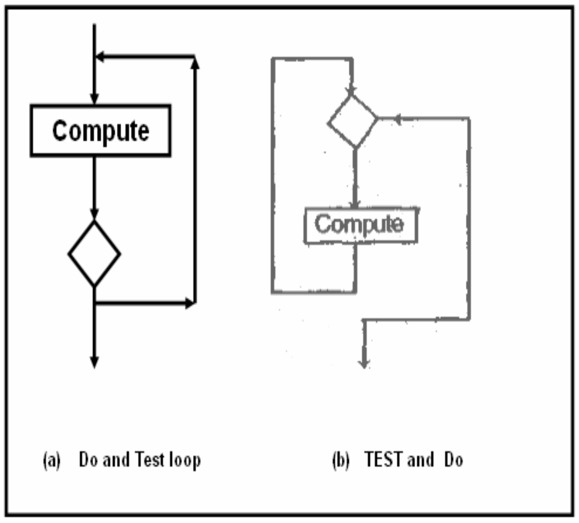
Example: Find out the greatest among three numbers



LOOPING:

Repeat the statement again and again is called **loop**. And this process is called **looping**. There are two types of loop. ***First*** is called **DO** and **Test loop** and ***second*** is called **TEST** and **DO** type loop.

a) **DO and Test loop:** In these loop firstly, we execute the statement and then Test the condition.
b) **TEST and DO:** In these loop firstly, we test the condition then execute the statement.



CHAPTER- 2

C FUNDAMENTALS

Introduction to ‘C’ Language

‘C’ is a general-purpose structured programming language. **C** has certain features which are best suited for low jobs and certain features which enable the programmer to perform high level jobs, thus bridging the gap between the machine language & high level language. Earlier there was a language **‘Basic combined Programming Language’** called **B**, which was modified by **Denis Ritchie** and was implemented, at **Bell Laboratories** in **1972**. This new language was called **‘C’** Language. **‘C’** is a middle-level language as it gives a minimal set of control and data-manipulation statements that they can use to define high-level constructs. It allows to efficiently communicating with a computer.

Features of ‘C’ Language

✓ **Small Instruction Code**
The code of C Language is very small as compared to most of the languages. It needs few instruction codes to perform a job.

✓ **Portable**
C Language is portable language. A language is said to be portable, if a program written in that language when taken to various other computers, runs with little or no modification.

✓ **No Formatting**
C does not impose any format restriction on user as in COBOL; one can start writing from any place and end it at any place.

✓ **Efficient & Fast**
Due to its variety of data types and powerful operators, programs written in C are fast & efficient.

Use of ‘C’ Language

C language is capable of system programming and application programming. System programming refers to a class of programs that either are part of or work with operating system of the computer. System programs make the computer able of performing useful work. System programs run very quickly.
C could be used for many types of application like- ***Developing operating system, Assemblers, Compilers, Interpreters, Editors, System Utilities, Writing TSR`s, Database Application, Networks systems.***

‘C’ as a structure programming

structure programming represents a procedure for describing the program in a series of stages or modules. It provides a clear and simple approach to program design. It is also called procedural language that is each statement in the language tells the computer to do something such as Get some input, add the numbers, display the output etc. A program in a procedural language is a list of instructions. The programmer creates the list of instructions and the computer carries them out.

Structure of ‘C’ Program is given below:-

```
#include directive(s)
main( )
{
        declaration of constants and variables
        sequence of instructions
}
```

Explanation of abovementioned structure program is as follow-
C program is organized into routines:

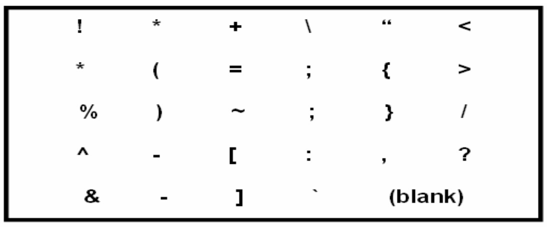
Main( ) –
It represents the routine where program execution must begin and as the name suggests is the main or controlling routine.

{ (Opening brace) & } (closing brace)-
The declaration of constants and variables and the language statements that make up the sequence of instructions contained in with in a function are delimited by the opening brace { and closing brace }.

; (semi-colons)) –
The declarations and executable statements within the function are separated by semi-colons.

The C Character Set

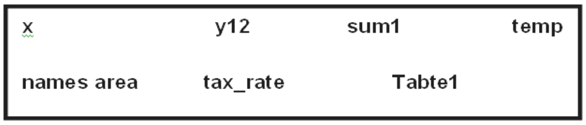
C uses the ***uppercase letters*** **A to Z**, the ***lowercase letters*** **a to z**, the ***digits*** **0 to 9**, and certain special characters as building blocks to form basic program elements (e.g., ***constants, variables, operators, expressions***). The special characters are listed below,



!	*	+	\	“	<
*	(	=	;	{	>
%	)	~	;	}	/
^	-	[	:	,	?
&	-	]	`	(blank)	

Identifier

The program element is called **identifier**.
Identifiers consist of letters and digits, in any order, except that the first character must be a letter. Both upper and lowercase fetters are permitted, the underscore character (_) can also be included to be a letter. An underscore is often used in the middle of an identifier.
Example: the following names are valid identifiers.



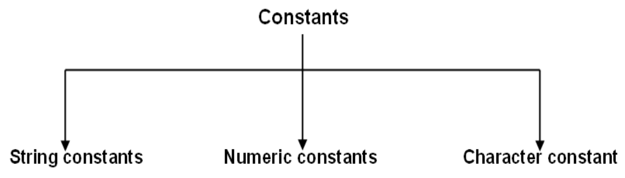
x	y12	sum1	temp
names area	tax_rate	Tabte1	

As per the characteristic of identifier we can .classified identifier into **variable and Constant**.

(a) Variable
(b) Constant

(a) Variables
The value of identifier change during the programming is called variable.

(b) Constant
The Fix the value of identifier in during the programming is called Constants. There are three types of constants; string constants, numeric constants and character constants.



1. String constants

A String constant is a sequence of alphanumeric characters enclosed in double quotation marks whose maximum length is 255 characters. *Following are the examples of valid string constants:-*

- **"The result = "**
- **"Rs. 2000,00"**
- **"This is test program by Arjun"**

2. Numeric Constants

Numeric constants are positive or negative numbers. There are four types of numeric constant. Integer constant, floating point constant. hex constant and octal constant. An integer constant may either be a short integer or a long integer. A floating point constant may either be of single precision or double precision.

Numeric constants	**Integer Integer**
	Short integer (short)
	Long integer (long)
Float	**(float)**
	(double)
	Long double
	Unsigned char
	Unsigned integer
Unsigned	**Unsigned short integer**
	Unsigned long integer
Hex	**Short hexadecimal**
	Long hexadecimal
Octal	**Short octal**
	Long octal

(a) Integer constants

Integer constants do not contain decimal points, Variables can be declared as integers in the following ways.

```
Int x, y;
Short int x,y;
Long int x,y;
```

In the first statement, the variables x and y take integer values. In the second statement x, y are of the short integer type. The third statement declares x, y as long integer type variables.

I Integer type data
The keyword 'int' stands for the integer data type in C change its size is either 16 or 32 bits.

II Short Integer data type
Normally, the 'short int' is used to declare the short integer data type in C whose maximum size is 16 bits longs.

III Long integer data type
Usually, the 'long int' stands for the long integer data type used in c and its size is 32 bits.

IV Unsigned data type
The undersigned numbers are whole numbers and always hold positive values and the sign bit also belongs to the value. The unsigned numbers may be classified into four : unsigned char, unsigned integer, unsigned short integer and unsigned long integer. The maximum size of the unsigned integer is 16 or 32 bits.

(b) Floating point constants-

Positive or negative numbers are represented in the exponential from (similar to scientific notation).

Examples-

9010E10
7810.11E-11
-10.990e8
-1.oole-1

The following floating point constant with their size: Float, Double, long double.

Data Types	Size in Bytes
Float	4
Double	8
long double	12 or 16

(c) Hex constants

Hexadecimal numbers are integer numbers of base 16 and their digits are 0 to 9 and A to F (a to f).

Examples-

0x0
0x3
22x3

(d) Octal constants

Octal numbers are integer numbers of base **8** and their digits are **0 to 7**.

Data Types	Size in Bytes
Char	1
Short	2
int	2 or 4
long	4 or 8

3. Character Constants

A character represented within single quotes denotes a character constant.

Here are some examples:-

‘A’
‘a’
‘.’
,
‘?’
Declaration of the character constants
char x;
char x,y,z;

The following characters are used as non-graphic characters. The backslash(\) is used to denote non-graphic character and other special characters for specific operation.

‘\ a’	**alert a bell character**
‘\ n’	**newline (line feed)**
‘\ t’	**horizontal tab**
‘\ b’	**backspace**
‘\ r’	**carriage return ?**
‘\ f’	**form feed**
‘\ v’	**vertical tab**
‘\ \’	**back slash**
‘\’’ `	**single quote**
‘\ o’	**null character**
‘\ ?’	**question mark**
‘\ ooo’	**octal value such as \033**
‘\ xhhh’	**hexadecimal value such as/xlh**

The following table summarizes the data types in c.		
Name	**Meaning**	**Size in Bytes**
char	character-	1
int	integer	2
long int	long integer (more digits)	4
short int	Short integer (fewer digits)	2
unsigned char	unsigned character	1
unsigned int	unsigned integer	2
unsigned short int	unsigned short integer	2
unsigned long int	unsigned long integer	4
float	floating point number	4
double	double precision-floating point number	8

Parts of C Program

1. Pre-Processor Directives,
2. Main Function.
3. Other Functions, Definition of Function etc.
4. All instructions write in tower case letter

Example:

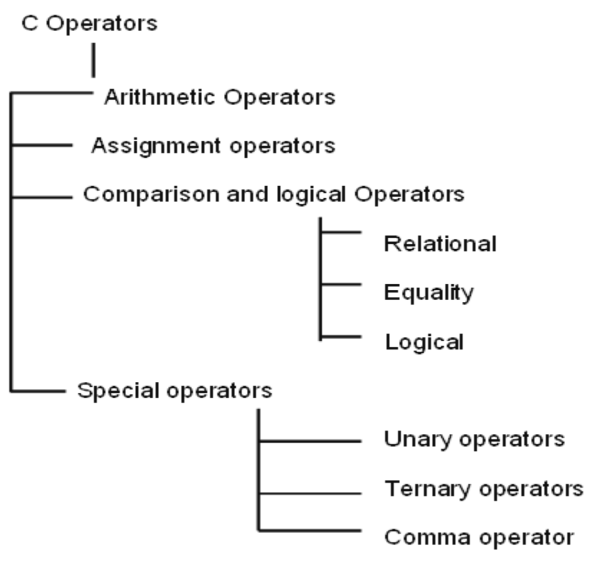
```
#include<stdio.h>
main()

{
            print(“Arjun\n”);
}
```

CHAPTER- 3

OPERATORS AND EXPRESSIONS

OPERATORS



1. Arithmetic operator

Arithmetic operator is used for arithmetic operations.

Operator	Meaning
+	Addition
=	subtraction
*	multiplication
/	division
%	modulo (remainder of an integer division)

Precedence rules for arithmetic Operators:

( )	For grouping the variables
-	(unary for negative number)
*, /	(multiplication and division)
+ -	(addition and subtraction)

For example: if the following expression is not properly grouped using parentheses then the computer will evaluate as per the precedence.

X + y * x – z where x = 5, y = 6 and z = 8
5 + (6 * 5) – 8
(5 Z + 30) – 8
35 – 8
27 ***The result is 27***

Suppose, if we intended to evaluate the above expression as **(x + y) * (x – z)**
Then the parentheses will be evaluated because it has the highest priority and the result will be different from the pervious expression.

(5 + 6) * (5 – 8)
11 * - 3
- 33

2. Assignment Operators

An assignment is used to assign back to a variable, a modified value of the present holding.

OPERATOR	MEANING
=	Assign right hand side (RHS) value to the left hand side(LHS)
+=	Value of LHS variable will be added to the value of RHS and assign it back to the variable in LHS.
-=	Value of RHS variable will be subtracted from the value of LHS and assign it back to the variable in LHS
*=	Value LHS variable will be multiplied by the value of RHS and assign in back to the variable in LHS
/=	Value of LHS variable will be divided by the value of RHS and assigning in back to the variable in LHS
%=	The remainder will be stored back to the LHS after integer division is carried out between the LHS variable and the RHS variable

For Example:

X + = y is equal to x = x + y
X - = y is equal to x = x - y

3. Comparison and Logical Operators

The comparison operators and logical operator can be grouped into three. They are relational operators, equality operators and logical operators.

OPERATORS	MEANING
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
! =	Not equal to
&&	Logical AND
I I	Logical Or
!	Not

a) Relational Operators
Relational operator is used to compare the values.

OPERATOR	MEANING
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

For Example:

EXPRESSION	RESULT
3 > 4	false
6< = 2	false
10 > -32	true
(23*7) >=(-67 + 89)	true

b) Equality Operators:

These operator is use to check the value is equal or not.

OPERATOR	MEANING
==	Equal to
! =	Not equal to

For Example: a = 4, b = 6, and c = 8

EXPRESSION	RESULT
A = = b	False
(a*b) ! = c	True
“s” = = ‘y’	false

c) Logical Operators:

OPERATOR	MEANING
&&	Logical AND
I I	Logical Or
!	Not

<u>SPECIAL OPERATORS</u>

1. Unary Operators

OPERATOR	MEANING
*	Contents of the storages field to which a pointer is pointing.
&	Address of a variable
-	Negative value (minus sign)
!	Negation
~	Bitwise complement
++	Incrementer
- -	Decrementer
type	Forced type of conversion
sizeof	Size of the subsequent data type or type in byte.

2. Ternary operator (?:):

These operators are used for three expressions.

The Syntax:

```
expression1?expression2:expression3
```

if the expression1 is true then expression2 is evaluate otherwise expressions will be evaluate.

For Example:

c=a>5:3:20

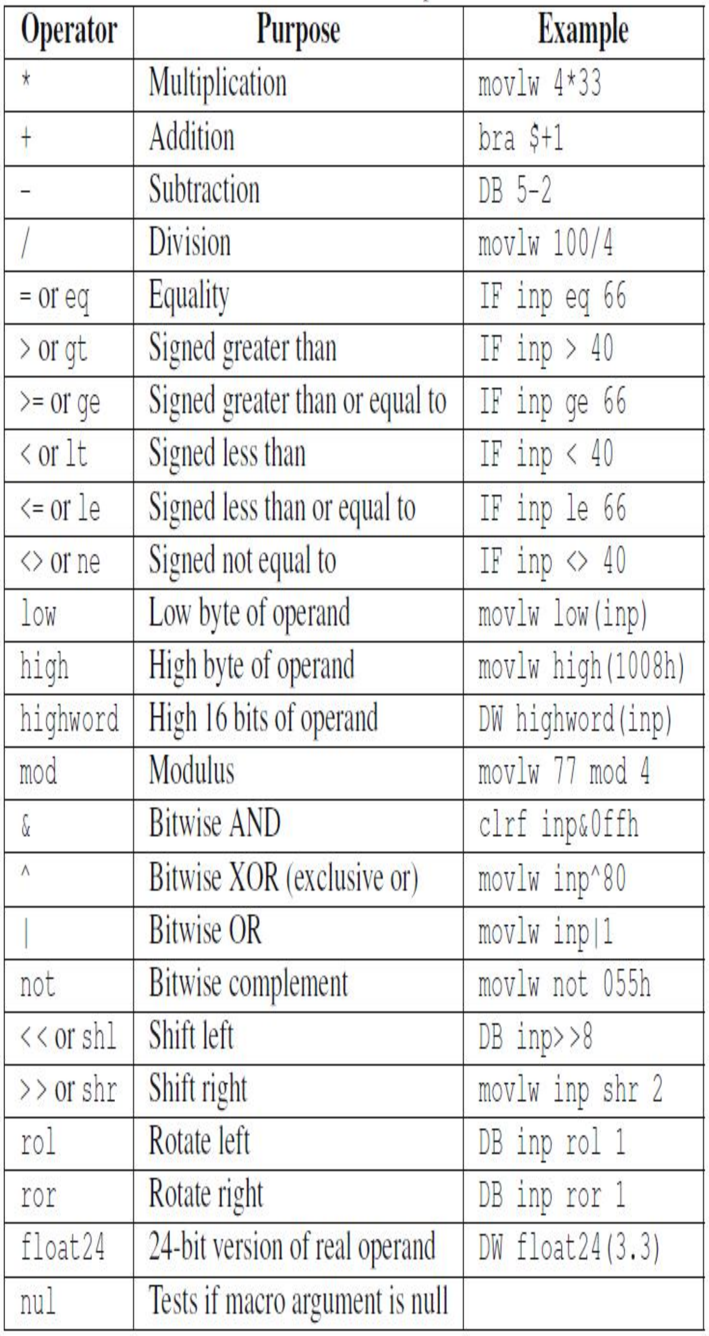
If c is greater than 5 than c store 3 otherwise c store 20 values.

3. Comma Operator :

This operator is used as a separator in variable declaration.

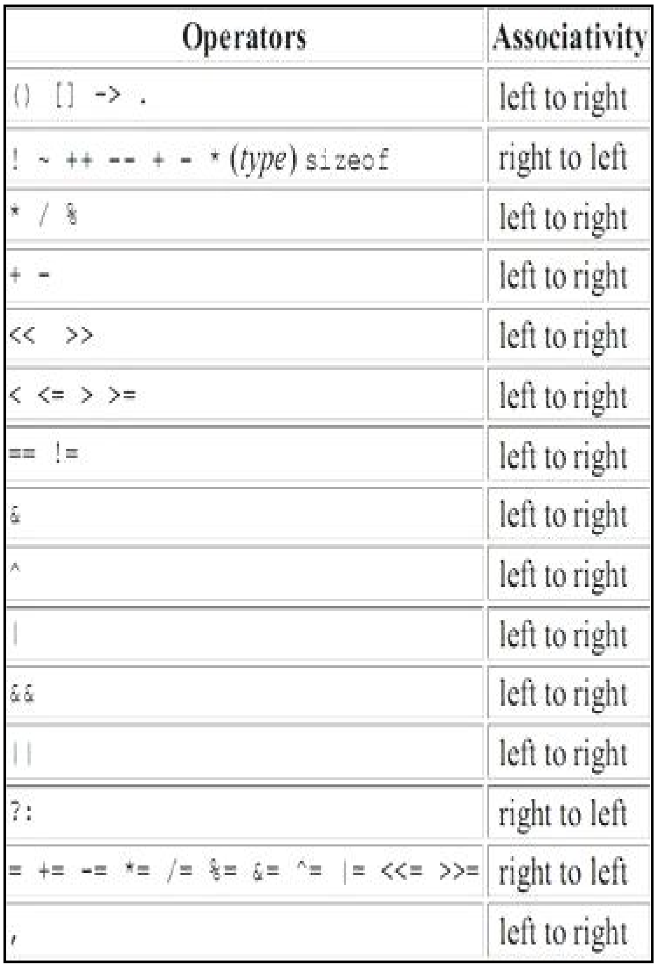
int a, b, c:
Float x, y, z;

<u>Operator, Purpose & Example</u>



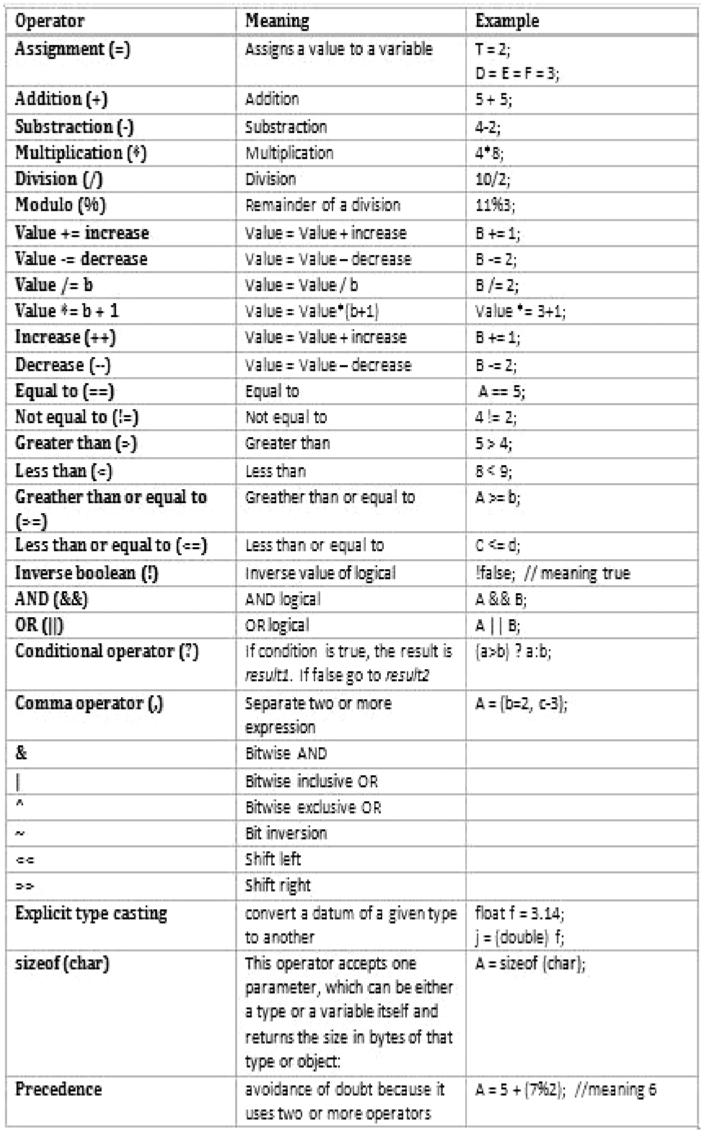
Operator	Purpose	Example
*	Multiplication	movlw 4*33
+	Addition	bra $+1
-	Subtraction	DB 5-2
/	Division	movlw 100/4
= or eq	Equality	IF inp eq 66
> or gt	Signed greater than	IF inp > 40
>= or ge	Signed greater than or equal to	IF inp ge 66
< or lt	Signed less than	IF inp < 40
<= or le	Signed less than or equal to	IF inp le 66
<> or ne	Signed not equal to	IF inp <> 40
low	Low byte of operand	movlw low(inp)
high	High byte of operand	movlw high(1008h)
highword	High 16 bits of operand	DW highword(inp)
mod	Modulus	movlw 77 mod 4
&	Bitwise AND	clrf inp&0ffh
^	Bitwise XOR (exclusive or)	movlw inp^80
|	Bitwise OR	movlw inp|1
not	Bitwise complement	movlw not 055h
<< or shl	Shift left	DB inp>>8
>> or shr	Shift right	movlw inp shr 2
rol	Rotate left	DB inp rol 1
ror	Rotate right	DB inp ror 1
float24	24-bit version of real operand	DW float24(3.3)
nul	Tests if macro argument is null	

Operators & Associativity



Operators	Associativity
`() [] -> .`	left to right
`! ~ ++ -- + - *` (*type*) `sizeof`	right to left
`* / %`	left to right
`+ -`	left to right
`<< >>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right

Summary of Operator



Operator	Meaning	Example

Assignment (=)	Assigns a value to a variable	T = 2;
		D = E = F = 3;

Addition (+)	Addition	5 + 5;
Substraction (-)	Substraction	4-2;
Multiplication (*)	Multiplication	4*8;
Division (/)	Division	10/2;
Modulo (%)	Remainder of a division	11%3;
Value += increase	Value = Value + increase	B += 1;
Value -= decrease	Value = Value – decrease	B -= 2;
Value /= b	Value = Value / b	B /= 2;
Value *= b + 1	Value = Value*(b+1)	Value *= 3+1;
Increase (++)	Value = Value + increase	B += 1;
Decrease (--)	Value = Value – decrease	B -= 2;
Equal to (==)	Equal to	A == 5;
Not equal to (!=)	Not equal to	4 != 2;
Greater than (>)	Greater than	5 > 4;
Less than (<)	Less than	8 < 9;
Greather than or equal to (>=)	Greather than or equal to	A >= b;
Less than or equal to (<=)	Less than or equal to	C <= d;
Inverse boolean (!)	Inverse value of logical	!false; // meaning true
AND (&&)	AND logical	A && B;
OR (||)	OR logical	A || B;
Conditional operator (?)	If condition is true, the result is *result1*. If false go to *result2*	(a>b) ? a:b;
Comma operator (,)	Separate two or more expression	A = (b=2, c-3);
&	Bitwise AND	
|	Bitwise inclusive OR	
^	Bitwise exclusive OR	
~	Bit inversion	
<<	Shift left	
>>	Shift right	

Explicit type casting	convert a datum of a given type to another	float f = 3.14;
		j = (double) f;

sizeof (char)	This operator accepts one parameter, which can be either a type or a variable itself and returns the size in bytes of that type or object:	A = sizeof (char);
Precedence	avoidance of doubt because it uses two or more operators	A = 5 + (7%2); //meaning 6

CHAPTER- 4

DATA INPUT AND OUTPUT

Single Character Input-The getchar Function

Single characters can be entered into the computer using the C library function getchar.
In general terms, a reference to the getchar function is written as: **char variable = getchar ();**

Single character output -The putchar function

Single characters can be displayed (i.e., written out of the computer) using the C library function put char.
In general terms a reference to the putchar function is written as: **putchar (character variable);**

Entering input data by the scant function

Input data can be entered into the computer from a standard input device by means of the C library function scanf.
***Syntax*:**

```
Scanf(“%conversion character”,&variable);
```

Conversion Character-
It indicates the type of the corresponding date item.

& -
lt indicates the data item store in computers memory.

Example:

Int a;
Scanf(“%d”,&a);

Conversion character	Meaning
c	data item is a single character
d	data item is a decimal integer
e	data item is a floating-point value
f	data item is a floating-point, value
g	data item is floating-point, value
i	date item is a decimal, hexadecimal or octel integer
o	date item is an octal integer
s	data item is a string
0	data item is an octal integer
u	date item is an unsigned decimal integer

character	Conversion Meaning
C	data item is a single char
X	data item is a hexadecimal integer
[...]	data item is a string which may include whitespace characters.

Writing Output Date- The printf Function

Output data can be written from the computer onto a standard output device using the library function printf.
***Syntax*:**

```
printf (“%conversion charactei”,variable);
```

Conversion Character-
It indicates the type of the corresponding date item.
***Example*:**

int a;
printf("%d",a);

Conversion Character	Meaning
c	data item is a displayed as a single character
d	data item is displayed a decimal integer
e	data item is displayed a floating-point value
f	data item is displayed^ floating-point value
g	data item is displayed floating-point value
h	data item is displayed a short integer
i	data item is displayed a decimal, hexadecimal or octal integer
o	data item is displayed an octal integer
s	data item is displayed a string
0	data item is displayed an octal integer
u	data item is displayed an unsigned decimal integer
x	data item is displayed a hexadecimal integer

THE GETS AND PUTS FUNCTIONS

gets ( ) : These function is used to input one or more than string in variable.

***Example*:**

```
char name[20];
void main()
 {
gets(name);
 }
```

puts ( ) : These function is used to display one or more than string in variable on output.

***Example*:** Wap for entered and display line.

```
char name[20];
void main()
{
gets(name);
puts(name);
}
```

1. Example program: 1 WAP for display **"SARVA"**

```
#include<stdio.h>
#include<conio.h>
 void main()
 {
printf("SARVA\n");
getch();
 }
```

Output SARVA

2. Example program: 2 WAP for Addition of three Nos.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
printf("enter the value of a \n");
scanf(“%d",&a);
printf(“enter the value of b \n");
scanf("%d",&b);
c=a+b;
printf("the value of c=%d”,c);
getch();
}
```

Output:

```
enter the value of a
6
enter the value of b
6
the value of c=12
```

3. Example program: 3 WAP for Simple interest

```
#include<stdio.h>
#include<conio.h>
void main()
{
float p,r,t,si;
clrscr()
printf(“enter the value of principle\n”
scanf(“%f”,&p);
printf("enter the value of rate \n");
scanf(“%f”,&r);
printf(“enter the value of tim e \n");
scanf( “%f”,&t);
si=(p*r*t)/100;
printf(“the result of si =%f”,si);
getch();
}
```

Output

```
Enter the value of principle
1000
enter the value of rate
2
enter the value of time
2
the result of si=40.000000
```

4. Example program: 4 WAP for entered line and Display line.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[10];
clrscr();
printf("enter the line\n");
gets(name);
puts(name);
}
```

Output

```
enter the line
India is great
India is great
```

5. Example program: 5 Wap for entered & display a single character

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name;
printf(“enter a character”);
name=getchar();
putchar(name);
getch();
}
```

Output

```
Enter a character
a
```

6. Example program: 6 Wap for Area of Circle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float pi=3.14,r,a;
clrscr();
printf(“enter the radius\n”);
scanf(“%f”,&r);
a=pi*(r*r);
printf(“the area of circle=%f”,a);
getch();
}
```

Output

```
enter the radius
3
the area of circle=28.260000
```

CHAPTER- 5

DECISION CONTROL STRUCTURES

Introduction

In **C** programming the following main type of Decision statement used for implementing the decision; control instruction.

1. **The if- statement**
2. **The if- else statement**
3. **The conditional operator.**
4. **Switch case statement**

1. The if- statement

```
Syntax:
        if (boolean expression )
        statement;
```

***Example*:**

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

Here, we given the condition that if **a** is greater then b that time the **a** is greater than **b** will be displayed on the screen

2. The If- else statement

These statements execute statement1 when is condition true and also execute statement2. when is condition false.

```
Syntax:
        if (condition)
                statement1;
        else
                statement2;
```

***Example*:**

```
        if (a>b)
        printf("a is greatest");
        else
        printf(" b is greatest");
```

Here, if a>b then print **a** is greatest statement.
And if condition is false then it will print **b** is greatest.

In use nested if we use another if in under of if

```
Syntax:
        If (condition1)
        Statement1;
else
        if (condition2)
        statement2;
else
        statements3;
```

Example: if(a>b)

```
                printf('ca is greter than b");
        else
        if(a= =b)
        printf(" a is eqaul to b");
        else
        printf (" b is greater than a");'
```

Here, if a>b, in true it will print a is greter than b else check again the a = = b, in true it will print a is eqaul to b else print b is greater than a.

3. The conditional operator

***Syntax*:**

```
expression 1 ?expression 2:expressioh3
```

***example*:**

```
        int a,c;
        c=(a>8?3:6);
```

Here, this statement will store 3 in c if a is greater then 8 otherwise it will store 6 in c.

4. Switch case statement

This statement uses a particular group of statements to be choosed from several available groups.

```
Syntax:
        switch(variable)
        {
        case constant 1: statement-1;
        break;      ,
        case constant n: statement-n;
        break;
        default: statement
        }
```

***Example program* 1: Wap for greatest among two Nos.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
printf ("enter the value of a\n");
scanf("%d",&a);
printf ("enter the value of b\n");
scanf("%d",&b);
if(a>b)
printf(" a is greater than b");
else
printf("b is greter than a");
getch();
}
```

Output

```
enter the value of a
5
enter the value of b
6
b is greter than a
```

***Example program* 2: Wap for greatest among three Nos.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("enter the value of a\n");
scanf("%d",&a);
printf ("enter the value of b\n");
scanf("%d",&b);
printf("enter the value of c\n");
scanf("%d",&c);
if((a>b)&(a>c))
printf ("a is greater");
else
if(b>c)
printf("b is greater");
else
printf("c is greater ");
getch();
}
```

Output

```
enter the value of a
 1
 enter the value of b
 2
 enter the value of c
 3
 c is greter
```

***Example program* 3: Wap for leap year.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int year;
printf("enter the year\n");
scanf("%d",&year);
if (year%4==0)
printf("the year is leap year");
else
'printf("the year is not leap");
getch();
}
```

Output

```
enter the year
2000
the year is leap year
```

***Example program* 4: Wap for print seven day of the week.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int ch;
printf("enter the choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1 : printf("Sunday");
break;
case 2 : printf ("Monday");
break;
Case 3 : printf("Tuesday");
break;
case 4 : printf("Wednesday");
break;
case 5 : printf("Thursday");
break;
case 6 : printf("Friday");
break;
case 7 : printf("Saturday");
break;
default: printf(" Invalid choice");
getch();
        }}
```

Output

```
enter the choice
6
 Friday
```

CHAPTER- 6

LOOP CONTROL STRUCTURES

Introduction

1. **while Statement**
2. **do-while Statement**
3. **for Statement**

   In every loop three things are main- initialization of counter variable, condition, counter variable (increment/decrement purpose)

1. **while Statement**

   This loop is test-Do type of loop, means first it check the condition and then execute the statement.

```
Syntax:
          initialization of variable;
          while(condition)
                    {
                              statement
                              counter variable
          };

Example: Print  ARJUN word 10 times.
          int  i=0;
                   while(i<10)
                             {
printf("ARJUN\n");
I++;
};
```

2. **do-while Statement**

   This loop is Do-test type of loop, means first it executes the statement and then checks the condition .

```
Syntax:
          initialization of variable;

          do  {
                         statement;
                         counter variable;
          }while (condition );
```

***Example*: Print ARJUN word 10 times.**

```
          int  i=0;


                        do  {
print("ARJUN\n");
I++
}while(i<9);
```

3. **for Statement**

This loop is test-Do type of loop, means first it check the condition and then execute the statement.

```
Syntax:
            for(intialization;condition;counter variable)
           {
                         statement;
           }
```

Exmple program 1: Wap to print 10 Nos using while loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
i=0;
Clrscr ();
printf("Print 10 number\n");
while(i<10)
{
printf("%d\n",i);
i++;
}
getch();
}
```

Output

```
Print 10 number
0
      1
      2
      3
      4
      5
      6
      7
      8
      9
```

Example program 2: Wap to print 10 Nos using do-white loop.

```
#include<stdio.h>
#jnclude<conio.h>
void main()
{
int i;
i=0;
clrscr();
printf("Print 10 number\n");
do
{
printf("%d\n",i);
i++;
}
while(i<10);
getch();
}
```

Output

```
Print 10 number
0
1
2
3
4
5
6
7
8
9
```

Example program 3: Wap to print 10 Nos using for loop

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
printf("Print 10 number\n");
for(i=0;i<10;i++)
 {
 printf("%d\n",i);
 }
 getch();
 }
```

Output

```
Print 10 number
0
1
2
3
4
5
6
7
8
9
```

CHAPTER- 7

FUNCTION

Introduction

The self contain program is called function.
For create the function the following three major steps are necessary.

1. **Function Prototype**
2 **Function definition**
3. **Function calling**

1. Function Prototype

In these, we define the type of function.
As per the data type we can define function also.
(int function, float function, double function, char function and void function.)

In first four type of function return the value but void function not return nothing.

As per the use argument in function there are following type

According to the argument there are following function:

Without argument

In these we not use any argument.
Example:

```
void arjun();
```

With Argument

In these we use any argument
Example:

```
void arjun(int  a,  int  b);
```

With Argument and return

In these we use any argument with return type
Example:

```
void arjun(int  a);
return(a);
```

2. Function definition

In these we write the statement which we want to execute by the function.

Example:

```
void arjun()
{
        print("hallo ARJUN");
}
```

3. Function calling

In these we execute the function definition where we write these function.

```
        #include<stdio.h>
        #include<conio.h>
        void main();
        void mahakal();
        void shiva();
        void nirmala();
        void main()
{
        arjun();
        mahakal();
        nirmala();
        getch();
        }
        void arjun()
        {
        printf("l am ARJUN function\n");
        }
        void mahakal()
        {
        printf("I am MAHAKAL function\n");
        }
        void nirmala()
        {
        printf("l am NIRMALA function\n");
        }
        void shiva()
        {
        printf("I am SHIVA function\n");
        getch();
        }
```

Output

```
WE ARE IN MAIN FUNTION
   I am ARJUN function
   I am MAHAKAL function
   I am SHIVA function
   I am NIRMALA function
```

Example program 2: Wap for with argument function.

```
#include<stdio.h>
#include<conio.h>
void addition(int a,int b);
void main()
{
int a,b;
clrscr();
printf(" enter the value of a\n");
scanf("%d",&a);
printf(" enter the value of b\n");
scanf("%d",&b);
addition(a,b);
getch();
}
void addition(int a,int b)
{
int c;
c=a+b;
printf("the result of addition=%d",c);
getch();
}
```

Output

```
enter the value of a,
  1
   enter the value of b
  2
  the result of addition=3
```

Example program 3: Wap for argument with return type function.

```
#include<stdio.h>
#include<conio.h>
void addition(int a,int b);
void main()
{
int a,b;
clrscr();
printf(" enter the value of a\n");
scanf("%d",&a);
printf(" enter the value of b\n");
scanf("%d",&b);
addition(a,&b);
getch();
}
void addition(int a,int b)
{
int c;
c=a+b;
getch();
      }
```

Output

```
Enter the value of a
1
enter the value of b
2
the result=3
```

CHAPTER- 8

ARRAY

Introduction

Array is collection of same type of data.

<u>Declaration of Array</u>

One dimensional Array:

In General definition one dimensional of array data store in one row or column.

data type variable [size of array];

Example: int a [10];

Two dimensional Array:

In General definition two dimensional of array data Store in one row or column.

data type variable[size of array][size of array];

Example program 1 : Wap for entered and display element in array.

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a[10],i;
clrscr();
printf("enter the element in array\n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("entered the element in array\n");
for(i=0;i<10;i++)
{
printf("%d",a[i]);
}
getch();
}
```

Output

```
enter the element in array
1
2
3
4
5
6
7
8
9
10
enter the element in array
12345678910
```

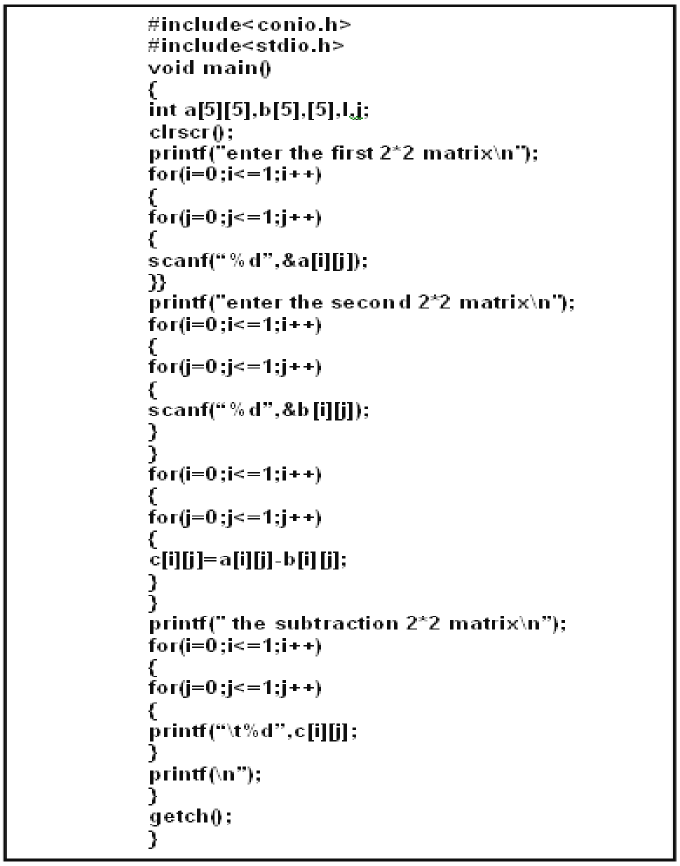
Example program 2: Wap for addition of two matrix.

```
#include<conio.h>
#include<stdio.h>
void main()
{
int a[5],[5],b[5],[5],i,j;
clrscr();
printf("enter the first 2*2 matrix\n");
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
scanf("%d",&a[i][j]);
}}
printf("enter the second 2*2 matrix\n");
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
scanf("%d",&b[i][j]);
}
}
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}
printf(" the addition 2*2 matrix\n");
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
printf("\t%d",c[i][j]);
}
printf("\n");
}
getch();
}
```

Output

```
Enter the first 2*2 matrix
1
1
1
1
enter the second 2*2 matrix
1
1
1
1
the addition 2*2 matrix
2     2
2     2
```

Example program3: Wap for subtraction of two matrix



```
#include<conio.h>
#include<stdio.h>
void main()
{
int a[5][5],b[5],[5],I,j;
clrscr();
printf("enter the first 2*2 matrix\n");
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
scanf(“%d”,&a[i][j]);
}}
printf("enter the second 2*2 matrix\n");
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
scanf(“%d”,&b[i][j]);
}
}
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
c[i][j]=a[i][j]-b[i][j];
}
}
printf(" the subtraction 2*2 matrix\n”);
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
printf(“\t%d”,c[i][j];
}
printf(\n”);
}
getch();
}
```

Output

```
Enter the first 2*2 matrix
2
2
2
2
enter the second2*2 matrix
1
1
1
1
the subtraction 2*2 matrix
    1    1
    1    1
```

Unit-II

C++ LANGUAGE

CHAPTER- 1

'C++' FUNDAMENTAL

INTRODUCTION TO C++
C++ is a Object oriented programming language. C++ was originally developed in 1983 by Bajaame Stroustrup.
A C++ program is a collection of commands, which tell the computer to do "*something*". This collection of commands is usually called C++ source code, source code or just code. Commands are either "functions" or "keywords". Keywords are a basic building block of the language, while functions are, in fact, usually written in terms of simpler functions--you'll see this in our very first program, below. (Confused? Think of it a bit like an outline for a book; the outline might show every chapter in the book; each chapter might have its own outline, composed of sections. Each section might have its own outline, or it might have all of the details written up.) Thankfully, C++ provides a great many common functions and keywords that you can use. But how does a program actually start? Every program in C++ has one function, always named main, that is always called when your program first executes. From main, you can also call other functions whether they are written by us or, as mentioned earlier, provided by the compiler.

The C++ character set
C++ character set same as c uses the uppercase letters A to Z, the lowercase letters a to z, the digits 0 to 9, and certain special characters as building block to form basic program elements (e.g., constants, variables, operator, expressions).

Identifier
The C++ identifier is same as c programming language so refer c programming part. As per the characteristic of identifier we can classify identifier into variable and constant.

(a) Variable **(b)** Constant

(a) Variables:

The C++ **Variables** is same as c programming language so refer c programming part.

(b) Constant:

The C++ **Constant** is, same as c programming Language so refer c programming part. There are ***three*** types of constants:- ***String constants, Numeric constants and Character constants***.

<u>String Constants</u>
A String constant is a sequence of alphanumeric characters enclosed in double quotation marks whose maximum length is 255 characters. Following are the examples of valid string constants.

1. "The result - "
2. "Rs. 2000,00"
3. "This is test program by Sarva"

<u>Numeric Constants</u>
Numeric constants are positive or negative numbers. There are four types of numeric constants, integer constant and floating point constant, hex constant and octal constant. An integer constant may either be a short integer or a long integer. A floating point constant may either be of single precision or double precision.

Numerate constants	Integer integer
	Short integer (short)
	Long integer (long)
Float	(float)
	(double)
	Long double
Unsigned	Unsigned char
	Unsigned integer
	Unsigned short integer
	Unsigned long integer
Hex	Short hexadecimal
	Long hexadecimal
Octal	Short octal
	Long octal

(a) Integer constants:

Integer constants do not contain decimal points, Variables can be declared as integers in the following ways.

```
int x, y;
short  int  x;y;
long int x,y;
```

In the first statement, the variables x and y take integer values. In the second statement x, y are of the short integer type. the third statement declares x, y as long integer type variables.

(i) Integer type data:- The keyword 'int' stands for the integer data type in C an its size is either 16 or 32 bits. A 16 bit integer may fall in the range of 2^{15} to 2^{15}-1, while 32 bit integers may fall in the range of -2^{32} to 2^{31} -1.

(ii) Short integer data type:- Normally, the 'short int' is used to declare the short integer data type in C whose maximum size is 16 bits long. It may fall in the range between -32,768 to 32,767 or 2^{15} to 2^{15}-1.

(iii) Long integer data type:- Usually, the 'long int' stands for the long integer data type used in C++ and its size is 32 bits, It may fall in the range of -2,147,483,648 to 2,147.483.647 or -2^{31} to 2^{31}- 1.

(iv) Unsigned data type:- The unsigned numbers are whole number s and always hold positive values and the sign bit also belongs toe the value. The unsigned numbers may be classified into four types; unsigned char, unsigned integer, unsigned short integer and unsigned long integer. The maximum size of the unsigned integer is 16 or 32 bits.

(b) Floating point constants- Positive or negative numbers are represented in the exponential form (similar to scientific notation).

Examples:
9010E10
7810.11E-11
-10.990e8
-1.oole-1
The following floating point constant with their size: Float, Double, long double.

Data Types	Size in Bytes
Float	4
Double	8
long double	12 or 16

(C) Hex constants:- Hexadecimal numbers are integer numbers of base 16 and their digits are 0 to 9 and A to F (a to f).
Example:-
0x0
-Ox3
22x3.

(d) Octal constants:- Octal numbers are integer numbers of base 8 and their
digits are 0 to 7

Data Types	Size in Bytes
char	1
short	2
int	2 or 4
long	4 or 8

Character Constants

A character represented within single quotes denotes a character constant.

Here are some examples,
'A'
'a'
'.'
"?'

Declaration of .the character constants.
char x
char x,y,z;

The following table summarizes the data types in C++

Name	Meaning	Size In Bytes
char	character	1
int	integer	2
long int	long integer (more digits)	4
short int	Short integer (fewer digits)	2
unsigned char	unsigned character	1
unsigned int	unsigned integer	2
unsigned short int	unsigned short integer	2
unsigned long int	unsigned long integer	4
float	floating point number	4
double	double precision floating	8
	point number	

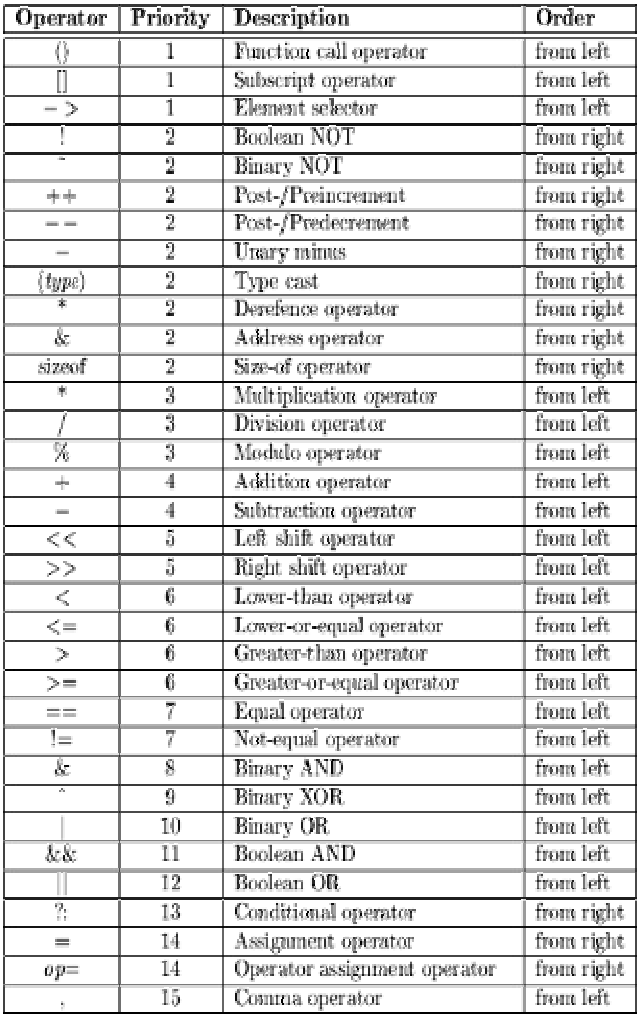
CHAPTER- 2

OPERATORS AND EXPRESSIONS

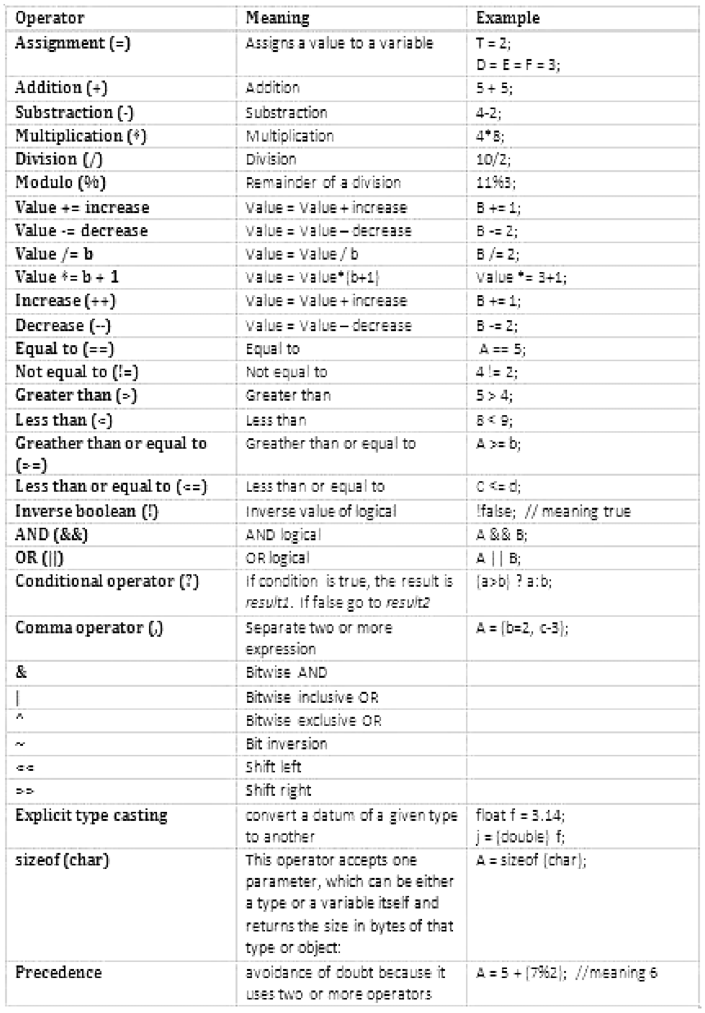
OPERATORS

The **C++ Operators** is same as c programming c programming part.

Summary & Example of C++ Operators



Operator	Priority	Description	Order
()	1	Function call operator	from left
[]	1	Subscript operator	from left
− >	1	Element selector	from left
!	2	Boolean NOT	from right
~	2	Binary NOT	from right
++	2	Post-/Preincrement	from right
−−	2	Post-/Predecrement	from right
−	2	Unary minus	from right
(type)	2	Type cast	from right
*	2	Derefence operator	from right
&	2	Address operator	from right
sizeof	2	Size-of operator	from right
*	3	Multiplication operator	from left
/	3	Division operator	from left
%	3	Modulo operator	from left
+	4	Addition operator	from left
−	4	Subtraction operator	from left
<<	5	Left shift operator	from left
>>	5	Right shift operator	from left
<	6	Lower-than operator	from left
<=	6	Lower-or-equal operator	from left
>	6	Greater-than operator	from left
>=	6	Greater-or-equal operator	from left
==	7	Equal operator	from left
!=	7	Not-equal operator	from left
&	8	Binary AND	from left
^	9	Binary XOR	from left
|	10	Binary OR	from left
&&	11	Boolean AND	from left
||	12	Boolean OR	from left
?:	13	Conditional operator	from right
=	14	Assignment operator	from right
op=	14	Operator assignment operator	from right
,	15	Comma operator	from left



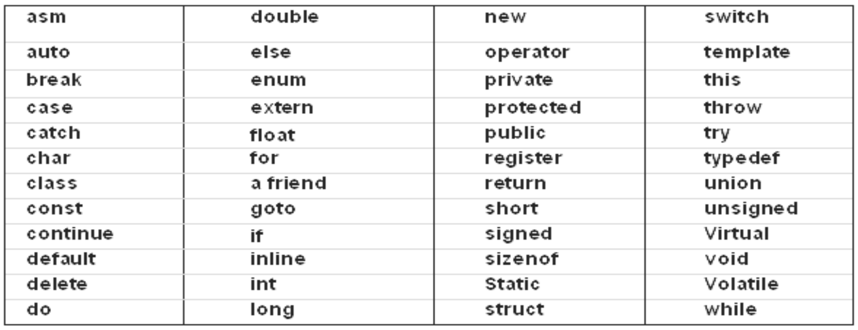
Operator	Meaning	Example

Assignment (=)	Assigns a value to a variable	T = 2;
		D = E = F = 3;

Addition (+)	Addition	5 + 5;
Substraction (-)	Substraction	4-2;
Multiplication (*)	Multiplication	4*8;
Division (/)	Division	10/2;
Modulo (%)	Remainder of a division	11%3;
Value += increase	Value = Value + increase	B += 1;
Value -= decrease	Value = Value – decrease	B -= 2;
Value /= b	Value = Value / b	B /= 2;
Value *= b + 1	Value = Value*(b+1)	Value *= 3+1;
Increase (++)	Value = Value + increase	B += 1;
Decrease (--)	Value = Value – decrease	B -= 2;
Equal to (==)	Equal to	A == 5;
Not equal to (!=)	Not equal to	4 != 2;
Greater than (>)	Greater than	5 > 4;
Less than (<)	Less than	8 < 9;
Greather than or equal to (>=)	Greather than or equal to	A >= b;
Less than or equal to (<=)	Less than or equal to	C <= d;
Inverse boolean (!)	Inverse value of logical	!false; // meaning true
AND (&&)	AND logical	A && B;
OR (||)	OR logical	A | | B;
Conditional operator (?)	If condition is true, the result is *result1*. If false go to *result2*	(a>b) ? a:b;
Comma operator (,)	Separate two or more expression	A = (b=2, c-3);
&	Bitwise AND	
|	Bitwise inclusive OR	
^	Bitwise exclusive OR	
~	Bit inversion	
<<	Shift left	
>>	Shift right	

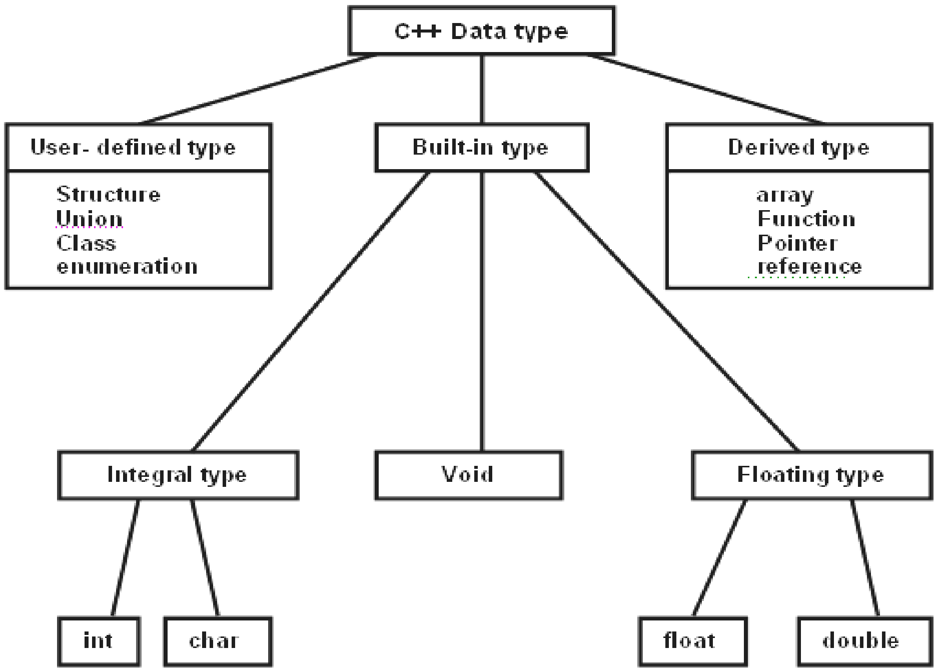
Explicit type casting	convert a datum of a given type to another	float f = 3.14;
		j = (double) f;

sizeof (char)	This operator accepts one parameter, which can be either a type or a variable itself and returns the size in bytes of that type or object:	A = sizeof (char);
Precedence	avoidance of doubt because it uses two or more operators	A = 5 + (7%2); //meaning 6



asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	a friend	return	union
const	goto	short	unsigned
continue	if	signed	Virtual
default	inline	sizenof	void
delete	int	Static	Volatile
do	long	struct	while

BASIC DATA TYPES



Type	Bytes	Range
char	1	-128 to 127
unsigned char	1	o to 255
signed char	1	-128 to
int	2	-32768 to 32767
unsigned int	2	o to 65535
signed int	2	-31768 to 32767
short int	2	-31768 to 32767
unsigned short int.	2	0 to 65535
signed short int	2	32768 to 32767
long int	4	-214783648 to 2147483647
signed short int	4	-21474648 to 2147483647
using long int	4	o to 4294967295
float	4	3.4E-38 to 3.4E + 38
double	8	1.7-E-308 to 1.7E + 308
long double	10	3.4E-4932 to 1.1E + 4932

CHAPTER- 3

DATA INPUT AND OUTPUT

Entering input data by the cin statement
Input data can be entered into the computer from a standard input device by cin statement.

Syntax:

Cin>>variable;
Int a;
Cin>>a

Writing Output Data by cout statement
Output data can be written from the computer onto a standard output by cout statement.
Syntax:

int a=10;
cout«variable;
Cout<<a;
cout<<"INDIA"

Example program: 2 Wap for Addition of two Nos.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b,c;
cout<<"enter the value of a \n";
cin>>a;
cout<<"enter the value of b \n";
cin>>b;
c=a+b;
cout<<"the value of c="<<c;
getch();
}
```

Output:
```
enter the value of a
6
enter the value of b
6
the value of c=12
```

Example program: 3 Wap for Simple interest

```
#include<iostream.h>
#include<conio.h>
void main()
{
float p,r,t,si;
clrscr();
 cout<<"enter the value of principle \n";
cin>>p;
cout<<"enter the value of rate \n";
cin»r;
cout«"enter the value of time \n";
cin>>t;
si=(p*r*t)/100;
cout <<"the result of si="<<si;
getch();
}
```

Output
```
enter the value of principle
1000
enter the value of rate
2
enter the value of time
 2
the result of si= 40.000000
```

Example program: 4 Wap for entered Hue and display line.

```
#include<iostream.h>
#include<stdio.h>
void main()
{
char name[10];
{
cout<<"enter the line\n";
gets(name);
puts(name);
}
```

Output
```
enter the line
India is great
India is great
```

Example program: 5 Wap for entered and display one character.

```
#include<iostream.h>
#include<stdio.h>
void main()
{
char name[10];
{
cout<<"enter the line\n";
gets(name);
puts(name);
}
```

Output
```
enter a character
a
```

Example program: 6 Wap for Area of circle.

```
#include<iostream.h>
#include<conio.h>
void main()
{
float pi=3.14,r,a;
Clrscr ();
cout<<"enter the radius\n";
cin>>r;
a=pi*(r*r);
cout<<"the area of circle="<<a;
getch();
}
```

Output
```
enter the radius
3
the area of circle=28.260000
```

CHAPTER- 4

DECISION CONTROL STRUCTURES

Introduction

In C++ programming the following main type of Decision statement used for implementing the decision control instruction same as c programming.

1. The if- statement
2. The if- else statement
3. The conditional operator.
4. Switch case statement

1. The if- statement :
Syntax:

```
if (boolean expression )
statement;
```

Example:
If (a>b)
printf("a is greater then b”);
here, we given the condition that if **a** is greater then **b** that time the **a** is greater then **b** will be displayed on the screen.

2. The if- else statement:

```
Syntax:
    if (condition)
                statement1;
        else
                statement2;
```

These statements execute statement1 when is condition true and also execute statement2 when is condition false.

Example:

```
if (a>b)
        cout<<”a is greatest";
else
        cout<<" b is greatest”;
```

here, if a>b then print a is greatest statement and if condition is false then it will print b is greatest. In use nested if we use another if in under of if.

```
Syntax:
        (condition1)
        statement1;
else
        if (condition2)
        statement2;
else
        statement3;
```

Example:

```
if(a>b)
        cout<<" a is greter than b";
else
else if(a==b)
        cout<<" a is eqaul to b";
else
        cout<<" b is greater than a";
```

here, if a>b, in true it will print a is greater than b else check again the a = = b , in true it will print a is equal to b else print b is greater than a.

3. The conditional operator

Example:

```
syntax:
        expression1?expression 2:expression3
```

```
int a,c;
c=(a>8?3:6);
```

here, this statement will store 3 in c if a is greater then 8 otherwise it will store 6 in c.

```
Syntax:
Switch(variable)
{
        case constant 1: statement- 1;
        break;
        case constant n: statement-n;
        break;
        default: statement
}
```

Example program 1 : Wap for greatest among two Nos.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
cout<<"enter the value of a\n";
cin>>a;
cout<<"enter the value of b\n";
cin>>b;
if(a>b)
cout<<" a is greater than b";
cout<<"b is greater than a";
getch ();
}
```

Output

```
enter the value of a
5
enter the value of b
6
b is greter than a
```

Example program 2: Wap for greatest among three Nos.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
cout<<"enter the value of a\n";
cin>>a;
cout<<"enter the value of b\n";
cin>>b;
cout<<"enter the value of c\n";
cin>>c;
if ((a>b)&(a>c))
cout<<"a is greater";
else
if (b>c)
cout<<"b is greater";
else
cout<<"c is greater";
getch();
}
```

Output

```
enter the value of a
1
enter the value of b
2
enter the value of c
3
c is greter
```

Example program 3 : Wap for leap year.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int year;
cout<<"enter the year\n"
cin>>year;
if (year%4==0)
cout<<"the year is leap year";
else
cout<<" the year is not leap year";
getch();
}
```

Output

```
enter the year
2000
the year is leap year
```

Example program 4 : Wap for display seven day in a week.

```
#include<iostrearn.h>
#include<conio.h>
void main()
{
cout<<"enter the choice\n";
cin>>ch;
switch(ch)
{
case 1 :cout<<"Sunday";
break;
case 2 : cout<<"Monday";
break;
case 3 : cout<<"Tuesday";
break; ;
case 4 : cout<<"Wednesday";
break;
case 5 : cout<<"Thursday";
break;
case 6 : cout<<"Friday";
break;
case 7 : cout<<"Saturday";
break;
default :cout<<"Invalid choice");
getch();
}
```

Output

```
Enter the choice
6
Friday
```

CHAPTER- 5

LOOP CONTROL STRUCTURES

Introduction

1. **while Statement**
2. **do-while Statement**
3. **for Statement**

In every loop three things are main- initialization of counter variable, condition, counter variable (increment/decrement purpose) .

1. while Statement

These loop is test-Do type of loop, means first it check the condition and then execute the statement.

```
Syntax:
initialization of variable;
while(condition)
{
statement;
counter variable;
};
Example: Print SARVA word 10 times.
Int i=0;
while(i<10)
{
cout<<"SARVA\n";
i++;
}
```

2. do while Statement

This loop is Do-test type of loop, means first it executes the statement and then check the condition.

```
Syntax:
                initialization of variable;

                do {
                                 statement;
                                 counter variable;
                }while(condition);
```

Example: Print SARVA word 10 times.

```
Int i=0;
        do {
cout<<"SARVA\n";
i++;
}while(i<9);
```

3. For Statement

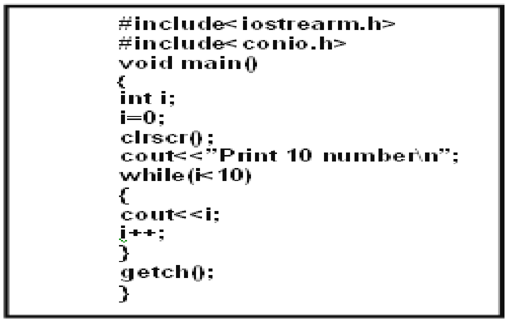
This loop is test-Do type of loop, moans first it checks the condition and then executes the statement.

```
Syntax:-
     for(initialization;condition;increment/decrement)
     {
               statement;
     }
```

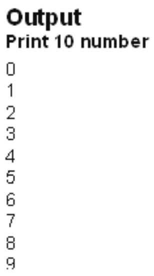
Example: Print ARJUN word 10 times.

```
int I;
   For(i=0;<10;i++)
  {
      cout<<"SARVA";
  }
```

Example program 1: Wap to print 10 Nos using while loop.

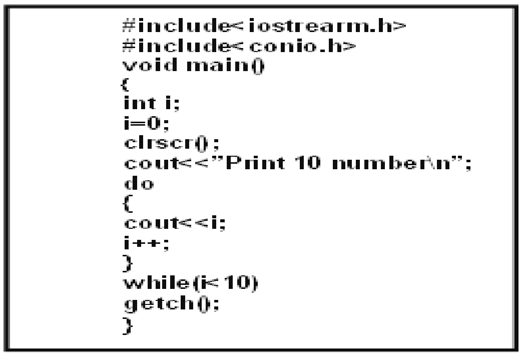


```
#include<iostrearm.h>
#include<conio.h>
void main()
{
int i;
i=0;
clrscr();
cout<<”Print 10 number\n”;
while(i<10)
{
cout<<i;
i++;
}
getch();
}
```

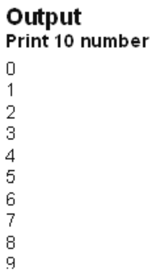


```
Output
Print 10 number
0
1
2
3
4
5
6
7
8
9
```

Example program 2: Wap to print 10 Nos using do-while loop.

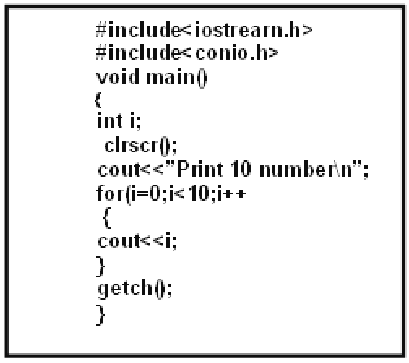


```
#include<iostrearm.h>
#include<conio.h>
void main()
{
int i;
i=0;
clrscr();
cout<<”Print 10 number\n”;
do
{
cout<<i;
i++;
}
while(i<10)
getch();
}
```

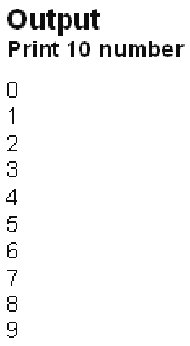


```
Output
Print 10 number
0
1
2
3
4
5
6
7
8
9
```

Example program 3 : Wap to print 10 Nos using for loop.



```
#include<iostrearn.h>
#include<conio.h>
void main()
{
int i;
 clrscr();
cout<<”Print 10 number\n”;
for(i=0;i<10;i++
 {
cout<<i;
}
getch();
}
```



```
Output
Print 10 number
0
1
2
3
4
5
6
7
8
9
```

CHAPTER- 6

FUNCTION

Introduction

The self contain program is called function.

For create the function the following three major steps are necessary**:-**

1. **Function**
2. **Function definition**
3. **Function calling**

1. **Function Prototype**

   In these, we define the type of function.
   As per the data type we can define function also.
   (int function, float function. double function, char function and void function.)
   In first four type of function return the value but void function not return void nothing.As per the use argument in function there are following type without argument. In these we do not use any argument.

Example:

```
void sarva();
```

With Argument

In these we use any argument.

Example:

```
void sarva(int a, int  b);
```

With Argument and return

In these we use any argument with return type.

Example:

```
void sarva(int a);
return(a);
```

2. **Function definition**

   In these we write the statement which we want to execute by the function.

Example:

```
void sarva()
{
        cout<<"hallo SARVA";
}
```

3. **Function calling**

   In these we execute the function definition where we write these function.

Example program 1:
Wap for without argument function.

```
    #include<iostream.h>
    #include<conio.h>
    void sarva();
    void mahakal();
    void shiva();
    void nirmala();
    void main()
    {
    sarva();
    mahakal();
    shiva();
    nirmala()
    getch();
    }
void sarva() //function defination
             {
 cout<<"I am SARVA function\n";
 }
 void mahakal()
 {cout<<"I am MAHAKAL function\n";
 }
 void nirmala()
 {
 cout<<"I am NIRMALA function\n";
 }
 void shiva()
 {
 cout<<"I am SHIVA function\n";
 getch();
 }
```

Output

```
WE ARE IN MAIN FUNCTION
I am ARJUN function
I am MAHAKAL function
I am SHIVA function
I am NIRMALA function
```

Example program 2:
Wap for with argument function.

```
#include<stdio.h>
#include<conio.h>
void addition(int a,int b);
void main()
{
int a,b;
clrscr();
printf(" enter the value of a\n");
scanf("%d",&a);
prirtf(" enter the value of b\n");
scanf("%d",&b);
addition(a,b);
getch();
}
void addition(int a.int b)
{
int c;
c=a+b;
printf("the result of addition=%d",c);
}
getch();
}
```

Output

```
enter the value of a
1
enter the value of b
2
the result of addition=3
```

Example program 3:
Wap for argument with return type function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b;
clrscr();
cout<<(" enter the value of a\n");
cin>>a;
cout<<(" enter the value of b\n");
cin>>b;
addition(a,b);
getch();
}
int addition(int a,.int b)
{
int c;
c=a+b;
cout<<"the result of addition="<<c);
}
```

Output

```
Enter the value of a
1
enter the value of b
2
the result =3
```

CHAPTER- 7

ARRAY

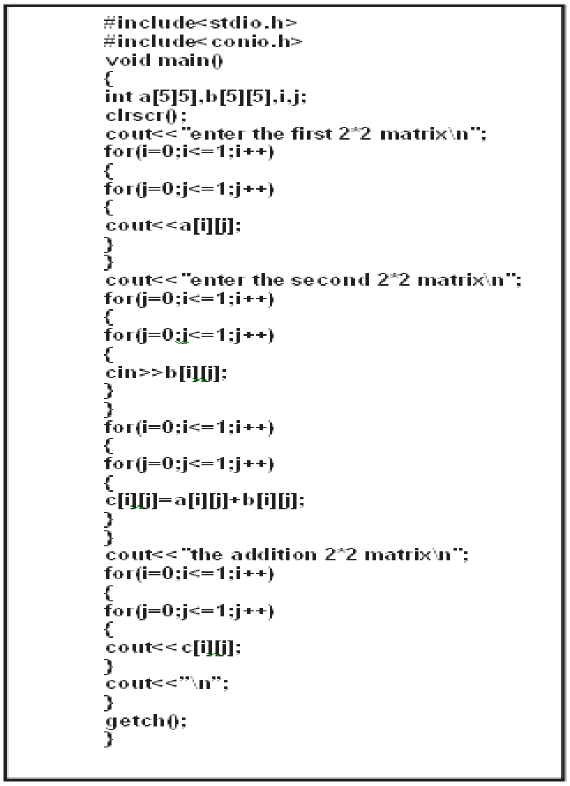
Introduction

Array is collection of same type of data.

Declaration of Array

1. **One dimensional Array:**
   In general definition one dimensional of array data store in one row or column.
   data type variable[size of array];
   ***Example*: int a[10];**
2. **Two dimensional Array:** n general definition two dimensional of array data store in one row or column. *data type variable[size of array][size of array];*

Example program 1: Wap for addition of two matrix.

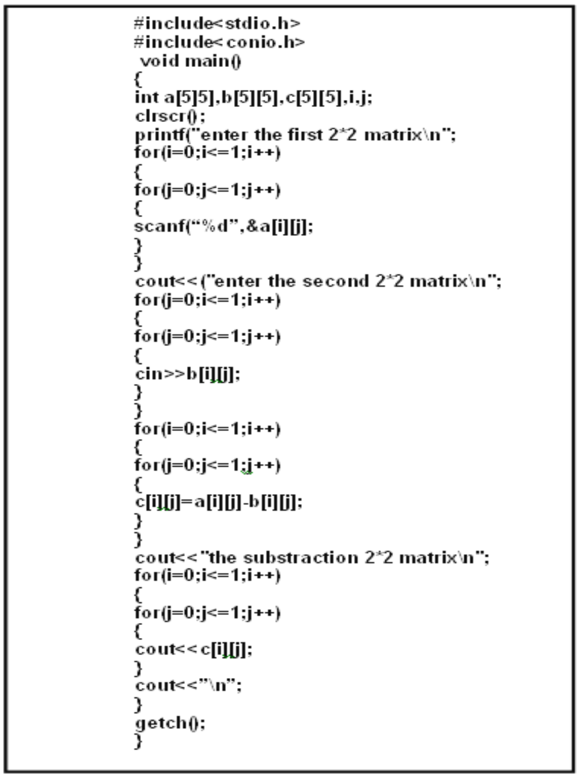


```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5]5],b[5][5],i,j;
clrscr();
cout<<"enter the first 2*2 matrix\n";
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
cout<<a[i][j];
}
}
cout<<"enter the second 2*2 matrix\n";
for(j=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
cin>>b[i][j];
}
}
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}
cout<<"the addition 2*2 matrix\n";
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
cout<<c[i][j];
}
cout<<"\n";
}
getch();
}
```

Output

```
enter the first 2*2 matrix
1
1
1
1
enter the second 2*2 matrix
1
1
1
1
the addition 2*2 matrix
2     2
2     2
```

Example program 2: Wap for subtraction of two matrix



```
#include<stdio.h>
#include<conio.h>
 void main()
{
int a[5]5],b[5][5],c[5][5],i,j;
clrscr();
printf("enter the first 2*2 matrix\n";
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
scanf("%d",&a[i][j];
}
}
cout<<("enter the second 2*2 matrix\n";
for(j=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
cin>>b[i][j];
}
}
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
c[i][j]=a[i][j]-b[i][j];
}
}
cout<<"the substraction 2*2 matrix\n";
for(i=0;i<=1;i++)
{
for(j=0;j<=1;j++)
{
cout<<c[i][j];
}
cout<<"\n";
}
getch();
}
```

Output

```
Enter the first 2*2 matrix
2
2
2
2
enter the second 2.2 matrix
1
1
1
1
the substraction 2*2 matrix
1   1
1   1
```

CHAPTER- 8

CLASSES

Introduction to Classes

The word class is a fundamental and powerful keyword in c++. It is significantly useful as it is used to combine the data and operations of a structure into a single entity. The class construct provides support for hiding, abstraction, encapsulation, single inheritance, multiple inheritance, polimorphism and public interface functions (methods) for passing message between objects.

Data abstraction: In **oop's,** the data abstraction is defined as a collection of data and methods (function).

Data Hiding: In C++, the class allows to declare data and methods as public private and protected group.

Data encapsulation: The internal data (the member data) of a class are first separated 'from the outside world (the defined class).

Inheritance: C++ allows a programmer to build hierarchy of classes. The basic feature of classes (parent classes of basic classes) can be passed onto the derived classes (child classes).

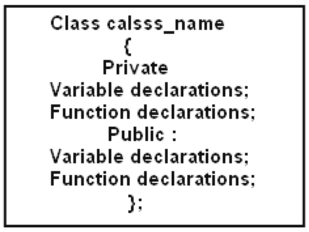
Polymorphism: In **oop**, polymorphism is defined as how to carry out different processing steps by a function having the same messages.

SPECIFYING A CLASS

A Class specification has **two parts:**

1. Class declaration.
2. Class function definition

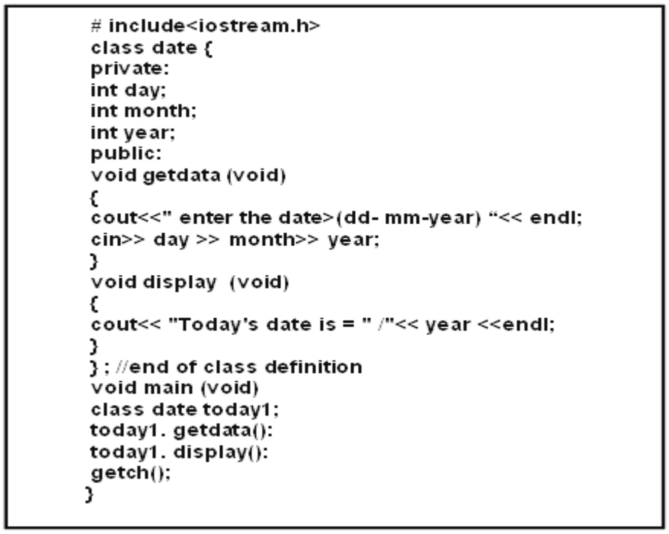
The general form of a class declaration is:



```
Class calsss_name
      {
       Private
Variable declarations;
Function declarations;
        Public :
Variable declarations;
Function declarations;
        };
```

The body of a class is enclosed within braces and terminated by a semicolon The class body contains the declaration of variables and functions. These functions and variables are called class members. They are grouped under two part namely, private and public, the class member that have been declared as In keyword private can be accessed only from within the class, IN public members can be accessed from outside the class also.

Example: Class with data and member function

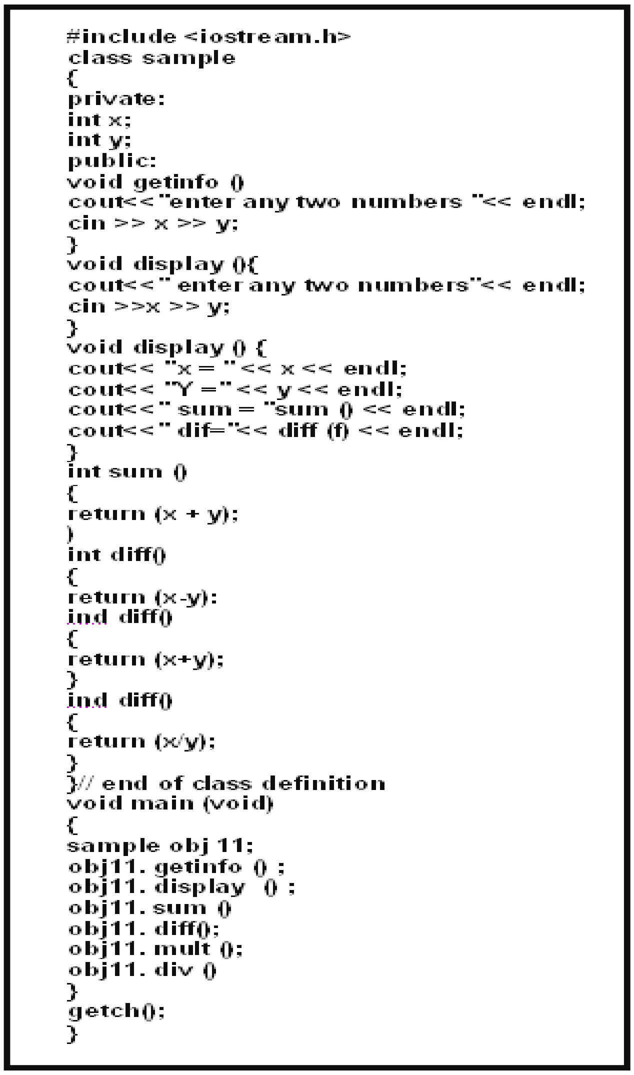


```
# include<iostream.h>
class date {
private:
int day;
int month;
int year;
public:
void getdata (void)
{
cout<<" enter the date>(dd- mm-year) “<< endl;
cin>> day >> month>> year;
}
void display  (void)
{
cout<< "Today's date is = " /"<< year <<endl;
}
}; //end of class definition
void main (void)
class date today1;
today1. getdata():
today1. display():
getch();
}
```

Output

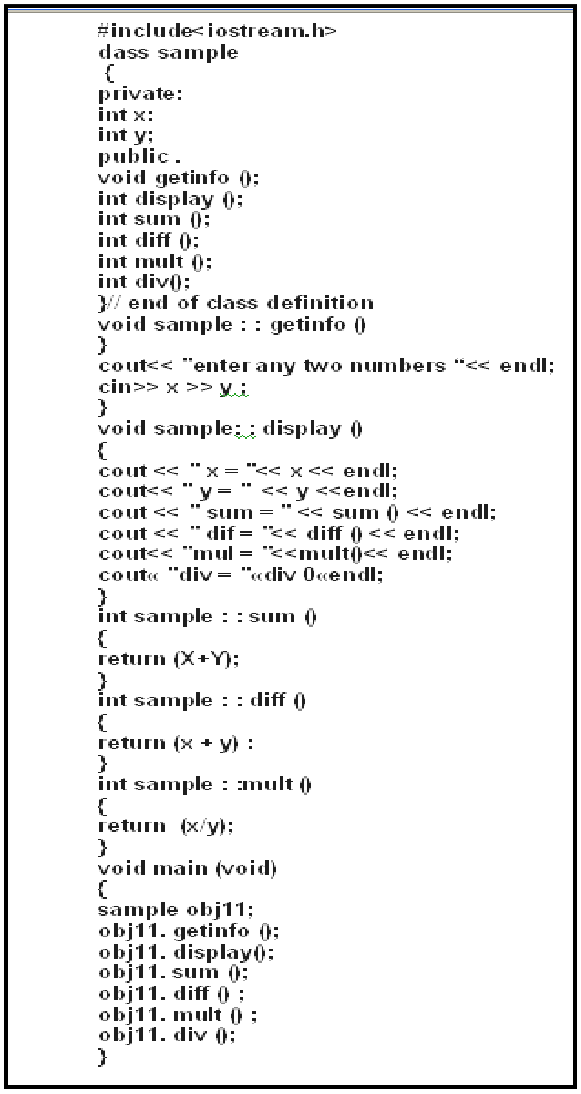
```
enter the date (dd-mm-year)
12  1  99
Today's date is = 12/1/99
```

Example: member functions are defined wit



```
#include <iostream.h>
class sample
{
private:
int x;
int y;
public:
void getinfo ()
cout<<"enter any two numbers "<< endl;
cin >> x >> y;
}
void display (){
cout<<" enter any two numbers"<< endl;
cin >>x >> y;
}
void display () {
cout<< "x = " << x << endl;
cout<< "Y =" << y << endl;
cout<<" sum = "sum () << endl;
cout<<" dif="<< diff (f) << endl;
}
int sum ()
{
return (x + y);
}
int diff()
{
return (x-y):
ind diff()
{
return (x+y);
}
ind diff()
{
return (x/y);
}
}// end of class definition
void main (void)
{
sample obj 11;
obj11. getinfo () ;
obj11. display  () ;
obj11. sum ()
obj11. diff();
obj11. mult ();
obj11. div ()
}
getch();
}
```

Example: methods are defined out of the scope of the class



```
#include<iostream.h>
dass sample
 {
private:
int x:
int y:
public .
void getinfo ();
int display ();
int sum ();
int diff ();
int mult ();
int div();
}// end of class definition
void sample : : getinfo ()
}
cout<< "enter any two numbers "<< endl;
cin>> x >> y ;
}
void sample: : display ()
{
cout << " x = "<< x << endl;
cout<< " y = " << y <<endl;
cout << " sum = " << sum () << endl;
cout << " dif = "<< diff () << endl;
cout<< "mul = "<<mult()<< endl;
cout« "div = "«div 0«endl;
}
int sample : : sum ()
{
return (X+Y);
}
int sample : : diff ()
{
return (x + y) :
}
int sample : :mult ()
{
return  (x/y);
}
void main (void)
{
sample obj11;
obj11. getinfo ();
obj11. display();
obj11. sum ();
obj11. diff () ;
obj11. mult () ;
obj11. div ();
}
```

Output

```
Enter any two numbers
1 2
x = 1
y = 2
sum = 3
dif = -1
mul = 6
div = 0
```

CHAPTER- 9

SOLVED PROGRAM

EXAMPLE 1: ***WAP TO INPUT A NO. ANDTO PRINT ITS CUBE BY POW () FUNCTION*/***

```
#include<stdio.h>
#include<conio.h>
#include<math.h>        //for pow()function math.h
void main()
{
        int a;
        clrscr();
        printf("Enter any number:");
        scanf(%d",&a);
        printf("\n The cube of the number is :%g",pow(a,3));
        /*pow(x,y)=x to the power y*/
        getch();
}
```

Output

```
Enter any number:
3
the cube of the number is: 27
```

EXAMPLE 2: ***WAP TO DISPLAY CUBEROOT OF A NO.*/***

```
#include<stdio.h>
#include<conio.h>
#indude<math.h>        //for pow()function math.h
void main()
{
        float a, y;
        clrscr();
        printf ("enter value of a");
        scanf("%f", &a);
        y=(pow(a,1.03));
        printf("\ncubroot of a is  %g ",y);
        getch();
}
```

Output

```
Enter the value of a
8
cube root of a is 2
```

EXAMPLE 3: ***WAP TO CONVERT GRAM TO KILOGRAM.***

```
#include<stdio.h>
#include<conio.h>
#include<math.h>        /for pow()function math, h is must*/
void main()
{
        float grams,kilogram;
        Clrscr();
        printf("enter grams");
        scanf("%f",& grams);
        kilogram=grams/pow(10,3);
                                /* 1kilogram=1000 grams*/
        printf("\nquantity in Quintal is =%g ",kilogram);
        getch();
        }
```

Output

```
Enter grams
1000
quantity in kilogram is =1
```

EXAMPLE 4 : ***WAP TO INCREASE AND DECRESE THE INPUT VALUE BY 1.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int k,j;
        clrscr();
        printf("Enter k");
        scanf ("%d", &k);
        k++;                /*for increment by 1*/
        printf'("\n The increment value of k is %d.",k);
        printf("\n enter j");
        j--;                /*for decrement by 1*/
        printf ("\n The decremented value of j is %d",j));
        getch();
}
```

Output

```
Enter k
2
the incremented value of k is 3
enter j
3
The decremented value of j is 2
```

EXAMPLE 5 : ***WAP TO DISPLAY AREA AND PERIMETER OF RACTANGGLE.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
        float x,y, area, perimeter;
        clrscr();
        printf("Enter two sides x, y : ");
        scanf ("%f%f", &x,&y);
        area=x*y;
        perimeter=2* (x+y);
        prntf("\nArea of rectangle is %g sqr. Units ",area);
        prntf("\n Perimeter of a rectangle is %g", perimeter);
        getch();
}
```

Output

```
Enter two sides x y :
        1 4
Area of rectangle is 8 sqr. units
Perimeter of rectangle is 12
```

EXAMPLE 6 : ***WAP TO INPUT TWO VALUES & EXCHANGE THEM (SWAPING) BY USING THIRD TEMPORARY VARIABLET.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
            int x,y,temp;
            clrscr();
            printf("Enter two values");
            scanf ("%d,%d", &x,&y);
            prntf("\nentered values are x=%d, y=%d", x,y);
            temp=x;
            x=y;
            y=temp;
            prntf("\nvalues after swaping are x=%d, y=%d", x,y);
            getch();
            }
```

Output

```
Enter two values
4  5
entered values are x=4, y=5
values after swaping x=5, y=4
```

EXAMPLE 7 : ***WAP TO FIND TOTAL EXPENSES.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
            float t,d,l, totalexp;
            clrscr();
            printf("Enter Travelling exp. In Rs.");
            scanf ("%f", &t);
            prntf("\nEntered dry cleaning exp. In Rs.")
            scanf ("%f", &d);
            prntf("\nEntered lodging and boarding exp. In Rs.")
            scanf("%f", &l);
            totalexp=t+d+l;
            prntf("\n The total exp is Rs.%g",totalexp);
            getch();

              }
```

Output

```
Enter travelling exp in Rs.
2000
Enter dry cleaning exp. In Rs.
200
Enter lodging and boarding exp. In Rs.
300
The total exp are Rs. 2500
```

EXAMPLE 8 : ***WAPTO FIND AREA OF TRAINGLE.***

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void  main()
 {
            float x,y,z,s, area;
            clrscr();i
            Printf("Enter three sides x y z :");
            scanf ("%f%f%f", &x,&y,&z);
            s=(x+y+z)/2;
            area =sqrt(s*(s-x)*(s-y)*(s-z)));
            printf("\nArea of Triangle is %g sqr. Units",area);
            getch();
        }
```

Output

```
Enter three sides x y z :
3 4 5
Area of Triangle is 6 sqr units
```

EXAMPLE 9: ***WAP TO FIND THAT ENTERED YEAR IS A LEAP YEAR OR NOT.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
            int  t,yrs;
            clrscr();
            printf("\nenter year");
            scanf("%d", &yrs);
            scanf(yrs%4=0) /* % operator will show remainder*/
            printf("\nleap year");
            else
            Printf (not a leap year");
            getch();
}
```

Output

```
enter year
2002
leap year
```

EXAMPLE 10 : ***WAP TO CALCULATE GROSS SALARY BY CONSIDERING HOUSE RENT ALLOWANCE, DEARNESS ALLOWANCE.***

```
#include<stdio.h>
#include<conio.h>
void main()
{
            Float bs,gs,da,hra;
            clrscr();
            printf("\n The basic salary ");
            scanf(%f", &bs);
            if(bs<=200)
            {
            hra=bs*10/100;
            da=bs*20/100;
            }
            else
            {
            hra=500;
            da=bs*25/100;
            }
            gs=bs+hra+da;
            printf("\n Gross salary=Rs.%g",gs);
            getch();
               }
```

Output

```
The basic salary
500
gross salary=Rs. 1125
```

EXAMPLE 11 : WAP TO INPUT NO. AND FIND OUT IT IS EVEN OR ODD NO.

```
#include<stdio.h>
#include< conio.h>
void  main()
{
              int  n,j;
              clrscr();
              printf("\n enter no.");
              scanf ("%d",&n);
              j=n%2;
              if(j==0)
              printf("even");
              else
                          printf("\n  odd");
getch ();
 }
```

Output

```
enter no.
24
even
```

EXAMPLE 12 : ***WAPTO FIND THAT THE NO. IS DIVISIBLE BY SIX OR NOT.***

```
#include<stdio.h>
#include<conio.h>
void main()
  {
          int i;
          clrscr();
          printf("\n enter i");
              scanf(%d,&i);
          if(i%6)                 /* operator will show remainder so if the
                                   is non-zero than this condition will be
                                    true and if statement will be executed
                                   */
                  printf ("\n The number is not divisible by 6");
                  else
                  printf ("\n The number is not divisible by 6");
          getch();
}
```

Output

```
enter i
36
The number is divisible by 6 .
```

EXAMPLE 13 : ***WAP TO FIND PROFIT OR LOSS OR NO PROFIT NO LOSS 1N SELLING.***

```
#include<conio.h>
void main()
  {
     float cp,sp,profit,loss;
     clrscr();
     printf("\n Enter cp: ");       /*input cost price*/
     scanf("%g",&cp);
     printf("\n Enter sp: ");       /* enter selling price*/
     scanf("%g",&sp);
     if (sp>cp)
     {
            profit = sp-cp;
            printf("\n The profit is Rs. %g"-profit);
     }
     else
     {
            if (cp==sp)
                      printf("\n The loss is Rs.%g",loss);
            else
            }
                       loss =cp-sp;
                       printf("\n The loss is Rs.%g",loss);
            }
            }
       getch();
         }
```

Output

```
Enter cp:
200
Enter sp:
90
The toss is Rs.110
```

Example 14 : ***WAP TO GENERATE THE TABLE OF NUMBER ENTERED BY USER.***

```
#include<stdio.h>
#include<conio.h>
int table (int, int);
void main()
{     int i, n;
clrscr() ;
printf("Enter any integer") ;
scanf("%d", &n) ;
for(i =1 ; i<=10 ; i++)
printf("%d x %d = %d\n", n, i, table (n, i)) ;
 }
int table (int n, int i)
{
return (n)
}
return (table (n, i -1) + n);
}
}
getch();
}
```

Output

```
Enter the number2
2x1 =2
2x2=4
2x3=6
2x4=8
2x5=10
2x6=12
2x7=14
2x 8=16
2x9=18
2x10 = 20
```

Example 15: ***WAP FOR STRING COPY***

```
#include<conio.h>
#include<stdio.h>
main()
{
 char* sourse ="arjun";
            char*des="chandel";
            int i;
            clrscr();
            printf("Enrter the siring");
            gets(sourse);
            for(i=0;sourse[i]!='\0';i++)
            des[i]='\0';
            printf("The string is; %s",des);
            printf("\nPress any key to continue; ");
            getch();
             }
```