

➡ First year: 1st sem

Theory:

▶ Subject title: Programming for problem solving

Subject code:20ES03

Internalexam:30

External exam:70

➡ Total:100

Lab:

▶ Lab Title: programming for problem solvinglaboratory

Lab code:20ES1252

Internalexam:30

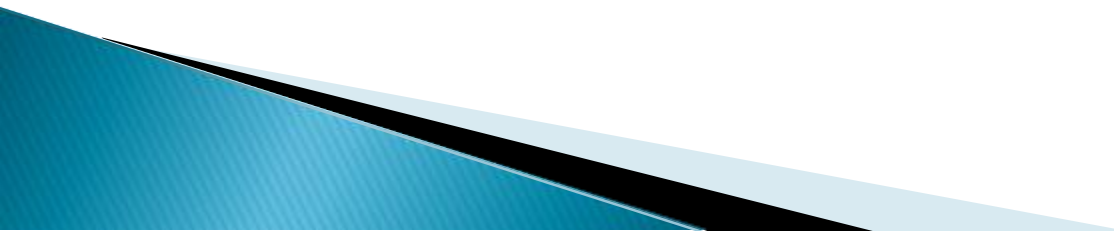
External exam:70

➡ Total:100

COURSE OUTCOMES

Upon successful completion of the course, the student will be able to:

CO1	Understand the different types of problem solving approaches
CO2	Apply the selections, loops, arrays, and string concepts in C to solve problems.
CO3	Apply functions and pointer concepts in C to solve problems.
CO4	Solve problems using enum, structures, unions, and file handling functions.

- ▶ Pseudo code
 - ▶ Algorithm
 - ▶ Statement (or) instruction
 - ▶ Program
 - ▶ Flowchart
 - ▶ compiler
- 

Pseudo code

Ex: Addition of two numbers

Start

Read the variables

Add the numbers together to the 3rd variable

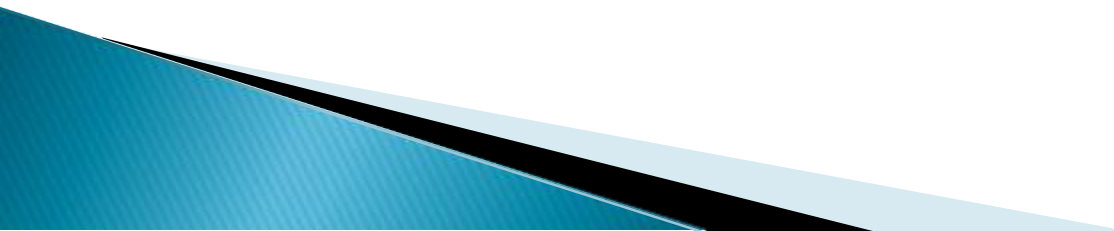
Print result

End

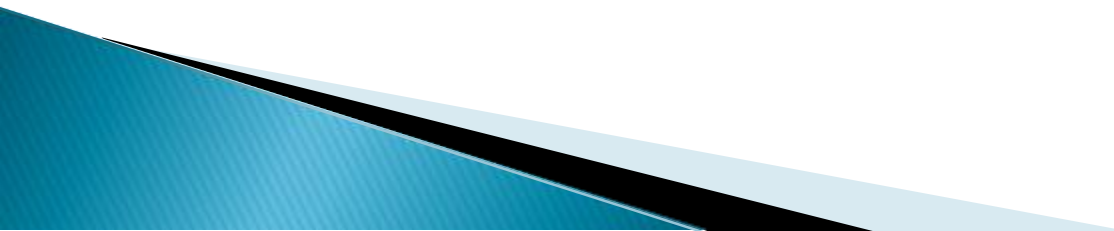




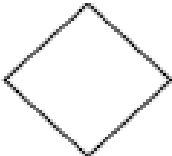




Algorithm:

Characteristics:

- ▶ Finiteness
 - ▶ Definiteness
 - ▶ Unambiguous
 - ▶ Input
 - ▶ Output:
- 

Key features:

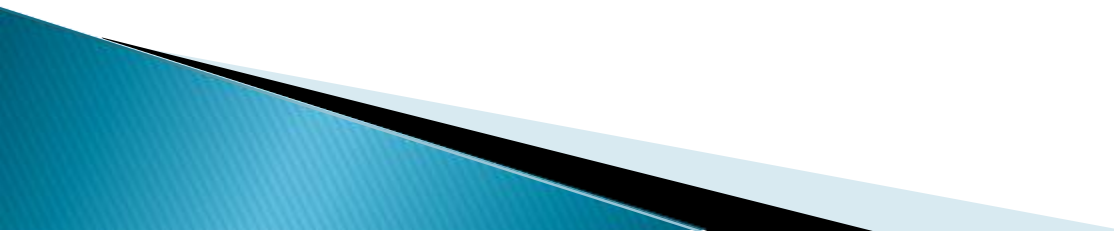
- ▶ Sequence
 - ▶ Selection
 - ▶ Repetition
- 

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

History of c language

Year	Language	Developed by
1960	ALGOL	International committee
1963	Cpl	Cambridge university
1967	BCPL	Martin Richards
1970	B	Ken Thomson
1972	C	Dennis Ritchie

Features of language

- ▶ Portability
 - ▶ Design efficiency
 - ▶ Procedure oriented
 - ▶ Middle level language
 - ▶ Important to learn new language
 - ▶ Case sensitive
- 

Structure of a c program



Documentation section

Link section

Definition section

Global declaration section

main () Function section

{

Declaration part

Executable part

}

Subprogram section

Function 1

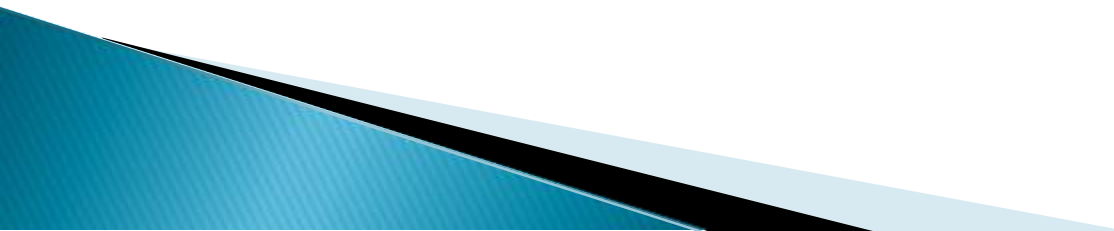
Function 2

.....

.....

Function n

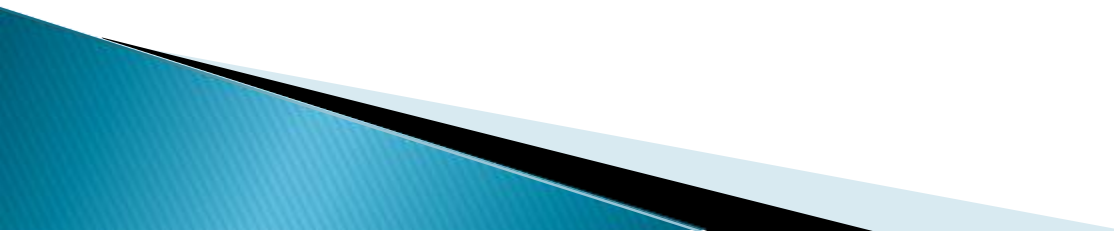
(User defined functions)

- ▶ **Documentation Section:** The documents section contains a set of comment lines giving the name of the program and other details. They are two types of comment lines
 - ▶ 1.single line comment
 - ▶ 2.multi line comment
- 

The single line comment start with // and continue until the end of the line

Ex:

```
#include<stdio.h>
{
    Void main()
    {
        printf("hello");
        //    printf("hai");
        getch();
    }
}
```



2.The multi line comment **start with /* and ends with */.**

All the words and statements written between the symbols

Ex:

/* write a c program to print hello world */

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    Printf(“hello world”);
```

```
    getch();
```

```
}
```



Link Section:

- ▶ The link section provides **instructions to the compiler** to link the functions from system library.
- ▶ C program depends on some header files Each header file extension is '.h'.
- ▶ The header files are included at the beginning of the program in the C language.

Ex:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
#include<math.h>
```

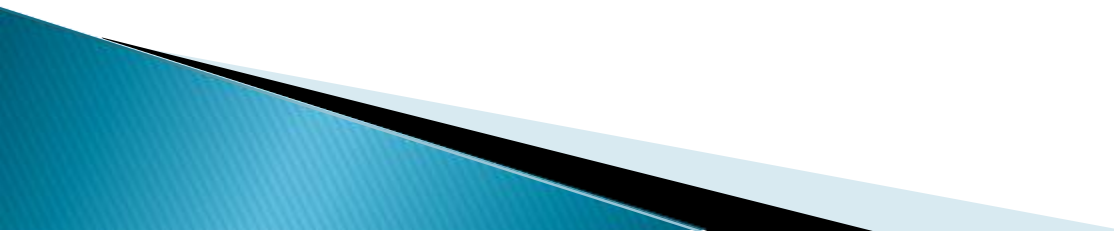
```
#include<string.h>
```



- ▶ **Definition Section :** The definition section contains the all symbolic constants.
- ▶ Ex: `#define pi 3.14`
- ▶ `#define pow 2`
- ▶ **Global Declaration Section:** There are some variables and those variables are declared in this section that is outside of all functions
- ▶ .

- ▶ **main() function:**

- ▶ Every C program must have one main() function section
 - ▶ This function returns nothing and takes no arguments.
 - ▶ The program contains statements that are enclosed within the braces. The opening braces { and closing braces }.
- In these two braces main() function contains two parts, declaration and executable part. It is user defined function.

- ▶ Declaration part:
 - ▶ Executable part:
 - ▶ **Subprogram Section :**
 - ▶ The subprogram section contains all the user defined functions that are called in the main function.
 - ▶ Number functions are included in this section
- 

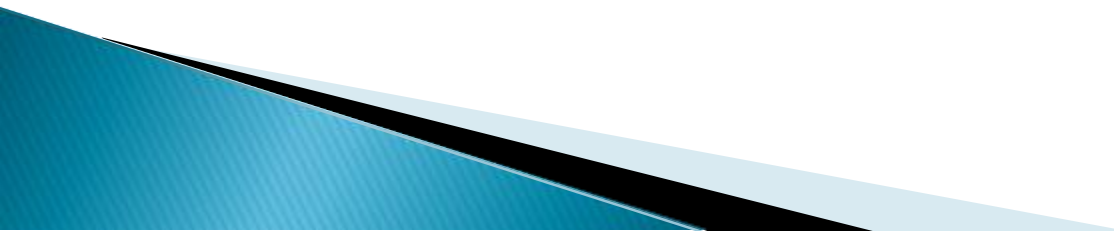
▶ Example:

program to find the area of circle /*Documentation
Section */

#include<stdio.h> /*link section*/

#include<conio.h> /*link section*/

#define PI 3.14 /*definition section*/



```
float area; /*global declaration section*/

void main()

{

    float r;                /*declaration part*/

    /*executable part starts here*/

    printf("Enter the radius of the circle\n");

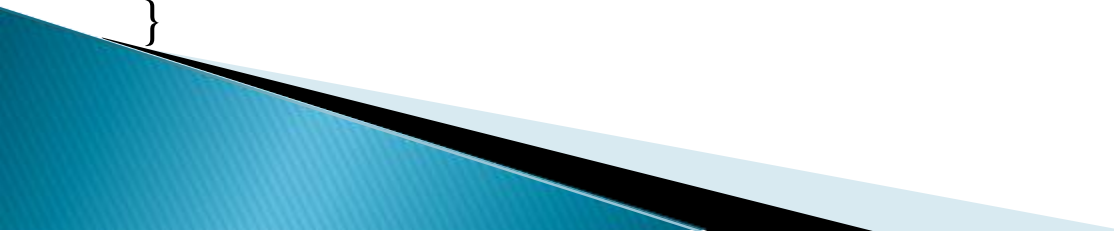
    scanf("%f",&r);

    area=PI*r*r;

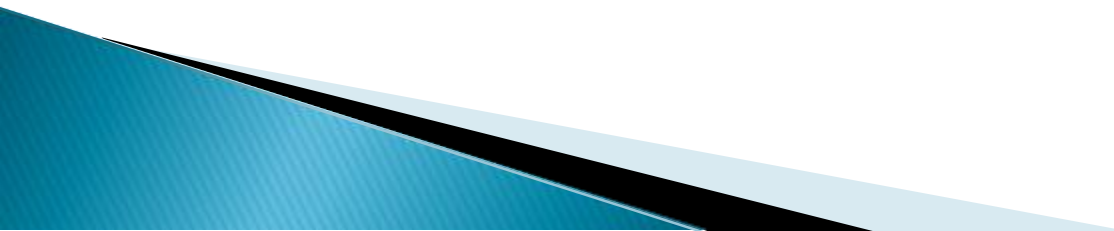
    printf("Area of the circle=%f",area);

    getch();

}
```



Data types

- ▶ Primary data types: **int, float , char**
 - ▶ Derived data types: **Array , pointers , functions**
 - ▶ User defined data types: **structures , unions, enum ,
typedef**
- 

Primary data types

- ▶ Integer types
- ▶ Float types
- ▶ Character type

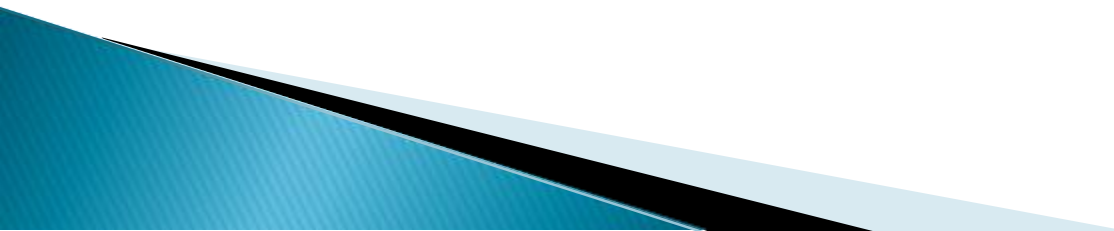
Integer types:

- ▶ The c programming language offers three different integer data types they are int ,short int and long int.
- ▶ The difference between these three integers are the number bytes to occupy and the range of values

Integer types

Type	size	Range
Int	2	-32768 To 32767
Short int	2	-32768 To 32767
Long int	4	-2147483848 To 2147483847
Unsigned int	2	0 To 65535
Unsigned short int	2	0 To 65535
Unsigned long int	4	0 To 4294967295

Float types:

- ▶ The c programming language offers three different float types they are **float** ,**double** and **long double**.
 - ▶ The difference between these three floats are the **number of bytes** to occupy and the **range of values**
- 

Float types

Types	Size	Range
Float	4	$3.4\text{E}-38$ TO $3.4\text{E}+38$
Double	8	$1.7\text{E}-308$ TO $1.7\text{E}+308$
Long double	10	$3.4\text{E}-4932$ TO $3.4\text{E}+4932$

▶ Character type:

- ▶ The c programming language offers two different character data types they are **signed char** and **unsigned char**.
- ▶ The difference between these two character types are the **range of values only** (number of bytes is the same)

Character types

Type	Size	Range
Signed char	1	-128 TO 127
Unsigned char	1	0 TO 255

Variable

- ▶ Declaration of variables

- ▶ Syntax:

- ▶ Data type variable names;

- ▶ Ex:

- ▶ int a,b,c;

- ▶ float height,temp;

- ▶ char ch;

▶ Ex: `int a,b;` 2 ,3

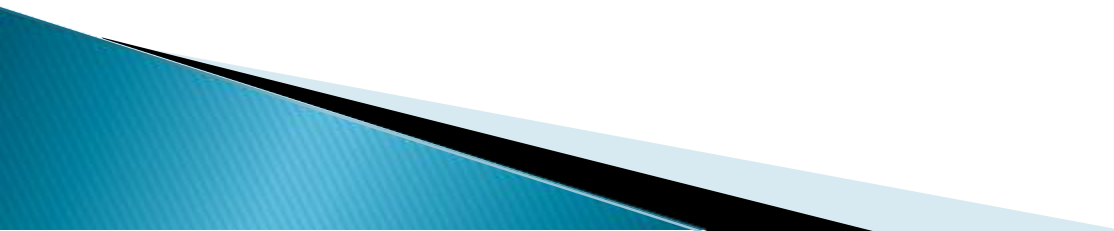


- `&a=1024`
- `&b=1025`

C Tokens

⇒ c has six types of tokens are available they are.

- ▶ 1.keywords
- ▶ 2. Identifiers
- ▶ 3.Constants
- ▶ 4.Strings
- ▶ 5.Special characters
- ▶ 6.Operators

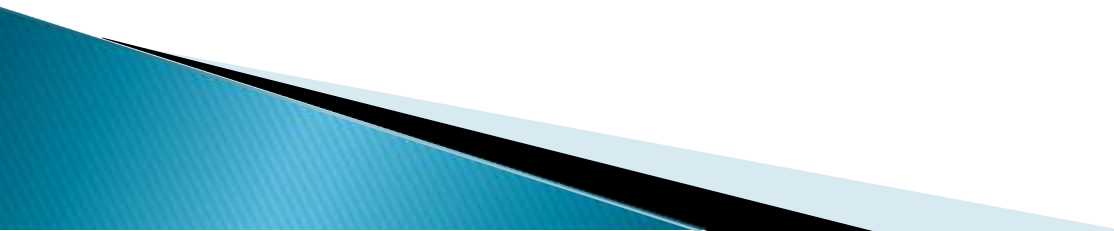
- ▶ **Keywords:**
 - ▶ Keywords are the **basic building blocks** for program statements
 - ▶ Every c statement is classified as either **a keyword** or an **identifier**
 - ▶ All keywords have **fixed meanings** and these meaning cannot be changed.
 - ▶ All keywords must be written in **lowercase letters only**.
 - ▶ Total **32 keywords** are available in c they are.
- 

Keywords

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

identifiers

- ▶ Identifiers refers to the names. These are the user defined names and consists of letters, digits and sequence of characters
- ▶ The identifiers allows both uppercase and lowercase letters but lowercase letters are commonly used

- ▶ The under score character is also permitted in identifier ,in general the underscore character is used as a **link between two long identifiers**
 - ▶ **Rules for constructing the identifiers :**
 - ▶ First character must be an **alphabet**
 - ▶ Identifiers consist of only **letters , digits and underscore character**
 - ▶ Special characters are not allowed except underscore character
 - ▶ Keywords cannot be used as a identifiers
- 

Constants

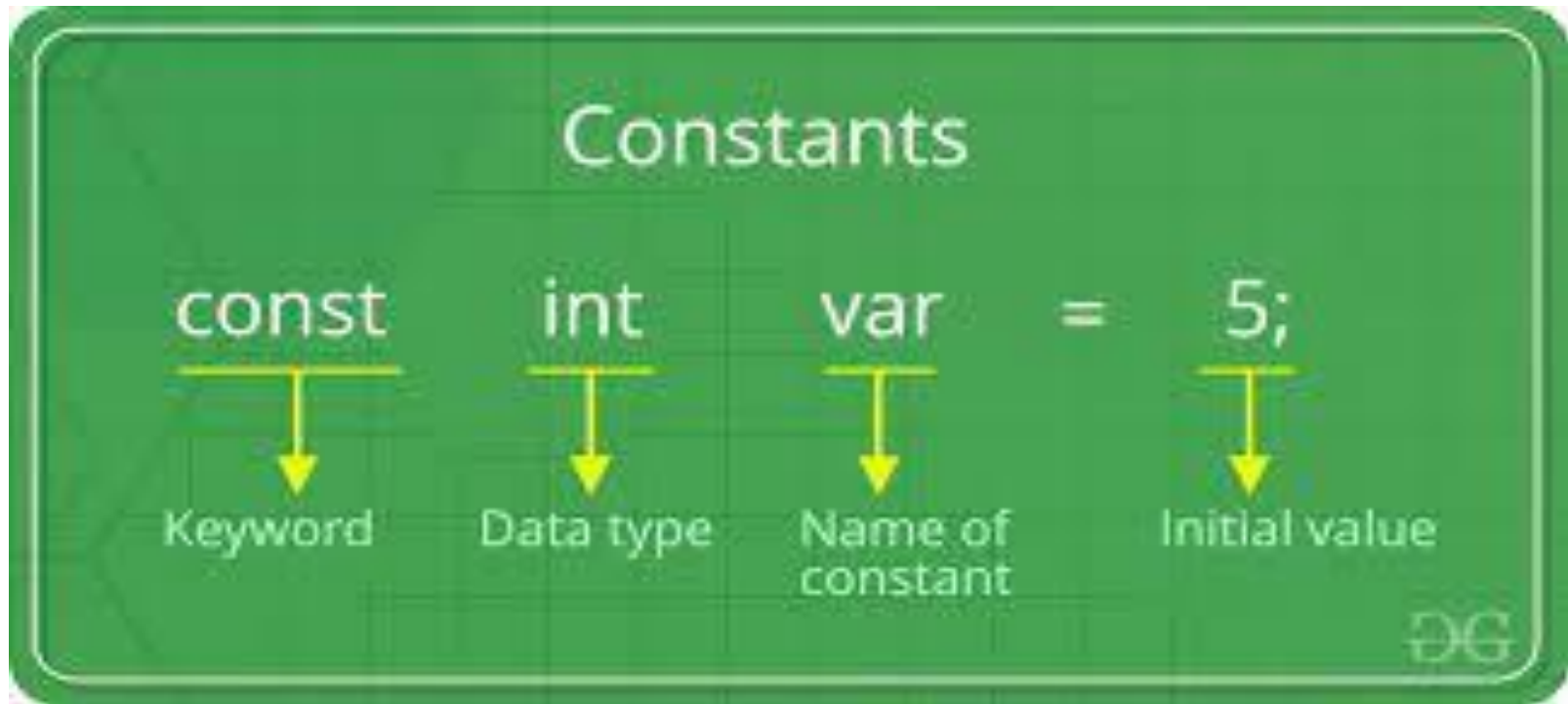
Constants in c are the **fixed values** and these values cannot be changed during execution of a program

➡ Declaration of constant variable:

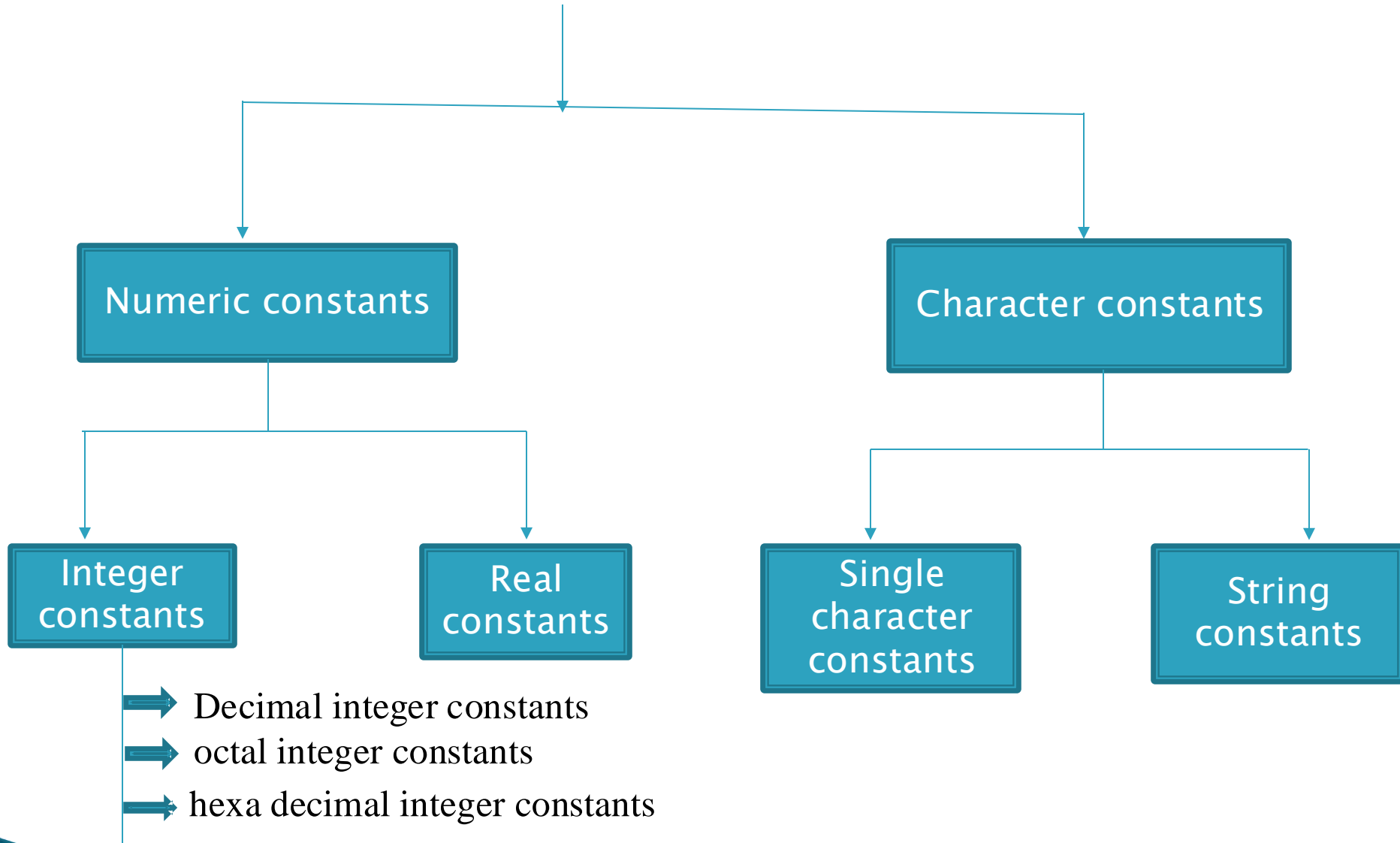
Syntax:

```
const data type variable = value;
```

Ex:



Constants



1.Integer constants:

Integer constants refers to a **sequence of digits**.

⇒ **Types:**

▶ **Decimal integer constant:**

Decimal integer constant consist of a set of digits from **0** to **9**, preceded by an optional - or + sign

Ex: `const int a=12, b=10;`

Note: spaces , commas and non digit characters are not permitted between the digits

Ex: `const int a=17 750 , b=20,000, c=$1000`



Ex:

```
void main()
{
    const int a=12, b=10;
    int c;
    c=a+b;
    printf("c=%d",c);
    getch();
}
```

o/p:22



2. Octal integer constant:

- ▶ An octal integer constant consist set of digits from 0 to 7 with leading 0.

Ex:

```
const int a=037,b=043;
```

Ex:

```
void main()
```

```
{
```

```
    const int a=037,b=043;
```

```
    int c;
```

```
    c=a+b;
```

```
    printf("c=%d",c);
```

```
    getch();
```

```
}
```



3.Hexadecimal integer constant:

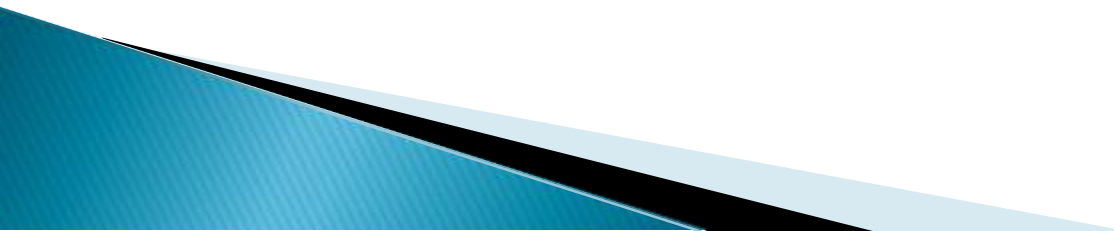
- ▶ An hexadecimal integer constant consist of set of digits from 0 to F with leading '0X'

Ex:

```
const int a=0X17,b=0XB;
```

Ex:

```
void main()  
{  
    const int a=0X17,b=0XB;  
    int c;  
    c=a+b;  
    printf("c=%d",c);  
    getch();  
}
```



2.Real constants:

- ▶ The real constants are generally used for the quantities such as distance ,height , temperature and price.
- ▶ These quantities are represented by the numbers these numbers contains fractional parts like 2.5 and 3.2 such numbers are called real or floating point constants

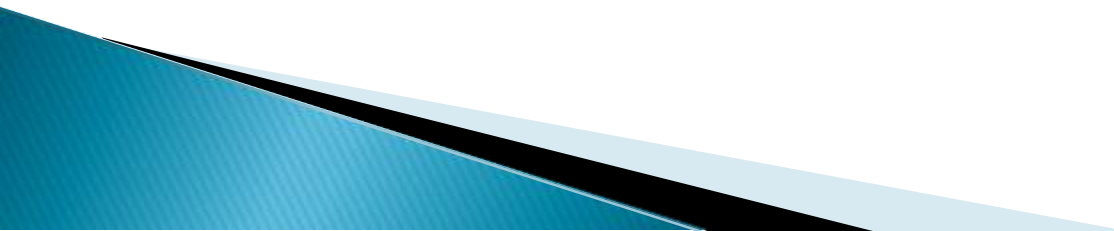
Ex:

```
const float a=2.5 ,b=3.2 ;
```

Ex:

```
void main()
{
    const float a=2.5,b=3.2;
    float c;
    c=a+b;
    printf("c=%f",c);
    getch();
}
```

o/p:c=5.70



Character constant

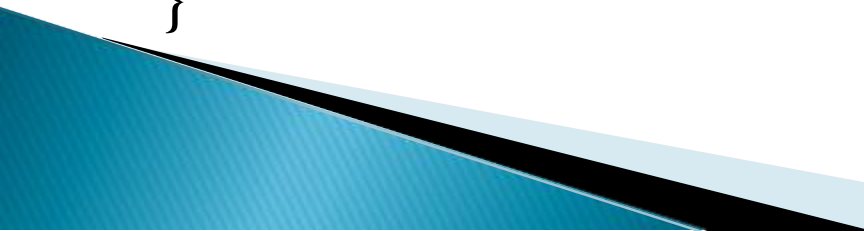
- ▶ 1.Single character constant
- ▶ 2.String constant
- ▶ **Single character constant:**
- ▶ A single character constant contains **a single character** enclosed within a pair of **single quotation mark**
- ▶ **Ex:** '5' 'b' 'c' '?'
- ▶ **Note:** the character constant **'5'** is not same as 5

- ▶ The character constant have integer values known as ASCII values
- ▶ **Ex:** `printf(“ % d”, 'a');` o/p: 97
- ▶ `printf(“% c”, '97');` o/p: a

- ▶ **String constant:**
- ▶ A string constant is a **sequence of characters** enclosed within in a pair of **double quotation mark** .
- ▶ The string may be a letters ,numbers ,special characters and blank space
- ▶ **Ex:** "hello" "1987" "wel come" "!.....?" "5+3"
- ▶ **Note:** character constant ' b ' is not same as the string constant " b "

▶ Ex1:

```
#include<stdio.h>
#include<conio.h>
#define max 10
void main()
{
    const int a=4;
    int c;
    clrscr();
    c=max+a;
    printf("c=%d",c);
    getch();
}
```



▶ Ex2:

```
#include<stdio.h>
#include<conio.h>
#define max 10
void main()
{
    const int  cmax=5;
    int a,b;
    clrscr();
    a=max+5;
    b=cmax;
    printf("a=%d\nb=%d",a,b);
    getch();
}
```

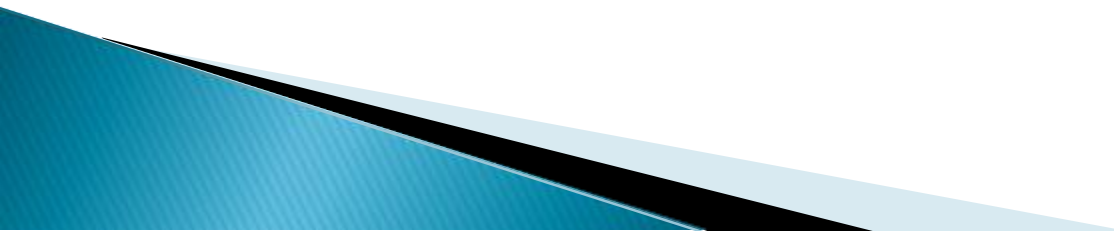
► Ex3:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    const int a=2,b=3;
    clrscr();
    add=a+b;
    printf("add=%d",add);
    getch()
}
```

Operators

- ▶ **1.Arithmetic operators:** perform numerical calculations

Operator	meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	multiplication
/	division
%	Modulus or reminder

- ▶ + operator perform **addition operation** of an two operands
 - ▶ - operator perform **subtraction operation** of an two operands
 - ▶ * operator perform **multiplication operation** of an two operands
 - ▶ / truncates any **fractional part**
 - ▶ % produce the **remainder** of integer division
- 

▶ Ex:

▶ $a+b$ $a-b$ $a*b$ a/b $a\%b$

▶ Here a and b variables or operands and $+$ $-$ $*$ $/$ $\%$ are the operators

▶ The modulus operator cannot be used on floating point data

categories

- ▶ Integer arithmetic=> $c=a+b$

$$c=6$$

if $a=4$ $b=2$

- ▶ Real arithmetic => $x=a/b$

$$x=0.8571$$

if $a=6.0$ $b=7.0$

- ▶ Mixed mode arithmetic=> $z=a/b$

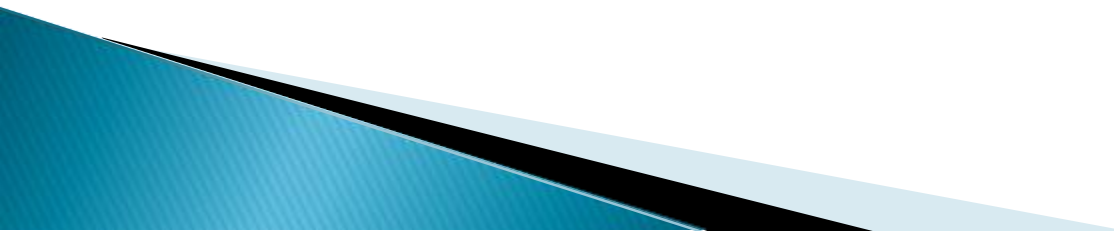
$$z=1.5$$

if $a=15$ $b=10.0$

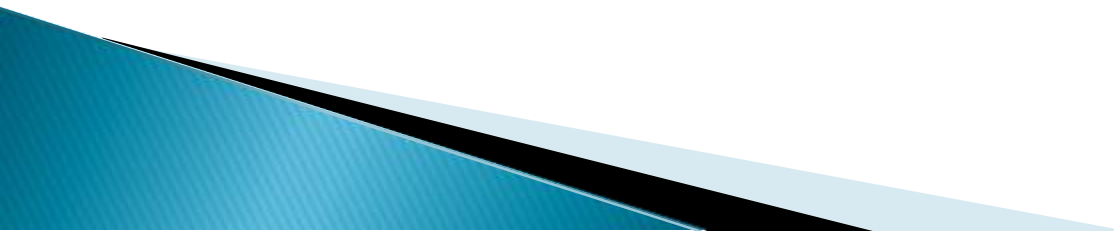

```
▶ Ex: void main()  
  
    {  
  
        const int a=10,b=5;  
  
        printf("%d",a+b);  
  
        printf("%d",a-b);  
  
        printf("%d",a*b);  
  
        printf("%d",a/b);  
  
        printf("%d",a%b);  
  
        getch();  
  
    }
```

▶ 2.Relational operators :

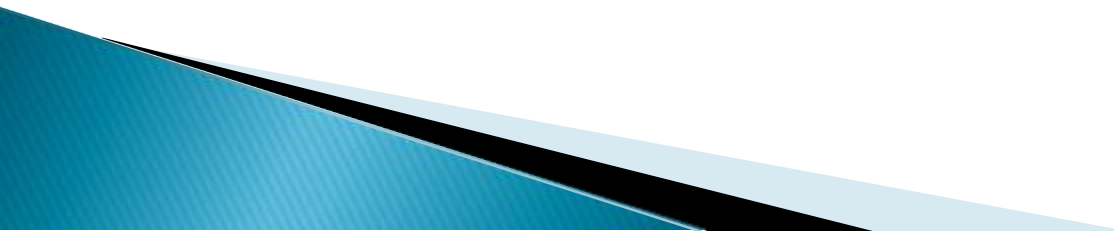
▶ <u>operator</u>	<u>meaning</u>
▶ <	is lesser than
▶ <=	lesser than or equal
▶ >	greater than
▶ >=	greater than or equal
▶ ==	is equal
▶ !=	is not equal

- ▶ ‘<’ operator checks whether the **first operand is lesser than the second operand**. If so, it returns true. Otherwise it returns false.
 - ▶ Ex: **6<5** will return false.
 - ▶ ‘<=’ operator checks whether the **first operand is lesser than or equal to the second operand**. If so, it returns true. Otherwise it returns false.
 - ▶ Ex: **5<=5** it returns true.
- 

- ▶ ‘>’ operator checks whether the first operand is greater than the second operand. If so, it returns true. Otherwise it returns false. For example, $6 > 5$ will return true
- ▶ ‘>=’ operator checks whether the first operand is greater than or equal to the second operand. If so, it returns true. Otherwise it returns false. For example, $5 \geq 5$ will return true.

- ▶ ‘==’ operator checks whether the **two given operands are equal or not**. If so, it returns true. Otherwise it returns false. For example, **5==5** will return true.
 - ▶ ‘!=’ operator checks whether the **two given operands are equal or not**. If not, it returns true. Otherwise it returns false. It is the exact boolean complement of the ‘==’ operator. For example, **5!=5** will return false.
- 

▶ Relational operators complements:

- ▶ $>$ is complement of $<$
 - ▶ $<$ is complement of $>$
 - ▶ $<=$ is complement of $>=$
 - ▶ $>=$ is complement of $<=$
 - ▶ $==$ is complement of $!=$
- 

▶ Ex: if a=4 b =6

▶ $a < b \Rightarrow \text{TRUE}$

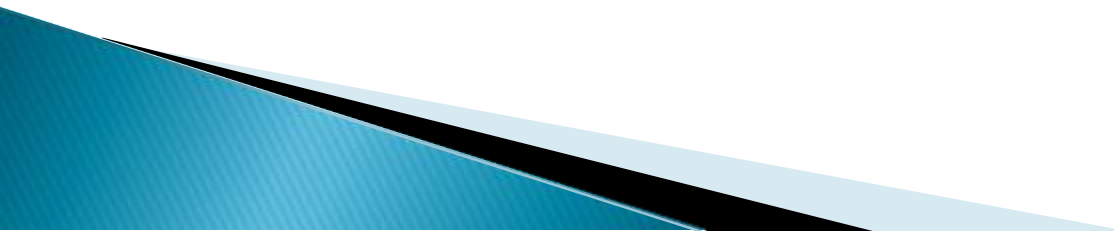
▶ $a > b \Rightarrow \text{FALSE}$

▶ $a \leq b \Rightarrow \text{FALSE}$

▶ $a \geq b \Rightarrow \text{TRUE}$

▶ $a == b \Rightarrow \text{FALSE}$

▶ $a != b \Rightarrow \text{TRUE}$



▶ Ex:

```
void main()
{
    const int a=10,b=5;
    printf("%d",a<b);
    printf("%d",a<=b);
    printf("%d",a>b);
    printf("%d",a>=b);
    printf("%d",a==b);
    printf("%d",a!=b);
    getch();
}
```


▶ 3.Logical operators:

▶ <u>Operator</u>	<u>meaning</u>
▶ &&	logical AND
▶ 	logical OR
▶ !	Logical NOT

- ▶ **Logical AND:** The '**&&**' operator returns true when both the conditions are satisfied. Otherwise it returns false. For example, **a && b** returns true when both a and b are true (i.e. non-zero).

- ▶ **Logical OR:** The ‘||’ operator returns true when one (or both) of the conditions in consideration is satisfied. Otherwise it returns false. For example, **a || b** returns true if one of a or b is true (i.e. non-zero). Of course, it returns true when both a and b are true.
- ▶ **Logical NOT:** The ‘!’ operator returns true the condition in consideration is not satisfied. Otherwise it returns false. For example, **!a** returns true if a is false, i.e. when a=0.

▶ Truth table:

▶ 1.LOGICAL AND(&&):

<u>exp1</u>	<u>exp2</u>	<u>exp1 && exp2</u>
T	T	T
T	F	F
F	T	F
F	F	F

▶ 2.LOGICAL OR(||):

<u>exp1</u>	<u>exp2</u>	<u>exp1 exp2</u>
T	T	T
T	F	T
F	T	T
F	F	F

▶ 3.LOGICAL NOT(!):

<u>exp1</u>	<u>!(exp1)</u>
-------------	----------------

T	F
---	---

F	T
---	---

```
▶ Ex:    void main()

        {

            const int a=10,b=20,c=30;

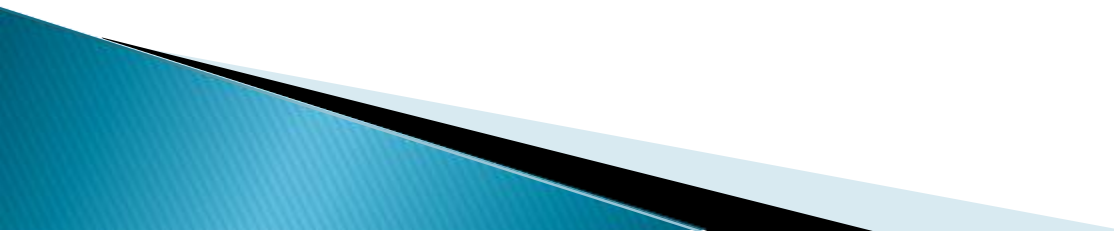
            printf(“%d”,(a>b)&&(a<c));

            printf(“%d”,(a>b)||(a<c));

            printf(“%d”,!(a>b));

            getch();

        }
```



- ▶ **4. Assignment operator:** The assignment operator is used to assign the value to a variable

- ▶ **Syntax:**

variable assignment ope value ;

max = 10;

- ▶

<u>Operator</u>	<u>meaning</u>
-----------------	----------------

- ▶

=	assignment
---	------------

short hand /compound assignment operator the basic form is

=> V operator=exp;

Shorthand assignment operator	Normal assignment operator
<code>a+=1</code>	<code>a=a+1</code>
<code>a-=1</code>	<code>a=a-1</code>
<code>a*=n+1</code>	<code>a=a*n+1</code>
<code>a%=b</code>	<code>a=a%b</code>

▶ Ex:

```
void main()
```

```
{
```

```
    int a=10;
```

```
    clrscr();
```

```
    printf(“%d”,a);
```

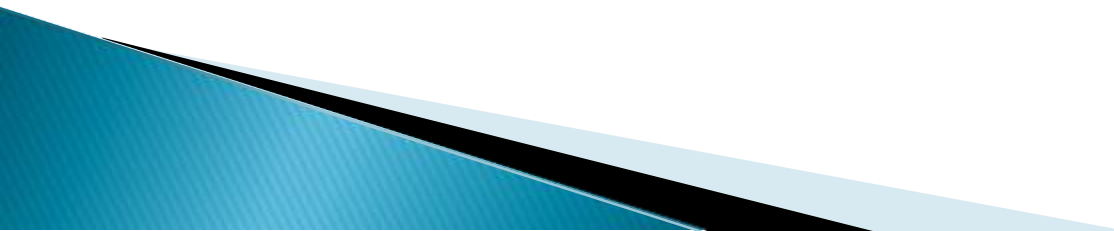
```
    printf(“%d”,a+=10);
```

```
    getch();
```

```
}
```



▶ 5.Increment and decrement operators :

- | ▶ <u>Operator</u> | <u>meaning</u> |
|---------------------|--------------------|
| ▶ ++ | increment operator |
| ▶ -- | decrement operator |
| ▶ Postfix increment | => m++ |
| ▶ Prefix increment | => ++m |
| ▶ Postfix decrement | => m-- |
| ▶ Prefix decrement | => --m |
- 

▶ **Ex:**

▶ 1.Postfix increment

▶ If $M=5$

▶ $y=m++$

▶ o/p: $y=5$ and $m=6$

▶ 2.prefix increment

▶ if $m=5$

▶ $y=++m$

▶ o/p: $y=6$ and $m=6$

- ▶ 3.postfix decrement

- ▶ if $m=5$

- ▶ $y=m--$

- ▶ o/p: $y=5$ and $m=4$

- ▶ 4.prefix decrement

- ▶ If $m=5$

- ▶ $y>--m$

- ▶ o/p: $y=4$ and $m=4$

► Ex1:

post - increment

```
void main()
{
    const int a=10,z;
    z=a++;
    printf("z=%d",z);
    printf("a=%d",a);
    getch();
}
```

► Ex2:

pre-increment

```
void main()
{
    const int a=10,z;
    z=++a;
    printf("z=%d",z);
    printf("a=%d",a);
    getch();
}
```

► Ex3:

post-decrement

```
void main()
{
    const int a=10,z;
    z=a--;
    printf("z=%d",z);
    Printf("a=%d",a);
    getch();
}
```

▶ Ex4:

pre-decrement

```
void main()
{
    const  int a=10,z;
    z=--a;
    printf("z=%d",z);
    Printf("a=%d",a);
    getch();
}
```


▶ 6.Conditional operators

▶ Operator meaning

▶ **?:** conditional or ternary operator

▶ **Conditional operator basic form is:**

▶ $\text{Exp1} ? \text{Exp2} : \text{Exp3}$

▶ **1. $a > b ? a : b$** if $a=10$ **2. $a > b ? a : b$** if $a=20$

▶ $b=15$ $b=10$

▶

▶ o/p=15 o/p:20

▶ Ex:

```
void main()
```

```
{
```

```
    int z;
```

```
    z=(5>3)?1:0;
```

```
    printf(“%d”,z);
```

```
    getch();
```

```
}
```



▶ 7.Bit wise operators :

- ▶ The bitwise operators are used to perform an operations on bits like 0s and 1s.

<u>operator</u>	<u>meaning</u>
&	bitwise AND
	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right
~	compliment operator

▶ Ex:

```
void main()
{
    const int a=12,b=10;
    printf("%d",a&b);
    printf("%d",a|b);
    printf("%d",a^b);
    getch();
}
```

► a&b:

►

	8	4	2	1
a=12	1	1	0	0
b=10	1	0	1	0
	<hr/>			
	1	0	0	0

a&b=8

▶ $a|b$:

▶

	8	4	2	1
a=12	1	1	0	0
b=10	1	0	1	0
	<hr/>			
	1	1	1	0

$a|b=14$

► $a \wedge b$:

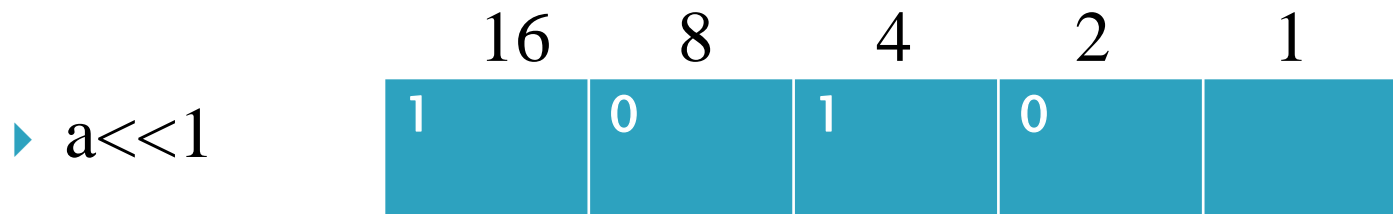
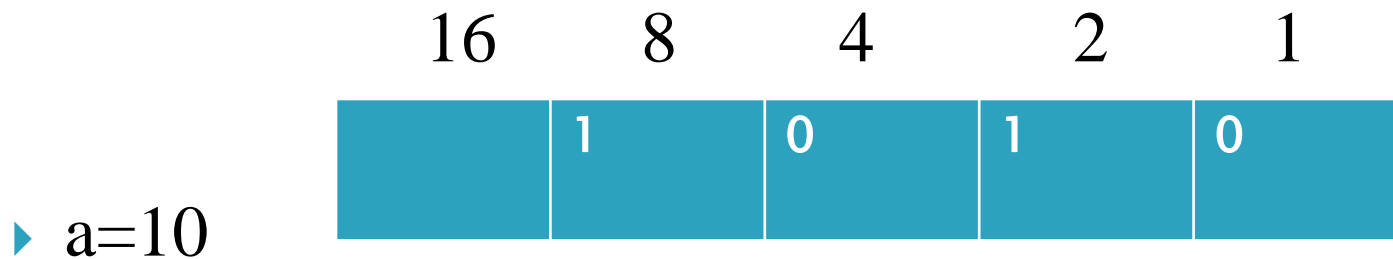
►

	8	4	2	1
$a=12$	1	1	0	0
$b=10$	1	0	1	0
	<hr/>			
	0	1	1	0

$$a \wedge b = 6$$

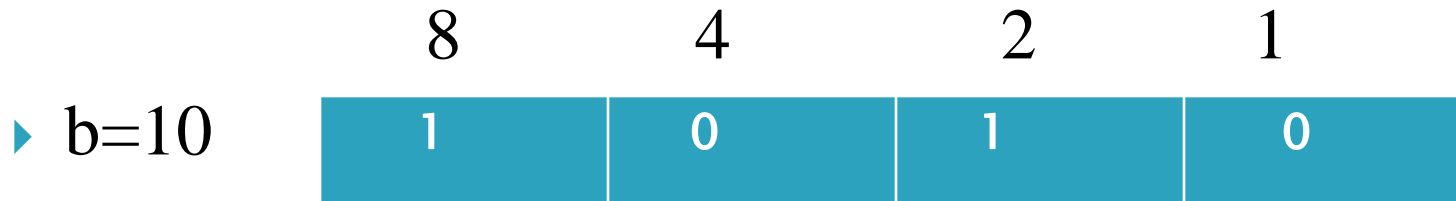
- ▶ Shift left: if the value of the variable shifted left one time ,then its value get doubled

- ▶ Ex: $a=10$ then $a \ll 1 = 20$



o/p: 20

- ▶ Shift right: if the value of the variable shifted right one time, then the value becomes half the original value
- ▶ Ex: $b=10$ then $b>>1=5$

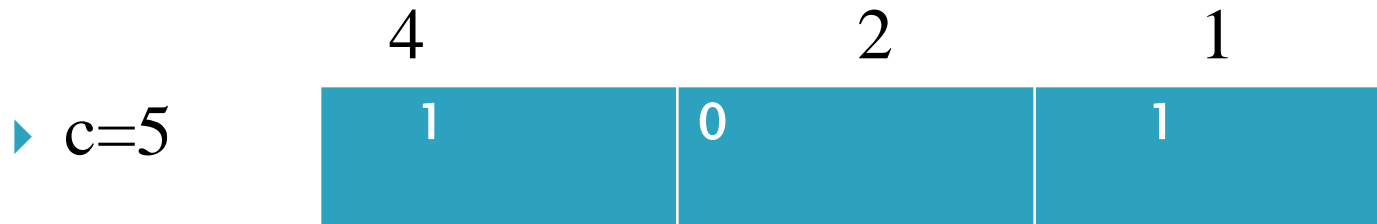


o/p: 5

► **Ones compliment:**

► It converts all ones to zeros and all zeros to ones.

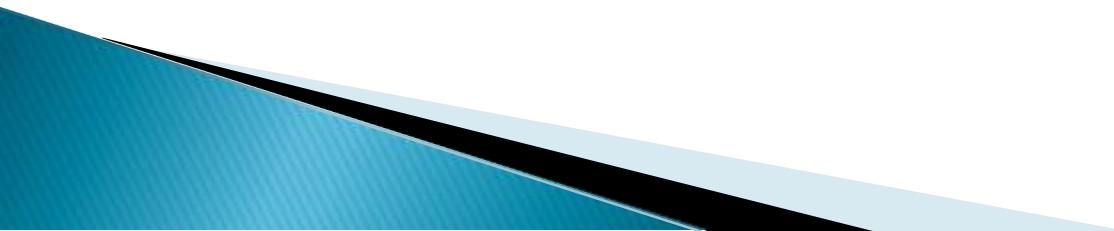
► Ex: $c=5$ then $\sim c=2$



► o/p: 2

▶ Ex:

```
void main()
{
    const int a=10,b=10,c=5;
    clrscr();
    printf("%d",a<<1);
    printf("%d",a>>1);
    printf("%d",~c);
    getch();
}
```



▶ Swap with exclusive OR:

- ▶ $a = a \oplus b;$ if $a=4$ $b=5$
▶ $b = a \oplus b;$
▶ $a = a \oplus b;$

Explanation:

1. $a = 1\ 0\ 0$

$b = 1\ 0\ 1$

$\underline{0\ 0\ 1}$

▶ $a = 0\ 0\ 1$

2. $a = 0\ 0\ 1$

$b = 1\ 0\ 1$

$\underline{1\ 0\ 0}$

$b = 1\ 0\ 0 \Rightarrow 4$

▶ 3. $a = 0\ 0\ 1$

▶ $b = \underline{1\ 0\ 0}$

▶ $\quad 1\ 0\ 1$

$a = 1\ 0\ 1 \Rightarrow 5$

▶ Even or odd using bitwise ope:

1.If a=4

▶ a= 1 0 0

▶ 0 0 1 To perform & operation with 1

▶ 0 0 0

▶ All zeros means to print the given number is even

2. If a=5

▶ a=1 0 1

▶ 0 0 1

▶ 0 0 1


▶ The print odd.

▶ 8.Special operators :

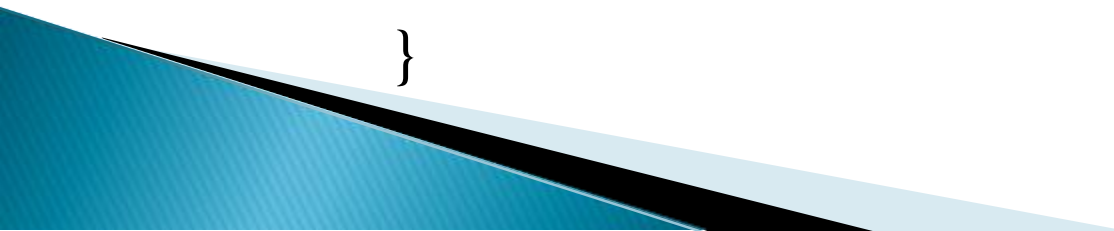
▶ Operator

meaning

▶ ,	comma operator
&	Address operators
sizeof	size of operator

- ▶ Comma operator: It is used to separate the variable
 - ▶ **Ex:** `a=10,b=20`
 - ▶ Address operator: It is used to get the address of a variable
 - ▶ **Ex:** `&a`
 - ▶ Sizeof operator: It is used to get the size of data type of the variable in bytes
 - ▶ **Ex:** `int a=10;`
`sizeof(a);`
- 

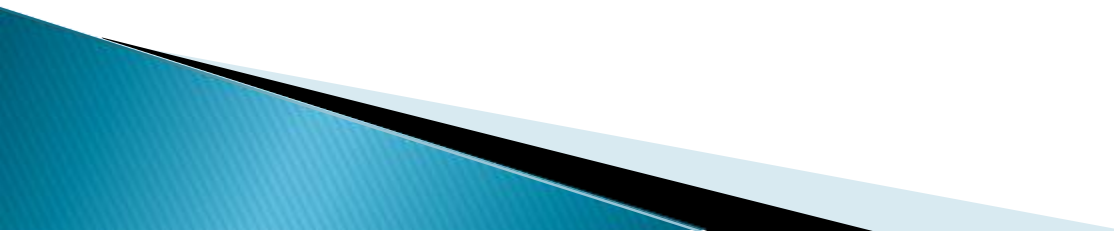
```
▶ Ex:    void main()
        {
            const int a=10;
            const float b=20;
            printf("a=%db=%f",a,b);
            printf("address of a=%u",&a);
            printf("address of b=%u",&b);
            printf("size of a=%d",sizeof(a));
            printf("size of b=%d",sizeof(b));
            getch();
        }
```



operators

- ▶ <https://www.youtube.com/watch?v=d-IgV-KZwwwg>
- ▶ <https://www.youtube.com/watch?v=BfJmJ1t6Ric>
- ▶ <https://www.youtube.com/watch?v=zLwlnCojxYE>

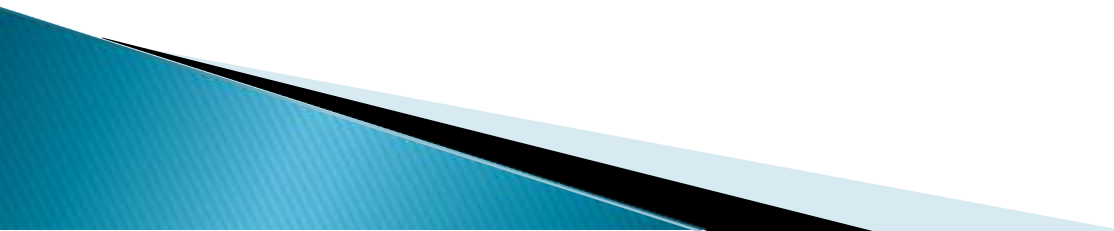
Type conversion

- ▶ Two types of type conversions
 - ▶ 1.Implicit type conversion
 - ▶ 2.Explicit type conversion
- 

Implicit type conversion

► **Ex:**

```
void main()
{
    int i=17,sum;
    char ch='a';
    clrscr();
    sum=i+ch;
    Printf("sum=%d",sum);
    getch();
}
```



Explicit type conversion

► Ex:

```
void main()
{
    int i=17,count=5;
    float  mean;
    clrscr();
    mean=(float) i /count;
    printf(mean=%f',mean);
    getch()
}
```

Expressions

- ▶ Expression : $a+b$, $a-b$, $a*b$, a/b
- ▶ Types:
- ▶ 1.simple expression : $a+b$, $a-b$, $a*b$, a/b , $a\%b$
- ▶ 2.complex expression : $a+b*c$, $a*b/c$
- ▶ 3.unary expression : $+a$, $+b$, $+c$
- ▶ 4.ternary expression : $a>b?a:b$
- ▶ 5.postfix expression: $a++$
- ▶ 6.prefix expression: $++a$

Expression Evaluation

- ▶ **Example1:**

- ▶ $a > b + c \&\& d$ $+ \quad 12$
- ▶ $b + c$ $> \quad 10$
- ▶ $a > b + c$ $\&\& \quad 5$
- ▶ $a > b + c \&\& d$

Left to Right

- ▶ Example3:

- ▶ $3*8/4\%4*5$

$* / \% 13$

- ▶ $3*8$

- ▶ $3*8/4$

- ▶ $3*8/4\%4$

- ▶ $3*8/4\%4*5$

Right to left

▶ Example 4

▶ $a+=b*=c-=5$

▶ $c-=5$

▶ $b*=c-=5$

▶ $a+=b*=c-=5$

$a=3, b=5, c=8$

$c=c-5$

$b=b*c=c-5$

$a=a+b=b*c=c-5$

$(a=a+(b=b*(c=c-5)))$

$(a=3+(b=5*(c=8-5)))$

$a=18$

Precedence and associativity

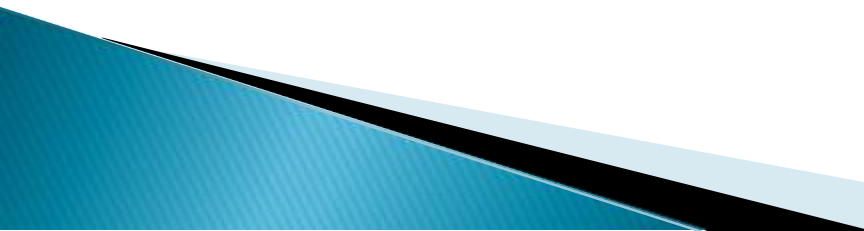
Operator	Meaning	priority	Associativity
,	Comma operator	1	Left-Right
= += -= *= /= %= >>= <<= &= ^= /=	Short hand assignment operators	2	Right - Left
?:	Conditional operator	3	Right - Left
	Logical OR operator	4	Left -Right

&&	Logical AND operator	5	Left – Right
	Bitwise OR	6	Left – right
^	Bit wise exclusive OR	7	Left – right
&	Bit wise AND	8	LEFT – RIGHT
== !=	Equal and not equal	9	Left –right
< <= > >=	Comparison	10	Left – right
<< >>	Shift left and shift right	11	Left – right
+ -	Addition and subtraction	12	Left –right

* / %	Multiply division modulus	13	Left – right
()	Type cast	14	Right –left
++ --	Pre-increment and pre-decrement	15	Right –left
Sizeof	Size in bytes		
~	Ones complement		
!	Not		
+ -	plus minus(+a -a)		
&	Pointer address		
*	Indirection		

[]	Array index		
++ --	Postfix increment and decrement	16	Left – right
. ->	Member selection operator		

Input and output statements

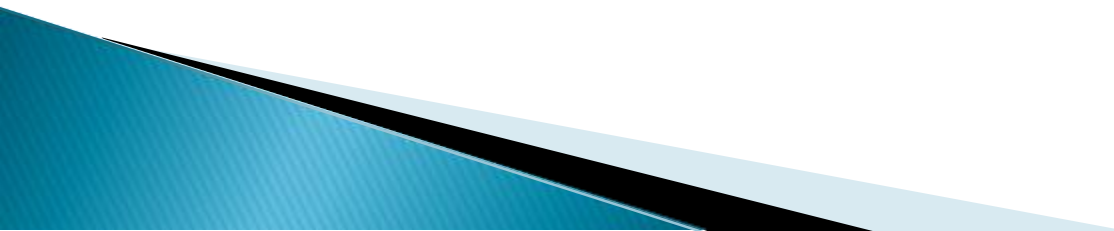
- ▶ Formatted I/O operation
 - ▶ Unformatted I/O operation
 - ▶ Formatted I/O operation:
 - ▶ scanf(): printf():
 - ▶ scanf():
 - ▶ syntax:
 - ▶ scanf (“controlstring”,address of the variable)
- 

- ▶ Examples:
- ▶ Integer data read operation:

```
scanf(“%d”,&a);
```

```
scanf( “%d%d”,&a,&b);
```

```
scanf(“%d%d%d”,&a,&b,&c);
```



▶ Floating point data read statements:

```
scanf(“%f”,&a);
```

```
scanf( “%f%f”,&a,&b)
```

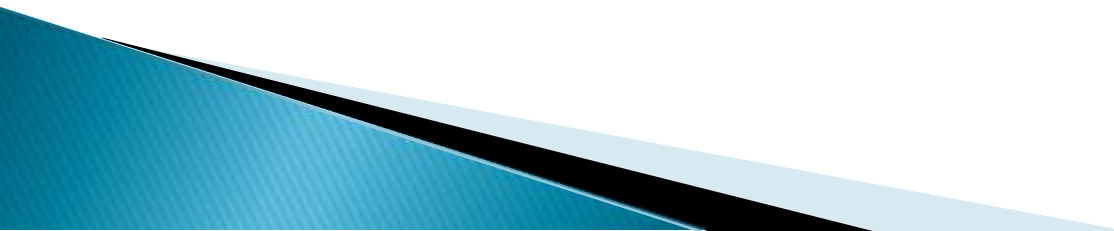
```
scanf(“%f%f%f”,&a,&b,&c);
```

▶ Character data read operation:

```
scanf("%c",&a);
```

```
scanf( "%c%c",&a,&b) ;
```

```
scanf("%c%c%c",&a,&b,&c);
```



▶ **Printf():**

▶ **Syntax:**

▶ **Printf (“control string”,variables);**

- ▶ Examples:

- ▶ Integer data print operation:

Printf(“%d”,a);

Printf(“%d%d”,a,b);

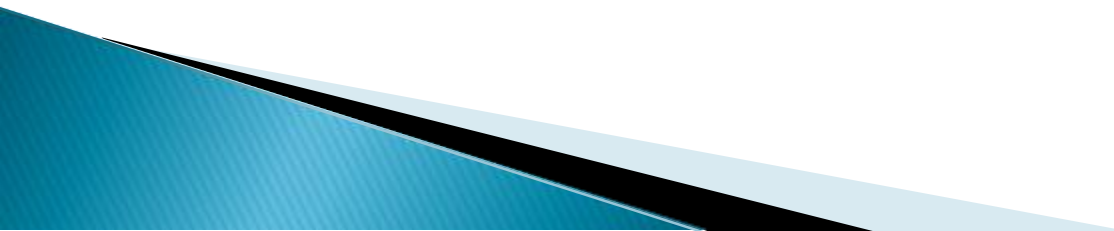
Printf(“%d%d%d”,a,b,c);

► Float point data print operation:

```
Printf(“%f”,a);
```

```
Printf(“%f%f”,a,b);
```

```
Printf(“%f%f%f”,a,b,c);
```

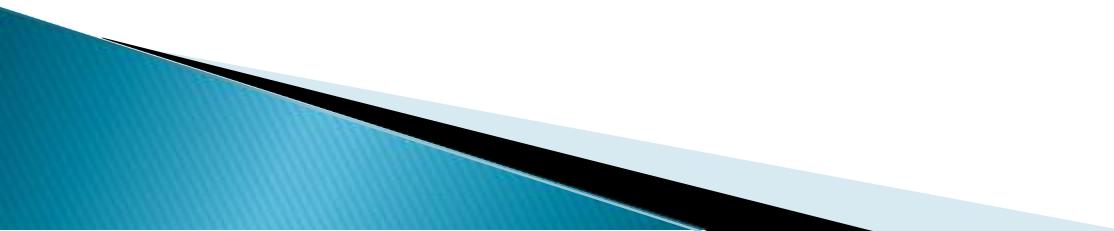


▶ Character data print operation:

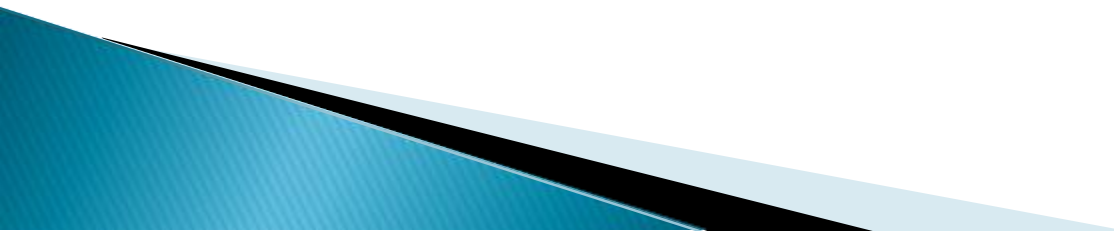
```
Printf(“%c”,a);
```

```
Printf(“%c%c”,a,b);
```

```
Printf(“%c%c%c”,a,b,c);
```



Unformatted I/O operation

- ▶ `getchar()`:
 - ▶ `gets()`:
 - ▶ `putchar()` :
 - ▶ `puts()`:
- 

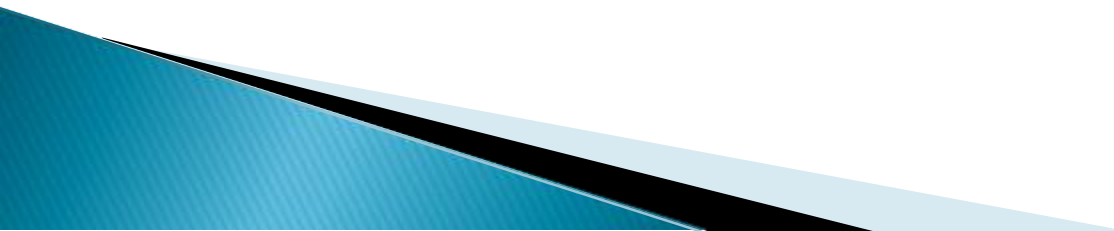
▶ getchar():

▶ Syntax:

ch=getchar()

getchar(a);

Scanf(“%c”,&a)



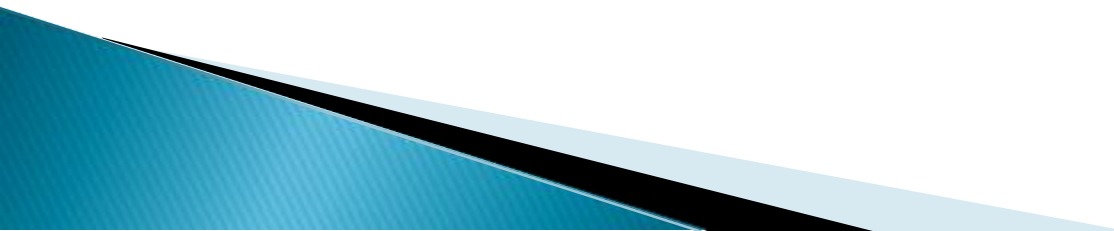
▶ gets():

▶ Syntax:

gets(variablename)

gets(a);

Scanf(“%s”,a);



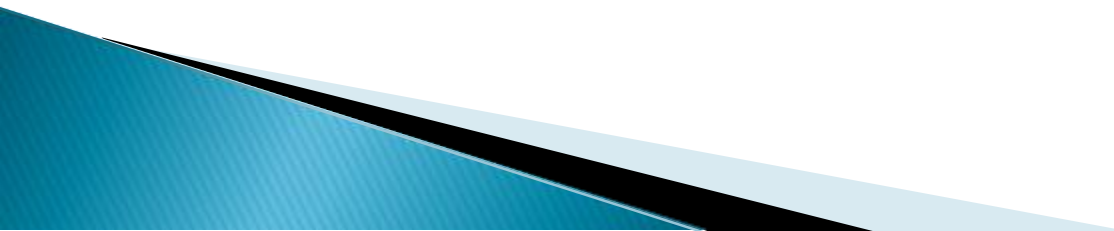
▶ **Putchar():**

▶ **Syntax:**

putchar(variablename)

Putchar(a);

Printf(“%c”,a)



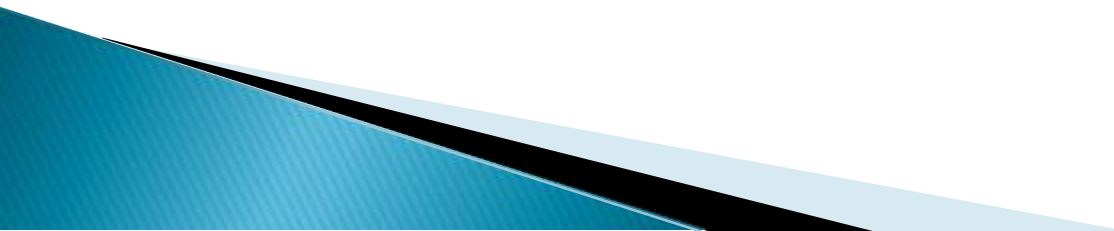
▶ **Puts():**

▶ **Syntax:**

puts(variablename)

Puts(a);

Printf(“%s”,a);

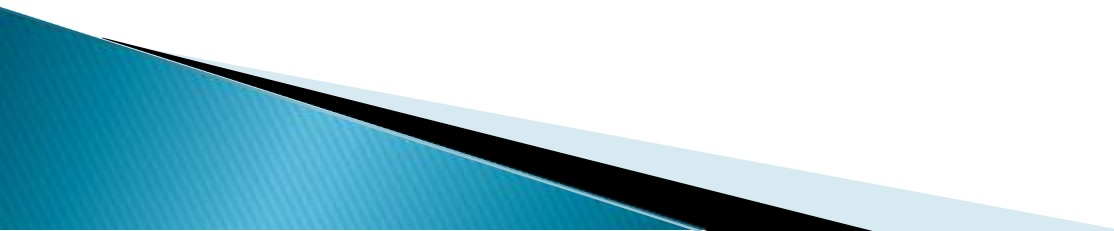


▶ 1. Which of the following is not a valid C variable name?

- a) int number;
- b) float rate;
- c) int variable_count;
- d) int \$main;

▶ 2. All keywords in C are in _____

- a) LowerCase letters
- b) UpperCase letters
- c) CamelCase letters
- d) None of the mentioned

- ▶ 3. The format identifier '%i' is also used for _____ data type.
 - a) char
 - b) int
 - c) float
 - d) double
 - ▶ 4. What is the size of an int data type?
 - a) 4 Bytes
 - b) 8 Bytes
 - c) Depends on the system/compiler
 - d) Cannot be determined
- 

▶ 5.

```
#include<stdio.h>
```

```
Void mian()
```

```
{
```

```
    signed char ch;
```

```
    ch=128;
```

```
    printf(“%d”,ch);
```

```
}
```



▶ 6.

```
#include<stdio.h>
```

```
#define a 10
```

```
Void main()
```

```
{
```

```
    const int a=5;
```

```
    Printf(a=%d"\n",a);
```

```
}
```



▶ 7.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int a=010;
```

```
    Printf(“%d”,a);
```

```
}
```



▶ 8.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    Const int p;
```

```
    P=4;
```

```
    Printf(“p=%d”,p);
```

```
}
```



▶ 9.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int const k=5;
```

```
    k++;
```

```
    printf(“k=%d”,k);
```

```
}
```



▶ 10.

▶ #include<stdio.h>

Void main()

{

j=10;

Printf(“%d”,j++);

}

a)10

b)11

c)compile time error

d)0

▶ 11.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int  a=10/3;
```

```
    printf(“%d”,a);
```

```
}
```



▶ 12.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int _ = 10;
```

```
    int _ = 20;
```

```
    int _ = _ + _;
```

```
    printf(“_ % d”,_ ) ;
```

```
}
```



▶ 13.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int x;
```

```
    x=10,20,30;
```

```
    printf(“%d”,x);
```

```
}
```



► 14.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int a=1,b=1,c;
```

```
    c=a++ + b;
```

```
    printf(“%d,%d”, a,b);
```

```
}
```

a) a=1,b=1

b) a=2,b=1

c) a=1,b=2

d) a=2,b=2

▶ 15.

```
#include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int i=0;
```

```
    int j=i++ + i;
```

```
    printf(“%d”,j);
```

```
}
```

- ▶ a) 0 b) 1 c) 2 d) compile time error

▶ 16.

```
include<stdio.h>
```

```
Void main()
```

```
{
```

```
    int i=2;
```

```
    int j= ++i + i;
```

```
    printf(“%d”,j);
```

```
}
```

- ▶ a) 6 b)5 c)4 d) compile time error

▶ 17.

```
#include<stdio.h>  
int a=4;b=6  
printf(“%d”,a==b);
```

▶ o/p:

▶ **18.**

Which of the following has compilation error in C?

a) `int n = 32;`

b) `char ch = 65;`

c) `float f = (float) 3.2;`

d) None of the above

▶ 19.

Which of the following is a Compound assignment operators?

a) +=

b) *=

c) /=

d) All the above

▶ **20.**

What will be the output of $5.0 / 2$?

a)2

b)3

c)0

d)2.5

▶ **21.**

What is %f, %d, %s and %c?

a) Number Specifier

b) Format Specifier

c) Access Specifier

d) None of the above

▶ **22.**

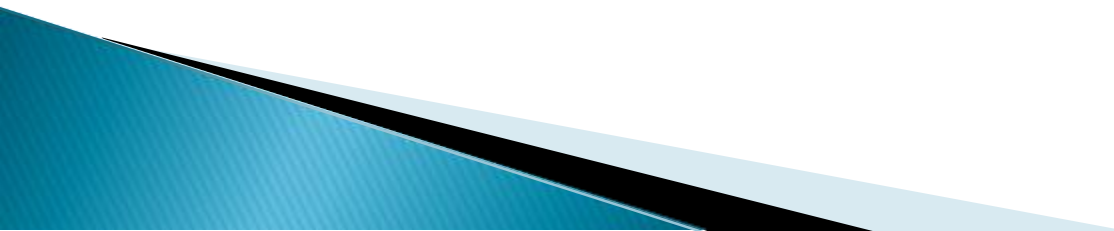
▶ **Which of the following is correct set of keywords?**

a) unsigned, external, typedef, signed

b) unsigned, volatile, typedef, every

c) unsigned, volatile, typedef, sizeof

d) None of the above



▶ **23.**

The keywords are also called

a) Safe words

b) Static words

c) Reserved words

d) Reused words

► 24.

Every C Program must have one function called?

a)switch()

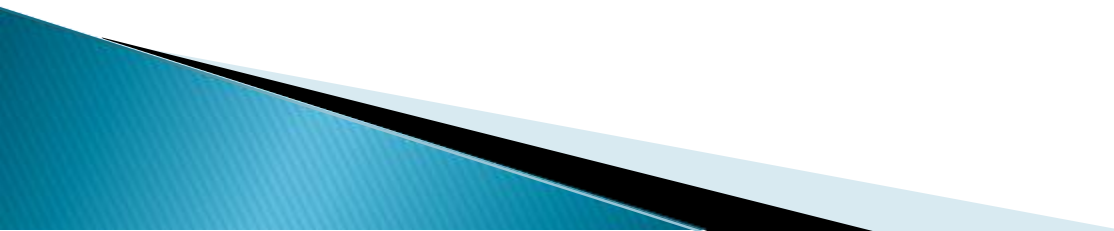
b)main()

c)struct()

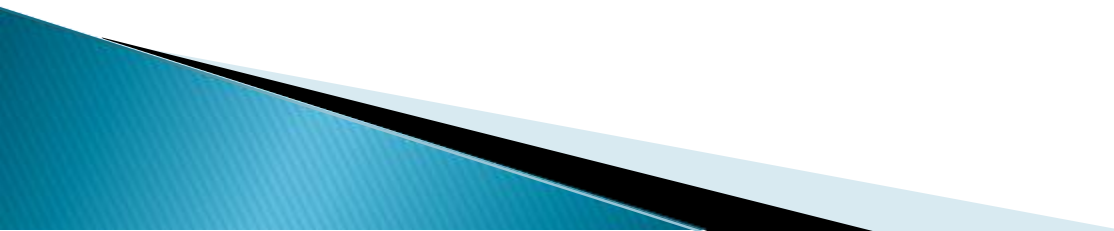
d)for()

25.

The "C" language is

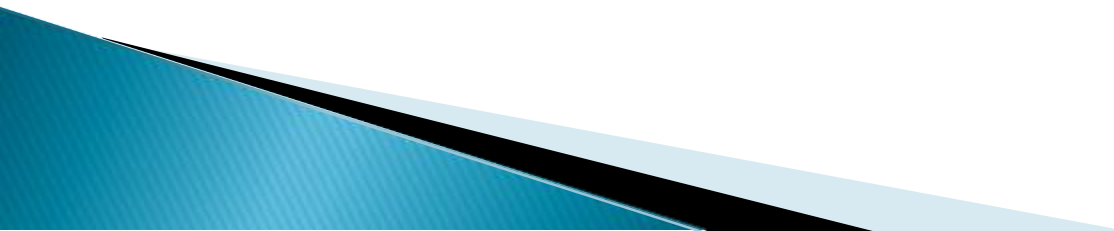
- a) Context free language
 - b) Context sensitive language
 - c) Regular language
 - d) None of the above
- 

Sample programs

- ▶ 1.To perform arithmetic operations
 - ▶ 2.To swap the two numbers with and without using third variable
 - ▶ 3.To calculate simple interest by using $\text{PTR}/100$.
- 

1.To perform arithmetic operations

```
▶ Void main()  
▶ {      int a,b,add,sub,mul,div;  
▶      clrscr();  
▶      Printf("enter two values");  
▶      Scanf("%d%d",&a,&b);  
▶      add=a+b;  sub=a-b;  mul=a*b;  div=a/b;  
▶      Printf("%d%d%d%d",add,sub,mul,div);  
▶      getch();  
▶ }
```



2.To swap the two numbers with and without using third variable

- ▶ Without using third variable

- ▶ $a=a+b;$

- ▶ $b=a-b;$

- ▶ $a=a-b;$

- ▶ With using third variable

- ▶ $temp=a;$

- ▶ $a=b;$

- ▶ $b=temp;$

Control statements

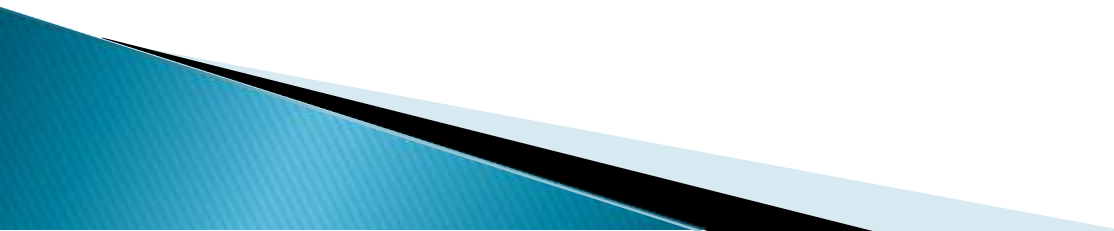
- ▶ 1.selection /decision making statements
- ▶ 2.Iterative /looping statements

▶ Conditional type

- ▶ if
- ▶ if else
- ▶ else if ladder
- ▶ Nested if
- ▶ switch statement

un conditional type

- break
- continue
- goto



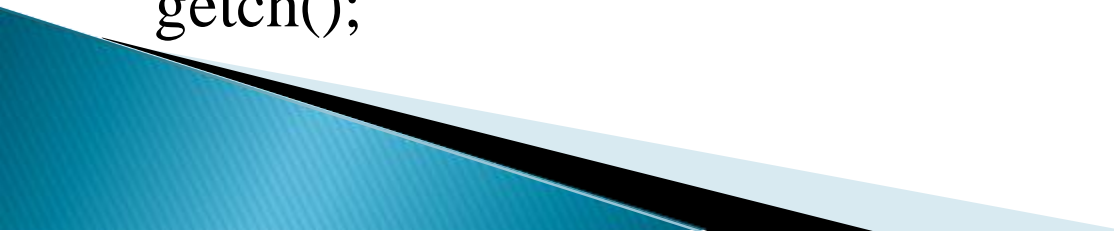
if statement

- ▶ Syntax:

```
if (condition)
{
    Statement;
}
```

To swap two values using if

```
▶ void main()
{
int a,b,temp;
Printf("enter values")
Scanf("%d%d",&a,&b);
if(a>b)
{
    temp=a;
    a=b;
    b=temp;
}
Printf("%d%d",a,b);
getch();
```



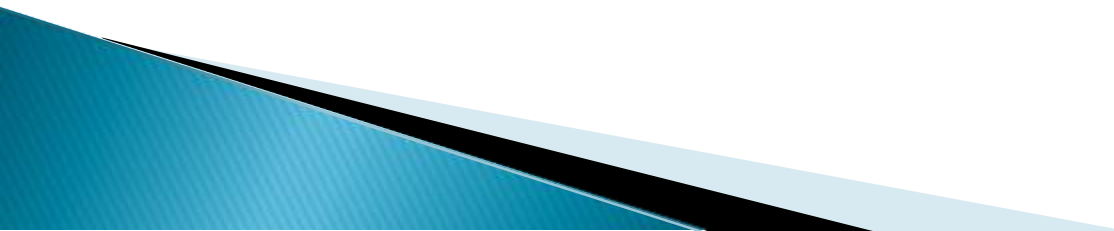
To check two numbers are equal

- ▶ `if(a==b)`
- ▶ `{`
 - `Equal;`
 - `}`
 - `getch();`

If –else statement

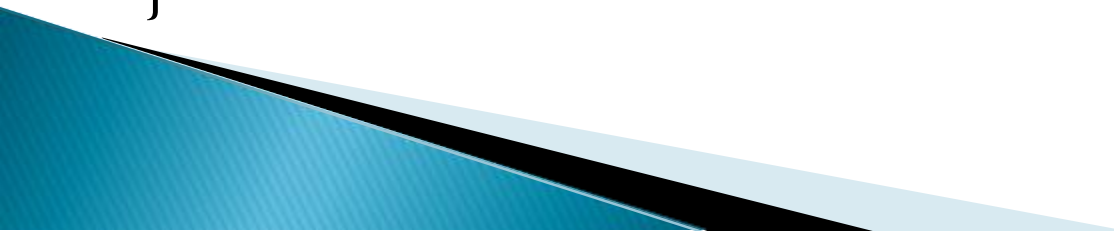
- ▶ Syntax:

```
if (condition)
{
    Statement;
}
else
{
    Statement;
}
```

- ▶ 1.To find the given number is even or odd
 - ▶ 2.To check given two numbers are equal or not
 - ▶ 3.To check the given number is +ve or –ve
 - ▶ 4.Eligible for voting or not
- 

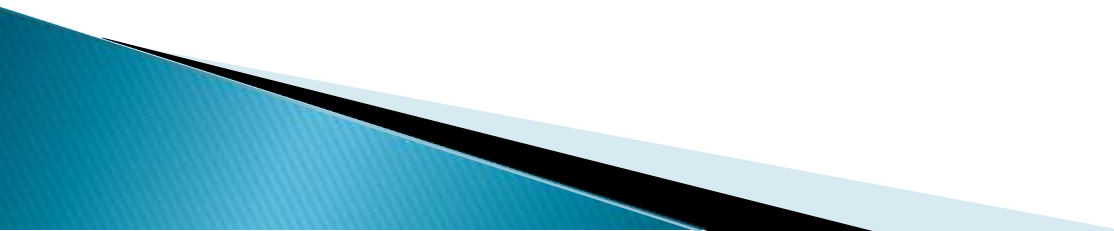
else-if ladder

```
▶ if(condition)
{
    Statement 1;
}
else if(condition)
{
    Statement 2;
}
else
{
    Statement 3
}
```



- ▶ 1.First of all condition_expression_One is tested and if it is true then statement1 will be executed and control comes out out of whole if else ladder.
- 2.If condition_expression_One is false then only condition_expression_Two is tested. Control will keep on flowing downward, If none of the conditional expression is true.
- 3.The last else is the default block of code which will gets executed if none of the conditional expression is true

Examples

- ▶ To check whether given number is +ve , -ve and zero value
 - ▶ To find biggest number given any three numbers
 - ▶ Write a C-Program to print Electricity Bill with Slabs.
 - ▶ Write a C-Program to Print Student grades according to the marks.
- 

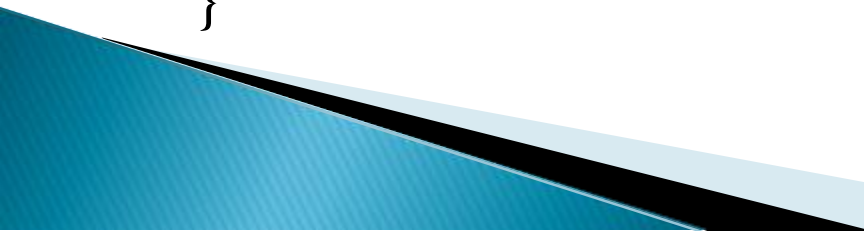
- ▶ Write a c program to enter the temperature and print the following message according to the given temperature.

conditions:

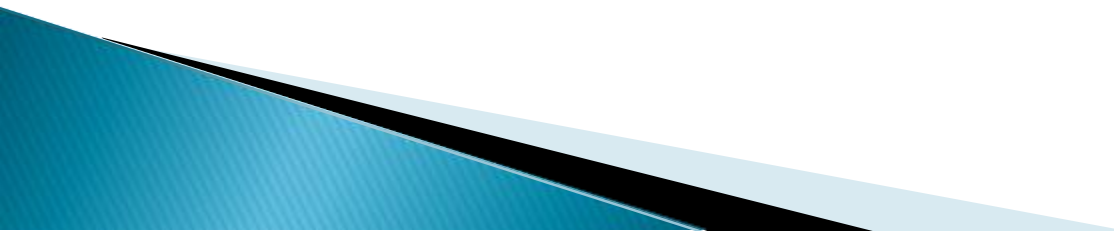
1. $T \leq 10$ \Rightarrow its very very cold
2. $T > 10 \ \&\& \ T \leq 20$ \Rightarrow its cold
3. $T > 20 \ \&\& \ T \leq 30$ \Rightarrow its warm
4. $T > 30 \ \&\& \ T > 40$ \Rightarrow its hot

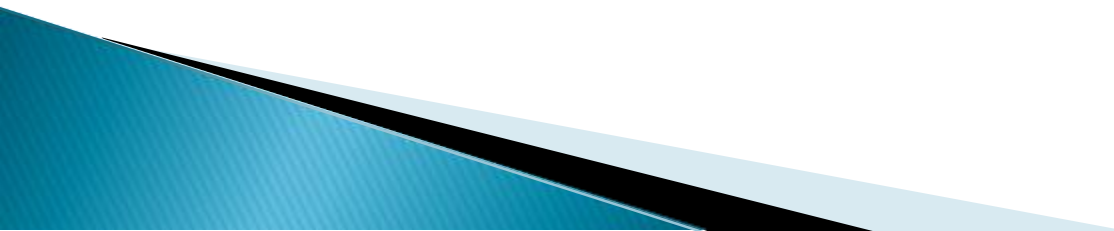
Nested if statement

```
▶ if(condition)
{
    if(condition)
    {
        statement;
    }
    else
    {
        statement;
    }
}
else
{
    statement;
}
```



Switch statement

- ▶ In computer programming languages, a **switch statement** is a type of **selection control mechanism** used to allow the **value of a variable** to change the control flow of program execution .
 - ▶ Switch statement is a control statement that allows to **choose only one choice** among the **many given choices**.
- 

- ▶ The default statement is optional. Even if the switch case statement do not have a default statement, it would run without any problem.
 - ▶ The **break statement is used inside the switch to terminate a statement sequence.** When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement
- 

- ▶ **Valid expressions for switch:**

- ▶ // Constant expressions allowed

- ▶ `switch(1+2+23)`

- ▶ `switch(1*2+3%4)`

- ▶ **Invalid switch expressions for switch:**

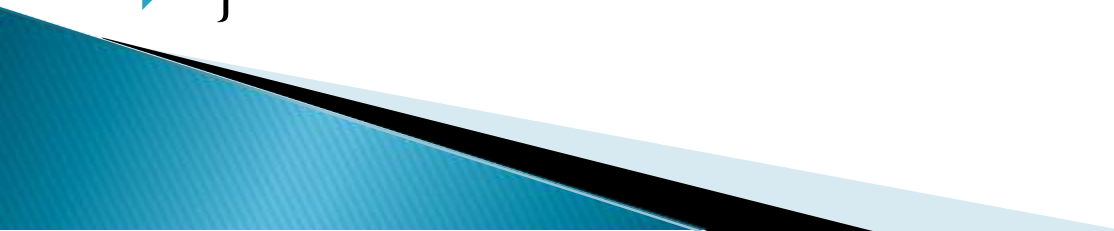
- ▶ // Variable expression not allowed

- ▶ `switch(ab+cd)`

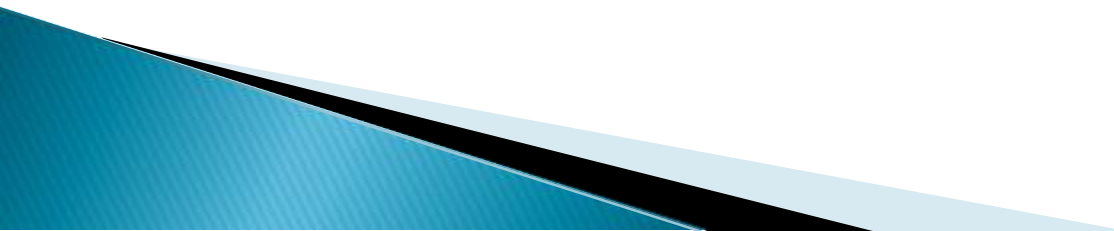
- ▶ `switch(a+b+c)`



```
▶ switch(expression)
▶ {
▶     case label1 : statements;
▶                 break;
▶     case label2 : statements;
▶                 break;
▶     .
▶     .
▶     case label n:statements;
▶                 break;
▶     default : statement;
▶             break;
▶ }
```



Example

- ▶ 1. Write a c program to perform addition subtraction multiplication and division by using switch statement.
 - ▶ 2. Write a c program to implement calculator application by using switch statement.
 - ▶ 3. Write a c program to check whether given number even or odd by using switch
- 

- ▶ 4.write a c program to check whether given character is vowel or not by using switch statement.

Standard functions

- ▶ Mathematical Functions:
 - ▶ “**math.h**” header file supports all the mathematical related functions in C language.
 - ▶ 1. **sqrt()** : This function is used to find square root of the argument passed to this function.
 - ▶ 2. **pow()**: This function is used to find the power of the given number.
 - ▶ 3. **sin()**:This function is used to calculate sine value
 - ▶ 4. **cos()**:This function is used to calculate cosine value
- 