2.1 Input/Output statements:

2.1.1 Concepts of Header files (STDIO,CONIO)

2.1.1.1 Concepts of pre-compiler directives.

2.1.1.2 Use of #include and #define

Preprocessors or What do you mean by preprocessor statement?

Preprocessors directive starts always with # characters. The preprocessor expands the program and gives expanded program to the compiler.

- There are mainly three types of preprocessors.
- 1) Macro substitution 2) File inclusion 3) conditional inclusion.

1) Macro substitution:

- It starts with #define.
- Syntax - #define macroname string

- Macro used for different purposes.

- 1) The first use of the macro is defining **symbolic constant**.
  e.g. #define max 10
  Preprocessors replace the macro max everywhere in program by 10. The main advantage of this macro is that when value of constant changes we need to change at only one place.

- 2) Another use of this macro is to give short name for the long name or strings. e.g #define PRT printf .

- 3) Another use of the macro is macro with parameters. It can take parameters like function. E.g

```
#include <conio.h>
#include <stdio.h>
#define max(x,y) ((x>y)?(x):(y))
 void main()
 {
    int a=10,b=20;
    clrscr();
    printf("max no is %d",max(a,b));
    getch();
    }
    Output:
max no is 20
```

  - If macro is simple and short then we avoid overhead associated with function calls. Otherwise replace macro with function.

2) file inclusion:

- The file inclusion starts with #include.
- Syntax- #include <filename> or #include “filename”
- Filename is the name of the file to be included. This means entire file is physically copied in place of #include statements.
- #include <…> is used to include the header file which are in ..\include directory. The #include “..” is used to include the header file which are in current directory.

3) Conditional inclusion

a) #ifdef, #else and #endif

“#ifdef” directive checks whether particular macro is defined or not. If it is defined, “If” statements are included in source file Otherwise, “else” statements are included.

```
#include <stdio.h>
#define max 100


void main()
{
#ifdef max
        printf("\n max is %d",max);
#else
        printf("\n max is not define");
#endif
getch();
}
```

Output: max is 100

b) #ifndef and #endif

#ifndef exactly acts as reverse as #ifdef directive. If particular macro is not defined, “If” statements are included in source file Otherwise else statements are included

```
#include <stdio.h>
#define max 100


void main()
{
clrscr();
#ifndef min
printf("\n min is not define");
#define min 10
#else
printf("\n min is %d",min);
#endif
getch();
}
```

Output: min is not define

c) #if, #elif, #else and #endif

- "If" statement is included if given condition is true otherwise elif statement is included otherwise else is included.

```
#include <stdio.h>
//#define max 100


void main()
{
clrscr();
#if (max==100)
 printf("\n max is %d",max);
 #elif (max==200)
printf("\n max is %d",max);
#else
printf("\n max is not define so it is define now");
#define max 300
#endif
getch();
}
```

Output: max is 100

Undef preprocessor

This directive undefines existing macro in the program.

```
#include <stdio.h>
#define max 100


void main()
{
clrscr();
printf("\n max is %d",max);
#undef max
#define max 200
printf("\n max is %d",max);
getch();
}
```

Output:

```
max is 100
max is 200
```

2.2 Input/Output Statements:

2.2.1 Input statements : scanf(), getc(), getch(), gets(), getchar()

Formatted input or read data from keyboard by scanf() function

- Scanf() function allows the user to input any type of data i.e. integer , char, float, string .
- syntax: scanf(“control string” ,&variable1, &variable);

Control string specifies the format of data being received. &variable1….. Specifiy the address of location where the data is stored. Control string and variable are separated by commas.

Example

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a;
float b;
char c,d[10];
clrscr();
printf("\n Enter value for a b c d");
scanf("%d %f %c %s",&a,&b,&c,d);
printf("\n value of a b c d");
printf("%d %f %c %s",a,b,c,d);
getch();
}
```

getchar()

- Reading a single character.
- Syntax : variable_name = getchar();
  Variable_name is valid c name declared as char type.
- Eg.   Char c;
            C=getchar();
- When this statements is executed , it waits for the user to enter characters and when user press enter key then it assigns the first character to the variable c .

gets()

- Gets used to read string of text containing whitespace.
- Syntax: gets(variable);
- E.g   gets(str);

- It read character into str from keyboard until a new line character is encounter and then appends a null character (‘\0’) to end of the string.

Getche function
C=getche();
The getche() function assign letter to the variable c as soon as it is enter ,without waiting for the enter key to pressed.

Getch() function
It get single character from the keyboard it does not wait for the enter key to be pressed. The character pressed will not be displayed on the screen.

getc()
syntax char variable=getc(file pointer);
read a character from a file

2.2.2 Output Statements: printf(), putc(),puts(), putchar()

1) **Formatted output or Printf statements**

- Printf() is used to display values of variable on screen or display message on screen.
- General form: printf(“control string”, vari1,vari2….varin);
- Control string consists of three types of item
  - Character that will display on screen.
  - Format specification that define the output format for display of each item (%d,%c,%f,%s)
  - Escape sequence character such as \n, \t.

- **Printing integer number**
  Format specifer to print integer number is %d
- **Printing real number**
  Format specification for print float value is %f

- **Print single character**
  The format specification for printing a character is %c
- **Print string**
  The format specification for printing a string is %s

e.g

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=10;
  float b=12.1;
  char c='s',d[10]="c language";
  clrscr();
  printf("\n\ta=%d \tb=%f \tc=%c \td=%s",a,b,c,d);
  getch();
}
```

Write string using puts.

- Puts used to print string value.
- syntax: puts(variable);
- e,g. puts(str)
- It print the value of the string variable str and then moves the cursor to the beginning of the next line on the screen.
- E.g

```
#include <stdio.h>
#include <conio.h>
void main()
{
  char c[50];
  clrscr();
  gets(c);
  puts(c);
  getch();
}
```

Write a character by putchar()

- It writes one character at a time.
- Syntax: putchar (variable_name);
- E.g putchar(c)

e.g // to read and write character by getchar and putchar

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char c;
   clrscr();
   printf("\n Enter alphabet:");
   c=getchar();
   printf("\n Alphabet is: ");
   putchar(c);
   getch();
}
```

putc()

syntax putc(char variable, file pointer)
write a character to a file

2.2.3 Type specifiers (formatting strings) : %d, %ld, %f, %c, %s, %lf

2.3 Operators :

The symbols which are used to perform operations are called operators. C language has many types of operators.

2.3.1 Arithmetic operators (+, -, *, /, %, ++, --,)

- Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus in C programs.

S.no	Arithmetic Operators	Operation	Example
1	+	Addition	A+B
2	-	Subtraction	A-B
3	*	multiplication	A*B
4	/	Division	A/B
5	%	Modulus	A%B

2.3.2 Logical Operators (&&, ‖, !)

- There are 3 logical operators in C language.
- Logical AND (&&), logical OR (||) and logical NOT (!).

S.no	Operators	Name	Example	Description
1	&&	logical AND	(a>b)&&(a>c)	It returns true when both conditions are true
2	||	logical OR	(x>=10)||(y>=10)	It returns true when at-least one of the condition is true
3	!	logical NOT	!((x>5)&&(y<5))	It reverses the result If "((x>5) && (y<5))" is true, logical NOT operator makes it false

2.3.3 Relational Operators (>, <, ==, >=, <=, !=)

- Relational operators are used to compare the values of two variables in a C program.

S.no	Operators	Example	Description
1	>	x > y	x is greater than y
2	<	x < y	x is less than y
3	>=	x >= y	x is greater than or equal to y
4	<=	x <= y	x is less than or equal to y
5	==	x == y	x is equal to y
6	!=	x != y	x is not equal to y

2.3.4 Bit-wise operators (&, |, ^ , <<, >>)

Shift operators(Right shift and left shift)

Right shift operator

It operate on single variable it represent by ">>" sign. It shift each bit in the operand to the right. The numbers of the place the bits are shift depend on the number following the operand.

e.g. 11010111>>1

Output

01101011

Bits are shifted to right the blank create on the left this blanks are always fill with zero.

Left shift operator

it represent by “<<” sign. It is similar to the right shift operator. The only difference is that the bits are shifted to the left and zero is added to the right.

e.g. 11010111<<1
Output
10101110

AND operator

It is represent by “&” sign.

```
e.g.    1100
      & 1010
output 1000
```

If both bits are 1 the output is 1 otherwise output is 0. It is used to turn off a particular bit in a number.

OR operator

It is represent by “|” sign.

```
e.g.    1100
        | 1010
output 1110
```

If any one of two bits or both bits are 1 the result will be 1. It is used to put on particular bit in a number.

XOR operators

This operator is representing as “^”. It is also known as exclusive OR operator. The OR operator return 1 when any one of two bits or both the bits are 1 Where as XOR return 1 only if one of the two bits is 1.

```
e.g. 0 ^ 0 =0
     1 ^ 0 =1
     1 ^ 1 =0
     0 ^ 1 =1
```

2.3.5 Assignment operators (=, +=, -=, *=, /=, %=)

- In C programs, values for the variables are assigned using assignment operators (=).
- Other assignment operators in C language are given below.

Operators		**Example**	**Explanation**
Simple assignment operator	=	sum = 10	10 is assigned to variable sum
Compound assignment operators Or Short hand assignment operators	+=	sum += 10	sum = sum + 10
	-=	sum -= 10	sum = sum – 10
	*=	sum *= 10	sum = sum * 10
	/+	sum /= 10	sum = sum / 10
	%=	sum %= 10	sum = sum % 10
	&=	sum&=10	sum = sum & 10

	^=	sum ^= 10	sum = sum ^ 10

2.3.6 Ternary Operator and use of sizeof() function.

- Conditional operators return one value if condition is true and returns another value is condition is false.
- This operator is also called as ternary operator.

```
Syntax    :      (Condition? true_value: false_value);
Example
#include <stdio.h>
#include <conio.h>

void main()
{
    int a,b,x;
    a=10;
    b=15;
    x=(a>b)?a:b;
    printf(“%d”,x);
    getch();
}
```

2.4 Important Built-in functions:

2.4.1 Use of <string.h> : (strlen, strcmp, strcpy, strcat, strrev)

strlen(): this function count and return number of character in a string.
Syntax: strlen(string)
Example: printf("\n\t len is %d",strlen(“cppm”));
Output: len is 4

strcmp(): compare two string character by character if they are equal then return 0. If they are not equal then return difference between the first non matching characters in specified string.
Syntax: strcmp(string1,string2);
Example: printf("\n\t %d",strcmp(“that,”than”));
Output: 6

strcpy(): This function is used to copy one string to other string.
Syntax: strcpy(string1, string2)
Example: char c[20];
printf(“%s”,strcpy(c,”delhi”));

strcat(): this function is used to combine two string.
Syntax: strcat(string1, string2);
Example: printf(“%s”,strcat(“c”, “language”));
Output: c language

strrev(): this function is used to reverse string.
Syntax: strrev(string);
Example: printf("\n\t %s",strrev(“cppm”));
Output: mppc

2.4.2 Use of <math.h> : (abs(), floor(), round(), ceil(), sqrt(), exp(), log(), sin(), cos(), tan(), pow() and trunc())

We should include the line **#include <math.h>** in beginning of the program.

1) **Abs() :** It returns absolute value of integer parameter.
   Example-□ printf("\n\t x=%d",abs(-24));
   Output: x=24

2) **Ceil(x):** It return smallest integer which is not less than the argument x.
   Example-□ printf("\n\t x=%f",ceil(24.3));
   Output: x=25

3) **Floor(x) :** It return largest integer which is less than or equal to argument x.
   Example-□ printf("\n\t x=%f",floor(24.6));
   Output: x=24

4) **Cos(x):** Cosine of x
5) **Sin(x):** Sine of x
6) **Tan(x):** Tangent of x

7) **Exp(x):** E to the x power
   Example-□ printf("\n\t x=%f",exp(10));
   Output: x=2.178282

8) **Log(x):** Natural log of x, x>0
9) **log10(x):** Base 10 log of x, x>0

10) **Pow(x,y):** x to power y.
    Example-□ printf("\n\t x=%f",pow(2,3));
    Output: x=8.000000

11) **Sqrt(x) :**Square root of x, x>=0
    Example-□printf("\n\t x=%f",sqrt(9));
    Output: x=3.000000