UNIT-2: Input/Output Statements and Operators

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?

Preprocessor directives always begin with the # character. The preprocessor expands the program and provides the expanded program to the compiler. There are mainly three types of preprocessors:
1) Macro substitution, 2) File inclusion, and 3) Conditional inclusion.

1)Macro substitution
It begins with #define. Macros are utilized for various purposes.
General Syntax: #define macroname string

The initial use of the macro is for defining a symbolic constant, for example, #define max 10. Preprocessors replace the macro max throughout the program with the value 10. The primary advantage of this macro is that when the value of the constant changes, we only need to modify it in one place.

Another use of the macro is a macro with parameters. It can take parameters similar to a function.

Example 2.1 Macro with parameters

```
#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 a macro is simple and short, we can avoid the overhead associated with function calls. Otherwise, we should replace the macro with a function.

2) file inclusion

File inclusion begins with #include.
Syntax: #include <filename> or #include "filename".
Filename refers to the name of the file to be included. This means that the entire file is physically copied in place of the #include statements.
#include <...> is utilized to include header files located in the ..\\include directory(Root directory).
On the other hand, #include "..." is used to include header files from the current directory.

3) Conditional inclusion

a) #ifdef, #else and #endif

The "#ifdef" directive checks whether a particular macro is defined or not. If it is defined, "if" statements are included in the source file. Otherwise, "else" statements are included.

Example 2.2 #ifdef example

```
#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 acts as the reverse of the #ifdef directive. If a particular macro is not defined, "if" statements are included in the source file. Otherwise, else statements are included.

Example 2.3 #ifndef

```
#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

An "if" statement is included if the given condition is true. Otherwise, an "elif" statement is included. If neither the "if" condition nor the "elif" condition is met, the "else" statement is included.

Example 2.4 #if,#elif,#else and #endif

```
#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 defined so it is define now");
          #define max 300
          #endif
          getch();
}
```
Output: max is 100

Undef preprocessor

This directive undefines an existing macro in the program.

Example 2.5 Undef

```
#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

The `scanf()` function allows the user to input any type of data, such as integer, char, float, or string. syntax is as follows: scanf("control string", &variable 1, &variable 2);
The control string specifies the format of the data being received. `&variable 1` and `&variable 2` specify the addresses where the data is stored. The control string and variables are separated by commas.

Example 2.6 Input/Output Statements

```
#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();
}
Output:
Enter value for a b c d 1 1.2 a hello
value of a b c d 1 1.200000 a hello
```

getchar() Reading a single character.
Syntax : variable_name = getchar();
The variable_name is a valid C name declared as a char type.
For example, char c; c = getchar();
When this statement is executed, it waits for the user to enter characters.
Upon the user pressing the enter key, it assigns the first character to the variable c.

gets() function is used to read a string of text containing whitespace.
The syntax is as follows: gets(variable); For example, gets(str);
It reads characters into the variable str from the keyboard until a new line character is encountered, and then appends a null character ('\0') to the end of the string.

getche() getche() function assigns a letter to the variable c as soon as it is entered, without waiting for the enter key to be pressed.For example, char c;c=getche();

getch() function is used to read a single character directly from the console without echoing it to the screen.It gets a single character from the keyboard without waiting for the Enter key to be pressed. The character pressed will not be displayed on the screen.

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

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

1) Formatted output/printf statements

printf() is used to display the values of variables on the screen or to display messages on the screen. General form - printf("control string", var1, var2, …, varn).
The control string consists of three types of items: characters that will be displayed on the screen, format specifications that define the output format for the display of each item (%d, %c, %f, %s), and escape sequence characters such as \n, \t.

Format specifiers
%d: for printing integers
%f: for printing floating-point numbers
%c: for printing characters
%s: for printing strings

Example 2.7 Input/output statements
```
#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();
}
Output:
a=10 b=12.100000 c=s d=c language
```

puts() function is used to print a string value.The syntax for using "puts" is as follows: puts(variable); For example, puts(str) prints the value of the string variable "str" and then moves the cursor to the beginning of the next line on the screen.

Example 2.8 Read and write string using gets() and puts()

```
#include <stdio.h>
#include <conio.h>
void main()
{
        char c[50];
        clrscr();
        puts("Enter string");
        gets(c);
        puts("String is ");
        puts(c);
        getch();
}
Output :
Enter string
C Language
String is
C Language
```

putchar() writes one character at a time.
The syntax for using the 'putchar' function is as follows: putchar(variable_name);
For example, putchar(c) writes the character c to the output.

Example 2.9 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();
}
```

```
Output:
Enter alphabet: a
Alphabet is:a
```

putc() write a character to a file in C, you can use the putc() function.
The syntax for using putc() is putc(char variable, file pointer).
For example, putc('a', file_ptr) writes the character 'a' to the file pointed to by file_ptr.

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

2.3 Operators

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

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

Arithmetic operators in C programs are used to perform mathematical calculations such as addition, subtraction, multiplication, division, and modulus.The modulus operator returns the remainder of a division operation between two integers. For example, the expression '5 % 2' would return 1.The modulus operator is commonly used in programming for tasks such as checking for even or odd numbers.

No	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 three logical operators in the C language: logical AND (&&), logical OR (||), and logical NOT (!).

No	Operator	Meaning	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 conditions is true.
3	!	logical NOT	!((x>5)&&(y<5))	It reverses the result. If "((x>5) && (y<5))" is true, the 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.

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 Bitwise operators (&, |, ^ , <<, >>)

Shift operators(Right shift and left shift)

The right shift operator operates on a single variable and is represented by the ">>" sign. It shifts each bit in the operand to the right. The number of places the bits are shifted depends on the number following the operand.

e.g. 11010111>>1
Output
01101011

Bits are shifted to the right, creating a blank space on the left. This blank space is always filled with zeros.

The left shift operator is represented by the "<<" sign. It is similar to the right shift operator, with the only difference being that the bits are shifted to the left and zeros are added to the right.

e.g. 11010111<<1
Output
10101110

AND operator is represented by the "&" sign.

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

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

OR operator is represented by the "|" sign.

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

If any one of the two bits or both bits are 1, the result will be 1. It is used to put on a particular bit in a number.
XOR operators are represented by the "^" symbol and are also known as the exclusive OR operator.
The OR operator returns 1 when either or both of the bits are 1, whereas XOR returns 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 variables are assigned using assignment operators (=). Other assignment operators in the C language are listed below.

Operators		Example	Description
Simple assignment operator	=	sum = 10	10 is assigned to variable sum
Compound assignment operators Or Shorthand 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.
Ternary operators return one value if the condition is true and another value if the condition is false. This operator is also known as the conditional operator.
Syntax : (Condition? true_value: false_value);

Example 2.10 Ternary Operator
```
#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();
 }
Output: 15
```

2.4 Important Built-in functions

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

strlen(): This function counts and returns the number of characters in a string.
Syntax: strlen(string)
Example: printf("\n\t len is %d", strlen("cppm"));
Output: len is 4

strcmp(): compares two strings character by character. If they are equal, it returns 0.
If they are not equal, it returns the difference between the first non-matching characters in the specified strings.
Syntax: strcmp(string1, string2);
Example: printf("\n\t %d", strcmp("that", "than"));
Output: 6

strcpy(): This function is used to copy string2 into string1.
Syntax: strcpy(string1, string2).
Example:
printf("%s", strcpy(“”, "Surat"));
Output: Surat

strcat(): This function is used to combine two strings.
Syntax: strcat(string1, string2);
Example: printf(“%s”,strcat(“Hello”, “World”));
Output: HelloWorld

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

Example 2.11 String function

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
  clrscr();
  printf("\n%d",strlen("cppm"));
  printf("\n%d",strcmp("that","than"));
  printf("\n%s",strcpy("","Surat"));
  printf("\n%s",strcat("Hello","World"));
  printf("\n%s",strrev("cppm"));
  getch();
}
```

Output:

```
4
6
Surat
```

```
HelloWorld
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 header file #include <math.h> at the beginning of the program.

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

ceil(x): It returns the smallest integer that is not less than the argument x.
Example: printf("\n\t x=%.2lf", ceil(24.3));
Output: x=25.00

Floor(x) : It returns the largest integer that is less than or equal to the argument x.
Example, printf("\n\t x=%.2lf", floor(24.6));
Output: x=24.00

Cos(x): represents the cosine of x.
Example : printf("\n\t %.2lf",cos(0));
Output : 1.00

Sin(x): represents the sine of x
Example : printf("\n\t %.2lf",sin(0));
Output : 0.00

Tan(x): represents the tangent of x
Example : printf("\n\t %.2lf",tan(0));
Output : 0.00

Exp(x): Exponent value of x.
Example: printf("\n\t x=%lf",exp(5.0));
Output: x=148.413159

Log(x): Natural log of x with base e, x>0
Example: printf("\n\t x= %lf",log(2));
Output :x=0.693147

log10(x): Base 10 log of x, x>0
Example: printf("\n\t x=%lf",log10(2));
Output : x=0.301030

Pow(x,y): x to power y.
Example- printf("\n\t power=%.2lf",pow(2,3));
Output: power=8.00

Sqrt(x) : Square root of x, x>=0
Example- printf("\n\t x=%.2lf",sqrt(9));
Output: x=3.00

Example 2.12 Math function

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

void main()
{
  clrscr();
  printf("\n\t x=%d", abs(-24));
  printf("\n\t x=%.2lf", ceil(24.3));
  printf("\n\t x=%.2lf", floor(24.6));
  printf("\n\t x=%lf",exp(5.0));
  printf("\n\t power=%.2lf",pow(2,3));
  printf("\n\t x=%.2lf",sqrt(9));

  printf("\n\t x= %lf",log(2));
  printf("\n\t x=%lf",log10(2));

  printf("\n\t %.2lf",cos(0));
  printf("\n\t %.2lf",sin(0));
  printf("\n\t %.2lf",tan(0));
getch();
}
```

Output:

```
        x=24
        x=25.00
        x=24.00
        x=148.413159
        power= 8.00
        x=3.00
        x=0.693147
        x=0.301030
        1.00
        0.00
        0.00
```

Programming exercises and solutions.

1. Write a C program to calculate simple interest(si).

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

void main()
{
  int p=1000,t=1;
  float r=1.1,si;
  clrscr();
  printf("\n p=%d",p);
  printf("\n t=%d",t);
  printf("\n r=%.2f",r);
  si=(p*r*t)/100;
  printf("\n simple interest is =%f",si);
  getch();
}
```

2. Write a C program to calculate the addition, subtraction, multiplication and division of two numbers.

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

void main()
{
  int a=20,b=10;
  clrscr();
  printf("\n a=%d",a);
  printf("\n b=%d",b);
  printf("\n sum is =%d",a+b);
  printf("\n sub is =%d",a-b);
  printf("\n multiplication is =%d",a*b);
  printf("\n division is =%d",a/b);
  getch();
}
```

3. Write a C program to input two numbers and swap them.

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

void main()
{
        int a,b;
        clrscr();
        printf("\n enter a");
        scanf("%d",&a);
        printf("\n enter b");
        scanf("%d",&b);

        clrscr();
        printf("\n value of a and b before swapping");
        printf("\n%d %d",a,b);

        //swapping without third variable
        a=a+b;
        b=a-b;
        a=a-b;

        printf("\n value of a and b after swapping");
        printf("\n%d %d",a,b);
        getch();
}
```

4. Write a C program to print a marksheet in the proper format.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  char name[10];
  int rn;
  float m1,m2,m3,m4,m5,t,p;
  clrscr();
  printf("\n enter rn:");
  scanf("%d",&rn);
  printf("\n enter name:");
  scanf("%s",name);
  printf("\n enter m1 m2 m3 m4 m5");
  scanf("\n %f %f %f %f %f",&m1,&m2,&m3,&m4,&m5);
  t=m1+m2+m3+m4+m5;
  p=t/5;
  clrscr();
  printf("\n\t\t\t MARKSHEET                    ");
  printf("\n\t\t\t=====================");
  printf("\n\t\t\t NAME- %s",name);
  printf("\n\t\t\t=====================");
  printf("\n\t\t\t ROLL NO -%d",rn);
  printf("\n\t\t\t=====================");
  printf("\n\t\t\t M1 -%f",m1);
  printf("\n\t\t\t M2 -%f",m2);
  printf("\n\t\t\t M3 -%f",m3);
  printf("\n\t\t\t M4 -%f",m4);
  printf("\n\t\t\t M5 -%f",m5);
  printf("\n\t\t\t=====================");
  printf("\n\t\t\t total - %f",t);
  printf("\n\t\t\t per   - %f",p);
  printf("\n\t\t\t=====================");
  getch();
}
```