Module 2 – Introduction to Programming.

THEORY EXERCISES:

1. Overview of C Programming.

1. Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today.

Ans:

- C programming language was developed in the early 1970s by Dennis Ritchie at bell laboratories.
- It was mainly created to develop to the UNIX operating system.
- Over time, C was standardized to ensure consistency across systems.

- C is important because it is fast, portable and give direct control over hardware.
- Many modern languages such as C++, JAVA, and PYTHON are influenced by C.

- Today C is still widely used because it is efficient, powerful, and close to hardware.

2. SETTING UP ENVIRONMENT.

2. Describe the steps to install a C compiler (e.g. , Gcc) and set up an integrated development environment (IDE) like Dev c++, vs code or code books.

1. Install C compiler.

- Download MinGW (Windows) or use built-in GCC (Linux/MacOS).
- Install it and add the bin folder to the PATH.
- Check installation by typing: gcc –version.

2. Install IDE:

- Dev C++: download and install (comes with GCC)
- **VS Code**: Install VS Code → install C/C++ extension → use GCC.
- **Code:Blocks**: Install version with MinGW → compiler auto-detected.

3. Start coding:

- Create a new C file.
- Write code, compile, and run using the IDE.

3. BASIC STRUCTURE OF C PROGRAM.

3. Explain the basic structure of C program, including header, main function, comments, data types, and variables, provide examples.

Ans:

1. HEADER FILE.

- Header file contains pre-define functions and declarations that help the program to perform specific task.
- Including using "#include"

- EXAMPLE

  - Stdio.h: - standard input/output functions
  - Conio.h: - console input/output (compiler-specific)

- EXAMPLE

  - #include<stdio.h>

2. MAIN FUNCTION.

- The main() function is the entry point of every C program.
- Program execution always start from main().
    - Return an integer(int).
    - Return0; indicates successful program execution

- EXAMPLE

```
Int main()
{
    Return0;
}
```

3. COMMENTS.

- Comments are used to explain code and are ignored by the compiler.

TYPES OF COMMENTS.

- Single line comment.
- Multi line comment.

- SINGLE LINE COMMENT.

- // This is a single line comment

- MULTI LINE COMMENT.

  - /* This is a multi

    Line comment. */

4. DATA TYPES.

- Data type specify the type of data a variable can store.

- COMMON DATA TYPES.

  - INT – Integers
  - FLOAT – Decimal numbers.
  - CHAR – Single character.
  - DOUBLE – Large decimal numbers.

5. VARIABLES.

- Variable is the name of the memory location which store some data.

- Variables are case sensitive.
- 1st character is alphabet or underscore.
- no comma, no blank space.

4. OPERATORS IN C.

- Operator are symbol that perform operations on oprands.

1. ARITHMETIC OPERATORS.

- Arithmetic operators are used to perform basic mathematical calculations.

Operator	Meaning	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus(remainder)	a % b

2. RELATION OPERATORS.

- Used to compare values.

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

3. LOGICAL OPERATOR.

- Used to combine condition.

&&	AND
||	OR
!	NOT

4. ASSIGNMENT OPERATOR.

- Used to assign value to variable.

=	a=10	Assign
+=	a+=10	a=a+10
-=	a-=10	a=a-10
*=	a*=2	a=a*2
/=	a/=2	a=a/2

5. INCREMENT & DECREMENT OPERATORS.

- Increase and decrease by 1.

i++	INCREMENT
--i	DECREMENT

EXAMPLE.

```
Int x=10;
x++; // 11
--x; // 10
```

6. BITWISE OPERATPRS.

- Work at bit level.

&	AND
|	OR
^	XOR
~	NOT
<<	LEFT SHIFT
>>	RIGHT SHIFT

7. TERNARY OPERATOR

- SHORTHAND FOR IF-ELSE.

SYNTAX

Condition? value_if_true : value_if_false;

EXAMPLE.

Int max = (a > b)? a: b;

8. SPECIAL OPERATOR.

- SOME C-SPECIFIC ONES.

Sizeof	Size of variable or type
&	Address of
*	Pointer dereferences
,	Comma operator

5. CONTROL FLOW STATEMENTS.

- Control statements control the flow of execution of a program.
- By default, c executes code line by line (top to bottom).

- Control statements allow,

    - 1. Decision making.
    - 2. Repetition.
    - 3. Jumping to another part of program.

1. DECISION MAKING STATEMENTS OR CONDITIONAL STATEMENTS

- Conditional statements allow a program to decisions and execute different code block based on condition.

TYPES OF CONDITONAL STATEMENTS.

1. if
2. if-else
3. else-if ladder
4. nested if
6. switch
7. conditional

1. if

- Execute code only if the condition is true.

Syntax.

```
If(condition)
{
// statement;
}
```

2. if-else

- chose between two alternatives.

Syntax.

```
If(condition)
{
   // true block
}
Else
{
   // false block
}
```

3. else-if ladder.

- Used when multiple conditions must be checked.

Syntax.

```
If(condition-1){
}
Else if(condition-2){
}
else if(condition-3){
}
Else{
}
```

4. Nested – if.

- if statement inside another if statement.

Syntax.

```
If(condition-1){
    If(condition-2){
    }
}
```

5. switch statement.

- used to choose one block of code from many options based on the value of an expression.

Syntax.

```
switch (expression) {
  case constant1:
    // code
    break;

  case constant2:
    // code
    break;

  default:
    // code if no case matches
}
```

6. condition statement.

- A short form of if-else.

Syntax.

```
(condition) ? expression1 : expression2;
```

6. LOOPS IN C.

- A loop is used to repeats the same code again and again.
- It saves time and reduces code length.
- Loops make program easier to write and understand.

There are two types of loops in c.

1. Entry controlled loops

   - A condition is checked before executing loop.
   - It also called as a pre-checking loop.

2. Exit controlled loops.

   - A condition is checked after executing loop.
   - It is called as a post-checking loop.

- ENTRY CONTROLLED LOOPS.

1. for loop
2. while loop

1. for loop

- A for loop is used to repeat a block of code when the number of repetitions is known.

SYNTAX.

```
For(initialization ;condition ;increment )
{
    // block of code
}
```

2. while loop

- A while loop is used when you want to repeat a block of code as long as a given condition is true.

SYNTAX.

```
While (condition) {
        // block of code
        // increment    }
```

- EXIT CONTROLLED LOOPS.

1. do-while loop.

- A do-while loop is similar to a while loop, but the loop body runs at least once, no matter what.

- condition is checked after the loop body execute.

SYNTAX.

```
Do {
    // block of code.
}
While(condition)
```

7. LOOP CONTROL STATEMENTS.

- Loop control statements in c are used to control the execution of loop by altering the normal sequence of execution.

THERE ARE THREE TYPES OF LOOP CONTROL STATEMENTS IN C.

1. Break.

- break statement terminates the loop immediately.

- control moves to the statement after the loop.

SYNTAX.

Break;

2. Continue.

- continue is used to skips the remaining code in the current iteration.

- Loop continues with the next iteration.

SYNTAX.

Continue;

3. goto.

- By using this goto statements we can transfer the control from current location to anywhere in the program.

- To do all this we have to specify a label with goto and the control will transfer to the location where the label is specified.

8. FUNCTION IN C.

- A function is a block of code that perform a specific task.

- There are two types of function in c.

1. Library – function.
2. User – define function.

- Types of library function.

- library function is pre-defined in c.
- main (), int main (), printf (), scanf ().

- Types of users define function.

- A users define function is defined by user.

- function with parameter, with return type.

- function with parameter, without return type.

- function without parameter, with return type.

- function without parameter, without return type.

- FUNCTION DECLARATION.

SYNTAX.

Return_type function_name (parameter);

EXAMPLE.

Int add (int a, int b);

// int = return type // add=function name // int = a, b = parameter.

9. ARRAY IN C.

- Array is a group of elements having same data type.

- ARRAY DECLARATION.

```
Int numbers [ 5 ];
```

Int = data type.

Numbers = array name.

[5] = array size.

- Initialization an array.

```
Int number [ 5 ] = { 1 , 2 , 3 , 4 , 5 };
```

THERE ARE TWO TYPES OF ARRAYS IN C.

1. One - dimensional array.
2. Multi – dimensional array.

One – Dimensional	Multi – Dimensional
An array with single index.	An array with more than one index.
Linear list of elements.	Elements arranged in rows, columns etc.
Data_type arr [size];	Data_type arr [row][column];
Int a [5];	Int a [2] [3];

10. POINTERS IN C.

- Pointer in c are variables that store the memory address of another variable.

PONTER DECLARATION AND INITIALIZATION.

- In c, Pointers are declared using the ‘ * ’ symbol, and they must be initialized to a valid memory address before use.

POINTER DECLARATION.

```
Int *ptr;    // Declares a pointer to an integar.
```

POINTER INITIALIZATION.

- You can initialize a pointer by assigning it the address of a variable using the address of operator “ & ”.

```
Int x = 10;
Int *ptr = & x;  //  pointer ptr holds the address of variable x.
```

- Now, ptr points to the memory location where x is stored.
- You can access the value of x through *ptr.

WHY ARE POINTERS IMPORTANT IN C.

- Pointers give direct memory location.
- Enable dynamic memory.
- Power array and string.
- Make c fast and efficient.

11. STRING IN C.

- String is a group of character and, ends with NULL character '\0'.

SYNTAX.

Char string_name[length];

- The compiler automatically place '\0' at the end of the string. When we initialize the array.

H	A	R	S	H	I	L	\0

NULL CHARACTER.

FUNCTIONS OF STRING.

- strcpy()
- strlen()
- strcmp()
- strcat()

strcpy() :-

- strcpy() function is used to copies one string to another.

strlen() :-

- strlen() function is find the length of the string.

strcmp() :-

- strcmp() function is used to compare the two string, if the string match, the function return zero 0.

Strcat() :-

- strcat() function is used to append one string to another, creating a single string out of two.

12. STRUCTURES IN C.

- In c, Structure is a user-defined data type, that allow grouping different types of data under a single name.

- Each data elements in a structure are called a member, and they can be of different type, such as integer, float, or arrays.

- Structure is used to represent complex data entities, such as records in data base.

FUNCTION DECLARATION.

- Functions are defined using the ‘ STRUCT ’ keyword.

```
Struct structure_name {
        Data_type member 1;
        Data_type member 2;
}
```

STRUCTURE INITIALIZATION.

- A structure variable can be initialized separately, directly or using array.

- Separately.

S1.age=20;

- Directly.

Struct student s2={“harshil”,20};

- using array.

Struct student[2]={{“harshil”,20},{“hello”,21}};

ACCESING STRUCTURE MEMBER.

- Using dot (.) operator.

- s1.name;

13. FILE HANDLING.

- File handling in c, is the process of creating, reading, writing, and closing file.

- A file permanently store the data on a disk. Rather than losing it when program is terminated.

- This process function from standard input/output library.

- A special type of pointer FILE.

IMPORTANCE OF FILE HANDLING.

- Permanent data storage.
- Handles large data.
- Data sharing.
- Data backup and recovery.
- Used in real applications.

FILE OPERATIONS IN C.

- Use the file structure from standard input/output library.

- <stdio.h>

1. fopen() :-

- opening a file.

- A file is open using fopen(“filename”, “mode”);

COMMON FILE MODES.

R	Open for reading
W	Open for writing
A	Open for appending
R+	Read or write.
W+	Read or write.
A+	Read or append

2. fclose() :-

- Used to Close a file

EXAMPLE.

Fclose(fp);

3. WRITING TO A FILE.

- Using fprintf (), fputc (), fputs ().

4. REDING FROM A FILE.

- Using fscanf (), fgetc (), fgets ().