B.Sc. (I.T.) / M.Sc. (I.T.) 1s Semester

Course: 104: Fundamentals of Programming Using C-1

2.1 Introduction to Computer Programming Language and Program

Definition:
A Computer Program is a set of instructions written in a programming language to instruct the computer to perform a specific task.

Programming Language:
It is a formal language used to communicate with a computer to develop programs.

Why Programming is Needed?
- To automate tasks.
- To solve real-life problems (billing, banking, simulations).
- To perform calculations and data processing.

Example Program (in C):

```
#include <stdio.h>
int main() {
   printf("Hello, World!");
   return 0;
}
```

👉 This is the simplest program that prints "Hello, World!".

2.2 Programming Languages and Levels

Programming languages can be categorized into different levels:

1. **Machine Language (1st Generation)**
   - Written in binary (0s and 1s).
   - Example: 10110000 01100001
   - Very difficult for humans.

2. **Assembly Language (2nd Generation)**
   - Uses mnemonics (symbols) instead of binary.
   - Example: MOV A, 5
   - Easier than machine language but still hardware dependent.

3. **High-Level Languages (3rd Generation)**
   - Similar to English, easy to understand.
   - Example: C, C++, Java, Python.
   - Require a translator (compiler/interpreter).

4. **Very High-Level Languages (4th Generation)**
   - Problem-oriented, closer to human language.
   - Example: SQL, MATLAB.

5. **5th Generation Languages (Artificial Intelligence)**
   - Used in AI, Machine Learning, Expert Systems.
   - Example: Prolog, LISP.

2.3 Language Translators

Since computers understand only machine language, programs written in other languages need to be translated.

2.3.1 Compiler

- Translates entire program at once into machine code.
- Generates an object file.
- Faster execution.
- Example: C Compiler (GCC).
- Error Handling: All errors shown after full compilation.

2.3.2 Interpreter

- Translates line by line into machine code.
- Slower execution.
- Easier debugging (error shown immediately).
- Example: Python Interpreter.

2.3.3 Assembler

- Translates assembly language → machine language.
- Example: MASM (Microsoft Assembler).

2.4 Program Verification

After writing a program, it must be tested for correctness.

2.4.1 Program Correctness

- A program is correct if it gives the desired output for all possible inputs.
- Two types of correctness:
  1. Partial correctness: Correct for some inputs.
  2. Total correctness: Correct for all valid inputs.

2.4.2 Program Bugs & Testing

- Bug: An error or fault in a program that causes incorrect output.
    - Example: Using = instead of == in C.
- Debugging: Process of finding and fixing bugs.
- Testing: Running a program with different inputs to check its correctness.

Types of Testing:

1. Unit Testing – Testing individual modules.
2. Integration Testing – Testing combined modules.
3. System Testing – Testing entire program in real environment.

Quick Revision Notes

- **Program** = Set of instructions.
- **Language Levels:**
    - Machine → Assembly → High-Level → Very High-Level → 5th Gen (AI).
- **Translator Types:** Compiler (whole program), Interpreter (line by line), Assembler (assembly → machine).
- **Verification:** Correctness, Bugs, Debugging, Testing.
- **Classic Exam Q:** Differences between Compiler & Interpreter.