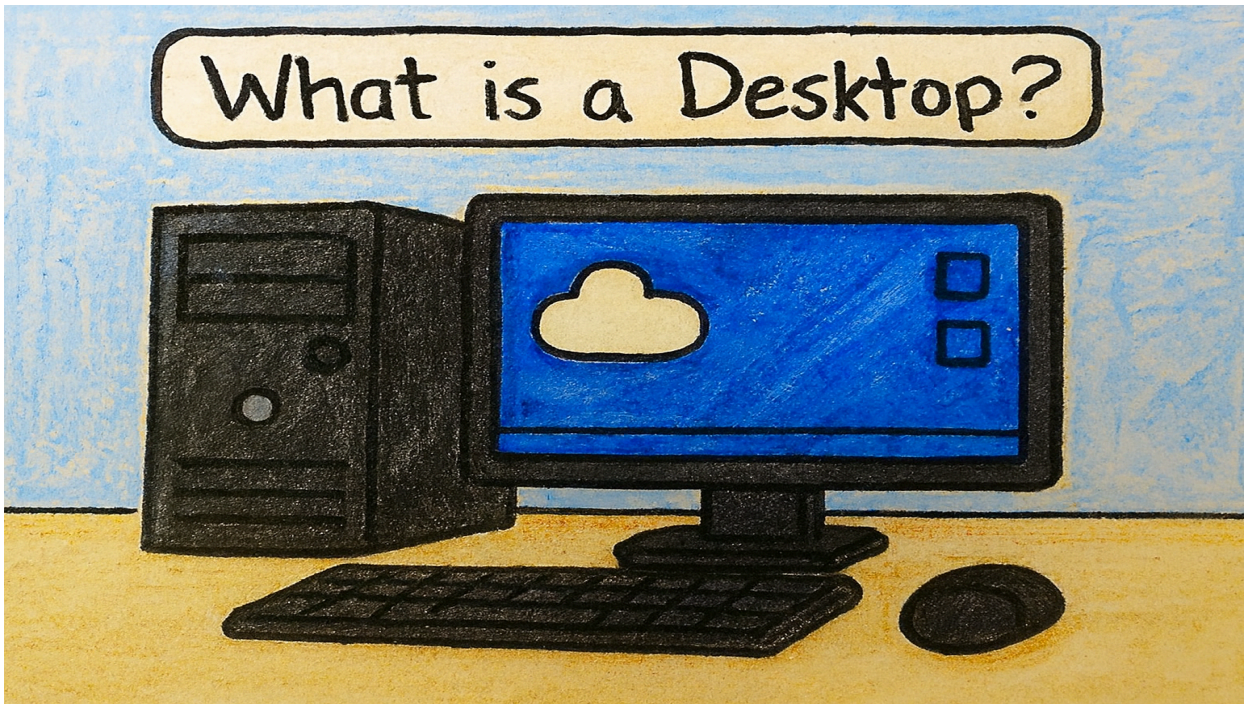


Class: 12th

Subject: Computer

Chapter 8: GETTING STARTED WITH C

📍 Important MCQs:

1. A computer works on the basis of:

(a) Data

(b) Instructions ✅

(c) Programs

(d) Hardware

2. A well-defined set of instructions is called:

(a) Data

(b) Program ✅

(c) Software

(d) Algorithm

3. A computer program is written in:

(a) Machine language

(b) Binary code

(c) Programming language ✅

(d) Assembly language

4. C programming language was developed by:

(a) Ken Thompson

(b) Bill Gates

(c) Dennis Ritchie ✅

(d) Charles Babbage

5. C language was developed in:

(a) 1969

(b) 1970

(c) 1972 ✅

(d) 1985

6. C language was developed at:

(a) IBM

(b) Microsoft

(c) AT & T Bell Laboratories ✅

(d) Apple

7. C language was derived from:

(a) BASIC

(b) FORTRAN

(c) B language ✅

(d) Pascal

8. B language was developed by:

(a) Dennis Ritchie

(b) Ken Thompson ✅

(c) Brian Kernighan

(d) James Gosling

9. C was originally designed to write:

(a) Business programs

(b) Game programs

(c) System programs ✅

(d) Web applications

10. C language was designed for which operating system?

(a) Windows

(b) UNIX ✅

(c) DOS

(d) Linux

11. The earlier version of C is known as:

(a) ANSI C

(b) Turbo C

(c) K&R C ✅

(d) Standard C

12. K&R stands for:

(a) Ken & Ritchie

(b) Kernel & Ritchie

(c) Kernighan & Ritchie ✅

(d) Knowledge & Research

13. ANSI C was standardized by:

(a) ISO

(b) IEEE

(c) ANSI ✅

(d) AT & T

14. A program written in C is called:

(a) Object program

(b) Machine program

(c) Source program ✅

(d) Executable program

15. Turbo C++ is a product of:

(a) Microsoft

(b) Borland International ✅

(c) IBM

(d) Apple

16. Turbo C++ provides:

(a) Only editor

(b) Only compiler

(c) IDE ✅

(d) Operating system

17. The default name given by Turbo C++ while saving is:

(a) FILE00.c

(b) NONAME00.cpp ✅

(c) PROGRAM.c

(d) TEMP.c

18. The default folder for saving files in Turbo C++ is:

(a) TC

(b) BIN ✅

(c) SOURCE

(d) DATA

19. To compile a C program in Turbo C++, press:

(a) F2

(b) Ctrl + F9

(c) Alt + F9 ✅

(d) Alt + F

20. The compiler converts source program into:

(a) Text file

(b) Machine language

(c) Object program ✅

(d) Executable file

21. Linking in C is the process of:

(a) Editing the program

(b) Compiling the program

(c) Combining object file with library files ✅

(d) Executing the program

22. The program that performs linking is called:

(a) Compiler

(b) Loader

(c) Linker ✅

(d) Editor

23. The linker produces a file with extension:

(a) .c

(b) .obj

(c) .exe ✅

(d) .txt

24. In Turbo C++, linking is done through:

(a) Run | Run

(b) Compile | Compile

(c) Compile | Link ✅

(d) File | Save

25. Which file is produced by the compiler before linking?

(a) Source file

(b) Executable file

(c) Object file ✅

(d) Text file

26. Executing a program means:

(a) Writing the program

(b) Compiling the program

(c) Loading program into memory ✅

(d) Saving the program

27. The program that loads executable file into memory is called:

(a) Compiler

(b) Linker

(c) Loader ✅

(d) Editor

28. To run a program in Turbo C++, press:

(a) Alt + F9

(b) Ctrl + F9 ✅

(c) Alt + F5

(d) F2

29. To view output screen in Turbo C++, press:

(a) Ctrl + F9

(b) Alt + F9

(c) Alt + F5 ✅

(d) Ctrl + F5

30. By default, Turbo C++ stores object and executable files in:

(a) INCLUDE folder

(b) LIB folder

(c) BIN folder ✅

(d) SOURCE folder

31. The basic structure of C program includes:

(a) Only main function

(b) Only header files

(c) Preprocessor directives and main function ✅

(d) Variables only

32. C is a:

(a) Unstructured language

(b) Object-oriented language

(c) Structured programming language ✅

(d) Machine language

33. Execution of a C program starts from:

(a) printf function

(b) First statement

(c) main function ✅

(d) Header file

34. Preprocessor directives always begin with:

(a) @

(b) $

(c) # ✅

(d) %

35. Which of the following is a preprocessor directive?

(a) printf

(b) main

(c) #include ✅

(d) scanf

36. The header file for standard input/output functions is:

(a) math.h

(b) conio.h

(c) stdio.h ✅

(d) string.h

37. The #define directive is used to:

(a) Declare variables

(b) Define functions

(c) Define constant macros ✅

(d) Include files

38. The symbol used to terminate a statement in C is:

(a) ,

(b) :

(c) ; ✅

(d) .

39. Braces { } in C are called:

(a) Operators

(b) Identifiers

(c) Delimiters ✅

(d) Constants

40. The printf function is used to:

(a) Take input

(b) Perform calculations

(c) Display output on screen ✅

(d) Compile program

41. Errors in a computer program are called:

(a) Viruses

(b) Bugs ✅

(c) Faults

(d) Crashes

42. The process of finding and removing bugs is called:

(a) Editing

(b) Compiling

(c) Debugging ✅

(d) Linking

43. How many types of programming errors are there in C?

(a) Two

(b) Three ✅

(c) Four

(d) Five

44. Which of the following is NOT a type of programming error?

(a) Syntax error

(b) Runtime error

(c) Logic error

(d) Compile-time error ✅

45. A syntax error is detected by:

(a) Programmer

(b) Compiler ✅

(c) Linker

(d) Loader

46. Missing semicolon (;) causes:

(a) Runtime error

(b) Logic error

(c) Syntax error ✅

(d) Hardware error

47. Dividing a number by zero causes:

(a) Syntax error

(b) Logic error

(c) Runtime error ✅

(d) Compile-time error

48. Logical errors are detected by:

(a) Compiler

(b) Computer

(c) Programmer through testing ✅

(d) Linker

49. Which error does NOT stop program execution but gives wrong output?

(a) Syntax error

(b) Runtime error

(c) Logical error ✅

(d) Link error

50. Programming languages are broadly divided into:

(a) Three categories

(b) Four categories

(c) Two categories ✅

(d) Five categories

51. Machine language consists of:

(a) English words

(b) Symbols

(c) Binary digits (0s and 1s) ✅

(d) Numbers and letters

52. Assembly language uses English-like words called:

(a) Keywords

(b) Operators

(c) Mnemonics ✅

(d) Identifiers

53. Assembly language is translated into machine language by:

(a) Compiler

(b) Interpreter

(c) Assembler ✅

(d) Linker

54. High level languages are:

(a) Close to machine language

(b) Difficult to learn

(c) English-like languages ✅

(d) Hardware dependent

55. Which of the following is a high-level language?

(a) Machine language

(b) Assembly language

(c) C language ✅

(d) Binary code

📍 Important Short Questions:

1. What is a computer program?

Answer:

👉 A computer program is a well-defined set of instructions given to a computer to perform a specific task.

Example: A program that calculates students' marks.

2. What is a programming language?

Answer:

👉 A programming language is a language used to write instructions so that a computer can understand and execute them.

Example: C, C++, Java.

3. Who developed the C programming language?

Answer:

👉 The C programming language was developed by Dennis Ritchie.

Example: C language was developed at AT & T Bell Laboratories.

4. In which year was C language developed?

Answer:

👉 C language was developed in the year 1972.

Example: It was developed for system programming.

5. From which language was C derived?

Answer:

👉 C language was derived from the B language.

Example: B language was developed by Ken Thompson.

6. What is Turbo C++?

Answer:

👉 Turbo C++ is a compiler developed by Borland International used to write and execute C programs.

Example: Turbo C++ is commonly used in educational institutions.

7. What is an IDE?

Answer:

👉 IDE (Integrated Development Environment) is software that provides editor, compiler, and debugger in one place.

Example: Turbo C++ IDE.

8. What is a compiler?

Answer:

👉 A compiler is a program that converts a source program into an object program or machine language.

Example: C compiler.

9. What is a source program?

Answer:

👉 A source program is a program written in a high-level language such as C.

Example: A program saved with .c extension.

10. What is an object program?

Answer:

👉 An object program is the translated form of a source program produced by the compiler.

Example: A file with .obj extension.

11. What is linking in a C program?

Answer:

👉 Linking is the process of combining the object file produced by the compiler with library files to create an executable file.

Example: Combining hello.obj with standard library files to produce hello.exe.

12. What is the function of a linker?

Answer:

👉 A linker combines object files and library files to produce a final executable file.

Example: Linking math.obj with stdio.lib to run a math program.

13. Which file is produced after linking a C program?

Answer:

👉 An executable file with .exe extension is produced.

Example: program.exe.

14. How can the linker be invoked in Turbo C++?

Answer:

👉 By selecting Compile → Link from the menu bar.

Example: Click Compile → Link to link example.obj into example.exe.

15. What is meant by executing a C program?

Answer:

👉 Executing a program means loading the executable file into memory and running it.

Example: Running hello.exe to display output on screen.

16. What is a loader?

Answer:

👉 A loader is a program that loads the executable file into memory for execution.

Example: Turbo C++ loader loads hello.exe for running.

17. Which key is used to run a program in Turbo C++?

Answer:

👉 Press Ctrl + F9 to run the program.

Example: Run test.c using Ctrl + F9.

18. How can you view the output screen in Turbo C++?

Answer:

👉 Select Window → User Screen or press Alt + F5.

Example: After running, press Alt + F5 to see "Hello World" output.

19. What is the default directory for object and executable files in Turbo C++?

Answer:

👉 The default directory is the BIN subdirectory of the TC folder.

Example: C:\TC\BIN\.

20. Why should output and source directories be set to the same folder?

Answer:

👉 To keep source, object, and executable files together for easy management.

Example: D:\MyPrograms\ for hello.c, hello.obj, hello.exe.

21. What is the basic structure of a C program?

Answer:

👉 A C program has preprocessor directives and main function, enclosed in braces { }.

Example:

C

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

```
void main() {
   printf("Hello World");
}
```

22. What is a preprocessor directive?

Answer:

👉 Commands that give instructions to the C preprocessor before compilation, starting with #.

Example: #include <stdio.h>.

23. What is the purpose of the #include directive?

Answer:

👉 It includes standard library functions in the program.

Example: #include <math.h> allows using sqrt() function.

24. What is the main function in C?

Answer:

👉 main() is the entry point of a C program where execution starts.

Example: Every program must have:

C

void main() { printf("Hello"); }

25. What is the function of printf in C?

Answer:

👉 printf() is used to display output on the screen.

Example: printf("Hello World"); prints Hello World on the screen.

26. What are programming errors in C?

Answer:

👉 Programming errors, or "bugs," are mistakes in a program that prevent it from running correctly. They can occur in the program's logic, syntax, or execution.

Example: printf("Hello World") (semicolon missing) or int a = 5/0; (illegal operation).

27. What is debugging?

Answer:

👉 Debugging is the process in which a programmer finds and fixes errors (bugs) in a program.

Example: Correcting int a = 5/0; to prevent division by zero.

28. What is a syntax error?

Answer:

👉 A syntax error occurs when a program violates the grammar rules of C. The compiler detects it, and the program cannot run.

Example: printf("Hello World") instead of printf("Hello World"); (semicolon missing).

29. Give some causes of syntax errors.

Answer:

👉 **Common causes:**

- Missing semicolon (;)
- Using an undeclared variable
- Missing { or } braces

Example: int a = 5 (semicolon missing).

30. What is a runtime error?

Answer:

👉 A runtime error occurs when the program performs an illegal operation during execution. The program stops and shows an error message.

Example: int a = 5/0; ⟶ Division by zero.

31. What happens when a runtime error occurs?

Answer:

👉 When a runtime error occurs:

- Program execution stops immediately
- A diagnostic message is displayed
- **Example**: Trying to open a file that does not exist → runtime error.

32. What is a logical error?

Answer:

👉 A logical error occurs when a program runs correctly but produces wrong output because of faulty logic.

Example: area = length + width; instead of area = length * width;.

33. Can the compiler detect logical errors?

Answer:

👉 No, the compiler cannot detect logical errors. They are recognized only by checking the output.

Example: Using the wrong formula in a calculation.

34. What are low-level programming languages?

Answer:

👉 Low-level languages are close to machine hardware.

Types:

- Machine language (binary code)
- Assembly language (mnemonics)
- **Example**: 10110000 01100001 (machine) or MOV A, 61H (assembly).

35. What is machine language?

Answer:

👉 Machine language is the native language of a computer, consisting of binary 0s and 1s. The computer can understand it directly.

Example: 11001010 10101100 ⟶ a CPU instruction.

36. What is assembly language?

Answer:

👉 Assembly language uses English-like mnemonics instead of binary code, making programming easier for humans.

Example: MOV A, 61H (move value 61H to register A).

37. What is an assembler?

Answer:

👉 An assembler is a translator program that converts assembly language into machine language.

Example: MOV A, 61H ⟶ 10110000 01100001 (binary machine code).

38. What are high-level programming languages?

Answer:

👉 High-level languages use English-like instructions, making them easy to write, read, and debug.

Example: C, C++, Java: printf("Hello World");.

39. Name some common high-level languages.

Answer:

👉 Common high-level languages include:

- C, C++, Java
- Pascal, FORTRAN, BASIC, COBOL

Example: scanf("%d",&a); (C input statement).

40. What are the advantages of high-level languages?

Answer:

👉 Advantages:

- Easy to learn, read, modify, and debug
- Focus on problem-solving rather than hardware
- Machine independent (can run on different computers with small changes)

Example: A C program can run on Intel and Motorola processors with minor modifications.

📍 Exercise 8c

1. Fill in the blanks:

(i) The set of instruction given to the computer to solve any kind of problem is called -------.

Answer: Program

Explain: 👉 A program is a sequence of instructions written to perform a specific task on a computer.

(ii) ANSI stands for -------.

Answer: American National Standards Institute

Explain: 👉 ANSI is the organization that developed the standard version of C, called ANSI C, to ensure consistency across different platforms.

(iii) The program written in high level language is known as ---------.

Answer: Source program

Explain: 👉 A source program is written in a high-level language like C and must be translated into machine language by a compiler or interpreter.

(iv) ------ is a program that places the executable file in memory

Answer: Loader

Explain: 👉 The loader loads the executable (.exe) file into the computer's memory so that the CPU can execute it.

(v) In -------- programming language the entire logic of the program is implemented in a single module.

Answer: Unstructured

Explain: 👉 In unstructured programming, all code is written in a single block, making it difficult to debug, modify, or understand.

(vi) -------- are commands that give instruction to C preprocessor

Answer: Preprocessor directives

Explain: 👉 Preprocessor directives, starting with #, instruct the C preprocessor to modify or include files before compilation.

(vii) ------- is a name that is replaced by particular constant before program is sent to the compiler.

Answer: Constant macro

Explain: 👉 A constant macro is defined using #define and is replaced by its value wherever it appears in the program.

(viii) The ------- directive gives a program access to a library file

Answer: #include

Explain: 👉 The #include directive tells the compiler to include the contents of a standard or user-defined library file in the program.

(ix) C program is divided into units, called -------

Answer: Functions

Explain: 👉 Functions are blocks of code in C that perform a specific task, helping to organize programs efficiently.

(x) Every statement in a C program terminates with a -------

Answer: Semicolon (;)

Explain: 👉 The semicolon marks the end of a statement in C. Omitting it causes a syntax error.

(xi) A language translator for Assembly language is called -------

Answer: Assembler

Explain: 👉 An assembler converts assembly language code, which uses mnemonics, into machine code executable by the computer.

(xii) A set of rules for writing program in high level language is known as ---------

Answer: Syntax

Explain: 👉 Syntax defines the correct structure and rules of a programming language, ensuring the compiler can understand the program.

2. Choose the correct option:

(i) C is a:

(a) High Level Language ✅

(b) Low Level Language

(c) Assembly Language

(d) Machine Language

(ii) Turbo C++ can compile:

(a) C++ programs only

(b) C and C++ programs ✅

(c) Turbo C programs only

(d) Turbo C++ programs only

(iii) Debug is the process of:

(a) Creating bugs in program

(b) Identifying and removing errors ✅

(c) Identifying Errors

(d) Removing Errors

(iv) C was designed to write programs for:

(a) Windows operating system

(b) Solaris operating system

(c) Unix operating system ✅

(d) OS/2 operating system

(v) Preprocessor directives are commands for:

(a) Microprocessor

(b) Language processor

(c) C preprocessor ✅

(d) Loader

(vi) The expression in define directive:

(a) can only be changed at the end of the program

(b) can not be changed ✅

(c) can not be changed but can be redefine

(d) can not be assigned a value

(vii) Which of the following language requires no translator to execute the program:

(a) C

(b) C++

(c) Machine language ✅

(d) Assembly language

(viii) .exe file is produced by the:

(a) Linker ✅

(b) Loader

(c) Compiler

(d) Interpreter

(ix) Which of the following key is used to save a file?

(a) F2 ✅

(b) F3

(c) F5

(d) F9

(x) void occupies how many bytes in memory?

(a) zero ✅

(b) one

(c) two

(d) four

3. Write T for true and F for false statement:

(i) The C programming language was developed by Dennis Ritchie in 1972.

Answer: **T** ✅

(ii) C was derived from earlier programming language named B.

Answer: **T** ✅

(iii) The B was developed by Ken Thomson in 1980.

Answer: **F** ❌ (Correct year: 1969–70)

(iv) The short-key for compiling a program is Alt+F9.

Answer: T ✅

(v) The compiler produces the source file from an object file.

Answer: F ❌ (Compiler produces object file from source file)

(vi) The linker is a program that combines the object program with additional files.

Answer: **T** ✅

(vii) The short-key for executing the C program is Alt + F5.

Answer: **F** ❌ (Correct key: Ctrl + F9)

(viii) In structured programming the entire program is divided into smaller modules.

Answer: **T** ✅

(ix) The value of a constant can be changed during the program execution.

Answer: **F** ❌ (Constant value cannot be changed during execution)

(x) High level language provide machine independence.

Answer: **T** ✅

🌟 Q.4: Briefly describe the history of C (Detailed Explanation)

❖ **Answer:**

The C programming language is one of the most influential programming languages in the world. Its history is as follows:

1. Development by Dennis Ritchie:

- C was developed by Dennis Ritchie in 1972 at AT&T Bell Laboratories.
- It was designed to provide a flexible and efficient language for system programming, especially for developing the UNIX operating system.

2. Origin from B Language:

- C was derived from an earlier programming language called B.
- The B language was developed by Ken Thompson in 1969–1970.
- B was simple but limited in functionality, so C was created to overcome these limitations and provide more features like data types, structures, and efficient memory management.

3. K&R C:

- The first widely used version of C was known as K&R C, named after Brian Kernighan and Dennis Ritchie who documented it in their famous book.
- K&R C laid the foundation of how C programs are written and compiled.

4. ANSI C Standard:

- Later, to ensure uniformity and standardization, the American National Standards Institute (ANSI) developed a standardized version called ANSI C.
- ANSI C made C programming consistent across different compilers and systems.

5. Importance and Use:

- Initially designed for system programming, C's power, efficiency, and flexibility made it popular for a wide range of applications including software development, embedded systems, and application programming.
- C influenced many later languages, including C++, Java, C#, and Python.

✦ **Summary:**

- Developed by Dennis Ritchie in 1972
- Derived from B language (Ken Thompson, 1969–70)
- K&R C was the first widely used version
- ANSI standardized C as ANSI C
- Designed for UNIX but widely used in industry

🌟 **Q.5: List two reasons why it would be preferable to write a program in C rather than machine language**

❖ **Answer:**

1. Readability and Understandability:

- Machine language consists of long sequences of 0s and 1s, which are very difficult for humans to read, write, or interpret.
- C is a high-level language that uses English-like instructions, keywords, and syntax. This makes the program easier to understand and follow, even for complex tasks.

Example: Writing printf("Hello World!"); in C is much easier to understand than writing its equivalent in machine code.

2. Ease of Writing, Debugging, and Modifying Programs:

- Writing programs in machine language is extremely time-consuming and prone to errors because each instruction must be manually encoded in binary.
- In C, you can write programs faster and with fewer errors.
- C also provides structured programming features, modularity (functions), and debugging tools like compilers and IDEs, which make it easier to locate and fix errors.
- Programs written in C can be easily modified, extended, or reused, unlike machine language programs which are rigid and hard to maintain.

✦ **Summary:**

Writing in C is preferable to machine language because it is human-readable, less error-prone, faster to develop, and easier to maintain and debug. Machine language is difficult, time-consuming, and prone to mistakes.

🌟 6. What necessary steps are taken to prepare a C program for execution? Explain with diagram.

❖ **Answer:**

To prepare a C program for execution, the following steps are taken:

Step 1: Writing the Source Program

- The programmer writes the C program using a text editor or IDE (e.g., Turbo C++).
- The program is saved with a .c extension.

Example: hello.c contains the code:

C

```
#include <stdio.h>

void main() {
```

```
    printf("Hello World!");
}
```

Step 2: Compiling the Program

- The compiler translates the source program (.c) into object code (.obj).
- Compilation checks for syntax errors and produces an object file if no errors are found.
- Shortcut in Turbo C++: Alt + F9

Step 3: Linking the Program

- Linking is the process of combining the object file with library files required for input/output and other functions.
- The linker produces an executable file (.exe) that can run on the computer.
- Shortcut in Turbo C++: Compile ⟶ Link

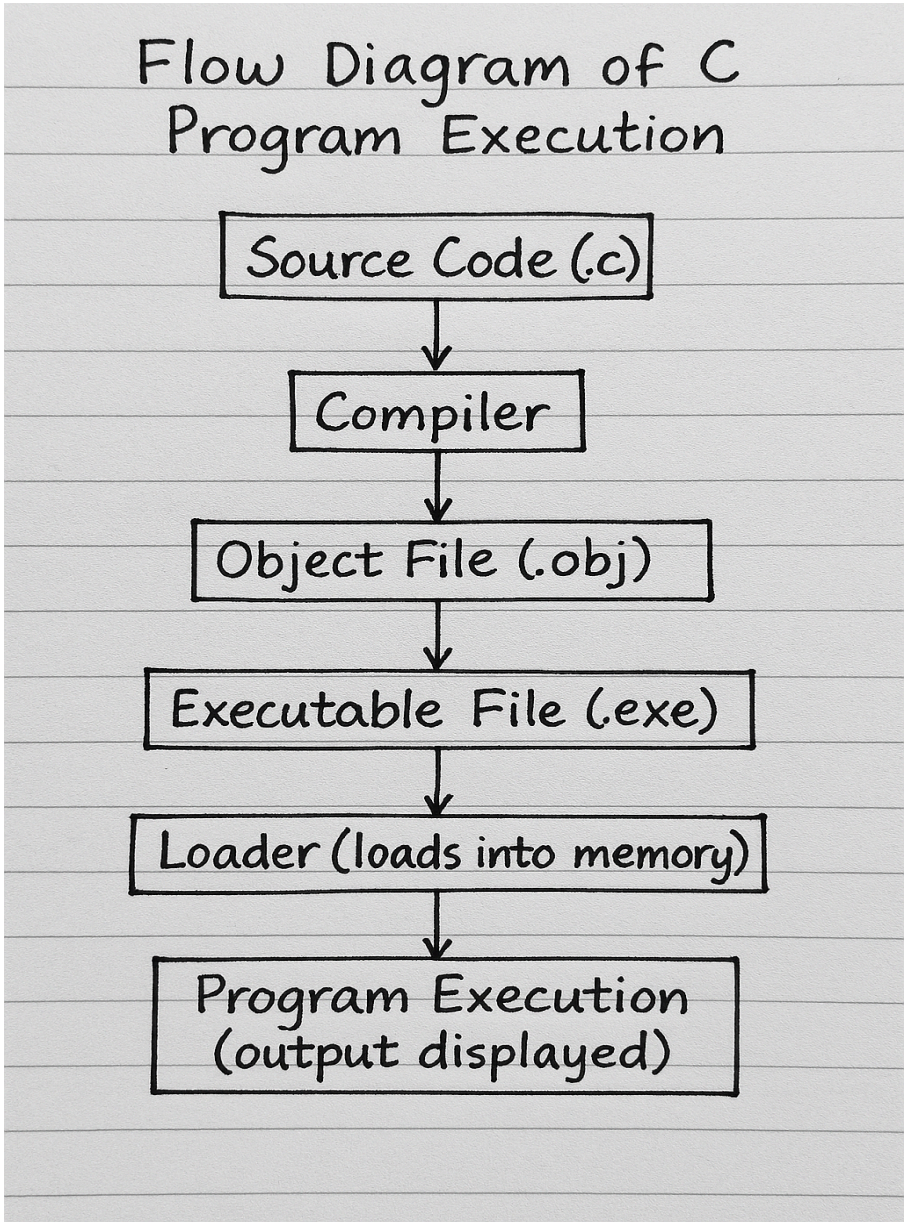
Step 4: Loading the Program

- The loader places the executable file into memory so it can be run by the CPU.
- Shortcut in Turbo C++: Ctrl + F9

Step 5: Executing the Program

- The program starts executing from the main() function.
- Output is displayed on the screen using functions like printf().
- In Turbo C++, to view the output: Alt + F5

✦ **Digram:**



✦ **Summary:**

- Write source code ⟶ 2. Compile to object file ⟶ 3. Link with libraries ⟶ 4. Load into memory ⟶ 5. Execute program.
- This flow ensures that a human-readable C program is translated into machine-readable instructions and executed correctly.

🌟 **Q.7: Define a bug. Discuss some debugging features of Turbo C++.**

❖ **Answer:**

Definition of Bug:

A bug is an error or fault in a computer program that prevents it from running correctly or produces incorrect results. Finding and removing these errors is called debugging.

Debugging Features of Turbo C++

Turbo C++ provides several tools to help detect and fix errors in a program:

1. Error Messages

- The compiler displays error messages when it detects syntax errors during compilation.
- **Example:** Missing semicolon, undeclared variable, etc.

2. Breakpoint

- A breakpoint is a marker set on a line of code where execution will pause.
- This allows the programmer to examine the program's state and variables at that point.

3. Step Execution

- Allows executing the program one statement at a time to observe how each statement affects the program.

4. Watch Variables

- Enables monitoring the value of variables during program execution.
- Helps in detecting logical or runtime errors.

5. Call Stack

- Shows the sequence of function calls at a given point during execution.
- Useful for tracing the program flow and locating errors.

6. Run-Time Error Messages

- Turbo C++ displays messages when a runtime error occurs, like division by zero or invalid memory access.

✦ **Summary:**

- A bug is a program error that affects correct execution.
- Turbo C++ provides features like error messages, breakpoints, step execution, watch variables, call stack, and runtime error messages to make debugging easier and faster.

🌟 **Q.8: While writing a C program, how many types of errors can occur? Discuss briefly. Which one is the most difficult to locate and remove? Justify your answer.**

❖ **Answer:**

When writing a C program, the programmer may come across errors called bugs, and finding and removing them is called debugging. There are three types of programming errors:

1. Syntax Errors

- Occur when the program violates the grammar rules of C language.

- Detected by the compiler during translation.

Example causes:

- Missing semicolon (;)
- Using a variable without declaration
- Missing braces { }

2. Runtime Errors

- Occur when the program directs the computer to perform an illegal operation.
- Detected by the computer during execution.
- **Example:** Dividing a number by zero.
- **Effect:** The program stops, and a diagnostic message is displayed.

3. Logical Errors

- Occur when the program follows a faulty algorithm.
- Not detected by the compiler, so no error message appears.
- They do not crash the program, making them difficult to find.
- Can be recognized by observing wrong output.
- Most Difficult to Locate and Remove

- Logical errors are the most difficult because the compiler cannot detect them and the program may still run without crashing. Only thorough testing can find these errors.

✦ **Summary:**

- **Errors in C:** Syntax, Runtime, Logical
- **Most difficult:** Logical errors because they require careful testing to detect.

🌟 **Q.9: What is a programming language? Discuss the two main categories of programming languages.**

❖ **Answer:**

Programming Language

A programming language is a formal language used to write computer programs. It allows a programmer to give a set of instructions to the computer to perform specific tasks. These instructions are later translated into machine language so that the computer can understand and execute them.

Two Main Categories of Programming Languages

1. Low-Level Languages

- These languages are closer to machine language and less understandable for humans.

Divided into two types:

a) Machine Language:

- Native language of the computer (binary code: 0s and 1s).
- No translator is needed.
- Difficult to write and error-prone.

b) Assembly Language:

- Uses English-like mnemonics instead of binary code.
- Requires an assembler to convert code into machine language.

2. High-Level Languages

- Resemble English language and are easier to learn and use.
- Require a compiler or interpreter to translate code into machine language.
- **Examples:** C, C++, Java, Pascal, FORTRAN, BASIC, COBOL.

Advantages:

- Easy to read, write, modify, and debug

- Focus on problem-solving rather than machine architecture
- Provide machine independence, i.e., programs can run on different types of computers with minimal modification

✦ **Summary:**

- Programming languages are used to write computer programs.
- Low-level languages are closer to machine and difficult for humans.
- High-level languages are user-friendly, readable, and provide machine independence.

🌟 **Q.10: Describe characteristics of high-level programming languages**

❖ **Answer:**

High-level programming languages are designed to be user-friendly and closer to human language. Their key characteristics are:

1. English-like Syntax

- Instructions resemble English words, making them easy to read and understand.

2. Ease of Learning

- These languages are simple to learn compared to low-level languages.

3. Machine Independence

- Programs written in high-level languages can run on different types of computers with little or no modification.

4. Easy to Modify and Debug

- High-level languages allow programmers to easily correct errors and make changes in the program.

5. Focus on Problem Solving

- Programmers can concentrate on solving the problem rather than dealing with computer hardware details.

6. Structured Programming Support

- Provides a well-defined way of writing programs, allowing programs to be divided into smaller modules or functions.

7. Readable and Maintainable

- Programs are more readable, easier to maintain, and can be shared among programmers.

✦ **Summary:**

High-level languages are user-friendly, machine-independent, readable, and easy to debug, enabling programmers to focus on problem-solving rather than machine-level operations.

🌟 **Q.11: Briefly describe the basic structure of a C program**

❖ **Answer:**

A C program is written in a structured way and has a flexible structure, which makes programming easier and less error-prone. The basic structure of a C program includes the following parts:

1. Preprocessor Directives

- These are commands that give instructions to the C preprocessor.
- They begin with # (hash symbol).
- **Example:** #include <stdio.h> gives the program access to standard input/output functions like printf and scanf.

Main Function

- Every C program has a main function, where execution of the program starts.

Syntax:

```
void main(void)

{

   // body of the main function

}
```

- void before main indicates that the function does not return any value.
- The body of the main function contains the actual statements that implement the program logic.

3. C Statements

- Each statement in C ends with a semicolon (;).
- **Example:** printf("Hello World!"); displays output on the screen.

4. Delimiters

- Braces { } are used to enclose the body of the main function or any block of code.
- { indicates the beginning, and } indicates the end of a block of code.

5. Functions (Optional)

- A C program can be divided into smaller units called functions, where each function performs a specific task.
- This makes the program modular, easier to understand, and debug.

✦ **Summary:**

A basic C program consists of preprocessor directives, main function, C statements, and delimiters. The program starts execution from the main function and can be divided into smaller functions for modularity.

🌟 **Q.12: How would you create, edit, compile, link and execute a C program?**

❖ **Answer:**

To prepare and run a C program, the following stepwise process is used:

1. Creating a C Program

- Open the Turbo C++ IDE.
- Select File ⟶ New to open an edit window.
- Type the C program in the editor.

Example:

```
#include <stdio.h>

void main(void)

{

   printf("Hello World!");

}
```

2. Editing a C Program

- Use the editor to modify or update the program.
- Navigate using arrow keys or scroll bars.
- Make changes as required before saving.

3. Saving a C Program

- Select File ⟶ Save or press F2.
- Save with .c extension for standard C programs.
- Choose a proper directory for saving the source file.

4. Compiling a C Program

- Select Compile ⟶ Compile or press Alt + F9.
- The compiler converts the source program into an object program (.obj file).
- If there are syntax errors, the compiler will display error messages.

5. Linking a C Program

- Linking is the process of combining the object program with library files.
- Select Compile ⟶ Link in Turbo C++ to produce an executable (.exe) file.
- The linker ensures all functions used in the program are properly connected.

6. Executing a C Program

- Loading the program into memory is done by the loader.
- Press Ctrl + F9 or select Run ⟶ Run.
- To view output, select Window ⟶ User Screen or press Alt + F5.
- The output of the program is displayed on the screen.

✦ **Summary Flow:**

Create ⟶ Edit ⟶ Save ⟶ Compile ⟶ Link Execute ⟶ View Output

🌟 Q.13: Differentiate the following:

- (i)Preprocessor Directive and the Compiler
- (ii) Structured and Unstructured programming languages
- (iii) Linker and Loader

❖ Answer:

(i) Preprocessor Directive vs Compiler

1. Preprocessor Directive:

1. Preprocessor directives are special instructions given to the C preprocessor before compilation starts.
2. They always start with the symbol #, for example: #include, #define.
3. The #include directive is used to include header files, like stdio.h for input/output functions.
4. The #define directive is used to define constants or macros before compilation.
5. Preprocessor does not generate object code; it only prepares the source code for the compiler.

6. **Example:** #define PI 3.1416 tells the compiler to replace every occurrence of PI with 3.1416 before compilation.

Compiler:

1. The compiler translates the prepared source code into object code (machine language).
2. It checks the program for syntax errors and reports them.
3. Produces an object file (with .obj extension), which is later linked to create an executable file.
4. Without the compiler, the high-level program cannot be understood by the computer.

Key Difference: Preprocessor prepares the code, while the compiler translates the code into machine language.

(ii) Structured vs Unstructured Programming Languages

Structured Programming Languages:

1. Programs are divided into smaller modules or functions.
2. Each module has a specific task (like input, output, calculation).
3. Makes the program easy to read, debug, and maintain.
4. Reduces errors because logic is separated into small units.

Example: C, C++

Unstructured Programming Languages:

- The program is written as a single block or module.
- All instructions are executed sequentially.
- Difficult to understand, debug, or modify.
- Higher chances of errors due to lack of modularity.
- **Example**: Early BASIC programs

Key Difference: Structured programming is modular and organized, while unstructured programming is linear and non-modular.

(iii) Linker vs Loader

Linker:

- A linker combines object files and library files to create an executable (.exe) file.
- Resolves references to functions or variables used across multiple files.
- Works after compilation but before program execution.
- **Example:** When your program uses printf() from stdio.h, the linker links the object code for printf() with your program.

Loader:

- The loader loads the executable file into memory so it can be run by the CPU.
- Prepares the program for execution, allocating memory and setting up necessary resources.
- Works at runtime when the program is executed.
- **Example**: In Turbo C++, pressing Ctrl + F9 loads the program into memory and executes it.

Key Difference: Linker creates the executable file, while loader loads the executable into memory for running.

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.