C Programming

Chapters

Unit-1

Programming Language Fundamentals

- Introduction to Program and Programming Language
- Types of Programming Language (Low Level and High-Level Language)
- Language Translator (Assembler, Compiler and Interpreter)
- Program Error, Types of Error (Syntax, Semantic, Runtime Error)
- Program Design Tools (Algorithm, Flowchart)

Program:

- In computer science, a program is a set of instructions written in a programming language.
- It is also known as software or code.
- A program performs a specific task or solve a problem.

Programming:

- Programming is the process of writing computer programs.
- Broadly, it is the process of designing, writing, testing, debugging, and maintaining computer programs.
- The main goal of programming is to:
    - create efficient and reliable software
    - perform specific tasks
    - solve complex problems
    - and improve productivity.
- Programming can be used for various purposes, such as creating games, websites, or applications.
- Programming is done using programming languages, such as Python, Java, or C++ etc.

Programming Language:

- A programming language is a formal language designed to communicate with computer.
- With programming language, we can write programs and software applications.
- It is used to give instructions to computer to perform specific tasks or solve some problem.
- There are many different programming languages, such as C, C++, Python, Java, Ruby, JavaScript etc.

Programmer:

- A programmer is a person who writes computer programs.
- A programmer can also be called a developer, a coder, or a software engineer.
- A programmer uses a programming language to create and modify software for various purposes, such as games, websites, or applications.
- A programmer can work in different fields, such as web development, application development, game development, data science, or artificial intelligence.

Types of Programming Language

Programming languages are broadly categorized into two types:

1. Low level language
2. High level language

1. Low level language

- They are also called machine-level languages.
- Machines can easily understand it.

Low level languages are of two types:

a. Machine Level Language

- Machine language is lowest level of programming language.
- It handles binary data i.e., 0's and 1's.
- It directly interacts with system.
- Machine language is difficult for human beings to understand as it comprises combination of 0's and 1's.

Advantages:

- Machines can easily understand it.
- Machine-friendly.
- Execution is very fast

Disadvantages:

- Difficult to understand by human.
- Difficult to write code.
- Not portable.

b. Assembly Level Language

- Assembly language (ASM) is also a type of low-level programming language.
- It represents the set of instructions in a symbolic and human-understandable form.
- It uses an assembler to convert the assembly language to machine language.
- Assembly language is often used in systems programming, device driver development, and other low-level applications.

Advantages:

- Easier to read and understand.
- Faster and efficient than high level languages.

Disadvantages:

- Difficult to write programs than high level language.
- Not portable.

2. High level language

- They are designed to be more human-readable.
- They are easier to use than low-level languages like assembly language.
- Easy to understand and debugging is easy.
- They are generally portable between various platform and CPU architecture.
- They are very widely used and popular in today's times.
- High-level languages are often used in application development, web development, data science, machine learning etc.
- High-level programming language includes Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, Swift etc.

Advantages

- Easier to read, write, and understand than low-level languages
- Easier to learn.
- Faster to develop software.
- High-level languages are generally portable. i.e., same code that runs on different platforms.

Disadvantages

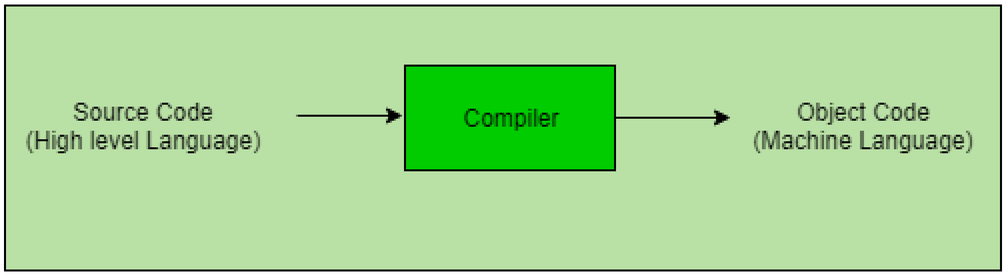
- High-level languages are less efficient.
- They are slower than low-level languages.
- Less suitable for low-level programming tasks.

Language Translator

- Computers can not understand anything except machine language i.e. 0's & 1's.
- So, the programs written in programming language should be translated to machine code.
- A programming language translator is a tool that is used to translate code written in one programming language into usually machine language.
- There are three main types of translators:

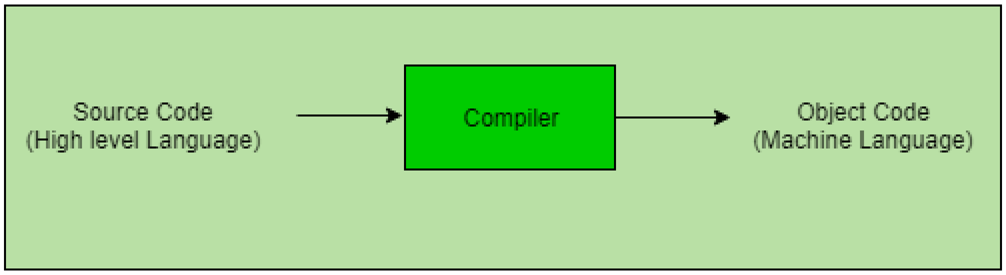
1. Compiler:

- A compiler is a program that translates source code into machine code.
- The compiler checks the entire program **at once** and translates it into an executable file.
- Errors are detected at compile-time, before the program is run



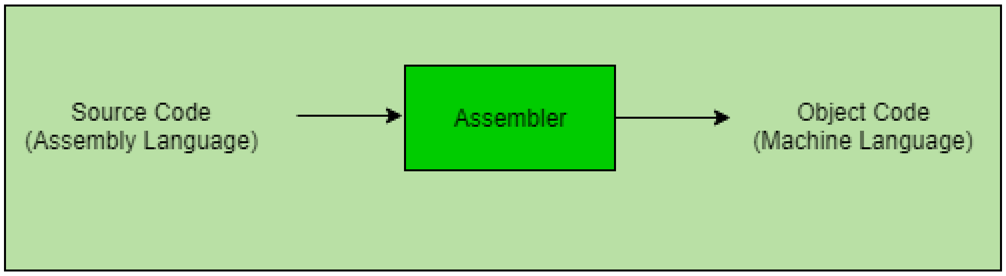
2. Interpreter:

- An interpreter is a program that translates source code to machine code.
- The interpreter translates/checks the code **line-by-line**.
- It does not generate an executable file; it translates & executes directly.
- Errors are detected at run-time, when the interpreter encounters a problem



3. Assembler

- An assembler is a program that translates assembly language code into machine code.
- Assembly language is a low-level programming language.



Compiler vs Interpreter

	Compiler	Interpreter
1	It checks and translates the whole program at once	It checks line by line.
2	It generates executable file.	It doesn't generate executable file.
3	Execution is faster.	Execution is slower.
4	Errors are detected at compile-time.	Errors are detected at run-time.
5	C, C++, C#, etc. are programming languages that are compiler-based.	Python, PHP, JavaScript etc. are programming languages that are interpreter-based.

Program Error

- In computing, an error is an unexpected/abnormal behavior of a system or program.
- It is also known as software errors.
- Errors can be caused by a variety of factors, such as:
    - incorrect user input,
    - hardware malfunctions,
    - software bugs,
    - and programming errors.

Types of program error

Syntax Errors:

- They occur when the code is not written correctly, not following the language grammar and syntax rules.
- Examples include missing semicolons, incorrect parentheses or brackets, misspelled function names, etc.
- The code won't compile and won't execute until all syntax errors are fixed.

Semantic Errors:

- They occur when the code is syntactically correct but computer can’t understand the logic.
- Examples include passing incorrect arguments to a function, using a variable before it's been initialized, etc.
- The code will compile and run, but will produce unexpected or incorrect results.

Runtime Errors:

- They occur when the code is running and attempting to execute an instruction that cannot be completed.
- Examples include division by zero, array index out of bounds etc.
- The code will crash and the error message will be displayed when there is a runtime error.

Program design tools

- Program design tools are software tools that help to design and plan programs before coding.
- They help to create a blueprint or a visual representation of the program's structure, logic, and flow.
- We can plan, create, test and improve their programs.
- Some common program design tools are: Flowchart, Algorithm, Pseudocode, Data flow diagrams, ER diagrams, UML diagrams etc.

Algorithm

- An algorithm is step-by-step process for solving a problem.
- An algorithm can be written in natural language, pseudocode, flowcharts.
- Algorithms provide a systematic approach to problem-solving that can lead to more accurate and efficient solutions.

Example

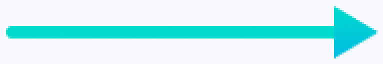
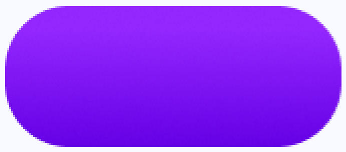
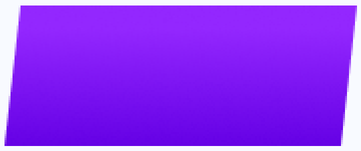
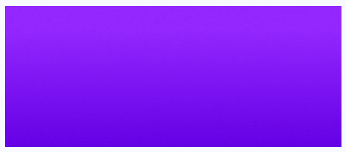
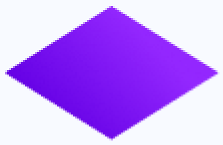
To find sum of two numbers:

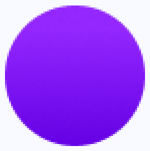
```
Step 1. Start
Step 2. Display “Enter any two numbers”
Step 3. Read two numbers as a and b
Step 4. Calculate c = a + b
Step 5. Display c
Step 6. Stop
```

Flowchart

- A flowchart is a visual representation of an algorithm.
- It uses arrows and symbols to show the sequence of steps.
- It represents different types of actions, such as start and end points, inputs and outputs, decisions, loops and processing steps.
- Arrows are used to connect the symbols.

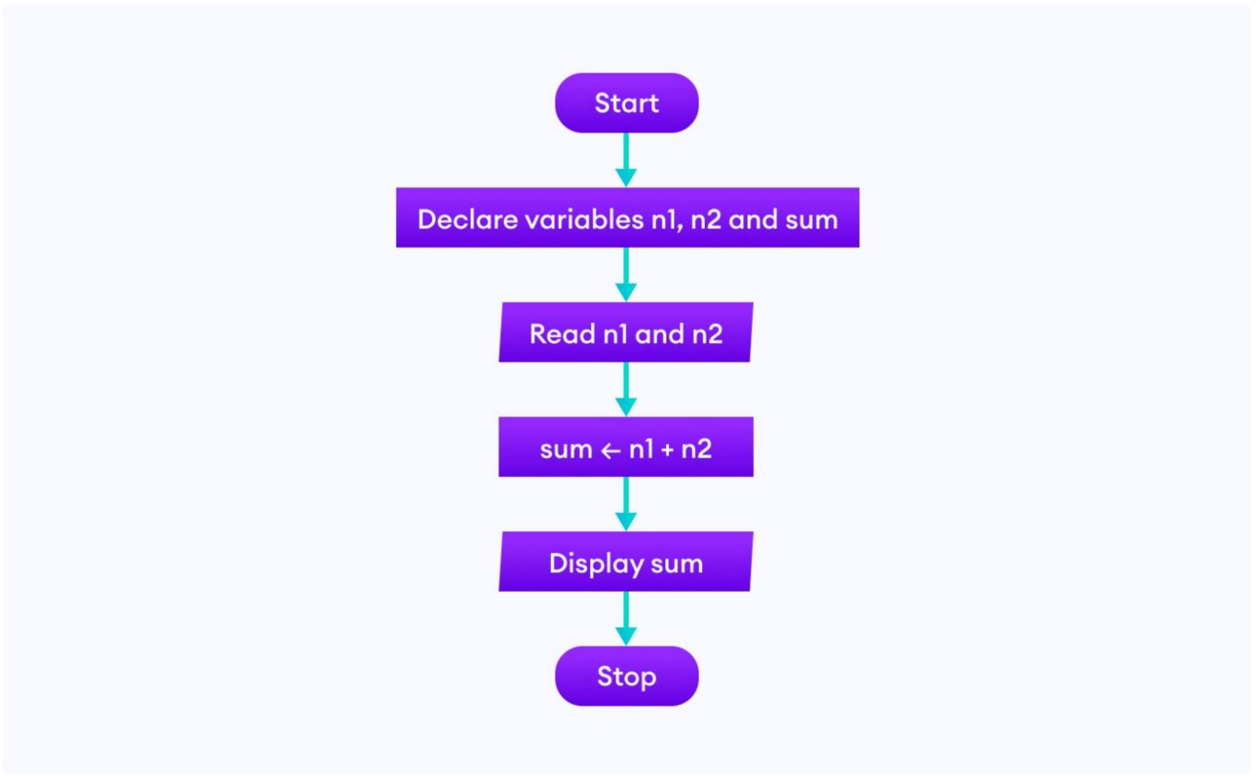
Symbols Used in Flowchart

Symbol	Purpose	Description
	Flow line	Indicates the flow of logic.
	Terminal (Stop/Start)	Represents the start and the end of a flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.
	Decision	Used for decision making between two or more alternatives.

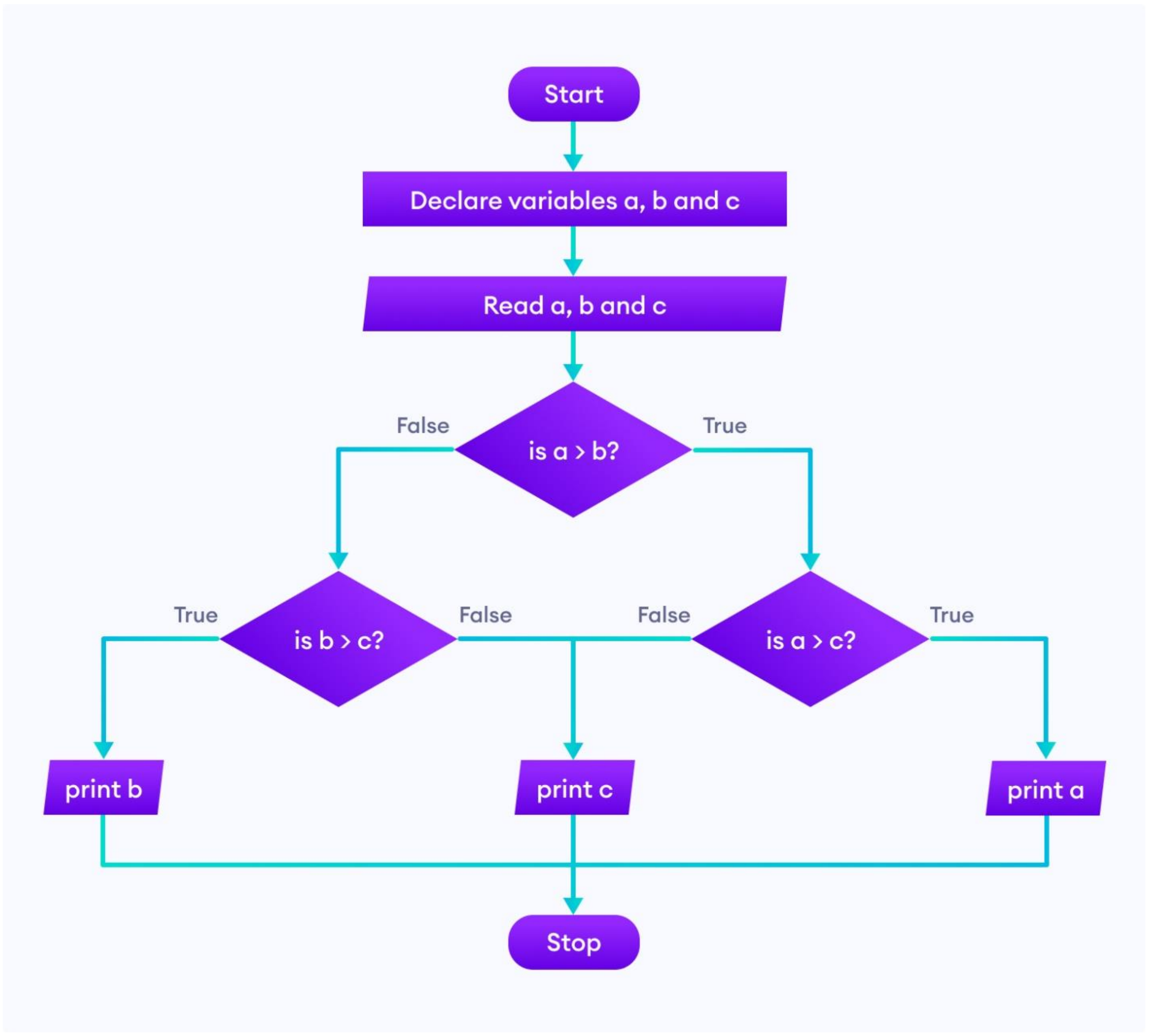
	On-page Connector	Used to join different flowline

Examples of flowcharts in programming

1. Add two numbers entered by the user.



2. Find the largest among three different numbers entered by the user.



https://engineerstutor.com/2018/08/27/examples-of-algorithms-and-flow-charts-with-c-code/

Assignments;

1. Differences between Low level language and High-Level Language.

2. Write an algorithm and draw flow chart to calculate Simple Interest.

3. Write an algorithm and draw flow chart to calculate area and perimeter of a rectangle.

4. Write an algorithm and draw flow chart to print first 10 natural numbers.

5. Write an algorithm and draw flow chart to check if the given number is even or odd.