Programming Fundamental

Lecture 2

Programming Languages

By

Asst. Lect. Ali Al-khawaja

Computer Program

- A program is a set of instructions following the rules of the chosen language.
- Without programs, computers are useless.
- A program is like a recipe.
- It contains a list of ingredients (called variables) and a list of directions (called statements) that tell the computer what to do with the variables.

Programming Language

- A vocabulary and set of grammatical rules (syntax) for instructing a computer to perform specific tasks.
- Programming languages can be used to create computer programs.
- The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal.

Programming Language

- As human languages are too difficult for a computer to understand in an unambiguous way, commands are usually written in one or other languages specially designed for the purpose.

Programming Language

- You eventually need to convert your program into machine language so that the computer can understand it.
- There are two ways to do this:
  - Compile the program
  - Interpret the program

Programming Language

- **Compile** is to transform a program written in a high-level programming language from source code into object code.
- This can be done by using a tool called **compiler**.
- A compiler reads the whole source code and translates it into a complete machine code program to perform the required tasks which is output as a new file.

Programming Language

- **Interpreter** is a program that executes instructions written in a high-level language.
- An interpreter reads the source code one instruction or line at a time, converts this line into machine code and executes it.

Programming Language

- Computer programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs.
- This source code is written in a programming language like C++, JAVA, Perl etc.

Programming Language

Common Features of All Program

All programs could be structured in the following four ways:

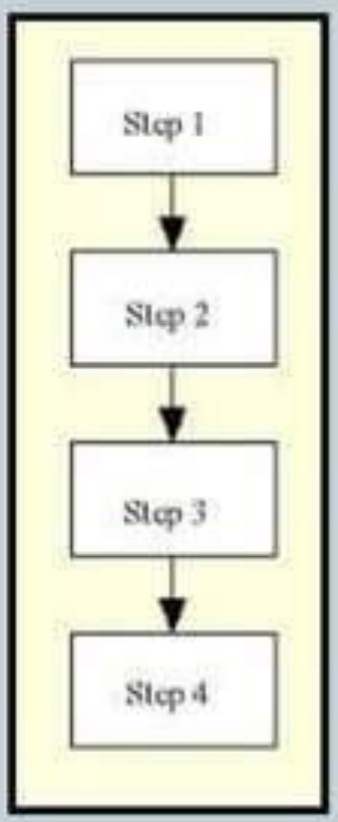
- Sequences of instructions
- Branches
- Loops
- Modules

Programming Language

Common Features of All Program

Sequences of Instructions

The program flows from one step to the next in strict sequence.

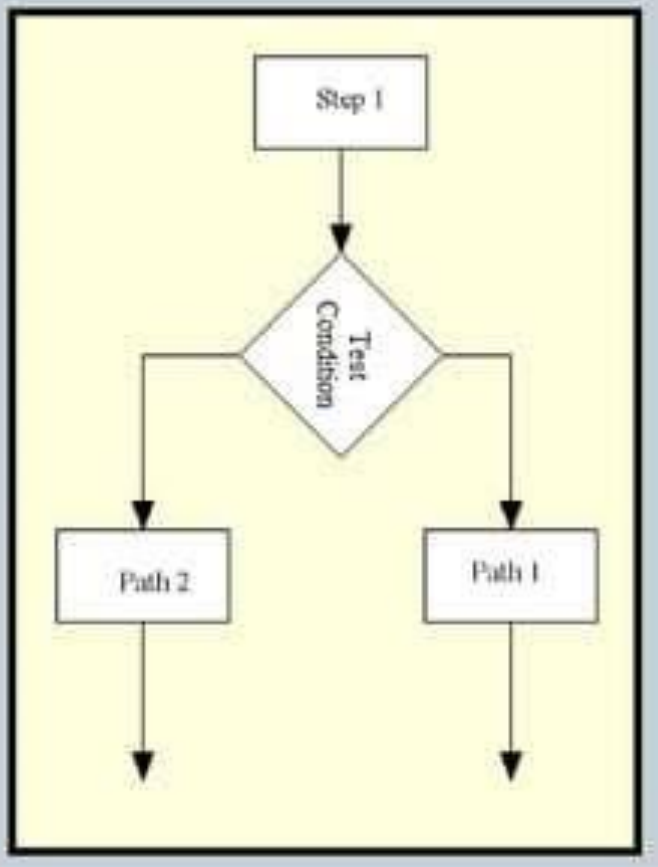


Programming Language

Common Features of All Program

Branches

The program reaches a decision point and if the result of the test is true then the program performs the instructions in Path 1, and if false it performs the actions in Path 2

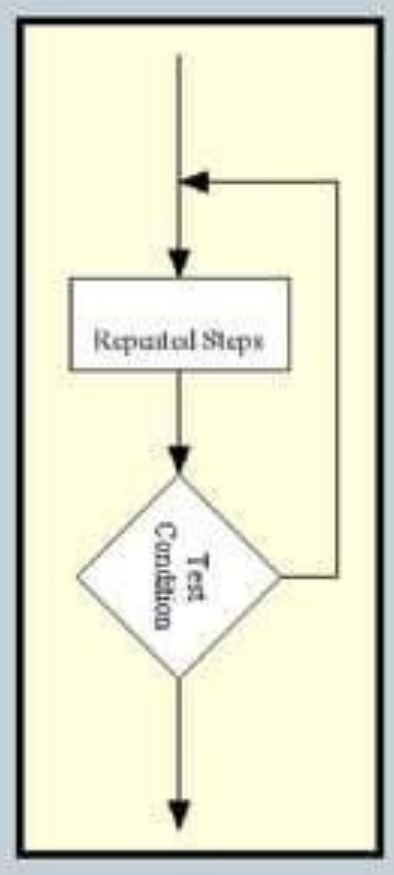


Programming Language

Common Features of All Program

Loops

The program steps are repeated continuously until some test condition is reached, at which point control then flows past the loop into the next piece of program logic



Programming Language

Common Features of All Program

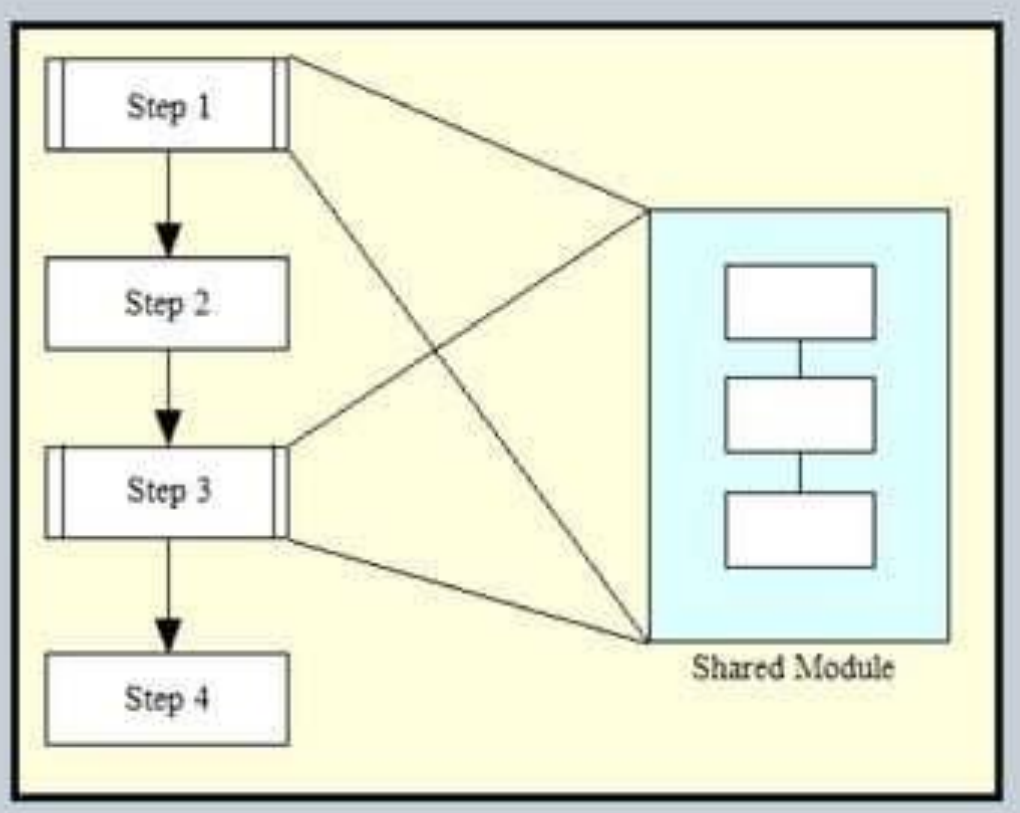
Modules

The program performs an identical sequence of actions several times. For convenience these common actions are placed in a module, which is a kind of mini-program which can be executed from within the main program.

Programming Language

Common Features of All Program

Modules



Programming Language

Common Features of All Program

Along with these structures programs also need a few more features to make them useful:

- Data (we take a closer look at data in the Raw Materials topic)
- Operations (add, subtract, compare etc)
- Input/Output capability (e.g. to display results)

Computer Programmer

- A **programmer** is someone who writes computer program.
- Computer programmers write, test, and maintain programs or software that tell the computer what to do.

What Skills are Required to Become a Programmer?

- **Programming** - Writing computer programs for various purposes.
- **Writing** - Communicating effectively with others in writing as indicated by the needs of the audience.
- **Reading Comprehension** - Understanding written sentences and paragraphs in work-related documents.
- **Critical Thinking** - Using logic and analysis to identify the strengths and weaknesses of different approaches.

What Skills are Required to Become a Programmer?

- **Computers and Electronics** - Knowledge of electric circuit boards, processors, chips, and computer hardware and software, including applications and programming.
- **Mathematics** - Knowledge of numbers, their operations, and interrelationships including arithmetic, algebra, geometry, calculus, statistics, and their applications.
- **Oral Expression** - The ability to communicate information and ideas in speaking so others will understand.

What Skills are Required to Become a Programmer?

- **Oral Comprehension** - The ability to listen to and understand information and ideas presented through spoken words and sentences.
- **Written Expression** - The ability to communicate information and ideas in writing so others will understand.
- **Written Comprehension** - The ability to read and understand information and ideas presented in writing.

What Skills are Required to Become a Programmer?

- **Deductive Reasoning** - The ability to apply general rules to specific problems to come up with logical answers. It involves deciding if an answer makes sense.
- **Information Organization** - Finding ways to structure or classify multiple pieces of information.

Generations of Programming Language

- The **first generation languages**, or 1GL, are low-level languages that are machine language.
- The **second generation languages**, or 2GL, are also low-level languages that generally consist of assembly languages.
- The **third generation languages**, or 3GL, are high-level languages such as C.

Generations of Programming Language

- The **fourth generation languages**, or 4GL, are languages that consist of statements similar to statements in a human language. Fourth generation languages are commonly used in database programming and scripts.
- The **fifth generation languages**, or 5GL, are programming languages that contain visual tools to help develop a program. A good example of a fifth generation language is Visual Basic.

Types of Programming Language

- There are three types of programming language:
  - **Machine language (Low-level language)**
  - **Assembly language (Low-level language)**
  - **High-level language**
- Low-level languages are closer to the language used by a computer, while high-level languages are closer to human languages.

Types of Programming Language

Machine Language

- Machine language is a collection of binary digits or bits that the computer reads and interprets.
- Machine languages are the only languages understood by computers.
- While easily understood by computers, machine languages are almost impossible for humans to use because they consist entirely of numbers.

Types of Programming Language

Machine Language

Machine Language

169 1 160 0 153 0 128 153 0 129 153 130 153 0 131 200 208 241 96

High level language

5 FOR I=1 TO 1000: PRINT "A";: NEXT I

Types of Programming Language

Machine Language

Example:

- Let us say that an electric toothbrush has a processor and main memory.
- The processor can rotate the bristles left and right, and can check the on/off switch.
- The machine instructions are one byte long, and correspond to the following machine operations:

Types of Programming Language

Machine Language

Machine Instruction	Machine Operation
0000 0000	Stop
0000 0001	Rotate bristles left
0000 0010	Rotate bristles right
0000 0100	Go back to start of program
0000 1000	Skip next instruction if switch is off

Types of Programming Language

Assembly Language

- A program written in assembly language consists of a series of instructions mnemonics that correspond to a stream of executable instructions, when translated by an assembler, that can be loaded into memory and executed.
- Assembly languages use keywords and symbols, much like English, to form a programming language but at the same time introduce a new problem.

Types of Programming Language

Assembly Language

- The problem is that the computer doesn't understand the assembly code, so we need a way to convert it to machine code, which the computer does understand.
- Assembly language programs are translated into machine language by a program called an **assembler**.

Types of Programming Language

Assembly Language

- Example:
  - Machine language :

    10110000 01100001
  - Assembly language :

    mov a1, #061h
  - Meaning:

Move the hexadecimal value 61 (97 decimal) into the processor register named "a1".

Types of Programming Language

High Level Language

- **High-level** languages allow us to write computer code using instructions resembling everyday spoken language (for example: **print**, **if**, **while**) which are then **translated** into machine language to be executed.
- Programs written in a **high-level** language need to be translated into **machine language** before they can be executed.
- Some programming languages use a **compiler** to perform this translation and others use an **interpreter**.

Types of Programming Language

High-Level Language

- Examples of High-level Language:
  - ADA
  - C
  - C++
  - JAVA
  - BASIC
  - COBOL
  - PASCAL
  - PHYTON

Choosing a Programming Language

Before you decide on what language to use, you should consider the following:

- your server platform
- the server software you run
- your budget
- previous experience in programming
- the database you have chosen for your backend

The Programming Process

The Programming Process

Step 1: Defining the problem.
Step 2: Planning the solution.
Step 3: Code the program.
Step 4: Test the program.
Step 5: Document everything.

The Programming Process

Step 1: Defining the problem

- The task of defining the problem consists of identifying what it is you know (input-given data), and what it is you want to obtain (output-the result).
- Eventually, you produce a written agreement that, among other things, specifies the kind of input, processing, and output required.

The Programming Process

Step 1: Defining the problem

Example:

- What must the program do?
- What outputs are required and in what form?
- What inputs are available and in what form?

The Programming Process

Step 2: Planning the solution

- Two common ways of planning the solution to a problem are to draw a **flowchart** and to write **pseudocode**, or possibly both.
- A **flowchart** is a pictorial representation of a step-by-step solution to a problem.
- It consists of arrows representing the direction the program takes and boxes and other symbols representing actions.
- It is a map of what your program is going to do and how it is going to do it.

The Programming Process

Step 2: Planning the solution

- **Pseudocode** is an English-like nonstandard language that lets you state your solution with more precision than you can in plain English but with less precision than is required when using a formal programming language.
- Pseudocode permits you to focus on the program logic without having to be concerned just yet about the precise syntax of a particular programming language.

The Programming Process

Step 3: Code the program

- You will translate the logic from the flowchart or pseudocode-or some other tool-to a programming language.
- Program Coding means expressing the algorithm developed for solving a problem, in a programming language.

The Programming Process

Step 4: Test the program

- Almost all programs may contain a few errors, or bugs.
- Testing is necessary to find out if the program produces a correct result.
- Usually it is performed with sample data
- Debugging is the process of locating and removing errors

The Programming Process

Step 4: Test the program

Types of Error

Syntax Errors: Violation of syntactic rules in a Programming Language generates syntax errors.

Effect? Interpreter or Compiler finds it in Syntax Check Phase.

The Programming Process

Step 4: Test the program

Types of Error

Semantic Errors: Doing logical mistakes causes semantic errors in Source code.

Effect? Interpreters and Compilers can not notice them, but on execution, they causes unexpected results.

The Programming Process

Step 4: Test the program

Types of Error

Run-time Errors: Occur on program execution. Mostly caused by invalid data entry or tries to use not existing resources.

Effect? It occurs on run time and may crash the program execution

The Programming Process

Step 5: Document everything

- Documentation is a written detailed description of the programming cycle and specific facts about the program.
- Typical program documentation materials include the origin and nature of the problem, a brief narrative description of the program, logic tools such as flowcharts and pseudocode, data-record descriptions, program listings, and testing results.
- Comments in the program itself are also considered an essential part of documentation.