

Programming Concepts

Objectives

- Define computer programming languages
- Define a computer program
- Understand the basic terminologies
- Explore different types of programming languages
- List different programming language generations
- Identify different programming tools
- Explore different types of programming structures
- Identify the main steps to solve a problem using programming languages

Programming Languages

- Programming languages are made up of keywords and grammar rules designed for creating computer instructions
- Computer Program: a set of instructions that tell the computer how to perform a task.
- Programmer: a person who writes the program instructions (source code)

Terminologies

- Keyword/Command
 - It is a word with a predefined meaning that is reserved by a program that defines commands and specific parameters for that code set. The number of keywords may differ from one language to another.
- executable vs. non-executable statements
 - **Executable statement:** It usually starts with a key word and initiates actions. In other words, it is a description of what and how the program should take an action
 - **Non-Executable statement:** It provides info about the nature of the data or about the way the processing is to be done without causing any processing action.
- Syntax vs. Semantic
 - **Syntax** is the grammar rules that are used whenever a program in a computer language is written. (like grammar in the natural language)
 - **Semantics** is the function of the command. (like meaning in the natural language)

Terminologies(cont'd)

- Variable vs. Constant
 - **Variable:** it is a memory location and its values are normally changed during the course of program execution.
 - Naming a variable is part of the language syntax. Programmers should follow the language guidelines to name variables. It could be different from one language to others.
 - Variable Declaration : it is to specify a variable's name and characteristics.
 - Variable Datatype: It is a set of possible values and a set of allowed operations on it. A data type tells the compiler or interpreter how the programmer intends to use the data.
 - **Constant:** it is a value that should not be altered by the program during normal execution.
 - Note: if it used as type qualifier then it is used to declare a constant variable like in C language.
- Variable/constant name should be descriptive

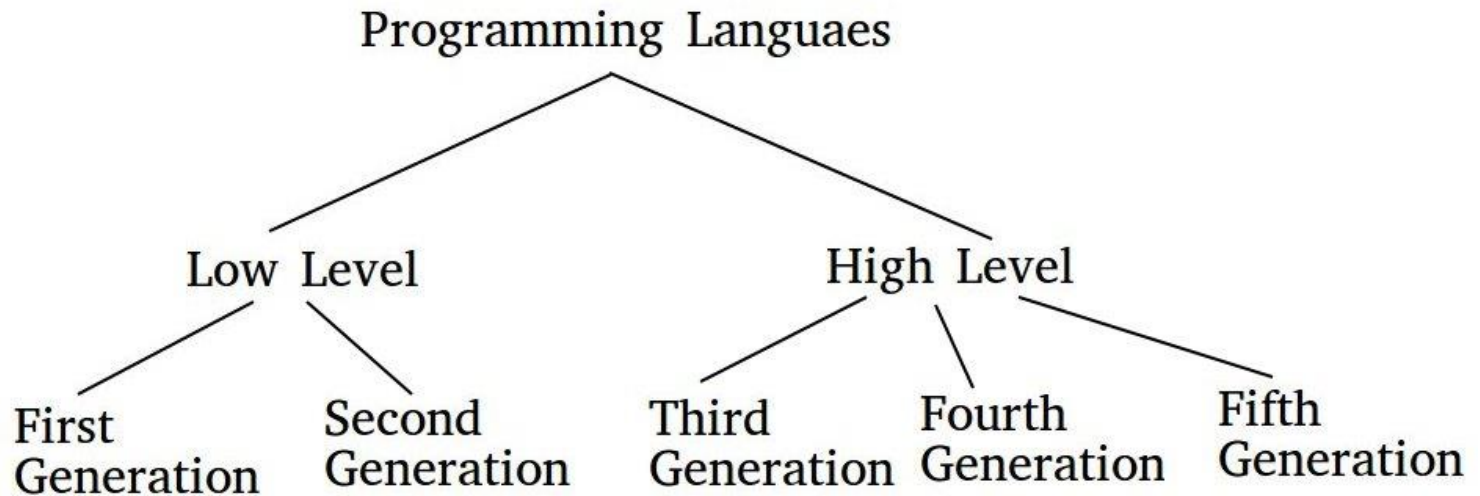
Terminologies (cont'd)

- File name vs. filename extension
 - File name: is a framework for naming a file in a way that describes what they contain and how they relate to other files.
 - Filename extension: It indicates a characteristic of the file contents or its intended use. A filename extension is typically delimited from the filename with period(.).
 - Example: obj, exe, dat, etc.
- File types
 - Source
 - Object
 - Executable
 - Data

Terminologies (cont'd)

- Interpreter:
 - A program that translates source code into some efficient intermediate representation or object code and immediately executes that.
 - It executes one statement at a time.
- Compiler:
 - A system program that translates high-level language (source code) to machine language (object code)
 - It translates the entire source code (statements) at one time.
 - Assembler: it translates the assembly program (source code) to the object code.

Programming Languages Types & Generations



Programming Languages Types

- Low – Level Language
- High – Level Language

Programming Language Types

- Low-level languages typically include commands specific to a particular CPU or microprocessor family. It is easy for the machine to understand
 - Machine Language. Programs are coded in binary (0s and 1s)
 - Assembly Language. It is a symbolic coded sequence of instructions

Programming Language Types

- High-level Language:
 - High-level languages use command words and grammar based on human languages
 - Languages that are easy to understand and to write by humans using words from the English language.
 - Java
 - C#
 - C and C++

Computer Programming Language Generations

- First Generation:
 - Machine Language:
 - It consists of binary instructions (0's and 1's) that a computer can understand and respond to directly.
 - Examples: 0101110110111001
- Second Generation
 - Assembly Language:
 - It is a low level programming language using the human readable instructions.
 - It is used in kernels and hardware drives.
 - Example: Sub, Add, Mov.

Computer Programming Language Generations (cont'd)

- Third Generation:
 - Easy-to-remember command words
 - Procedure Languages: It is a programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program.
 - These are high-level languages like C, Fortran, and Basic.

Computer Programming Language Generations (cont'd)

- Fourth Generation:
 - More closely resembles human language
 - Object Oriented Language: It is a programming paradigm that represents concepts as "objects" that have data fields and associated methods.
 - Languages that consist of statements that are similar to statements in the human language. These are used mainly in database programming and scripting. Examples of these languages include Java and Python.

Computer Programming Language Generations (cont'd)

- Fifth-generation languages
 - Based on a declarative programming paradigm.
 - These are the programming languages that have visual tools to develop a program. Examples of these languages include Prolog.

programming paradigm

- The programming paradigm refers to a way of conceptualizing and structuring the tasks a computer performs.

Programming Tools

- An SDK (software development kit) is a collection of language-specific programming tools that enables a programmer to develop applications for a specific computer platform
- An IDE (integrated development environment) is a type of SDK that packages a set of development tools into a sleek programming application
- A component is a prewritten module, typically designed to accomplish a specific task
- An API is a set of application programs or operating system functions that programmers can access from within the programs they create
- A VDE (visual development environment) provides programmers with tools to build substantial sections of a program

Solving problems using programming language

- Identify and Understand the problem
- State/outline the solution
- Program planning
 - Algorithm
 - Flowchart
 - Pseudocode
- Write the program source code
- Program testing and documentations

Program Planning




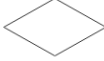



- The problem statement defines certain elements that must be manipulated to achieve a result or a goal
- You accept assumptions as true to proceed with program planning
- Known information helps the computer to solve a problem
- Determine variables & constants

Algorithms

- Set of steps for carrying out a task that can be written down and implemented
- Start by recording the steps you take to solve the problem manually
- Specify how to manipulate information
- Specify what the algorithm should display as a solution
- Expressing an Algorithm
 - Flowchart
 - Pseudocode

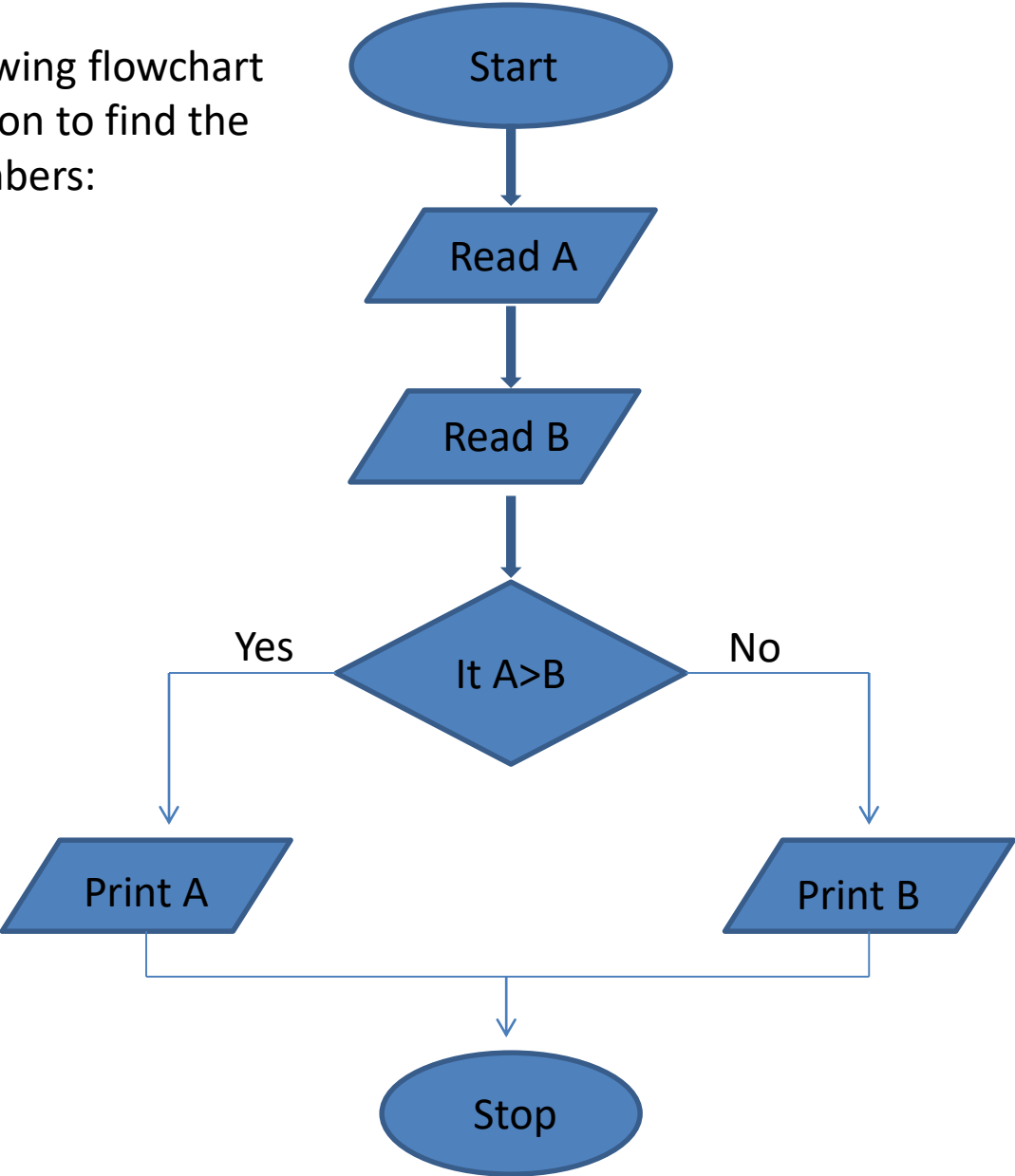
Flowchart

- A flowchart is a graphical representation of solution steps (algorithms and programming logic) for a given problem.
- Flowchart symbols are the elements you can use to describe the steps involved in a workflow process.

SYMBOL	PURPOSE	DESCRIPTION
	Terminal (Stop / Start)	The terminal symbols present in every programming flowchart as the process starts with a "start" command, and a "stop" command shows the end of the whole process on the flowchart. The example also has both the start and end symbols represented by a rectangular sign with curved edges to signify the beginning and end of a programming flowchart procedure.
	Input / Output	Input/ Output: The commands of Input and Output in operation are most crucial. To get a logical flow to go through the processing, the user needs to give input. The system reads the inputs to give an output. The symbols for inputs and outputs are parallelograms.
	Processing	For a process to complete successfully, the method must include the function of processing. The processing part occurs between the input and the output. The rectangle shapes represent the processing work.
	Decision	When there is a need to decide between true or false, this function gets used. The diamond-shaped symbols are useful when the function is taking a series of decisions to get the result.
	On-page Connector	When there is a need to connect different flowlines, on-page connectors are present at the junction.
	Off-page Connector	The off-page connectors connect different flowlines when they are present on separate pages.
	Predefined Process/Function	When a group of some statements performs a predefined work, its representation occurs with this symbol.

[Flowchart Symbols - Bing images](#)

Example: the following flowchart represents a solution to find the largest of two numbers: (A &B).



Pseudocode

- A Pseudocode is defined as a step-by-step description of an algorithm. It uses the simple English language text as it is intended for human understanding rather than machine reading.
- Example: Find the largest of two numbers:
(A & B)?

Pseudocode

Step 1: Start

Step 2: Read number a

Step 3: Read number b

Step 4: if $a > b$, print a (Compare a and b using greater than operator)

Step 5: else print b

Step 6: Stop

(Output: Largest number between a and b)

Writing the program source code

- Use the appropriate language to write the source code to find the largest of two numbers.
- Example C, C++, C #, Java, etc.

```
// C++ Program to Find Largest of two numbers
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    int a, b, largest;
    Cout << "Enter two number : ";
    cin>> a >> b;
    if(a>b)
    {
        largest=a;
    }
    else
    {
        largest=b;
    }
    cout<< "Largest of the two number is " <<largest;
}
```

Program Testing and Documentation

- A computer program must be tested to ensure that it works correctly.
- Program errors include:
 - Syntax errors
 - Runtime errors
 - Logic errors
- A debugger can help a programmer read through lines of code and solve problems.
- Comments are a form of documentation that programmers insert into the program code.
 - Example:
 - `//` symbol for writing a single line comment and `/*` multiline comment in C++ and Java.
 - `#` symbol for writing a single line or multiline comments in Python.

Program structures

- Sequence Structure
- Selection Structure
- Repetition Structure

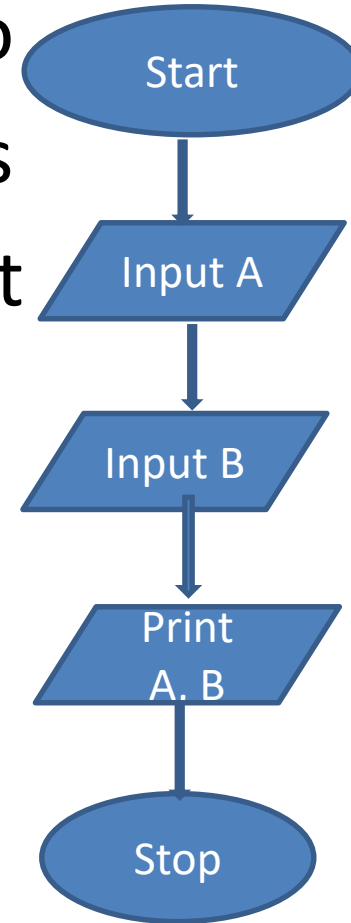
Sequence

Structure:

Sequence means that the given instructions are executed in the same order that they were given.

Example:

Read two numbers and print them.

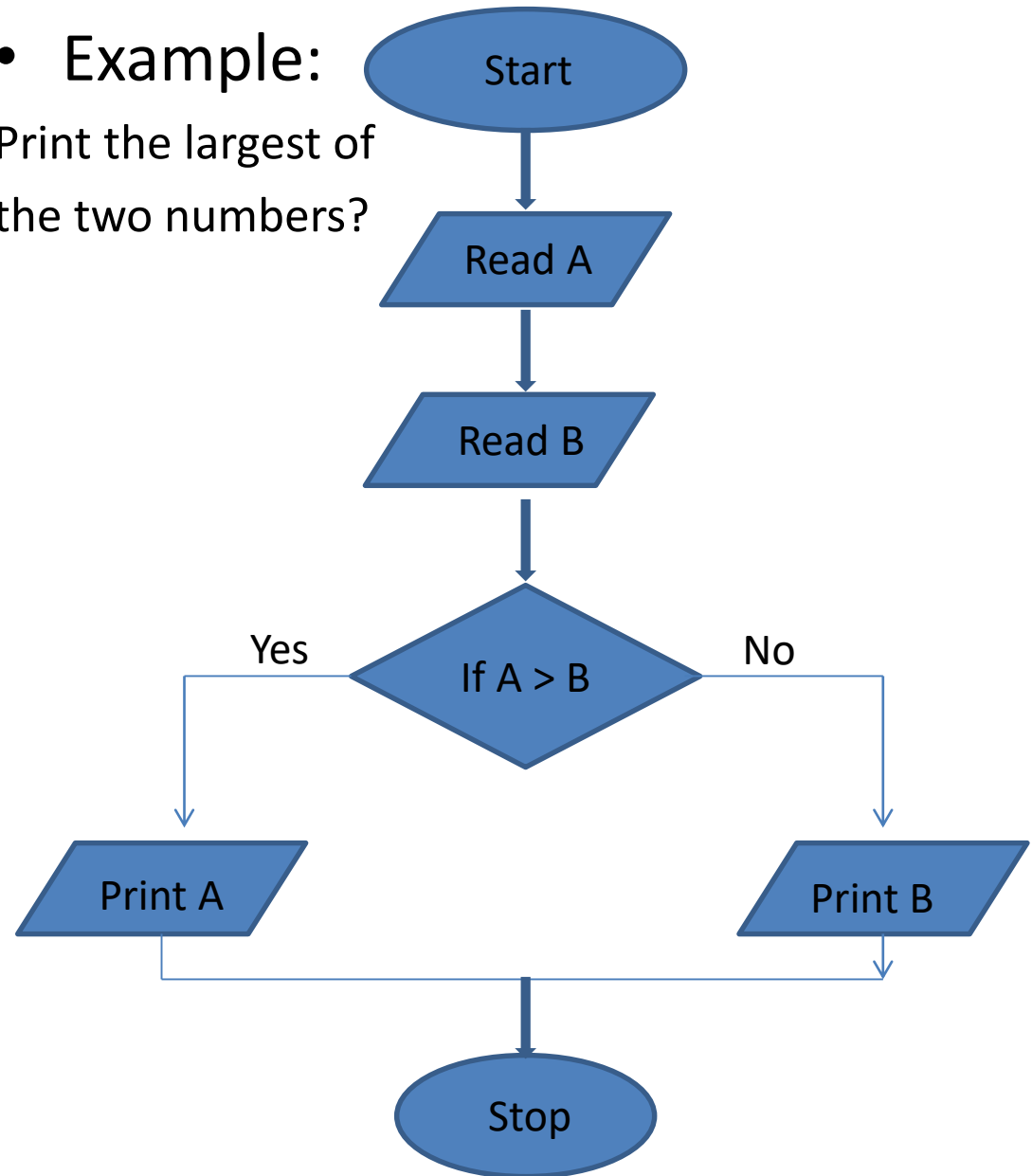


Stop

Selection Structure:

A selection structure is to be able to select between two or more alternate paths.

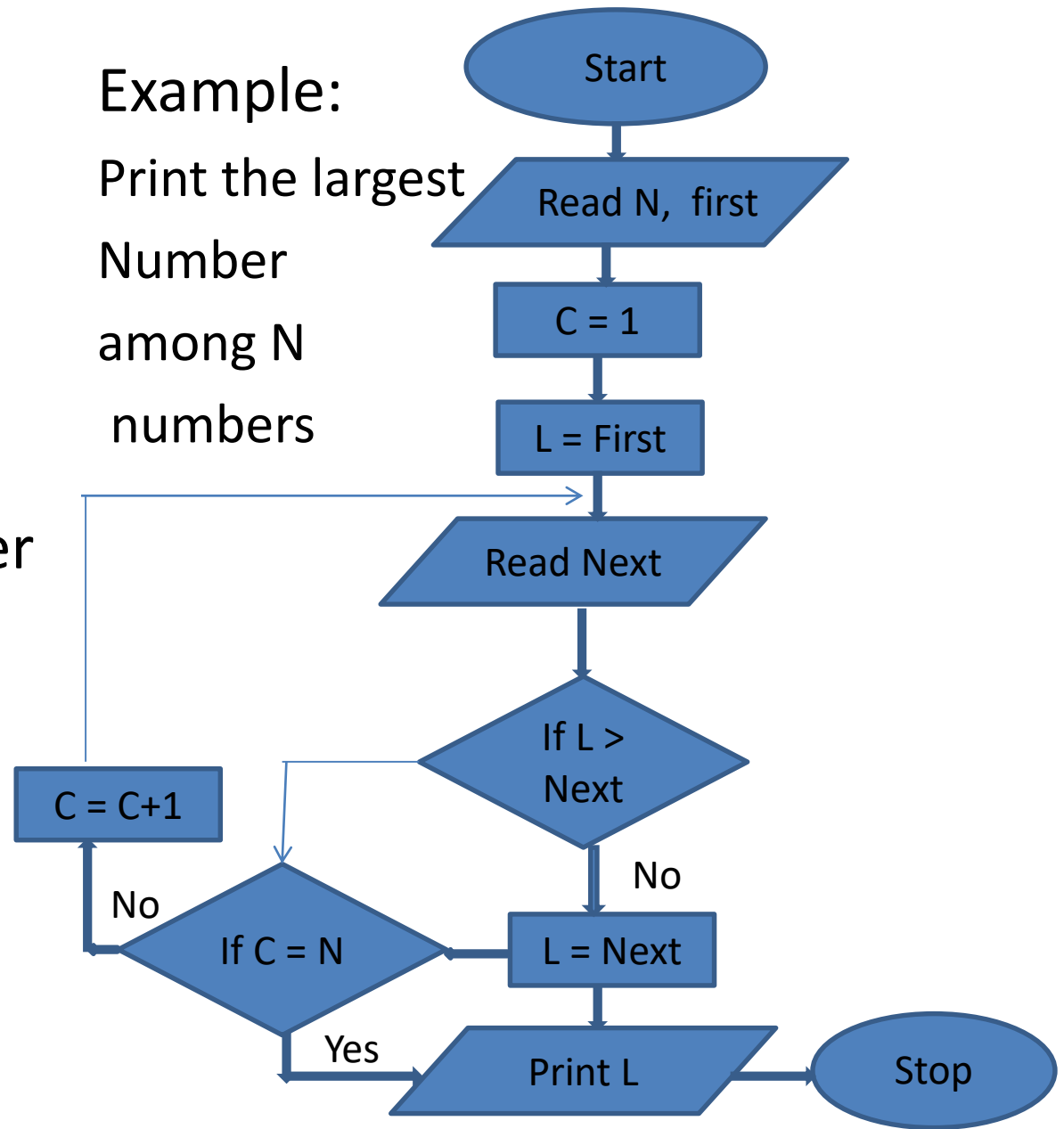
- Example:
Print the largest of the two numbers?



Repetition Structure

It allows a set of instructions (code section) to be executed a number of times based on pre-condition or post-condition.

Example:
Print the largest
Number
among N
numbers



References & Additional Resources

- New Perspectives on Computer Concepts, Comprehensive, by Dan Oja.
- [Introduction To The Concepts Of Computer Programming](#)
- [Computer Programming Tutorial](#)
- [Introduction to Programming and Computer Science - Full Course video](#)