



Programming in C

Module Number: 01

Module Name: Overview of programming

Overview of programming

AIM:

The aim of this module is to familiarize the students with computer based problem solving techniques and programming environment.

Students participating in this course will learn about:

- Understand the basic concepts of computer system, program design and implement issues.
- Devise algorithms (model) and build programs (implement) that solve concrete problems, using:
 - a) basic data structures (variables, arrays, and possibly others)
 - b) numerical and Boolean expressions
 - c) control structures (such as decisions and loops)
 - d) functionality factored out in functions
- Perform tests on algorithms (such as number guessing game) and corresponding programs.

Overview of programming

Objectives:

The Objectives of this module are:

- Explain computer-based problem solving.
- Demonstrate the usage of algorithms in computer problem solving.
- Identify the importance of programming techniques.
- Apply the perception of an abstract data structure.
- Illustrate the working of top design approach in problem solving.
- Describe the various types of Programming Languages.
- Demonstrate the working of an Assembler.
- Explain the working of a Compiler.
- Elaborate the functions of Assembler, Compilers and Interpreters.

Overview of programming

Outcome:

At the end of this module, you are expected to:

- Identify a problem definition.
- Solve abstract and complex problems using modular design methodology.
- Design an algorithmic solution to a problem using problem decomposition and step-wise refinement.
- Differentiate between low-level, middle-level and high-level Programming Languages.
- Outline the various functions of Assembler.
- Identify the important functions of Assembler, Compiler and Interpreter.
- Compare Compiler and Interpreter.

Overview of programming

Contents

1. Introduction
2. Computer-Based Problem Solving
3. Algorithm and Flowchart for Problem Solving
4. Programming and Implementation Issues
 - System Design Technique
 - Programming Technique
5. Programming Languages
6. Assemblers/ Compilers/ Interpreter

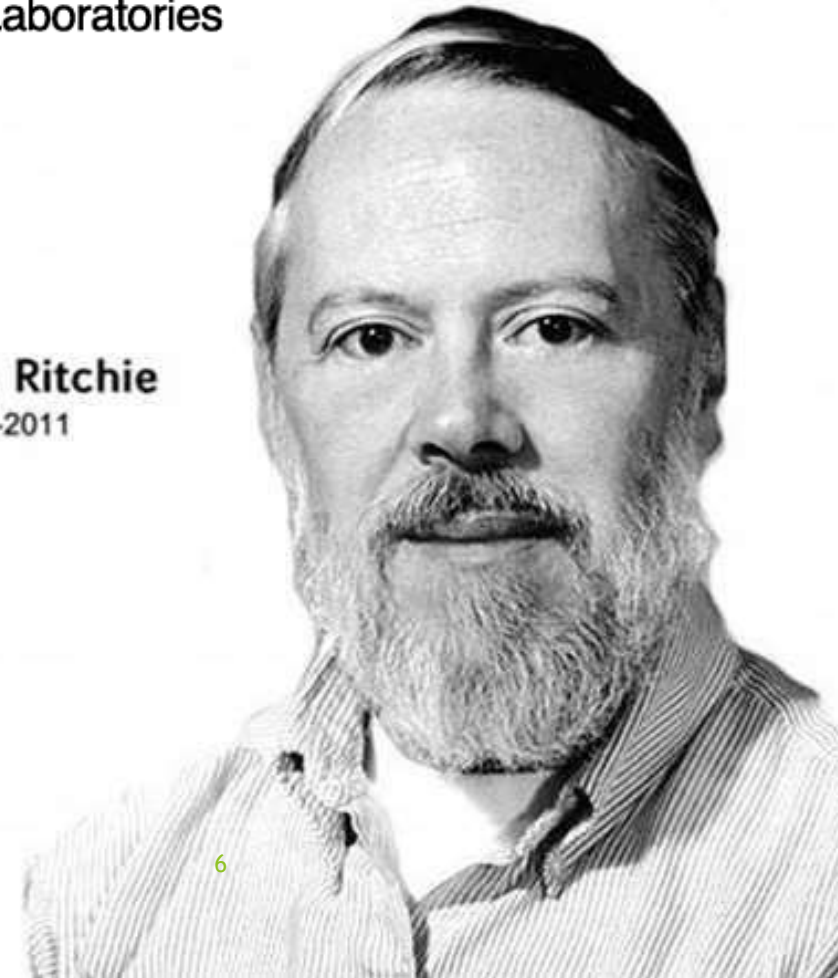
Overview of programming

◀ Introduction to C developer

In the early 1970s, **Dennis Ritchie** developed C programming language at **AT&T Bell Laboratories**. Even before being published, the language started gaining popularity and support in its revolutionary phase. Because of its enormous success, very soon needs were found to standardize the definition of C language. In 1983, a committee was established by the American Nation Standards Institute, known as ANSI C Committee to standardize the C programming language.



Dennis Ritchie
1941-2011



Introduction

Overview of programming

What is Programming C?

C is a general purpose, structured programming language. There is a big debate in considering whether C is a high-level programming language or a low-level programming language. In reality, it provides capabilities that help the user to interact with machine level which is a much lower level of the computer. It has been nearly connected with the UNIX system where it was developed subsequent to both the system and the greater part of the projects that keep running on it are developed in C.



Overview of programming

Computer-Based Problem Solving

Overview of programming

What is Computer-Based Problem Solving Process?

Computer-Based Problem Solving Process is a work intended to offer a systematic treatment to the theory and practice of designing, implementing, and using software tools during the problem solving process. This method is obtained by enabling computer systems to be more intuitive with human logic rather than machine logic. Instead of software dedicated to computer experts, the author advocates an approach dedicated to computer users in general. This approach does not require users to have an advanced computer education, though it does advocate a deeper education of the computer user in his or her problem domain logic.

Overview of programming

The following is the list with the programming tools:

- ◀ Flow charts
- ◀ Pseudo code
- ◀ Direction of numbered NYC streets Algorithm
- ◀ Class Average Algorithm

Overview of programming

What is Program identification?

◀ Definition:

- ◀ This is the first and important step to determine the information needs. Problem can be identified by taking the existing system as reference. A question and answer session also can help in problem identification. Expected information needs can be stated or users also can interpret in suggesting information needs to define a problem.
- ◀ Simply a problem is nothing but a crisis. In case of computer-based system, it may be related to either software or hardware which may affect the user requirements of system administrators, managers or simple users. In such cases, problem identification indicates towards the understanding of information needs of various users. So any gap in understanding the information needs indicates failure in the future. While identifying problem definition, you should also keep in mind about user's level of understanding. On the basis of that, users can be of two groups: one, who are totally unaware of advantages of immediate access to information through computers and two who are well aware of the advantages of computers.

Overview of programming

There are some other concerns which may create problem while identifying information needs:

- ◀ **Speed of Processing:** It indicates the time required for processing any transaction. As you know, for some system it is very crucial to get the response in fraction of a second. For example, railway reservation system, online payment and so on.
- ◀ **Volume of Work:** To avoid any kind of hazards, the volume of information to be handled at each transaction should be calculated.
- ◀ **Reliability:** Design should be robust enough to provide reliability. In case of any breakdown, alternative computing facility should be available.

Overview of programming

There are some other concerns which may create problem while identifying information needs

- ✦ **Accuracy:** To provide accuracy, the machine should process the data precisely.
- ✦ **Security:** It is the major concern of any computer-based problem. Confidentiality, privacy and security of data should be provided to make a successful solution for any computer-based problem.
- ✦ **Cost:** It indicates the operational and developmental cost of data processing system. It includes manpower, hardware, software, consumables and supplies and overhead costs.

Algorithm and Flowchart for Problem Solving

Overview of programming

Programming Tools:

A Programmer uses various programming languages to create programs. But before writing a program in a programming language, a programmer first needs to find a procedure for solving the problem which is known as planning the program. The program written without proper pre-planning have higher chances of errors. The tools that are used to plan or design the problem are known as programming tools. Algorithm and flowchart are widely used programming tools.

← **Flow charts**

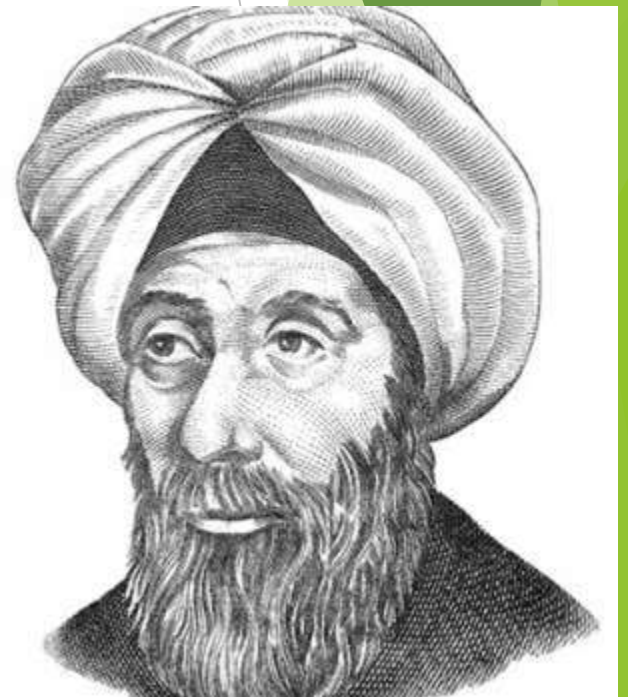
← **Algorithm**

Overview of programming

What is Algorithm?

The word 'algorithm' relates to the name of the mathematician Muhammad ibn Musa Al-Khwarizmi, which means a procedure or a technique. Programmer commonly uses an algorithm for planning and solving the problems.

An algorithm is a specific set of meaningful instructions written in a specific order for carrying out or solving a specific problem.

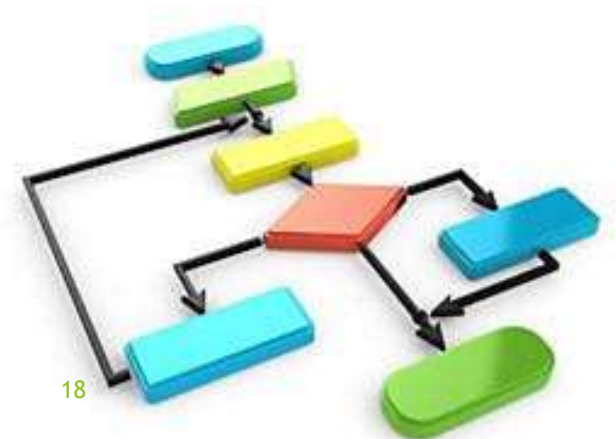


Overview of programming

What is Algorithm?

Algorithms are self-contained set of operations to be performed step by step.

In reality, algorithms have been generated from the concept of mathematics. In mathematics, we follow certain rules and maintain a sequence to solve any problems. Similarly, an algorithm also consists some rules and follows a sequence to solve the problem. An algorithm can be defined for any kind of a problem (For example, Mathematical, scientific or business oriented problems).



Overview of programming

Types of Algorithm

The algorithm and flowchart are classified into three types of control structures:

1. Sequence
2. Branching (Selection)
3. Loop (Repetition)

According to the condition and requirement, these three control structures can be used. In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down. Whereas in branch control, there is a condition and according to that condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE condition, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.

Overview of programming

An algorithm has a few properties as discussed below:

- ✦ **Finiteness:** In an algorithm, a number of steps should be finite, i.e., after any mechanical execution it should stop automatically.
- ✦ **Definiteness:** An algorithm should be simple and unambiguous so that the computer can interpret it easily.
- ✦ **Generality:** Algorithm should be designed in such a way that it can solve all the problems of a similar kind.
- ✦ **Effectiveness:** An algorithm should be effective so that it can give a unique solution to the problem it is designed for.
- ✦ **Input-output:** Each algorithm works with some input or initial data to generate some outputs. So, algorithms should be careful about run-time problems (especially mathematical problems) which may arise unexpectedly.

Overview of programming

Following is an example of the concept of an algorithm:

Example: How to make a cup of tea?

- ✦ **Step 1:** Switch on the electric heater.
- ✦ **Step 2:** Place a kettle on the heater.
- ✦ **Step 3:** Let the water boil in the kettle.
- ✦ **Step 4:** Add milk, sugar and tea bags.
- ✦ **Step 5:** Remove the tea bag and serve the tea.

From the above-mentioned example, we can see how a step by step approach can give a solution for a particular problem.

Overview of programming

Examples:

1. Write an algorithm to find the smallest number between two numbers.

Step 1: Start

Step 2: Input two numbers, say A and B

Step 3: If $A < B$ then small = A

Step 4: If $B < A$ then Small = B

Step 5: Print Small

Step 6: End

Overview of programming

Examples:

2. Write an algorithm to check odd or even number

Step 1: Start

Step 2: Read/Input a number and store in A

Step 3: Is $A < 0$?

If YES then $C = \text{"ODD"}$

If NO then $C = \text{"Even"}$

Step 4: Display C

Step 5: Stop

The Loop or Repetition allows a statement or block of statements to be executed repeatedly based on certain loop condition. 'While' and 'for' construct are used to represent the loop structure in most programming languages. Loops are of two types: Bounded and Unbounded loop. In bounded loop, the number of iterations is fixed while in unbounded loops the condition has to satisfy to end the loop.

Overview of programming

Characteristics of a good algorithm:

← The Finite number of steps:

After starting an algorithm for any problem, it has to terminate at some point.

← Easy Modification

There can be numbers of steps in an algorithm depending on the type of problem. It supports modification of Steps.

← Easy and simple to understand

A Simple English language is used while writing an algorithm. It is not dependent on any particular programming language. People without the knowledge of programming can read and understand the steps in the algorithm.

← Output

An algorithm is just a design of a program. Every program needs to display certain output after processing the input data. So one always expects the result as an output from an algorithm. It can give output at different stages. The result obtained at the end of an algorithm is known as the end result and if the result is obtained at an intermediate stage of process or operation then the result is known as an intermediate result. Also, the output has to be as expected having some relation to the inputs.

easy

Overview of programming

Classification of Algorithms:

According to the various factors, algorithms can be classified as shown below:

**Deterministic,
Non-Deterministic and
Random Algorithm**

**Direct and Indirect
Algorithm**

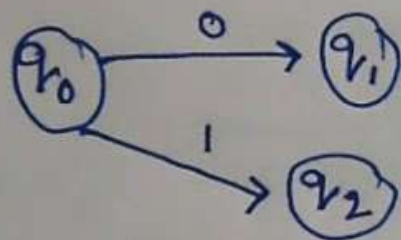
Infinite Algorithm

Let us discuss the three basic components of algorithm development to control the flow of information processing. They are sequential flow, conditional flow and repetitive flow.

Deterministic Algorithm

1> given a particular input the computer will give always same ~~int~~ output

2>



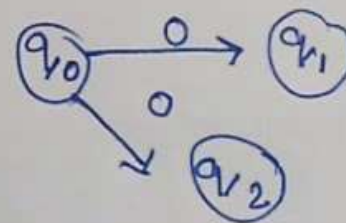
3> can solve the problem in polynomial time NP.

4> can determine what is the next step.

Non Deterministic Algorithm

1> in this case, the computer did not understand what is the actual output.

2>



3> can not solve the problem in polynomial time NP

4> can not determine.

Overview of programming

Sequential flow: As the name suggests, all the steps are organised in a predefined sequence. For example, student fees calculation. Suppose admission fee per student is A, per semester fee is S, total number of semesters in a program is T, total semester cost of the program C and total cost V for a program to be calculated and printed on a leaflet, then the algorithm will look as:

INPUT A, S, E, T

$C = S * T$

$V = A + C$

Print Leaflet

Overview of programming

Conditional flow: If in an algorithm, a decision is made based on a condition, then that algorithmic flow is known as conditional flow. For example, a student's result system. If a student scores more than 40%, then print pass or print fail.

INPUT P

If $P > 40\%$

Then Print 'Pass'

Or else

Print 'Fail'

Repetitive flow: It indicates the repetition of flow. If the condition will be true, the same steps will be followed.

Overview of programming

What is Flow chart?

The first design of flowchart goes back to 1945 which was designed by **John Von Neumann**. Unlike an algorithm, Flowchart uses different symbols to design a solution to a problem. It is another commonly used programming tool.

In general, a flowchart is a diagram that uses different symbols to visually present the flow of the data. By looking at a flow chart, one can understand the operations and sequence of operations performed in a system. This is why flowchart is often considered as a blueprint of a design used for solving a specific problem.

A flowchart is defined as a symbolic or a graphical representation of an algorithm that uses different standard symbols.

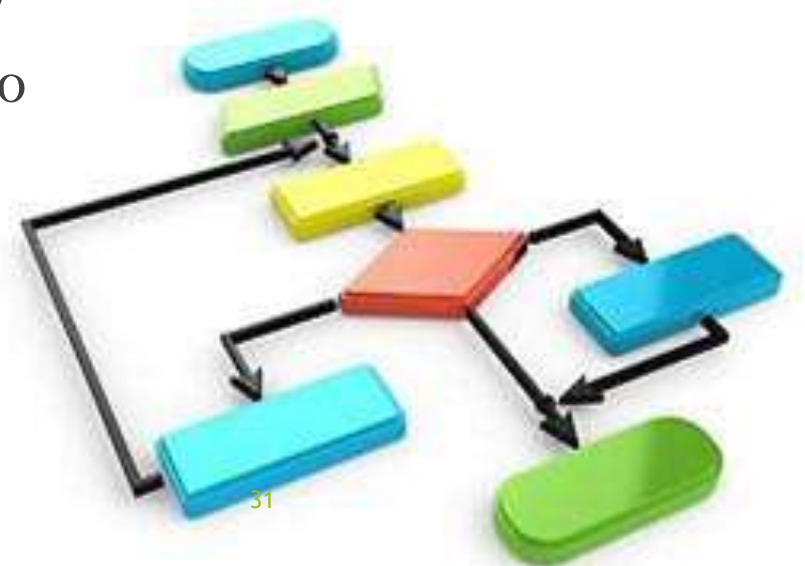


John Von Neumann (1903-1957)

Overview of programming










What is Flow chart?

Algorithm is a step-by-step sequence of instructions to solve a problem. Similarly, flowchart is a pictorial representation of the problem-solving sequence. Alternatively, it can be defined as the graphical representation of an algorithm. It acts like a ready roadmap for a programmer and guides from the starting point to the final outcome of the program. Flowchart includes a finite number of symbols and instructions, which are connected to each other by an arrow mark.



Overview of programming

Symbols used in flowchart:

Symbols Used in Flowcharts			
Picture	Shape	Name	Action Represented
	Oval	Terminal Symbol	Represents start and end of the Program
	Parallelogram	Input/Output	Indicates input and output
	Rectangle	Process	This represents processing of action. Example, mathematical operator
	Diamond	Decision	Since computer only answer the question yes/no, this is used to represent logical test for the program
	Hexagon	Initialization/Preparation	This is used to prepare memory for repetition of an action
	Arrow Lines & Arrow Heads	Direction	This shows the flow of the program
		Annotation	This is used to describe action or variables
	Circle	On page connector	This is used to show connector or part of program to another part.
	Pentagon	Off-page connector	This is used to connect part of a program to another part on other page or paper

Overview of programming

Guidelines for drawing a flowchart:

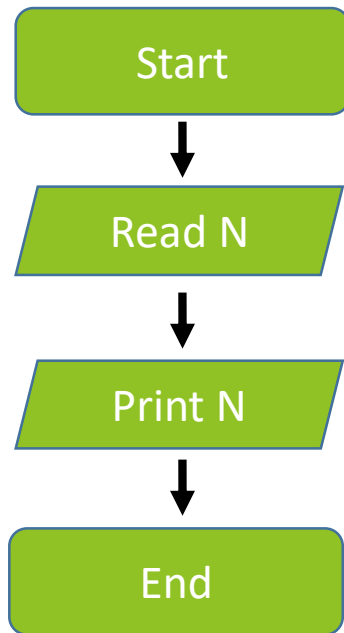
- ← The Title for every flowchart is compulsory.
- ← There must be START and END point for every flowchart.
- ← The symbols used in flowchart should have only one entry point on the top. The exit point for symbols (except for decision/diamond symbol) is on the bottom.
- ← There should be two exit points for decision symbol; exit points can be on the bottom and one side or on the sides.
- ← The flow of the flowchart is generally from top to bottom. But in some cases, it can also flow to upward direction.
- ← The direction of the flow of control should be indicated by arrowheads.
- ← The operations for every step should be written inside the symbol.
- ← The language used in flowchart should be simple so that it can be easily understood.
- ← The flowlines that show the direction of flow of flowchart must not cross each other.
- ← While connecting different pages of the same flowchart, Connectors must be used.

Overview of programming

Examples:

1. Design a flowchart that will accept and display a number. Write its equivalent algorithms.

Start Print N Read N End Algorithm:



Algorithm:

Step 1 : Read in the value of N.

Step 2 : Print the value of N.

Overview of programming

Self-Assessment Question

1. _____ of data should be provided to make a successful solution for any computer-based problem.
 - A) Confidentiality
 - B) Privacy
 - C) Security
 - D) All of the above

Overview of programming

Self-Assessment Question

2. In algorithm, the number of steps should be _____, so that after any mechanical execution, it should stop automatically.
- A) Finite
 - B) Definite
 - C) General
 - D) Effective

Overview of programming

Self-Assessment Question

3. Algorithm should be _____ so that it can give a unique solution to the problem it is designed for.
- A) Finite
 - B) Definite
 - C) General
 - D) Effective

Overview of programming

Self-Assessment Question

4. If the algorithm is capable of exploring a large number of alternatives simultaneously to reach out to a correct solution, then it is known as_____.

- A) Deterministic
- B) Random
- C) Non-deterministic
- D) Direct

Overview of programming

Self-Assessment Question

5. When in an algorithm, a decision is made based on a condition, then that algorithmic flow is known as conditional flow.

- A) True
- B) False

Programming and Implementation Issues

Overview of programming

What is Programming and implementation issues?

By now, you would have learnt how to solve a computational problem using sequential or computation steps. But how you will instruct the steps to the computer? The programming language will do that. Programming language converts each sequence of computational steps into understandable notations to the computer. This notation with any specific programming language is known as a program. Developing that program is known as programming and the developer is called a programmer.

Overview of programming

What is Program design and implementation issues ?

Program Design Language (or PDL, for short) is a method for designing and documenting methods and procedures in software. It is related to pseudocode, but unlike pseudocode, it is written in plain language without any terms that could suggest the use of any programming language or library.

PDL was originally developed by the company Caine, Farber and Gordon and has been modified substantially since they published their initial paper on it in 1975. It has been described in some detail by Steve McConnell in his book Code Complete.

Overview of programming

Top–Down Design:

This approach is adopted when you find it difficult to manage a big problem. In such cases, the big problem is divided into a number of smaller and simpler problems which can be handled easily. It works as:

1. The main program is written first and tested before the sub-programs are written.
2. Stubs are created to replace the actual sub-program and see the data flow.
3. Each module is written and tested.
4. A simple main program is written to test the sub-program.
5. If it runs successfully, then sub-programs are tested with the main program.
6. If the whole program runs successfully, then write and recheck it.

Overview of programming

Advantages:

1. Easy to discover the error in each module.
2. Easy to debug.
3. Small modules are desirable, maximum 100 lines long.

Overview of programming

Bottom–Up Design

- It is just opposite to the Top–Down Approach. In this approach, basic sub-routines are written first then those sub-routines are used to make more sophisticated sub-routines. Sometimes, very-low-level sub-routines that were predicted earlier may be wasted for a particular program. So, this approach is not advisable always.

Overview of programming

Programming Techniques

- There are two techniques for programming: Linear programming and Structured programming.
- **Linear programming** is similar to sequential programming. It is a straightforward way of programming and does not have any decision-making.

Overview of programming

Example of linear Programming:

Step 1: Input values for P and Q →	Read data value
Step 2: Add Q to P and store in ADD →	Compute result and use that if necessary to compute desired answer
Step 3: Display the value of ADD →	Print the answer
Step 4: End →	Stop

Overview of programming

Structured programming

Structured programming includes conditional flow. In terms of complex program, branching and looping are very important to write the algorithm. Branching is possible by decision-making. For example,

```
#include<stdio.h>
main()
{
    int a, b, c;
    printf("enter two numbers for a and b");
    scanf ("%d,%d",&a,&b);
    c= a+b;
    printf("The of two numbers %d and %d is %d", a, b, c);
    getch();
}
```


Overview of programming

Structured programming

Structured programming is used to write less error-prone complex programs that are easy to debug. In this technique, the whole program is broken into separate pieces of modules. Each module in turn is broken down into smaller pieces which can also be further subdivided. Modules must be chosen such that you can specify how they are to interact.

Overview of programming

Advantages of Structured programming

- ← The heart of efficient programming is problem decomposition. It breaks the big problem into small tasks. With structured programming, program can be written more easily and quickly.
- ← Errors can be found easily, hence it takes less time to debug.
- ← Easily maintainable.
- ← For application program development, productivity becomes high with structured programming.

Overview of programming

Basic constructs of Structured programming

In structured programming three constructs are available. They are sequence, selection and repetition.

Sequence: It is a continuous set of actions where one is followed by another till the expected result is obtained. The structure of sequence constructs is given below:

Statement 1

Statement 2

Statement 3

Statement n

Selection: It is used in conditional execution. When you take some action based on the results of logical test done by you, then that will be considered as selection constructs. The structure of a selection constructs is illustrated below:

Overview of programming

Basic constructs of Structured programming

If (condition is true)

Sequence of statement

Else

Another sequence of statement

end if

Iteration: Iteration is the repetitive execution of consecutive instructions. It is also called looping. It may either be conditional looping or unconditional looping. Conditional looping is executed until some logical condition has been satisfied, but in unconditional looping, the instructions are repeated for a specified number of times.

Overview of programming

Modular Designs of Program

- ← Initialization
- ← Input
- ← Input data validation
- ← Processing
- ← Output
- ← Error handling
- ← Closing procedure

Overview of programming

Self-Assessment Question

6. _____converts each sequence of computational steps into understandable notations to the computer.

- A) Flowchart
- B) Algorithm
- C) Programing Language
- D) Programmer

Overview of programming

Self-Assessment Question

7. In _____ design, the big problem is divided into a number of smaller and simpler problems which can be handled easily.

- A) Top-down
- B) Bottom-up
- C) Sequential
- D) Flowchart

Overview of programming

Self-Assessment Question

8. _____ is repetitive execution of consecutive instructions, which is also known as looping.

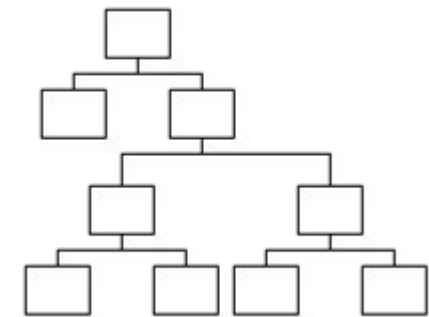
- A) Selection
- B) Development
- C) Sequence
- D) Iteration

Top-down Design and Step-wise Refinement

Overview of programming

What is Top–down design?

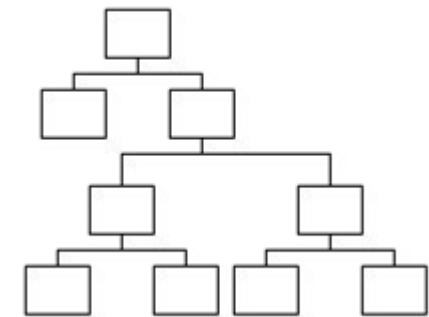
- A top–down approach (also known as stepwise design) is essentially the breaking down of a system to gain insight into the sub-systems that make it up. In a top–down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. Once these base elements are recognized then we can build these as computer modules. Once they are built we can put them together, making the entire system from these individual components.



Overview of programming

Top-down design

- Till now, we have learnt various methods to solve a problem using the computer oriented programming method. It is clear that from problem identification to execution, the data are the main entity used throughout in the process. To solve a problem, algorithm is a step-by-step finite sequence of instructions. It may be designed based on top-down approach or bottom-up approach. As we have seen in the top-down algorithm design, a hierarchy of modules is used. We break the problem or specification into simpler and simpler pieces and control is passed downward to the structure. Each piece or module or method serves a specific function.



Overview of programming

What is stepwise refinement?

- Stepwise Refinement—A Definition. Stepwise Refinement. A way of developing a computer program by first describing general functions, then breaking each function down into details which are refined in successive steps until the whole program is fully defined. Also called top–down design.

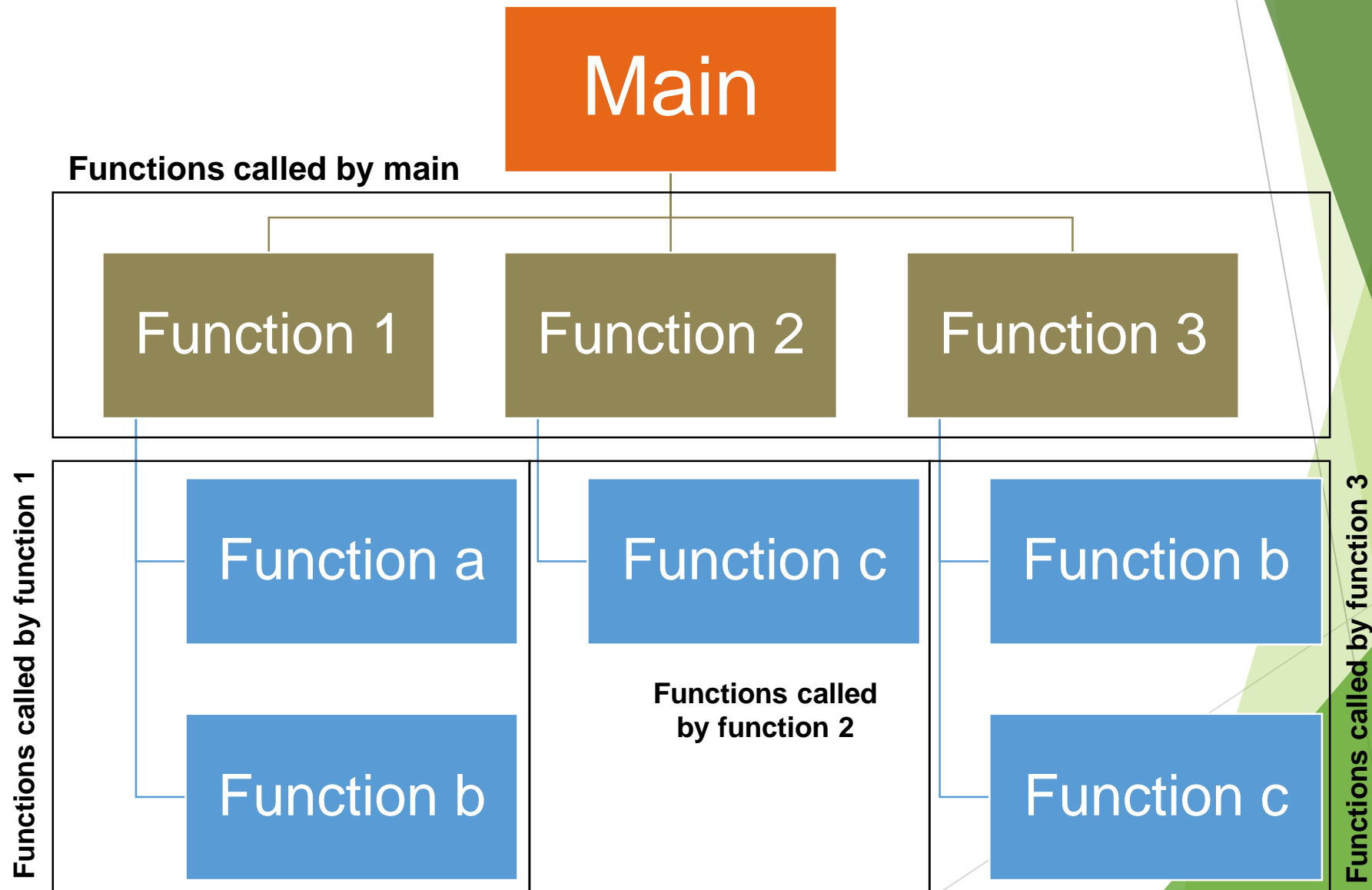
Overview of programming

Example of stepwise refinement

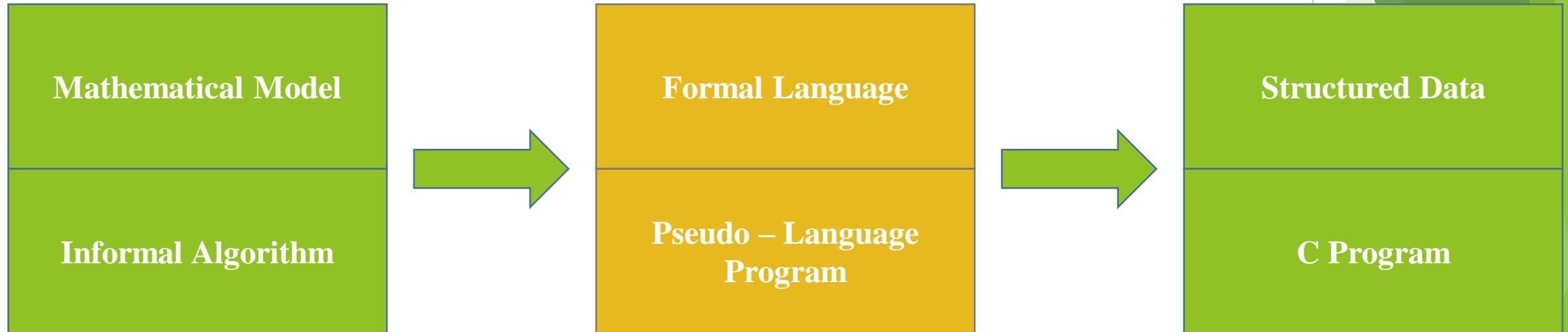
Below is an example of the top-level design from earlier refined to an even more detailed design.

```
Set the search result to false
Start at the beginning of the list
For each item in the list,
  if the list item matches the item we are looking for
    let the search result = true
  end - if
end - for
display the search result state.
```

Overview of programming



Overview of programming



Overview of programming

Self-Assessment Question

9. A module can be subdivided until it contains _____ elementary process.

- A) No
- B) Single
- C) Multiple
- D) Double

Overview of programming

Self-Assessment Question

10. Choose the correct sequence of step-wise refinement program.

1. Formal algorithm design
2. Program implementation phase
3. Mathematical model define.

- A) 1>2>3
- B) 2>3>1
- C) 3>1>2
- D) 2>1>3

Overview of programming

Self-Assessment Question

11. In _____stage, problem is represented using any mathematical model.

- A) Formal algorithm design
- B) Program implementation phase
- C) Informal representation
- D) Problem definition

Overview of programming

Self Assessment Question

12. Algorithm is a step-by-step infinite sequence of instructions.

- A) True
- B) False

Programming Languages

Overview of programming

Programming Language Concepts

- What is a programming language?
- Why there are so many programming languages?
- What are the types of programming languages?
- Does the world need new languages?

Overview of programming

What is a Programming Language?

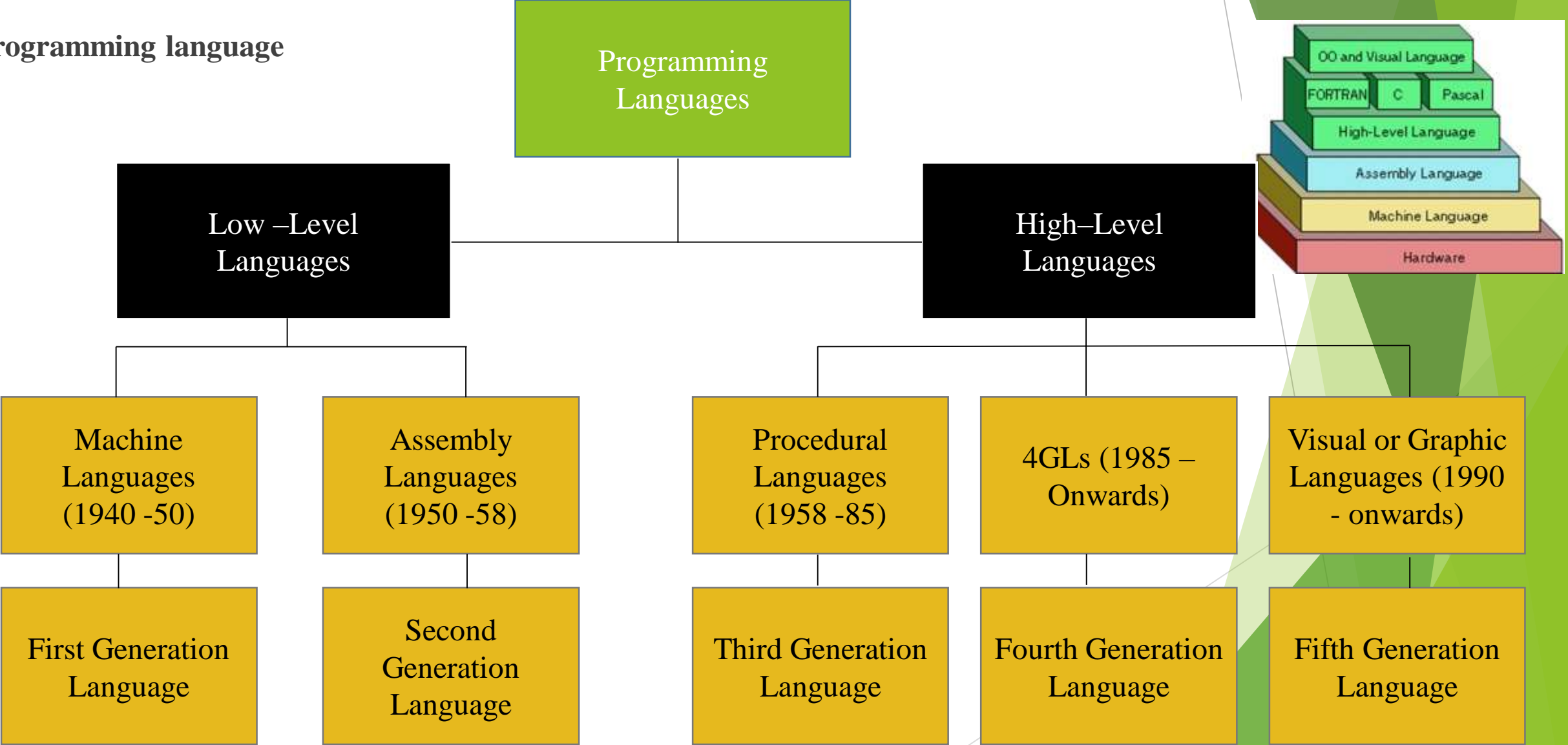
- ◀ A programming language is a set of rules that provides a way of telling a computer what operations to perform.
- ◀ A programming language is a set of rules for communicating an algorithm.
- ◀ It provides a linguistic framework for describing computations.
- ◀ **Introduction:** A programming language is a notational system for describing computation in a machine-readable and human-readable form.

Overview of programming

- ⬅ A programming language is a tool for developing executable models for a class of problem domains.
- ⬅ English is a natural language. It has words, symbols and grammatical rules.
- ⬅ A programming language also has words, symbols and grammatical rules.
- ⬅ The grammatical rules are called syntax.
- ⬅ Each programming language has a different set of syntax rules.

Overview of programming

Programming language



Overview of programming

Levels of Programming language

High-level program

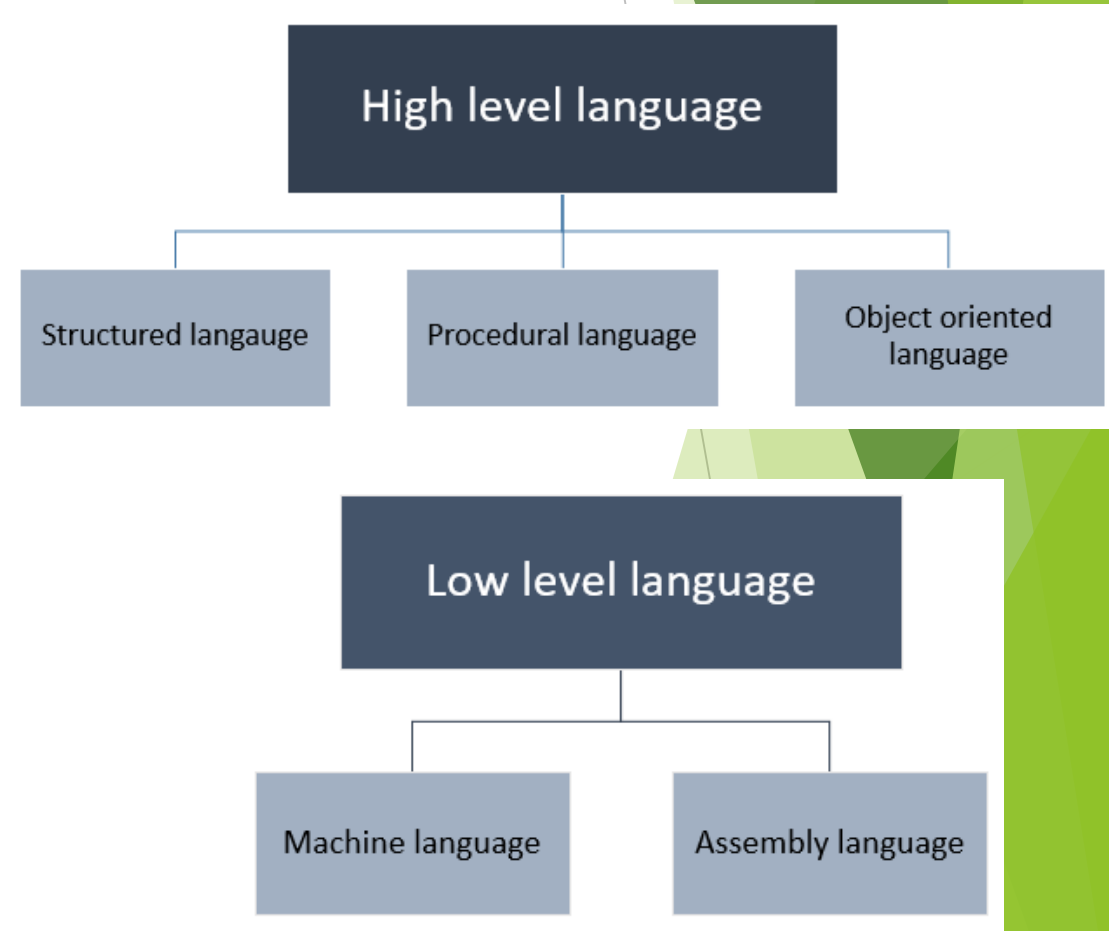
```
class Triangle {  
    ...  
    float surface()  
    return b*h/2;  
}
```

Low-level program

```
LOAD r1,b  
LOAD r2,h  
MUL r1,r2  
DIV r1,#2  
RET
```

Executable Machine code

```
0001001001000101  
0010010011101100  
10101101001...
```



Overview of programming

First Generation Languages:

➤ Machine language

- ⬅ Operation code – such as addition or subtraction.
- ⬅ Operands – that identify the data to be processed.
- ⬅ Machine language is machine dependent as it is the only language the computer can understand.
- ⬅ Very efficient code but very difficult to write.

Overview of programming

Advantages:

- ◀ Due to direct handling of memory and registers, the code runs very fast.
- ◀ With the use of machine language, storage, CPU and registers can be managed efficiently.

Disadvantages:

- ◀ Difficult to learn machine languages, so in case of error, very difficult to develop and debug programs as well.
- ◀ Programs in machine language are not portable, because the machine language for one computer could be significantly different from the other.
- ◀ Programs are more error prone and difficult to edit, since you have to remember binary equivalent of instructions.

Overview of programming

Second-Generation Languages:

➤ Assembly languages

- ✦ Symbolic operation codes replaced binary operation codes.
- ✦ Assembly language programs needed to be ‘assembled’ for execution by the computer. Each assembly language instruction is translated into one machine language instruction.
- ✦ Very efficient code and easier to write.

For example,

MOV X, Y;

Mnemonics ‘MOV’ is used to represent move or transfer operation and to specify the register location, where the data is located, X, Y are used.

Overview of programming

Advantages:

- ◀ Comparatively easier than the machine language as it is easy to remember the symbolic names of instructions.
- ◀ No need to know the exact address or location of data items.
- ◀ An assembler is useful for detecting programming errors.

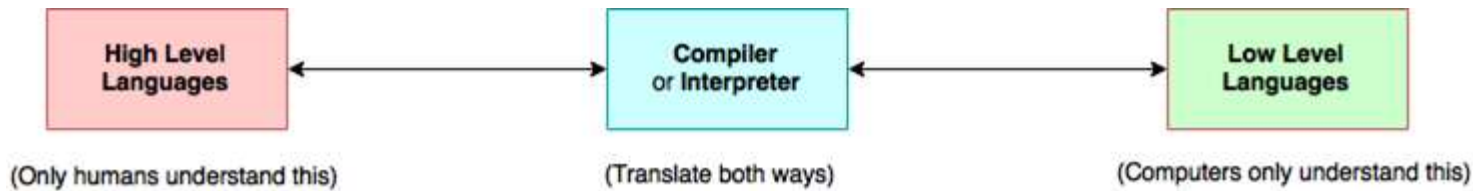
Disadvantages:

- ◀ Direct execution is not possible.
- ◀ Like machine language, this language is also machine dependent, so not portable.
- ◀ High level of programming skill required

Overview of programming

Third-Generation Languages:

- ↖ Closer to English but included simple mathematical notation.
- ↖ Programs written in source code which must be translated into machine language programs called object code.
- ↖ The translation of source code to object code is accomplished by a machine language system program called a **compiler**.



- ↖ Alternative to compilation is interpretation that is accomplished by a system program called an interpreter.
- ↖ Common third-generation languages include:
 - ↖ FORTRAN
 - ↖ COBOL
 - ↖ C, C++ and Java
 - ↖ Visual Basic

Overview of programming

Procedural Language

Advantages:

- ✦ Easy to write, read and understand the program written in procedural language.
- ✦ Libraries of subroutines can be incorporated that can be used in many other programs.
- ✦ Translator converts the source code into object code, which makes the language portable.

Disadvantages:

- ✦ To write the program in third-generation language, more memory is required.
- ✦ Program execution time is more compared with machine or assembly language.

Overview of programming

+

Fourth-Generation Languages:

- ◀ A high-level language (4GL) that requires fewer instructions to accomplish a task than a third-generation language.
- ◀ Used with databases
 - ◀ Query languages
 - ◀ Report generators
 - ◀ Forms designers
 - ◀ Application generators

Overview of programming

Advantages:

- ← Compared with 3GLs, it is easier to use.
- ← These languages effectively utilize programmer's capabilities.
- ← It reduces the overall time, effort and cost of the software development.

Disadvantages:

- ← In development process more planning and control is required.

Overview of programming

Fifth-Generation Languages:

- ← Declarative languages
- ← Functional(?): Lisp, Scheme, SML
 - ← Also called applicative
 - ← Everything is a function
- ← Logic: Prolog
 - ← Based on mathematical logic
 - ← Rule or Constraint based
- ← Although no clear definition at present, natural language programs generally can be interpreted and executed by the computer with no other action by the user than stating their question.
- ← Limited capabilities at present.

Overview of programming

Visual or Graphics Languages

Advantages:

- ◀ It is a high-level language, uses artificial intelligence to develop software program.
- ◀ Use problem constraints and logic to solve the problem.

Disadvantages:

- ◀ There are some doubts on the existence of the fifth-generation language.
- ◀ Difficult to use for software development.

Overview of programming

The Principal paradigms

- Imperative Programming (C)
- Object Oriented Programming (C++)
- Logic/Declarative programming (Prolog)
- Functional/Applicative programming (Lisp)

Overview of programming

Traditional Programming Languages

➤ FORTRAN

- ⬅ **FOR**mula **TRAN**slation.
- ⬅ Developed at IBM in the mid-1950s.
- ⬅ Designed for scientific and mathematical applications by scientists and engineers.

➤ COBOL

- ⬅ **CO**mmun **B**usiness **O**riented **L**anguage.
- ⬅ Developed in 1959.
- ⬅ Designed to be common to many different computers.
- ⬅ Typically used for business applications.

Overview of programming

Traditional Programming Languages

➤ BASIC

- ✚ Beginner's All-purpose Symbolic Instruction Code.
- ✚ Developed at Dartmouth College in mid-1960s.
- ✚ Developed as a simple language for students to write programs with which they could interact through terminals.

➤ C

- ✚ Developed by Bell Laboratories in the early 1970s.
- ✚ Provides control and efficiency of assembly language while having third-generation language features.
- ✚ Often used for system programs.
- ✚ UNIX is written in C.

Overview of programming

Object-Oriented Programming Languages

➤ **Simula**

- ✚ First Object-Oriented Language.
- ✚ Developed by Ole Johan Dahl in the 1960s.

➤ **Small talk**

- ✚ First purely object-oriented language.
- ✚ Developed by Xerox in mid-1970s.
- ✚ Still in use on some computers.

Overview of programming

Object-Oriented Programming Languages (Contd.)

➤ C++

- ↩ It is C language with additional features.
- ↩ Widely used for developing system and application software.
- ↩ Graphical user interfaces can be developed easily with visual programming tools developed by Ole Johan Dahl in the 1960s.

➤ JAVA

- ↩ An object-oriented language similar to C++ that eliminates lots of C++'s problematic features.
- ↩ Allows a web page developer to create programs for applications, called applets that can be used through a browser.
- ↩ The objective of JAVA developers is that JAVA can be machine, platform and operating system independent.

Overview of programming

Special Programming Languages

➤ Scripting Languages

- ← JavaScript and VB script

- ← PHP and ASP.

- ← Perl and Python

➤ Command Languages

- ← Sh, csh and bash

➤ Text Processing Languages

- ← LaTeX and PostScript

Overview of programming

Special Programming Languages (Contd.)

➤ HTML

- ↩ Hypertext Markup Language.
- ↩ Used on the internet and the World Wide web (WWW).
- ↩ Web page developer puts brief codes called tags in the page to indicate how the page should be formatted.

➤ XML

- ↩ Extensible Markup Language.
- ↩ A language used for defining other languages.

Overview of programming

Self-Assessment Question

13. In 1936, Konrad Zuse designed a mechanical computer named as _____ .

- A) Z-1
- B) M-1
- C) P-1
- D) X-1

Overview of programming

Self-Assessment Question

14. _____ language is the oldest and most basic of all the languages.

- A) Assembly
- B) Procedural
- C) Visual
- D) Machine

Overview of programming

Self-Assessment Question

15. In machine language, all the instructions are provided to the computer in a _____ format.

- A) Hexadecimal
- B) Binary
- C) Electrical
- D) Numerical

Overview of programming

Self-Assessment Question

16. _____-generation language is called as procedure-oriented language.

- A) First
- B) Second
- C) Third
- D) Fourth

Overview of programming

Self-Assessment Question

17. C, C++, JAVA are the examples of _____-generation language.

- A) First
- B) Second
- C) Third
- D) Fourth

Overview of programming

Self-Assessment Question

18. Match the following:

Set A

Set B

1) C, C++, JAVA, BASIC

2) PROLOG, Mercury

3) VB, SQL, MS Excel

a) Fifth generation

b) Fourth generation

c) Third generation

A) 1.a, 2.b, 3.c

B) 1.c, 2.a, 3.b

C) 1.c, 2.b, 3.a

D) 1.a, 2.c, 3.b

Overview of programming

Self-Assessment Question

19. In programming with _____-generation language, a programmer prefers to solve problems based on the constraints provided with the program rather than using an algorithm to develop the program.

- A) Fifth
- B) Second
- C) Third
- D) Fourth

Overview of programming

Self-Assessment Question

20. Compared with low-level language, it is difficult to write codes in high-level language.

- A) True
- B) False

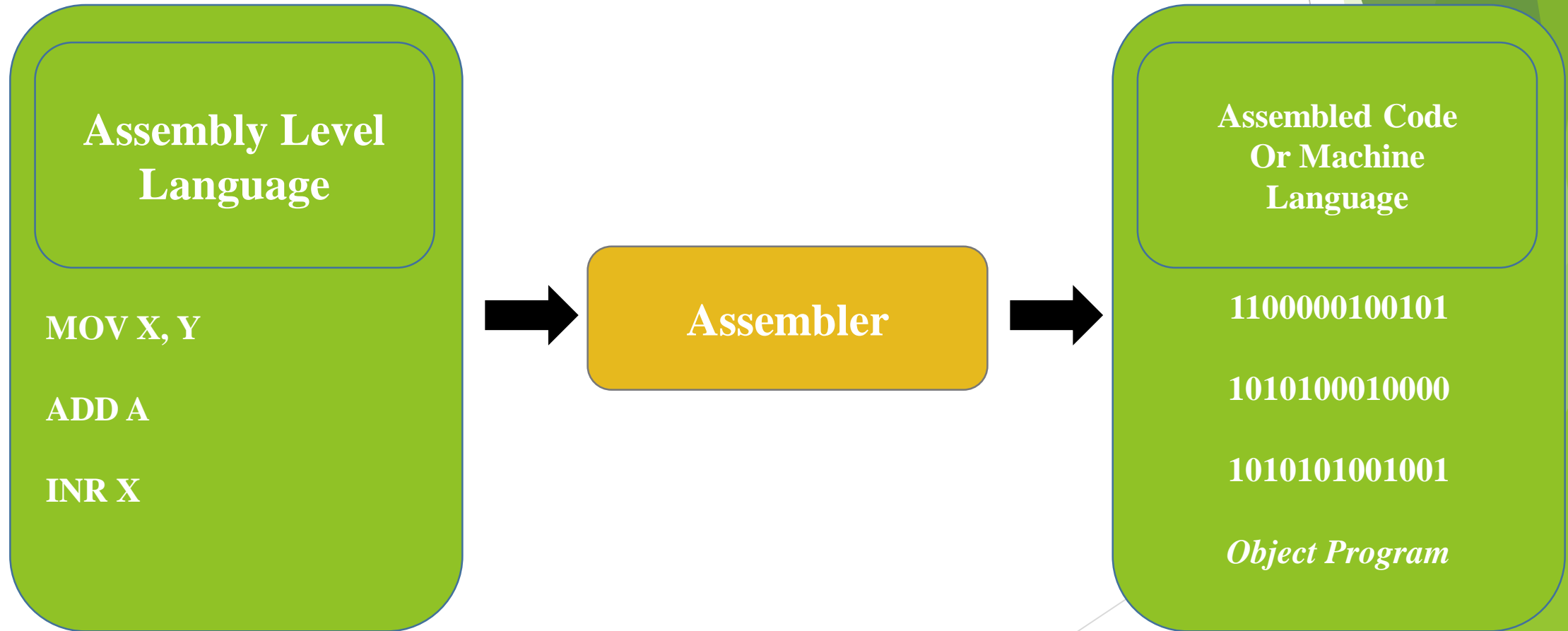
Overview of programming

➤ Assemblers

- ◀ An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer.
- ◀ An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components.
- ◀ An assembler is sometimes referred to as the compiler of the assembly language. It also provides the services of an interpreter.
- ◀ This is a list of assemblers: computer programs that translate assembly language source code into binary programs.

Overview of programming

➤ Assemblers



Overview of programming

➤ Assemblers

Two popular examples of assemblers are Microsoft Assembler Program (MASM) and Borland Turbo Assembler Program (TASM).

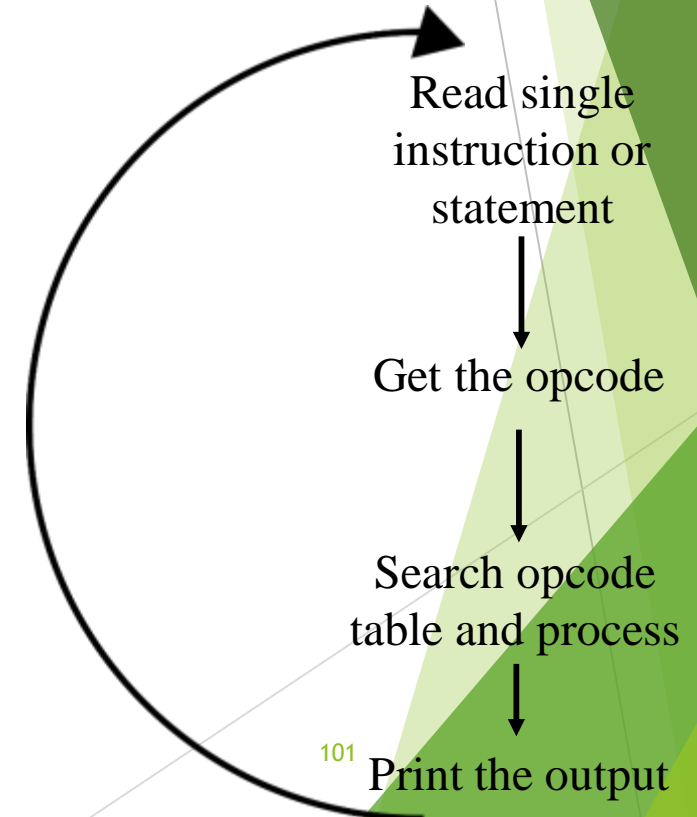
There are two types of assemblers: one pass and two pass.

Pass one:

- It assigns addresses to all statements in source code.
- Save addresses assigned to labels for use in pass two.
- Process directives.

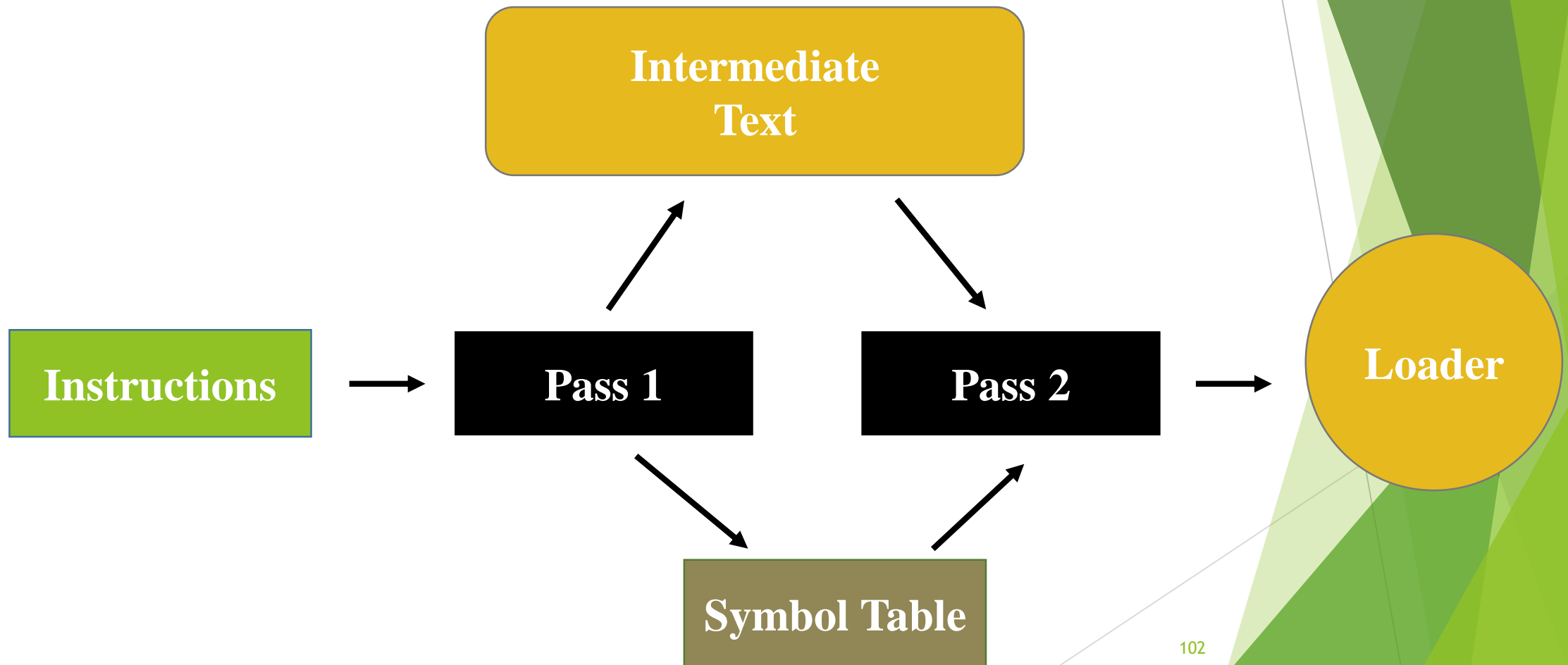
Pass two:

- Translate instructions.
- Convert label to addresses.
- Generate values defined by BYTE and WORD.
- Process the directives not done in pass one.
- Write object code to output device.



Overview of programming

➤ Assemblers



Overview of programming

➤ Compilers

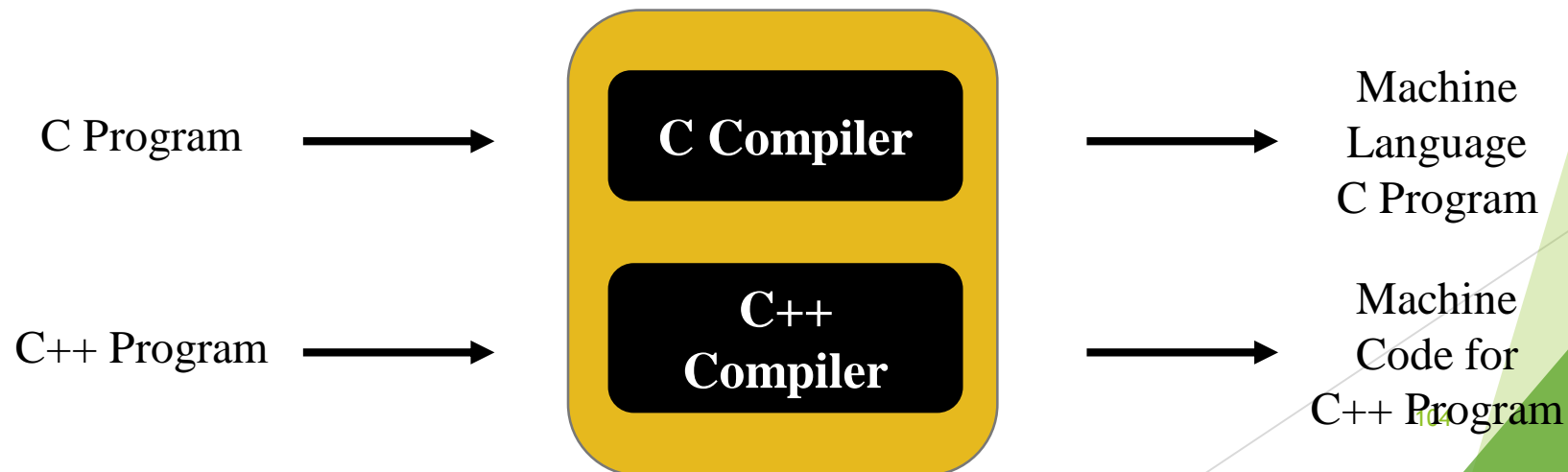
- ◀ Compilers are used to convert high-level languages (like C, C++) into machine code.
- ◀ For example, GCC and Microsoft Visual Studio.



Overview of programming

➤ Compilers

◀ Every language has its own assembler and it can translate only those programs which are written in that particular language as illustrated below.



Overview of programming

➤ Compilers

◀ A compiler can detect the following errors:

- a) Illegal characters
- b) Illegal combination of characters
- c) Improper sequencing of instructions in a program

Overview of programming

➤ Interpreter

- ◀ An interpreter is a computer program which executes a statement directly (at runtime).
- ◀ For example, python , LISP and Ocaml.

Advantages of interpreter over compiler:

- ◀ It quickly translates the source program into object program.
- ◀ Interpreters are easy to write and are simpler programs compared with compilers.
- ◀ It does not require large memory space.

Disadvantages:

- ◀ It is time-consuming because it translates each statement in the source program line by line.

Overview of programming

Self-Assessment Question

21. Which is responsible for transferring object program into the computer's primary memory?

- A) Assembler
- B) Linker
- C) Loader
- D) Compiler

Overview of programming

Self-Assessment Question

22. What type of errors are not detected by a compiler?

- A) Illegal characters
- B) Semantical errors
- C) Syntactical errors
- D) Logical errors

Overview of programming

Self-Assessment Question

23. _____ translates only one statement or instruction at a time.

- A) Assembler
- B) Compiler
- C) Interpreter
- D) Linker

Overview of programming

Self-Assessment Question

24. Interpreter is time-consuming because it translates each statement in the source program line by line.

State where true or false.

- A) True
- B) False

Overview of programming

Summary

- ✚ C is a general purpose, structured programming language.
- ✚ Software development starts with required information gathering for the proposed software which is known as the problem definition phase. The next phase is problem solving. A software is developed to solve certain problems.
- ✚ Few concerns which may create problem while identifying information needs are speed of processing, volume of work, reliability, accuracy, security and last but not the least cost. Costs include manpower, hardware, software, consumables and supplies and overhead costs.
- ✚ Algorithms are self-contained set of operations to be performed step by step. Its basic properties are: finiteness, definiteness, generality, effectiveness and input–output.
- ✚ Basic types of algorithm are: deterministic, non-deterministic, random algorithm, direct and indirect algorithm and infinite algorithm.
- ✚ Programming language consists of a set of commands which is understood by computers. Basically, programming languages are of two types: low-level and high-level languages.
- ✚ Machine language is the oldest and most basic of all the languages. In this language, all the instructions are given to the computer in binary format which accepts only '0' and '1'.

Overview of programming

Summary (Contd.)

- ✚ In second-generation language, mnemonics are used to replace the binary codes (opcodes) of machine language.
- ✚ An assembler converts the instructions written in second-generation language into machine language.
- ✚ Third-generation language is a procedure-oriented language in which a programming task is broken into a collection of variables, data structures and functions known as procedures. C, C++, JAVA and BASIC are the examples of third-generation language.
- ✚ Fourth-generation languages are more focused on what is to be accomplished rather than how it is to be accomplished. It includes interactive coding in the form of an on-screen menu selection to formulate an enquiry. dBase, Foxbase, Foxpro, Oracle, SQL, MS-Excel are few examples of 4GL.
- ✚ While programming with fifth-generation language, a programmer prefers to solve problems based on the constraints provided with the program rather than using an algorithm to develop the program. PROLOG (programming logic), OPS5 and Mercury are examples of fifth-generation language.
- ✚ An assembler translates the source code of assembly language program into machine-level language, which is known as object program.
- ✚ Similar to compiler and assembler, interpreters also work as translators and are easy to write and are simpler programs compared with compiler.

Overview of programming

Terminal Questions

- ◀ List the properties of algorithm and discuss them briefly.
- ◀ Define flowchart with example.
- ◀ Describe top–down system design technique along with their advantages and disadvantages.
- ◀ Compare advantages and disadvantages of the various types of programming languages.
- ◀ Discuss how an assembler works.
- ◀ Identify the differences between one pass and two pass assembler.
- ◀ Discuss the working of compiler and interpreter.

Overview of programming

Answer Keys

Self Assessment	
Question No.	Answers
1	D
2	A
3	D
4	C
5	A
6	C
7	A
8	D
9	B
10	C
11	C
12	B

Self Assessment	
Question No.	Answers
13	A
14	D
15	B
16	C
17	C
18	B
19	A
20	B
21	B
22	D
23	C
24	A

Overview of programming

Activity : Attend Both 1 and 2 Activity

1. **Activity Type:** Offline

Duration: 30 Minutes

Description: Discuss on the structure of algorithm and develop a basic structure of an algorithm which has top–down design approach.

OR

- ✦ Take a scenario of making pancakes. Follow the top–down design approach and write the procedure of making pancakes.
- ✦ Draw a top–down design tree to three-level depth for making a modern First-Person Shooter.

2. **Activity Type:** Online and Offline

Duration: 15 Minutes

Description: Write a simple program using if-else statement.

Introduction to Information Security

Document Links

Topics	URL	Notes
Introduction to C programming	https://www.slideshare.net/DiwakarPratapSinghDeva/introduction-to-problem-solving-in-c	This link explains the definition and overview of C
Top down design & Stepwise Refinement	https://www.sqa.org.uk/e-learning/Planning02CD/page_14.htm https://web.stanford.edu/class/cs106j/lectures/03-Stepwise-Refinement/03-Stepwise-Refinement.pdf	This link guides the details about top down design & stepwise Refinement
Program design and implementation issues	http://csis.pace.edu/~marchese/CS389/L7/Ch7_summary.pdf	This link explain about design details
Types of Computer Languages	https://owlcation.com/stem/Types-of-Computer-Languages-with-Advantages-and-Disadvantages	This link describes complete details about languages
Compiler / Interpreter	https://www.tutorialspoint.com/computer_programming/computer_programming_environment.htm	This link explains programming environment

Overview of programming



Video Links

Topics	Links
Concepts of Algorithm, Flow Chart & C Programming	https://www.youtube.com/watch?v=DF2XAc07eI0
C Programming 03 - Algorithms and Flowcharts	https://www.youtube.com/watch?v=txSKBEuDLJ0
Introduction to Problem Solving and Programming	https://www.youtube.com/watch?v=8BeXwhIjq2g&list=PL94CA590D7781A9B9
How the Compiler Works: Compilation Stages	https://www.youtube.com/watch?v=IhC7sdYe-Jg https://www.youtube.com/watch?v=VDslRumKvRA https://www.youtube.com/watch?v=2dan4hJlOv0
Programming Concepts - Compilers & Interpreters	https://www.youtube.com/watch?v=1veGrAjgJBY
Downloading and Installing Compiler, IDE for C Programming	https://www.youtube.com/watch?v=5sOIuzmd4w&index=3&list=PLfVsf4Bjg79CZ5kHTiQHcm-l2q8j06ofd

Thank you