UNIT 2: Programming Paradigms

Unstructured programming, Structured programming, Object oriented programming, characteristics of a good program. Types of programming languages; Machine language, Assembly language; procedural language, Natural programming language, Visual programming language, graphical programming language, scripting language, Hypertext Markup language, Extensible Markup language. Program Translation: Program translation hierarchy; compiler, Interpreter, linker, loader; Features of a good programming language.

Unstructured Programming

A programming language in which the entire logic of the program is written as a single continuous (nonstop or unbroken) block is called "unstructured Programming".

Characteristics of Unstructured Programming

- There are both high- and low-level programming languages that use non-structured programming.
- A program in a non-structured language usually consists of sequentially ordered commands, or statements, usually one in each line. The lines are usually numbered or may have labels: this allows the flow of execution to jump to any line in the program.
- It also introduces basic control flow concepts such as loops, branches and jumps
- There is no concept of procedures in the non-structured paradigm, subroutines are allowed. No concept of local variables in non-structured programming
- There is no (automatic) context refresh when calling a subroutine, so all variables might retain their values from the previous call.
- The depth of nesting also may be limited to one or two levels.
- *Allows only basic data types, such as numbers, strings and arrays***.**
- **For example**: early versions of **BASIC** (such as MSX BASIC and GW-BASIC), JOSS, FOCAL, MUMPS, TELCOMP, COBOL, **machine-level code**.

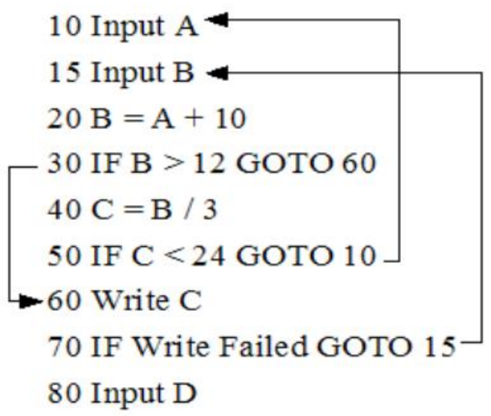
Limitations of Unstructured Programming

- Very difficult to modify and to debug.
- It is historically the earliest programming paradigm
- Produces spaghetti code i.e. hardly-readable code.

What is a spaghetti code ?

A complex and tangled control structure, especially one using many GOTO statements, exceptions, threads, or other "unstructured" branching constructs.

Example



```
10 Input A
15 Input B
20 B = A + 10
30 IF B > 12 GOTO 60
40 C = B / 3
50 IF C < 24 GOTO 10
60 Write C
70 IF Write Failed GOTO 15
80 Input D
```

Structured Programming

A programming language in which the entire logic of the program is written by dividing it into smaller units or modules is called "structured programming Language".

Characteristic of Structured Language

- In C, the smaller units are referred to as functions.
- A function is written separately to perform a specific task. Each function has a unique name. It is called for execution in the main body of program with reference to its name.
- Program written in structured programming language is very easy to modify and to debug.
- For example: C language.

What are the elements of structured programming

1. **Control Structures**
2. **Subroutines**
3. **Blocks**

Control structures

Following the structured program theorem, all programs are seen as composed of three control structures:

- "**Sequence**"-ordered statements or subroutines executed in sequence.
- "**Selection**"- one or a number of statements is executed depending on the state of the program. This is usually expressed with keywords such as **if..then..else..endif.**
- "**Iteration**"-a statement or block is executed until the program reaches a certain state, or operations have been applied to every element of a collection. This is usually expressed with keywords such as while, repeat, for **or do..until.**

Subroutines

Callable units such as procedures, functions, methods, or subprograms are used to allow a sequence to be referred to by a single statement.

Blocks

Blocks are used to enable groups of statements to be treated as if they were one statement.

Block-structured languages have a syntax for enclosing structures in some formal way, such as an if-statement bracketed by if..fi or a code section bracketed by BEGIN..END, or the curly braces {...} of C and many later languages.

Difference between Structured and Unstructured Programming

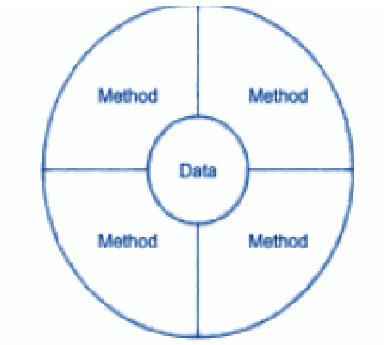
	Structured Language	**Unstructured language**
1	Code compartmentalization can be done	Code compartmentalization cannot be done
2	Loops can be created	Loops cannot be created
3	These are new languages	These are old languages
4	Do not require a strict field concept.	Strict field concept is used mostly.
5	Examples: C, C++, JAVA, ADA, PASCAL, MODULA -2	Examples: BASIC, COBOL, FORTRAN

Object Oriented Programming

Object means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic

- The first step in OOP is to identify all the objects the programmer wants to manipulate and how they relate to each other, an exercise often known as data modeling.
- Once an object has been identified, it is generalized as a class of objects which defines the kind of data it contains and any logic sequences that can manipulate it.
- Each distinct logic sequence is known as a method. Objects communicate with well-defined interfaces called *messages*.



Basic concepts of OOP

<u>Object</u>

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

<u>Class</u>

Collection of objects is called class. It is a logical entity.**Classes define:**

- Set of *attributes*
  - Also called *state*
  - Represented by variables and properties
- Behavior
  - Represented by methods

<u>Inheritance</u>

When one object acquires all the properties and behaviors of parent object that is known as inheritance. The child will have all the properties of the parent and can have its own additional properties as well.

<u>Polymorphism</u>

When **one task is performed by different ways**, it is known as polymorphism. For example: water exists in 3 different forms- ice, water (liquid) and water vapor.

<u>Abstraction</u>

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

Encapsulation

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

Advantages of OOPS

Some advantages of OOPs over Procedure-oriented programming language

1. Emphasis is on data rather than procedure.
2. Programs are divided into objects.
3. Data Structures are designed such that they characterize the objects.
4. Methods that operate on the data of an object are tied together in the data structure.
5. Data is hidden and cannot be accessed by external functions.
6. Objects may communicate with each other through methods.
7. New data and methods can be easily added whenever necessary.
8. Bottom-up approach

What is Procedure Oriented Programming (POP) ?

- Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call.
- Procedures, also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out.
- Any given procedure might be called at any point during a program's execution, including by other procedures or itself.
- Procedural programming languages include C.

What are the characteristic of procedure oriented programming (POP) ?

- Large problems are divided into smaller programs known as functions.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transfer data from one form to another.
- Employs top-down approach in program designing.
- In the cases of large program, bringing change is difficult and time consuming.
- Appropriate and effective techniques are unavailable to secure data of a function from others.

Difference Between POP and OOP

	POP	OOP
1	In POP, program is divided into small parts called functions	Program is divided into parts called objects
2	Importance is not given to data but to functions as well as sequence of actions to be done.	Importance is given to the data rather than procedures or functions because it works as a real world
3	Top Down approach	Bottom Up approach
4	Data can move freely from function to function in the system	Objects can move and communicate with each other through member functions
5	To add new data and function in POP is not so easy	OOP provides an easy way to add new data and function.
6	Most function uses Global data for sharing that can be accessed freely from function to function in the system	Data can not move easily from function to function, it can be kept public or private so we can control the access of data.
7	Does not have any proper way for hiding data so it is less secure.	Provides Data Hiding so provides more security.
8	Example of POP are : C	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

Characteristics of a Good Computer Program

A good computer program should have following characteristics:

- **Portability**: Portability refers to the ability of an application to run on different platforms (operating systems) with or without minimal changes
- **Readability**: The program should be written in such a way that it makes other programmers or users to follow the logic of the program without much effort
- **Efficiency**: Every program requires certain processing time and memory to process the instructions and data
- **Structural**: To develop a program, the task must be broken down into a number of subtasks. These subtasks are developed independently
- **Flexibility**: A program should be flexible enough to handle most of the changes without having to rewrite the entire program
- **Generality**: Apart from flexibility, the program should also be general. Generality means that if a program is developed for a particular task, then it should also be used for all similar tasks of the same domain. For example, if a program is developed for a particular organization, then it should suit all the other similar organizations.
- **Documentation**: Documentation is one of the most important components of an application development

What is a Programming Language ?

A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.

Types Of Programming Language

- **Machine Language**
- **Assembly Language**
- **Procedural Language**
- **Natural Programming Language**
- **Visual Programming Language**
- **Scripting Language**
- **HTML**
- **XML**

Machine Language

Machine Language is the only language that is directly understood by the computer. It does not needs any translator program. We also call it machine code and it is written as strings of 1's (one) and 0's (zero). When this sequence of codes is fed to the computer, it recognizes the codes and converts it in to electrical signals needed to run it. For example, a program instruction may look like this: 1011000111101

Advantage Machine Language:
The only advantage is that program of machine language run very fast because no translation program is required for the CPU.

Disadvantages Machine Language:
1. It is very difficult to program in machine language. The programmer has to know details of hardware to write program.
2. The programmer has to remember a lot of codes to write a program which results in program errors.
3. It is difficult to debug the program.

Assembly Language

Assembly languages have the same structure and set of commands as machine languages, but they enable a programmer to use names instead of numbers.

Assembler

A program for converting instructions written in low-level symbolic code into machine code.

Advantages of Assembly Language:

1.The symbolic programming of Assembly Language is easier to understand and saves a lot of time and effort of the programmer.

2.It is easier to correct errors and modify program instructions.

3.Assembly Language has the same efficiency of execution as the machine level language. Because this is one-to-one translator between assembly language program and its corresponding machine language program.

Disadvantages of Assembly Language:
1.One of the major disadvantages is that assembly language is machine dependent. A program written for one computer might not run in other computers with different hardware configuration.

Procedural Programming

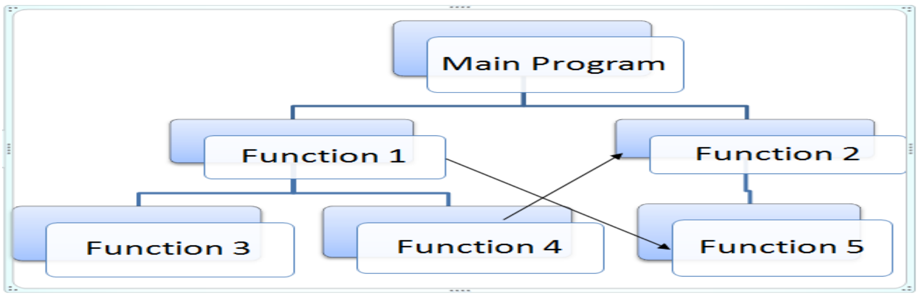
Procedural programming is the standard approach used in traditional computer language such as **C, Pascal, FORTRAN & BASIC.**

The basic idea is to have a program specify the sequence of steps that implements a particular algorithm .Procedural programming is a term used to denote the way in which a computer programmer writes a program.

Procedural programming creates a step by step program that guides the application through a sequence of instructions. Each instruction is executed in order.

Procedural programming focuses on processes. In procedural programming data and functions are stored in separate memory location

Programs are made up of modules, which are parts of a program that can be coded and tested separately, and then assembled to form a complete program.



Advantages of Procedural Programming:

- Its relative simplicity, and ease of implementation of compilers and interpreters.
- The ability to re-use the same code at different places in the program without copying it.
- An easier way to keep track of program flow.
- The ability to be strongly modular or structured.
- Needs only less memory.

Disadvantages of Procedural Programming:

- Data is exposed to whole program, so no security for data.
- Difficult to relate with real world objects.
- Difficult to create new data types reduces extensibility.
- Importance is given to the operation on data rather than the data.

Natural language :

Natural Language Programming (NLP) is an ontology-(assisted way of programming in terms of natural language sentences), e.g. English. A structured document with Content, sections and subsections for explanations of sentences forms a NLP document, which is actually a computer program.

Example

Inform 7 - Inform is very specifically for creating text adventure games

Python

```
Stage is a room.

The old lady is a woman in the Stage. Understand "mother" or
"stepmother" as the old lady. The old lady is active. The description
of the lady is "She looks plucked: thin neck with folds of skin
exposed, nose beaky, lips white. Perhaps when her fortunes are mended
her cosmetics too will improve."

The Prince is a man in the Stage. The description of the prince is
"He's tolerably attractive, in his flightless way. It's hard not to
pity him a little." The prince carries a glass slipper. The glass
slipper is wearable. Understand "shoe" or "heel" or "toe" or "foot"
as the slipper. The description of the slipper is "It is very small
for an adult woman's foot."
```

Complete code can be found here.
This is a small simple example...it can actually handle a surprisingly robust set of ideas.

It should be pointed out that the code isn't really a strange cypher where the constructs have hidden meanings...this code does more or less what you would expect. For example:

```
The old lady is a woman in the Stage. Understand "mother" or
"stepmother" as the old lady.
```

Creates an object that happens to be a female person, names that object "old lady", and places that object within the room object called the "Stage". Then two aliases ("mother" and "stepmother" are created that also both reference the "old lady" object.

Advantages
1. Relieves burden of learning syntax
2. No Training
3. Spoken Natural language allow busy hand

Disadvantages
1. Requires clarification dialogue
2.May require more keystrokes
3. May not show context
4. Is unpredictable
5. Miss interpretation because of Knowledge

Visual programming language :

Visual programming language (VPL) is a programming language that uses graphical elements and figures to develop a program.

VPL employs techniques to design a software program in two or more dimensions, and includes graphical elements, text, symbols and icons within its programming context.

Visual programming language is also known as executable graphics language.

Visual programming language enables the development of software programs by eliminating textual software code with a series of visual graphics elements. VPL incorporates these graphical elements as the primary context of the language arranged in a systematic order. The graphics or icons included within a visual program serve as input, activities, connections and/or output of the program.

Visual language has a few types, such as icon-based languages, diagramming languages and form-based language. Visual languages should not be confused with GUI-based programming language as they only provide graphical program authoring services. However, their code/context is completely textual.

Examples

- MIT Scratch
- App Inventor for Android
- Microsoft Kodu
- Alice
- StarLogo

Advantages

- Rapid application development,
- Sophisticated UI development

Disadvantages

- Not platform independent
- Use a lot more memory and CPU for their dynamic needs

Web Languages

Used for creating and editing pages on the web. Can do anything from putting plain text on a webpage, to accessing and retrieving data from a database. Vary greatly in terms of power and complexity.

- **HTML**
  Hyper Text Markup Language. The core language of the world wide web that is used to define the structure and layout of web pages by using various tags and attributes
- **XML**
  Extensible Markup Language. A language developed by the W3C which works like HTML, but unlike HTML, allows for custom tags that are defined by programmers. XML allows for the transmission of data between applications and organizations through the use of its custom tags.
- **Javascript**
  A language developed by Netscape used to provide dynamic and interactive content on webpages. With Javascript it is possible to communicate with HTML, create animations, create calculators, validate forms, and more.
- **VBScript**
  Visual Basic Scripting Edition. A language developed by Microsoft that works only in Microsoft's Internet Explorer web browser and web browsers based on the Internet Explorer engine such as FlashPeak's Slim Browser. VBScript Can be used to print dates, make calculations, interact with the user, and more.
- **PHP**
  Hypertext Preprocessor. A powerful language used for many tasks such as data encryption, database access, and form validation. PHP was originally created in 1994 By RasmusLerdorf.

HTML

- HTML is a **markup** language for **describing** web documents (web pages).HTML stands for **H**yper **T**ext **M**arkup **L**anguage. A markup language is a set of **markup tags.**
- HTML documents are described by **HTML tags.** Each HTML tag **describes** different document content.
- HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page.
- The markup tells the Web browser how to display a Web page's words and images for the user.
- Each individual markup code is referred to as an element (but many people also refer to it as a tag). Some elements come in pairs that indicate when some display effect is to begin and when it is to end. HTML tags are keywords surrounded by angle brackets like <html> .HTML tags normally come in pairs like <b> and </b>.The first tag in a pair is the start tag (opening tags), the second tag is the end tag(closing tags)

Where to Write HTML Code

WHERE WE WRITE CODE :
- 1.Text Editor
- 1.Wordpad (In Windows OS)
- 2.Gedit Text Editor (Ubundu in LINUX)
- 2.FrontPage or Dreamweaver

WHERE WE EXECUTE :
- 1.Double Click that HTML File. (or)
- 2.Right click – Open With Internet Explorer

Ex:

```
<html>

   <body>

         <h1>First Planet</h1>

         <h6>First Planet</h6>

   </body>

</html>
```

Basic Tags in HTML

Heading

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
```

Paragraphs

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Links

```
<a href="http://www.kristujayanti.edu.org">This is a link</a>
```

Images

```
<imgsrc="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

Attributes

```
<htmllang="en-US">
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Advantages of HTML:

1. First advantage it is widely used.
2. Every browser supports HTML language.
3. Easy to learn and use.
4. It is by default in every windows so you don't need to purchase extra software.

Disadvantages of HTML:

1. It can create only static and plain pages so if we need dynamic pages then HTML is not useful.
2. Need to write lot of code for making simple webpage.
3. Security features are not good in HTML.
4. If we need to write long code for making a webpage then it produces some complexity.

XML

XML (Extensible Markup Language) is a general-purpose specification for creating custom markup languages. The term extensible is used to indicate that a markup-language designer has significant freedom in the choice of markup elements

Advantages of XML

- It is a simultaneously human- and machine-readable format.
- It supports Unicode, allowing almost any information in any written human language to be communicated.
- It can represent the most general computer science data structures: records, lists and trees.
- Its self-documenting format describes structure and field names as well as specific values.
- XML is heavily used as a format for document storage and processing, both online and offline.
- It is based on international standards.
- It is platform-independent, thus relatively immune to changes in technology.

Disadvantages of XML

- XML syntax is redundant or large relative to binary representations of similar data.
- The redundancy may affect application efficiency through higher storage, transmission and processing costs.
- XML syntax is too verbose relative to other alternative 'text-based' data transmission formats.
- No intrinsic data type support: XML provides no specific notion of "integer", "string", "boolean", "date", and so on.

Scripting languages:

Scripting languages, which can be embedded within HTML, commonly are used to add functionality to a Web page, such as different menu styles or graphic displays or to serve dynamic advertisements. These types of languages are client-side scripting languages, affecting the data that the end user sees in a browser window. Other scripting languages are server-side scripting languages that manipulate the data, usually in a database, on the server.

Scripting languages came about largely because of the development of the Internet as a communications tool. JavaScript, ASP, JSP, PHP, Perl, Tcl and Python are examples of scripting languages.

Advantages

- Easy to learn and use
- Minimum programming knowledge or experience required
- Allow complex tasks to be performed in relatively few steps
- Allow simple creation and editing in a variety of text editors
- Allow the addition of dynamic and interactive activities to web pages
- Editing and running code is fast.

Disadvantages

- Executable code can inadvertently be downloaded from a remote server to a web browser's machine
- Programs can be downloaded without valid authenticity.
- The user is probably unaware of anything devious occurring.

Difference between HTML and XML

1. HTML was designed to display data with focus on how data looks while XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.

2. HTML is a markup language itself while XML provides a framework for defining markup languages.

3. HTML is a presentation language while XML is neither a programming language nor a presentation language.

4. HTML is case insensitive while XML is case sensitive.

5. HTML is used for designing a web-page to be rendered on the client side while XML is used basically to transport data between the application and the database.

6. HTML has its own predefined tags while what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document.

7. HTML is not strict if the user does not use the closing tags but XML makes it mandatory for the user the close each tag that has been used.

8. HTML does not preserve white space while XML does.

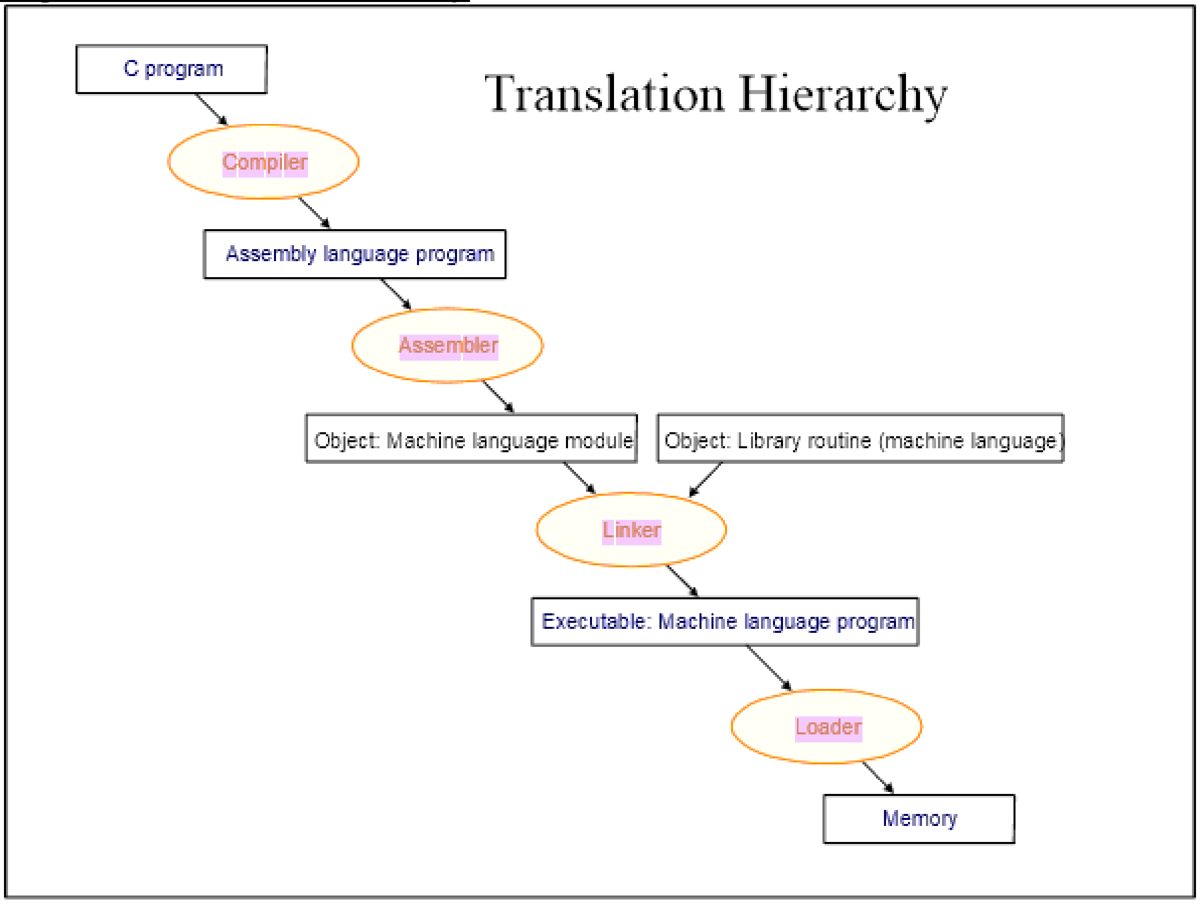
9. HTML is about displaying data, hence static but XML is about carrying information, hence dynamic.

Program Translation

A translator is a computer program that performs the translation of a program written in a given programming language into a functionally equivalent program in a different computer language, without losing the functional or logical structure of the original code.

These include translations between high-level and human-readable computer languages such as C++, Java and COBOL, intermediate-level languages such as Java bytecode, low-level languages such as the assembly language and machine code.

Program translation hierarchy


Translation Hierarchy
C program
Compiler
Assembly language program
Assembler
Object: Machine language module
Object: Library routine (machine language)
Linker
Executable: Machine language program
Loader
Memory

Translation Hierarchy

- **Compiler**
- **Assembler**
- **Loader**
- **Linker**

Compiler
A **compiler** is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language, often having a binary form known as object code).(i.e)Translates high level language program into assembly language.

Assembler

- Converts assembly language programs into Object files

- Object files contain a combination of machine instructions, data, and information needed to place instructions properly in memory

• Typically, assemblers make two passes over the assembly file

– **First pass:** reads each line and records *labels* in a *symbol table*
– **Second pass:** use info in symbol table to produce actual Machine code for each line.

Object file format

Object file header	Text segment	Data segment	Relocation information	Symbol table	Debugging information

- Object file header describes the size and position of the other pieces of the file
- Text segment contains the machine instructions
- Data segment contains binary representation of data in assembly file
- Relocation info identifies instructions and data that depend on absolute addresses
- Symbol table associates addresses with external labels and lists unresolved references
- Debugging info

Linker

A **linker** or link editor is a computer program that takes one or more objects generated by a compiler and combines them into a single executable program.

• Three tasks of Linker
– Searches the program to find library routines used by program, e.g. printf(), math routines,…
– Determines the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references
– Resolves references among files.

Loader

A **loader** is the part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

• Steps of a Loader
– Read executable file's header to determine the size of text and data segments
– Create a new address space for the program
– Copies instructions and data into address space

– Copies arguments passed to the program on the stack
– Initializes the machine registers including the stack ptr
– Jumps to a startup routine that copies the program's arguments from the stack to registers and calls the program's main routine.

Difference Between Complier And Interpreter

No	Compiler	Interpreter
1	Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input.
2	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement** : **More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
5	Program need not be **compiled** every time	Every time higher level program is converted into lower level program
6	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
7	**Example** : C Compiler	**Example** : BASIC

Characteristics of a Programming Language

- **Readability:** A good high-level language will allow programs to be written in some ways that resemble a quite-English description of the underlying algorithms. If care is taken, the coding may be done in a way that is essentially self-documenting.
- **Portability:** High-level languages, being essentially machine independent, should be able to develop portable software.
- **Generality:** Most high-level languages allow the writing of a wide variety of programs, thus relieving the programmer of the need to become expert in many diverse languages.
- **Brevity:** Language should have the ability to implement the algorithm with less amount of code. Programs expressed in high-level languages are often considerably shorter than their low-level equivalents.
- **Error checking:** Being human, a programmer is likely to make many mistakes in the development of a computer program. Many high-level languages enforce a great deal of error checking both at compile-time and at run-time.
- **Cost:** The ultimate cost of a programming language is a function of many of its characteristics.
- **Familiar notation:** A language should have familiar notation, so it can be understood by most of the programmers.
- **Quick translation:** It should admit quick translation.
- **Efficiency:** It should permit the generation of efficient object code.

- **Modularity:** It is desirable that programs can be developed in the language as a collection of separately compiled modules, with appropriate mechanisms for ensuring self-consistency between these modules.
- **Widely available:** Language should be widely available and it should be possible to provide translators for all the major machines and for all the major operating systems.

Static and Dynamic Web Pages

- "Static" means unchanged or constant, while "dynamic" means changing or lively.
- Static Web pages contain the same prebuilt content each time the page is loaded.
- A static website contains Web pages with fixed content. Each page is coded in HTML and displays the same information to every visitor
- Dynamic Web pages can be generated on-the-fly. Standard HTML pages are static Web pages
- Dynamic content is built and personalized to an individual user based on his or her relevant historical information.