Coding

- The coding is the process of transforming the design of a system into a computer language format.
- This coding phase of software development is concerned with software translating design specification into the source code.
- It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified.
- Coding is done by the coder or programmers who are independent people than the designer.
- The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a later stage.
- The cost of testing and maintenance can be significantly reduced with efficient coding.
- **Goals of Coding:**
  - **To translate the design of system into a computer language format**
  - **To reduce the cost of later phases**
  - **Making the program more readable**

Programming standards and procedures

- General coding standards refers to how the developer writes code.
- Different modules specified in the design document are coded in the Coding phase according to the module specification.
- The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test this code.
- Good software development organizations want their programmers to maintain to some well-defined and standard style of coding called coding standards.
- They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop.
- It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.
- **Purpose of Having Coding Standards:**
  - A coding standard gives a uniform appearance to the codes written by different engineers.
  - It improves readability, and maintainability of the code and it reduces complexity also.
  - It helps in code reuse and helps to detect error easily.
  - It promotes sound programming practices and increases efficiency of the programmers.
- **Advantages of Coding Guidelines and standards:**
  - Coding guidelines increase the efficiency of the software and reduces the development time.
  - Coding guidelines help in detecting errors in the early phases, so it helps to reduce the extra cost incurred by the software project.
  - If coding guidelines are maintained properly, then the software code increases readability and understandability thus it reduces the complexity of the code.
  - It reduces the hidden cost for developing the software.

Programming/coding Standards

1. **Indentation**: Proper and consistent indentation is essential in producing easy to read and maintainable programs.
   Indentation should be used to:
   - Emphasize the body of a control structure such as a loop or a select statement.
   - Emphasize the body of a conditional statement
   - Emphasize a new scope block
2. **Inline comments**: Inline comments analyze the functioning of the subroutine, or key aspects of the algorithm shall be frequently used.
3. **Rules for limiting the use of global**: These rules file what types of data can be declared global and what cannot.
4. **Structured Programming**: Structured (or Modular) Programming methods shall be used. "GOTO" statements shall not be used as they lead to "spaghetti" code, which is hard to read and maintain, except as outlined line in the FORTRAN Standards and Guidelines.
5. **Avoid using a coding style that is too difficult to understand:** Code should be easily understandable. The complex code makes maintenance and debugging difficult and expensive.
6. **Naming conventions for global variables, local variables, and constant identifiers**: A possible naming convention can be that global variable names always begin with a capital letter, local variable names are made of small letters, and constant names are always capital letters.
7. **Error return conventions and exception handling system**: Different functions in a program report the way error conditions are handled should be standard within an organization. For example, different tasks while encountering an error condition should either return a 0 or 1 consistently.
8. **Code should be well documented:**The code should be properly commented for understanding easily. Comments regarding the statements increase the understandability of the code.

- **Major aspects of Programming are :**
  - **Control structures, algorithms,data structures**
- **Control Structure:**
  - preserve the control structure suggested by architecture and design
  - should not jump wildly through the code
  - modularity
  - generality is a virtue (do not over do it)
  - dependence among components must be visible
- **Algorithms**
  - Choose wisely!
  - Algorithms effects execution time
  - “do not sacrifice clarity and correctness for speed”
- **Data Structure:**
  - Keeping the program simple == give the data structures simple
  - Using a data Structure to Determine a Program Structure4e.g. recursive data structure require recursive procedures

- **Other additional guidelines or aspects for good programming practices are :**
- **1. Line Length:** It is considered a good practice to keep the length of source code lines at or below 80 characters. Lines longer than this may not be visible properly on some terminals and tools. Some printers will truncate lines longer than 80 columns.
- **2. Spacing:** The appropriate use of spaces within a line of code can improve readability.
- **3. The code should be well-documented:** As a rule of thumb, there must be at least one comment line on the average for every three-source line.
- **4. The length of any function should not exceed 10 source lines:** A very lengthy function is generally very difficult to understand as it possibly carries out many various functions. For the same reason, lengthy functions are possible to have a disproportionately larger number of bugs.
- **5. Do not use goto statements:** Use of goto statements makes a program unstructured and very tough to understand.
- **6. Inline Comments:** Inline comments promote readability.
- **7. Error Messages:** Error handling is an essential aspect of computer programming. This does not only include adding the necessary logic to test for and handle errors but also involves making error messages meaningful.

Selecting Languages and Tools

- **How to choose the best programming language for a software development project?**

1. Types of Application

- Web applications, mobile applications, embedded firmware, etc.
- the type of application you're developing heavily influences what languages are available for us to choose from.
- General programming languages like Java, JavaScript, Python and C# can build different types of applications on different platforms.
- There are also cases where specific languages work better. For example, building a native Android app requires knowledge of Java, building a native iOS app requires a Swift or Objective-C skill set, whereas knowledge of C or C++ is critical if you're working with embedded firmware.

2.Complexity of the application

- The size and complexity of a project play an important role in deciding which stack or technology we want to use, which in turn is a deciding factor on what programming language to use.

3.**Company culture**

- Every company has preferences regarding open source versus proprietary software and internal versus external technical support.

- **4. Time to market**
  - Exploring new languages and technologies is better saved for projects with longer timelines.
  - impacts time to market is the physical environment you're currently operating in or what you may aspire to be in.
- **5.Maintainability**
  - In order to engage with our clients long-term, we may need to transition a codebase to another team or for the long term utilization and change adoption purpose we need maintainability .
- **6.Scalability and performance**
  - Scalability is more about a technology stack, which in turn determines the programming languages rather than the programming language itself.
- 7. **Security**
  - Every application has different requirements in terms of security.

Approaches to program design and development

- Procedural Programming
  - An approach to program design in which a program is separated into small modules that are called by the main program or another module when needed
    - Procedure call—locating specific tasks in procedures (modules or subprograms) that are called by the main program when needed
    - Allows each procedure to be performed as many times as needed; multiple copies of code not needed
    - Prior to procedural programming, programs were one large set of instructions (used GOTO statements)
- Structured Programming
  - Goes even further, breaking the program into small modules (Top-down design)

- Object-Oriented Programming (OOP)
  - Programs consist of a collection of objects that contain data and methods to be used with that data
    - Class
      – Group of objects that share some common properties
      - Instance
        - An individual object in a class
        - Inherits the attributes and methods of the class
    - Attributes
      - Data that describes the object
      - Can be in a variety of formats
    - Methods
      - Perform actions on an object
      - Can be used with different types of objects
  - Objects can be accessed by multiple programs
    - Class libraries

- Aspect-Oriented Programming (AOP)
  - Separates functions so program components can be developed and modified individually from one another
  - The components can be easily reused with separate nonrelated objects
- Adaptive Software Development
  - Designed to make program development faster and more efficient and focuses on adapting the program as it is being written
  - Features iterative and/or incremental development
- Agile Software Development
  - Goal is to create software quickly
  - Focuses on building small functional program pieces as the project progresses
  - Emphasizes teams of people working closely together (programmers, managers, business experts, customers, and so forth)
  - Some mobile developers are using continuous mobile innovation

Tools for facilating program development

- Tools include:
  - Requirements management
    - – Keeping track of and managing the program requirements as they are defined and then modified
  - Configuration management
    - – Keeping track of the progress of a program development project
  - Issue tracking
    - – Recording issues such as bugs or other problems that arise during development or after the system is in place
- Application Generators
  - Software program that helps programmers develop software
    – Macros
    - • Sequence of saved actions that can be replayed when needed
      • Programmers write them in a macro programminglanguage such as Visual Basic for Applications
  - – Report Generators and User Interface (UI) Builders
    - Report generator
      – Tool that prepares reports to be used with a software program quickly and easily
    - User interface (UI) builders
      - – Create the menus, forms, and input screens used with a program or database
    - • Integrated development environment (IDE)
      - – A set of programming tools for writing software applications

- Device Development Tools
  - – Assist with developing embedded software to be used on devices, such as cars, ATM machines, and consumer devices
- Integrated Development Environments (IDEs)
  - – Collection of tools used with a particular programming language to develop and test software
- Software Development Kits (SDKs)
  - – Programming package designed for a particular platform
  - – Enables programmers to develop applications for that platform more quickly and easily
- Application Program Interfaces (APIs)
  - – Help applications interface with a particular operating system
  - – Often used in conjunction with Web sites
  - – Google's Maps API and Google's OpenSocial API allow developers to add Google Maps or social networking applications easily to Web sites, respectively
- Mobile App Builders
  - – Many tools are available to help develop mobile apps and deploy them on various platforms
    - – One example is appsbar
    - – After the app is created, appsbar tests it and then submits it to major app markets for publication

IDE's

- Integrated development environments (IDE) are applications that facilitates the development of other applications.
- Designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools a developer needs, including:
  - **Code editor:** Designed for writing and editing source code, these editors are distinguished from text editors because work to either simplify or enhance the process of writing and editing of code for developers
  - **Compiler:** Compilers transform source code that is written in a human readable/writable language in a form that computers can execute.
  - **Debugger:** Debuggers are used during testing and can help developers debug their application programs.
  - **Build automation tools:** These can help automate developer tasks that are more common to save time.
  - **Other ides** are:
    - **Class browser:** Used to study and reference properties of an object-oriented class hierarchy.
    - **Object browser:** Used to inspect objects instantiated in a running application program.
    - **Class hierarchy diagram:** Allows developers to visualize the structure of object-oriented programming code.

Types of Programming languages

- Programming languages are used to control the performance of the computer or machine.
- Programming languages are the formal language, with a set of instructions which provides the desired output.
- For implementing various algorithms in our machines we started using the Programming language.
- A set of specific instructions are used in programmable machines, rather than general programming languages.
- Computer programming languages are used to communicate with a computer.
- Each and every Programming language are based on certain syntactic and semantic rules.
- Types of Programming Languages
  - **Low Level Languages:**The programming languages that are very close to machine code (0s and 1s) are called low-level programming languages.
    - Machine Languages
    - Assembly Languages
  - **High Level Languages:**The programming languages that are close to human languages (example like English languages) are called the high-level languages.
    - Procedural languages
    - Non procedural languages
    - Object oriented programming languages

- **Machine Languages**
  - The instructions in binary form, which can be directly understood by the computer (CPU) without translating them, is called a machine language or machine code.
  - Machine language is also known as first generation of programming language.
  - Machine language is the fundamental language of the computer and the program instructions in this language is in the binary form (that is 0's and 1's).
  - This language is different for different computers.
  - It is not easy to learn the machine language.
- **Advantage of Machine Language**
  - The only advantage of machine language is that the program of machine language runs very fast because no translation program is required for the CPU.
- **Disadvantage of Machine Language**
  - Machine Dependent: differs from one computer to another
  - Difficult to Modify :Checking machine instructions to locate errors is very difficult and time consuming.
  - Difficult to Program

- **Assembly Language**
  - Instructions written in this language are close to machine language.
  - Assembly language is also known as second generation of programming language.
  - With assembly language, a programmer writes instructions using symbolic instruction code instead of binary codes.
  - Symbolic codes are meaningful abbreviations such as SUB is used for substation operation, MUL for multiply operation and so on. Therefore this language is also called the low-level symbolic language.
  - The set of program instructions written in assembly language are also called as mnemonic code.
  - Assembly language provides facilities for controlling the hardware.
- **Advantages**
  - Easy to understand and use
  - Easier to locate and correct errors
  - Easy to modify
- **Disadvantages**
  - Machine dependent
  - Knowledge of hardware required
  - Machine level coding

High Level Languages

- The programming languages that are close to human languages (example like English languages) are called the high-level languages.
- The examples of high-level languages are: Fortran,COBOL,Basic,Pascal, C,C++,Java
- The high level languages are similar to English language.
- The program instructions are written using English words, for example print, input etc.
- But each high level language has its own rule and grammar for writing program instructions. These rules are called syntax of the language.
- The program written in high level language must be translated to machine code before to run it.
- Each high level language has its own translator program.
- The high level programming languages are further divided into:
  - Procedural languages
  - Non procedural languages
  - Object oriented programming languages

- **Procedural Languages:**
  - Procedural languages are also known as third generation languages (3GLs).
  - In a procedural language, a program is designed using procedures.
  - A procedure is a sequence of instructions having a unique name.
  - The instructions of the procedure are executed with the reference of its name.
  - In procedural programming languages, the program instructions are written in a sequence or in a specific order in which they must be executed to solve a specific problem. It means that the order of program instructions is very important.
  - Example of Procedural Languages are Fortran,Cobal,Pascal,Ada,C etc.
- **Non Procedural Languages**
  - Non procedural programming languages are also known as fourth generation languages(4GLs).
  - In non procedural programming languages, the order of program instructions is not important. The importance is given only to, what is to be done.
  - With a non procedural language, the user/programmer writes English like instructions to retrieve data from databases.
  - These languages are easier to use than procedural languages.
  - These languages provide the user-friendly program development tools to write instructions.
  - The programmers have not to spend much time for coding the program.
  - The most important non procedural languages and tools are SQL(Structured Query Languages, RPG(report Program Generator)

- **Object Oriented Programming Languages**
  - The object oriented programming concept was introduced in the late 1960s, but now it has become the most popular approach to develop software.
  - In object oriented programming, the software is developed by using a set of interfacing object.
  - An object is a component of program that has a set of modules and data structure.
  - The modules are also called methods and are used to access the data from the object.
  - The modern technique to design the program is object oriented approach.
  - It is a very easy approach, in which program designed by using objects. Once an object for any program designed, it can be re-used in any other program.
  - Now-a-days, most popular and commonly used object oriented programming (OOPs) languages are C++ and Java.
  - OOPS provides various concepts like inheritance,polymorphism,data abstraction,class,object etc.

- ## Advantages of High Level Languages
  - **Easy to learn** - the high level languages are very easy to learn than low level languages. The statements written for the program are similar to English-like statements.
  - **Easy to understand** - the program written in high level language by one programmer can easily be understood by another because the program instructions are similar to the English language.
  - **Easy to write program** - in high level language, a new program can easily be written in a very short time. The larger and complicated software can be developed in few days or months.
  - **Easy to detect and remove errors** - the errors in a program can be easily detected and removed. mostly the errors are occurred during the compilation of new program.
  - **Built-in library functions** - Each high level language provides a large number of built-in functions or procedures that can be used to perform specific task during designing of new programs. In this way, a large amount of time of programmer is saved.
  - **Machine Independence** - program written in high level language is machine independent. It means that a program written in one type of computer can be executed on another type of computer.
- ## Disadvantages of High Level Languages
  - **Low efficiency** - a program written in high level languages has lower efficiency than one written in a machine/assembly language to do the same job. That is, program written in high level languages result in multiple machine language instruction that may not be optimize, taking more time to execute and requiring more memory space.
  - **Less flexibility** - high level languages are less flexible than assembly languages because they do not normally have instructions or mechanism to control a computer's CPU, memory and register.