**THE PARADIGMS OF PROGRAMMING**

1. Procedural programming
2. Declarative programming
3. Structural programming
4. Prototype-based programming
5. Modular programming

**PROCEDURAL PROGRAMMING:** This is a programming paradigm that is strictly based on the concept of procedure calls, where programs are composed of procedures or functions that perform specific tasks. It is based on a step-by-step approach where a program is broken down into smaller manageable codes. Many programming languages, such as C, Java and Python support procedural programming.

**DECLARATIVE PROGRAMMING:** This is a high level programming concept, which adopts a method that abstract the control flow for logic required for software to perform an action. It is a paradigm that focuses on defining what a program should accomplish, rather than how it should accomplish. Examples of declarative programming languages include SQL, and HTML.

**STRUCTURED PROGRAMMING**: Structured programming is a programming paradigm aimed at improving clarity, quality and development time of a computer program by making extensive use of the structured control flow construct of selection (if, then, else) and repetition (while and for), blocked structures and subroutines in contrast to using simple tests and jumps. It is a programming paradigm that focuses on organizing code in a logical and structured way, making it easier to:

1. Read and understand
2. Maintain and modify
3. Debug

It is widely used in various programming languages including C, Pascal and Java.

**PROTOTYPE-BASED PROGRAMMING:** It is a style of object-oriented programming in which behavior reuse (known as inheritance) is performed via a process of reusing existing objects that serve as prototypes. It is simply a paradigm that focuses on creating objects that can serve as templates or prototype for other objects. Its key concept includes:

1. Creating prototype
2. Cloning Prototype
3. Customizing cloned prototypes

**MODULAR PROGRAMMING**: It is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains every thing necessary to execute only aspect of the desired functionality. Its key principles included:

1. Modularity: breaking down a large system into smaller, independent modules
2. Separation of concerns: each module focuses on a specific task or functionality
3. Reusability: Modules can be reused in other parts of the system or even in different projects.