### 1. Data Structures

A **data structure** is a collection of data organized for efficient storage and retrieval. Examples include:

* **Arrays**
* **Linked Lists**
* **Stacks**
* **Queues**

These structures facilitate efficient data management and manipulation.

### 3. Key Aspects of Algorithms

1. **Method** is used in computer science to describe a problem solving for implementation as a program.
2. **Algorithm** are use as data in the computations. These are the processes of creating Object which is called Data Structure and it is a central objects of study in computer science
3. **Computers** are designed to solve problems accurately; they must also use time or space as efficiently as possible.
4. In solving major problems, one must be careful about the effectiveness of:

**process of solving =>** to be developed computer program which must be clearly understood and defined.

**problem to be solved =>** managing its complexity and break the elements into smaller values ​​that can be easily implemented.

1. Computer systems is a contributor to:

**program =>** there is a change in methodology computing environments (HW & SW) to achieve good success of the problem. (make our solutions more portable and longer lasting)

1. Best algorithm for any part of the work, it can be given as complicated process, or is sophisticated mathematical analysis. => this type of study is called analysis of algorithms (analytical study on performance and experiences, to comparative performance of the methods)

### 4. Concept of Good Programming

**Good programming** entails creating programs that:

* Run correctly.
* Are efficient in time and space.
* Are easy to read and understand.
* Are easy to debug and modify.

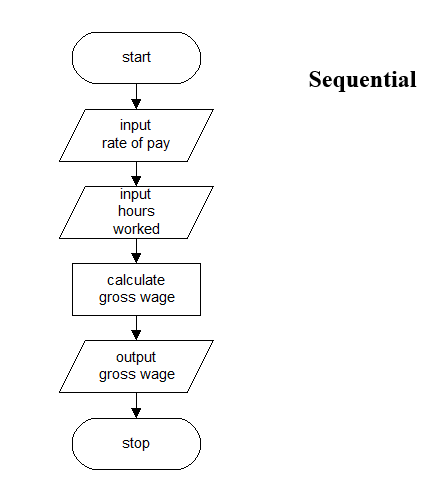
Utilizing appropriate data structures and algorithms enhances performance.

### 5. Steps to Write a Computer Program

1. **Problem Analysis:** involve studying the problem -> to understand the nature of the problem and determine how to solve it
2. **Designing and Testing an Algorithm:** an algorithm is a solution to a problem and normally consists of steps. Having designed a solution, the next step is to trace through the algorithm with test data into verify that the solution contains no logical errors. Logical errors are mistakes in the design of a program, such as a branch to a wrong statement or the use of wrong mathematical formula.
3. **Coding the Algorithm:** to use a suitable computer language to code the algorithm into a corresponding computer program. The operation defined in the flowchart should translate directly into instructions in a high level language.
4. **Testing the Code:** compiled program might list any errors -> known as syntax errors and are associated with the wrong construction of language statement. It is necessary to test the solution to the problem and verify the program does indeed function correctly
5. **Documentation:** it is used and produced during the other four activities and for this reason documentation can be regarded as an activity that occurs throughout the entire programming cycle. Documentation involves documenting the purpose of the program, the method of solution, the stages of testing that it has undergone, and other necessary facts.

### 6. What is a Flowchart?

A **flowchart** contains a description of how to solve a problem written in a style independent of any programming language compose of flowchart:



### 7. Execution Time of an Algorithm

Most algorithms have one main parameter (N) that affects running time and has characteristics of a polynomial degree (file sizes, number of characters in a text string). This leads to a dependence on mathematical formulas, which then determines the running times of the algorithms.

**1** Instruction is executed once or a few times => running time is constant.

**logN** Program slows down as N grows -> this program should solve a big problem.  
N=1000 => logN=3; N can double, but logN can double only if N increases to N².

**N** The running time of the program is linear. The characteristic of this type of algorithm is that it often processes N inputs or produces N outputs.

**NlogN** Algorithm solves problems by breaking them down into parts or subproblems (solution to it is independent). Then just put it back together. When N doubles, the running time has more value than N doubles.

**N2** The running time of the algorithm is quadratic. It can happen when it processes all pairs of data items (case of doubling -> nested loop). If N doubles, the running time increases up to four times.

**2N** An algorithm that has a running time that is exponential is used in cases where a solution is forced. If N doubles, the running time is squared.

In conclusion, the running time in the section of the program has the value of some constant multiplied by one of these terms plus some smaller terms (the values of the constant coefficient and the terms included are the results of the analysis on implementation details).

### 8. Big-Oh Notation

**Big-Oh notation** describes the upper limit of an algorithm's running time or space requirements as a function of input size, helping classify algorithms based on efficiency by focusing on the leading term and ignoring constant factors.

### 9. Importance of Arrays in Data Structures

**Arrays** are fundamental due to:

* Direct access to elements via indices for efficient retrieval.
* Contiguous memory allocation, enhancing performance through locality of reference.
* Simple implementation for various algorithms.

### 10. Stack and Its Evolution​

A **stack** is a **LIFO (Last In, First Out)** data structure, where the last element added is the first to be removed. Its evolution includes:

* Serving as an abstract data type in programming.
* Managing function calls and local variables in programming languages.
* Supporting algorithms like depth-first search.

### 12. FIFO vs. LIFO

* **FIFO (First In, First Out)**: The first element added is the first to be removed (like a queue).
* **LIFO (Last In, First Out)**: The last element added is the first to be removed (like a stack).

### 13. Converting Postfix and Infix

* **Postfix to Infix** for 598+467+\*: ((5 + (9 \* 8)) \* (4 \* 6) + 7)
* **Infix to Postfix** for 5(((9+8)(4\*6))+7): 598+46\*7+\*

### 14. Postfix Theory

To evaluate a postfix expression using a stack:

1. Push operands onto the stack.
2. For each operator, pop the required operands, perform the operation, and push the result back.
3. Continue until the expression is fully processed.

### 15. Analyzing Algorithm Statements

1. cout << L; - Outputs L.
2. area = I \* W; - Calculates area (2 fetch + 1 multiplication).
3. c = 5/9 \* (F - 32); - Temperature conversion calculation (3 fetch + 1 multiplication).
4. return i; - Returns the value of i (1 fetch).

### 16. Binary Tree vs. Graph

* **Binary Tree**: A tree structure where each node has at most two children, useful for hierarchical data.
* **Graph**: A collection of vertices connected by edges, allowing for complex relationships.

### 17. Types of Spanning Trees

1. **Minimum Spanning Tree (MST)**: Connects all vertices with the minimum edge weight.
2. **Maximum Spanning Tree**: Connects all vertices with the maximum edge weight.
3. **Unique Spanning Tree**: A single spanning tree exists for a given graph.
4. **Random Spanning Tree**: Randomly selects edges to connect all vertices.

### 18. Bubble Sort Algorithm

### 

### 19. Insertion Sort Algorithm

### 

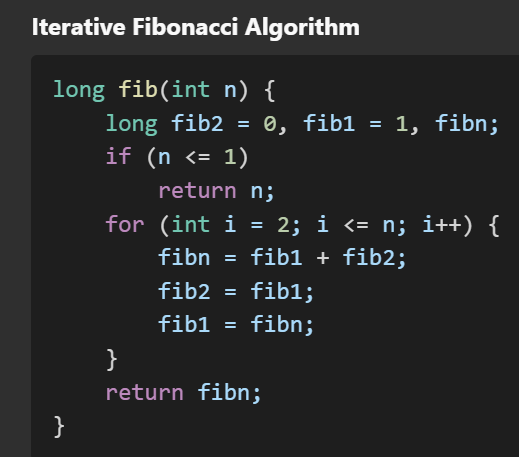
### 20. Selection Sort Algorithm

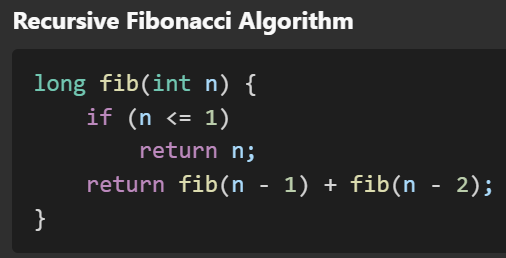
### 

### 22. Tower of Hanoi Recursive Algorithm

### 

### 23. Fibonacci: Recursive and Iterative Algorithms





### 24. Factorial: Recursive and Iterative Algorithms

### 

### 

### 25. Prime Number Algorithm

### 

### 26. Euclid's GCD Algorithm

### 