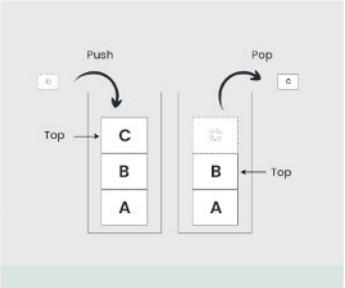
Stack

quarta-feira, 16 de outubro de 2024 13:48

Linear data structure that follows an order

↳ LIFO → Last in first Out

↳ FILO → First in Last Out



Key Operations on Stack Data Structures

- **Push**: Adds an element to the top of the stack.
- **Pop**: Removes the top element from the stack.
- **Peek**: Returns the top element without removing it.
- **IsEmpty**: Checks if the stack is empty.
- **IsFull**: Checks if the stack is full (in case of fixed-size arrays).

The functions associated with stack are:

empty() – Returns whether the stack is empty – Time Complexity : O(1)

size() – Returns the size of the stack – Time Complexity : O(1)

top() – Returns a reference to the top most element of the stack – Time Complexity : O(1)

push(g) – Adds the element 'g' at the top of the stack – Time Complexity : O(1)

pop() – Deletes the most recent entered element of the stack – Time Complexity : O(1)

Applications of Stack Data Structures

- Recursion
- Expression Evaluation and Parsing
- Depth-First Search (DFS)
- Undo/Redo Operations
- Browser History
- Function Calls

Advantages of Stacks:

- **Simplicity:** Stacks are a simple and easy-to-understand data structure, making them suitable for a wide range of applications.
- **Efficiency:** Push and pop operations on a stack can be performed in constant time **(O(1))**, providing efficient access to data.
- **Last-in, First-out (LIFO):** Stacks follow the LIFO principle, ensuring that the last element added to the stack is the first one removed. This behavior is useful in many scenarios, such as function calls and expression evaluation.
- **Limited memory usage:** Stacks only need to store the elements that have been pushed onto them, making them memory-efficient compared to other data structures.

Disadvantages of Stacks:

- **Limited access:** Elements in a stack can only be accessed from the top, making it difficult to retrieve or modify elements in the middle of the stack.
- **Potential for overflow:** If more elements are pushed onto a stack than it can hold, an overflow error will occur, resulting in a loss of data.
- **Not suitable for random access:** Stacks do not allow for random access to elements, making them unsuitable for applications where elements need to be accessed in a specific order.
- **Limited capacity:** Stacks have a fixed capacity, which can be a limitation if the number of elements that need to be stored is unknown or highly variable.