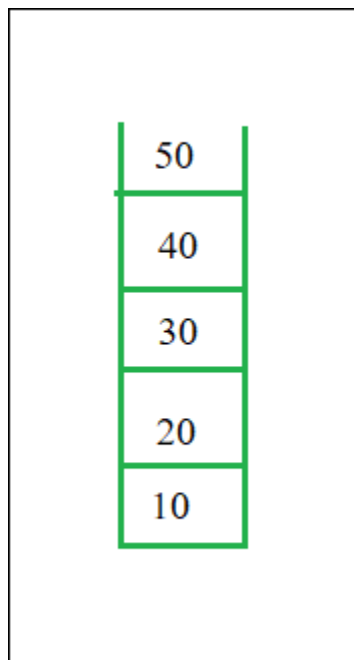


Stack ADT

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations. The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called “abstract” because it gives an implementation-independent view.



Stack ADT



- In Stack ADT Implementation instead of data being stored in each node, the pointer to data is stored.
- The program allocates memory for the *data* and *address* is passed to the stack ADT.
- The head node and the data nodes are encapsulated in the ADT. The calling function can only see the pointer to the stack.
- The stack head structure also contains a pointer to *top* and *count* of number of entries currently in stack.
- `push()` – Insert an element at one end of the stack called top.
- `pop()` – Remove and return the element at the top of the stack, if it is not empty.

- `peek()` – Return the element at the top of the stack without removing it, if the stack is not empty.
- `size()` – Return the number of elements in the stack.
- `isEmpty()` – Return true if the stack is empty, otherwise return false.
- `isFull()` – Return true if the stack is full, otherwise return false.

