Stack

Data Structures Made Easy

DUBLIN CITY UNIVERSITY

Contents

1. ***Definition***

A stack is a non-linear data structure that implements the addition of new elements and the deletion of existing elements within the stack. Deletion of an element involves removing the elements from the top of the stack. A stack inherits the LIFO (Last In First Out) rule.

There are two types of stacks:

- **Bounded Stack**: A stack that has a maximum size implemented upon it. It contains a sequence of objects that increments and decrements at one end of the stack only.

- **Unbounded Stack:** Similar to a bounded stack, but there is no bound on the stack size.

2. ***Implementation***

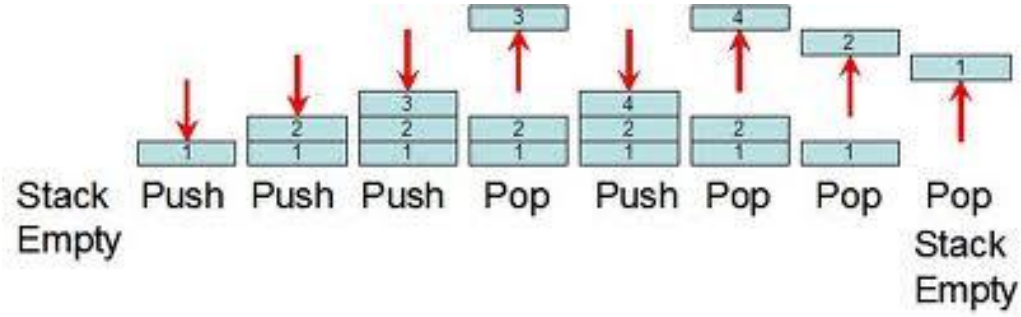
A stack works as follows:

1. Specify the type of data that is desired to be stored.

2. Push the elements onto the stack.

3. To retrieve the last inserted element in the stack, pop it off.

3. ***Example***



4. ***Functions***

The implementation of a stack involves the following functions

- **Stack<Object>()**
  This constructor creates an empty stack that holds the object type, which is specified upon the creation of the stack.

- **push(Object obj)**
  This function inserts / pushes the object onto the top of the stack. An object may be in the form of elements or data.

- **pop()**
  This function is used to remove the object from the top position of the stack.

5. ***Bounded Stack Pseudocode***

```
function push ( T t ){

        if size of array >= length of sequence_1
                sequence_2 = new sequence [ length of sequence_1 * 2 ]
                copy sequences
                sequence_1 = sequence_2

        sequence [ size ] = t
        increment size
}

function pop(){

        if size is 0
                return
        else
                temporary = sequence [ size ]
                decrement size
                return sequence [ size ]
}
```

6. ***Unbounded Stack Pseudocode***

```
class Node < T >{

        T item
        Node null

        constructor Node
}

function push ( T t ){

        head = new Node
        return
}

function pop(){

        if size is 0
                return

        T = head.item
        head = head.Node
        return
}
```

6. ***Complexity***

The only considerations to be made in relation to the complexity of a stack are the operations it performs. The operations of pushing and popping possess a time complexity of O(1).

7. ***Advantages of Stacks***

A stack is considered to be a data structure that is easy to implement. A strong advantage of using a stack is that it has a low requirement for computer hardware. The accessibility range is also taken into consideration, as anyone who has access can edit the program.

8. ***Disadvantages of Stacks***

A stack is considered to be of minimal use when compared to other optimal data structures. The inflexibility of a stack and the lack of scalability have deteriorated its use in various applications.

9. ***References***

Tech Terms 2013. *Stack* [Online]. Available from: http://www.techterms.com/definition/stack

Rose India 2007. *Implementing a Stack in Java* [Online]. Available from: http://www.roseindia.net/java/example/java/util/StackImplement.shtml