Stack

Stacks are a type of container adapters with LIFO(Last In First Out) type of working, where a new element is added at one end (top) and an element is removed from that end only. Stack uses an encapsulated object of either vector or deque (by default) or list (sequential container class) as its underlying container, providing a specific set of member functions to access its elements.

1. To use a stack, you must include it with stack header file.

```
#include <stack>
```

2. The syntax to define a stack is:

```
std::stack<dataType> stackName;
```

3. Accessing elements using square brackets is not possible.

Functions

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)
- **swap(g)** - Swaps the stack with another stack 'g'.

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<int> stack;
    stack.push(21);// The values pushed in the stack should be of the same da
    stack.push(22);
    stack.push(24);
```

```
    stack.push(25);
    int num=0;
    stack.push(num);
    stack.pop();
    stack.pop();
    stack.pop();
    while (!stack.empty()) {
        cout << stack.top() <<" ";
        stack.pop();
    }
}
```