Stack

Stacks are a type of container adaptors 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.

Syntax

template<class T, class Container = deque<T> > class stack;

This data structure works on the LIFO technique, where LIFO stands for Last In First Out. The element which was first inserted will be extracted at the end and so on. There is an element called as 'top' which is the element at the upper most position. All the insertion and deletion operations are made at the top element itself in the stack.

Stacks in the application areas are implied as the container adaptors.

The containers should have a support for the following list of operations:

empty

size

back

push_back

pop_back

Template Parameters

T: The argument specifies the type of the element which the container adaptor will be holding.

Container: The argument specifies an internal object of container where the elements of the stack are hold.

Member Types

Given below is a list of the stack member types with a short description of the same.

Member Types	Description
value_type	Element type is specified.
container_type	Underlying container type is specified.
size_type	It specifies the size range of the elements.

Functions

With the help of functions, an object or variable can be played with in the field of programming. Stacks provide a large number of functions that can be used or embedded in the programs. A list of the same is given below:

Function	Description
(constructor)	The function is used for the construction of a stack container.
empty	The function is used to test for the emptiness of a stack. If the stack is empty the function returns true else false.
size	The function returns the size of the stack container, which is a measure of the number of elements stored in the stack.
top	The function is used to access the top element of the stack. The element plays a very important role as all the insertion and deletion operations are performed at the top element.
push	The function is used for the insertion of a new element at the top of the stack.
pop	The function is used for the deletion of element, the element in the stack is deleted from the top.
emplace	The function is used for insertion of new elements in the stack above the current top element.
swap	The function is used for interchanging the contents of two containers in reference.
relational operators	The non member function specifies the relational operators that are needed for the stacks.
uses allocator<stack>	As the name suggests the non member function uses the allocator for the stacks.

Example: A simple program to show the use of basic stack functions.

```
#include <iostream>
#include <stack>
using namespace std;
void newstack(stack <int> ss)
{
   stack <int> sg = ss;
   while (!sg.empty())
   {
      cout << '\t' << sg.top();
      sg.pop();
```

```
    }
    cout << '\n';
}
int main ()
{
    stack <int> newst;
    newst.push(55);
    newst.push(44);
    newst.push(33);
    newst.push(22);
    newst.push(11);

    cout << "The stack newst is : ";
    newstack(newst);
    cout << "\n newst.size() : " << newst.size();
    cout << "\n newst.top() : " << newst.top();
    cout << "\n newst.pop() : ";
    newst.pop();
    newstack(newst);
    return 0;
}
```

Output:

```
The stack newst is :        11      22      33      44      55
 newst.size() : 5
 newst.top() : 11
 newst.pop() :       22      33      44      55
```