Stack STL

Introduction

Stack is a linear data structure that follows the LIFO (Last In First Out) principle. In C++ programming language, the Standard Template Library (STL) provides the implementation of stack data structure through the stack container class. The stack container class in STL provides various member functions to perform different operations on the stack.

Creating a Stack:

To use a stack in C++, we need to include the <stack> header file. The syntax to create a stack is as follows:

```
stack<datatype> stackName;
```

Here, "datatype" is the data type of elements that the stack will hold and "stackName" is the name of the stack. For example, to create a stack of integers, we can write:

```
stack<int> myStack;
```

Stack Member Functions:

The stack container class provides the following member functions to perform different operations on the stack:

1. push() - This function is used to add an element to the top of the stack. It takes an element of the same data type as the stack as an argument.
2. pop() - This function is used to remove the top element from the stack.
3. top() - This function is used to access the top element of the stack without removing it.
4. empty() - This function returns a boolean value indicating whether the stack is empty or not.
5. size() - This function returns the number of elements in the stack.

Example:

Let's consider the following example to understand how to use stack in C++ STL:

```
#include <iostream>
#include <stack>
using namespace std;

int main() {
   stack<int> myStack; // create an empty stack of integers

   // push elements into the stack
   myStack.push(1);
   myStack.push(2);
   myStack.push(3);

   // print the top element of the stack
   cout << "Top element of the stack: " << myStack.top() << endl;

   // remove the top element from the stack
   myStack.pop();

   // print the size of the stack
   cout << "Size of the stack: " << myStack.size() << endl;

   // check if the stack is empty
   if(myStack.empty())
      cout << "Stack is empty" << endl;
   else
      cout << "Stack is not empty" << endl;

   return 0;
}
```

Output:

```
Top element of the stack: 3
Size of the stack: 2
Stack is not empty
```

Conclusion:

In conclusion, the stack container class in C++ STL provides an easy and efficient way to implement a stack data structure. It provides various member functions to add, remove, and access elements from the stack. The STL stack is a useful tool for various applications and can greatly simplify the implementation of certain algorithms.