Using Stack in C++ Standard Template Library (STL)

1. Introduction

Stack is a container in the C++ STL that operates on a **LIFO (Last In, First Out)** principle. This means the last item added to the stack (the top item) is the first one to be removed. It is like a stack of plates: we place a new plate on top, and we must take the top one off before reaching the others.

Unlike vectors or arrays, stacks do not support random access (we cannot use [] to grab an item in the middle). We only have access to the "top" item.

2. Setup and initialization

To use a stack, we must include the `<stack>` header first.

```
#include <stack>
```

To declare a stack, we use this syntax:

```
std::stack<type> <stack name>;
```

Ex:

```
std::stack<int> s;
```

3. Core operations.

Stack supports a set of operations focused mainly on processing the top item of the current container.

3.1/ Adding items (push)

The `push(item)` adds a new item to the top of the stack (O(1) time complexity).

Ex:

```
s = [25, 12, 50, 43]
s.push(6)
s = [25, 12, 50, 43, 6]
```

3.2/ Accessing the Top (top)

`top()` returns a reference to the top item (O(1) time complexity). It does not remove the item itself, it returns a non-defined behavior if the current stack is empty.

Ex:

```
s = [25, 12, 50, 43]
s.top() = 43
```

3.3/ Removing items (pop)

`pop()` removes the top item (newest added item) of the current stack (O(1) time complexity). It returns **void** (nothing) so if we want to know the removed item, we need to call `top()` right before `pop()`.

Ex:

```
s = [25, 12, 50, 43]
s.pop()
s = [25, 12, 50]
```

4. Status operations

We often need to check the status of the stack before performing an operation.

`empty()` Returns **true** if the stack has no items, false otherwise.

`size()` Returns the number of items currently in the stack.

5. Conclusion

The `std::stack` is commonly used for:

- Function call management (recursion).
- Undo mechanisms in software (reversing actions).
- Parsing expressions (like matching parentheses).

It restricts access to strictly the "top" item, ensuring data is processed in the exact reverse order it was received.