C++ 标准库 <stack>

在 C++ 中，标准库提供了多种容器和算法来帮助开发者更高效地编写程序。

<stack> 是 C++ 标准模板库（STL）的一部分，它实现了一个后进先出（LIFO，Last In First Out）的数据结构。这种数据结构非常适合于需要"最后添加的元素最先被移除"的场景。

<stack> 容器适配器提供了一个栈的接口，它基于其他容器（如 deque 或 vector）来实现。栈的元素是线性排列的，但只允许在一端（栈顶）进行添加和移除操作。

基本操作

- push(): 在栈顶添加一个元素。
- pop(): 移除栈顶元素。
- top(): 返回栈顶元素的引用，但不移除它。
- empty(): 检查栈是否为空。
- size(): 返回栈中元素的数量。

语法

以下是使用 <stack> 的基本语法：

```
#include <iostream>
#include <stack>

int main() {
    std::stack<int> s;

    // 向栈中添加元素
    s.push(1);
    s.push(2);
    s.push(3);

    // 访问栈顶元素
    std::cout << "Top element is: " << s.top() << std::endl;

    // 移除栈顶元素
    s.pop();
    std::cout << "After popping, top element is: " << s.top() << std::endl;

    // 检查栈是否为空
    if (!s.empty()) {
        std::cout << "Stack is not empty." << std::endl;
    }

    // 打印栈的大小
    std::cout << "Size of stack: " << s.size() << std::endl;
```

```
    return 0;
}
```

实例

下面是一个使用 <stack> 的完整示例，包括输出结果：

实例

```
#include <iostream>
#include <stack>

int main() {
    std::stack<int> s;

    // 向栈中添加元素
    s.push(10);
    s.push(20);
    s.push(30);

    // 打印栈顶元素
    std::cout << "Top element is: " << s.top() << std::endl; // 输出: Top element is:
30

    // 移除栈顶元素
    s.pop();
    std::cout << "After popping, top element is: " << s.top() << std::endl; // 输出:
After popping, top element is: 20

    // 检查栈是否为空
    if (!s.empty()) {
        std::cout << "Stack is not empty." << std::endl; // 输出: Stack is not empty.
    }

    // 打印栈的大小
    std::cout << "Size of stack: " << s.size() << std::endl; // 输出: Size of stack:
2

    // 继续移除元素
    s.pop();
    s.pop();

    // 检查栈是否为空
    if (s.empty()) {
        std::cout << "Stack is empty." << std::endl; // 输出: Stack is empty.
    }

    return 0;
}
```

输出结果：

```
Top element is: 30
After popping, top element is: 20
```

```
Stack is not empty.
Size of stack: 2
Stack is empty.
```

注意事项

<stack> 不提供直接访问栈中元素的方法，只能通过 top() 访问栈顶元素。

尝试在空栈上调用 top() 或 pop() 将导致未定义行为。

<stack> 的底层容器可以是任何支持随机访问迭代器的序列容器，如 vector 或 deque。