ЛАБОРАТОРНА РОБОТА 2
Стек

ТЕОРЕТИЧНИЙ МАТЕРІАЛ

Стек - *динамічна* структура даних для тимчасового їх зберігання, яка має єдину точку доступу (вершину) і функціонує за принципом **FILO** (першим прийшов - останнім пішов).

Інтерфейс стека:

Методи зміни стану: **push, pop**

Методи перевірки стану: **isEmpty, peek, size**

Способи програмної реалізації стека:

Стек можна змоделювати класом, що інкапсулює пам'ять стека та вершину.

1. **Стек на основі неперервної пам'яті** використовує масив для зберігання даних. Індекс першої вільної комірки масиву визначає вершину стеку. Потрібно контролювати переповнення масиву і, за потреби, виділяти більший масив.
2. **Стек на основі зв'язної пам'яті** – це лінійний однозв'язний список, для якого дозволено тільки дві операції: 1) додавання ланки на початок списку; 2) вилучення першої ланки. Вершина стеку – голова списку.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Написати власну програмну реалізацію (клас) стека двома способами:

1) на основі масиву;
2) на основі зв'язного списку.

Продемонструвати роботу програми на прикладі перевірки введеної послідовності дужок на правильність[1]. Реалізувати цю ж перевірку, використовуючи бібліотеку <stack> (STL).

Примітка: Іноземним студентам дозволяється реалізувати стек тільки одним з двох згаданих способів і продемонструвати роботу на прикладі програми виведення послідовності чисел у зворотньому порядку.

---

[1] Приклади правильних послідовностей дужок: (), (())(), ()(), ((())).
Приклади неправильних послідовностей дужок:)(, ())((), (,)))), ((()).

Алгоритм: розглядаємо по черзі символи символи послідовності зліва направо; якщо символ — відкриваюча дужка, то додаємо її в стек, якщо закриваюча, то вилучаємо елемент зі стеку (цим елементом обов'язково має бути відкриваюча дужка. Послідовність правильна, якщо в кінці отримаємо порожній стек.