**Міністерство освіти і науки України**

**Івано-Франківський національний технічний університет нафти і газу**

**Кафедра ІПЗ**

**Лабораторна робота №3**

**Алгоритми для роботи зі стеками**

**Виконав:**

**ст.групи ІП-22-1**

**Чурчак В.В.**

**Перевірив:**

**Григорчук Л.І.**

**Івано-Франківськ**

**2023**

**Мета:** ознайомитися зі стеками реалізувати операції з ними відповідно до варіанту завдання.

**Завдання:**

1. Створити програму, що передбачає введення в стек цілих чисел з клавіатури. Вивести стек на екран (числа виведуться в оберненому порядку)

2. Створити та вивести 2 стеки - з парними та непарними числами серед введених з клавіатури

3. Створити програму яка перевіряє правильність послідовності круглих та квадратних дужок за час, що не перевищує константу множену на довжину цієї послідовності

**Короткі теоретичні відомості**

Стек - це структура даних, що працює за принципом "останній увійшов, першим вийшов" (Last In, First Out, LIFO). Елементи додаються та видаляються тільки з вершини стеку.

Операції зі стеками:

- Push: Додає елемент на вершину стеку.

- Pop: Видаляє елемент з вершини стеку.

- Peek (або Top): Повертає значення верхнього елемента без його видалення.

- isEmpty: Перевіряє, чи стек порожній.

Реалізації:

- Масиви: Стек можна реалізувати за допомогою масиву, де вершина стеку відповідає індексу останнього доданого елемента.

- Вказівники: Стек можна також реалізувати за допомогою вказівників, де кожен елемент містить вказівник на попередній елемент у стеці.

- Стандартні бібліотеки: Багато мов програмування мають вбудовані або стандартні бібліотечні класи для стеків. Наприклад, у мові C++ існує `std::stack` у бібліотеці STL.

Застосування стеків:

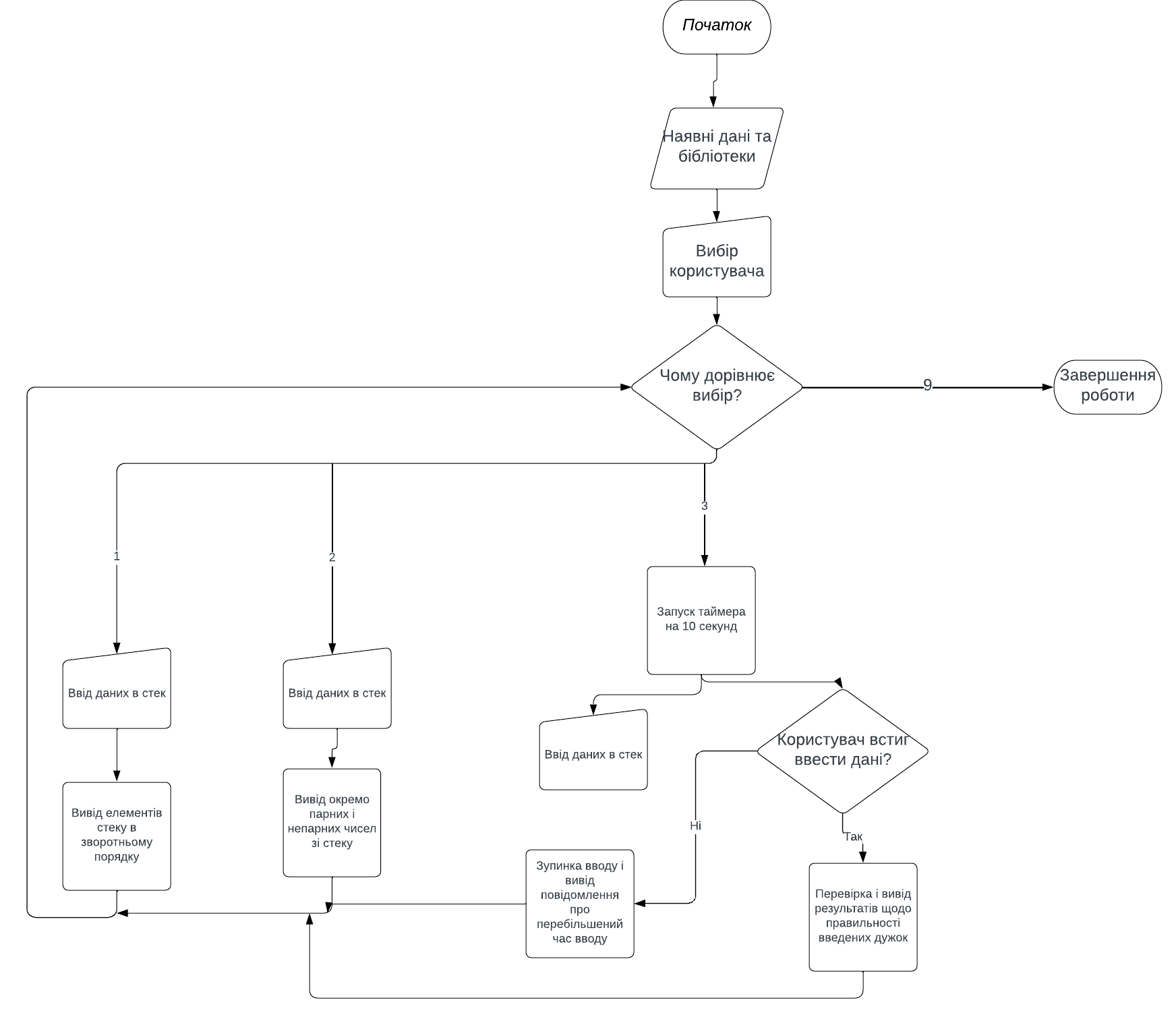
- Керування викликами функцій Стек використовується для зберігання інформації про виклик функцій та повернення з них.

- Обробка виразів: Стек використовується для перетворення і обчислення арифметичних виразів, в тому числі в інших форматах, таких як обернений польський запис (RPN).

- Управління відсотками: Стек може бути використаний для відстеження відкритих та закритих дужок для перевірки коректності виразів.

- Обробка рекурсії: Стек використовується для зберігання інформації про виклик рекурсивних функцій.

**Графічний алгоритм**



**Виконання**

#include <iostream>

#include <stack>

#include <Windows.h>

#include <ctime>

#include <future>

#include <chrono>

#include <thread>

using namespace std;

void firstQuest() {

// Створюємо стек для зберігання цілих чисел

std::stack<int> myStack;

// Заповнюємо стек числами, введеними з клавіатури

int n;

std::cout << "Введіть кількість елементів у стеку: ";

std::cin >> n;

std::cout << "Введіть цілі числа для стеку:\n";

for (int i = 0; i < n; ++i) {

int value;

std::cin >> value;

myStack.push(value);

}

// Виводимо стек на екран у зворотньому порядку

std::cout << "Елементи стеку у зворотньому порядку:\n";

while (!myStack.empty()) {

std::cout << myStack.top() << " ";

myStack.pop();

}

std::cout << "\n";

}

void secondQuest() {

// Створюємо стеки для парних і непарних чисел

std::stack<int> evenStack;

std::stack<int> oddStack;

// Заповнюємо стеки числами, введеними з клавіатури

int n1;

std::cout << "Введіть кількість елементів: ";

std::cin >> n1;

std::cout << "Введіть цілі числа:\n";

for (int i = 0; i < n1; ++i) {

int value;

std::cin >> value;

// Визначаємо, чи є число парним чи непарним

if (value % 2 == 0) {

evenStack.push(value); // Додаємо парне число до стеку для парних чисел

}

else {

oddStack.push(value); // Додаємо непарне число до стеку для непарних чисел

}

}

// Виводимо стек парних чисел

std::cout << "Елементи стеку парних чисел:\n";

while (!evenStack.empty()) {

std::cout << evenStack.top() << " ";

evenStack.pop();

}

std::cout << "\n";

// Виводимо стек непарних чисел

std::cout << "Елементи стеку непарних чисел:\n";

while (!oddStack.empty()) {

std::cout << oddStack.top() << " ";

oddStack.pop();

}

std::cout << "\n";

}

bool isBalanced(const std::string& sequence) {

std::stack<char> brackets;

for (char bracket : sequence) {

if (bracket == '(' || bracket == '[') {

brackets.push(bracket); // Додаємо відкриту дужку до стеку

}

else if (bracket == ')' || bracket == ']') {

// Перевіряємо відповідність закритої дужки відкритій

if (brackets.empty() ||

(bracket == ')' && brackets.top() != '(') ||

(bracket == ']' && brackets.top() != '[')) {

return false;

}

brackets.pop(); // Видаляємо відповідну відкриту дужку зі стеку

}

}

// Перевіряємо, чи всі відкриті дужки були закриті

return brackets.empty();

}

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int choice = 0;

do {

cout << "1. Вивід стеку з цілими числами на екран в зворотньому порядку\n";

cout << "2. Вивід двох стеків з парними та непарними числами серед введених з клавіатури\n";

cout << "3. Перевірка правильності введення дужок за 10 секунд часу\n";

cout << "4. Вихід\n";

cout << "\nВиберіть дію: ";

std::cin >> choice;

switch (choice)

{

case 1:

firstQuest();

break;

case 2:

secondQuest();

break;

case 3: {

std::string sequence;

std::cout << "Введіть послідовність дужок: ";

// Використовуємо std::async для асинхронного вводу

auto inputFuture = std::async(std::launch::async, [&]() {

std::cin >> sequence;

}

);

const double maxExecutionTime = 10.0;

// Очікуємо завершення вводу протягом вказаного максимального часу

auto status = inputFuture.wait\_for(std::chrono::seconds(static\_cast<int>(maxExecutionTime)));

if (status == std::future\_status::timeout) {

std::cout << "\nВвід занадто довгий. Програма завершується. Натисніть ENTER для продовження\n";

break;

}

else {

// Перевіряємо баланс дужок

if (isBalanced(sequence)) {

std::cout << "Послідовність правильна.\n";

}

else {

std::cout << "Послідовність НЕправильна.\n";

}

}

break;

}

default:

break;

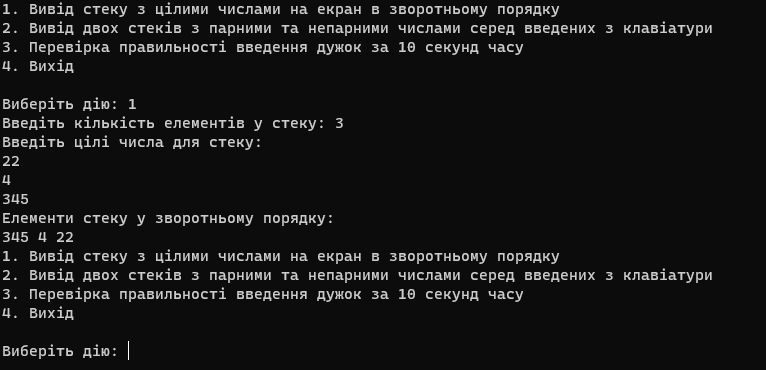
}

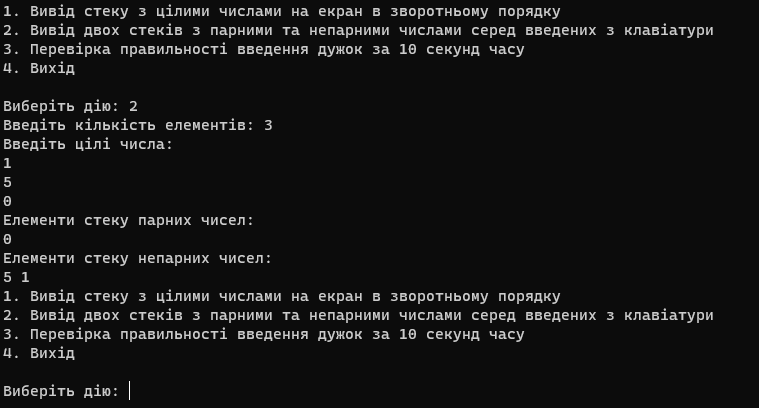
} while (choice!=4);

return 0;

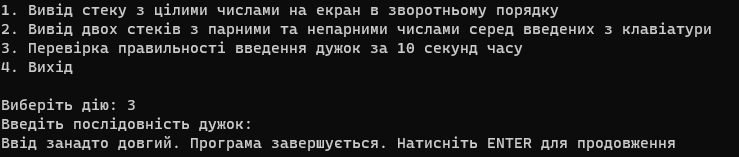
}

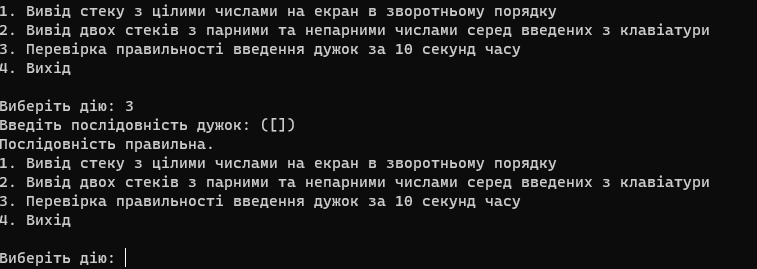
**Результати:**

**Ввід і вивід елементів стеку у зворотному порядку**

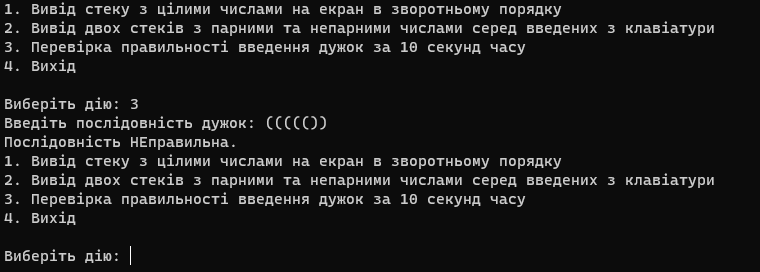
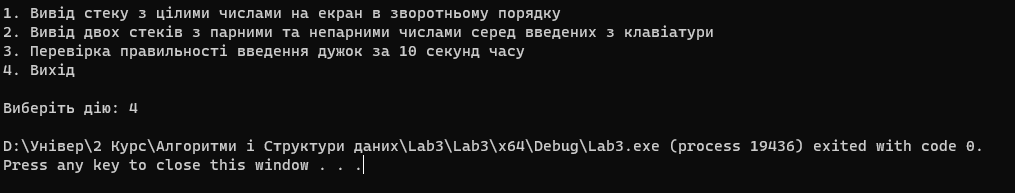
**Ввід елементів в стек і окремий вивід парних і непарних елементів **

**Неможливість вводу після 10 секунд очікування**

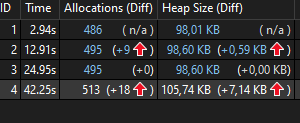
****

**Ввід дужок і вивід правильності їхнього вводу**

**Ввід дужок і вивід неправильності їхнього вводу**

**Завершення роботи прогами**

**Визначення затрат пам’яті на різних етапах виконання програми**



Затрати пам’яті значно збільшуються при використанні стеку типу char, але після одного заповнення затрати не змінюються. Отже введення дужок затрачає найбільше пам’яті.

**Висновок:** під час виконання лабораторної роботи я ознайомився з теоретичними відомостями і виконав завдання відповідно до варіанту. Була реалізована робота з стеками: їх заповнення, вивід елементів зі стеків та перевірка послідовності вводу.