Лабораторна робота № 7

Тема: Реалізація та опрацювання стеків.

Мета роботи: набуття практичних навичок побудови стеків та їх використання для розв’язування практичних задач.

Хід роботи

Завдання для виконання

1. Виконати Завдання 1 та Завдання 2, продемонструвавши роботу програми на прикладі рядка Вашого прізвища. Доповнити реалізацію програм невеликим меню, з якого можна вибирати операції заштовхування даних в стек, їх зчитування, перегляду без вилучення, перевірки на порожність тощо та викликати пункти меню у рандомному порядку не менш як двічі. Результат роботи виводити на екран після кожної операції та відобразити у звіті.

Завдання 1. Реалізуйте стек із типовими операціями на базі списку.

Лістинг 7.1 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <string>
#include <cstdlib>   // rand, srand
#include <ctime>     // time
using namespace std;
struct Node {
	char data;
	Node* next;
};
class Stack {
private:
	Node* top;
public:
	Stack() : top(nullptr) {}

	~Stack() {
		while (!isEmpty()) pop();
	}
	void push(char value) {
		Node* newNode = new Node{ value, top };
		top = newNode;
		cout << "push('" << value << "') — виконано.\n";
	}
	char pop() {
```

```
            if (isEmpty()) {
                  cout << "pop() — помилка: стек порожній.\n";
                  return '\0';
            }
            Node* tmp = top;
            char v = tmp->data;
            top = top->next;
            delete tmp;
            cout << "pop() -> '" << v << "'\n";
            return v;
      }
      char peek() const {
            if (isEmpty()) {
                  cout << "peek() — стек порожній.\n";
                  return '\0';
            }
            cout << "peek() -> '" << top->data << "'\n";
            return top->data;
      }

      bool isEmpty() const {
            return top == nullptr;
      }
      void display() const {
            if (isEmpty()) {
                  cout << "display(): [стек порожній]\n";
                  return;
            }
            cout << "display(): вершина -> ";
            Node* cur = top;
            while (cur) {
                  cout << cur->data;
                  if (cur->next) cout << " ";
                  cur = cur->next;
            }
            cout << " <- низ\n";
      }
};
void Text() {
      cout << "\nМеню (тести виконуються в випадковому порядку):\n"
            << "1 — push (додавання символу)\n"
            << "2 — pop (видалення вершини)\n"
            << "3 — peek (перегляд вершини)\n"
            << "4 — isEmpty (перевірка порожності)\n"
            << "5 — display (вивід стеку)\n\n";
}
int main() {
      setlocale(LC_ALL, "");
      srand((unsigned)time(nullptr));

      Stack st;
```

```
	string surname = "Chavurska"; // прізвище для демонстрації
	cout << "Демонстрація роботи стеку на прикладі прізвища: " << surname
<< "\n\n";
	Text();
	const int OPERATIONS = 5;
	const int REPEAT = 2; // кожну операцію повторити двічі
	const int TOTAL = OPERATIONS * REPEAT; // 10
	int ops[TOTAL];
	int idx = 0;
	for (int op = 1; op <= OPERATIONS; ++op) {
		for (int r = 0; r < REPEAT; ++r) {
			ops[idx++] = op;
		}
	}
	for (int i = TOTAL - 1; i > 0; --i) {
		int j = rand() % (i + 1);
		int tmp = ops[i]; ops[i] = ops[j]; ops[j] = tmp;
	}
	int Index = 0; // який символ прізвища вставляти далі при push
	for (int i = 0; i < TOTAL; ++i) {
		int choice = ops[i];
		cout << "\n=== Операція " << (i + 1) << " (випадковий вибір):
";
		switch (choice) {
		case 1: { // push
			cout << "push\n";
			if (Index < (int)surname.length()) {
				char c = surname[Index++];
				st.push(c);
			}
			else {
				cout << "push — немає більше символів у прізвищі для
додавання.\n";
			}
			break;
		}
		case 2: {
			cout << "pop\n";
			st.pop();
			break;
		}
		case 3: {
			cout << "peek\n";
			st.peek();
			break;
		}
		case 4: {
			cout << "isEmpty\n";
			if (st.isEmpty()) cout << "isEmpty() -> так (порожній)\n";
			else cout << "isEmpty() -> ні (містить елементи)\n";
			break;
```

```
            }
            case 5: {
                  cout << "display\n";
                  st.display();
                  break;
            }
            default:
                  break;
            }
            cout << "Поточний стан після операції:\n";
            st.display();
      }
      cout << "\n=== Завершення демонстрації ===\n";
      cout << "Кінцевий стан стеку:\n";
      st.display();
      cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7\n";
      return 0;
}
```

```
Демонстрац?я роботи стеку на приклад? пр?звища: Chavurska


Меню (тести виконуються в випадковому порядку):
1 - push (додавання символу)
2 - pop (видалення вершини)
3 - peek (перегляд вершини)
4 - isEmpty (перев?рка порожност?)
5 - display (вив?д стеку)


=== Операц?я 1 (випадковий виб?р): push
push('C') - виконано.
Поточний стан п?сля операц?ї:
display(): вершина -> C <- низ

=== Операц?я 2 (випадковий виб?р): peek
peek() -> 'C'
Поточний стан п?сля операц?ї:
display(): вершина -> C <- низ

=== Операц?я 3 (випадковий виб?р): display
display(): вершина -> C <- низ
Поточний стан п?сля операц?ї:
display(): вершина -> C <- низ

=== Операц?я 4 (випадковий виб?р): pop
pop() -> 'C'
Поточний стан п?сля операц?ї:
display(): [стек порожн?й]

=== Операц?я 5 (випадковий виб?р): peek
peek() - стек порожн?й.
Поточний стан п?сля операц?ї:
display(): [стек порожн?й]

=== Операц?я 6 (випадковий виб?р): push
push('h') - виконано.
Поточний стан п?сля операц?ї:
display(): вершина -> h <- низ

=== Операц?я 7 (випадковий виб?р): isEmpty
isEmpty() -> н? (м?стить елементи)
Поточний стан п?сля операц?ї:
display(): вершина -> h <- низ

=== Операц?я 8 (випадковий виб?р): pop
pop() -> 'h'
Поточний стан п?сля операц?ї:
display(): [стек порожн?й]

=== Операц?я 9 (випадковий виб?р): display
display(): [стек порожн?й]
Поточний стан п?сля операц?ї:
display(): [стек порожн?й]

=== Операц?я 10 (випадковий виб?р): isEmpty
isEmpty() -> так (порожн?й)
Поточний стан п?сля операц?ї:
display(): [стек порожн?й]

=== Завершення демонстрац?ї ===
К?нцевий стан стеку:
display(): [стек порожн?й]

Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.1 – Вікно результатів програми

Завдання 2. Реалізуйте стек із типовими операціями як рекурсивну структуру даних.

Лістинг 7.2 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;
struct Node {
	char data;
	Node* next;
};
class RecursiveStack {
private:
	Node* top;

	// допоміжна рекурсивна функція для виводу стеку
	void displayRecursive(Node* node) const {
		if (node == nullptr) return;
		cout << node->data << " ";
		displayRecursive(node->next);
	}

	// допоміжна функція для очищення (рекурсивний деструктор)
	void clear(Node* node) {
		if (node == nullptr) return;
		clear(node->next);
		delete node;
	}

public:
	RecursiveStack() : top(nullptr) {}
	~RecursiveStack() { clear(top); }

	bool isEmpty() const {
		return top == nullptr;
	}

	// рекурсивне додавання елемента
	void push(char value) {
		Node* newNode = new Node{ value, top };
		top = newNode;
		cout << "push('" << value << "') – виконано.\n";
	}

	// рекурсивне вилучення
	char pop() {
		if (isEmpty()) {
```

```
            cout << "pop() — стек порожній.\n";
            return '\0';
        }
        char value = top->data;
        Node* temp = top;
        top = top->next;
        delete temp;
        cout << "pop() -> '" << value << "'\n";
        return value;
    }

    // рекурсивне отримання вершини
    char peek() const {
        if (isEmpty()) {
            cout << "peek() — стек порожній.\n";
            return '\0';
        }
        cout << "peek() -> '" << top->data << "'\n";
        return top->data;
    }

    // рекурсивний вивід стеку
    void display() const {
        if (isEmpty()) {
            cout << "display(): [стек порожній]\n";
            return;
        }
        cout << "display(): вершина -> ";
        displayRecursive(top);
        cout << "<- низ\n";
    }
};

void Menu() {
    cout << "\n===== МЕНЮ ОПЕРАЦІЙ =====\n";
    cout << "1 — push (додати символ)\n";
    cout << "2 — pop (вилучити верхній символ)\n";
    cout << "3 — peek (переглянути вершину)\n";
    cout << "4 — isEmpty (перевірка порожності)\n";
    cout << "5 — display (вивести стек)\n";
    cout << "==========================\n";
}

int main() {
    setlocale(LC_ALL, "");
    srand((unsigned)time(nullptr));

    RecursiveStack stack;
    string surname = "Chavurska";

    cout << "Рекурсивна реалізація стеку для рядка: " << surname << endl;
```

```
Menu();

// створимо випадкову послідовність із 10 операцій (по 2 рази кожна)
const int N = 10;
int operations[N];
int k = 0;
for (int op = 1; op <= 5; op++) {
	operations[k++] = op;
	operations[k++] = op;
}
// перемішуємо операції (Fisher-Yates shuffle)
for (int i = N - 1; i > 0; i--) {
	int j = rand() % (i + 1);
	int tmp = operations[i];
	operations[i] = operations[j];
	operations[j] = tmp;
}
int nextChar = 0;
cout << "\n=== Початок демонстрації ===\n";
for (int i = 0; i < N; i++) {
	int choice = operations[i];
	cout << "\nОперація " << i + 1 << ": ";
	switch (choice) {
	case 1:
		cout << "push()\n";
		if (nextChar < (int)surname.size()) {
			stack.push(surname[nextChar++]);
		}
		else {
			cout << "push() — усі символи вже додано.\n";
		}
		break;
	case 2:
		cout << "pop()\n";
		stack.pop();
		break;
	case 3:
		cout << "peek()\n";
		stack.peek();
		break;
	case 4:
		cout << "isEmpty()\n";
		if (stack.isEmpty())
			cout << "Стек порожній.\n";
		else
			cout << "Стек містить елементи.\n";
		break;
	case 5:
		cout << "display()\n";
		stack.display();
		break;
```

```
            }

            // показати стан після кожної операції
            cout << "Поточний стан стеку:\n";
            stack.display();
        }
        cout << "\n=== Завершення демонстрації ===\n";
        stack.display();
        cout << "\nЧавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7\n";
        return 0;
}
```

```
Рекурсивна реал?зац?я стеку для рядка: Chavurska

===== МЕНЮ ОПЕРАЦ?Й =====
1 – push (додати символ)
2 – pop (вилучити верхн?й символ)
3 – peek (переглянути вершину)
4 – isEmpty (перев?рка порожност?)
5 – display (вивести стек)
==========================

=== Початок демонстрац?ї ===

Операц?я 1: isEmpty()
Стек порожн?й.
Поточний стан стеку:
display(): [стек порожн?й]

Операц?я 2: push()
push('C') – виконано.
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 3: display()
display(): вершина -> C <- низ
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 4: peek()
peek() -> 'C'
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 5: display()
display(): вершина -> C <- низ
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 6: isEmpty()
Стек м?стить елементи.
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 7: peek()
peek() -> 'C'
Поточний стан стеку:
display(): вершина -> C <- низ

Операц?я 8: pop()
pop() -> 'C'
Поточний стан стеку:
display(): [стек порожн?й]

Операц?я 9: push()
push('h') – виконано.
Поточний стан стеку:
display(): вершина -> h <- низ

Операц?я 10: pop()
pop() -> 'h'
Поточний стан стеку:
display(): [стек порожн?й]

=== Завершення демонстрац?ї ===
display(): [стек порожн?й]

Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.2 – Вікно результатів програми

2. Виконати Завдання 3 та перевірити роботу програми, перевівши восьмизначне десяткове число, утворене з дати Вашого народження у всі системи числення з базами від 2 до 16. Результат роботи вивести на екран та занести у звіт.

Завдання 3. Реалізуйте алгоритм конвертування числа з десяткової системи числення у довільну від 2 до 16.

Лістинг 7.3 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <string>
using namespace std;
// Функція конвертації десяткового числа у систему з базою base (2–16)
string func(int number, int base) {
	if (number == 0) return "0";
	string digits = "0123456789ABCDEF";
	string result = "";
	// Рекурсивне побудування результату
	if (number > 0) {
		result = func(number / base, base);  // поділили число
		result += digits[number % base];              // додали остачу
	}
	return result;
}
int main() {
	setlocale(LC_ALL, "ukr");
	// Наприклад, дата народження: 15.06.2004 → 15062004
	int number = 15062004;
	cout << "Перетворення десяткового числа " << number<< " у всі системи числення від 2 до 16:\n\n";
	for (int base = 2; base <= 16; ++base) {
		string converted = func(number, base);
		cout << "Основа " << base << ": " << converted << endl;
	}
	cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7";
	return 0;
}
```

```
Перетворення десяткового числа 15062004 у вс? системи числення в?д 2 до 16:

Основа 2: 01110010111010011111110100
Основа 3: 01001100020012000
Основа 4: 0321131033310
Основа 5: 012323441004
Основа 6: 01254455300
Основа 7: 0242011356
Основа 8: 071351764
Основа 9: 031306160
Основа 10: 015062004
Основа 11: 08558341
Основа 12: 05064530
Основа 13: 03174939
Основа 14: 020010D6
Основа 15: 014C7C39
Основа 16: 0E5D3F4
Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.3 – Вікно результатів програми

3. Виконати Завдання 4 (додатково) та перевірити роботу алгоритму для трьох чисел у довільній системі числення, крім десяткової у п’ять інших випадкових систем числення. Результати занести у звіт.

Завдання 4. Реалізуйте алгоритм конвертування числа з довільної системи числення у довільну від 2 до 16.

Лістинг 7.4 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
// Функція для переведення числа з будь-якої системи у десяткову
int toDecimal(string number, int fromBase) {
	int value = 0;
	int power = 1;
	int len = number.length();
	for (int i = len - 1; i >= 0; i--) {
		char c = toupper(number[i]);
		int digit;
		if (c >= '0' && c <= '9') digit = c - '0';
		else digit = c - 'A' + 10;
		if (digit >= fromBase) {
			cout << "Помилка: символ '" << c
				<< "' недійсний для системи з основою " << fromBase
<< endl;
			return -1;
		}
```

```
            value += digit * power;
            power *= fromBase;
      }
      return value;
}
// Функція для переведення десяткового числа у будь-яку систему
string fromDecimal(int number, int toBase) {
      if (number == 0) return "0";
      string digits = "0123456789ABCDEF";
      string result = "";
      while (number > 0) {
            result = digits[number % toBase] + result;
            number /= toBase;
      }
      return result;
}
// Функція для переведення між довільними системами числення
string convertBase(string number, int fromBase, int toBase) {
      int decimalValue = toDecimal(number, fromBase);
      if (decimalValue < 0) return "Помилка!";
      return fromDecimal(decimalValue, toBase);
}
int main() {
      setlocale(LC_ALL, "ukr");
      // Демонстрація роботи для трьох чисел
      string numbers[3] = { "101101", "7A3", "2F" };
      int fromBases[3] = { 2, 16, 8 }; // бази, з яких конвертуємо
      cout << "Перетворення чисел з різних систем у 5 інших
випадкових:\n\n";
      for (int i = 0; i < 3; i++) {
            cout << "Число: " << numbers[i]
                  << " (основа " << fromBases[i] << ")\n";
            // Випадкові цільові системи (крім початкової)
            int targets[5] = { 3, 5, 7, 10, 12 };

            for (int j = 0; j < 5; j++) {
                  if (targets[j] != fromBases[i]) {
                        string converted = convertBase(numbers[i],
fromBases[i], targets[j]);
                        cout << "  → Основа " << targets[j] << ": " <<
converted << endl;
                  }
            }

            cout << "----------------------------------------\n";
      }

      cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7\n";
      return 0;
}
```

```
  ? Основа 7: 63
  ? Основа 10: 45
  ? Основа 12: 39
----------------------------------------
Число: 7A3 (основа 16)
  ? Основа 3: 2200102
  ? Основа 5: 30310
  ? Основа 7: 5462
  ? Основа 10: 1955
  ? Основа 12: 116B
----------------------------------------
Число: 2F (основа 8)
Помилка: символ 'F' нед?йсний для системи з основою 8
  ? Основа 3: Помилка!
Помилка: символ 'F' нед?йсний для системи з основою 8
  ? Основа 5: Помилка!
Помилка: символ 'F' нед?йсний для системи з основою 8
  ? Основа 7: Помилка!
Помилка: символ 'F' нед?йсний для системи з основою 8
  ? Основа 10: Помилка!
Помилка: символ 'F' нед?йсний для системи з основою 8
  ? Основа 12: Помилка!
----------------------------------------
Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.4 – Вікно результатів програми

4. Написати програму, що реалізовує алгоритм перевірки на правильність дужкової послідовності (Завдання 5) та реалізувати її. Перевірити правильність роботи програми на не менш як 7ми власних прикладах. Результати занести у звіт.

Завдання 5. Реалізуйте алгоритм перевірки на правильність дужкової послідовності.

Лістинг 7.5 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <string>
using namespace std;
// Реалізація стеку
class Stack {
private:
	char data[100];
	int top;
public:
```

```
    Stack() { top = -1; }

    void push(char c) {
        if (top < 99)
            data[++top] = c;
    }
    char pop() {
        if (top >= 0)
            return data[top--];
        return '\0';
    }
    char peek() const {
        if (top >= 0)
            return data[top];
        return '\0';
    }

    bool isEmpty() const {
        return top == -1;
    }
};
bool isBalanced(const string& expr) {
    Stack stack;
    for (char c : expr) {
        if (c == '(' || c == '[' || c == '{')
            stack.push(c);
        else if (c == ')' || c == ']' || c == '}') {
            if (stack.isEmpty()) return false;
            char top = stack.pop();
            if ((c == ')' && top != '(') ||
                (c == ']' && top != '[') ||
                (c == '}' && top != '{'))
                return false;
        }
    }
    return stack.isEmpty();
}
int main() {
    setlocale(LC_ALL, "ukr");
    string examples[7] = {
        "()",           // правильна
        "(a+b)*[c-d]",  // правильна
        "{[()]}",       // правильна
        "((a+b)",       // не вистачає дужки
        "[a+b]*(c+d})", // неправильне закриття
        "(([]){})",     // правильна
        "{[(])}"        // неправильна вкладеність
    };
    cout << "Перевірка правильності дужкових послідовностей:\n\n";
    for (int i = 0; i < 7; i++) {
        cout << "Приклад " << i + 1 << ": " << examples[i] << endl;
```

```
            if (isBalanced(examples[i]))
                cout << "  Правильна \n";
            else
                cout << "  Неправильна \n";
            cout << "-----------------------------------\n";
        }
        cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7\n";
        return 0;
}
```

```
Перев?рка правильност? дужкових посл?довностей:

Приклад 1: ()
  Правильна
-----------------------------------
Приклад 2: (a+b)*[c-d]
  Правильна
-----------------------------------
Приклад 3: {[()]}
  Правильна
-----------------------------------
Приклад 4: ((a+b)
  Неправильна
-----------------------------------
Приклад 5: [a+b]*(c+d})
  Неправильна
-----------------------------------
Приклад 6: (([]){})
  Правильна
-----------------------------------
Приклад 7: {[(])}
  Неправильна
-----------------------------------
Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.5 – Вікно результатів програми

5. (ДОДАТКОВО) Написати програму, що обчислює арифметичний вираз, заданий у рядковій формі (Завдання 6) та реалізувати її. Перевірити правильність роботи програми на не менш як 7ми власних прикладах. Результати занести у звіт.

Завдання 6. Реалізуйте клас, що обчислює вираз, заданий рядом токенів. На його основі розробіть повний алгоритм та реалізуйте програму обчислення арифметичного виразу, заданого рядком.

Лістинг 7.6 – Код програми

```
//Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7
#include <iostream>
#include <stack>
#include <string>
#include <cctype>
#include <sstream>
using namespace std;
int precedence(char op) {
	switch (op) {
	case '+':
	case '-':
		return 1;  // найнижчий пріоритет
	case '*':
	case '/':
		return 2;  // вищий пріоритет
	case '^':
		return 3;  // ще вищий (для степеня, якщо додати)
	default:
		return 0;  // не оператор
	}
}
double a(double a, double b, char op) {
	switch (op) {
	case '+': return a + b;
	case '-': return a - b;
	case '*': return a * b;
	case '/': return b != 0 ? a / b : 0;
	default: return 0;
	}
}
double e(const string& expr) {
	stack<double> values;  // числа
	stack<char> ops;       // оператори
	for (size_t i = 0; i < expr.length(); i++) {
		if (expr[i] == ' ') continue; // пропускаємо пробіли

		if (isdigit(expr[i])) {
			double val = 0;
			while (i < expr.length() && (isdigit(expr[i]) || expr[i]
== '.')) {
				if (expr[i] == '.') {
					double frac = 0.1;
					i++;
					while (i < expr.length() && isdigit(expr[i])) {
						val += (expr[i] - '0') * frac;
						frac /= 10;
						i++;
					}
					break;
```

```
                    }
                    val = val * 10 + (expr[i] - '0');
                    i++;
                }
                values.push(val);
                i--;
            }
            else if (expr[i] == '(') {
                ops.push(expr[i]);
            }
            else if (expr[i] == ')') {
                while (!ops.empty() && ops.top() != '(') {
                    double val2 = values.top(); values.pop();
                    double val1 = values.top(); values.pop();
                    char op = ops.top(); ops.pop();
                    values.push(a(val1, val2, op));
                }
                ops.pop();
            }
            else {
                while (!ops.empty() && precedence(ops.top()) >=
precedence(expr[i])) {
                    double val2 = values.top(); values.pop();
                    double val1 = values.top(); values.pop();
                    char op = ops.top(); ops.pop();
                    values.push(a(val1, val2, op));
                }
                ops.push(expr[i]);
            }
        }
        while (!ops.empty()) {
            double val2 = values.top(); values.pop();
            double val1 = values.top(); values.pop();
            char op = ops.top(); ops.pop();
            values.push(a(val1, val2, op));
        }
        return values.top();
}
int main() {
        setlocale(LC_ALL, "ukr");
        string examples[7] = {
            "3 + 5",
            "(3 + 5) * 2",
            "10 + 2 * 6",
            "100 * (2 + 12) / 14",
            "7 + (6 * 5^2 + 3)",
            "50 / (5 * 2)",
            "8 + 2 * (3 + 4) - 10 / 2"
        };
        cout << "Перевірка правильності роботи алгоритму:\n\n";
        for (int i = 0; i < 7; i++) {
```

```
            cout << "Вираз " << i + 1 << ": " << examples[i] << endl;
            cout << "Результат = " << e(examples[i]) << endl;
            cout << "-----------------------------------\n";
        }
        cout << "Чавурська Єлизавета, П-31, АіСД, Варіант 20, ЛР7\n";
        return 0;
}
```

```
Перев?рка правильност? роботи алгоритму:

Вираз 1: 3 + 5
Результат = 8
-----------------------------------
Вираз 2: (3 + 5) * 2
Результат = 16
-----------------------------------
Вираз 3: 10 + 2 * 6
Результат = 22
-----------------------------------
Вираз 4: 100 * (2 + 12) / 14
Результат = 100
-----------------------------------
Вираз 5: 7 + (6 * 5^2 + 3)
Результат = 10
-----------------------------------
Вираз 6: 50 / (5 * 2)
Результат = 5
-----------------------------------
Вираз 7: 8 + 2 * (3 + 4) - 10 / 2
Результат = 17
-----------------------------------
Чавурська Єлизавета, П-31, А?СД, Вар?ант 20, ЛР7
```

Рисунок 7.6 – Вікно результатів програми

6. Зробити висновок про роботу та записати у звіт.
7. Бути готовим до захисту.

Висновок: на цій лабораторній роботі я набула практичних навичок побудови стеків та їх використання для розв’язування практичних задач.