**Лабораторна робота №4**

**Тема**: Зворотний польський запис та обчислення математичного виразу.

**Мета**: Використовуючи теоретичне підґрунтя про зворотний польський запис розробити програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

**Завдання до лабораторної роботи**

Використовуючи теоретичне відомості розробити програму яка на вхід отримує математичний вираз з довільною кількістю операндів, операторів та дужок. В першу чергу сформувати послідовність символів у ЗПН. На другому етапі виконання лабораторної роботи вирахувати результат послідовності, що була сформована, використовуючи алгоритм запису математичного виразу у ЗПН.

**Хід роботи**

Створив функцію infix\_to\_postfix який перетворює інфіксний вираз у ЗПН.

1. Цей код створює дві змінні:

* stack: для тимчасового зберігання операторів.
* postfix: для зберігання результату (виразу в ЗПН).

1. Для кожного символу у виразі:

* Якщо символ — число, додає його до postfix.
* Якщо символ — ліва дужка '(', додає його до stack.
* Якщо символ — права дужка ')', витягує оператори зі стеку до postfix, доки не знайде '('.

1. Якщо символ — оператор (+, -, \*, /) витягує оператори зі стеку в postfix, доки у стека вищий або рівний пріоритет. Додає оператор у стек. Потім після обробки виразу додає всі залишки зі стеку в postfix, та повертає результат у вигляді рядка.

def infix\_to\_postfix(expression):

    precedence = {'+': 1, '-': 1, '\*': 2, '/': 2, '(': 0, ')': 0}

    stack = []

    postfix = []

    for char in expression:

        if char.isnumeric():  # Якщо це операнд (число)

            postfix.append(char)

        elif char == '(':  # Якщо це ліва дужка

            stack.append(char)

        elif char == ')':  # Якщо це права дужка

            while stack and stack[-1] != '(':

                postfix.append(stack.pop())

            stack.pop()  # Викидаємо '('

        else:  # Якщо це оператор

            while stack and precedence[char] <= precedence.get(stack[-1], 0):

                postfix.append(stack.pop())

            stack.append(char)

    while stack:

        postfix.append(stack.pop())

    return ''.join(postfix)

Створив функцію evaluate\_postfix яка обчислює значення виразу, представленого у ЗПН.

1. Створює стек stack.
2. Для кожного символу в ЗПН:

* Якщо символ — число, додає його в стек.
* Якщо символ — оператор:
* Витягує два останні числа зі стеку (a і b).
* Застосовує оператор до цих чисел (a + b, a - b, тощо).
* Додає результат назад у стек.

1. Після завершення обробки виразу у стеку залишається лише результат.
2. Повертає результат.

def evaluate\_postfix(expression):

    stack = []

    for char in expression:

        if char.isnumeric():  # Якщо це операнд

            stack.append(int(char))

        else:  # Якщо це оператор

            b = stack.pop()

            a = stack.pop()

            if char == '+':

                stack.append(a + b)

            elif char == '-':

                stack.append(a - b)

            elif char == '\*':

                stack.append(a \* b)

            elif char == '/':

                stack.append(a / b)

    return stack.pop()

Код програми:

# Функція для переведення виразу в ЗПН

def infix\_to\_postfix(expression):

    precedence = {'+': 1, '-': 1, '\*': 2, '/': 2, '(': 0, ')': 0}

    stack = []

    postfix = []

    for char in expression:

        if char.isnumeric():  # Якщо це операнд (число)

            postfix.append(char)

        elif char == '(':  # Якщо це ліва дужка

            stack.append(char)

        elif char == ')':  # Якщо це права дужка

            while stack and stack[-1] != '(':

                postfix.append(stack.pop())

            stack.pop()  # Викидаємо '('

        else:  # Якщо це оператор

            while stack and precedence[char] <= precedence.get(stack[-1], 0):

                postfix.append(stack.pop())

            stack.append(char)

    while stack:

        postfix.append(stack.pop())

    return ''.join(postfix)

# Функція для обчислення результату виразу в ЗПН

def evaluate\_postfix(expression):

    stack = []

    for char in expression:

        if char.isnumeric():  # Якщо це операнд

            stack.append(int(char))

        else:  # Якщо це оператор

            b = stack.pop()

            a = stack.pop()

            if char == '+':

                stack.append(a + b)

            elif char == '-':

                stack.append(a - b)

            elif char == '\*':

                stack.append(a \* b)

            elif char == '/':

                stack.append(a / b)

    return stack.pop()

# Тестування

expression = input("Введіть інфіксний вираз: ")

# Перетворюємо вираз у ЗПН

postfix = infix\_to\_postfix(expression)

print(f"Вираз у ЗПН: {postfix}")

# Обчислюємо результат

result = evaluate\_postfix(postfix)

print(f"Результат: {result}")

**Висновок:** Використовуючи теоретичне підґрунтя про зворотний польський запис я розробив програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.