Звіт до лабораторної роботи №4

**Зворотний польський запис та обчислення 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.

**Виконання роботи:**

**Текст програми:**

|  |
| --- |
| class InfixToPostfixConverter:      """Клас для перетворення інфіксного виразу в зворотний польський запис (ЗПЗ)."""        precedence = {'+': 1, '-': 1, '\*': 2, '/': 2, '^': 3}        def \_\_init\_\_(self, expression):  *self*.expression = expression  *self*.stack = []  *self*.output = []        def validate(self):          """Перевірка на правильність інфіксного виразу."""          balance = 0          prev\_token = None          for token in *self*.expression.replace("(", " ( ").replace(")", " ) ").split():              if token == '(':                  balance += 1              elif token == ')':                  balance -= 1              elif token in *self*.precedence and (prev\_token in *self*.precedence or prev\_token == '('):                  return False  # Неправильне розташування оператора              elif not token.isdigit() and token not in *self*.precedence:                  return False  # Не допустимий символ              prev\_token = token          return balance == 0        def convert(self):          """Метод для перетворення виразу в ЗПЗ."""          if not *self*.validate():              raise ValueError("Invalid infix expression")            for token in *self*.expression.replace("(", " ( ").replace(")", " ) ").split():              if token.isdigit():  *self*.output.append(token)              elif token == '(':  *self*.stack.append(token)              elif token == ')':                  while *self*.stack[-1] != '(':  *self*.output.append(*self*.stack.pop())  *self*.stack.pop()              else:  # Оператор                  while *self*.stack and *self*.precedence.get(*self*.stack[-1], 0) >= *self*.precedence[token]:  *self*.output.append(*self*.stack.pop())  *self*.stack.append(token)            while *self*.stack:  *self*.output.append(*self*.stack.pop())            return *self*.output  class PostfixEvaluator:      """Клас для обчислення виразу в зворотному польському записі (ЗПЗ)."""        operations = {          '+': lambda a, b: a + b,          '-': lambda a, b: a - b,          '\*': lambda a, b: a \* b,          '/': lambda a, b: a / b if b != 0 else 'Zero Division Error',          '^': lambda a, b: a \*\* b,      }        def \_\_init\_\_(self, postfix\_expression):  *self*.postfix\_expression = postfix\_expression  *self*.stack = []        def evaluate(self):          """Метод для обчислення ЗПЗ виразу."""          for token in *self*.postfix\_expression:              if token.isdigit():  *self*.stack.append(int(token))              else:                  b, a = *self*.stack.pop(), *self*.stack.pop()                  result = *self*.operations[token](a, b)                  if result == 'Zero Division Error':                      return result  *self*.stack.append(result)          return *self*.stack[0]  class ExpressionEvaluator:      """Клас для повного обчислення виразу від інфіксної форми до результату."""        def \_\_init\_\_(self, expression):  *self*.expression = expression        def evaluate(self):          # Перетворюємо інфіксну форму в ЗПЗ          converter = InfixToPostfixConverter(*self*.expression)          try:              postfix\_expression = converter.convert()          except ValueError as e:              return str(e)            # Виводимо ЗПЗ          print("Postfix expression:", ' '.join(postfix\_expression))            # Обчислюємо результат ЗПЗ          evaluator = PostfixEvaluator(postfix\_expression)          result = evaluator.evaluate()          return result  # Приклад використання  if \_\_name\_\_ == "\_\_main\_\_":      expression = input("Enter an infix expression: ")      evaluator = ExpressionEvaluator(expression)      result = evaluator.evaluate()      print(f"Result: {result}") |

**Посилання на github:**

<https://github.com/Alexxxxo/TP-KB-231-Oleksandr-Kozachok/tree/main/lab_04>

Висновок: у результаті виконання лабораторної роботи було розроблено програму, яка перетворює математичні вирази з інфіксної форми на зворотний польський запис (ЗПН) і дозволяє обчислити результат цього виразу. Процес перетворення в ЗПН базується на алгоритмі, який використовує стек для обробки операторів і дужок. Завдяки цьому, математичні вирази можуть бути представлені без необхідності в дужках для визначення пріоритету операцій. Після того як вираз перетворений в ЗПН, результат обчислюється за допомогою другого етапу програми, що здійснює посторечне обчислення.