Технології програмування

Лабораторна робота №4

**Зворотний польський запис та обчислення математичного виразу**

**Новик Максим Вікторович, КБ-231**

**Мета роботи**: 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.

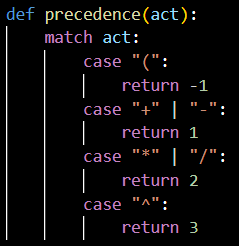
Текст програми разом зі звітом розмістити в директорії lab\_04. Директорію lab\_04 розмістити в директорії, що використовується для виконання практичних завдань по кожній лекції та має назву **TP-KB-23[1 or 2]-Name-Surname**.

**Хід виконання завдання:**

Перш за все додаю список інструментів, знаків, що можуть бути в прикладі:

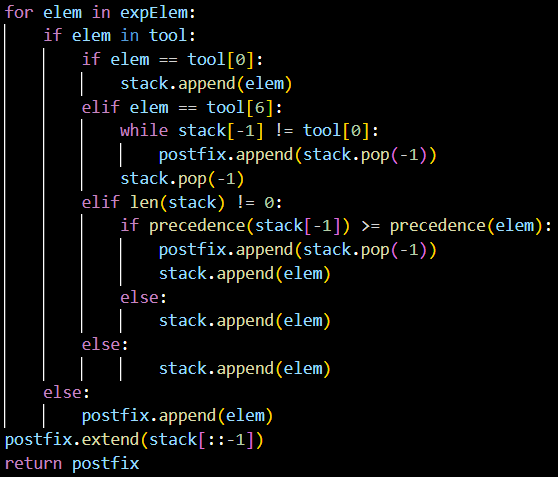


Потім будую пріоритетність операторів для обробки виразів:



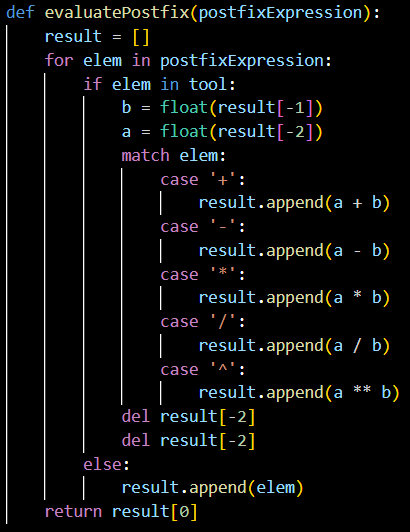
* Дужка ( має найнижчий пріоритет (-1).
* + і - мають пріоритет 1.
* \* і / мають пріоритет 2.
* ^ (степінь) має найвищий пріоритет 3.

Створюю функцію, що перетворює вираз з інфіксного (звичайного) формату в постфіксний (зворотний польський запис, RPN):



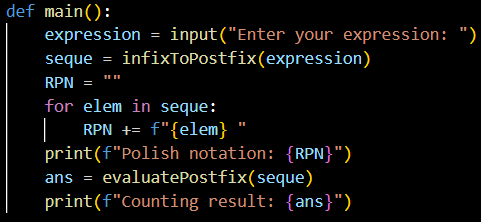
* Розбиває вираз на елементи (expElem), використовуючи пробіли як роздільник.
* Використовує два списки:
* stack для тимчасового зберігання операторів та дужок.
* postfix для результатного постфіксного виразу.
* **Для кожного елемента:**
* Якщо це відкрита дужка (, додається в стек.
* Якщо це закрита дужка ), всі оператори до відкритої дужки переміщуються в постфіксний вираз.
* Якщо це оператор, порівнюється його пріоритет із верхнім елементом стека:
* Якщо пріоритет оператора нижчий або рівний, верхній елемент переміщається в postfix.
* Інакше оператор додається в стек.
* Якщо це число/операнд, додається в postfix.
* Наприкінці залишені в стеку оператори додаються до postfix.

Створюю функцію для обчислення значень виразу у постфіксному форматі:



* Використовує стек result для проміжних результатів.
* **Для кожного елемента:**
* Якщо це оператор:
* Дістає два останніх числа зі стеку (a і b).
* Виконує відповідну операцію (+, -, \*, /, ^).
* Результат додає назад у стек.
* Якщо це число/операнд:
* Додає його в стек.
* Наприкінці повертає єдиний елемент у стеку — результат обчислення.

Та функція main(), що забезпечує інтерфейс для вводу виразу та виводу результатів:



* Приймає вираз у інфіксному форматі від користувача.
* Використовує infixToPostfix() для перетворення виразу у постфіксний формат.
* Виводить постфіксний вираз (RPN).
* Обчислює результат за допомогою evaluatePostfix().
* Виводить результат обчислення.

Текст програми:

|  |
| --- |
| tool = ["(", "+", "-", "\*", "/", "^", ")"]  def precedence(act):      match act:          case "(":              return -1          case "+" | "-":              return 1          case "\*" | "/":              return 2          case "^":              return 3  def infixToPostfix(expression):      expElem = expression.split()      stack = []      postfix = []        for elem in expElem:          if elem in tool:              if elem == tool[0]:                  stack.append(elem)              elif elem == tool[6]:                  while stack[-1] != tool[0]:                      postfix.append(stack.pop(-1))                  stack.pop(-1)              elif len(stack) != 0:                  if precedence(stack[-1]) >= precedence(elem):                      postfix.append(stack.pop(-1))                      stack.append(elem)                  else:                      stack.append(elem)              else:                      stack.append(elem)          else:              postfix.append(elem)      postfix.extend(stack[::-1])      return postfix  def evaluatePostfix(postfixExpression):      result = []      for elem in postfixExpression:          if elem in tool:              b = float(result[-1])              a = float(result[-2])              match elem:                  case '+':                      result.append(a + b)                  case '-':                      result.append(a - b)                  case '\*':                      result.append(a \* b)                  case '/':                      result.append(a / b)                  case '^':                      result.append(a \*\* b)              del result[-2]              del result[-2]          else:              result.append(elem)      return result[0]  def main():      expression = input("Enter your expression: ")      seque = infixToPostfix(expression)      RPN = ""      for elem in seque:          RPN += f"{elem} "      print(f"Polish notation: {RPN}")      ans = evaluatePostfix(seque)      print(f"Counting result: {ans}")    if \_\_name\_\_ == "\_\_main\_\_":      main() |

**Висновок:**

Під час виконання лабораторної роботи я використовував теоретичне підґрунтя про зворотний польський запис розробляв програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

**Посилання на github:**

https://github.com/GAGGAGX/TP-KB-231-Novyk-Maksym

**Знімок екрану з посилання на github:**

