Технології програмування

Лабораторна робота №4

**Зворотний польський запис та обчислення математичного виразу**

**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 ЗПЗ. На другому етапі виконання лабораторної роботи вирахувати результат послідовності, що була сформована, використовуючи алгоритм запису математичного виразу у ЗПЗ.

**Хід роботи:**

1. Теоретичні відомості про ЗПЗ:

Зворотний польський запис (ЗПЗ) — це форма запису математичних виразів, у якій оператори слідують після своїх операндів, наприклад, вираз `a b +` замість звичного `a + b`. Цей запис, запропонований польським логіком Ян Лукасевичем, має важливу особливість: його можна обчислювати за один прохід, використовуючи стек. Алгоритм перетворення інфіксного виразу в ЗПЗ, відомий як «сортувальна станція», використовує стек для зберігання операторів і забезпечує правильний порядок операцій, враховуючи пріоритети операторів. Після перетворення виразу в ЗПЗ, обчислення відбувається шляхом обробки кожного елемента виразу зліва направо, де числа додаються в стек, а оператори виконуються над двома верхніми елементами стеку. Цей підхід дозволяє ефективно виконувати обчислення без необхідності враховувати пріоритети операторів під час обчислень.

1. Розробила програму, що призначена для перетворення математичних виразів з інфіксної нотації (де операції знаходяться між операндами, наприклад, 3 + 4) у постфіксну нотацію (так звану обернену польську нотацію, де операції йдуть після операндів, наприклад, 3 4 +), а також для обчислення результату постфіксного виразу.

Програма складається з декількох функцій, кожна з яких виконує конкретне завдання:

* precedence(op): Функція, яка визначає пріоритет оператора.
* selectedOperator(op, b, a): Функція для виконання операцій над двома числами.
* infixToPostfix(expression): Функція для перетворення інфіксного виразу в постфіксний.
* evaluatePostfix(expression): Функція для обчислення постфіксного виразу.
* main(): Головна функція для введення математичного виразу та виведення результату.

1. Код програми:

|  |
| --- |
| def precedence(op):  match op:  case '^': return 3  case '\*' | '/': return 2  case '+' | '-': return 1  case \_: return 0    def selectedOperator(op, b, a):  match op:  case '+': return a + b  case '-': return a - b  case '\*': return a \* b  case '/': return a / b  case '^': return a \*\* b    def infixToPostfix(expression):  stack = []  output = []  for char in expression:  match char:  case digit if digit.isdigit():  output.append(digit)  case '(':  stack.append(char)  case ')':  while stack and stack[-1] != '(':  output.append(stack.pop())  stack.pop()  case operator:  while stack and precedence(stack[-1]) >= precedence(operator):  output.append(stack.pop())  stack.append(operator)    while stack:  output.append(stack.pop())    return ''.join(output)    def evaluatePostfix(expression):  stack = []  for char in expression:  if char.isdigit():  stack.append(int(char))  else:  b = stack.pop()  a = stack.pop()  result = selectedOperator(char, b, a)  stack.append(result)  return stack[0]    def main():  infixExpr = input("Enter a mathematical expression: ")  postfixExpr = infixToPostfix(infixExpr)  print("Reverse Polish Entry:", postfixExpr)  result = evaluatePostfix(postfixExpr)  print("Result:", result)    if \_\_name\_\_ == "\_\_main\_\_":  main() |

1. Опис коду:

* precedence(op): Ця функція приймає один параметр op (оператор) і повертає його пріоритет:

^ (степінь) має пріоритет 3.

\* і / (множення та ділення) мають пріоритет 2.

+ і - (додавання та віднімання) мають пріоритет 1.

Якщо оператор не розпізнано, функція повертає 0.

Це потрібно для того, щоб при перетворенні виразу на постфіксну форму коректно враховувати пріоритет операторів.

* selectedOperator(op, b, a): Функція приймає оператор op і два числа a і b, виконуючи відповідну математичну операцію:

Для оператора + виконується додавання.

Для оператора - — віднімання.

Для оператора \* — множення.

Для оператора / — ділення.

Для оператора ^ — піднесення до степеня.

* infixToPostfix(expression): Ця функція перетворює інфіксний вираз (наприклад, 3 + 4 \* 2) на постфіксну форму (наприклад, 3 4 2 \* +):

Використовуються два основних елементи: стек (stack) для зберігання операторів і список (output) для результату.

Програма проходить по кожному символу виразу:

Якщо символ — це цифра, він додається в результат.

Якщо символ — це відкриваюча дужка (, він додається в стек.

Якщо символ — це закриваюча дужка ), то програма виводить всі оператори з стека до відкриваючої дужки.

Якщо символ — це оператор, то програма перевіряє пріоритет оператора в стеки і додає в результат операції, що мають вищий або рівний пріоритет.

Після того як всі символи оброблено, решта операторів у стека виводяться в результат.

* evaluatePostfix(expression): Ця функція обчислює значення постфіксного виразу:

Програма проходить по кожному символу виразу:

Якщо це цифра, вона додається в стек.

Якщо це оператор, то два останніх числа з стека витягуються і виконуються з ними відповідні операції.

В кінці в стеке залишається тільки одне число — це результат обчислення.

* main(): Головна функція програми, яка виконує:

Запит користувача на введення математичного виразу.

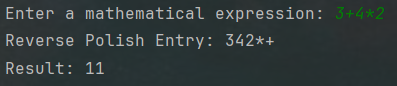
Перетворення введеного виразу в постфіксну нотацію.

Виведення постфіксного виразу.

Обчислення результату постфіксного виразу і виведення результату.

1. Приклад виконання:

Користувач вводить вираз: 3 + 4 \* 2. Програма перетворює цей вираз на постфіксну форму: 3 4 2 \* +. Програма обчислює результат постфіксного виразу: 11.



У функції infixToPostfix вираз 3 + 4 \* 2 спочатку перетворюється в 3 4 2 \* +. Оператор множення має вищий пріоритет, тому спочатку обчислюється 4 \* 2 = 8, і потім додається до 3, даючи результат 11.

Посилання на GitHub: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub>

**Висновки:** Розроблена програма успішно перетворює математичні вирази в зворотний польський запис і обчислює результат, враховуючи пріоритет операторів і правильний порядок виконання операцій.