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.

**Теоретичні відомості**

**Зворотний польський запис**

Звичною формою запису виразів є інфіксна, коли знак бінарної операції записують між позначеннями операндів цієї операції, наприклад, a + b. Розглянемо запис знаків операцій після позначень операндів, тобто постфіксний запис, наприклад, a b +. Такий запис має також назву зворотного польського, бо його запропонував польський логік Ян Лукасевич. Далі словосполучення: «зворотний польський запис» позначатимемо ЗПЗ. Позначення для функції традиційно записують перед аргументами. Природно такий запис назвати префіксним. При описі ЗПЗ переважно обмежуються перетворенням інфіксного запису у ЗПЗ.

Зворотний польський запис має чудові властивості, які перетворюють її на ідеальну проміжну ланку при трансляції коду програми.

Обчислення виразу, записаного в зворотному польському записі, можна проводити шляхом однократного перегляду ЗПЗ.

Зворотний польський запис виразу з арифметичними діями та піднесенням до степеня можна отримати, дотримуючись алгоритму, запропонованого Дейкстpою. Алгоритм отримав назву «сортувальна станція», за подібність його операцій із тим, що відбувається на залізничних сортувальних станціях. Як і алгоритм обчислення ЗПЗ, алгоритм сортувальної станції ґрунтується на стеку. У перетворенні беруть участь дві текстові змінні: вхідний і вихідний рядки. У процесі перетворення використовується стек, що зберігає ще не додані до вихідного рядка операції. Перетворювальна програма читає вхідний рядок послідовно символ за символом (символ – це не обов'язково буква), виконує на кожному кроці деякі дії залежно від того, який символ було прочитано.

**Алгоритм**

У випадку, коли є символи для обробки необхідно зчитати символ.

Якщо символ є числом або додаємо його до вихідного рядка.

Якщо символ є дужкою, поміщаємо його в стек.

Якщо символ є дужкою, що закривається то доки верхнім елементом стека не стане відкриваюча дужка, виштовхуємо елементи зі стека у вихідний рядок. При цьому дужка, що відкриває, видаляється зі стека, але у вихідний рядок не додається. Якщо стек закінчився раніше, ніж ми зустріли дужку, це означає, що у виразі або неправильно поставлений роздільник, або не узгоджені дужки.

Якщо символ є бінарною операцією та операція на вершині стеку має більший або такий самий пріоритет, то необхідно “виштовхнути” верхній елемент до вихідного рядка. Помістити операцію в стек.

Коли вхідний рядок закінчився, виштовхуємо всі символи зі стека у вихідний рядок

**Пріоритетність операцій:**

Найвищий – вираз в дужках.

Високий – піднесення до степеня.

Середній – множення або ділення.

Низький – додавання або віднімання.

**Приклад:**

|  |
| --- |
| Вхід: 3 + 4 \* 2 / (1 - 5) ^ 2  Читаємо «3»  Додаємо «3» до вихідного рядка  Вихід: 3  Читаємо «+»  Кладемо «+» у стек  Вихід: 3  Стек: +  Читаємо «4»  Додамо «4» до вихідного рядка  Вихід: 3 4  Стек: +  Читаємо «\*»  Кладемо «\*» у стек  Вихід: 3 4  Стек: + \*  Читаємо «2»  Додамо «2» до вихідного рядка  Вихід: 3 4 2  Стек: + \*  Читаємо «/»  Виштовхуємо «\*» зі стека у вихідний рядок, кладемо «/» у стек  Вихід: 3 4 2 \*  Стек: +/  Читаємо «(»  Кладемо «(» у стек  Вихід: 3 4 2 \*  Стек: + / (  Читаємо «1»  Додамо «1» до вихідного рядка  Вихід: 3 4 2 \* 1  Стек: + / (  Читаємо «−»  Кладемо «−» у стек  Вихід: 3 4 2 \* 1  Стек: + / ( −  Читаємо «5»  Додамо «5» до вихідного рядка  Вихід: 3 4 2 \* 1 5  Стек: + / (-  Читаємо «)»  Виштовхуємо «−» зі стека у вихідний рядок, виштовхуємо «(»  Вихід: 3 4 2 \* 1 5 −  Стек: +/  Читаємо «^»  Кладемо «^» у стек  Вихід: 3 4 2 \* 1 5 −  Стек: +/^  Читаємо «2»  Додамо «2» до вихідного рядка  Вихід: 3 4 2 \* 1 5 − 2  Стек: +/^  Кінець виразу  Виштовхуємо всі елементи зі стека в рядок  Вихід: **3 4 2 \* 1 5 − 2 ^ / +** |

**Обчислення виразу**

Використовуючи алгоритм ЗПЗ математичний вираз 3 + 4 \* 2 / (1 - 5) ^ 2 був записаний у вигляді 3 4 2 \* 1 5 − 2 ^ / +

Обчислення проводиться зліва направо. Якщо в запису зустрічається число, то число поміщається в стек. Якщо в запису зустрічається оператор, то він застосовується до двох верхніх елементів стеку які виштовхуються із стеку, а результат виконання поміщається в стек.

Запис інтерпретується як зазначено у наведеній нижче таблиці (зазначено стан стека після виконання операції, вершина стека виділена червоним кольором)

|  |  |  |
| --- | --- | --- |
| Символ | Дія | Стек |
| 3 | помістити в стек | 3 |
| 4 | помістити в стек | 3 4 |
| 2 | помістити в стек | 3 4 2 |
| \* | множення | 3 8 |
| 1 | помістити в стек | 3 8 1 |
| 5 | помістити в стек | 3 8 1 5 |
| - | віднімання | 3 8 -4 |
| 2 | помістити в стек | 3 8 -4 2 |
| ^ | піднесення до степеню | 3 8 16 |
| / | ділення | 3 0.5 |
| + | додавання | 3.5 |

Результат **3.5**, в кінці обчислень знаходиться на вершині стека.

**Хід роботи:**

Розробив програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

Код програми:

mathSymb = ('-', '\*', '+', '^', '/')

numbers = ('0','1','2','3','4','5','6','7','8','9')

def Response(input):

    stack = []

    output = []

    for token in input:

        if Ncheck(token):

            output.append(token)

            continue

        if token == '(':

            stack.append(token)

            continue

        if token == ')':

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

                output.append(stack.pop())

            stack.pop()

            continue

        if token in mathSymb:

            opPrior = prior(token)

            while stack and stack[-1] in mathSymb and prior(stack[-1]) >= opPrior:

                output.append(stack.pop())

            stack.append(token)

    while stack:

        output.append(stack.pop())

    return output

def Ncheck(numm):

    return all(char in numbers for char in numm)

def mat(respons):

    stack = []

    for token in respons:

        if Ncheck(token):

            stack.append(int(token))

        else:

            operandNumTwo = stack.pop()

            operandNumOne = stack.pop()

            result = 0

            if token == '+':

                result = operandNumOne + operandNumTwo

            elif token == '-':

                result = operandNumOne - operandNumTwo

            elif token == '\*':

                result = operandNumOne \* operandNumTwo

            elif token == '/':

                result = operandNumOne / operandNumTwo

            elif token == '^':

                result = operandNumOne \*\* operandNumTwo

            stack.append(result)

    return stack

def prior(operation):

    if operation in ('+', '-'):

        return 0

    elif operation in ('\*','/'):

        return 1

    elif operation == '^':

        return 2

if \_\_name\_\_ == '\_\_main\_\_':

    input = "5-3^2+(7-2)^4"

    respons = Response(input)

    print("Reverse: ", respons)

    print("Result: ", mat(respons))

**Висновок:** Отже, в ході виконання даної роботи було ретельно вивчено теоретичні аспекти зворотного польського запису (ЗПЗ) та його використання для обчислення математичних виразів. Згідно з метою роботи, була розроблена програма, яка приймає на вхід математичний вираз у вигляді ЗПЗ з довільним набором операндів, операторів та дужок, і виводить обчислений результат.