|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт Информационных Технологий |
| Кафедра Вычислительной техники |

|  |  |
| --- | --- |
| **ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ**  **№1-2**  «Преобразование алгебраического выражения в обратную польскую запись на Python» | |
| **по дисциплине** |  |
| **«**Теория формальных языков**»** | |
| Выполнил студент группы ИКБО-15-22 | *Оганнисян Г.А.* |
| Принял старший преподаватель | *Боронников А.С.* |

|  |  |
| --- | --- |
| Практическая работа выполнена | « » 2023 г. |
| «Зачтено» | « » 2023 г. |

Москва 2021

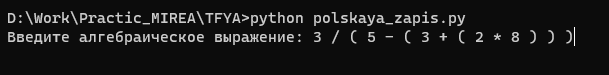


Рисунок 1 – Ввод примера в программу

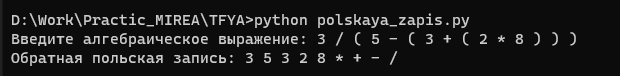


Рисунок 2 – Вывод программы

Этот код выполняет преобразование инфиксного алгебраического выражения (обычного математического выражения с операторами вроде +, -, \*, /) в обратную польскую запись. Обратная польская запись (или постфиксная запись) - это форма записи математических выражений, в которой операторы идут после своих операндов.

1. **is\_operator(token):**
   * Проверяет, является ли токен оператором.
2. **precedence(token):**
   * Определяет приоритет оператора.
3. **infix\_to\_postfix(expression):**
   * Преобразует инфиксное выражение в постфиксное.
4. **input\_expression = input("Введите алгебраическое выражение: "):**
   * Запрашивает у пользователя ввод алгебраического выражения.
5. **postfix\_expression = infix\_to\_postfix(input\_expression):**
   * Вызывает функцию **infix\_to\_postfix** для преобразования введенного выражения в постфиксное.
6. **print("Обратная польская запись:", postfix\_expression):**
   * Выводит результат преобразования в постфиксное выражение.

Листинг кода

*# Функция для проверки, является ли токен оператором*

def is\_operator(token):

    return token in "+-\*/"

*# Функция для определения приоритета оператора*

def precedence(token):

    if token in "+-":

        return 1

    if token in "\*/":

        return 2

    return 0  *# Приоритет 0 для чисел и скобок*

*# Функция для преобразования инфиксного выражения в постфиксное*

def infix\_to\_postfix(expression):

    output = []  *# Список для хранения выходного постфиксного выражения*

    stack = []   *# Стек для временного хранения операторов и скобок*

*# Разбиваем входное выражение на токены (числа, операторы и скобки)*

    for token in expression.split():

        if token.isdigit():

            output.append(token)  *# Если токен - число, добавляем его в выход*

        elif token == "(":

            stack.append(token)   *# Если токен - "(", помещаем его в стек*

        elif token == ")":

*# Если токен - ")", извлекаем операторы из стека и добавляем их в выход,*

*# пока не встретим "("*

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

                output.append(stack.pop())

            stack.pop()  *# Удаляем "(" из стека*

        elif is\_operator(token):

*# Если токен - оператор, извлекаем операторы из стека и добавляем их в выход,*

*# пока приоритет текущего оператора не станет меньше операторов в стеке*

            while stack and is\_operator(stack[-1]) and precedence(stack[-1]) >= precedence(token):

                output.append(stack.pop())

            stack.append(token)  *# Помещаем текущий оператор в стек*

*# После обработки всех токенов, извлекаем оставшиеся операторы из стека и добавляем их в выход*

    while stack:

        output.append(stack.pop())

*# Возвращаем постфиксное выражение, объединив элементы списка через пробел*

    return " ".join(output)

*# Получаем входное алгебраическое выражение от пользователя*

input\_expression = input("Введите алгебраическое выражение: ")

*# Вызываем функцию для преобразования и выводим результат*

postfix\_expression = infix\_to\_postfix(input\_expression)

print("Обратная польская запись:", postfix\_expression)

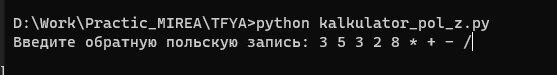


Рисунок 3 – Ввод данных для 2 программы

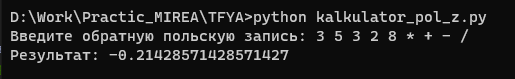


Рисунок 4 – Вывод данных 2 программы

Этот код - реализация программы для вычисления математических выражений, представленных в обратной польской записи (RPN). Обратная польская запись - это форма записи математических выражений, где операторы следуют после своих операндов. Пример ввода выражения: "3 5 3 2 8 \* + - /"

Программа использует стек для хранения операндов и операторов. Она также отслеживает инфиксную запись выражения для каждого шага вычисления. В случае неверного выражения (например, недостаточно операндов) программа выводит сообщение об ошибке.

Для каждого токена (числа или оператора) в выражении, программа либо помещает число в стек операндов, либо выполняет операцию с двумя верхними элементами стека и помещает результат обратно в стек. В процессе выполнения выражения, программа также строит инфиксное выражение для каждого шага.

В конце выполнения, если на стеке остался только один элемент, то это и есть результат выражения. В противном случае программа выводит сообщение об ошибке.

Пример:

* Ввод: "3 5 3 2 8 \* + - /""
* Вывод:
  + Результат: -0.21428571428571427

Листинг кода

def evaluate\_postfix(expression):

    stack = []  *# Стек для хранения операндов*

*# Разбиваем входное выражение на токены*

    tokens = expression.split()

    for token in tokens:

        if token.isdigit():

*# Если токен - число, помещаем его в стек*

            stack.append(int(token))

        else:

*# Если токен - оператор, извлекаем два операнда из стека*

            operand2 = stack.pop()

            operand1 = stack.pop()

*# Выполняем операцию, соответствующую оператору*

            if token == "+":

                result = operand1 + operand2

            elif token == "-":

                result = operand1 - operand2

            elif token == "\*":

                result = operand1 \* operand2

            elif token == "/":

                result = operand1 / operand2

*# Результат операции помещаем обратно в стек*

            stack.append(result)

*# В конце, стек должен содержать только один элемент, который и будет результатом*

    return stack[0]

*# Получаем входное выражение от пользователя*

input\_expression = input("Введите обратную польскую запись: ")

*# Вызываем функцию для вычисления и выводим результат*

result = evaluate\_postfix(input\_expression)

print("Результат:", result)