|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Языки Программирования»  Вариант 15 | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 1 курса  Черанёва Н.А.  «\_\_\_» \_\_\_\_\_\_\_\_2023 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 4](#_Toc153130029)

[Код программы 4](#_Toc153130030)

# Постановка задачи

Используя структуру стека подсчитать значение арифметического выражения, записанного в обратной польской записи (постфиксной записи) при условии, что используются знаки операций +, –, \*, / и операнды являются вещественными положительными числами. Во вводимой пользователем строке числа и знаки операций разделяются одним пробелом.

# Алгоритм решения

Решение задачи вычисления значения выражения по постфиксной нотации реализовано через класс Stack, который представляет собой структуру данных "стек". Стек — это абстрактный тип данных, последний добавленный элемент будет первым удаленным. Выбор стека как структуры данных обусловлен его способностью эффективно решать задачу без необходимости преобразования выражения в инфиксную или префиксную нотацию. В процессе выполнения алгоритма:

Каждый токен выражения обрабатывается по порядку.

Если токен является числом, он помещается в стек.

Если токен является оператором, из стека извлекаются два операнда, выполняется соответствующая операция, а результат снова помещается в стек.

После обработки всех токенов в стеке должно остаться ровно одно число, которое и будет результатом выражения.

Типы данных и структуры используются следующим образом:

Список (list) используется для реализации стека. Список позволяет легко добавлять (append) и удалять (pop) элементы с конца.

Числа (float) используются для хранения операндов и результата выражения.

Строки (str) используются для представления каждого токена выражения. Это обеспечивает гибкость при работе с различными типами данных, включая числа и операторы.

Методы класса Stack:

\_\_init\_\_: Инициализирует пустой стек.

push: Добавляет элемент в конец стека.

pop: Удаляет последний элемент из стека. Если стек пуст, генерируется исключение.

is\_empty: Проверяет, является ли стек пустым.

Функция calculate\_postfix принимает строку, представляющую постфиксное выражение, и возвращает результат его вычисления. Она использует экземпляр класса Stack для временного хранения операндов и промежуточных результатов.

# Тестирование

Тест на ввод обычных данных.





Тесты на ввод неправильных данных.









# Код программы

class Stack: #Стек данных

    def \_\_init\_\_(self): # Пустой список для хранения элементов стека

        self.items = []

    def push(self, item): # Метод, добавляющий элемент в конец списка

        self.items.append(item)

    def pop(self): # Метод, удаляющий последний элемент из списка

        if not self.is\_empty():

            return self.items.pop()

        else:

            raise ValueError("Неверно введены данные, попробуйте ещё раз.")

    def is\_empty(self): # Метод, проверяющий, является ли стек пустым

        return len(self.items) == 0

def calculate\_postfix(expression): # Вычисляет значение в соответствии с постфиксной нотацией

    stack = Stack()

    tokens = expression.split()

    if not expression.strip():

        raise ValueError('Вы не ввели выражение.')

    for token in tokens:

        try:

            if token.replace('.', '', 1).isdigit(): # Помещение нецелочисленного числа в стек

                stack.push(float(token))

            else:

                operand2 = stack.pop()

                operand1 = stack.pop()

                if operand2 == 0 and token == '/':

                    raise ValueError('Невозможно деление на 0.')

                if token == '+':

                    result = operand1 + operand2

                elif token == '-':

                    result = operand1 - operand2

                elif token == '\*':

                    result = operand1 \* operand2

                elif token == '/':

                    result = operand1 / operand2

                else:

                    raise ValueError(f'Ошибка: {token}')

                stack.push(result)

        except IndexError:

            raise ValueError('Error')

    if len(stack.items) > 1: # Проверка, что в стеке остался только один элемент

        raise ValueError('Формат выражения не верен.')

    return stack.pop()

while True:

    try:

        expression = input("Введите выражение (дробное число вводите с точкой): ")

        result = calculate\_postfix(expression)

        print(result)

        break

    except Exception as e:

        print(e)