ТИТУЛЬНИК

**Задача:** реализовать преобразование выражений в обратную польскую запись, а также реализовать алгоритм вычисления выражения, записанного в обратной польской записи.

Листинг 1 —Преобразование выражения в обратную польскую запись

1. def expressionToPostfixForm(input\_string: str) -> str:  
    *"""Переводит в постфиксную запись выражение,  
    поддерживаются только операторы ()+-/\*"""* input\_string = input\_string.replace(" ", '')  
    result = []  
    stack = []  
     
    priority\_dictionary = {"+": 2, "-": 2, "\*": 3, "/": 3, "(": 1, ")": 1}  
    variable = [] # для хранения многосимвольных переменных и чисел  
    for current\_symbol in input\_string: # перебираем текущий символ  
    # добавим условие для работы многосимвольными переменными и числами  
    if current\_symbol not in priority\_dictionary:  
    variable.append(current\_symbol)  
    else: # символ - знак операции  
    if variable:  
    result.append(''.join(variable))  
    variable = []  
    if current\_symbol == '(': # пункт в)  
    stack.append(current\_symbol)  
    elif current\_symbol == ")": # пункт г)  
    while stack and stack[-1] != "(":  
    result.append(stack.pop(-1))  
    stack.pop() # уничтожаем (  
    else:  
    while stack and priority\_dictionary[stack[-1]] >= priority\_dictionary[current\_symbol]: # пункт б)  
    result.append(stack.pop(-1))  
    if not stack or priority\_dictionary[stack[-1]] < priority\_dictionary[current\_symbol]: # пункт а)  
    stack.append(current\_symbol)  
    if variable:  
    result.append(''.join(variable))  
    while stack:  
    result.append(stack.pop(-1))  
    return ' '.join(result)

Листинг 2 —Вычисления выражения, записанного в ОПЗ

1. def calculatePostfixExpression(input\_string: str) -> int:  
    *"""Алгоритм вычисления выражения, записанного в обратной польской записи  
    Не поддерживает переменные"""* operations = {"+", "-", "/", "\*"} # множество операций  
    stack\_of\_numbers = []  
    number = [] # для хранения многосимвольного числа  
    for current\_symbol in input\_string:  
    # Если текущий символ - операция или пробел и если набралось число  
    if (current\_symbol == " " or current\_symbol in operations) and number:  
    stack\_of\_numbers.append(int(''.join(number))) # добавляем число в стек  
    number = [] # освобождаем массив  
     
    if current\_symbol in operations: # если встречаем операцию, то достаем 2 числа и производим операцию  
    second\_number = stack\_of\_numbers.pop(-1)  
    first\_number = stack\_of\_numbers.pop(-1)  
    res = first\_number + second\_number # для сложения  
    if current\_symbol == "-": # для вычитания  
    res = first\_number - second\_number  
    elif current\_symbol == "/": # для деления  
    res = first\_number / second\_number  
    elif current\_symbol == "\*": # для умножения  
    res = first\_number \* second\_number  
    stack\_of\_numbers.append(res)  
    elif current\_symbol != " ": # если символ - число  
    number.append(current\_symbol)  
    return stack\_of\_numbers[-1]

Таблица 1 - тесты для программы преобразования выражений в ОПЗ

|  |  |  |
| --- | --- | --- |
| *Входные данные* | *Выходные данные* | *Ожидаемые данные* |
| 1+2 | 1 2 + | 1 2 + |
| (6+9-5) / (8+1\*2)+7 | 6 9 + 5 - 8 1 2 \* + / 7 + | 6 9 + 5 - 8 1 2 \* + / 7 + |
| (6+91-532) / (81+21\*2)+75 | 6 91 + 532 - 81 21 2 \* + / 75 + | 6 91 + 532 - 81 21 2 \* + / 75 + |
| first\_number+b/a+54\*33/(12) | first\_number b a / + 54 33 \* 12 / + | first\_number b a / + 54 33 \* 12 / + |

Таблица 2 - тесты для программы вычисления выражения в ОПЗ

|  |  |  |
| --- | --- | --- |
| *Входные данные* | *Выходные данные* | *Ожидаемые данные* |
| 1 2 + | 3 | 3 |
| 6 9 + 5 - 8 1 2 \* + / 7 + | 8 | 8 |
| 60 3 / 542 + 23 43 \* - 75 + | -352 | -352 |
| 33 11 + 44 / 0 - 213 \* 22 + 43 2222 - \* 512 1000 \* + | -65 | -65 |

**Вывод:** были успешно реализованы и протестированы программы по преобразованию выражений в обратную польскую запись, а вычислению выражения, записанного в обратной польской записи, не только с односимвольными переменными и числами, но и многосимвольными.