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 и №2

«Преобразование алгебраического выражения в обратную польскую запись на C++ и вычисление его значения при помощи обратной польской записи»

по дисциплине
«Теория формальных языков»

Выполнил студент группы ИКБО-04-22 *Яковлев П.А.*

Принял ассистент *Боронников А.С.*

Практическая работа выполнена «__»_____2023 г.

«Зачтено» «__»_____2023 г.

Москва 2023

1 ПОСТАНОВКА ЗАДАЧИ

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

Пример работы первой программы:

ввод: (10 + 2) * 2
вывод: 10 2 + 2 *

Пример работы второй программы:

ввод: 10 2 + 2 *
вывод: 24

2 РЕЛИЗАЦИЯ АЛГОРИТМОВ

Листинг 1 —Преобразование выражения в обратную польскую запись

```
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

```
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]
```

3 ТЕСТИРОВАНИЕ

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

4 ЗАКЛЮЧЕНИЕ

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.