

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра Вычислительной техники (ВТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1-2**

**по дисциплине**

**«Теория формальных языков»**

Тема. Преобразование выражений в обратную польскую запись. Вычисление значения выражения в обратной польской записи.

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-04-22 |  | Основин А.И. |
| Принял преподаватель |  | Боронников А.С. |

Москва 2023

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc153123324)

[1.1 Задание 1 3](#_Toc153123325)

[1.2 Задание 2 3](#_Toc153123326)

[1.3 Язык программирования 3](#_Toc153123327)

[2 РЕАЛИЗАЦИЯ ПРОГРАММЫ 4](#_Toc153123328)

[3 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 6](#_Toc153123329)

[3.1 Тестирование функции перевода выражения в ОПЗ 6](#_Toc153123330)

[3.2 Тестирование функции вычисления выражения в ОПЗ 6](#_Toc153123331)

[4 ВЫВОД 7](#_Toc153123332)

# ПОСТАНОВКА ЗАДАЧИ

## Задание 1

На любом языке программирования реализовать преобразование простого алгебраического выражения в обратную польскую запись.

Пример работы программы:

ввод: (10 + 2) \* 2

вывод: 10 2 + 2 \*

## Задание 2

На любом языке программирования реализовать простой калькулятор алгебраических выражений в обратной польской записи с целыми числами. Поддерживаемые операции: сложение, вычитание, умножение, деление.

Пример работы программы:

ввод: 10 2 + 2 \*

вывод: 24

## Язык программирования

Для реализации поставленных задач был выбран язык программирования Python.

# РЕАЛИЗАЦИЯ ПРОГРАММЫ

В Листинге 1 представлена реализация программы для преобразования выражения в обратную польскую запись и вычисления значения выражения в обратной польской записи.

*Листинг 1 – Код программы на языке программирования Python*

def transform(input: list, params: dict = {}) -> list:  
 signs = {"\*": 3, "/": 3, "+": 2, "-": 2, "(": 1}  
 stack = list()  
 output = list()  
  
 for symbol in input:  
 if symbol == " ":  
 continue  
  
 symbol = symbol.strip()  
 if symbol.isdigit() or symbol in params.keys():  
 output.append(symbol)  
  
 elif symbol == "(":  
 stack.append(symbol)  
  
 elif symbol == ")":  
 temp = stack.pop()  
 while temp != "(":  
 output.append(temp)  
 temp = stack.pop()  
  
 elif symbol in signs.keys():  
 while len(stack) != 0 and signs[stack[-1]] >= signs[symbol]:  
 output.append(stack.pop())  
  
 stack.append(symbol)  
  
 else:  
 print(f"Can't parse symbol while transforming: {symbol}.")  
 exit(0)  
  
 while len(stack) != 0:  
 output.append(stack.pop())  
  
 return output  
  
  
def count\_res(input: list, params: dict = {}) -> float:  
 stack = list()  
 for symbol in input:  
 if symbol.isdigit() or symbol in params.keys():  
 stack.append(symbol)  
  
 elif symbol in ["\*", "/", "+", "-"]:  
 a = stack.pop()  
 b = stack.pop()  
  
 if a.isalpha():  
 a = params[a]  
 if b.isalpha():  
 b = params[b]  
  
 stack.append(str(eval(b + symbol + a)))  
  
 else:  
 print(f"Can't parse symbol while counting: {symbol}.")  
 exit(0)  
  
 return float(stack.pop())  
  
  
def prepare(string: str) -> list:  
 substr = ""  
 in\_list = list()  
  
 for symbol in string:  
 if symbol in ["\*", "/", "+", "-", "(", ")"]:  
 if len(substr) != 0:  
 in\_list.append(substr)  
  
 in\_list.append(symbol)  
 substr = ""  
  
 else:  
 substr += symbol  
  
 if len(substr) != 0:  
 in\_list.append(substr)  
  
 return in\_list  
  
  
def main():  
 expression = input('Input your equation: ')  
 expression = prepare(expression)  
 print('enter the values of all variables that participate in the expression (then enter "\\)')  
  
 params = {}  
 # params = {'a': '2', 'b': '3'}  
 key = input('Enter variable: ')  
 while key != "\\":  
 params[key] = input('Enter value: ')  
 key = input('Enter variable: ')  
  
 result = transform(expression, params)  
 print(' '.join(result))  
 print(count\_res(result, params))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

## Тестирование функции перевода выражения в ОПЗ

Сначала необходимо протестировать функцию перевода выражения в ОПЗ. В Таблице 1 приведены примеры ввода данных в программу, вывод разработанной программы и ожидаемый результат работы программы. Как видно из таблицы, ожидаемый результат совпадает с выводом программы, следовательно, разработанная функция перевода выражения в ОПЗ работает корректно.

*Таблица 1 – Тестирование программы перевода выражения в ОПЗ*

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод программы | Правильный ответ |
| ((5+3)-4)/2\*3 | 53+4-2/3\* | 53+4-2/3\* |
| (6-7)/10 | 67-10/ | 67-10/ |

## Тестирование функции вычисления выражения в ОПЗ

Далее необходимо протестировать функцию вычисления выражения в ОПЗ. В Таблице 2 приведены примеры ввода данных в программу, вывод разработанной программы и ожидаемый результат работы программы. Как видно из таблицы, ожидаемый результат совпадает с выводом программы, следовательно, разработанная функция вычисления выражения в ОПЗ работает корректно.

*Таблица 2 – Тестирование программы вычисления выражения в ОПЗ*

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод программы | Правильный ответ |
| 10 2 2 \* + | 14 | 14 |
| 6 8 2 / - | 2 | 2 |

# ВЫВОД

В ходе выполнения данной практической работы был изучен алгоритм перевода простейших алгебраических выражений в обратную польскую нотацию, также был изучен алгоритм обработки простейших алгебраических выражений в обратном польской записи. В результате проделанной работы была разработана программа, которая полностью соответствует приведенному алгоритму и отвечает поставленным требованиям.