Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы № 2

по дисциплине «Методы разработки трансляторов»

Тема: «Перевод исходной программы в

обратную польскую запись»

Выполнил: ст. гр. 39/1

Козлов Э.Д.

Проверил: Преподаватель

Усов П.Е

Краснодар

2025

Ход работы:

Постановка задачи: Требуется разработать программу для перевода текста программы в ОПЗ.

**Основные понятия**

**Обратная польская запись (ОПЗ)** – способ записи выражений, при котором операнды записываются перед операторами. Например, стандартное выражение

a + b \* (a - b)

преобразуется в ОПЗ как

a b a b - \* +.

**Преимущества ОПЗ** заключаются в том, что порядок вычислений определяется структурой записи, а скобки становятся избыточными. Это упрощает алгоритмы вычисления и генерацию машинного кода.

**Алгоритм Дейкстры**

Преобразование инфиксного выражения в ОПЗ основано на алгоритме Дейкстры, который реализует так называемую «сортировочную станцию». Основные этапы алгоритма:

1. **Обработка операндов:** При встрече операндов (идентификаторов, чисел, констант) они немедленно добавляются в выходную строку.

2. **Обработка операторов:**

• Если стек пуст или оператор, находящийся на вершине, имеет меньший приоритет, текущий оператор помещается в стек.

• Если приоритет оператора на вершине стека выше или равен приоритету текущего оператора, производится извлечение операторов из стека в выходную строку до тех пор, пока это условие выполняется, после чего текущий оператор заносится в стек.

3. **Обработка скобок:**

• При встрече открывающей скобки она всегда помещается в стек.

• При встрече закрывающей скобки из стека извлекаются операторы до первой открывающей скобки, которая удаляется. Скобки при этом не попадают в итоговую запись.

4. **Выталкивание оставшихся операторов:** После завершения прохода по выражению оставшиеся операторы извлекаются из стека и добавляются к выходной записи.

В предыдущей лабораторной работе была разработана программа, которая выполняет лексический анализ исходного кода и формирует файл в формате .txt, содержащий лексемы (каждая лексема представлена парой «тип-значение»). На основе этого файла была создана программа для перевода инфиксных выражений в обратную польскую запись (ОПЗ) с использованием алгоритма Дейкстры. Программа читает файл с лексемами, обрабатывает операторы, операнды и скобки с учетом приоритетов, а затем записывает результат – лексемы, представленные в виде ОПЗ – в новый файл .txt.

Проведем тесты программы.

Исходный код на ЯП С:

if ((a - b) > 8) {  
if ((a + b) == 10){  
 c = 3; }  
 }

Ожидаемый результат:

a b - 8 > УПЛ

a b + 10 == УПЛ

c 3 - =

Фактический результат:



Вывод: В ходе работы была разработана программа перевода в ОПЗ. Разработанная программа успешно преобразует файл с лексемами в обратную польскую запись, корректно обрабатывая операторы и операнды.

Листинг программы

import re  
  
*# Приоритет операторов*OPERATOR\_PRIORITY = {  
 "=": 1,  
 "||": 2, "&&": 3,  
 "<": 4, "<=":4, ">": 4, ">=":4, "==": 4, "!=": 4,  
"+": 5, "-": 5,  
"\*": 6, "/": 6, "%": 6,  
"(": 0, ")": 0  
}  
  
*# Функция перевода в ОПЗ (алгоритм Дейкстры)*def to\_opz(expression\_tokens):  
 output = []  
 stack = []  
  
 for token in expression\_tokens:  
 if token.isalnum(): *# Переменная или число* output.append(token)  
 elif token == "(": *# Открывающая скобка* stack.append(token)  
 elif token == ")": *# Закрывающая скобка* while stack and stack[-1] != "(":  
 output.append(stack.pop()) *# Выталкиваем всё до `(`* if stack and stack[-1] == "(":  
 stack.pop() *# Убираем `(`, не добавляя в ОПЗ* elif token in OPERATOR\_PRIORITY: *# Оператор* while stack and stack[-1] != "(" and OPERATOR\_PRIORITY[stack[-1]] >= OPERATOR\_PRIORITY[token]:  
 output.append(stack.pop())  
 stack.append(token)  
  
 while stack: *# Выталкиваем остатки из стека* output.append(stack.pop())  
  
 return output  
  
  
*# Функция обработки `if`, учитывая вложенные скобки*def process\_if\_statement(expression):  
 *# Добавляем пробелы вокруг скобок, чтобы `split()` работал правильно* expression = expression.replace("(", " ( ").replace(")", " ) ")  
 tokens = expression.split()  
 opz\_condition = to\_opz(tokens) *# Переводим в ОПЗ* return " ".join(opz\_condition) + " УПЛ"  
  
  
*# Основная функция обработки кода*def convert\_to\_opz\_plain(code\_lines):  
 result = []  
 for line in code\_lines:  
 line = line.strip()  
  
 *# Убираем `{}` перед обработкой* if line in ["{", "}"]:  
 continue  
  
 *# Обрабатываем `if`* if line.startswith("if"):  
 condition = line[line.index("(") + 1: line.rindex(")")]  
 result.append(process\_if\_statement(condition))  
 continue  
  
 *# Обрабатываем присваивание* if "=" in line:  
 var\_name, expr = line.split("=")  
 var\_name = var\_name.strip()  
 expr = expr.strip().rstrip(";")  
  
 *# Разделяем токены правильно (добавляем пробелы)* expr = expr.replace("(", " ( ").replace(")", " ) ")  
 expr\_tokens = expr.split()  
  
 opz\_expr = to\_opz(expr\_tokens)  
 result.append(f"{var\_name} {' '.join(opz\_expr)} =")  
 continue  
  
 result.append(line)  
  
 return result  
  
code\_example = [  
 "if ((a - b) > 8) {",  
 "if ((a + b) == 10){",  
 "c = 3 - 3; }",  
 "}"  
]  
  
opz\_result = convert\_to\_opz\_plain(code\_example)  
  
for line in opz\_result:  
 print(line)