Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Методы трансляции

**ОТЧЕТ**

к лабораторной работе № 2   
на тему «Лексический анализ»

Выполнил:

студент гр. 153504

Михалевич M.П.

Проверил:

Гриценко Н.Ю.

Минск 2024

СОДЕРЖАНИЕ

[1 Цели работы 3](#_Toc158376181)

[2 Краткие теоретические сведения 4](#_Toc158376182)

[3 Результаты выполнения лабораторной работы 5](#_Toc158376183)

[Вывод 6](#_Toc158376185)

[Список использованных источников 7](#_Toc158376186)

[Приложение А (обязательное) Листинг кода 8](#_Toc158376187)

## ЦЕЛИ РАБОТЫ

Целью данной лабораторной работы является разработка лексического анализатора подмножества языка программирования С, определение лексических правил, перевод потока символов в поток лексем (токенов). Также при определении неверной последовательности символов, необходимо обнаружить эту ошибку и выдать сообщение о ней.

## КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

На стадии лексического разбора, также известной как токенизация, процесс компиляции начинается с преобразования исходного кода, написанного на C, в набор лексем – фундаментальных элементов языка, включающих в себя переменные, числа, строки и операторы. Этот процесс критичен для упрощения анализа структуры кода, поскольку он превращает сложный текст программы в упорядоченную серию токенов, готовых к дальнейшему синтаксическому и семантическому анализу.[1]

Использование сложных алгоритмов и структур данных, таких как словари для идентификации ключевых слов и операторов, а также алгоритмы приведения типов для различения литералов, позволяет точно классифицировать каждый элемент кода. Этот этап не только упрощает последующие этапы компиляции, но и выявляет базовые ошибки в коде, такие как неправильное использование символов или некорректные идентификаторы.[2]

В разработке лексического анализатора для языка C использовались следующие подходы и инструменты:

1. Применение Python для анализа исходного кода. Основная логика анализатора реализована на Python, обеспечивая гибкость и мощные возможности для обработки текста.
2. Алгоритмы обработки строк и регулярные выражения. Для извлечения токенов из кода и их классификации использовались специализированные алгоритмы и регулярные выражения, что позволило точно определить структуру исходного текста.
3. Запись результатов в файл: После анализа исходного кода и классификации токенов результаты были сохранены в файл для последующего использования, что упростило процесс проверки и доработки анализатора.
4. Проверка корректности идентификаторов и литералов: В процессе анализа особое внимание уделялось проверке идентификаторов на соответствие правилам именования переменных в C, а также анализу числовых и строковых литералов на корректность.

Примененные инструменты активно способствовали созданию лексического анализа для языка программирования C.

## РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ

## РАБОТЫ

В ходе лабораторной работы был написан лексический анализатор для языка программирования C. Результат работы анализатора показан на рисунке 3.1.

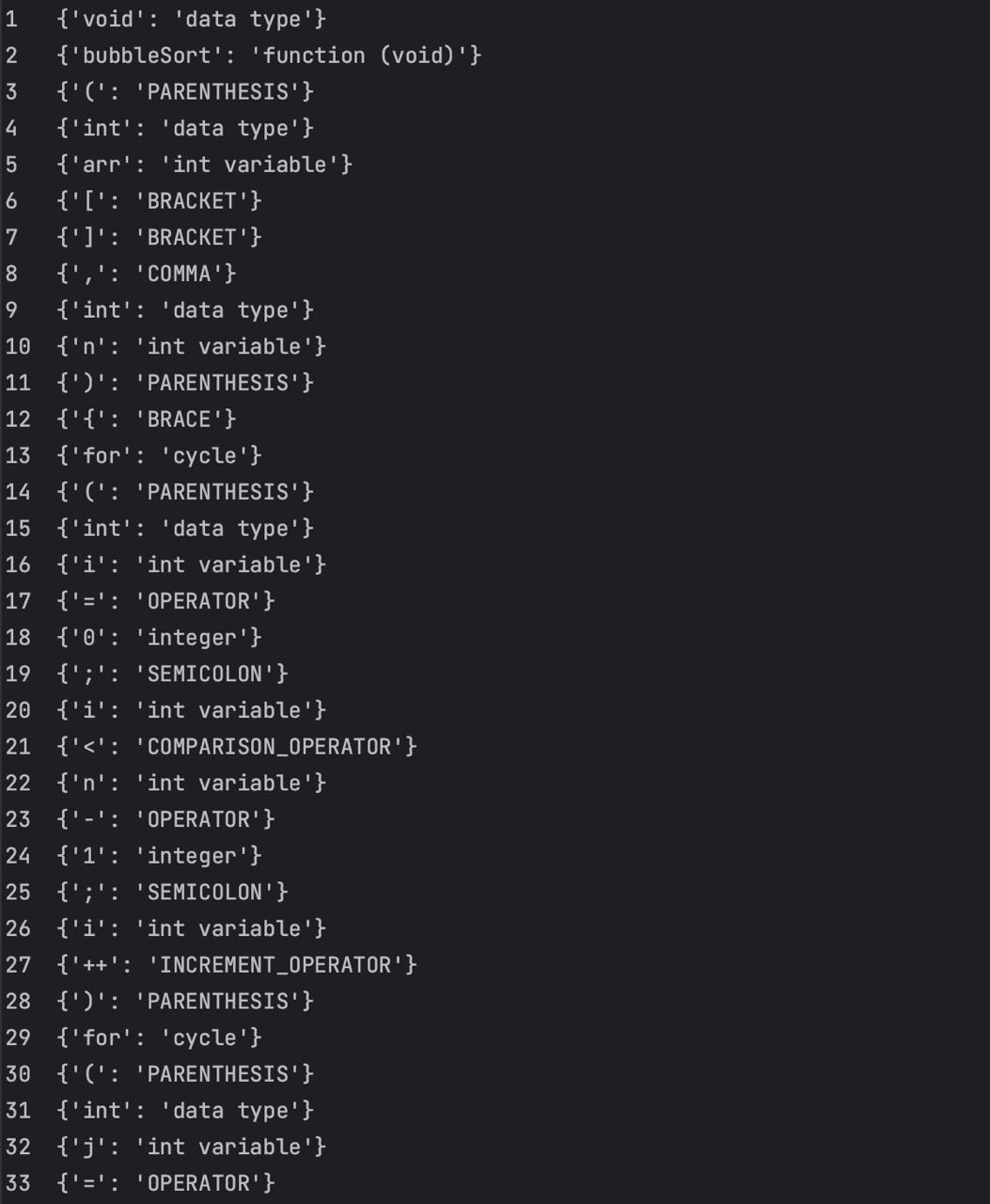


Рисунок 3.1 – Результат работы анализатора

Также программа определяет лексические ошибки. Например, если написать неправильный символ в ключевом слове, программа выведет ошибку и напишет слово, которое возможно имелось в виду. Результат показан на рисунке 3.2.

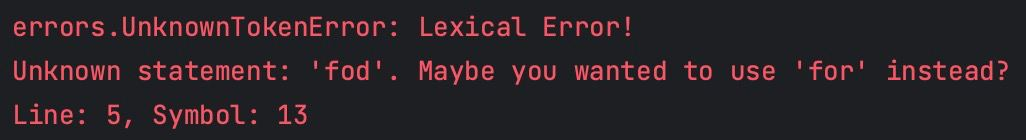


Рисунок 3.2 – Вывод лексической ошибки при использовании некорректного оператора

Если при названии переменной не учитывались правила составления названий для языка C, то анализатор покажет соответствующую ошибку Результат нахождения данной ошибки представлен на рисунке 3.3.

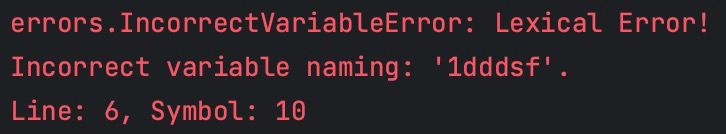


Рисунок 3.3 – Вывод лексической ошибки при неправильном названии переменной

Если допустить ошибку в написании директивы «#include», ответственной за подключение сторонних библиотек, анализатор так же покажет ошибку. Рассматриваемая ошибка представлена на рисунке 3.4.

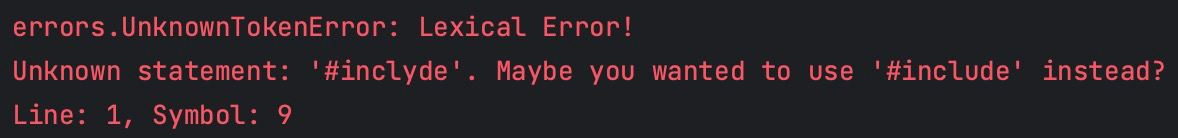


Рисунок 3.4 – Вывод лексической ошибки при неправильном написании директивы препроцессора

При использовании неизвестных для языка программирования C слов, анализатор также выдаст ошибку. Вывод ошибки показа на рисунке 3.5.

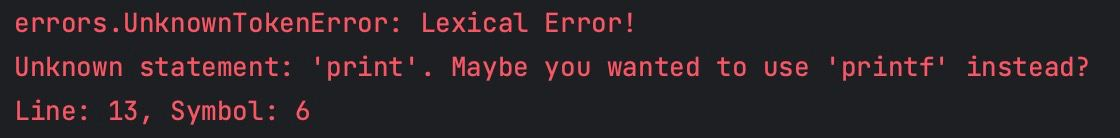


Рисунок 3.5 – Вывод лексической ошибки при использовании неизвестных команд

В итоге, был реализован лексический анализатор, который переводит потока символов в поток токенов, а также определяет лексические ошибки.

## ВЫВОД

В ходе выполнения лабораторной работы был разработан лексический анализатор подмножества языка программирования С, определены лексические правила, переведен поток символов в поток лексем (токенов). Также определены неверные последовательности символов и выдается сообщение об ошибке.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лексический анализ — Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Лексический\_анализ. – Дата доступа: 18.02.2024.

[2] Теория компиляция [Электронный ресурс]. – Режим доступа: https://spravochnick.ru/informatika/teoriya\_kompilyacii\_leksicheskiy\_analizator/. – Дата доступа: 18.02.2024.

## ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Программный код разработанного скрипта

#!/bin/bash

inputFile="text.txt"

outputFile="result.txt"

if [ ! -f "$inputFile" ]; then

echo "Input file not found!"

exit 1

fi

sed -E ':a;N;$!ba;s/(\.\s\*|\?\s\*|!\s\*|^)([a-z])/\1\u\2/g' $inputFile > $outputFile