высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Энгельсский технологический институт

Кафедра Естественные и математические науки

Специальность 09.03.04 Программная инженерия

Отчёт по Лабораторной работе №2по дисциплине “Теория языков программирования и методы трансляции”

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы ПИНЖ-31  очной формы обучения  Баймухамбетов Д.В. |

Энгельс 2023

**2.Вариант № 5**

**Условие:**

1.Получить вариант задания у преподавателя.

2. В соответствии с выданным вариантом выполнить следующее:

2.1. Составит техническое задание (ТЗ) на разработку программы сканера, производящей лексический анализ произвольных текстов в пределах установленного алфавита.

2.2. Согласовать ТЗ с преподавателем.

2.3. Разработать программу-сканер на языках Паскаль, С++ или в интегрированных средах по собственному усмотрению.

2.4. Провести тестирование и отладку программы (предусмотреть все случаи вывода сообщений об ошибках пользователю).

2.5. Составить отчёт по работе и приложить к нему ТЗ.

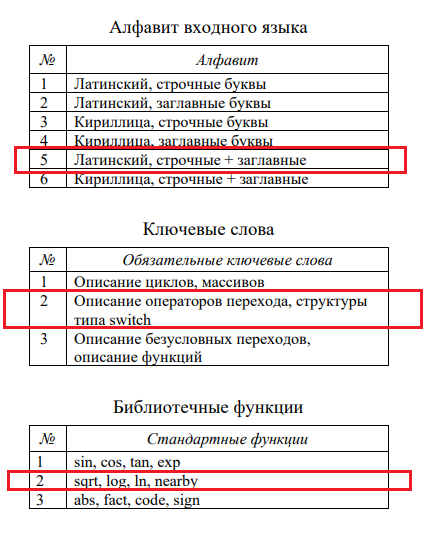
****

Рисунок 1.Изначальные данные для анализатора

**Выбранные ключевые слова:**

['BEGIN', 'PROGRAM', 'CASE', 'THEN', 'BREAK', 'VAR', 'IF', 'ELSE', 'END', 'SWITCH']

**Выбранные стандартные функции:**

sqrt, log, ln, nearby

**3.Внутренние таблицы сканера:**

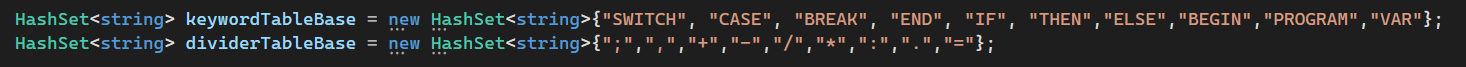


Рисунок 2.Таблица ключевых слов и разделителей

**4.Техническое задание**

**Основания для разработки**

Наименование документа: 2\_5280506778256160813

Организация утвердивший этот документ:СГТУ ЭТИ

Наименование разработки: Программы (сканера)

**Назначение разработки.**

**Требования к программе или программному изделию.**

**Требования к функциональным характеристикам:**

Программа должна производить лексический анализ входного текста для транслятора:

* создание таблиц выделенных лексем для конечных классов;
* кодирование идентификаторов, разделителей и констант;
* проверка правильности написания ключевых слов, операторов, стандартных функций и использование служебных символов;
* отображение дескрипторного текста.

**Требования к надежности:**

Программа должна стабильно обрабатывать текстовую информацию и производить лексический анализ.

**Условия эксплуатации:**

При использование программы пользователь должен облать минимальными навыками работы с языком программирования Python.

**Требования к информационно и программной совместимости:**

Программа должна быть запущенна на устройстве соответствующем минимальным требованиям языка Python:

* 64- или 32-разрядный процессор на архитектуре Intel или AMD;
* ГБ оперативной памяти, желательно 4 ГБ;
* 5 ГБ свободного места в памяти компьютера;
* Windows ,macOS ,Linux

Требования к программной документации

**Требования к программной документации.**

В комплекте с программой должен присутствовать документ со следующим содержимым:

* Код программы
* Вводимые текстовые данные
* Вывод программы

Стадии и этапы разработки

**Стадии и их этапы:**

Разработка технического задания:

1. Составление требования
2. Прописывание функций

Разработка программы:

1. Создание таблиц Ключевые слова и Разделители
2. Написание проверки на Идентификаторы и Константы
3. Вывод полученных таблиц (Ключевые слова, Разделители, Идентификаторы, Константы).
4. Написание алгоритма преобразование текста в дескрипторный текст
5. Вывод дескрипторного текста

Написание документа “Отчёт”:

1. Прописывание условий задачи
2. Вывод листинга программы
3. Вывод отладочного примера работа программы

**Сроки разработки:**

Дата начала разработки: 30.10.23

Дата завершения разработки:31.10.23

|  |  |
| --- | --- |
| Стадии: | Затраченное время |
| Разработка технического задания | 2 часа |
| Разработка программы | 2 часа |
| Написание документа “Отчёт”: | 1 час |

**Исполнители:**

Баймухамбетов Данат Владимирович

**Порядок контроля и приёмки**

**5.Листинг программы**

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

class Program

{

static void Main(string[] args)

{

string code = @"

VAR

a,b: Double;

N: Integer;

BEGIN

CASE N of

1: BEGIN

a:=sqrt(4);b:=1;

end;

2: BEGIN

a:=log(2);b:=1;

END;

ELSE WriteLn('ERROR N!');

END;

";

HashSet<string> keywordTableBase = new HashSet<string>{"SWITCH", "CASE", "BREAK", "END", "IF", "THEN","ELSE","BEGIN","PROGRAM","VAR"};

HashSet<string> dividerTableBase = new HashSet<string>{";",",","+","-","/","\*",":",".","="};

List<string> keywordTable = new List<string>();

List<string> dividerTable = new List<string>();

List<string> identifierTable = new List<string>();

List<string> constTable = new List<string>();

string pattern = @"[a-zA-Za][a-zA-Zа\_d]\*|;|,|\+|-|/|\\*|:|\.|=|\d+\.\d+|\d+";

List<string> tokens = new List<string>();

foreach (Match match in Regex.Matches(code, pattern))

{

tokens.Add(match.Value);

}

List<string> consts = new List<string>();

foreach (Match match in Regex.Matches(code, @"\d+\.\d+|\d+"))

{

consts.Add(match.Value);

}

Console.WriteLine("Ключевые слова:");

foreach (var keyword in keywordTableBase)

{

Console.WriteLine(keyword);

}

Console.WriteLine("\nРазделители:");

foreach (var divider in dividerTableBase)

{

Console.WriteLine(divider);

}

foreach (var token in tokens)

{

if (keywordTableBase.Contains(token))

{

keywordTable.Add(token);

}

else if (dividerTableBase.Contains(token))

{

dividerTable.Add(token);

}

else if (Regex.IsMatch(token, @"^[a-zA-Za][a-zA-Zа\_d]\*$"))

{

if (!keywordTable.Contains(token))

{

identifierTable.Add(token);

}

}

}

Console.WriteLine("Полученные Ключевые слова(K1):");

foreach (var keyword in keywordTable)

{

Console.Write(keyword + " ");

}

Console.WriteLine("\nПолученнные Разделители(K2):");

foreach (var divider in dividerTable)

{

Console.Write(divider + " ");

}

Console.WriteLine("\nПолученные Идентификаторы(K3):");

foreach (var identifier in identifierTable)

{

Console.Write(identifier + " ");

}

Console.WriteLine("\nПолученные Константы(K4):");

foreach (var constant in consts)

{

Console.Write(constant + " ");

}

Console.WriteLine("\n\nДескрипторный текст:");

foreach (var token in tokens)

{

if (keywordTableBase.Contains(token))

{

Console.Write("(К1," + (keywordTableBase.ToList().IndexOf(token)+1) + ") ");

}

else if (dividerTableBase.Contains(token))

{

Console.Write("(К2," + (dividerTableBase.ToList().IndexOf(token)+1) + ") ");

}

else if (Regex.IsMatch(token, @"^[a-zA-Za][a-zA-Zа\_d]\*$"))

{

if (!keywordTable.Contains(token))

{

Console.Write("(К3," + (identifierTable.IndexOf(token)+1) + ") ");

}

}

else if (Regex.IsMatch(token, @"\d+\.\d+|\d+"))

{

Console.Write("(K4," + (consts.IndexOf(token) + 1) + ") ");

}

}

}

}

**6.Отладочный пример работы программы**

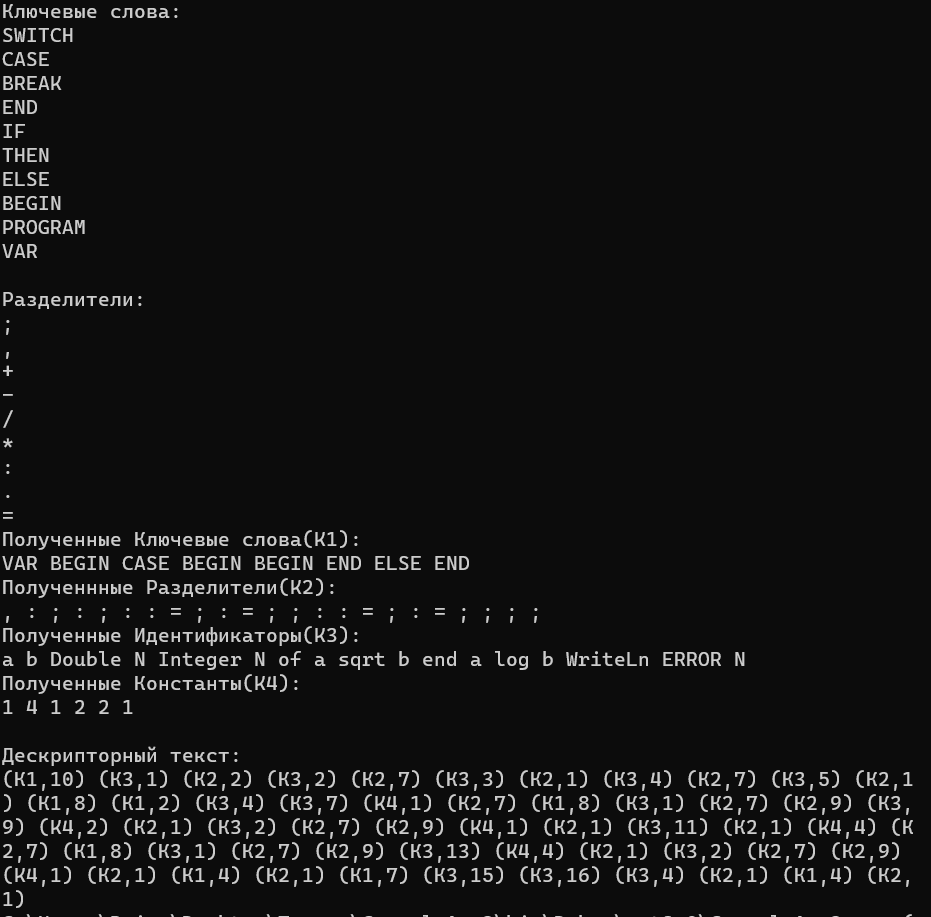


Рисунок 3.Пример работы программы