Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра «Информатика»

**Лабораторная работа № 4**

по дисциплине: **«Методы трансляции»**

**Разработка КС-грамматик**

Выполнил студент

группы ИП-32

*Кирпиченко Д.Д.*

Проверил доцент:

*Кравченко О.А.*

­

Гомель 2022

Лабораторная работа №4

**Разработка КС-грамматик**

***Цель работы***: *получить навыки создания КС-грамматик для заданного языка, разработка алгоритма и программы формирования множеств левых и правых символов для нетерминальных символов грамматики.*

***Задание***

Лабораторная работа предполагает выполнение двух этапов.

На первом этапе требуется разработать КС-грамматику, порождающую заданный язык.

На втором этапе необходимо разработать программу, которая для каждого нетерминального символа определяет множество его левых или правых символов:

L(*U*)={*W*|Ǝ(*U*→*Wx*) ˅ Ǝ(*U*→*Zx*) & *W*∈L(*Z*)} – множество левых символов нетерминального символа *U* состоит из символов, которые начинают цепочки, выводимые из *U*.

R(*U*)={*W*|Ǝ(*U*→*xW*) ˅ Ǝ(*U*→*xZ*) & *W*∈R(*Z*)} – множество правых символов нетерминального символа *U* состоит из символов, которые заканчивают цепочки, выводимые из *U*.

***Варианты индивидуальных заданий***

Вариант 3

Язык представляет собой множество арифметических выражений со скобочной структурой, операнды которого – целые числа (неограниченной длины), а операции – сложение и деление.

Пример предложения языка: ((52+37)/14+8/92+673/(12+600))/4

Для каждого нетерминального символа разработанной КС-грамматики программа должна построить множество его правых символов.

Разработанная КС-грамматика:

1. *S→B*
2. *B→A*
3. *A→T*
4. *A→AZT*
5. *T→U*
6. *T→TMU*
7. *U→H*
8. *H→R*
9. *H→(A)*
10. *R→I*
11. *I→D*
12. *I→ID*
13. *Z→+*
14. *M→/*
15. *D→δ*

Результат работы программы:

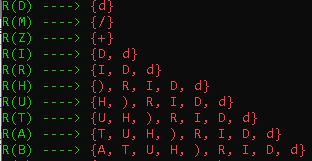


Рисунок 1- Результат работы программы

Код программы:

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab\_4

{

public class Program

{

private static bool \_isRight = true;

private static void Main(string[] args)

{

var grammatic = new List<Grammar>

{

new Grammar("S", "B"),

new Grammar("B", "A"),

new Grammar("A", "T"),

new Grammar("A", "AZT"),

new Grammar("T", "U"),

new Grammar("T", "TMU"),

new Grammar("U", "H"),

new Grammar("H", "R"),

new Grammar("H", "(A)"),

new Grammar("R", "I"),

new Grammar("I", "D"),

new Grammar("I", "ID"),

new Grammar("Z", "+"),

new Grammar("M", "/"),

new Grammar("D", "d"),

};

grammatic.Reverse();

var rules = grammatic.GroupBy(rule => rule.Key).ToDictionary(rule => rule.Key, rule => rule.ToList());

foreach (var key in rules)

{

var total = new List<char>();

AppendData(key.Key, rules, total);

var leftOrRightSign = \_isRight ? "R" : "L";

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine();

Console.Write($"{leftOrRightSign}({key.Key}) ----> ");

Console.ForegroundColor = ConsoleColor.Red;

Console.Write($"{{{string.Join(", ", total.Distinct())}}}");

}

Console.ReadLine();

}

private static void AppendData(string key, IReadOnlyDictionary<string, List<Grammar>> rules, ICollection<char> result)

{

if (!rules.ContainsKey(key)) return;

var values = rules[key];

foreach (var value in values)

{

var newKey = \_isRight ? value.Value.Last() : value.Value.First();

if (newKey.ToString().Equals(key))

{

result.Add(newKey);

continue;

}

result.Add(newKey);

AppendData(newKey.ToString(), rules, result);

}

}

}

public class Grammar

{

public Grammar(string key, string value)

{

Key = key;

Value = value;

}

public string Key { get; }

public string Value { get; }

}

}

**Вывод:** получили навыки создания КС-грамматик для заданного языка, разработали алгоритм и программу формирования множеств левых и правых символов для нетерминальных символов грамматики.