Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Курсовая работа по дисциплине**

**«Теория языков программирования и методы трансляции»**

Вариант 6

Выполнил:

Студент гр. ИП-815 \_\_\_\_\_\_\_\_\_\_\_\_\_/Кузьмин М. А./

ФИО студента

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Проверил:

Ассистент кафедры ПМиК \_\_\_\_\_\_\_\_\_\_\_\_\_/Павлова У. В./

ФИО преподавателя

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск 2021 г.

**Содержание**

**1.** Описание задания……………………………………………………..4 **2.** Описание алгоритма……………………………………………….….5 **3.** Описание функций………………………………………………….…6  
**4.** Результат работы………………………………………………………7  
**5.** Листинг…………………………………………………………………9

**Описание задания**

Написать программу, которая по предложенному описанию языка построит регулярную грамматику (ЛЛ или ПЛ – по заказу пользователя), задающую этот язык, и позволит сгенерировать с её помощью все цепочки языка в заданном диапазоне длин. Вариант задания языка: алфавит, кратность вхождения некоторого символа алфавита и обязательная фиксированная подцепочка, на которую заканчиваются все цепочки языка.

**Описание алгоритма**

Пользователь должен ввести алфавит, обязательную цепочку (поле с ней можно оставить пустым), символ и кратность. По желанию можно ввести начальную и конечную подцепочку выражения. Поля проходят валидацию – при необрабатываемых входных данных действия не будут выполнены.

При нажатии на кнопку «Правосторонняя» или «Левосторонняя» программа строит правостороннюю или левостороннюю грамматику соответственно, считывая алфавит, нужный символ и его кратность. Если пользователь ввёл конечную подцепочку, то программа продолжит работу, учитывая подцепочку.

Алгоритм вставляет подцепочку в конец выражения (проверяя при этом, нет ли там символов, которые были не указаны в алфавите), и добавляет символы алфавита в количестве, кратном введённому числу (учитывая длину подцепочки).

Для генерации цепочек в определённом диапазоне необходимо ввести два числа (поля проходят валидацию). Все возможные для алфавита цепочки генерируются в рекурсивном цикле, который проходит по всему алфавиту, учитывая, что их длина должна быть кратна введённому числу.

**Описание функций**

getAlphabet() – считывает посимвольно строку с алфавитом и записывает каждый символ в массив.

buildGrammarRight() – генерирует праволинейную грамматику. В качестве результата в соответствующее поле записываются правила, по которым строится язык. Алгоритм учитывает кратность нужного символа, и генерирует дополнительные правила, если возникает необходимость. При наличии подцепочки к одному из правил добавляется блок выхода, состоящий из требуемой подцепочки и символа «лямбда».

buildGrammarLeft() – генерирует леволинейную грамматику. В качестве результата в соответствующее поле записываются правила, по которым строится язык. Алгоритм учитывает кратность нужного символа, и генерирует дополнительные правила, если возникает необходимость. При наличии подцепочки к первому правилу добавляется блок выхода, состоящий из требуемой подцепочки и символа «лямбда».

generate() – функция для построения подцепочек определённой длинны. Сначала он проверяет, если сумма подцепочек кратна числу кратности, то он добавляет её в вывод. После он проходит по всем числам в заданном пользователем диапазоне, если находятся кратные, то он выполняет функцию loop(n), где n – кратное число, минут сумма длин подцепочек.

loop(int symbolsLeft, string finalString, int kratnSymbol) – рекурсивно вычисляет все подцепочки, состоящие из всех значений алфавита, количество одного (задаваемого пользователем) из символов в которых кратно введённому числу.

checkValidation() – проверяет правильность заполнения полей. В случае неверного заполнения полей функция возвращает false.

**Результат работы**

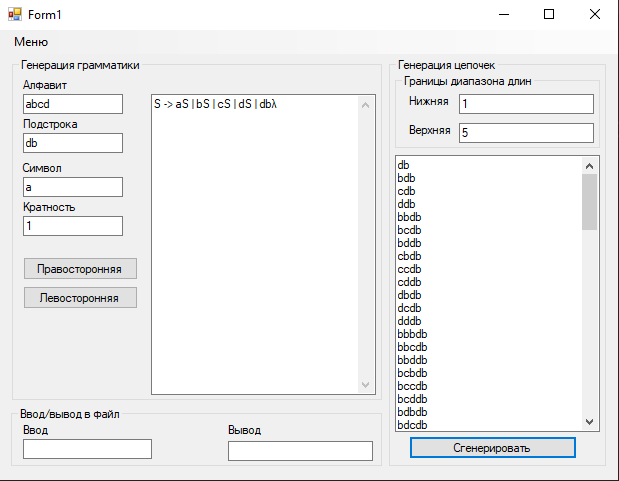


Рис. 1 – Результат работы программы

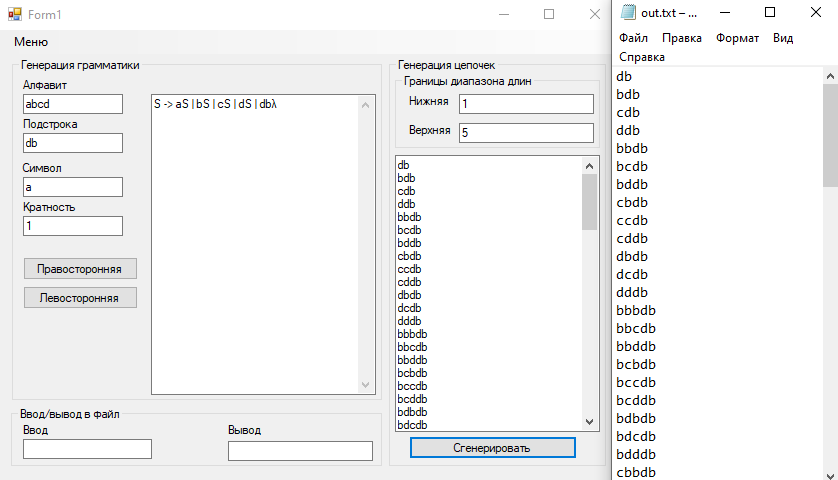


Рис. 2 – Вывод результатов в файл

**Листинг**

using System;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace mk

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

string[] alphabet;

string[] grammarAlpabet = {"S", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R"};

int alphabetLength;

bool isAlph = false;

string globalLoopString = "";

private void getAlphabet()

{

string alphabet = alphabetTextBox.Text;

alphabetLength = alphabet.Length;

int i = 0;

int repeatSym = 0;

foreach (char ch in alphabet)

{

string let = ch.ToString();

if (!this.alphabet.Contains(let))

{

this.alphabet[i] = let;

i++;

}

else repeatSym++;

}

alphabetLength -= repeatSym;

}

private void buildGrammarRight()

{

string substr = substringTextBox.Text;

string symbol = symbolTextBox.Text;

int multiple = Int32.Parse(multipleTextBox.Text); //кратность

if (multiple <= 0)

multiple = 1;

int symbolsLeft = multiple - (substr.Length - substr.Replace(symbol, "").Length) % multiple;

string res = "";

for (int i = 0; i < multiple; i++)

{

res += grammarAlpabet[i].ToString() + " -> ";

for (int j = 0; j < alphabetLength; j++)

{

if (alphabet[j] == symbol)

{

if (i == multiple - 1)

res += alphabet[j] + grammarAlpabet[0] + " | ";

else

res += alphabet[j] + grammarAlpabet[i + 1] + " | ";

}

else res += alphabet[j] + grammarAlpabet[i] + " | ";

}

if(i == symbolsLeft % multiple)

res += substr + "λ";

res += Environment.NewLine;

}

rulesTextBox.Text = res;

}

private void buildGrammarLeft()

{

string substr = substringTextBox.Text;

string symbol = symbolTextBox.Text;

int multiple = Int32.Parse(multipleTextBox.Text); //кратность

int symbolsLeft = (substr.Length - substr.Replace(symbol, "").Length) % multiple;

string res = "";

res += grammarAlpabet[0].ToString() + " -> " + grammarAlpabet[symbolsLeft + 1] + substr + Environment.NewLine;

for (int i = 1; i <= multiple; i++)

{

res += grammarAlpabet[i].ToString() + " -> ";

if (i == 1) res += "λ | ";

for (int j = 0; j < alphabetLength; j++)

{

if (alphabet[j] == symbol)

{

if (i == multiple - 1)

res += grammarAlpabet[1] + alphabet[j] + " | ";

else

res += grammarAlpabet[i + 1] + alphabet[j] + " | ";

}

else res += grammarAlpabet[i] + alphabet[j] + " | ";

}

res += Environment.NewLine;

}

rulesTextBox.Text = res;

}

private void generate()

{

string lower = lowerBound.Text;

string upper = upperBound.Text;

string substr = substringTextBox.Text;

string symbol = symbolTextBox.Text;

int multiple = Int32.Parse(multipleTextBox.Text); //кратность

int symbolsLeft = (multiple - (substr.Length - substr.Replace(symbol, "").Length) % multiple) % multiple;

int down = Int32.Parse(lower) - substr.Length;

int up = Int32.Parse(upper) - substr.Length;

if (down < 0)

down = 0;

for (int j = symbolsLeft; j <= up; j += multiple)

{

for (int i = down; i <= up; i++)

{

if (i >= j)

{

loop(i, "", j);

Console.WriteLine(i);

}

}

}

stringsTextBox.Text = globalLoopString;

}

private void loop(int symbolsLeft, string finalString, int kratnSymbols)

{

string substr = substringTextBox.Text;

if (symbolsLeft == 0 && kratnSymbols == 0)

globalLoopString += finalString + substr + Environment.NewLine;

else if(symbolsLeft > 0)

for (int i = 0; i < alphabetLength; i++)

{

if (alphabet[i] == symbolTextBox.Text)

loop(symbolsLeft - 1, finalString + alphabet[i], kratnSymbols - 1);

else

loop(symbolsLeft - 1, finalString + alphabet[i], kratnSymbols);

}

}

private Boolean checkValidation()

{

string substr = substringTextBox.Text;

string symbol = symbolTextBox.Text;

isAlph = true;

for (int i = 0; i < substr.Length; ++i)

{

string let = substr[i].ToString();

if (!alphabet.Contains(let))

{

return false;

}

}

if (!alphabet.Contains(symbol))

return false;

else

return true;

}

private void rightSideButton\_Click(object sender, EventArgs e)

{

rulesTextBox.Text = "";

string alphab = alphabetTextBox.Text;

string mult = multipleTextBox.Text;

string symbol = symbolTextBox.Text;

int res;

if (alphab.Length != 0 && mult.Length != 0 && int.TryParse(mult, out res) && symbol.Length == 1)

{

alphabet = new string[alphabetTextBox.TextLength];

getAlphabet();

if(checkValidation()) buildGrammarRight();

}

}

private void generateButton\_Click(object sender, EventArgs e)

{

string a = lowerBound.Text;

string b = upperBound.Text;

int result;

if (a.Length != 0 && b.Length != 0 && int.TryParse(a, out result) && int.TryParse(b, out result))

{

if (isAlph) generate();

globalLoopString = "";

}

}

private void leftSideButton\_Click(object sender, EventArgs e)

{

rulesTextBox.Text = "";

string alphab = alphabetTextBox.Text;

string mult = multipleTextBox.Text;

string symbol = symbolTextBox.Text;

int result;

if (alphab.Length != 0 && mult.Length != 0 && int.TryParse(mult, out result) && symbol.Length == 1)

{

alphabet = new string[alphabetTextBox.TextLength];

getAlphabet();

if (checkValidation()) buildGrammarLeft();

}

}

private void авторToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show($"Автор: Кузьмин Матвей \nИП-815");

}

private void заданиеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show($"Написать программу, которая по предложенному описанию языка построит регулярную грамматику (ЛЛ или ПЛ – по заказу пользователя), задающую этот язык, и позволит сгенерировать с её помощью все цепочки языка в заданном диапазоне длин.\nЗадание языка: алфавит, кратность вхождения некоторого символа алфавита и обязательная фиксированная подцепочка, на которую заканчиваются все цепочки языка.");

}

private void вводToolStripMenuItem\_Click(object sender, EventArgs e)

{

string read = inputTextBox.Text + ".txt";

if (inputTextBox.TextLength == 0)

read = "in.txt";

string[] lines = File.ReadAllLines(read);

int i = 0;

foreach (string s in lines)

{

if (i == 0) alphabetTextBox.Text = s;

if (i == 1) substringTextBox.Text = s;

if (i == 2) symbolTextBox.Text = s;

if (i == 3) multipleTextBox.Text = s;

i++;

}

}

private void выводToolStripMenuItem\_Click(object sender, EventArgs e)

{

string write = outputTextBox.Text + ".txt";

if (outputTextBox.TextLength == 0)

write = "out.txt";

string outText = stringsTextBox.Text;

File.WriteAllText(write, outText);

}

}

}