МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий

Кафедра «Прикладная математика»

Лабораторная работа №1

Выполнил:

Студент группы 20-ПМ-1

Бабушкин Е.В.

Проверила:

Кокоулина М.В.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc84372786)

[Теоретическая справка 4](#_Toc84372787)

[Основы теории 4](#_Toc84372788)

[Классификация грамматик по Хомскому 5](#_Toc84372789)

[Реализация 8](#_Toc84372790)

[Выводы 8](#_Toc84372791)

[Приложения 9](#_Toc84372792)

# **Постановка задачи**

При выполнении лабораторной работы следует реализовать следующие действия:

1) составить грамматику, порождающую формальный язык, заданный в соответствии с вариантом;

2) определить тип формальной грамматики и языка по классификации Хомского;

3) разработать программное средство, распознающее тип введенной пользователем грамматики по классификации Хомского.

# **Теоретическая справка**

# **Основы теории**

**Определение 1.1.** Алфавитом *V* называется конечное множество символов.

**Определение 1.2.** Цепочкой *α* в алфавите *V* называется любая конечная последовательность символов этого алфавита.

**Определение 1.3.** Цепочка, которая не содержит ни одного символа, называется пустой цепочкой и обозначается ε.

**Определение 1.4.** Формальное определение цепочки символов в алфавите *V*:

1. *ε* - цепочка в алфавите *V*;
2. если *α* - цепочка в алфавите *V* и *а* – символ этого алфавита, то *αа* – цепочка в алфавите *V*;
3. *β* - цепочка в алфавите *V* тогда и только тогда, когда она является таковой в силу утверждений 1) и 2).

**Определение 1.5.** Длиной цепочки *α* называется число составляющих ее символов (обозначается *|α |*).

Обозначим через *V\** множество, содержащее все цепочки в алфавите *V*, включая пустую цепочку *ε*, а через *V+* - множество, содержащее все цепочки в алфавите *V*, исключая пустую цепочку *ε*.

**Пример 1.1.** Пусть , тогда , а .

**Определение 1.6.** Формальной грамматикой называется четверка вида:

****, (1.1)

где *VN* - конечное множество нетерминальных символов грамматики (обычно прописные латинские буквы);

*VT* - множество терминальных символов грамматики (обычно строчные латинские буквы, цифры, и т.п.), *VT* ∩*VN =∅*;

*Р* - множество правил вывода грамматики, являющееся конечным подмножеством множества (*VT* ∪ *VN*)*+* ×(*VT* ∪ *VN*)*\**; элемент (*α, β*) множества *Р* называется правилом вывода и записывается в виде *α*→*β* (читается: «из цепочки *α* выводится цепочка *β*»);

S - начальный символ грамматики, *S* ∈*VN*.

Для записи правил вывода с одинаковыми левыми частями вида  используется сокращенная форма записи .

**Пример 1.2.** Грамматика *G*1*=*({0, 1}, {*A*, *S*}, *P*1, *S*), где множество *Р*1 состоит из правил вида: 1) *S→* 0*A*1;2)0*A→* 00*A*1;3) *A→ ε.*

**Определение 1.7.** Цепочка *β ∈* (*VT ∪ VN*)*\** непосредственно выводима из цепочки в грамматике **** (обозначается:*α* ⇒*β*), если  и , где ,  и правило вывода  содержится во множестве *Р*.

**Определение 1.8.** Цепочка β ∈ (VT ∪ VN)\* выводима из цепочки в грамматике  (обозначается α ⇒ \*β), если существует последовательность цепочек  (n≥0) такая, что .

**Пример 1.3.** В грамматике *G*1 *S* ⇒ \*000111, т.к. существует вывод .

**Определение 1.9.** Языком, порожденным грамматикой **,** называется множество всех цепочек в алфавите *VT*, которые выводимы из начального символа грамматики *S* c помощью правил множества *Р*, т.е. множество .

**Пример 1.4.** Для грамматики *G*1 *L*(*G*1)*=*{0*n*1*n* | *n>*0}.

#### **Определение 1.10.** Цепочка , для которой существует вывод S ⇒ \*α, называется сентенциальной формой в грамматике .

**Определение 1.11.** Грамматики *G*1 и *G*2 называются эквивалентными, если .

**Пример 1.5.** Для грамматики *G*1 эквивалентной будет грамматика *G*2 = ({0, 1}, {*S*}, *P*2, *S*), где множество правил вывода *P*2 содержит правила вида *S* → 0*S*1| 01.

# **Классификация грамматик по Хомскому**

#### **Тип 0.** Грамматика называется грамматикой типа 0, если на ее правила вывода не наложено никаких ограничений, кроме тех, которые указаны в определении грамматики.

**Тип 1.** Грамматика  называется контекстно-зависимой грамматикой (КЗ-грамматикой), если каждое правило вывода из множества *Р* имеет вид *α* → *β*, где *α* ∈ (*VT* ∪ *VN*)+, *β* ∈ (*VT* ∪ *VN*)\* и |*α*| ≤ |*β*|.

**Тип 2.** Грамматика  называется контекстно-свободной грамматикой (КС-грамматикой), если ее правила вывода имеют вид: , где  и 

**Тип 3.** Грамматика **** называется регулярной грамматикой (Р-грамматикой) выровненной вправо, если ее правила вывода имеют вид , где .

Грамматика **** называется регулярной грамматикой (Р-грамматикой) выровненной влево, если ее правила вывода имеют вид , где .

##### **Определение 1.12.** Язык *L*(*G*) называется языком типа *k*, если его можно описать грамматикой типа *k*, где *k* – максимально возможный номер типа грамматики.

##### Соотношение типов грамматик и языков представлено на рисунке 1.1.

##### Р – регулярная грамматика;

##### КС – контекстно-свободная грамматика;

КЗ – контекстно-зависимая грамматика;

Тип 0 – грамматика типа 0.

##### Рисунок 1.1 – Соотношение типов формальных языков и грамматик

##### **Пример 1.6.** Примеры различных типов формальных языков и грамматик по классификации Хомского. Терминалы будем обозначать строчными символами, нетерминалы – прописными буквами, начальный символ грамматики – *S*.

а) Язык типа 0 *L*(*G*)={} определяется грамматикой с правилами вывода:

1) *S* → *aaCFD*; 2) *AD* → *D*;

3) *F* → *AFB | AB*;4) *Cb* → *bC*;

5) *AB* → *bBA*;6) *CB* → *C*;

7) *Ab* → *bA*;8) *bCD* → *ε*.

б) Контекстно-зависимый язык *L*(*G*)={*anbncn* | *n*≥1} определяется грамматикой с правилами вывода:

1) *S* → *aSBC | abc* ;2) *bC* → *bc*;

###### 3) *CB* → *BC*;4) *cC* → *cc*;

5) *BB* → *bb*.

в) Контекстно-свободный язык *L*(*G*)={(*ac*)*n*(*cb*)*n* | *n*>0 } определяется грамматикой с правилами вывода:

1) *S* → *aQb | accb*;

2) *Q* → *cSc*.

г) Регулярный язык *L*(*G*)={*ω*⊥ | *ω*∈{*a, b*}+, где нет двух рядом стоящих *а*} определяется грамматикой с правилами вывода:

1) *S* → *A*⊥ *| B*⊥;

2) *A* → *a | Ba*;

3) *B* → *b | Bb | Ab*.

# **Реализация**

Дан формальный язык

G = (Vt, Vn, P, S)

Vt = {a, b, c}

Vn = {А, B, C}

P: S → abBcb | AabABcbCcbB

A → ab

C → cb

B → *ε*

S → abBcb → abcb

S → AabABcbCcbB → ababABcbCcbB → abababBcbCcbB → abababcbCcbB → abababcbcbcbB → abababcbcbcb

Грамматика G является контекстно-свободной (КС-грамматика, 2 тип)

# **Выводы**

1) Закрепили понятия «алфавит», «цепочка», «формальная грамматика» и «формальный язык», «выводимость цепочек», «эквивалентная грамматика»;

2) Сформировали умения и навыки распознавания типов формальных языков и грамматик по классификации Хомского, построения эквивалентных грамматик.

# **Приложения**

**Приложение 1:** Код программы

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

bool search(char x, vector<char> arr) {

for (unsigned int i = 0; i < arr.size(); i++){

if (x == arr[i]) return true;

}

return false;

}

int main()

{

fstream fileV;

fstream fileP;

vector<char> Vt;

vector<char> Vn;

vector<string> Pf;

vector<string> Ps;

string inp;

string cashF;

string cashS;

string a;

string b;

bool flag;

int type;

fileV.open("V.txt", ios::in);

if (!fileV){

cout << "Error file!" << endl;

return 1;

}

getline(fileV, inp);

for (unsigned int i = 0; i < inp.length(); i++){

Vt.push\_back(inp[i]);

}

getline(fileV, inp);

for (unsigned int i = 0; i < inp.length(); i++){

Vn.push\_back(inp[i]);

}

fileV.close();

fileP.open("P.txt", ios::in);

if (!fileP){

cout << "Error file!" << endl;

return 1;

}

while (getline(fileP, inp)){

flag = false;

cashF = "";

cashS = "";

for (unsigned int i = 0; i < inp.length(); i++){

if (inp[i] == '-'){

flag = true;

continue;

} else {

if (flag){

cashS += inp[i];

} else {

cashF += inp[i];

}

}

}

Pf.push\_back(cashF);

Ps.push\_back(cashS);

}

fileP.close();

type = 3;

for (unsigned int i = 0; i < Pf.size(); i++){

if (Pf[i] == ""){

type = 0;

break;

}

switch (type) {

case 3:

for (unsigned int j = 0; j < Pf[i].length(); j++){

if (!search(Pf[i][j], Vn)) {

type = 1;

break;

}

}

case 2:

for (unsigned int j = 0; j < Pf[i].length(); j++){

if (!search(Pf[i][j], Vn)) {

type = 1;

break;

}

}

}

}

for (unsigned int i = 0; i < Ps.size(); i++){

switch (type) {

case 3:

if (Ps[i] == ""){

type = 2;

break;

}

flag = false;

if (search(Ps[i][0], Vn)){

for (unsigned int j = 1; j < Ps[i].length(); j++){

if (flag){

if (!search(Ps[i][j], Vt)){

type = 2;

break;

}

}else if (!search(Ps[i][j], Vn)) flag = true;

}

}else if (search(Ps[i][0], Vt)){

for (unsigned int j = 1; j < Ps[i].length(); j++){

if (flag){

if (!search(Ps[i][j], Vn)){

type = 2;

break;

}

}else if (!search(Ps[i][j], Vt)) flag = true;

}

}else return 2;

case 1:

if (Pf[i].length() > Ps[i].length()) type = 0;

}

if (type == 0 or type == 2) break;

}

cout << "Type: " << type << endl;

return 0;

}

**Приложение 2:** Пример файла P.txt

Aa-aa

A-a

B-b

cC-cc

B-BB

C-c

A-AA

**Приложение 3:** Пример файла V.txt

abc

ABC

**Приложение 4:** Скриншот работы программы

