**Сибирский Государственный Университет Телекоммуникаций и Информатики**

Межрегиональный центр переподготовки специалистов

**КУРСОВАЯ РАБОТА**

по предмету: «Теория языков программирования и методы трансляции»

**Выполнил**: Савина И.С.

**Группа**: ПБВ-92

**Вариант**: 10

**Проверил**: преподаватель

Бах Ольга Анатольевна

2019

**Задание**

Написать программу для автоматического построения регулярной грамматики (леволинейной или праволинейной) по словесному описанию языка.

Вход программы: алфавит языка, обязательная конечная подцепочка, кратность длины всех цепочек языка, тип грамматики (ЛЛ либо ПЛ), 2 числа – диапазон длин для генерации цепочек.

Выход: построенная грамматика (все 4 элемента), результат генерации цепочек.

Подробно:

Язык задан своим алфавитом, обязательной конечной подцепочкой, которая должна присутствовать во всех цепочках языка, и указанием кратности длины всех цепочек языка. В конечной цепочке не должно находиться символов, не содержащихся в алфавите. В крайнем случае она может быть и пустой. Кратность длины цепочек задаётся любым натуральным числом. Кратность, равная единице, равносильна отсутствию ограничений на длину цепочек.

Программа должна:

1. по предложенному описанию регулярного языка строить регулярную грамматику, генерирующую этот язык, в том виде, как она рассматривалась в теории, раздел 1.3.1;

2. с помощью построенной грамматики генерировать все цепочки языка в заданном пользователем диапазоне длин.

Грамматика должна строиться и праволинейная, и леволинейная. Желаемый тип грамматики выбирается пользователем в меню. Все правила грамматики должны соответствовать выбранному типу! Отдельно следует указывать, какой нетерминальный символ является целевым. Если в грамматике используется пустое правило, то необходимо дать пояснение, каким именно символом обозначается пустая цепочка.

После построения грамматики пользователь может убедиться в её правильности путём генерации всех цепочек языка в том диапазоне длин, который он задаст. Генерацию каждой цепочки языка следует поэтапно отображать на экране в виде цепочки вывода (в соответствии с примерами раздела 1.4.1.). Генерация осуществляется в соответствии с лабораторной работой №1.

**Алгоритм решения задачи**

Входные данные программы состоят из символов алфавита языка, обязательной конечной цепочка, которая может быть пустой, кратности длин цепочек языка. Символы алфавита языка задаются слитной цепочкой символов без каких-либо разделителей. В алфавите нельзя указывать символ «л», используемой программой как лямбда-символ, и заглавные буквы латинского алфавита, используемые как нетерминальные символы.

Входные данные могут быть заданы с клавиатуры путем заполнения полей на экране программы, или введены из заранее подготовленного текстового файла. Из текстового файла считываются три строчки, в первой из которых должны быть записаны символы алфавита языка, во второй – обязательная конечная цепочка, в третьей – кратность длин цепочек языка. Содержимое эти трех строк переносится в соответствующие поля экрана, где они могут быть изменены при необходимости.

Тип строящейся программой регулярной грамматики (леволинейная или праволинейная) выбирается в пункте меню «Расчеты».

Выходными данными программы является построенная регулярная грамматика заданного типа, состоящая из множества нетерминальных символов, множества терминальных символов, правил грамматики, начального символа, в качестве которого используется «S». Грамматика отображается на экране и может быть записана в текстовый файл через меню «Сохранить в файл».

При генерации цепочек по построенной грамматике по запросу программы с клавиатуры вводится диапазон длин в виде минимальной и максимальной длины.

Результаты генерации цепочек отображаются на экране в виде отдельных строк вывода каждой цепочки. Каждая строка завершается сгенерированной цепочкой. Сгенерированные цепочки могут быть записаны в текстовый файл через меню «Сохранить в файл».

Построение грамматики и генерация цепочек выполняется через один из подпунктов меню «Расчеты».

При описании алгоритма построения регулярной грамматики кратность длин цепочек языка будет обозначаться K (эта переменная используется в программе).

В качестве начального символа используется символ S. Для следующих нетерминальных символов используются заглавные буквы латинского алфавита A, B, C и т.д. В общем случае, заданная кратность длин генерируемых цепочек обеспечивается циклическим (есть переход от последнего к начальному) чередованием K нетерминалов, начиная с начального символа S. Однако есть исключение, в случае ЛЛ-грамматики и непустой обязательной конечной цепочки добавляется еще один нетерминал. Это объясняется тем, что в указанном случае начальный символ S нельзя использовать в циклическом чередовании нетерминалов, т.к. в нем выводится обязательная конечная цепочка. Таким образом, в грамматику добавляется либо K-1, либо K нетерминальных символов. Например, если K равен 2, а обязательная конечная цепочка непустая, то для ЛЛ-грамматики Vn={S,A,B}, а для ПЛ-грамматики Vn={S,A}.

Вычисляется число k, которое будет определять порядковый номер нетерминала, в котором следует завершить генерацию цепочки. Для этого определяем число символов l, которое надо добавить к обязательной конечной цепочке, чтобы получить длину, кратную заданной длине цепочек языка. Если длина обязательной конечной цепочки кратна заданной кратности, т.е. l=0, то k=0. В противном случае, k=K-l. В ПЛ-грамматике начальный нетерминальный символ участвует в чередовании нетерминалов для обеспечения заданной кратности, поэтому вычисленное значение k равно порядковому номеру нетерминала во множестве Vn (нумерация идет с 0). В случае ЛЛ-грамматики и непустой обязательной конечной цепочке начальный нетерминальный символ не участвует в чередовании нетерминалов для обеспечения заданной кратности, поэтому вычисленное значение k увеличивается на 1.

Для ЛЛ-грамматики и непустой обязательной конечной цепочке записывается правило: S→A«обязательная конечная цепочка»

Выполняется цикл для K-1 нетерминалов, идущих подряд в Vn. Для всех символов алфавита записываются правила вида:

Vi→Vi+1tj ЛЛ-грамматика

или

Vi→ tj Vi+1 ПЛ- грамматика

,где tj – символ алфавита языка

Если номер нетерминала совпадает с вычисленным значением k, то добавляется правило:

Vi→λ ЛЛ- грамматика

или

Vi→ «обязательная конечная цепочка» ПЛ- грамматика

После завершения цикла для нетерминала Vi+1, который был использован на последнем шаге в правой части правил, записываются переходы по всем символам алфавита к нетерминалу, который был в левой части правил в начале цикла. В случае ЛЛ-грамматики это может быть символ S (пустая обязательная конечная цепочка) или A (непустая обязательная конечная цепочка). В случае ПЛ-грамматики переход всегда будет сделан к S. Если номер этого нетерминала совпадает с вычисленным значением k, то для него добавляется правило:

Vi+1→λ ЛЛ- грамматика

или

Vi+1→ «обязательная конечная цепочка» ПЛ- грамматика

Построение завершено.

Рассмотрим пример построения по описанному алгоритму праволинейной грамматики языка с алфавитом {a,b}, обязательной конечной цепочкой «abba» и кратностью длин цепочек языка 2.

Поскольку K=2, то Vn={S,A}

Длина обязательной конечной цепочки кратна 2, значит k=0.

Цикл будет выполнен только один раз, т.к. K-1=2-1=1. Для S добавляем правила:

S→aA

S→bA

Порядковый номер S (0) равен k, добавляем для S правило:

S→abba

На последнем шаге в правой части правил был использован нетерминал A. Для него записываются переходы по всем символам алфавита к нетерминалу, который был в левой части правил в начале цикла, т.е. S.

A→aS

A→bS

Регулярная праволинейная грамматика построена:

G=({S,A},{a,b},P,S)

P:

S→aA

S→bA

S→abba

A→aS

A→bS

**Описание основных переменных блоков и подпрограмм**

Основные переменные программы:

Vt – строка символов алфавита языка;

StringEnd – строка обязательной конечной цепочки;

Vn – строка множества нетерминальных символов регулярной грамматики;

Lr – массив длин минимальных цепочек, выводимых из каждого нетерминального символа;

R – двумерный массив строк правых частей правил регулярной грамматики;

K – кратность длин цепочек языка;

Min – минимальная длина цепочки;

Max – максимальная длина цепочки;

Основные программные блоки:

Grammar – функция построения регулярной грамматики;

GetLr – функция вычисления длин минимальных цепочек, выводимых из каждого нетерминального символа;

Generate – функция генерации цепочек языка в заданном диапазоне длин.

**Текст программы**

#pragma hdrstop

#include "Unit1.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

const AnsiString Lambda="л";

AnsiString R[26][10];

int Vr[26];

AnsiString Vn, Vt, StringEnd;

int K, Min, Max;

bool LL;

//поиск свободной заглавной буквы латинского алфавита для нетерминального символа

AnsiString \_\_fastcall TForm1::GetVN()

{

for (int i=0;i<26;i++) {

char v='A'+i;

if (Vn.Pos(v)==0) return AnsiString(v);

}

ShowMessage("Не хватает заглавных символов латинского алфавита для нетерминальных символов");

return NULL;

}

//вычисление длин минимальных цепочек, выводимых из нетерминалов грамматики

void \_\_fastcall TForm1::GetLr()

{

int l=0;

for (int i=0;i<26;i++) Vr[i]=-1;

for (int i=0;i<26;i++) {

if (Vr[i]!=-1) continue;

l=0;

for (int j=0;R[i][j]!="";j++) {

if (R[i][j]==Lambda) {

Vr[i]=0;

break;

}

else

for (int k=1;k<=R[i][j].Length();k++) {

int n=Vn.Pos(R[i][j][k]);

if (n==0) l++;

else

if (Vr[n]!=-1) l+=Vr[n];

}

if ((l>=0)&&((Vr[i]<0)||(Vr[i]>l))) Vr[i]=l;

}

}

for (int i=0;i<26;i++)

if (Vr[i]==-1) Vr[i]=1;

}

//рекурсивная генерация цепочек языка по построенной грамматике

void \_\_fastcall TForm1::Generate(AnsiString s, AnsiString sgen, int dr)

{

AnsiString sl, sr;

int i, k, l, n;

if (dr>50) return;

l=s.Length();

for (i=1;i!=l+1;i++)

if ((n=Vn.Pos(s[i]))!=0) break;

if (i==l+1) {

if ((l>=Min)&&(l<=Max)) {

if (s=="") sgen=sgen+Lambda;

for (int j=0;j<ListBox3->Items->Count;j++)

if (ListBox3->Items->Strings[j]==s) return;

ListBox1->Items->Insert(ListBox3->Items->Add(s),sgen);

}

return;

}

k=i-1;

for (int j=i;j<=l;j++) {

int m=Vn.Pos(s[j]);

if (m==0) k++;

else k+=Vr[m];

}

if (k>Max) return;

sl=s.SubString(1,i-1);

if (sl.Length()>Max) return;

sr=s.SubString(i+1,l-i);

if (sr.Length()>Max) return;

for (i=0;R[n][i]!="";i++)

if (R[n][i]==Lambda) Generate(sl+sr,sgen+"->"+sl+sr,dr+1);

else Generate(sl+R[n][i]+sr,sgen+"->"+sl+R[n][i]+sr,dr+1);

}

//построение заданной грамматики

bool \_\_fastcall TForm1::Grammar()

{

AnsiString s, v, v0;

int i, k, j, n;

N8->Enabled=false;

N10->Enabled=false;

Vt=Edit1->Text;

for (i=1;i<=Vt.Length();i++) {

if (Vt[i]==Lambda[1]) {

ShowMessage("В алфавите встречается символ, используемый как пустой 'л'");

return false;

}

for (j=i+1;j<=Vt.Length();j++)

if (Vt[i]==Vt[j]) {

ShowMessage("Символ алфавита встречается несколько раз");

return false;

}

}

StringEnd=Edit2->Text;

v="S";

Vn=v;

K=StrToIntDef(Edit3->Text,0);

if (K<1) {

ShowMessage("Ошибка в задании кратности длин цепочек");

return false;

}

for (i=0;i<26;i++)

for (j=0;j<10;j++) R[i][j]="";

if (Vt=="") {

ShowMessage("Не задан входной алфавит");

return false;

}

for (i=1;i<=StringEnd.Length();i++)

if (Vt.Pos(StringEnd[i])==0) {

ShowMessage("Посторонний символ в конечной цепочке");

return false;

}

ListBox2->Items->Clear();

k=StringEnd.Length()%K;

if (k>0) k=K-k;

n=1;

if (LL) {

if (StringEnd!="") {

s=v+"->";

v=GetVN();

Vn=Vn+v;

s=s+v+StringEnd;

ListBox2->Items->Add(s);

R[1][0]=v+StringEnd;

n++;

}

v0=v;

for (i=0;i<K-1;i++,n++) {

s=v+"->";

v=GetVN();

Vn=Vn+v;

for (j=1;j<=Vt.Length();j++) {

s=s+v+Vt[j];

if (j<Vt.Length()) s=s+"|";

R[n][j-1]=v+Vt[j];

}

if (k==i) {

s=s+"|"+Lambda;

R[n][j-1]=Lambda;

}

ListBox2->Items->Add(s);

}

s=v+"->";

for (j=1;j<=Vt.Length();j++) {

R[n][j-1]=v0+Vt[j];

s=s+v0+Vt[j];

if (j<Vt.Length()) s=s+"|";

}

if (k==i) {

s=s+"|"+Lambda;

R[n][j-1]=Lambda;

}

ListBox2->Items->Add(s);

}

else { //построение праволинейной грамматики

for (i=0;i<K-1;i++,n++) {

s=v+"->";

v=GetVN();

Vn=Vn+v;

for (j=1;j<=Vt.Length();j++) {

s=s+Vt[j]+v;

R[n][j-1]=AnsiString(Vt[j])+v;

if (j<Vt.Length()) s=s+"|";

}

if (k==i) {

s=s+"|";

if (StringEnd!="") {

R[n][j-1]=StringEnd;

s=s+StringEnd;

}

else {

R[n][j-1]=Lambda;

s=s+Lambda;

}

}

ListBox2->Items->Add(s);

}

s=v+"->";

for (j=1;j<=Vt.Length();j++) {

s=s+AnsiString(Vt[j])+"S";

if (j<Vt.Length()) s=s+"|";

R[n][j-1]=AnsiString(Vt[j])+"S";

}

if (k==i) {

s=s+"|";

if (StringEnd!="") {

R[n][j-1]=StringEnd;

s=s+StringEnd;

}

else {

R[n][j-1]=Lambda;

s=s+Lambda;

}

}

ListBox2->Items->Add(s);

}

for (i=1;i<=Vt.Length();i++)

if (Vn.Pos(Vt[i])>0) {

ListBox2->Items->Clear();

ShowMessage("В алфавите языка встречается символ, используемый как нетерминальный символ: "+

AnsiString(Vt[i]));

return false;

}

N8->Enabled=true;

N10->Enabled=true;

s="G({";

for (i=1;i<=Vn.Length();i++) {

s=s+Vn[i];

if (i<Vn.Length()) s=s+",";

}

s=s+"},{";

for (i=1;i<=Vt.Length();i++) {

s=s+Vt[i];

if (i<Vt.Length()) s=s+",";

}

s=s+"},P,S)";

Label1->Caption=s;

return true;

}

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//"Тема"

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{

ShowMessage("Построение регулярной грамматики (леволинейной или праволинейной) по словесному описанию языка");

}

//"Выход"

void \_\_fastcall TForm1::N5Click(TObject \*Sender)

{

Application->Terminate();

}

//"Автор"

void \_\_fastcall TForm1::N6Click(TObject \*Sender)

{

ShowMessage("Автор:");

}

//"Сохранить в файл"->"Цепочки"

void \_\_fastcall TForm1::N11Click(TObject \*Sender)

{

SaveDialog1->InitialDir=GetCurrentDir();

if (SaveDialog1->Execute()) ListBox1->Items->SaveToFile(SaveDialog1->FileName);

}

//"Ввести данные из файла"

void \_\_fastcall TForm1::N9Click(TObject \*Sender)

{

OpenDialog1->InitialDir=GetCurrentDir();

if (OpenDialog1->Execute()) {

ListBox1->Items->LoadFromFile(OpenDialog1->FileName);

Edit1->Text=ListBox1->Items->Strings[0];

Edit2->Text=ListBox1->Items->Strings[1];

Edit3->Text=ListBox1->Items->Strings[2];

ListBox1->Items->Clear();

}

}

//"Расчеты"->"Построить ЛЛ-грамматику"

void \_\_fastcall TForm1::N4Click(TObject \*Sender)

{

LL=true;

Grammar();

}

//"Расчеты"->"Построить ПЛ-грамматику"

void \_\_fastcall TForm1::N7Click(TObject \*Sender)

{

LL=false;

Grammar();

}

//"Расчеты"->"Генерировать цепочки"

void \_\_fastcall TForm1::N8Click(TObject \*Sender)

{

if (!Grammar()) return;

Min=StrToIntDef(InputBox("Диапазон длин цепочек","Минимальная длина","1"),-1);

Max=StrToIntDef(InputBox("Диапазон длин цепочек","Максимальная длина","5"),-1);

if ((Min<0)||(Max<0)||(Max<Min)) {

ShowMessage("Ошибка в задании диапазоны длин цепочек");

return;

}

ListBox1->Items->Clear();

ListBox1->Perform(LB\_SETHORIZONTALEXTENT,700,0);

ListBox3->Items->Clear();

ListBox3->Sorted=true;

N8->Enabled=false;

GetLr();

Generate("S","S",0);

N8->Enabled=true;

N11->Enabled=true;

}

//"Сохранить в файл"->"Грамматику"

void \_\_fastcall TForm1::N10Click(TObject \*Sender)

{

SaveDialog1->InitialDir=GetCurrentDir();

if (SaveDialog1->Execute()) {

ListBox2->Items->Add(Label1->Caption);

ListBox2->Items->SaveToFile(SaveDialog1->FileName);

ListBox2->Items->Delete(0);

}

}

**Результаты тестирования программы**

Задан язык: алфавит {a,b,c}, обязательная конечная цепочка ‘aab’, кратность длины 2. Анализируем задание: язык будет состоять из цепочек чётной длины, заканчивающихся на ‘aab’, например {сaab, аaab, bааb, caaааb, сabaab, …}.

Для примера выберем праволинейную грамматику (ПЛ). В правой части её правил вывода может быть только один нетерминал, и он должен располагаться справа от цепочек терминальных символов. Для соблюдения кратности длины цепочек будет использоваться чередование нетерминалов. Длина кратна двум, следовательно, потребуется два нетерминала: S→aA|bA|cA, А→aS|bS|cS. Выход должен происходить по обязательной конечной цепочке ‘aab’, в силу чётности длины цепочек перед ней должен быть как минимум один символ. Следовательно, эта цепочка должна порождаться нетерминалом А: А→aab.

Итак, построенная грамматика имеет вид:

G({a,b,c},{S,A},P,S), Р: S→aA|bA|cA; А→aS|bS|cS|aab.

Проверим это на нашей программе:

Сгенерируем цепочки языка в диапазоне длин от 4 до 6.

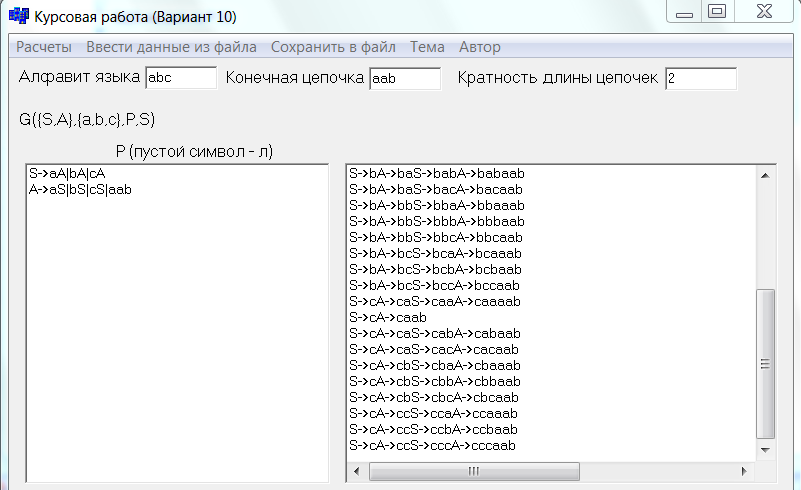


Рис.1

Как видим, программа выдала список возможных цепочек, весь результат не поместился на один экран, поэтому полный список можно посмотреть в сохраненном файле 1plCEP.txt, сохраненная грамматика в файле 1plGR.txt.

Построим также леволинейную грамматику для этого примера:

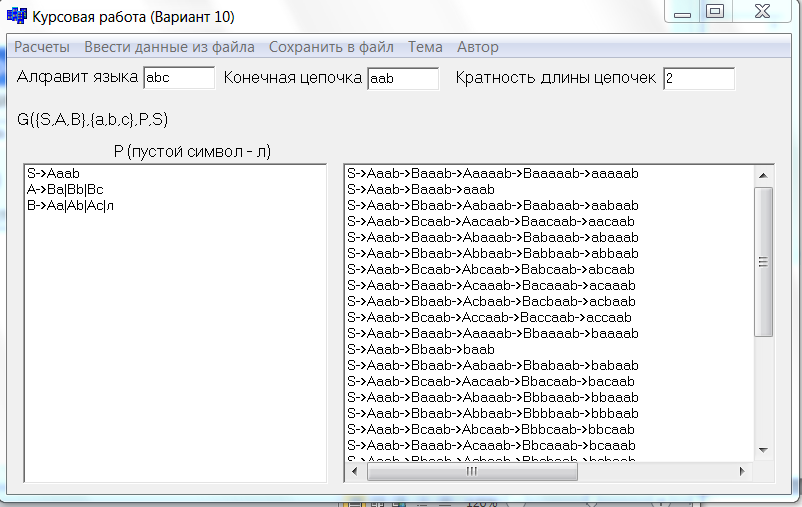


Рис.2

Построим леволинейную грамматику для языка с алфавитом {0,1}, пустой обязательной конечной цепочкой и кратностью длин цепочек языка 1. Cгенерируем по построенной грамматике цепочки языка в диапазоне 1 до 5. Результат представлен на рис.3.

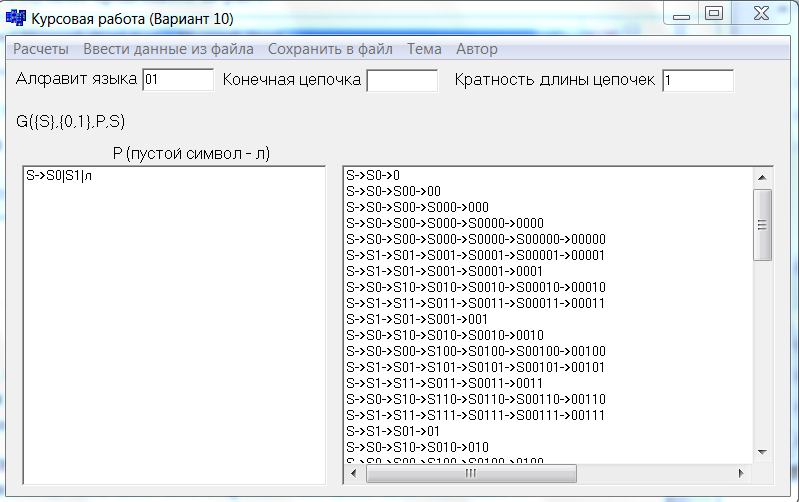


Рис.3

Таким образом, программа для автоматического построения регулярной грамматики по словесному описанию языка выдает корректные результаты, соответствующие изложенному теоретическому материалу.