МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий
Кафедра прикладной математики

Учебная дисциплина
"Формальные языки и алгоритмы"

Лабораторная работа №2

Вариант №10

Выполнила:

Козловская А.С.

19-ПМ-2

Проверил:

Доцент кафедры «Вычислительные системы и технологии»

Жевнерчук Д.В.

Нижний Новгород
2020

Содержание

Постановка задачи

Разработать программу, выполняющую следующие функции:

1. организация ввода грамматики и проверка ее на принадлежность к классу КС-грамматик;
2. проверка существования языка КС-грамматики;
3. реализация эквивалентных преобразований грамматики, направленных на удаление:
   a) не выводимых символов;
   b) недостижимых символов;
   c) смешанных цепочек;
   d) длинных цепочек;
   e) eps-правил;

Вариант: $G=(\{R, T, U, W, V\}, \{0, 1, +, -, *, /\}, P, R)$, где P:

1) $R \rightarrow T1T \mid T1U \mid W \mid \varepsilon$;
2) $T \rightarrow U \mid T01 \mid T10 \mid \varepsilon$;
3) $U \rightarrow +U \mid +0 \mid +1$
4) $W \rightarrow W\text{-}W \mid W+W$;
5) $V \rightarrow *0 \mid /1$.

Замечание: в программе “ε” обозначается как “e”.

Хранение грамматики и функции

Грамматика хранится в двух векторах — для символов и для правил.

Элементами векторов являются структуры Rule и Symbol:

```
struct Symbol
{
    int TermIndex; // 0 - нетерминальный, 1 - терминальный
    char Value; //сам символ
};

struct Rule
{
    string LeftStr; //левая часть правила
    string RightStr; //правая часть правила
};
```

Вектора передаются по ссылке передаются в функции, где и происходят преобразования.

Реализованы функции:

- инициализации грамматики (InitGrammar),
- вывода грамматики (PrintGrammar),
- определения принадлежности грамматики классу КС (IsContFree),
- проверки существования языка грамматики и удаления невыводимых (непорождающих) символов (DelNotOut),
- удаления недостижимых символов (DelNotReach),
- удаления смешанных цепочек (DelMix),
- удаления длинных цепочек (DelLong),
- удаления eps-правил (DelEps).

Результат работы программы

```
anna@anna-Aspire-A315-21:~/CLionProjects/LABA2$ ./app
Введите нетерминальные символы:
R T U W V
Введите терминальные символы:
0 1 + - * /
Введите правила:
R T1T R T1U R W R e
T U T T01 T T10 T e
U +U U +0 U +1
W W-W W W+W
V *0 V /1

Данная грамматика принадлежит классу КС-грамматик.

Существует язык КС-грамматики.

Удалены невыводимые символы. Грамматика:
Нетерминальные символы:
R T U V
Терминальные символы:
/ 0 1 + - *
Правила:
R -> T1T
R -> T1U
V -> /1
R -> e
T -> U
T -> T01
T -> T10
T -> e
U -> +U
U -> +0
U -> +1
V -> *0

Удалены недостижимые символы. Грамматика:
Нетерминальные символы:
R T U
Терминальные символы:
/ * 0 1 + -
Правила:
R -> T1T
R -> T1U
R -> e
T -> U
U -> +1
T -> T01
T -> T10
T -> e
U -> +U
U -> +0

Удалены смешанные цепочки. Грамматика:
Нетерминальные символы:
R T U
Терминальные символы:
/ * 0 1 + -
Правила:
R -> TAT
R -> TAU
R -> e
T -> U
```

```
Нетерминальные символы:
R T U
Терминальные символы:
/ * 0 1 + -
Правила:
R -> TAT
R -> TAU
R -> e
T -> U
U -> +1
T -> TBA
T -> TAB
T -> e
U -> CU
U -> +0
A -> 1
B -> 0
C -> +

Удалены длинные цепочки.Грамматика:
Нетерминальные символы:
R T U
Терминальные символы:
/ * 0 1 + -
Правила:
R -> MT
R -> MU
R -> e
T -> U
U -> +1
T -> NA
T -> MB
T -> e
U -> CU
U -> +0
A -> 1
B -> 0
C -> +
M -> TA
N -> TB

Удалены эпсилон-правила. Грамматика:
Нетерминальные символы:
R T U
Терминальные символы:
/ * 0 1 + -
Правила:
R -> MT
R -> MU
R -> e
T -> U
U -> +1
T -> NA
T -> MB
N -> TB
U -> CU
U -> +0
A -> 1
B -> 0
C -> +
M -> TA
anna@anna-Aspire-A315-21:~/CLionProjects/LABA2$
```

Код программы

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
struct Symbol
{
    int TermIndex; // 0 - нетерминальный, 1 - терминальный
    char Value; //сам символ
};
struct Rule
{
    string LeftStr; //левая часть правила
    string RightStr; //правая часть правила
};
//функция ввода грамматики
void InitGrammar(vector <Symbol> &ArrSymb, vector <Rule> &ArrRule)
{
    char input;
    cout << "Введите нетерминальные символы:" << endl;
    for (int i = 0; i < 5; i++)
    {
        cin >> input;
        ArrSymb[i].TermIndex = 0;
        ArrSymb[i].Value = input;
    }
    cout << "Введите терминальные символы:" << endl;
    for (int i = 5; i < 11; i++)
    {
        cin >> input;
        ArrSymb[i].TermIndex = 1;
        ArrSymb[i].Value = input;
    }
    cout << "Введите правила:" << endl;
    string initL, initR;
    for (int i = 0; i < 15; i++)
    {
        cin >> initL >> initR;
        ArrRule[i].LeftStr = initL;
        ArrRule[i].RightStr = initR;
    }
}
//функция вывода грамматики
void PrintGrammar(vector <Symbol> &ArrSymb, vector <Rule> &ArrRule)
{
    cout << "Нетерминальные символы:" << endl;
    int i = 0;
    while (i < ArrSymb.size())
    {
        if (ArrSymb[i].TermIndex == 0)
        {
            cout << ArrSymb[i].Value << ' ';
        }
        i++;
    }
    cout << endl;
    cout << "Терминальные символы:" << endl;
```

```
    i = 0;
    while (i < ArrSymb.size())
    {
        if (ArrSymb[i].TermIndex == 1)
        {
            cout << ArrSymb[i].Value << ' ';
        }
        i++;
    }
    i = 0;
    cout << endl;
    cout << "Правила:" << endl;
    while (i < ArrRule.size())
    {
        cout << ArrRule[i].LeftStr << " -> " << ArrRule[i].RightStr << endl;
        i++;
    }
}
//функция, выясняющая является ли грамматика КС
void IsContFree(vector <Rule> &ArrRule)
{
    bool result;
    for (int i = 0; i < 15; i++)
    {
        if ((ArrRule[i].LeftStr.size() > 1) || ((int)ArrRule[i].LeftStr[0] <= 65
&& (int)ArrRule[0].LeftStr[0] >= 90))
        {
            result = false;
        }
        else
        {
            result = true;
        }
    }
    if (result)
    {
        cout << "Данная грамматика принадлежит классу КС-грамматик." << endl;
    }
    else
    {
        cout << "Данная грамматика НЕ принадлежит классу КС-грамматик." << endl;
    }
}
//функция, удаляющая невыводимые (непорождающие) символы
void DelNotOut(vector <Rule> &ArrRule, vector <Symbol> &ArrSymb)
{
    string OutSymb;
    int CountOut = 0;
    //находим нетерминальные символы, переходящие в цепочку терминальных
    for (int i = 0; i < 15; i++)
    {
        for (int j = 0; j < ArrRule[i].RightStr.size(); j++)
        {
            for (int k = 0; k < ArrSymb.size(); k++)
            {
                if ((ArrRule[i].RightStr[j] == ArrSymb[k].Value) &&
(ArrSymb[k].TermIndex == 1) && (ArrRule[i].RightStr[j] != 'e'))
                {
                    CountOut++;
                }
```

```
            }
        }
        if (CountOut == (ArrRule[i].RightStr.size()))
        {
            OutSymb += ArrRule[i].LeftStr;
        }
        CountOut = 0;
    }
    //находим выводимые
    for (int l = 0; l < 5; l++)
    {
        for (int i = 0; i < 15; i++)
        {
            for (int j = 0; j < ArrRule[i].RightStr.size(); j++)
            {
                for (int k = 0; k < ArrSymb.size(); k++)
                {
                    if (((ArrRule[i].RightStr[j] == ArrSymb[k].Value) &&
(ArrSymb[k].TermIndex == 1) &&(ArrRule[i].RightStr[j] != 'e'))
                    || (OutSymb.find(ArrRule[i].RightStr[j]) != -1))
                    {
                        CountOut++;
                        break;
                    }
                }
            }
            if (CountOut == (ArrRule[i].RightStr.size()))
            {
                OutSymb += ArrRule[i].LeftStr;
            }
            CountOut = 0;
        }
    }
    if (OutSymb.find('R') != -1)
    {
        cout << "Существует язык КС-грамматики." << endl << endl;
    }
    else
    {
        cout << "Не существует языка КС-грамматики." << endl << endl;
    }
    //находим невыводимые
    vector <char> NotOut;
    bool check = true;
    for (int i = 0; i < ArrRule.size(); i++)
    {
        if (OutSymb.find(ArrRule[i].LeftStr[0]) == -1)
        {
            for (int j = 0; j < NotOut.size(); j++)
            {
                if (ArrRule[i].LeftStr[0] == NotOut[j])
                {
                    check = false;
                    break;
                }
            }
            if (check)
            {
                NotOut.push_back(ArrRule[i].LeftStr[0]);
            }
```

```
        }
    }
    //удаляем невыводимые символы из массива символов
    for (int i = 0; i < ArrSymb.size(); i++)
    {
        for (int j = 0; j < NotOut.size(); j++)
        {
            if (ArrSymb[i].Value == NotOut[j])
            {
                ArrSymb[i].Value = ArrSymb[ArrSymb.size() - 1].Value;
                ArrSymb[i].TermIndex = ArrSymb[ArrSymb.size() - 1].TermIndex;
                ArrSymb.pop_back();
                break;
            }
        }
    }
    //удаляем невыводимые из правил
    for (int i = 0; i < ArrRule.size(); i++)
    {
        for (int j = 0; j < NotOut.size(); j++)
        {
            if ((ArrRule[i].LeftStr[0] == NotOut[j]) ||
(ArrRule[i].RightStr.find(NotOut[j]) != -1))
            {
                ArrRule[i].LeftStr = ArrRule[ArrRule.size() - 1].LeftStr;
                ArrRule[i].RightStr = ArrRule[ArrRule.size() - 1].RightStr;
                ArrRule.pop_back();
                break;
            }
        }
    }
}
//фнукция, удаляющая недостижиые символы
void DelNotReach(vector <Rule> &ArrRule, vector <Symbol> &ArrSymb)
{
    //поиск достижимых
    string Reach = "R";
    for (int i = 0; i < ArrRule.size(); i++)
    {
        if (Reach.find(ArrRule[i].LeftStr[0]) != -1)
        {
            for (int j = 0; j < ArrRule[i].RightStr.size(); j++)
            {
                if (((int)ArrRule[i].RightStr[j] >= 65) &&
((int)ArrRule[i].RightStr[j] <= 90))
                {
                    Reach += ArrRule[i].RightStr[j];
                }
            }
        }
    }
    //поиск недостижимых
    string NotReach;
    bool check = true;
    for (int i = 0; i < ArrRule.size(); i++)
    {
        if (Reach.find(ArrRule[i].LeftStr[0]) == -1)
        {
            if (NotReach.size() >= 1)
            {
```

```
                if (NotReach.find(ArrRule[i].LeftStr[0]) != -1)
                {
                    check = false;
                }
            }
            if (check)
            {
                NotReach += ArrRule[i].LeftStr[0];
            }
        }
    }
    //удаление недостижимых символов
    for (int i = 0; i < ArrSymb.size(); i++)
    {
        for (int j = 0; j < NotReach.size(); j++)
        {
            if (ArrSymb[i].Value == NotReach[j])
            {
                ArrSymb[i].Value = ArrSymb[ArrSymb.size() - 1].Value;
                ArrSymb[i].TermIndex = ArrSymb[ArrSymb.size() - 1].TermIndex;
                ArrSymb.pop_back();
                break;
            }
        }
    }
    //удаление недостижимых из правил
    for (int i = 0; i < ArrRule.size(); i++)
    {
        for (int j = 0; j < NotReach.size(); j++)
        {
            if ((ArrRule[i].LeftStr[0] == NotReach[j]) ||
(ArrRule[i].RightStr.find(NotReach[j]) != -1))
            {
                ArrRule[i].LeftStr = ArrRule[i+1].LeftStr;
                ArrRule[i].RightStr = ArrRule[i+1].RightStr;
                ArrRule[i + 1].LeftStr = ArrRule[ArrRule.size() - 1].LeftStr;
                ArrRule[i + 1].RightStr = ArrRule[ArrRule.size() - 1].RightStr;
                ArrRule.pop_back();
                break;
            }
        }
    }
}
//функция, удаляющая смешанные цепочки
void DelMix(vector <Rule> &ArrRule, vector <Symbol> &ArrSymb)
{
    bool ExistNT, ExistT, check;
    int count = 0;
    for (int i = 0; i < ArrRule.size(); i++)
    {
        ExistNT = false;
        ExistT = false;
        //определяем, смешанная цепочка или нет
        for (int j = 0; j < ArrRule[i].RightStr.size(); j++)
        {
            if (((int)ArrRule[i].RightStr[j] >= 65) &&
((int)ArrRule[i].RightStr[j] <= 90))
            {
                ExistNT = true;
            }
```

```
            else if (ArrRule[i].RightStr[j] != 'e')
            {
                ExistT = true;
            }
        }
        //удаляем(изменяем) смешанную цепочку
        string NewLeft = "ABC";
        check = false;
        if (ExistT && ExistNT)
        {
            for (int j = 0; j < ArrRule[i].RightStr.size(); j++)
            {
                if ((((int)ArrRule[i].RightStr[j] < 65) ||
((int)ArrRule[i].RightStr[j] > 90)) && (ArrRule[i].RightStr[j] != 'e'))
                {
                    for (int k = 0; k < ArrRule.size(); k++)
                    {
                        if ((ArrRule[k].LeftStr[0] == 'A' ||
ArrRule[k].LeftStr[0] == 'B' || ArrRule[k].LeftStr[0] == 'C')
                        && ArrRule[i].RightStr[j] == ArrRule[k].RightStr[0])
//если сущ такое правило, в для терм
                        {
                            check = true;
                            ArrRule[i].RightStr[j] = ArrRule[k].LeftStr[0];
                        }
                    }
                    if (!check)
                    {
                        ArrRule.resize(ArrRule.size() + 1);
                        ArrRule[ArrRule.size() - 1].LeftStr = NewLeft[count];
                        ArrRule[ArrRule.size() - 1].RightStr =
ArrRule[i].RightStr[j];
                        ArrRule[i].RightStr[j] = ArrRule[ArrRule.size() -
1].LeftStr[0];
                        count++;
                    }
                }
            }
        }
    }
}
//функция, удаляющая длинные цепочки
void DelLong(vector <Rule> &ArrRule, vector <Symbol> &ArrSymb)
{
    string NewLeft = "MN";
    int count = 0;
    bool check = false;
    for (int i = 0; i < ArrRule.size(); i++)
    {
            check = false;
            if (ArrRule[i].RightStr.size() > 2)
            {
                for (int k = 0; k < ArrRule.size(); k++)
                {
                    if ((ArrRule[k].LeftStr[0] == 'M' || ArrRule[k].LeftStr[0]
== 'N')
                        && (ArrRule[i].RightStr[0] == ArrRule[k].RightStr[0] &&
                            ArrRule[i].RightStr[1] == ArrRule[k].RightStr[1]))
                    {
                        check = true;
```

```
                        ArrRule[i].RightStr[0] = ArrRule[k].LeftStr[0];
                        ArrRule[i].RightStr[1] = ArrRule[i].RightStr[2];
                        ArrRule[i].RightStr.resize(ArrRule[i].RightStr.size() -
1);
                    }
                }
                if (!check) {
                    ArrRule.resize(ArrRule.size() + 1);
                    ArrRule[ArrRule.size() - 1].LeftStr = NewLeft[count];
                    ArrRule[ArrRule.size() - 1].RightStr.clear();
                    ArrRule[ArrRule.size() -
1].RightStr.append(ArrRule[i].RightStr, 0, 2);
                    ArrRule[i].RightStr[0] = ArrRule[ArrRule.size() -
1].LeftStr[0];
                    ArrRule[i].RightStr[1] = ArrRule[i].RightStr[2];
                    ArrRule[i].RightStr.resize(ArrRule[i].RightStr.size() - 1);
                    count = 1;
                }
            }
    }
}
//функция, удаляющая эпсилон-правила
void DelEps(vector <Rule> &ArrRule)
{
    for (int i = 0; i < ArrRule.size(); i++)
    {
        if (ArrRule[i].RightStr[0] == 'e' && ArrRule[i].LeftStr[0] != 'R')
        {
            ArrRule[i].RightStr = ArrRule[ArrRule.size() - 1].RightStr;
            ArrRule[i].LeftStr = ArrRule[ArrRule.size() - 1].LeftStr;
            ArrRule.pop_back();
        }
    }
}
int main()
{
    vector <Symbol> ArrSymb(11);
    vector <Rule> ArrRule(15);
    InitGrammar(ArrSymb, ArrRule);
    cout << endl;
    IsContFree(ArrRule);
    cout << endl;
    DelNotOut(ArrRule, ArrSymb);
    cout << "Удалены невыводимые символы. Грамматика:" << endl;
    PrintGrammar(ArrSymb, ArrRule);
    cout << endl;
    DelNotReach(ArrRule, ArrSymb);
    cout << "Удалены недостижимые символы. Грамматика:" << endl;
    PrintGrammar(ArrSymb, ArrRule);
    cout << endl;
    DelMix(ArrRule, ArrSymb);
    cout << "Удалены смешанные цепочки. Грамматика:" << endl;
    PrintGrammar(ArrSymb, ArrRule);
    cout << endl;
    DelLong(ArrRule, ArrSymb);
    cout << "Удалены длинные цепочки.Грамматика:" << endl;
    PrintGrammar(ArrSymb, ArrRule);
    cout << endl;
    DelEps(ArrRule);
    cout << "Удалены эпсилон-правила. Грамматика:" << endl;
```

```
    PrintGrammar(ArrSymb, ArrRule);
   return 0;
}
```