Varianta 5

Este dată gramatica independentă de context

G=(VN, VT, P, S,),

VN ={S, A, Y, D, X},

VT ={a, b, c, d, e},

P={

1. S → A

2. A → Y X

3. X → ε

4. X → c Y X

5. Y → a

6. Y → b

7. Y → d D

8. D → A e

}.

Să se construiască tabelul de analiză LL(1) şi să se analizeze şirul

12782546463453

**Dacbcbe ca**

S → A

A → Y X

Y → d D → d D X

. D → A e→ d A e X

A → Y X → d Y X e X

Y → a→ d a X e X

X → c Y X → d a c Y X e X

Y → b → d a c b X e X

. X → c Y X → d a c b c Y X e X

Y → b → d a c b c b X e X

X → ε → d a c b c b e X

X → c Y X → d a c b c b e c Y X

Y → a → d a c b c b e c a X

X → ε→ d a c b c b e c a

Codul:

#include <iostream>

#include <cstring>

#include <utility>

using namespace std;

/\*

1) O::=E

2) E::=XB

3) X::=StBeX

4) X::=B

5) S::=iv

6) B::=p

\*/

bool check(const char\* \_str)

{

char table[11][6] = {

//p t e i v $

{ 1, 0, 0, 1, 0, 0 }, //O

{ 2, 0, 0, 2, 0, 0 }, //E

{ 4, 0, 0, 3, 0, 0 }, //X

{ 0, 0, 0, 5, 0, 0 }, //S

{ 6, 0, 0, 0, 0, 0 }, //B

{ 7, 0, 0, 0, 0, 0 }, //p

{ 0, 7, 0, 0, 0, 0 }, //t

{ 0, 0, 7, 0, 0, 0 }, //e

{ 0, 0, 0, 7, 0, 0 }, //i

{ 0, 0, 0, 0, 7, 0 }, //v

{ 0, 0, 0, 0, 0, 8 } //$

},

str[50], stack[50] = "$O"/\*<<< стартовое правило - O\*/, lenta[50] = " ",

iset[12] = "OEXSBpteiv$", jset[7] = "pteiv$";

int x, ilenta = 0, istack = 1, istr = 0, i = 0, j = 0;

strcpy(str, \_str);

x = strlen(str);

str[x] = '$';

str[x + 1] = 0;

while (str[istr] && stack[0] && table[i][j])

{

//cout<<str[istr]<<endl;

for (j = 0; str[istr] != jset[j] && jset[j]; j++);

for (i = 0; stack[istack] != iset[i] && iset[i]; i++);

//cout<<i<<" "<<j<<endl;

if (j == 6)

break;

if (1 <= table[i][j] && table[i][j] <= 6) {

lenta[ilenta++] = table[i][j] + 48;

lenta[ilenta] = 0;

}

//cout<<lenta<<endl;

switch (table[i][j]) {

case 0:

break;

case 1:

stack[istack] = 'E';

stack[istack + 1] = 0;

break;

case 2:

stack[istack] = 'B';

stack[++istack] = 'X';

stack[istack + 1] = 0;

break;

case 3:

stack[istack] = 'X';

stack[++istack] = 'e';

stack[++istack] = 'B';

stack[++istack] = 't';

stack[++istack] = 'S';

stack[istack + 1] = 0;

break;

case 4:

stack[istack] = 'B';

stack[istack + 1] = 0;

break;

case 5:

stack[istack] = 'v';

stack[++istack] = 'i';

stack[istack + 1] = 0;

break;

case 6:

stack[istack] = 'p';

stack[istack + 1] = 0;

break;

case 7:

case 8:

stack[istack--] = 0;

istr++;

break;

default:

cout << "Error" << endl;

}

//cout<<stack<<endl;

}

bool result = !str[istr] && !stack[0];

if (result)

cout << "Верная строка" << endl;

else

cout << "Неверная строка" << endl

<< "Ошибка при обработке " << istr + 1 << "-го символа" << endl

<< "В стеке осталось:" << endl

<< stack << endl;

cout << "Выходная лента:" << endl

<< lenta << endl;

return result;

}

template<typename T, size\_t N>

void run(T(&\_array)[N])

{

int allTestsCount = 0;

int failCount = 0;

for (size\_t i = 0; i < N; ++i)

{

T& s = \_array[i];

allTestsCount++;

cout << "Строка: " << s.second << endl;

if (s.first != check(s.second))

failCount++;

cout << "----------------" << endl;

}

if (!failCount)

cout << "Все тесты (" << allTestsCount << ") пройдены успешно." << endl;

else

cout << "Провалено " << failCount << " из " << allTestsCount << " тестов." << endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

std::pair<bool,const char\*> tests[] = {

{true,"pp"},

{false,"pp2"},

{true,"ivtpepp"},

{true,"ivtpeivtpepp"},

{true,"ivtpeivtpeivtpepp"},

{false,"ivtpeivtpeivtpeppp"}

};

run(tests);

system("pause");

return 0;

}