LAB7: Compute First and follow sets of given grammar.

IT057-Kathiriya Darshakkumar

Code:

#include <bits/stdc++.h>

using namespace std;

bool cmp(string &x, string &y)

{

return x.size() > y.size();

}

string filter$(string dar)

{

for (int i = 0; i < dar.length(); i += 3)

{

for (int j = i + 3; j < dar.length(); j += 3)

{

if (dar.substr(i, 3) == dar.substr(j, 3))

{

dar = dar.substr(0, j) + dar.substr(j + 3);

}

}

}

return dar;

}

string filterNull(string dar)

{

dar = filter$(dar);

for (int i = 0; i < dar.length(); i += 3)

{

if(dar.substr(i,3)=="^, ")

dar = dar.substr(0,i) + dar.substr(i+3);

}

return dar;

}

void solve57()

{

string input[100][100];

string nonTerm[100];

int proCount[100];

int cnt;

cout << "Enter total number of expressions: ";

cin >>cnt;

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

{

cout << "Enter non-terminal: ";

cin >> nonTerm[i];

cout << "Enter number of productions: ";

cin >> proCount[i];

int j = 0;

for (j = 0; j < proCount[i]; j++)

{

cout << "Enter production " << j + 1 << " : ";

cin >> input[i][j];

}

sort(input[i], input[i] + j, cmp);

}

cout << "\nGiven Input Expression:\n";

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

{

cout << nonTerm[i] << " -> ";

cout << input[i][0];

for (int j = 1; j < proCount[i]; j++)

{

cout << " | " << input[i][j];

}

cout << endl;

}

string first[cnt];

for (int dm = 0; dm < cnt; ++dm)

{

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

{

for (int j = 0; j < proCount[i]; j++)

{

//check for null(^) production

int m = 0;

for (; m < input[i][j].length(); ++m)

{

int non\_t\_index = -1;

for (int n = 0; n < cnt; ++n)

{

if (input[i][j].substr(m, nonTerm[n].length()) == nonTerm[n])

{

non\_t\_index = n;

break;

}

}

if (non\_t\_index == -1)

{

first[i] = filter$(first[i] + input[i][j].substr(m, 1) + ", ");

break;

}

if (first[non\_t\_index].find('^') != -1)

{

first[i] = filter$(first[i] + first[non\_t\_index]);

}

else

{

first[i] = filter$(first[i] + first[non\_t\_index]);

break;

}

}

}

}

}

cout << "\nFirst-set of Given Expression:\n";

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

{

cout << nonTerm[i] << " -> ";

cout << first[i];

cout << endl;

}

string follow[cnt];

cout << follow[1];

if (cnt)

follow[0] = "$, ";

for (int dm = 0; dm < cnt; ++dm)

{

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

{

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

{

for (int k = 0; k < proCount[j]; ++k)

{

for (int l = 0; l < input[j][k].length(); ++l)

{

if (input[j][k].substr(l, nonTerm[i].length()) == nonTerm[i])

{

string next\_string = input[j][k].substr(l + nonTerm[i].length());

if (next\_string.length() == 0)

{

follow[i] = filterNull (follow[i] + follow[j]);

continue;

}

else

{

int m = 0;

for (; m < next\_string.length(); ++m)

{

int non\_t\_index = -1;

for (int n = 0; n < cnt; ++n)

{

if (next\_string.substr(m, nonTerm[n].length()) == nonTerm[n])

{

non\_t\_index = n;

break;

}

}

if (non\_t\_index == -1)

{

follow[i] = filterNull (follow[i] + next\_string.substr(m, 1) + ", ");

break;

}

if (first[non\_t\_index].find('^') != -1)

{

follow[i] = filterNull (follow[i] + first[non\_t\_index]);

continue;

}

else

{

follow[i] = filterNull (follow[i] + first[non\_t\_index]);

break;

}

}

if (m == next\_string.length())

{

follow[i] = filterNull(follow[i] + follow[j]);

}

}

}

}

}

}

}

}

cout << "\nFollow-set of Given Expression:\n";

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

{

cout << nonTerm[i] << " -> ";

cout << follow[i];

cout << endl;

}

}

int main()

{

int i=1;

while(i)

{

solve57();

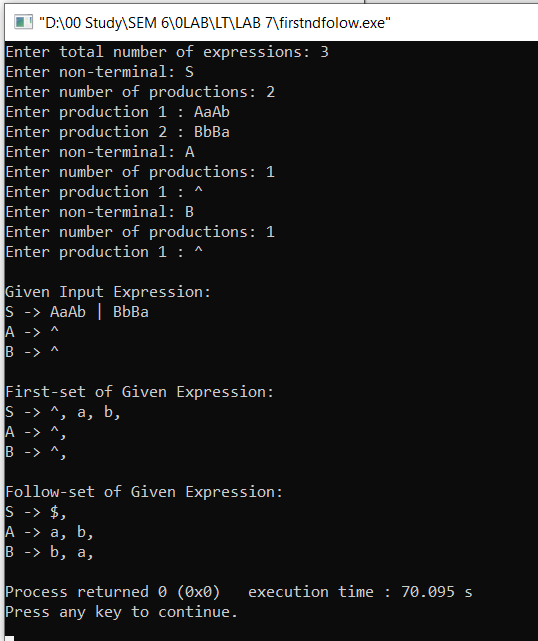
i--;

}

return 0;

}

Output1:



Output2:

