Universitatea Tehnică a Moldovei

Facultatea - Calculatoare Informatică și Microelectronică

Disciplina: *Limbaje Formale și Automate*

**Raport**

Lucrarea de laborator Nr.4

Tema: LL(1)

Varianta: 5

A efectuat: st.gr. TI-207 Bunescu Gabriel

A verificat: lect.univ. Duca Ludmila

Chișinău 2021

**Cuprins**

[I. Scopul lucrării: 3](#_Toc89086318)

[II. Date inițiale : 4](#_Toc89086319)

[III. First / Follow 4](#_Toc89086320)

[IV. Tabelul de analiză 4](#_Toc89086321)

[V. Analiza șirului 5](#_Toc89086322)

[VI. Verificarea în Jflap 5](#_Toc89086323)

[VII. Concluzie: 6](#_Toc89086324)

# I. Scopul lucrării:

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

II. Date inițiale :Este dată gramatica independentă de context

G=(VN, VT, P, S), VN ={S, A , X, Y, D}, VT ={a, b, c, d, e, },

P={ 1. S → A

2. A → YX

3. X → ε

4. X → cYX

5. Y→ a

6. Y → b

7. Y → dD

8. D → Ae

# III. First / Follow

Tabel 1 - Reprezentarea First & Follow

|  |  |  |
| --- | --- | --- |
|  | First | Follow |
| A | a,b,d | $, e |
| S | a,b,d | $ |
| X | c, ε | $, e |
| Y | a,b,d | $, c,e |
| D | a,b,d | $, c, e |

# IV. Tabelul de analiză

Tabel 2 - Reprezentarea tabelului de analiză

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | $ |
| S | S → A | S → A |  | S → A |  |  |
| A | A → YX | A → YX |  | A → YX |  |  |
| X |  |  | X → cYX |  | X → ε | X → ε |
| Y | Y→ a | Y→ b |  | Y → dD |  |  |
| D | D → Ae | D → Ae |  | D → Ae |  |  |

# V. Analiza șirului

|  |  |  |
| --- | --- | --- |
| **Stack** | **Input** | **Output** |
| $S | dacbcbeca$ |  |
| $A | dacbcbeca $ | S → A |
| $XY | dacbcbeca $ | A→ YX |
| $XDd | dacbcbeca $ | Y → dD |
| $XD | acbcbeca $ |  |
| $XeA | acbcbeca $ | D → Ae |
| $XeXY | acbcbeca $ | A→ YX |
| $XeXa | acbcbeca $ | Y→ a |
| $XeX | cbcbeca $ |  |
| $XeXYc | cbcbeca $ | X → cYX |
| $XeXY | bcbeca $ |  |
| $XeXb | bcbeca $ | Y→ b |
| $XeX | cbeca $ |  |
| $XeXYc | cbeca $ | X → cY X |
| $XeXY | beca $ |  |
| $XeXb | beca $ | Y→ b |
| $XeX | eca $ | X → ε |
| $Xe | eca $ |  |
| $X | ca $ |  |
| $XYc | ca $ | X → cY X |
| $XY | a $ | Y→ a |
| $Xa | a $ |  |
| $X | $ | X → ε |
| $ | $ |  |

# VI. Verificarea în Jflap

Figura 1. Introducerea producțiilor și obținerea tabelelor

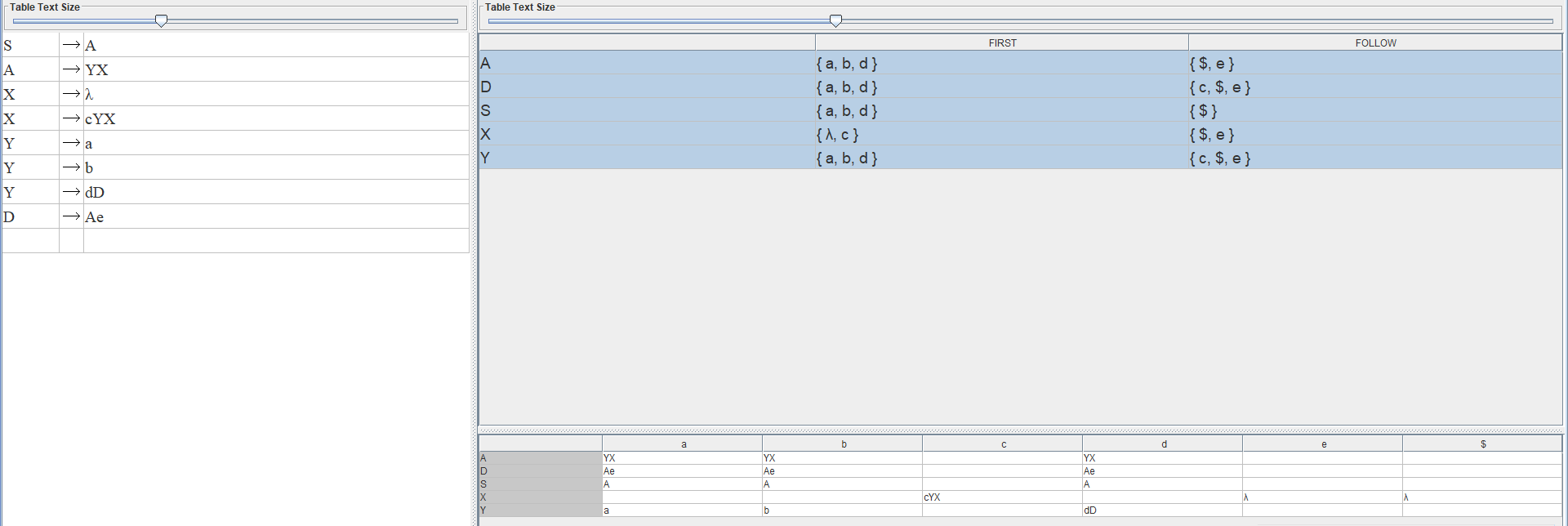
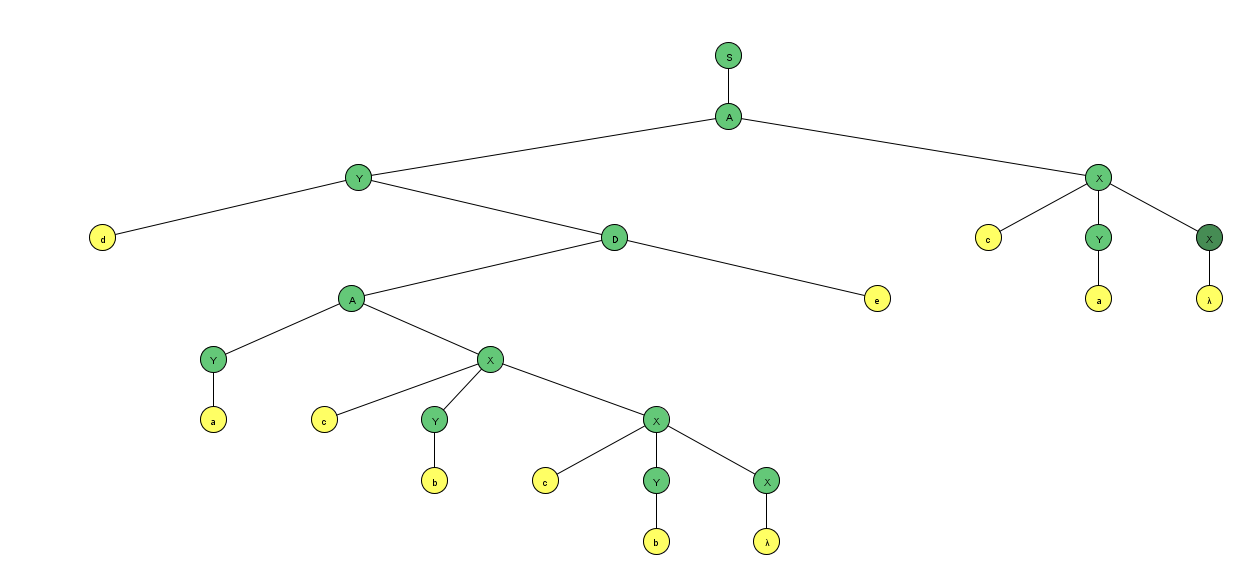
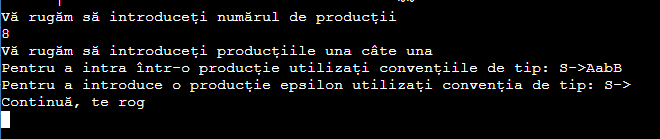
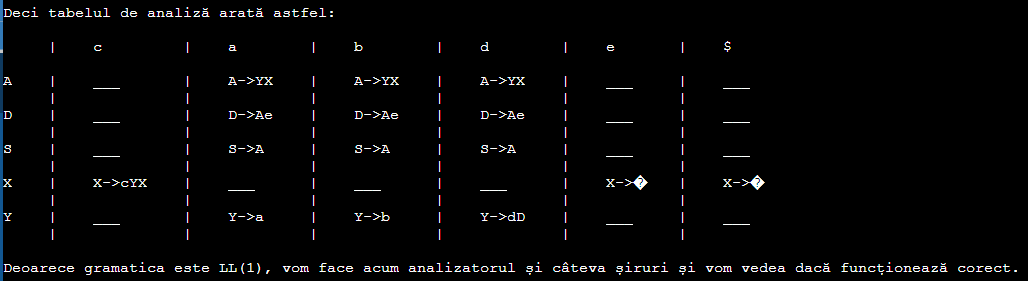
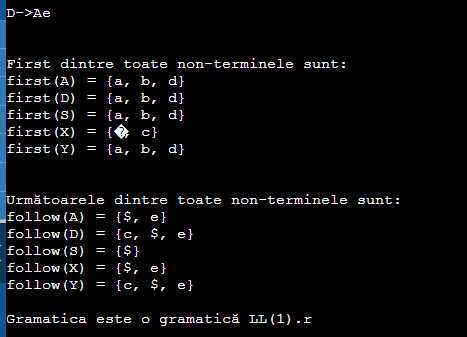
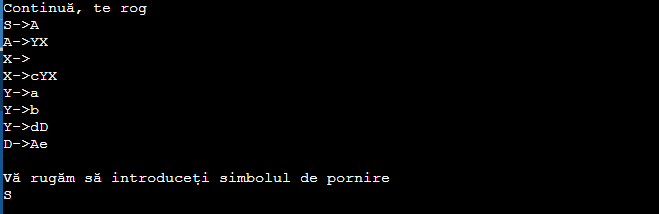
****

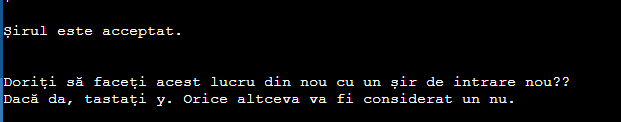
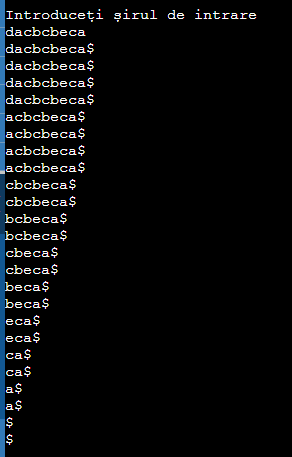
Figura 2. Verificarea acceptării cuvântului



**Executia din compilator:**

****





# VII. Concluzie:

În urma efectuării lucrării de laborator cu numărul 3, am antrenat lucrul cu automatele finite, însușind, individual, tema LL(1). Pentru o gramatică propusă, am identificat First și Follow, datorită cărui fapt, am realizat tabelul de analiză și a oferit posibilitatea de analiză a unui șir.

Spre final, afirm faptul că limbajele formale și automate oferă o gamă largă de posibilități, de realizări pe care noi le putem însuși.

**Codul sursa:**

**#include <iostream>**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**using namespace std;**

**char fol[26][20];**

**char first[26][20];**

**int \*fovi;**

**int \*\*rog;**

**char term[1000];**

**struct link\_list{**

**int pdno;**

**struct link\_list \* next;**

**};**

**struct link\_list \*\*papa[26];**

**int check\_amb(char \* right)**

**{**

**int i=3,count=0;**

**char uu=right[0];**

**while(right[i]!='\0')**

**{**

**if(right[i]==uu)**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**return count;**

**}**

**i++;**

**}**

**return 0;**

**}**

**int contains\_epsilon(char \* right)**

**{**

**int i=0;**

**char sub=157;**

**while(right[i]!='\0')**

**{**

**if(right[i]==sub)**

**{**

**return 1;**

**}**

**i++;**

**}**

**return 0;**

**}**

**int contains\_char(char \* right,char a)**

**{**

**int i=0;**

**while(right[i]!='\0')**

**{**

**if(right[i]==a)**

**{**

**return 1;**

**}**

**i++;**

**}**

**return 0;**

**}**

**void focal(char \* right,int j,int ki)**

**{**

**int co=right[j],yup=j;**

**char k=right[j],sub=157;**

**j++;**

**int count=0;**

**int flag=0;**

**char temp[20];**

**int ny,ky,flagch;**

**while(right[j]!='\0')**

**{**

**if(flag==0)**

**{**

**if(right[j]<65 || right[j]>90)**

**{**

**flag=1;**

**if(!(contains\_char(fol[co-65],right[j])))**

**{**

**temp[0]=right[j];**

**temp[1]='\0';**

**strcat(fol[co-65],temp);**

**}**

**}**

**else**

**{**

**if(!(contains\_epsilon(first[right[j]-65])))**

**{**

**flag=1;**

**}**

**if(k!=right[j])**

**{**

**ky=0;**

**while(first[right[j]-65][ky]!='\0')**

**{**

**if(first[right[j]-65][ky]!=sub)**

**{**

**ny=0;**

**flagch=0;**

**while(fol[co-65][ny]!='\0')**

**{**

**if(first[right[j]-65][ky]==fol[co-65][ny])**

**{**

**flagch=1;**

**break;**

**}**

**ny++;**

**}**

**if(flagch==0)**

**{**

**temp[0]=first[right[j]-65][ky];**

**temp[1]='\0';**

**strcat(fol[co-65],temp);**

**}**

**}**

**ky++;**

**}**

**}**

**}**

**}**

**if(right[j]>=65 && right[j]<=90)**

**{**

**count++;**

**if(count==1)**

**{**

**focal(right,j,ki);**

**if(flag==1)**

**{**

**return;**

**}**

**}**

**}**

**j++;**

**}**

**if(flag==0)**

**{**

**fovi[ki]=1;**

**rog[ki][yup]=1;**

**ky=0;**

**while(fol[right[0]-65][ky]!='\0')**

**{**

**ny=0;**

**flagch=0;**

**while(fol[co-65][ny]!='\0')**

**{**

**if(fol[right[0]-65][ky]==fol[co-65][ny])**

**{**

**flagch=1;**

**break;**

**}**

**ny++;**

**}**

**if(flagch==0)**

**{**

**temp[0]=fol[right[0]-65][ky];**

**temp[1]='\0';**

**strcat(fol[co-65],temp);**

**}**

**ky++;**

**}**

**}**

**return;**

**}**

**void follow\_up(char \* right, int co)**

**{**

**int ny,ky,flagch;**

**char temp[2];**

**ky=0;**

**while(fol[right[0]-65][ky]!='\0')**

**{**

**ny=0;**

**flagch=0;**

**while(fol[co-65][ny]!='\0')**

**{**

**if(fol[right[0]-65][ky]==fol[co-65][ny])**

**{**

**flagch=1;**

**break;**

**}**

**ny++;**

**}**

**if(flagch==0)**

**{**

**temp[0]=fol[right[0]-65][ky];**

**temp[1]='\0';**

**strcat(fol[co-65],temp);**

**}**

**ky++;**

**}**

**return;**

**}**

**int main()**

**{**

**while(1)**

**{**

**int i,n,count,pro,l,m,j,flag=0,p,flagger=0,lu;**

**int arr[26][2]={0,0};**

**arr[0][0]=65;**

**for(i=0;i<25;i++)**

**{**

**arr[i+1][0]=arr[i][0]+1;**

**}**

**int ambig[26]={0};**

**char k,sub,yo,no,uu,subr,starts;**

**//Input. And Preprocessing.**

**printf("Vă rugăm să introduceți numărul de producții\n");**

**scanf("%d",&n);**

**pro=n;**

**char input[10\*n][20],temp[20];**

**term[0]='\0';**

**printf("Vă rugăm să introduceți producțiile una câte una\n");**

**printf("Pentru a intra într-o producție utilizați convențiile de tip: S->AabB\n");**

**printf("Pentru a introduce o producție epsilon utilizați convenția de tip: S->\n");**

**printf("Continuă, te rog\n");**

**term[0]='\0';**

**for(i=0;i<n;i++)**

**{**

**scanf("%s",input[i]);**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(input[i][j]<65 || input[i][j]>90)**

**{**

**flag=0;**

**p=0;**

**while(term[p]!='\0')**

**{**

**if(term[p]==input[i][j])**

**{**

**flag=1;**

**break;**

**}**

**p++;**

**}**

**if(flag==0)**

**{**

**temp[0]=input[i][j];**

**temp[1]='\0';**

**strcat(term,temp);**

**}**

**}**

**j++;**

**}**

**if(strlen(input[i])==3)**

**{**

**sub=157;**

**temp[1]='\0';**

**temp[0]=sub;**

**strcat(input[i],temp);**

**}**

**l=input[i][0];**

**if(arr[l-65][1]==0)**

**{**

**arr[l-65][1]=1;**

**}**

**}**

**flag=0;**

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

**{**

**if(arr[i][1]==0)**

**{**

**l=i+65;**

**break;**

**}**

**}**

**//Verificați dacă este un CFG. Și verificați și Gramatica.**

**flag=0;**

**for(i=0;i<n;i++)**

**{**

**if(!(input[i][0]>=65 && input[i][0]<=90 && input[i][1]=='-' && input[i][2]=='>'))**

**{**

**flag=1;**

**break;**

**}**

**}**

**if(flag==1)**

**{**

**printf("\n\nHei Haide!!! Gramatica nu este nici macar un CFG!! Ce ar trebui să fac?");**

**printf("Vrei sa faci asta din nou??\n");**

**printf("Dacă da, tastați y. Orice altceva va fi considerat un nu.\n");**

**scanf("%c",&k);**

**scanf("%c",&k);**

**printf("\n\n");**

**if(k=='y')**

**continue;**

**else**

**break;**

**}**

**printf("\nVă rugăm să introduceți simbolul de pornire\n");**

**scanf("%c",&k);**

**while(1)**

**{**

**scanf("%c",&k);**

**p=k;**

**if(p>=65 && p<=90)**

**{**

**if(arr[p-65][1]==1)**

**{**

**starts=k;**

**break;**

**}**

**else**

**{**

**printf("Acest simbol nu este disponibil. Incearca din nou, te rog.\n");**

**}**

**}**

**else**

**{**

**printf("Acest simbol este un terminal. Incearca din nou, te rog.\n");**

**}**

**scanf("%c",&k);**

**}**

**//Ambiguitate Verificați și eliminați.**

**printf("\n\n");**

**int couri,brains;**

**for(i=0;i<pro;i++)**

**{**

**if(ambig[input[i][0]-65]==0)**

**{**

**couri=0;**

**ambig[input[i][0]-65]=1;**

**uu=input[i][0];**

**for(j=0;j<pro;j++)**

**{**

**if(input[j][0]==uu)**

**{**

**if(check\_amb(input[j])==2)**

**{**

**couri++;**

**}**

**}**

**}**

**if(couri==0)**

**{**

**}**

**else if(couri==1)**

**{**

**brains=i;**

**for(i=0;i<pro;i++)**

**{**

**if(input[i][0]!=uu)**

**continue;**

**j=0;**

**count=0;**

**k=input[i][0];**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**sub=l;**

**arr[l-65][1]=1;**

**while(1)**

**{**

**if(arr[l-65][1]==0)**

**{**

**break;**

**}**

**else**

**{**

**l++;**

**}**

**}**

**break;**

**}**

**j++;**

**}**

**j=3;**

**if(count==2)**

**{**

**count=0;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**flag=1;**

**input[i][j]=sub;**

**for(m=0;m<pro;m++)**

**{**

**if(i!=m && input[m][0]==k)**

**{**

**input[m][0]=sub;**

**}**

**}**

**count--;**

**break;**

**}**

**j++;**

**}**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**input[i][j]=sub;**

**count--;**

**}**

**j++;**

**}**

**strcpy(input[pro],input[i]);**

**temp[1]='\0';**

**temp[0]=sub;**

**strcpy(input[pro]+3,temp);**

**pro++;**

**}**

**}**

**i=brains;**

**}**

**else**

**{**

**brains=i;**

**sub=l;**

**arr[l-65][1]=1;**

**while(1)**

**{**

**if(arr[l-65][1]==0)**

**{**

**break;**

**}**

**else**

**{**

**l++;**

**}**

**}**

**subr=l;**

**arr[l-65][1]=1;**

**while(1)**

**{**

**if(arr[l-65][1]==0)**

**{**

**break;**

**}**

**else**

**{**

**l++;**

**}**

**}**

**for(i=0;i<pro;i++)**

**{**

**if(input[i][0]!=uu)**

**continue;**

**j=0;**

**count=0;**

**k=input[i][0];**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**break;**

**}**

**j++;**

**}**

**if(count!=2)**

**{**

**input[i][0]=subr;**

**}**

**}**

**for(i=0;i<pro;i++)**

**{**

**if(input[i][0]!=uu)**

**continue;**

**j=0;**

**count=0;**

**k=input[i][0];**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**break;**

**}**

**j++;**

**}**

**j=3;**

**if(count==2)**

**{**

**if(couri==1)**

**{**

**count=0;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**flag=1;**

**input[i][j]=subr;**

**count--;**

**}**

**j++;**

**}**

**j=3;**

**while(input[i][j]!='0')**

**{**

**if(k==input[i][j])**

**{**

**input[i][j]=sub;**

**break;**

**}**

**j++;**

**}**

**input[i][0]=sub;**

**strcpy(input[pro],input[i]);**

**temp[1]='\0';**

**temp[0]=subr;**

**strcpy(input[pro]+3,temp);**

**pro++;**

**strcpy(input[pro],input[i]);**

**input[pro][0]=k;**

**temp[1]='\0';**

**temp[0]=sub;**

**strcpy(input[pro]+3,temp);**

**pro++;**

**}**

**else**

**{**

**count=0;**

**while(input[i][j]!='\0')**

**{**

**if(k==input[i][j])**

**{**

**count++;**

**}**

**if(count==2)**

**{**

**flag=1;**

**input[i][j]=sub;**

**count--;**

**}**

**j++;**

**}**

**couri--;**

**}**

**count=0;**

**}**

**}**

**i=brains;**

**}**

**}**

**}**

**if(flag==0)**

**{**

**printf("Fără ambiguitate\n");**

**}**

**else**

**{**

**printf("\n\nProducțiile după eliminarea ambiguității sunt:\n");**

**for(i=0;i<pro;i++)**

**{**

**printf("%s\n",input[i]);**

**}**

**}**

**printf("\n\n");**

**//Verificați recursiunea stângă aici.**

**flag=0;**

**for(i=0;i<pro;i++)**

**{**

**if(input[i][0]==input[i][3])**

**{**

**flagger=1;**

**count=0;**

**sub=l;**

**temp[1]='\0';**

**temp[0]=sub;**

**count=0;**

**for(j=0;j<pro;j++)**

**{**

**if(i!=j && input[i][0]==input[j][0])**

**{**

**strcat(input[j],temp);**

**count++;**

**}**

**}**

**if(count!=0)**

**{**

**strcpy(temp,input[i]+4);**

**input[i][0]=sub;**

**strcpy(input[i]+3,temp);**

**temp[1]='\0';**

**temp[0]=sub;**

**strcat(input[i],temp);**

**strcpy(input[pro],temp);**

**strcat(input[pro],"-");**

**strcat(input[pro],">");**

**sub=157;**

**temp[1]='\0';**

**temp[0]=sub;**

**strcat(input[pro],temp);**

**pro++;**

**arr[l-65][1]=1;**

**while(1)**

**{**

**if(arr[l-65][1]==0)**

**{**

**break;**

**}**

**else**

**{**

**l++;**

**}**

**}**

**}**

**}**

**}**

**if(flagger==0)**

**{**

**printf("Gramatica nu avea recursivitate lăsată\n");**

**}**

**else**

**{**

**//Din nou un pic de procesare a datelor.**

**sub=157;**

**for(i=0;i<pro;i++)**

**{**

**if(input[i][3]==sub && input[i][4]!='\0')**

**{**

**strcpy(input[i]+3,input[i]+4);**

**}**

**}**

**printf("Producțiile după eliminarea recursiunii stângi sunt:\n");**

**for(i=0;i<pro;i++)**

**{**

**printf("%s\n",input[i]);**

**}**

**}**

**printf("\n\n");**

**//Factorizarea stângă**

**flag=0;**

**i=0;**

**while(i<pro-1)**

**{**

**count=0;**

**yo=input[i][0];**

**no=input[i][3];**

**for(j=i+1;j<pro;j++)**

**{**

**if(input[j][0]==yo && input[j][3]==no)**

**{**

**sub=l;**

**count++;**

**input[j][0]=sub;**

**strcpy(input[j]+3,input[j]+4);**

**}**

**}**

**if(count>0)**

**{**

**flag=1;**

**strcpy(input[pro],input[i]);**

**input[pro][0]=sub;**

**strcpy(input[pro]+3,input[i]+4);**

**temp[0]=sub;**

**temp[1]='\0';**

**strcpy(input[i]+4,temp);**

**pro++;**

**arr[l-65][1]=1;**

**while(1)**

**{**

**if(arr[l-65][1]==0)**

**{**

**break;**

**}**

**else**

**{**

**l++;**

**}**

**}**

**}**

**i++;**

**}**

**if(flag==1)**

**{**

**sub=157;**

**temp[1]='\0';**

**temp[0]=sub;**

**for(i=0;i<pro;i++)**

**{**

**if(strlen(input[i])==3)**

**{**

**strcat(input[i],temp);**

**}**

**}**

**printf("Gramatica mobilată după factoring stânga este:\n");**

**}**

**else**

**{**

**printf("Gramatica este deja luată în considerare\n\n");**

**printf("Cu toate acestea, gramatica finală este:\n");**

**}**

**for(i=0;i<pro;i++)**

**{**

**printf("%s\n",input[i]);**

**}**

**//First Calculation.**

**printf("\n\n");**

**char ad[50],gu;**

**int fco[26][2];**

**int it=0,nit,ky,sumnit,ny,flagch,kar;**

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

**{**

**fco[i][0]=0;**

**fco[i][1]=0;**

**temp[0]='\0';**

**strcpy(first[i],temp);**

**}**

**for(i=0;i<pro;i++)**

**{**

**if(!(input[i][3]>=65 && input[i][3]<=90))**

**{**

**fco[input[i][0]-65][0]=1;**

**temp[1]='\0';**

**temp[0]=input[i][3];**

**strcat(first[input[i][0]-65],temp);**

**}**

**}**

**it=0,nit=1;**

**sub=157;**

**while(1)**

**{**

**sumnit=0;**

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

**{**

**if(fco[i][it]==1)**

**{**

**uu=i+65;**

**for(j=0;j<pro;j++)**

**{**

**if(input[j][3]==uu)**

**{**

**strcpy(ad,first[input[j][0]-65]);**

**ky=0;**

**while(first[i][ky]!='\0')**

**{**

**ny=0;**

**flagch=0;**

**while(first[input[j][0]-65][ny]!='\0')**

**{**

**if(first[input[j][0]-65][ny]==first[i][ky])**

**{**

**flagch=1;**

**break;**

**}**

**ny++;**

**}**

**if(flagch==0)**

**{**

**flag=0;**

**k=157;**

**if(first[i][ky]==k)**

**{**

**kar=3;**

**while(input[j][kar+1]!='\0')**

**{**

**if(input[j][kar]>=65 && input[j][kar]<=90)**

**{**

**if(!(contains\_epsilon(first[input[j][kar]-65])))**

**{**

**flag=1;**

**break;**

**}**

**}**

**else**

**{**

**flag=1;**

**break;**

**}**

**kar++;**

**}**

**if(flag==0)**

**{**

**if(input[j][kar]>=65 && input[j][kar]<=90)**

**{**

**if(contains\_epsilon(first[input[j][kar]-65]))**

**{**

**temp[0]=first[i][ky];**

**temp[1]='\0';**

**strcat(first[input[j][0]-65],temp);**

**fco[input[j][0]-65][nit]=1;**

**}**

**}**

**}**

**}**

**else**

**{**

**temp[0]=first[i][ky];**

**temp[1]='\0';**

**strcat(first[input[j][0]-65],temp);**

**fco[input[j][0]-65][nit]=1;**

**}**

**}**

**ky++;**

**}**

**if(strcmp(ad,first[input[j][0]-65]))**

**{**

**sumnit++;**

**}**

**}**

**}**

**}**

**}**

**if(sumnit==0)**

**{**

**break;**

**}**

**it=!it;**

**nit=!it;**

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

**{**

**fco[i][nit]=0;**

**}**

**}**

**k=157;**

**for(i=0;i<pro;i++)**

**{**

**j=3;**

**while(input[i][j]>=65 && input[i][j]<=90)**

**{**

**if(contains\_epsilon(first[input[i][j]-65]))**

**{**

**if(input[i][j+1]!='\0')**

**{**

**if(input[i][j+1]<65 || input[i][j+1]>90)**

**{**

**if(!(contains\_char(first[input[i][0]-65],input[i][j+1])))**

**{**

**temp[0]=input[i][j+1];**

**temp[1]='\0';**

**strcat(first[input[i][0]-65],temp);**

**}**

**}**

**else**

**{**

**uu=input[i][j+1];**

**gu=input[i][0];**

**ky=0;**

**while(first[uu-65][ky]!='\0')**

**{**

**ny=0;**

**flagch=0;**

**while(first[gu-65][ny]!='\0')**

**{**

**if(first[gu-65][ny]==first[uu-65][ky])**

**{**

**flagch=1;**

**break;**

**}**

**ny++;**

**}**

**if(flagch==0)**

**{**

**if(first[uu-65][ky]!=k)**

**{**

**temp[0]=first[uu-65][ky];**

**temp[1]='\0';**

**strcat(first[gu-65],temp);**

**}**

**}**

**ky++;**

**}**

**}**

**}**

**}**

**else**

**{**

**break;**

**}**

**j++;**

**}**

**}**

**printf("First dintre toate non-terminele sunt: \n");**

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

**{**

**if(strlen(first[i]))**

**{**

**uu=i+65;**

**printf("first(%c) = {",uu);**

**ny=0;**

**printf("%c",first[i][ny]);**

**ny++;**

**while(first[i][ny]!='\0')**

**{**

**printf(", %c",first[i][ny]);**

**ny++;**

**}**

**printf("}\n");**

**}**

**}**

**printf("\n\n");**

**//Follow Computation.**

**////Unele preprocesare înainte de calcul Follow.**

**fovi=(int \*) malloc(pro\*sizeof(int));**

**rog=(int \*\*) malloc(pro\*sizeof(int \*));**

**for(i=0; i<pro; i++)**

**{**

**rog[i]=(int \*)malloc(20 \* sizeof(int));**

**}**

**for(i=0;i<pro;i++)**

**{**

**for(j=0;j<20;j++)**

**{**

**rog[i][j]=0;**

**}**

**}**

**for(i=0;i<pro;i++)**

**{**

**fovi[i]=0;**

**}**

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

**{**

**temp[0]='\0';**

**strcpy(fol[i],temp);**

**}**

**temp[0]=36;**

**temp[1]='\0';**

**strcpy(fol[starts-65],temp);**

**for(i=0;i<pro;i++)**

**{**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(input[i][j]>=65 && input[i][j]<=90)**

**{**

**focal(input[i],j,i);**

**break;**

**}**

**j++;**

**}**

**}**

**count=0;**

**for(i=0;i<pro;i++)**

**{**

**if(fovi[i]==1)**

**{**

**count++;**

**}**

**}**

**char foldo[count][20];**

**p=0;**

**for(i=0;i<pro;i++)**

**{**

**if(fovi[i]==1)**

**{**

**strcpy(foldo[p],fol[input[i][0]-65]);**

**p++;**

**}**

**}**

**for(i=0;i<pro;i++)**

**{**

**if(fovi[i]==1)**

**{**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(rog[i][j]==1)**

**{**

**follow\_up(input[i],input[i][j]);**

**}**

**j++;**

**}**

**}**

**}**

**while(1)**

**{**

**p=0;**

**count=0;**

**for(i=0;i<pro;i++)**

**{**

**if(fovi[i]==1)**

**{**

**if(strcmp(foldo[p],fol[input[i][0]-65]))**

**{**

**j=3;**

**while(input[i][j]!='\0')**

**{**

**if(rog[i][j]==1)**

**{**

**strcpy(foldo[p],fol[input[i][0]-65]);**

**follow\_up(input[i],input[i][j]);**

**}**

**j++;**

**}**

**count++;**

**}**

**p++;**

**}**

**}**

**if(count==0)**

**break;**

**}**

**printf("Următoarele dintre toate non-terminele sunt:\n");**

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

**{**

**if(strlen(fol[i]))**

**{**

**uu=i+65;**

**printf("follow(%c) = {",uu);**

**ny=0;**

**printf("%c",fol[i][ny]);**

**ny++;**

**while(fol[i][ny]!='\0')**

**{**

**printf(", %c",fol[i][ny]);**

**ny++;**

**}**

**printf("}\n");**

**}**

**}**

**//Realizarea tabelului de analiză.**

**//Mai întâi, o mulțime de preprocesare.**

**int terl=strlen(term);**

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

**{**

**if(arr[i][1]==0)**

**{**

**papa[i]=NULL;**

**}**

**else**

**{**

**papa[i]=(struct link\_list \*\*) malloc(terl\*sizeof(struct link\_list \*));**

**for(j=0;j<terl;j++)**

**{**

**papa[i][j]=NULL;**

**}**

**}**

**}**

**struct link\_list \*fookat;**

**struct link\_list \*stepdad=(struct link\_list \*) malloc(26\*sizeof(struct link\_list));**

**for(j=0;j<26;j++)**

**{**

**stepdad[j].pdno=0;**

**stepdad[j].next=NULL;**

**}**

**int breaker;**

**for(i=0;i<pro;i++)**

**{**

**for(j=0;j<terl;j++)**

**{**

**p=3;**

**flag=0;**

**breaker=0;**

**while(input[i][p]!='\0')**

**{**

**if(input[i][p]<65 || input[i][p]>90)**

**{**

**if(input[i][p]==term[j])**

**{**

**flag=1;**

**}**

**breaker=1;**

**}**

**else**

**{**

**if(contains\_char(first[input[i][p]-65],term[j]))**

**{**

**flag=1;**

**breaker;**

**}**

**else**

**{**

**if(!contains\_epsilon(first[input[i][p]-65]))**

**{**

**breaker=1;**

**}**

**}**

**}**

**if(flag==1)**

**{**

**if(papa[input[i][0]-65][j]==NULL)**

**{**

**papa[input[i][0]-65][j]=(struct link\_list \*) malloc(sizeof(struct link\_list));**

**papa[input[i][0]-65][j]->pdno=i;**

**papa[input[i][0]-65][j]->next=NULL;**

**}**

**else**

**{**

**fookat=(struct link\_list \*) malloc(sizeof(struct link\_list));**

**fookat->next=papa[input[i][0]-65][j];**

**fookat->pdno=i;**

**papa[input[i][0]-65][j]=fookat;**

**}**

**break;**

**}**

**if(breaker==1)**

**{**

**break;**

**}**

**p++;**

**}**

**}**

**}**

**//Imprimarea tabelului parser așa cum este în caiet.**

**//Verificați dacă este sau nu o gramatică LL1.**

**flag=0;**

**sub = 157;**

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

**{**

**if(papa[i]!=NULL)**

**{**

**flagch=0;**

**if(contains\_epsilon(first[i]))**

**{**

**flagch=1;**

**}**

**for(j=0;j<terl;j++)**

**{**

**count=0;**

**fookat=papa[i][j];**

**while(fookat!=NULL)**

**{**

**fookat=fookat->next;**

**count++;**

**}**

**if(count>1)**

**{**

**flag=1;**

**break;**

**}**

**else**

**{**

**if(count==1 && flagch && contains\_char(fol[i],term[j]))**

**{**

**flag=1;**

**break;**

**}**

**}**

**}**

**}**

**if(flag==1)**

**{**

**break;**

**}**

**}**

**if(flag==0)**

**{**

**printf("\nGramatica este o gramatică LL(1).r\n\n");**

**}**

**else**

**{**

**printf("\nGramatica nu este o gramatică LL(1).\n\n");**

**}**

**printf("\n\nDeci tabelul de analiză arată astfel:\n\n");**

**int max[terl];**

**for(i=0;i<terl;i++)**

**{**

**max[i]=3;**

**}**

**if(flag==0)**

**{**

**sub=36;**

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

**{**

**if(papa[i]!=NULL)**

**{**

**flag=0;**

**if(contains\_epsilon(first[i]))**

**{**

**flag=1;**

**}**

**for(j=0;j<terl;j++)**

**{**

**if(papa[i][j]==NULL)**

**{**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**if(max[j]<4)**

**{**

**max[j]=4;**

**}**

**}**

**}**

**else**

**{**

**fookat=papa[i][j];**

**while(fookat!=NULL)**

**{**

**if(max[j]<strlen(input[fookat->pdno]))**

**{**

**max[j]=strlen(input[fookat->pdno]);**

**}**

**fookat=fookat->next;**

**}**

**}**

**}**

**}**

**}**

**printf(" ");**

**for(j=0;j<terl;j++)**

**{**

**printf(" | ");**

**printf("%c",term[j]);**

**for(p=0;p<max[j]-1;p++)**

**{**

**printf(" ");**

**}**

**}**

**printf(" | ");**

**printf("%c",36);**

**printf("\n\n");**

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

**{**

**if(papa[i]!=NULL)**

**{**

**flag=0;**

**if(contains\_epsilon(first[i]))**

**{**

**flag=1;**

**}**

**printf("%c",i+65);**

**for(j=0;j<terl;j++)**

**{**

**if(papa[i][j]==NULL)**

**{**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**printf(" | ");**

**printf("%c->%c",i+65,157);**

**papa[i][j]=(struct link\_list \*) malloc(sizeof(struct link\_list));**

**papa[i][j]->pdno=-1;**

**papa[i][j]->next=NULL;**

**for(p=0;p<max[j]-4;p++)**

**{**

**printf(" ");**

**}**

**}**

**else**

**{**

**printf(" | ");**

**printf("\_\_\_");**

**for(p=0;p<max[j]-3;p++)**

**{**

**printf(" ");**

**}**

**}**

**}**

**else**

**{**

**fookat=papa[i][j];**

**while(fookat!=NULL)**

**{**

**printf(" | ");**

**printf("%s",input[fookat->pdno]);**

**for(p=0;p<max[j]-strlen(input[fookat->pdno]);p++)**

**{**

**printf(" ");**

**}**

**fookat=fookat->next;**

**}**

**}**

**}**

**if(flag==1 && contains\_char(fol[i],sub))**

**{**

**printf(" | ");**

**printf("%c->%c",i+65,157);**

**stepdad[i].pdno=-1;**

**}**

**else**

**{**

**printf(" | ");**

**printf("\_\_\_");**

**}**

**printf("\n");**

**printf(" ");**

**for(j=0;j<terl;j++)**

**{**

**printf(" | ");**

**for(p=0;p<max[j];p++)**

**{**

**printf(" ");**

**}**

**}**

**printf(" | ");**

**printf("\n");**

**}**

**}**

**printf("\nDeoarece gramatica este LL(1), vom face acum analizatorul și câteva șiruri și vom vedea dacă funcționează corect.\n\n");**

**while(1)**

**{**

**//Preluarea intrării șirului. Preprocesarea șirului.**

**char str[1000];**

**printf("\nIntroduceți șirul de intrare\n");**

**scanf("%s",str);**

**sub=36;**

**temp[0]=sub;**

**temp[1]='\0';**

**strcat(str,temp);**

**//Începe codul pentru parser.**

**char stack[100];**

**int sp;**

**int head=0;**

**temp[1]=starts;**

**temp[2]='\0';**

**strcpy(stack,temp);**

**flag=0;**

**while(str[head]!='\0')**

**{**

**sp=strlen(stack)-1;**

**if(stack[sp]<65 || stack[sp]>90)**

**{**

**if(str[head]!=stack[sp])**

**{**

**flag=1;**

**}**

**else**

**{**

**for(int i = head; i < 11; i++) {**

**cout << str[i];**

**}**

**cout << "\n";**

**head++;**

**stack[sp]='\0';**

**sp=strlen(stack)-1;**

**}**

**}**

**else**

**{**

**if(str[head]==sub)**

**{**

**if(stepdad[stack[sp]-65].pdno==-1)**

**{**

**stack[sp]='\0';**

**for(int i = head; i < 11; i++) {**

**cout << str[i];**

**}**

**cout << "\n";**

**sp=strlen(stack)-1;**

**}**

**else**

**{**

**flag=1;**

**}**

**}**

**else**

**{**

**k=str[head];**

**j=0;**

**while(term[j]!='\0')**

**{**

**if(term[j]==k)**

**break;**

**j++;**

**}**

**if(j==strlen(term))**

**{**

**flag=1;**

**}**

**else**

**{**

**if(papa[stack[sp]-65][j]==NULL)**

**{**

**flag=1;**

**}**

**else if(papa[stack[sp]-65][j]->pdno==-1)**

**{**

**stack[sp]='\0';**

**for(int i = head; i < 11; i++) {**

**cout << str[i];**

**}**

**cout << "\n";**

**sp=strlen(stack)-1;**

**}**

**else**

**{**

**i=papa[stack[sp]-65][j]->pdno;**

**stack[sp]='\0';**

**j=strlen(input[i])-1;**

**temp[1]='\0';**

**while(j>2)**

**{**

**temp[0]=input[i][j];**

**strcat(stack,temp);**

**j--;**

**}**

**for(int i = head; i < 11; i++) {**

**cout << str[i];**

**}**

**cout << "\n";**

**sp=strlen(stack)-1;**

**}**

**}**

**}**

**}**

**if(flag==1)**

**{**

**break;**

**}**

**}**

**if(flag==1)**

**{**

**printf("\nGramatica dată nu generează acest șir\n");**

**}**

**else**

**{**

**printf("\nȘirul este acceptat.\n");**

**}**

**printf("\n\nDoriți să faceți acest lucru din nou cu un șir de intrare nou??\n");**

**printf("Dacă da, tastați y. Orice altceva va fi considerat un nu.\n");**

**scanf("%c",&k);**

**scanf("%c",&k);**

**printf("\n\n");**

**if(k=='y')**

**continue;**

**else**

**break;**

**}**

**}**

**else**

**{**

**int imax[26];**

**int hope[26][terl];**

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

**{**

**imax[i]=1;**

**}**

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

**{**

**if(papa[i]!=NULL)**

**{**

**flag=0;**

**if(contains\_epsilon(first[i]))**

**{**

**flag=1;**

**}**

**for(j=0;j<terl;j++)**

**{**

**if(papa[i][j]==NULL)**

**{**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**if(max[j]<4)**

**{**

**max[j]=4;**

**}**

**}**

**hope[i][j]=1;**

**}**

**else**

**{**

**count=0;**

**fookat=papa[i][j];**

**while(fookat!=NULL)**

**{**

**if(max[j]<strlen(input[fookat->pdno]))**

**{**

**max[j]=strlen(input[fookat->pdno]);**

**}**

**count++;**

**fookat=fookat->next;**

**}**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**count++;**

**}**

**hope[i][j]=count;**

**if(imax[i]<count)**

**{**

**imax[i]=count;**

**}**

**}**

**}**

**}**

**}**

**printf(" ");**

**for(j=0;j<terl;j++)**

**{**

**printf(" | ");**

**printf("%c",term[j]);**

**for(p=0;p<max[j]-1;p++)**

**{**

**printf(" ");**

**}**

**}**

**printf(" | ");**

**printf("%c",36);**

**printf("\n\n");**

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

**{**

**if(papa[i]!=NULL)**

**{**

**flag=0;**

**if(contains\_epsilon(first[i]))**

**{**

**flag=1;**

**}**

**count=imax[i];**

**while(count!=0)**

**{**

**if(count==imax[i])**

**{**

**printf("%c",i+65);**

**}**

**else**

**{**

**printf(" ",i+65);**

**}**

**for(j=0;j<terl;j++)**

**{**

**if(count>imax[i]-hope[i][j])**

**{**

**if(papa[i][j]==NULL)**

**{**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**printf(" | ");**

**printf("%c->%c",i+65,157);**

**for(p=0;p<max[j]-4;p++)**

**{**

**printf(" ");**

**}**

**}**

**else**

**{**

**printf(" | ");**

**printf("\_\_\_");**

**for(p=0;p<max[j]-3;p++)**

**{**

**printf(" ");**

**}**

**}**

**}**

**else**

**{**

**fookat=papa[i][j];**

**for(lu=1;lu<=imax[i]-count;lu++)**

**{**

**fookat=fookat->next;**

**}**

**if(fookat!=NULL)**

**{**

**printf(" | ");**

**printf("%s",input[fookat->pdno]);**

**for(p=0;p<max[j]-strlen(input[fookat->pdno]);p++)**

**{**

**printf(" ");**

**}**

**}**

**else**

**{**

**if(flag==1 && contains\_char(fol[i],term[j]))**

**{**

**printf(" | ");**

**printf("%c->%c",i+65,157);**

**for(p=0;p<max[j]-4;p++)**

**{**

**printf(" ");**

**}**

**}**

**}**

**}**

**}**

**else**

**{**

**printf(" | ");**

**for(p=0;p<max[j];p++)**

**{**

**printf(" ");**

**}**

**}**

**}**

**sub=36;**

**if(count==imax[i])**

**{**

**if(flag==1 && contains\_char(fol[i],sub))**

**{**

**printf(" | ");**

**printf("%c->%c",i+65,157);**

**}**

**else**

**{**

**printf(" | ");**

**printf("\_\_\_");**

**}**

**}**

**else**

**{**

**printf(" | ");**

**}**

**printf("\n");**

**count--;**

**}**

**printf(" ");**

**for(j=0;j<terl;j++)**

**{**

**printf(" | ");**

**for(p=0;p<max[j];p++)**

**{**

**printf(" ");**

**}**

**}**

**printf(" | ");**

**printf("\n");**

**}**

**}**

**}**

**printf("\n\nVrei să faci asta din nou cu o nouă gramatică??\n");**

**printf("Dacă da, tastați y. Orice altceva va fi considerat un nu.\n");**

**scanf("%c",&k);**

**scanf("%c",&k);**

**printf("\n\n");**

**if(k=='y')**

**continue;**

**else**

**break;**

**}**

**getchar();**

**}**