Exercise – 2

Simulate FIRST and FOLLOW of a grammar

FOLLOW

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int n,m=0,p,i=0,j=0;

char a[10][10],followResult[10];

void follow(char c);

void first(char c);

void addToResult(char);

int main()

{

int i;

int choice;

char c,ch;

printf("Enter the no.of productions: ");

scanf("%d", &n);

printf(" Enter %d productions\nProduction with multiple terms should be give as separate productions \n", n);

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

scanf("%s%c",a[i],&ch);

do

{

printf("Find FOLLOW of -->");

scanf(" %c",&c);

follow(c);

printf("FOLLOW(%c) = { ",c);

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

printf("%c ",followResult[i]);

printf(" }\n");

printf("Do you want to continue(Press 1 to continue....)?");

scanf("%d%c",&choice,&ch);

} while(choice==1);

}

void follow(char c)

{

if(a[0][0]==c)

addToResult('$');

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

{

for(j=2;j<strlen(a[i]);j++)

{

if(a[i][j]==c)

{

if(a[i][j+1]!='\0')first(a[i][j+1]);

if(a[i][j+1]=='\0'&&c!=a[i][0])

follow(a[i][0]);

}

}

}

}

void first(char c)

{

int k;

if(!(isupper(c)))

addToResult(c);

for(k=0;k<n;k++)

{

if(a[k][0]==c)

{

if(a[k][2]=='$')

follow(a[i][0]);

else if(islower(a[k][2]))

addToResult(a[k][2]);

else

first(a[k][2]);

}

}

}

void addToResult(char c)

{

int i;

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

if(followResult[i]==c)

return;

followResult[m++]=c;

}

Output:

Enter the no.of productions: 8

Enter 8 productions

Production with multiple terms should be give as separate productions

E=TA

A=+TA

A=$

T=FB

B=\*FB

B=$

F=(E)

F=i

Find FOLLOW of -->E

FOLLOW(E) = { $ ) }

Do you want to continue(Press 1 to continue....)?1

Find FOLLOW of -->A

FOLLOW(A) = { $ ) }

Do you want to continue(Press 1 to continue....)?1

Find FOLLOW of -->T

FOLLOW(T) = { $ ) + }

Do you want to continue(Press 1 to continue....)?1

Find FOLLOW of -->B

FOLLOW(B) = { $ ) + }

Do you want to continue(Press 1 to continue....)?1

Find FOLLOW of -->F

FOLLOW(F) = { $ ) + \* }

Do you want to continue(Press 1 to continue....)?0

**FIRST**

#include<stdio.h>

#include<ctype.h>

void FIRST(char[],char );

void addToResultSet(char[],char);

int numOfProductions;

char productionSet[10][10];

main()

{

int i;

char choice;

char c;

char result[20];

printf("How many number of productions ? :");

scanf(" %d",&numOfProductions);

for(i=0;i<numOfProductions;i++)//read production string eg: E=E+T

{

printf("Enter productions Number %d : ",i+1);

scanf(" %s",productionSet[i]);

}

do

{

printf("\n Find the FIRST of :");

scanf(" %c",&c);

FIRST(result,c); //Compute FIRST; Get Answer in 'result' array

printf("\n FIRST(%c)= { ",c);

for(i=0;result[i]!='\0';i++)

printf(" %c ",result[i]); //Display result

printf("}\n");

printf("press 'y' to continue : ");

scanf(" %c",&choice);

}

while(choice=='y'||choice =='Y');

}

/\*

\*Function FIRST:

\*Compute the elements in FIRST(c) and write them

\*in Result Array.

\*/

void FIRST(char\* Result,char c)

{

int i,j,k;

char subResult[20];

int foundEpsilon;

subResult[0]='\0';

Result[0]='\0';

//If X is terminal, FIRST(X) = {X}.

if(!(isupper(c)))

{

addToResultSet(Result,c);

return ;

}

//If X is non terminal

//Read each production

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

{

//Find production with X as LHS

if(productionSet[i][0]==c)

{

//If X → ε is a production, then add ε to FIRST(X).

if(productionSet[i][2]=='$')

addToResultSet(Result,'$');

//If X is a non-terminal, and X → Y1 Y2 … Yk

//is a production, then add a to FIRST(X)

//if for some i, a is in FIRST(Yi),

//and ε is in all of FIRST(Y1), …, FIRST(Yi-1).

else

{

j=2;

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

{

foundEpsilon=0;

FIRST(subResult,productionSet[i][j]);

for(k=0;subResult[k]!='\0';k++)

addToResultSet(Result,subResult[k]);

for(k=0;subResult[k]!='\0';k++)

if(subResult[k]=='$')

{

foundEpsilon=1;

break;

}

//No ε found, no need to check next element

if(!foundEpsilon)

break;

j++;

}

}

}

}

return ;

}

/\* addToResultSet adds the computed

\*element to result set.

\*This code avoids multiple inclusion of elements

\*/

void addToResultSet(char Result[],char val)

{

int k;

for(k=0 ;Result[k]!='\0';k++)

if(Result[k]==val)

return;

Result[k]=val;

Result[k+1]='\0';

}

Output:

How many number of productions ? :8

Enter productions Number 1 : E=TA

Enter productions Number 2 : A=+TA

Enter productions Number 3 : A=$

Enter productions Number 4 : T=FB

Enter productions Number 5 : B=\*FB

Enter productions Number 6 : B=$

Enter productions Number 7 : F=(E)

Enter productions Number 8 : F=i

Find the FIRST of :E

FIRST(E)= { ( i }

press 'y' to continue : Y

Find the FIRST of :T

FIRST(T)= { ( i }

press 'y' to continue : Y

Find the FIRST of :F

FIRST(F)= { ( i }

press 'y' to continue : Y

Find the FIRST of :A

FIRST(A)= { + $ }

press 'y' to continue : Y

Find the FIRST of :B

FIRST(B)= { \* $ }

press 'y' to continue : N

Process returned 0 (0x0) execution time : 95.919 s

Press any key to continue.

Exercise – 3

**Develop an Operator precedence parser for a given grammar.**

/\*OPERATOR PRECEDENCE PARSER\*/

#include<stdio.h>

void main()

{

char stack[20],ip[20],opt[10][10][1],ter[10];

int i,j,k,n,top=0,col,row;

for(i=0;i<10;i++)

{ stack[i]=NULL;

ip[i]=NULL;

for(j=0;j<10;j++)

{ opt[i][j][1]=NULL;

}

}

printf("Enter the no.of terminals:");

scanf("%d",&n);

printf("\nEnter the terminals:");

scanf("%s",ter);

printf("\nEnter the table values:\n");

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

{

for(j=0;j<n;j++)

{

printf("Enter the value for %c %c:",ter[i],ter[j]);

scanf("%s",opt[i][j]);

}

}

printf("\nOPERATOR PRECEDENCE TABLE:\n");

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

{ printf("\t%c",ter[i]);

}

printf("\n");

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

{

printf("\n%c",ter[i]);

for(j=0;j<n;j++)

{ printf("\t%c",opt[i][j][0]);

}

}

stack[top]='$';

printf("\nEnter the input string:");

scanf("%s",ip);

i=0;

printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");

printf("\n%s\t\t\t%s\t\t\t",stack,ip);

while(i<=strlen(ip))

{

for(k=0;k<n;k++)

{

if(stack[top]==ter[k])

col=k;

if(ip[i]==ter[k])

row=k;

}

if((stack[top]=='$')&&(ip[i]=='$'))

{

printf("String is accepted");

break;

}

else if((opt[col][row][0]=='<') ||(opt[col][row][0]=='='))

{ stack[++top]=opt[col][row][0];

stack[++top]=ip[i];

printf("Shift %c",ip[i]);

i++;

}

else

{

if(opt[col][row][0]=='>')

{

while(stack[top]!='<'){--top;}

top=top-1;

printf("Reduce");

}

else

{

printf("\nString is not accepted");

break;

}

}

printf("\n");

for(k=0;k<=top;k++)

{

printf("%c",stack[k]);

}

printf("\t\t\t");

for(k=i;k<strlen(ip);k++)

{

printf("%c",ip[k]);

}

printf("\t\t\t");

}

}

Output

Enter the no.of terminals:4

Enter the terminals:+\*i$

Enter the table values:

Enter the value for + +:>

Enter the value for + \*:<

Enter the value for + i:<

Enter the value for + $:>

Enter the value for \* +:>

Enter the value for \* \*:>

Enter the value for \* i:<

Enter the value for \* $:>

Enter the value for i +:>

Enter the value for i \*:>

Enter the value for i i:e

Enter the value for i $:>

Enter the value for $ +:<

Enter the value for $ \*:<

Enter the value for $ i:<

Enter the value for $ $:e

OPERATOR PRECEDENCE TABLE:

+ \* i $

+ > < < >

\* > > < >

i > > e >

$ < < < e

Enter the input string:i+i\*i$

STACK INPUT STRING ACTION

$ i+i\*i$ Shift i

$<i +i\*i$ Reduce

$ +i\*i$ Shift +

$<+ i\*i$ Shift i

$<+<i \*i$ Reduce

$<+ \*i$ Shift \*

$<+<\* i$ Shift i

$<+<\*<i $ Reduce

$<+<\* $ Reduce

$<+ $ Reduce

$ $ String is accepted

Process returned 18 (0x12) execution time : 141.229 s

Press any key to continue.