**2CS701 Compiler Construction**

|  |  |
| --- | --- |
| **Practical 3** | |
| **Rollno:** 19BCE236 | **Name:** Samariya Darsh |
| **Date:** 12-10-2022 | **Batch:** D1 |

**Aim :-** Write a program to find first( ), and follow() set for each non-terminal of given grammar.

**Code :-**

// C program to calculate the First and

// Follow sets of a given grammar

#include<stdio.h>

#include<ctype.h>

#include<string.h>

void followfirst(char, int, int);

void follow(char c);

void findfirst(char, int, int);

int count, n = 0;

//store final first-follow sets

char calc\_first[10][100];

char calc\_follow[10][100];

int m = 0;

// Stores the production rules

char production[10][10];

char f[10], first[10];

int k;

char ck;

int e;

int main(int argc, char \*\*argv)

{

    int jm = 0;

    int km = 0;

    int i, choice;

    char c, ch;

    count = 8;

    // The Input grammar

    strcpy(production[0], "E=TR");

    strcpy(production[1], "R=+TR");

    strcpy(production[2], "R=#");

    strcpy(production[3], "T=FY");

    strcpy(production[4], "Y=\*FY");

    strcpy(production[5], "Y=#");

    strcpy(production[6], "F=(E)");

    strcpy(production[7], "F=i");

    printf("CFG\n");

    printf("E-->TR");

    printf("\nR-->+TR");

    printf("\nR-->#");

    printf("\nT-->FY");

    printf("\nY-->\*FY");

    printf("\nY-->#");

    printf("\nF-->(E)");

    printf("\nF-->i\n");

    printf("-----------------------------------------------\n\n");

    int kay;

    char done[count];

    int ptr = -1;

    for(k = 0; k < count; k++) {

        for(kay = 0; kay < 100; kay++) {

            calc\_first[k][kay] = '!';

        }

    }

    int point1 = 0, point2, xxx;

    for(k = 0; k < count; k++)

    {

        c = production[k][0];

        point2 = 0;

        xxx = 0;

        for(kay = 0; kay <= ptr; kay++)

            if(c == done[kay])

                xxx = 1;

        if (xxx == 1)

            continue;

        findfirst(c, 0, 0);

        ptr += 1;

        done[ptr] = c;

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

        calc\_first[point1][point2++] = c;

        for(i = 0 + jm; i < n; i++) {

            int lark = 0, chk = 0;

            for(lark = 0; lark < point2; lark++) {

                if (first[i] == calc\_first[point1][lark])

                {

                    chk = 1;

                    break;

                }

            }

            if(chk == 0)

            {

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

                calc\_first[point1][point2++] = first[i];

            }

        }

        printf("}\n");

        jm = n;

        point1++;

    }

    printf("\n");

    printf("-----------------------------------------------\n\n");

    char donee[count];

    ptr = -1;

    for(k = 0; k < count; k++) {

        for(kay = 0; kay < 100; kay++) {

            calc\_follow[k][kay] = '!';

        }

    }

    point1 = 0;

    int land = 0;

    for(e = 0; e < count; e++)

    {

        ck = production[e][0];

        point2 = 0;

        xxx = 0;

        for(kay = 0; kay <= ptr; kay++)

            if(ck == donee[kay])

                xxx = 1;

        if (xxx == 1)

            continue;

        land += 1;

        follow(ck);

        ptr += 1;

        donee[ptr] = ck;

        printf(" Follow(%c) = { ", ck);

        calc\_follow[point1][point2++] = ck;

        for(i = 0 + km; i < m; i++) {

            int lark = 0, chk = 0;

            for(lark = 0; lark < point2; lark++)

            {

                if (f[i] == calc\_follow[point1][lark])

                {

                    chk = 1;

                    break;

                }

            }

            if(chk == 0)

            {

                printf("%c, ", f[i]);

                calc\_follow[point1][point2++] = f[i];

            }

        }

        printf(" }\n\n");

        km = m;

        point1++;

    }

}

void follow(char c)

{

    int i, j;

    if(production[0][0] == c) {

        f[m++] = '$';

    }

    for(i = 0; i < 10; i++)

    {

        for(j = 2;j < 10; j++)

        {

            if(production[i][j] == c)

            {

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

                {

                    followfirst(production[i][j+1], i, (j+2));

                }

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

                {

                    follow(production[i][0]);

                }

            }

        }

    }

}

void findfirst(char c, int q1, int q2)

{

    int j;

    if(!(isupper(c))) {

        first[n++] = c;

    }

    for(j = 0; j < count; j++)

    {

        if(production[j][0] == c)

        {

            if(production[j][2] == '#')

            {

                if(production[q1][q2] == '\0')

                    first[n++] = '#';

                else if(production[q1][q2] != '\0'

                        && (q1 != 0 || q2 != 0))

                {

                    findfirst(production[q1][q2], q1, (q2+1));

                }

                else

                    first[n++] = '#';

            }

            else if(!isupper(production[j][2]))

            {

                first[n++] = production[j][2];

            }

            else

            {

                findfirst(production[j][2], j, 3);

            }

        }

    }

}

void followfirst(char c, int c1, int c2)

{

    int k;

    if(!(isupper(c)))

        f[m++] = c;

    else

    {

        int i = 0, j = 1;

        for(i = 0; i < count; i++)

        {

            if(calc\_first[i][0] == c)

                break;

        }

        while(calc\_first[i][j] != '!')

        {

            if(calc\_first[i][j] != '#')

            {

                f[m++] = calc\_first[i][j];

            }

            else

            {

                if(production[c1][c2] == '\0')

                {

                    follow(production[c1][0]);

                }

                else

                {

                    followfirst(production[c1][c2], c1, c2+1);

                }

            }

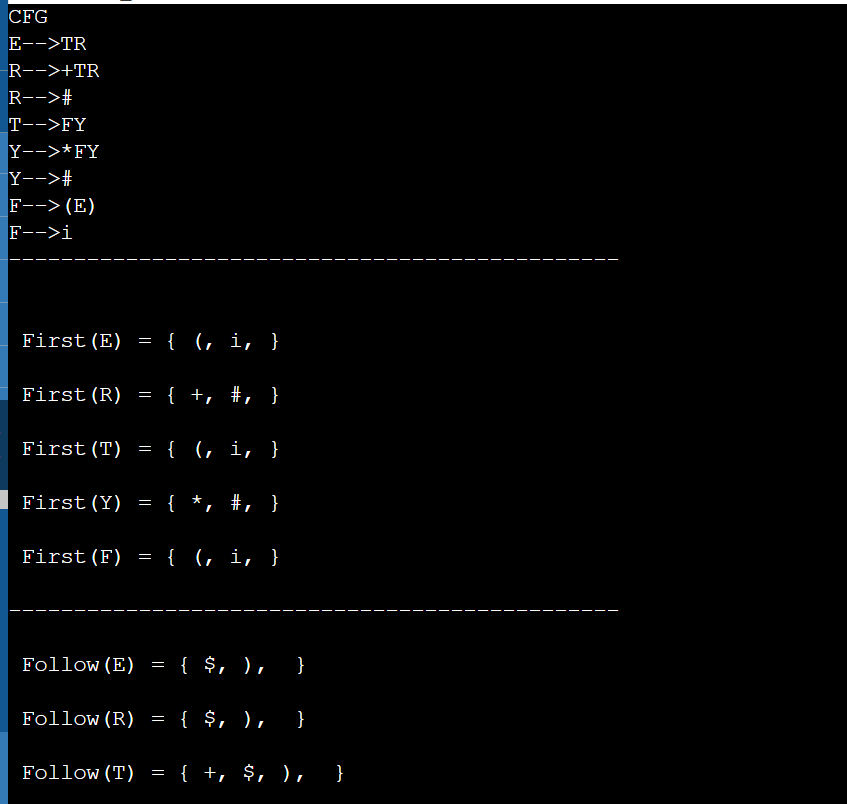
            j++;

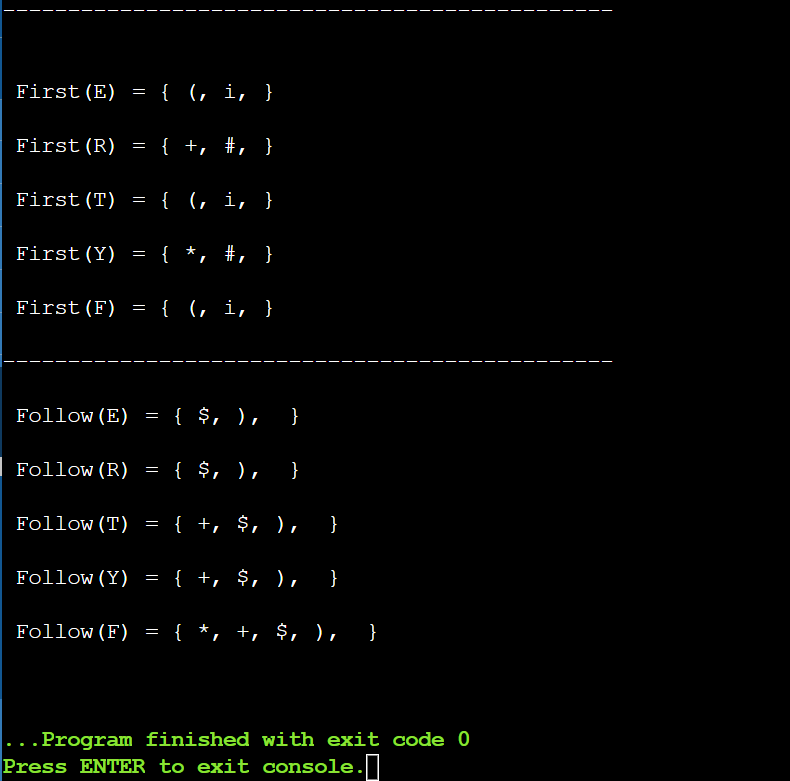
        }

    }

}

**Output :-**

****

****