**CS6612 – Compiler Lab**

**Ex no : 4 Name : Sreedhar V**

**Date : 28.02.2021 Reg no: 185001161**

**Specification**

Write a program in C to construct Recursive Descent Parser for the following grammar which is for arithmetic expression involving + and \*. Check the Grammar for left recursion and convert into suitable for this parser. Write recursive functions for every non-terminal. Call the function for start symbol of the Grammar in main().

**G: E🡪E+T|E-T|T**

**T🡪T\*F | T/F|F**

**F🡪(E)|i**

**Code**

/\*E->TE'

E'-> +TE' | -TE' | epsilon

T->FT'

T'->\*FT'| /FT' | epsilon

F->(E) | id\*/

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

char str[100];

int i=0;

int E();

int E1();

int T();

int T1();

int F();

void print\_tab(int depth);

void main()

{

    strcpy(str,"i+i");

    printf("\nThe Given String is %s\n",str);

    if(E(0))

        printf("\nThe String is Accepted!\n");

    else

        printf("\nThe String is not accepted!\n");

}

int E()

{

    printf("\nE() is called\n");

    if(T())

    {

        if(E1())

            return 1;

        else

            return 0;

    }

    else

        return 0;

}

int E1()

{

    printf("\nE1() is called\n");

    if(str[i]=='+')

    {

        i++;

        if(T())

        {

            if(E1())

                return 1;

            else

                return 0;

        }

        else

            return 0;

    }

    else if(str[i]=='-')

    {

        i++;

        if(T())

        {

            if(E1())

                return 1;

            else

                return 0;

        }

        else

            return 0;

    }

    else if(str[i]=='\0')

    {

        //i++;

        return 1;

    }

}

int T()

{

    printf("\nT() is called\n");

     if(F())

    {

        if(T1())

            return 1;

        else

            return 0;

    }

    else

        return 0;

}

int T1()

{

    printf("\nT1() is called\n");

    if(str[i]=='\*')

    {

        i++;

        if(F())

        {

            if(T1())

                return 1;

            else

                return 0;

        }

        else

            return 0;

    }

    else if(str[i]=='/')

    {

        i++;

        if(F())

        {

            if(T1())

                return 1;

            else

                return 0;

        }

        else

            return 0;

    }

    else if(str[i]=='\0')

    {

        //i++;

        return 1;

    }

}

int F()

{

    printf("\nF() is called\n");

    if(str[i]=='(')

    {

        i++;

        if(E())

        {

            if(str[i]==')')

            {

                i++;

                return 1;

            }

            else

                return 0;

        }

        else

            return 0;

    }

    else if(str[i]=='i')

    {

        i++;

        return 1;

    }

}

void print\_tabs(int depth)

{

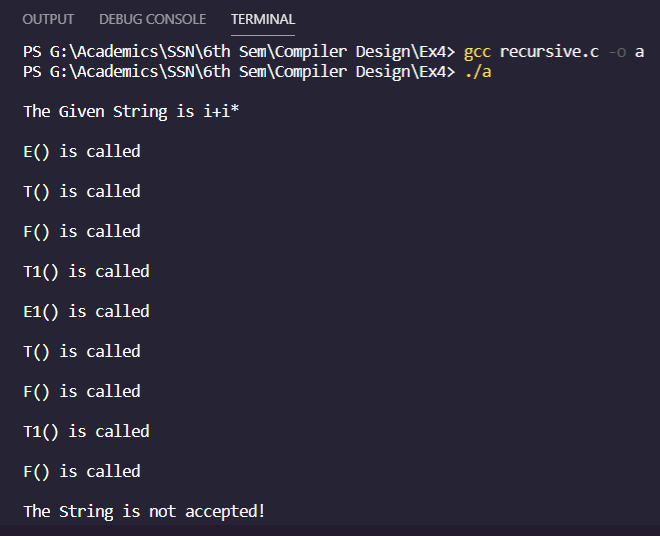
    int i=0;

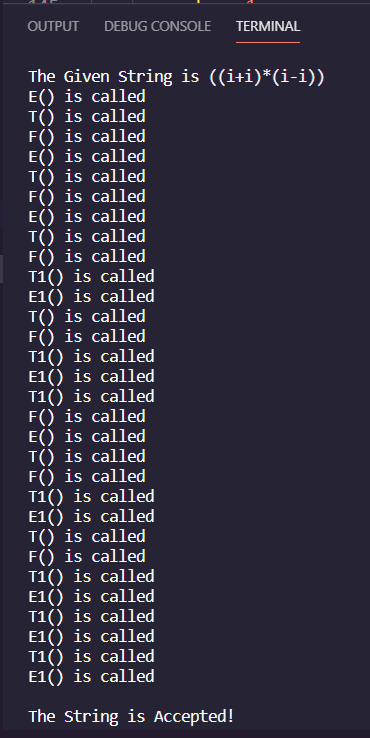
    for(i=0;i<depth;i++)

        printf("\t");

}

*(Output)*





**Learning Outcome:**

* I’ve learnt how to construct the recursive decent parser for the given input grammar
* I’ve learnt the internal working of the recursive parser and able to trace the input string whether it’s accepted or not.