Practical - 2

2CS701 – Compiler Construction

Harshit Gajipara

19bce059

**Aim:**

To Implement a Recursive Descent Parser Algorithm for the Grammar.

**Code:**

/\*

To Implement a Recursive Descent Parser Algorithm for the Grammer.

E --> T + E | T - E | T

T --> F \* T | F / T | F

F --> ID | NUM | (E)

\*/

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char input[20];

int i, error;

void E();

void T();

void F();

void main()

{

    i = 0;

    error = 0;

    printf("\nEnter an arithmetic expression : ");

    gets(input);

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

    E();

    if (strlen(input) == i && error == 0)

    {

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

        printf("\nString is successfully parsed!\n\n");

    }

    else

    {

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

        printf("Error in parsing String\n\n");

    }

}

void E()

{

    T();

    if (input[i] == '+' || input[i] == '-')

    {

        printf("%c\tE->T%cE \n", input[i], input[i]);

        i++;

        E();

    }

}

void T()

{

    F();

    if (input[i] == '\*' || input[i] == '/')

    {

        printf("%c\tT->F%cT \n", input[i], input[i]);

        i++;

        T();

    }

}

void F()

{

    if (isalpha(input[i]))

    {

        printf("%c\tF->ID \n", input[i]);

        i++;

    }

    else if (isalnum(input[i]))

    {

        printf("%c\tF->NUMBER \n", input[i]);

        i++;

    }

    else if (input[i] == '(')

    {

        printf("%c\tF->(E) \n", input[i]);

        i++;

        E();

        if (input[i] == ')')

        {

            printf("%c\tF->(E) \n", input[i]);

            i++;

        }

        else

            error = 1;

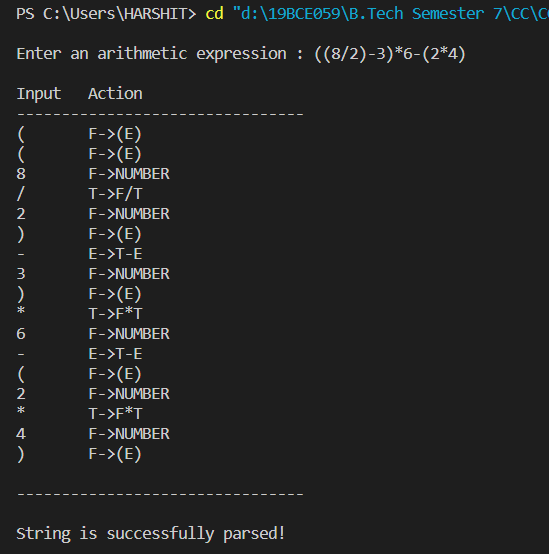
    }

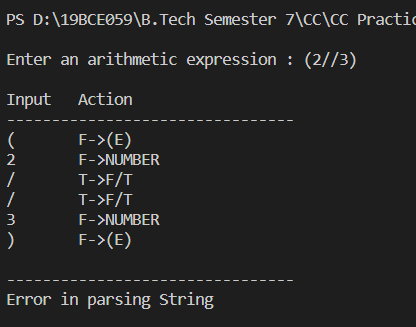
    else

        error = 1;

}

**Output:**





**Conclusion:**

In this practical, we learnt that using RDP we can parse any input for given grammar and check if the input is accepted by grammar or not.