Sharvan Ram Kumaran

Reg No:185001143

CSE C

**Exercise 4**

Objective:

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().

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int space = 0;

void indent(){

for(int i = 0;i<space;i++)printf(" ");

}

void F(char\* inp, int\* cur);

void read(char\* inp) {

printf("Enter code(Expression):");

scanf(" %[^\n]", inp);

int n = strlen(inp);

inp[n] = '$';

inp[n + 1] = '\0';

}

void advance(int\* cur){

(\*cur) = (\*cur) + 1;

}

void err(char \*inp, int \*cur){

printf("\tError:Excpected CONST token, found: %c, at column %d\n", inp[\*cur], \*cur);

exit(0);

}

void Tprime(char\* inp, int\* cur){

space++;

indent();

printf("Entered T'()\n");

if(inp[\*cur] == '\*' || inp[\*cur] == '/') {

advance(cur);

F(inp, cur);

Tprime(inp, cur);

}else{

indent();

space--;

printf("Exiting T'()\n");

return;

}

}

void T(char\* inp, int\* cur){

space++;

indent();

printf("Entered T()\n");

F(inp, cur);

Tprime(inp, cur);

indent();

space--;

printf("Exiting T()\n");

}

void Eprime(char\* inp, int\* cur){

space++;

indent();

printf("Entered E'()\n");

if(inp[\*cur] == '+' || inp[\*cur] == '-') {

advance(cur);

T(inp, cur);

Eprime(inp, cur);

}else{

indent();

space--;

printf("Exiting E'()\n");

return;

}

}

void E(char\* inp, int\* cur){

space++;

indent();

printf("Entered E()\n");

T(inp, cur);

Eprime(inp, cur);

indent();

space--;

printf("Exiting E()\n");

}

void F(char\* inp, int\* cur){

space++;

indent();

printf("Entered F()\n");

if(inp[\*cur] == '('){

advance(cur);

E(inp, cur);

if(inp[\*cur] == ')'){

advance(cur);

}else{

err(inp, cur);

}

}

else if(inp[\*cur] == 'i'){

advance(cur);

}else{

err(inp, cur);

}

indent();

space--;

printf("Exiting F()\n");

}

void solve(char\* inp, int\* cur) {

printf("\n\tResuilt:\t");

E(inp, cur);

if (inp[\*cur] == '$') {

printf("\tSuccess\n");

}else{

printf("\tFailure\n");

}

}

void main() {

char\* inp = (char \*)malloc(sizeof(char) \* 108);

int v= 0;

int\* cur = &v;

read(inp);

solve(inp, cur);

}

**Output:**

Text

Description automatically generated

Text

Description automatically generated

**Learning Outcome:**

A recursive descent parser was built using appropriate functions.