

Compiler Design Lab (CS 306)

Week 6: Implementation of Recursive Descent Parser

Week 6 Programs

1. Implement Recursive Descent Parser for the Expression Grammar given below.

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \epsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \epsilon \\ F &\rightarrow (E) \mid i \end{aligned}$$

Program:

C implementation of Recursive Descent Parser for the Expression Grammar is given below.

```
#include<stdio.h>
#include<string.h>
int E(),Edash(),T(),Tdash(),F();
char *ip;
char string[50];
int main()
{
    printf("Enter the string\n");
    scanf("%s",string);
    ip=string;
    printf("\n\nInput\tAction\n-----\n");
    if(E() && ip=='\0'){
        printf("\n-----\n");
        printf("\n String is successfully parsed\n"); }
    else{
        printf("\n-----\n");
        printf("Error in parsing String\n"); }
    }
int E()
{
    printf("%s\tE->TE' \n",ip);
    if(T())
    {
        if(Edash())
        {
            return 1;
        }
        else
        return 0;
    }
}
```

```

else
return 0;
}
int Edash()
{
if(*ip=='+')
{
printf("%s\tE'->+TE' \n",ip);
ip++;
if(T())
{
if(Edash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tE'->^ \n",ip);
return 1;
}
}
int T()
{
printf("%s\tT->FT' \n",ip);
if(F())
{
if(Tdash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
int Tdash()
{
if(*ip=='*')
{
printf("%s\tT'->*FT' \n",ip);
ip++;
if(F())
{
if(Tdash())

```

```

{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tT'->^ \n",ip);
return 1;
}
}
int F()
{
if(*ip=='(')
{
printf("%s\tF->(E) \n",ip);
ip++;
if(E())
{
if(*ip==')')
{
ip++;
return 0;
}
else
return 0;
}
else if(*ip=='i')
{
ip++;
printf("%s\tF->id \n",ip);
return 1;
}
else
return 0;
}
}

```

Test cases:

i+i*i	String is successfully parsed
i+i	String is successfully parsed
i*i	String is successfully parsed
i*i+i*i+i	String is successfully parsed
i++*+i	Error in parsing String
i+i*	Error in parsing String

2. Construct Recursive Descent Parser for the grammar

$G = (\{S, L\}, \{(,), a, ,\}, \{S \rightarrow (L) \mid a ; L \rightarrow L, S \mid S\}, S)$ and verify the acceptability of the following strings:

- i. (a,(a,a))
- ii. (a,((a,a),(a,a)))

You can manually eliminate Left Recursion if any in the grammar.

ans) After Eliminating the left recursion :

- $S \rightarrow (L) \mid a$
- $L \rightarrow SL'$
- $L' \rightarrow ,SL' \mid \epsilon$

Code :

```
#include<stdio.h>
#include<string.h>
int S(),Ldash(),L();
char *ip;
char string[50];
int main()
{
    printf("Enter the string\n");
    scanf("%s",string);
    ip=string;
    printf("\n\nInput\tAction\n-----\n");
    if(S()){
        printf("\n-----\n");
        printf("\n String is successfully parsed\n");
    }
    else{
        printf("\n-----\n");
        printf("Error in parsing String\n");
    }
}
int L()
{

```

```

printf("%s\tL->SL' \n",ip);
if(S())
{
if(Ldash())
{
return 1;
}
else
return 0;
}
else
return 0;
}

```

```

int Ldash()
{
if(*ip=='')
{
printf("%s\tL'->,SL' \n",ip);
ip++;
if(S())
{
if(Ldash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tL'->E\n",ip);
return 1;
}
}
}

```

```

int S()
{
if(*ip == '('){
printf("%s\tS->(L) \n",ip);
ip++;
if(L())
{
if(*ip == ')')
{
ip ++;
return 1;
}
}
}
}

```

```

else
return 0;
}
else
return 0;
}
else if(*ip == 'a'){
    ip++;
    printf("%s\t\tS->a \n",ip);
    return 1;
}
else
return 0;
}

```

```

Enter the string
(a, (a, a))

```

```

Input    Action
-----

```

```

(a, (a, a))      S->(L)
a, (a, a))      L->SL'
, (a, a))        S->a
, (a, a)) L' ->, SL'
(a, a)) S->(L)
a, a)) L->SL'
, a)) S->a
, a)) L' ->, SL'
)) S->a
)) L' ->E
) L' ->E

```

```

-----

String is successfully parsed

```

```

...Program finished with exit code 0
Press ENTER to exit console.

```