C PROGRAM FOR RECURSIVE DESCENT PARSER

March 3, 2013 C, Compiler Design 6,458 Views

C Program for implementation of Recursive Descent Parser for language given below

E-> TE'

E'-> +TE' | -TE' | null

T-> FT'

T'-> *FT'| /FT' | null

F-> id/ (E)/ num

Here 'id' can be a single letter. And 'num' can be a single digit number.

```
#include "stdio.h"
#include "conio.h"
char input[100];
char prod[100][100];
int pos=-1,l,st=-1;
char id,num;
void E();
void T();
void F();
void advance();
void Td();
void Ed();
void advance()
{
pos++;
if(pos&lt;l)
{
if(input[pos]&gt;='0'&amp;&amp; input[pos]&lt;='9')
{
num=input[pos];
id='\0';
}
if((input[pos]&gt;='a' || input[pos]&gt;='A')&amp;&amp;(input[pos]&lt;='z' || input[pos]&lt;='Z'))
{id=input[pos];
num='\0';
}
}
}
void E()
{
strcpy(prod[++st],"E-&gt;TE'");
T();
Ed();
}
void Ed()
{
int p=1;
if(input[pos]=='+')
{
p=0;
strcpy(prod[++st],"E'-&gt;+TE'");
advance();
T();
Ed();
}
if(input[pos]=='-')
{   p=0;
strcpy(prod[++st],"E'-&gt;-TE'");
advance();
T();
Ed();
}

                                    // Recursive Descent Parser
if(p==1)
{
strcpy(prod[++st],"E'-&gt;null");
}
}

void T()
{
strcpy(prod[++st],"T-&gt;FT'");
F();
Td();
}
void Td()
{
```

```
int p=1;
if(input[pos]=='*')
{
p=0;
strcpy(prod[++st],"T'-&gt;*FT'");
advance();
F();
Td();
}
if(input[pos]=='/')
{   p=0;
strcpy(prod[++st],"T'-&gt;/FT'");
advance();
F();
Td();
}
if(p==1)
strcpy(prod[++st],"T'-&gt;null");
}
void F()
{
if(input[pos]==id) {
strcpy(prod[++st],"F-&gt;id");
advance();          }
if(input[pos]=='(')
{
strcpy(prod[++st],"F-&gt;(E)");
advance();
E();
if(input[pos]==')')   {
//strcpy(prod[++st],"F-&gt;(E)");
advance();             }
}
if(input[pos]==num)
{
strcpy(prod[++st],"F-&gt;num");
advance();
}
}
int main()
```

Stay updated to Programming and latest Events! Your email Subscribe

```
scanf("%s",input);
l=strlen(input);
input[l]='$';
advance();
E();
if(pos==l)
{
printf("String Accepted\n");
for(i=0;i&lt;=st;i++)
{
printf("%s\n",prod[i]);
}
}
else
{
printf("String rejected\n");
}
getch();
return 0;
}
```

OUTPUT:

```
Enter Input String (a+b)
String Accepted
E->TE'
T->FT'
F->(E)
E->TE'
T->FT'
F->id
T'->null
E'->+TE'
T->FT'
F->id
T'->null
E'->null
T'->null
E'->null
```

If you like this then please share this page. You can comment you queries here. Or you can also ask for a program. We are happy to help you.

tags: C program for Recursive Decsent parser, compiler design programs, parsing techniques