CodingAlpha
Programming Codes and Tutorials

Implement Recursive Descent Parsing C Program

By Tushar Soni | October 18, 2016 10 Comments

Let us learn how to develop a recursive descent parsing program in C programming language using functions.

The header file **ctype.h** is used because the definition for **isalnum()** function is defined in it and similarly the definitions of **strlen()** method is in **string.h** header file.

What is a Recursive Descent Parser?

A recursive descent parser is a top-down parser. This is one of the most simple forms of parsing. It is used to build a parse tree from top to bottom and reads the input from **left to right**.

A form of recursive descent parsing that does not require backtracking algorithm is known as a **predictive parser**. The parsers that use backtracking may require exponential time. This parser is normally used for compiler designing purpose.

The parser gets an input and reads it from left to right and checks it. If the source code fails to parse properly, then the parser exits by giving an error (flag) message. If it parses the source code properly then it exits without giving an error message.

Must Read: C Program To Find First and Follow of a Grammar

Recursive Descent Parser Algorithm

```
for a = 1 to limit
      for b = 1 to Array - 1 do begin
            Increase Production: Array[a] ->Array[b]
      end for
      Remove Immediate Left recursion from A[a]
End for
```


Note: This recursive sescent parsing C program is compiled with GNU GCC compiler and written in gEdit Editor in Linux Ubuntu operating system.

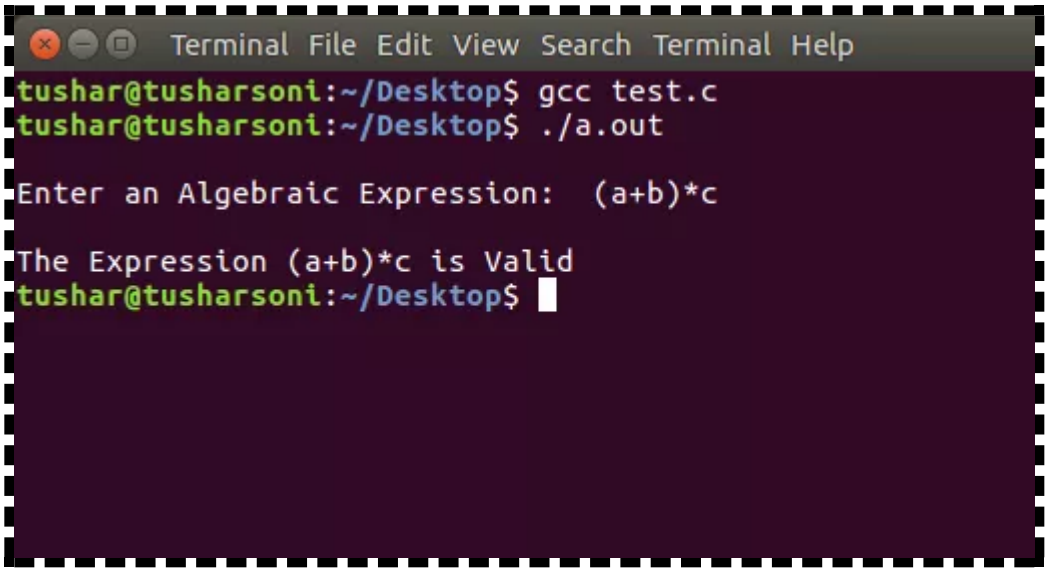
C Program To Develop A Recursive Descent Parser

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>

void Tprime();
void Eprime();
void E();
void check();
void T();


char expression[10];
int count, flag;

int main()
{
      count = 0;
      flag = 0;
```

```
	printf("\nEnter an Algebraic Expression:\t");
	scanf("%s", expression);
	E();
	if((strlen(expression) == count) && (flag == 0))
	{
		printf("\nThe Expression %s is Valid\n", expression);
	}
	else
	{
		printf("\nThe Expression %s is Invalid\n", expression);
	}
}

void E()
{
	T();
	Eprime();
}

void T()
{
	check();
	Tprime();
}

void Tprime()
{
	if(expression[count] == '*')
	{
		count++;
		check();
		Tprime();
	}
}

void check()
{
	if(isalnum(expression[count]))
	{
		count++;
	}
	else if(expression[count] == '(')
	{
		count++;
		E();
		if(expression[count] == ')')
		{
			count++;
		}
		else
		{
			flag = 1;
		}
	}
	else
	{
		flag = 1;
	}
}

void Eprime()
{
	if(expression[count] == '+')
	{
		count++;
```

```
            T();
            Eprime();
        }
}
```

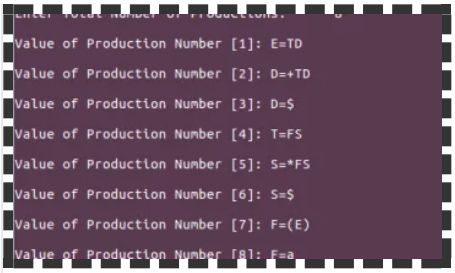
Output



If you have any compilation error or doubts in this C program for recursive descent parser code, let us know about it in the comment section below.

Share This Article!!!

Related



Find First and Follow of a Grammar C Program
October 9, 2016
In "C Programming"

Lexical Analyzer in C Programming
October 12, 2017
In "C Programming"

First Fit Algorithm C Program
November 4, 2016
In "C Programming"

Category: C Programming

10 thoughts on “Implement Recursive Descent Parsing C Program”

Akash Pillai
October 18, 2016

At last, I found a working program for this code. I am developing a basic compiler using recursive descent parsing technique. I hope it goes well.

ATHIRA P U
August 29, 2017

the prgm is really gud

Mahesh Ganguly
October 19, 2016

LL parsing technique is much more efficient than recursive descent parsing in C programming. Please try to implement LL parsing in C programming. LL parsers are always linear in time too.

Chaitrali Naik
October 19, 2016

Thanks for this parser generator c program. Recursive Descent Parsers can actually handle greater classes of grammars than LL1 Parsers. Both have their own pros and cons.

Manoj Sharma
October 20, 2016

Can you enlist other parsing algorithms as well?

Tushar Soni Post author
October 20, 2016

Apart from Recursive Descent Parser, there are so many other parsing algorithms used in compiler designing such as:

1. GLR
2. LL
3. LR
4. Simple Precedence Parser
5. Top-Down
6. Bottom-Up
7. LARL
8. Bounded Context
9. CYK
10. SLR

Akshay Bramhe
November 22, 2016

This recursive descent parser in C programming is really good. Thanks for the efforts.

Arshan Reddy
September 2, 2017

Actually it process the following strings also e=
e=e any symbols e*e/ so if any terminals
Other than (*,+,digit/num)these symbols it just
Comes out of all recursive procedures
SOL:
so in order to correct it keep the following
Additional condition to flag ==0 && (count==length)
Where length=strlen(expression)
If any symbols other than above mentioned symbols occurs it comes out so count does
Not gets equal to length it means it met with terminal that is not written in our code .

akon
September 27, 2017

hi

akon
September 27, 2017

can anybody give me production rule for above parser code