



Little C

COMPILER DESIGN GRAMMAR PRE.

What is Compiler?

A compiler translates the code written in one language to some other language without changing the meaning of the program. It is also expected that a compiler should make the target code efficient and optimized in terms of time and space.

WHAT IS COMPILER DESIGN?

Compiler design principles provide an in-depth view of translation and optimization process. Compiler design covers basic translation mechanism and error detection & recovery. It includes lexical, syntax, and semantic analysis as front end, and code generation and optimization as back-end.



OUR PROJECT

WHAT WE GONNA DO

The grammar that we have designed systematically describes the execution flow from the import statement of the library to the end of the execution of the program.

SPECIAL HIGHLIGHTS

- Import Library
- \$<Comments>\$
- Special Structural Types(Stack, Queue, and Array)
- Letter = a | ... | z | A | ... | Z
- Digit = 0 | ... | 9
- Logical Operator = and | or
- Conditional Operators < | > | <= | >= | != | ==
- Arithmetic Operators = + | - | * | / | %
- Functions Declarations

THE GRAMMAR

- Program \rightarrow Import [standard Library] Variable_declaration_list Funtion_list
- Library \rightarrow Math Library | String Library | Math | String | ϵ
- Variable_Declaration_List \rightarrow Variable ; Variable_Declaration_List | ϵ
- Funtion_List \rightarrow Function;Function_List | ϵ
- Function \rightarrow Type id (ParameterList) Block | sType id(ParameterList) Block
- sType \rightarrow Array | Stack | Queue
- Variable \rightarrow Type id | Type sType id[Num]
- Type \rightarrow Int | Float | Char | Bool | Void | Date_Time | String
- Id \rightarrow Letters Alpha
- Alpha \rightarrow Letters Alpha | Digit Alpha | _Alpha | ϵ
- Parameter_List \rightarrow Parameter Parameter_List | ϵ
- Parameter \rightarrow Type id | Type sType id | ϵ
- Block \rightarrow {(Variable_Declaration_List | Statement_List)*}
- Statement_List \rightarrow Statement Statement_List | ϵ
- Statement \rightarrow Expression | Loop_Statement | Conditional_Statement |
Function_Call_Statement | Return Expression | Return Id | break | Read(id); | Write(Text);

...CONTINUED

- Loop_Statement \rightarrow for(id=expression; id Conditional_Operator expression; id=id Arithmetic_Operator Num) Block | while(expression) Block | do Block while(expression)
- Expression \rightarrow Arithmetic_Expression | Conditional_Expression | Logical_Expression | id = Expression
- Arithmetic_Expression \rightarrow Num Arithmetic_Operator Arithmetic_Expression | Num | {Arithmetic_Expression} | [Arithmetic_Expression] | (Arithmetic_Expression)
- Conditional_Expression \rightarrow id Conditional_Operator id
- Logical_Expression \rightarrow Conditional_Expression Logical_Operator Logical_Expression | Conditional_Expression
- Conditional_Statement \rightarrow if(Expression) Block else Block | if(Expression) Block
- Function_Call_Statement \rightarrow Function_Name (List_Of_Identifiers)|id=Function_Name(List_Of_Identifiers)
- Text \rightarrow [a-zA-Z0-9\t\n ./<>?;:"'`!@#\$%^&*()\[\]\{\}_+=|\\"-]
- List_Of_Identifiers \rightarrow id | id, List_Of_Identifier | ϵ
- Conditional_Operator \rightarrow < | > | != | >= | <= | ==
- Arithmetic_Operator \rightarrow + | - | * | /
- Logical_Operator \rightarrow OR | AND
- Num \rightarrow [digit]+
- Letter \rightarrow [a-zA-Z]
- Digit \rightarrow [0-9]
- Num \rightarrow [+|-]?Digit[.]Digit

AN EXAMPLE

```
1 Import[Standard Math]
2     int a;
3     char d;
4 int main()
5 {
6     float z;
7     write ("Enter value")
8     read(number)
9     for(i=1;i<=number; i=i+2)
10    {
11        write(i)
12    }
13 }
```


REFERENCES

- Compiler principles, techniques, and tools by Ravi sat, Jeffery d.ulman
- torialspoint.com/compiler_design
- Programming in ANSI C by Balaguruswamy

OUR TEAM

- Ravi Shankar Kiradoo(201451007)
- Saurabh Nitnaware(201451017)
- Rahul Chaurasia(201451039)
- Chirag Garg(201451052)
- Mukesh Kumar Sain(201451060)