**Compiler Designing Strategies for Object Oriented Languages**

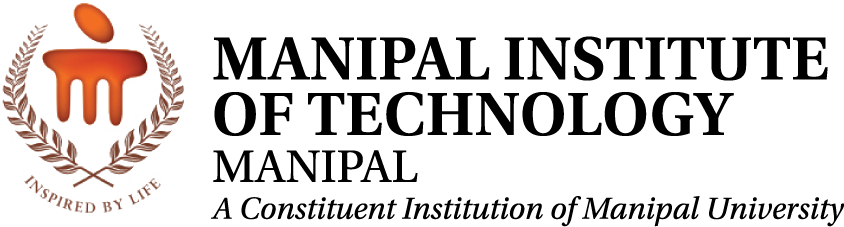
***Submitted by:***

**Krishna Birla (160905036)**

Computer Science & Engineering - D

Manipal Institute of Technology, MAHE

Manipal, Udupi - 576104

**Department of Computer Science & EngineeringCompiler Designing Strategies for Object Oriented Languages**

**Krishna Birla**

Manipal Institute of Technology, MAHE

Manipal, Udupi - 576104

8674916754 | krishnabirla16@gmail.com

*Abstract* – The paper describes the design and working of a simplified model of a compiler for object oriented languages. C# has been selected as the example language while the compiler itself has been developed on the ANSI C platform. The purpose of the paper is to introduce new methodologies in code parsing and enhance the existing algorithms to get productive results out of this redundant effort. The paper focuses on demonstrating smart use of data structures and algorithmic tools to make the compilation process simple, efficient and less costly in time. It explains in great detail, the trade-off between time and space complexities associated with the design and tries to sharpen the line that differentiates the two. While keeping the functionalities of the compiler under concern to a required minimal, the paper sheds light on all key points and service areas that an ideal compiler is expected to touch upon.

*Regular Definitions Used* –

* **IDENTIFIER** – **LETTER\_SCORE LETTER\_SCORE\_DIGIT\_EXTENSION**
* **LETTER\_SCORE** – [*a* - *z*] | [*A* - *Z*] | *\_*
* **LETTER\_SCORE\_DIGIT** - [*a* - *z*] | [*A* - *Z*] | *\_* | [*0* - *9*]
* **LETTER\_SCORE\_DIGIT\_EXTENSION** – **LETTER\_SCORE\_DIGIT** | *ϵ*
* **SIZE\_T** - [*0* - *9*] | [*0* - *9*] **SIZE\_T**
* **CONSTANT** – **NUMERIC\_CONSTANT** | **STRING\_CONSTANT**
* **DIGITS** - [*0* - *9*] | [*0* - *9*] **DIGITS**
* **SIGN** – + | - | *ϵ*
* **NUMERIC\_CONSTANT** – **SIGN** **DIGITS FRACTION EXPONENT**
* **FRACTION** – *.***DIGITS** | *ϵ*
* **EXPONENT** – *E***SIGN NUMERIC\_CONSTANT** | *e***SIGN NUMERIC\_CONSTANT** | *ϵ*
* **STRING\_CONSTANT** – *“***ASCII\_CHARS***”*
* **ASCII\_CHARS** – [**CHAR\_0** - **CHAR\_255**] **ASCII\_CHARS** | *ϵ*

*Grammar* *Used* –

* **PROGRAM** – **PREPROCESSOR APPLICATION**
* **PREPROCESSOR** –*using* **IDENTIFIER***;* **PREPROCESSOR** | *ϵ*
* **APPLICATION** – **NAMESPACE APPLICATION** | **CLASS APPLICATION** | *ϵ*
* **NAMESPACE** – *namespace* **IDENTIFIER** *{****CLASS****}* | *ϵ*
* **CLASS** – *class* **IDENTIFIER** *{***MEMBER***}* ***CLASS***| *ϵ*
* **MEMBER** – **MEMBER\_VARIABLE MEMBER** | **MEMBER\_METHOD MEMBER** | *ϵ*
* **MEMBER\_VARIABLE** – **ACCESS\_SPECIFIER DATA\_TYPE VECTOR IDENTIFIER VAR\_TERM**
* **VAR\_TERM** –*;* | *= new* **DATA\_TYPE***()* **VAR\_TERM**| *= new* **DATA\_TYPE VECTOR VAR\_TERM** | **CONSTANT VAR\_TERM**| *,* **IDENTIFIER VAR\_TERM**
* **ACCESS\_SPECIFIER** – *public* | *private* | *protected* | *internal* | *ϵ*
* **DATA\_TYPE** – **IDENTIFIER** | *int* | *char* | *double* | *float* | *String* | *decimal* | *bool* | *object* | *dynamic*
* **VECTOR** – *[***SIZE\_T***]* | *[]* | *ϵ*
* **MEMBER\_METHOD** – **ACCESS\_SPECIFIER DATA\_TYPE VECTOR IDENTIFIER** *(***ARGUMENT\_LIST***)* **METH\_TERM**
* **METH\_TERM** - *;* | *{***STATEMENTS***}*
* **ARGUMENT\_LIST** – **ARGUMENT ARG\_EXTENSION** | *ϵ*
* **ARG\_EXTENSION** – *,* ***ARGUMENT ARG\_EXTENSION***| *ϵ*
* **ARGUMENT** – **PASS\_TYPE DATA\_TYPE IDENTIFIER**
* **PASS\_TYPE** – *ref* | *out* | *ϵ*
* **STATEMENTS** – **STATEMENT STATEMENTS** | *ϵ*
* **STATEMENT** – **IF\_STATEMENT** | **FOR\_STATEMENT** | **WHILE\_STATEMENT** | **ASSIGNMENT** | **DECLARATION** | **METH\_CALL** | **RETURN** | **JUMP**
* **METH\_CALL** – **IDENTIFIER** *(***PARAM\_LIST***);*
* **PARAM\_LIST** – **PARAMETER PARAMETER\_EXTENSION** | *ϵ*
* **PARAMETER\_EXTENSION** - *,* **PARAMETER PARAMETER\_EXTENTION** | *ϵ*
* **PARAMETER** – **PASS\_TYPE IDENTIFIER**
* **ASSIGNMENT** – **IDENTIFIER ASSIGN****RIGHT\_SIDE**
* **ASSIGN** – *=* | *+=* | *-=* | *\*=* | */=*
* **RIGHT\_SIDE** – **METH\_CALL** | **EXPRESSION**
* **EXPRESSION** – **OPERAND EXPRESSION\_EXTENSION**
* **EXPRESSION\_EXTENSION** – **OPERATOR OPERAND EXPRESSION\_EXTENSION** | *ϵ*
* **OPERAND** – **CONSTANT** | **IDENTIFIER**
* **OPERATOR** – *+* | *-* | *\** | */* | *%*

*The remaining constructs have same regular definitions for all OOP languages and have been discussed in the paper whenever and wherever necessary.*