# The TBD Programming Language Reference Manual

## Preface

This is a reference manual for the TBD programming language. The TBD programming language is a minor programming language that is implemented by a minimalistic compiler written in the Rust programming language. Specifically, this manual serves as documentation of the major and minor components of TBD language as well as their function and use. Components that will be covered in this document will include:

* Lexical elements and corresponding values
* Basic Data Types and their implementation
* Available Operators, Expressions, Statements, and their usage
* The standard in which to declare, implement, and call Functions
* And The structure which manages variable scope and its application

While the TBD language compiler is written in the Rust programming language, this document does not contain documentation for the Rust programming language. Documentation for the Rust programming language can be found at <https://www.rust-lang.org/>

In this document, standard discrete mathematics and extended Backus-Naur form (EBNF) notation will be utilized when documenting linguistic and grammatical relationships. Similarly, this document will contain resources for neither, but documentation for discrete mathematics notation can be found at [www.javatpoint.com/language-and-grammar-in-discrete-mathematics](http://www.javatpoint.com/language-and-grammar-in-discrete-mathematics) , and for EBNF at <https://www.freecodecamp.org/news/what-are-bnf-and-ebnf/>.

## 1.0 Lexical Elements

After a program file is passed to the TBD compiler, its contents are broken into lexical elements referred to as Tokens. The TBD language recognizes 11 token categories:

### Brackets

### Separators

### Arithmetic

### Relational

### Logical

### Assignment

### Keywords

### Identifiers

### 1.9 Basic Types

### 1.10 Literals

### 1.11 End-Of-Input

## 2.0 Data Types

### 2.1 Primitives

### 2.2 Enumeration

### 2.3 Structures

### Arrays

## 3.0 Expressions and Operators

### Expressions

### Assignment Operators

### Arithmetic Operators

### Comparison Operators

### Logical Operators

### Statement and Declarations in Expressions

### 3.7 Operator Precedence

Put a table for this.

## 4.0 Statements

### 4.1 let Statement

### 4.2 Assignment Statement

### 4.3 if Statement

### 4.4 then Statement

### 4.5 else Statement

4.6 while Statement

4.7 return Statement

4.8 print Statement

## 5.0 Functions

### 5.1 Function Declaration

func function\_name(parameter\_list)

[

*Body of function*

]

### 5.2 Calling Functions

### 5.3 Function Parameters

### 5.4 The main Function

### 5.5 Recursive Functions

## 6.0 Program Structure and Scope

### 6.1 Program Structure

Contained in single text file, called from command line interface.

### 6.2 Scope

Essentially just describe how frames control the scope.

## 7.0 Sample Program

If we keep hardcoded examples in the final product, we should probably document them here.

### 7.1 Example

func factorial\_recursion(n)

[

if n < 2 [

return 1;

] else [

return n \* factorial\_recursion(n-1);

]

]

func factorial\_loop(n)

[

let p;

p = n;

while n > 0 [

n = n - 1;

p = p \* n;

]

return p;

]

func main()

[

let n;

n = 5;

print factorial\_loop(n);

print factorial\_recursion(n);

]

## References

* Document formatting derived from the *GNU C Reference Manual* and *PL\_F23\_Final*.*pdf*