Kennesaw State

Easily Interpreted Language (EIL) v1.0

Ethan Hunt

KSU CS Undergrad

Spring semester 2022

4/10/2022

Contents

[**Introduction to the EIL Language** 2](#_Toc98944011)

[**Basics of EIL** 2](#_Toc98944012)

[**EIL Basic Syntax and Semantics** 2](#_Toc98944013)

[**Main Method structure** 2](#_Toc98944014)

[**Assignment Statement structure** 2](#_Toc98944015)

[**Print Statement structure** 3](#_Toc98944016)

[**LOOP Construct structure** 3](#_Toc98944017)

[**IF Construct structure** 3](#_Toc98944018)

# **Introduction to the EIL Language**

The Easily Interpreted language (EIL) was developed to solve the problem of developing an interpreter for a more complex language. To do this both the syntax and semantics of EIL were created to provide an easy to interrupt language while still providing the experience of interpreting a programing language without all the extra possible permutations created by some of the higher functionality of languages like C++, java, etc.

## **Basics of EIL:**

EIL supports only one intrinsic data type which is the unsinged integer. Signed integers can be implemented but are not intrinsic. In addition to this data type EIL also supports up to 26 variables in one program. The variables do not require an initial declaration statement as all 26 variables are empty and available to be used to at the start of each program run time. Each variable is represented by a lowercase letter. As of v1.0 EIL only supports one function and a limited number of constructs.

## **EIL Basic Syntax and Semantics:**

In EIL all lexemes are separated by white space the grammar rules assume all white space has been separated out of input. EIL is punctuated by levels of statements with top level statements being the parent of another. Each level is specified by a unique punction lexeme

### **EIL Grammar/Structure:**

Let the nonterminal variables(**x**) be expresses as <**x**> and terminal variables be expressed as **x** Let the key variable of the level which can be either terminal or nonterminal be expressed as [**x**]

### **Main Method Structure:**

Each EIL program has the following structure

*Start M <body> End M*

The body segment holds many types of statement and constructs. This is known as parsing level zero and its only terminals tokens are Start, END, M. These terminal tokens are only allowed to be used once per program and must be used in each program written in EIL

### **Assignment Statement Structure:**

The *Assignment statement* has the following structure

*<variable\_ID> = <expression> <punctuation>*

An *expression* has the following structure

*<expression> -> <value> <operator> <value> <operator> … <value> <punctuation>*

Every <*expression>* must have exactly one less operator than values

<expression> operations are evaluated not by traditional “order of operations” by my first come serve in terms of operators.

Example: 1 + 2 – 3 \* 4 / 5

Iterations of evaluation

1: 1+2, expr

2: 3-3, expr

3: 0\*4, expr

4: 0 / 5, expr

5: 0

### **Print Statement structure:**

A *Print statement* structure has the following syntax

*P <punctuation>*

It prints out the values for all 26 variables

### **LOOP Construct structure:**

EIL’s Loop structure called loop which performs a given section of code predetermine set number of times.

A *LOOP* has the following structure

*L <varible\_ID> <loop\_body> <punctuation>*

### **IF Construct structure:**

EIL’s IF structure called IF which performs a given section of code based on a conditional

A *LOOP* has the following structure

*IF <varible\_ID> <logical\_operator> <varible\_ID> <IF\_body> <punctuation>*

## **EIL Examples**

// Ethan Hunt EIL v1.0

// Test Program

// date 3/20/2022

Start M

a = 0 ; x = 5 ;

L x

a = a + 1 : ;

P ;

End M

// Ethan Hunt EIL v1.0

// Test Program

// date 3/20/2022

Start M

a = 0 ; b = 2 ; c = 3 ; x = 1 ;

L c

IF d == a

a = a + 1 $ :

d = d + a + 1 : ;

P ;

End M

Start M

i = 6 ; z = 3 ; b = 1 ;

L i

IF a <= z

a = 1 + a $ : ;

P ;

End M

// Ethan Hunt EIL v1.0

// Test Program

// date 3/20/2022

Start M

a = 0 ; b = 2 ; c = 3 ; x = 1 ;

L c

IF d == a

a = a + 1 $ :

d = d + a + 1 : ;

P ;

End M