**Documentation for CRAWL**

**Team 25**

Arpit Mathur – 1209350970

Aveesha Sharma - 1207684331

Seema Suresh - 1209131348

**CONTENTS**

|  |  |
| --- | --- |
| Topic | Page |
| 1. Introduction | 3 |
| 2. Source code representation  3.2 Letters and Digits  3.3 Tokens  3.4 Identifiers  3.5 Keywords  3.6 Operators  3.8 Constants  3.9 Blocks  3.10 Declarations  3.11 Function declarations  3.12 If Else statement  3.13 While statement | 4 |
| 4. Sample Program | 8 |

**INTRODUCTION**

This documentation serves as a reference manual for the programming language CRAWL. It is not meant to be a tutorial to the language.

The language is strongly typed, simple and easily understood because of its strong influence from C, which is known to most beginners. Since the language is high level, the keywords are more or less words from English language which are easy to remember and self explanatory.

ANTLR4 tool (Another Tool for Language Recognition) reads in the CRAWL program and abstract syntax trees (AST) and intermediate code for CRAWL is generated. The runtime environment is built using Java.

The language is reliable, results consistent and tested rigorously. The programs can be run from the command prompt / terminal like any common shell program.

**SOURCE CODE REPRESENTATION**

Source code is Unicode text encoded in UTF-8. It is case sensitive and will result in error if used otherwise. Unlike C, there is no semicolon required at the end of the statements in a program and a new line at the end of the statement would signify the end of the statement.

**LETTERS & DIGITS**

letter = "A"... "Z", “a” … “z”

digit = "0" … "9"

**TOKENS**

Tokens form the vocabulary of the CRAWL language. There are the following classes: *identifiers*, *keywords*, and *operators*.

**IDENTIFIERS (ID)**

Identifiers name program entities such as variables and types. An identifier is a sequence of one or more letters and digits.

Identifier = letter {letter}

Examples: n, ip, xyz

**KEYWORDS**

The following keywords are reserved and may not be used as Identifiers:

* int, bool: Integer and Boolean Data Type declaration
* if, else, endif: conditional statement
* function: declare a function
* display: print strings or values of variables
* stack: inbuilt data structure used
* peek: to access top of the stack
* push: to add an element to the stack
* pop: to remove an element from the stack
* isEmpty: to check if the stack is empty or not
* BEGIN: program begins at this keyword
* END: program begins at this keyword

**OPERATORS**

The following character sequences represent operators and are used to perform arithmetic operations.

* Arithmetic Operators: +, -, \*, /, =
* Relational Operators

== : values equal

!= : values are not equal

> : greater than

>= : greater than equal to

< : less than

<= : less than equal to

**CONSTANTS**

A constant value is represented by an identifier denoting a constant.

**BLOCKS**

A *block* is sequence of declarations and statements within keywords BEGIN and END.

Example: BEGIN

program body

END

**DECLARATIONS**

A declaration binds a non-blank constant to a type variable name. Every identifier in a program must be declared as the language is strongly typed. No identifier may be declared twice in the same block.

Example:

int a = 7

**FUNCTION DEFINITIONS**

A function definition gives the steps/computing to a function along with its type and name.

Example:

function int a()

{

function body

return 0

}

**IF-ELSE STATEMENTS**

"If" statements specify the conditional execution of two branches according to the value of an expression. If the expression evaluates to true, the "if" branch is executed, otherwise, if present, the "else" branch is executed. The statement is concluded with an “endif”.

Example:

if (a == 1)

body

else

body

endif

**WHILE STATEMENTS**

A "while" statement specifies repeated execution of a block. The iteration is controlled by a condition, and a while clause.

Example:

while (a <= 9)

body

endwhile

**SAMPLE PROGRAM**

1. PROGRAM – IF ELSE DEMONSTRATION

BEGIN

int a = 5

int b = 7

function int show(){

if(a == 5)

display("a is 5")

else

display("a is not 5")

endif

return 0

}

function int main(){

display(“Let’s make a function call”)

show()

return 0

}

END

Output:

Let’s make a function call

a is 5