**Montana Tech MATLAB Algorithmic Language Framework**Version 0.9  
November 16, 2015

A. Frank Ackerman

*Computer Science  
Montana Tech*

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comment |
| 0.1 | 12/24/14 | Frank Ackerman | Initial version |
| 0.2 | 09/04/15 | Frank Ackerman | Added **Call** |
| 0.3 | 09/12/15 | Frank Ackerman | Added **Re-set** and **if-else** |
| 0.4 | 09/21/15 | Frank Ackerman | Add **Exchange,** **While**. **Break**; **Continue, Halt** |
| 0.5 | 09/27/15 | Frank Ackerman | Remove **Re-set** and add **Increment** |
| 0.6 | 10/04/15 | Frank Ackerman | Add **for** |
| 0.7 | 10/04/15 | Frank Ackerman | Add **Xlsread** to read an Excel worksheet |
| 0.8 | 10/31/15 | Frank Ackerman | Add **Invoke** |
| 09 | 11/16/15 | Frank Ackerman | Add **Seed** |

**Montana Tech *Programming with MATLAB* students:**

This standard encapsulates Dr. Ackerman’s decades of experience in the software industry, the IEEE software engineering standards, and many suggestions from various texts. You are invited to participate in the continuing evolution of this by studying it critically and making suggestions for its improvement and correction.

Purpose

The purpose of this document is to define a base language for the expression of algorithms that are to be translated into MATLAB programs. This base language is intended to be sufficient for expressing algorithms that can be translated into MATLAB computer programs or functions.

Introduction

This document is intended to provide *Programming with MATLAB* students with an easy-to-use and easy-to-understand method for unambiguously describing algorithms that can be translated into MATLAB programs, and that can be unambiguously mentally “executed” by fellow students

The constructs defined in this standard are divided into the following categories:

### **3. Sequential Constructs**

**4. Selection Constructs**

**5. Repetition Constructs**

Overview

A MATLAB algorithm that can be easily translated into a MATLAB program consists of a sequence of algorithm language constructs. Each construct begins with one of the “starter words” (initial letter capitalized) from the list below. Each element may begin with a label of the form A*nn* (see example in section 6) that is attached to the construct. When this is done all algorithm statements should be tabbed over so that the A*nn* labels are all aligned at the left margin.

As much as possible, language elements should be labeled in the sequence A00, A01, A02, .... . As the algorithm develops and new elements need to be inserted, these labels may be extended by adding a suffix of a, b, c, ...

The list of algorithm constructs given in the next section is the heart of this document. To facilitate clear intent, and to make an algorithm abstractly executable, only the listed constructs may be used.[[1]](#footnote-1)

The phrases after the keywords are not explicitly defined but should clearly describe the intention or meaning of the construct for that algorithm, as should any optional comment following terminating semicolon. Since algorithm text may included in a source file, which almost always uses a single fixed width font, ordinary English words should not be used to name objects. Use compound names that clearly reference the object or attribute. For example, use itemCount instead of count to reference an item count.

# Sequential Constructs

|  |  |
| --- | --- |
| **Call** *some function***;** | Invoke an FN or a MATLAB function. |
| **Clear Command Window;** | MATLAB:  clc |
| **Clear Workspace;** | MATLAB:  clear |
| **Compute** *text***;** | MATLAB:  *varName* = *some-computation*;[[2]](#footnote-2) |
| **Display** *text***;** | MATLAB: writing to the Command Window:  fprintf(*string*)  or  fprintf(*format-string*, *scalarDataItem*, ...) |
| **Exchange** *varble1 varbl2;* | Exchange the values of *varble1and varbl2* |
| **Halt;** | Used to quit execution in the Command Window and display a red message. |
| **Increment** *varble* **[by *N*];** | Add *N* to *varble*. Add 1 if “by *N*” is not present. |
| **Invoke** *text;* | Cause MATLAB to execute a command, function, program, or expression. |
| **Obtain** *text;* | Getting data from a user:  *varName* = input(*prompt*)  We will always terminate a prompt with ‘> ‘ |
| **Return;** | Exactly equivalent to return; |
| **Seed** *text***;** | Set the starting value of the random number generator. The expression: sum(100\*clock) is often used to give a different starting value every time the simulation is run. |
| **Set** *text***;** | General instruction. Usually involves setting a variable; sometimes via a function call and sometimes via an expression. |
| **Xlsread** *text***;** | To use xlsread() to read an MS Excel .xlsx file. |

# Selection Constructs

|  |  |
| --- | --- |
| **If (***some-condition***)**  *do-something*  **Else**  *do-something-else*  **End %If-Else** | Two-way selection |
| **If (***some-condition***)**  *do-something*  **End %If** | One-way selection |

# Repetition Constructs

|  |  |
| --- | --- |
| **Continue;** | Exactly equivalent to continue; |
| **Break;** | Exactly equivalent to break; |
| **For (**anIndex = *expr***)**  *do-something*  **End %For** | *expr* must evaluate to an array.  anIndex successively takes on the values in the columns of *expr* and *do-something* is executed with that value of anIndex.  Execution terminates after do-something is executed with anIndex set to the last value in the last column of *expr* or when a **Break** in *do-something* is executed.  Execution repeats when control reaches the **End** expression or when a **Continue** in *do-something* is executed |
| **While (***some-condition***)**  *do-something*  **End %While** | Basic repetition |

# Example

Problem statement: For any Fahrenheit temperature value in [-10^10, 10^10] compute the equivalent Centigrade temperature value.

An algorithm that gives this result is:

A01 **Clear Command Window**;

**Clear Workspace**;

**Set** format short;

A02 **Display** starting salutation;

A03 **Obtain** inFahr\_deg from user;

A04 **Compute** outCent\_deg:   
 outCent\_deg = 5/9 \* (inFahr\_deg - 32)

A05 **Display** outCent\_deg;

A06 **Display** ending salutation;

1. When we run into situations for which we have not yet defined a good algorithm “starter word” we will extend this standard to cover this situation. [↑](#footnote-ref-1)
2. In our MATLAB programs and functions we will always use a ; on assignment statements. When we want to display something we will use a **Display** construct; [↑](#footnote-ref-2)