

The MaSH Programming Language At the Methods Level

Andrew Rock
School of Information and Communication Technology
Griffith University
Nathan, Queensland, 4111, Australia
a.rock@griffith.edu.au

June 14, 2010

1 Introduction

This document defines the MaSH programming language at its methods level. The methods level encapsulates statements in methods (procedures and functions).

This document only describes what is different or new with respect to the statements and control levels.

2 Lexical syntax

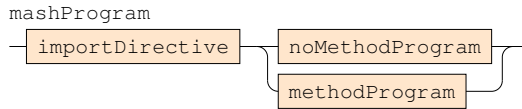
Same as for the statements and control levels.

3 Context-free grammar

Most of the grammar as presented for the statements and control levels is unchanged, except for the parts relating to how code is encapsulated and there is one new structured statement.

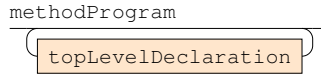
3.1 MaSH programs

A program still begins with an import directive that selects the environment within which this program will run. After the import directive comes the program: in the no-method program structure (as before); or in the method program structure (new).



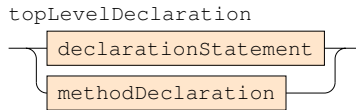
Method programs

A method program consists of a sequence of top-level declarations. At the top level we declare global variables, global constants and methods.



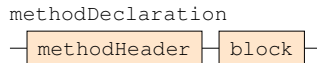
3.1.1 Top-level declarations

In a method program, we make a series of top-level declarations. At the top level we can declare and initialise variables and constants, and declare methods. All of the other kinds of statements must now be encapsulated within methods.



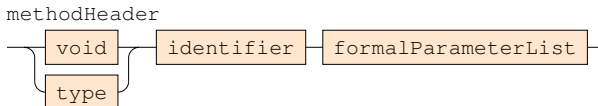
3.1.2 Method declarations

A method declaration has two parts, a heading followed by a block that will encapsulate the statements that perform the actions of the method.



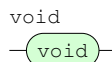
Method headers

The method header starts with either the type that the method will return (making the method a function) or void (making the method a procedure). Then follows the name of the method and then the formal parameter list.



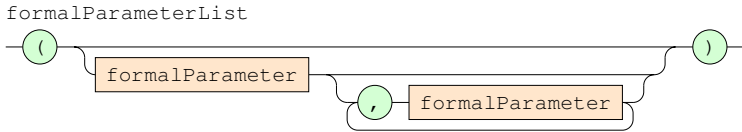
void

A void declaration consists only of the keyword `void`.



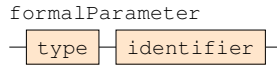
Formal parameter lists

A formal parameter list is a sequence of zero or more formal parameter declarations separated by commas and enclosed in parentheses.



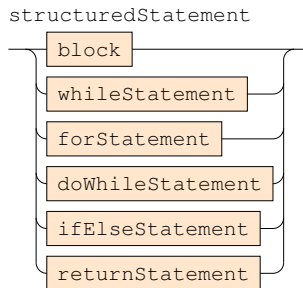
Formal parameters

A formal parameter declaration consists of the type of the parameter and introduces a name for this parameter.



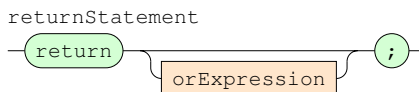
3.2 Structured statements

A structured statement now consists of all of the kinds of structured statements previously described, with the addition of the **return** statement.



3.2.1 return statements

A return statement consists of the keyword **return**, followed optionally by an expression, and then a semicolon. If the enclosing method is a function the expression should be there. If the enclosing method is a procedure, it should not.



4 Style conventions

All the conventions that apply at the statements and control levels continue to apply.

4.1 Indenting and spacing

- There should be a blank line between all top level declarations, except between closely related constants or variables.
- Since the statements inside a method are enclosed in a block, they should be indented within the braces.

5 Semantic rules

All semantic rules from the statements level still apply, with these additions.

5.1 The order of execution

5.1.1 No method programs

The statements in a no method program are executed from top to bottom (with repetitions and alternations as specified by the structured control statements in the program.)

5.1.2 Method programs

For method programs the order of execution is a little more complicated:

1. The top level constant and variable initialisations are performed first in the order they come in the program (from top to bottom).
2. What happens next depends on the kind of program this is, and that depends on the environment. There are two common kinds of environments:
 - A stand-alone program *usually* has a procedure named `main`. After the initialisations, the `main` method is invoked. It is up to `main` to call the other methods.
 - A plug-in, or service, for example an applet, is not really a stand alone program. It does not have a `main` method. Instead it has to provide other methods that will be defined by that environment, and will be called in an order that will make sense for that environment.

5.2 Variables, blocks and scope

The rules for when a variable is in scope are the same. The body of a method is a block like any other.

5.3 The return statement

5.3.1 Functions

The most important purpose of the `return` statement is to provide the value that will be returned by the function. Therefore if a method is a function (that is declared to return a type, not `void`) then there must be a `return` statement and it must have an expression that is assignment compatible with the type the function is declared to return.

The other thing a `return` statement does is force the method to finish.

5.3.2 Procedures

Procedures (declared with `void`) do not need a `return` statement at all. When the last statement in the methods block has finished, the method is finished. A `return` statement can be used to exit from somewhere else in the method, and for procedures, it must not return a value.