

The MaSH Programming Language At the Control 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 control level. The control level adds control structures, such as loops and selections.

This document only describes what is different or new with respect to the statements level.

2 Lexical syntax

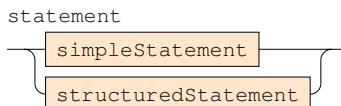
Same as for the statements level.

3 Context-free grammar

Most of the grammar as presented for the statements level is unchanged, except for the parts that describe statements.

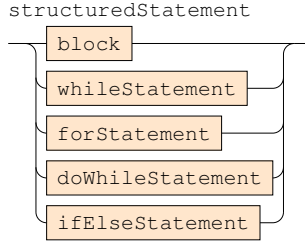
3.1 Statements

There are now two general categories of statements: simple statements (as in the statements level); and structured statements (new).



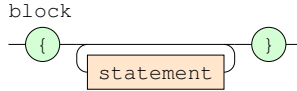
3.2 Structured statements

A structured statement is either: a block; a **while** statement; a **for** statement; a **do-while** statement; or an **if-else** statement.



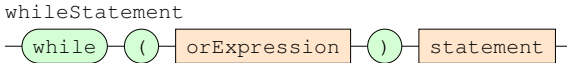
3.3 Blocks

A block is a sequence of zero-or-more statements enclosed in a pair of braces.



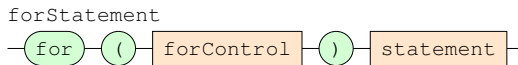
3.4 while statements

A **while** statement is a loop that executes a statement repeatedly while a boolean expression is true. A **while** statement consists of the keyword **while**, a boolean expression enclosed in parentheses (the *loop guard* or *condition*), followed by the statement that will be repeated. That statement is usually a block.



3.5 for statements

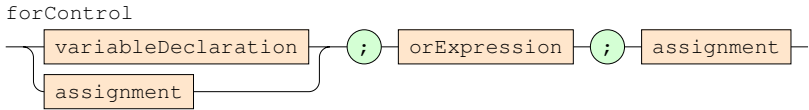
A **for** statement is a loop that should be used to repeat a statement a definite number of times. A **for** statement consists of the keyword **for**, then a complicated three-part definition that controls how many times the loop will execute (a *for control*, defined below) enclosed in parentheses, followed by the statement to be repeated. That statement is usually a block.



3.5.1 for controls

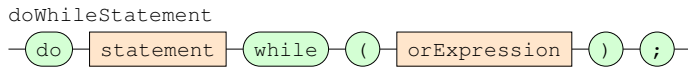
The three parts that control a for loop are: the declaration and initialisation or reassignment of a loop counter variable; a condition (a boolean expression);

and an assignment that counts off each execution of the loop statement. These three things are separated by semicolons.



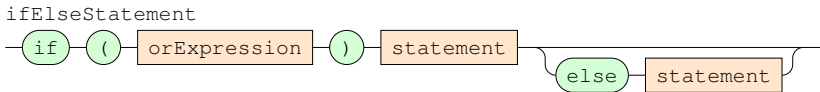
3.6 do-while statements

A **do-while** statement is a loop that executes a statement once and then repeats it while a boolean expression is true. A **do-while** statement consists of the keyword **do**, the statement (usually a block); the keyword **while**; a boolean expression enclosed in parentheses (the *loop guard* or *condition*), followed by a semicolon.



3.7 if-else statements

An **if-else** statement executes a statement if a condition is true. If the condition is not true, then the statement in an optional **else** part is executed. The **if-else** statement starts with the keyword **if**, followed by the condition in parentheses, followed by the statement (usually a block) to execute if the condition is true. These may optionally be followed by the keyword **else** and an alternate statement to execute if the condition is false.



4 Style conventions

All the conventions that apply at the statements level continue to apply.

4.1 Indenting

Each of the structured statements described above contain statements within them. Those statements can be structured statements themselves. All programmers use indenting to show the structure of code, that is what statements are inside each other. The deeper inside, the more a statement is indented.

Indenting is done with a consistent number of spaces each time. Most programmers will indent by three or four spaces. I prefer 3.

Warning: the compiler ignores the indenting. If you get the indenting wrong, you may misunderstand the real structure of the program. Braces may help.

4.2 Braces

Each of the structured statements described above contain *single* statements within them. Usually more than one thing needs to be done, so a block is used to make a sequence of statements appear as one. Making a block is as simple as enclosing the statements within braces.

It is actually a good habit to use braces even when there is only one statement within the structured statement. This makes the structure even clearer and may avoid errors as extra statements get inserted later.

There is a variety of opinions about how the use of braces should be combined with indenting. The examples in the next section show the most popular choices.

4.3 Examples

Code without indenting, good spacing or consistent use of braces:

```
import console;for(int i=0;i<10;i=i+1){for(int j=0;j<10;j=j+1)if(i<j)print('+');else print('-');println();}
```

Code with indenting, good spacing and use consistent of braces (method 1):

```
import console;

for (int i = 0; i < 10; i = i + 1) {
    for (int j = 0; j < 10; j = j + 1) {
        if (i < j) {
            print('+');
        } else {
            print('-');
        }
    }
}
println();
}
```

Code with indenting, good spacing and use consistent of braces (method 2):

```
import console;

for (int i = 0; i < 10; i = i + 1)
{
    for (int j = 0; j < 10; j = j + 1)
    {
        if (i < j)
        {
            print('+');
        }
    }
}
```

```
        else
        {
            print('-');
        }
    }
    println();
}
```

Method 1 is more compact and I prefer it, but method 2 may be easier to get right because it is easier to check which opening and closing braces go with each other.

Which method should you use? Use the *Golden Rule* for selecting programming styles: Use whichever you want, *but be consistent!*

5 Semantic rules

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

5.1 Variables, blocks and scope

- A variable declared inside a block is *in scope* (visible and usable) within that block, but *not* in scope outside of that block.
- A variable declared outside a block is in scope within that block.
- If you try to initialise a variable inside a structured statement, the compiler may still complain that it might not be initialised after that. The structured statements control whether some statements execute or not. The initialisation might not happen, and the compiler guesses it won't.