Programing Language Concepts Coursework
Programming Language Manual Syntax and Grammar Analysis

Samuel Edric Mott (sem1n14), Georgios Stefanos Pnevmatikakis (gsp1g15)

February - May, 2017

Contents

1 Running code from the command-line

To run your program code from the command line, simply open the console and type the name of the compiler (by default, mysplinterpreter), followed by the name of the program file. For example:

mysplinterpreter helloWorld.spl

2 Variables

2.1 Variable declaration initialisation

- <vName> = <value>; Simple Variable declaration and initiation.
- <vName> = <vName>; Simple Variable declaration and initiation from a variable.
- global <vName> = <value>; Global Variable declaration and initiation.
- <vName> = <value>; Variable value assignment.

2.2 Basic Types

- Integer: int
- Double: double
- String: string
- Boolean: bool
- Null: null
- End-of-file: eof

2.3 Type Conversion

- string_to_int(<string>); Converts the given string to an integer.
- string_to_bool(<string>); Converts the given string to an bool.
- string_to_double(<string>); Converts the given string to an double.
- var_to_string(<vName>); Converts the given variable to an string.

2.4 Other Types

2.4.1 Tables

1.3.1.1 Table declaration initialisation

- <vName> = {}; Simple table declaration and initiation with auto-defined keys.*
- <vName> = {<value>, <value>, ...}; Simple table declaration and initiation through direct input with auto-defined keys.
- <vName> = {<key>: <value>, <key>: <value>, ...}; Simple table declaration and initiation through direct input with self-defined keys.

1.3.1.2 Table Functions

- <tbl>[<key>]; Accessor for the value of that value. If none was specified then assume the index is from 0 to the number of elements given.
- #<tbl>; Function to get the length of table.
- <tbl>.sort(); Function to sort the table.
- <tbl>.append(<value>); Function to add an new value with an auto-defined keys.
- <tbl>.add(<key>, <value>); Function to add an new value with an self-defined keys.
- <tbl>.remove(<key>); Function to remove an key-value pair from a table.

3 Mathematical Operations

3.1 Main Mathematical operations

3.1.1 Addition

- <int> + <int>; Simple addition for integers.
- <double> + <double>; Simple addition for doubles.
- <string> + <string>; Simple addition for strings.
- <vType> += <vType>; Addition and replacement of the values to the original variable.

3.1.2 Subtraction

- <int> - <int>; Simple subtraction for integers.
- <double> - <double>; Simple subtraction for doubles.
- <vType> -= <vType>; Subtraction and replacement of the values to the original variable.

3.1.3 Multiplication

- <int> * <int>; Simple multiplication for integers.
- <double> * <double>; Simple multiplication for doubles.
- <vType> *= <vType>; Multiplication and replacement of the values to the original variable.

3.1.4 Division

- <int> / <int>; Simple division for integers.
- <double> / <double>; Simple division for doubles.
- <vType> /= <vType>; Devision and replacement of the values to the original variable.

3.1.5 Remainder

- <int> % <int>; Simple remainder for integers.
- <vType> %= <vType>; Modulus and replacement of the values to the original variable.

3.1.6 Simple addition or subtraction of 1

- <int>++; Simple increment of 1.
- <int>--; Simple decrease of 1.

3.2 Comparing operations

- <vName> == <vName>; Checks for equality.
- <vName> != <vName>; Checks for in-equality.
- <vName> > <vName>; Checks for greater.
- <vName> >= <vName>; Checks for greater or equal.
- <vName> < <vName>; Checks for less.
- <vName> <= <vName>; Checks for less or equal.

3.3 Logical Mathematics

- <compTest> && <compTest>; Logical 'AND'.
- <compTest> || <compTest>; Logical 'OR'.
- !<compTest>; Logical 'NOT'.

4 Loops

- break; A command to force a loop to terminate.

4.1 'FOR' Loop

- for (<expr>; <logicTest>; <expr>){<code>} Simple 'for' loop used in most programming languages. The first <expr> is used to initialise any counter variables in the loop. <logicTest> is performed before each iteration of a loop; if it returns false, then the loop will exist, otherwise the expressions in <code> will be executed.

4.2 'While' loop

- while (<logicTest>){<code>} Simple 'while' loop used in most programming languages. <logicTest> is tested before <code> is excuted; if it returns false, then the loop exists, else it continues.
- do {<code>} while (<logicTest>) Simple 'do while' loop used in other programming languages. <code> is executed at least once before <logicTest> is tested.

5 Logical Branching

- if (<logicTest>) {<code>} If <logicTest> returns true, then <code> will be executed, else it won't.
- if (<logicTest>) {<code>} else {<code>} If <logicTest> returns true, then the first <code> will be executed, else the second <code> will be executed.
- if (<logicTest>) {<code>} else if (<logicTest>) {<code>} [1] else {<code>} Simple chained if-else statements.

6 Functions

6.1 Declaration

- function <fName>(<varDec>*, <varDec>, ...){<code> return** <vName>;} Simple function declaration.

Anotations: *Variable declarations must not be initiating them.
**Return command needed only when the function returns something.

6.2 Pre-defined functions

6.2.1 File input-output

- read(<string>); Function to read from a file with the specified file name. Reads the entire file and returns the contents as a string.
- write(<string>, <vName>); Function to write a variable or a string to a file with the given file name.

6.2.2 File input-output

- input(); Function to read a line from the command line. Returns a string if there's input, else returns eof.
- print(<vName>); Function to write a variable or a string to the command line.
- println(<vName>); Function to write a variable or a string to the command line with a newline after.

7 Commenting

- /* <text>*/ Comment block. Nothing inside will be executed.

[1] Unlimmited number of else if clauses can be used.

Appendices

A Example Code

A.1 Looping constructs

```
/* Basic for-loop, printing numbers 0-100 */
for (i = 0; i <= 100; i++) {
    println(i);
}

/* Basic while-loop searching a table */
exampleTable = {"a", "b", "c", "d", "e"};
i = 0;
while (exampleTable[i] != "d") {
    println("Haven't found 'd' yet");
    i++;
}
print("Found 'd' at index: ");
println(i);
```

A.2 Creating and printing a table

```
/* Auto-indexed table printed to the command-line. */
exampleTable = {"a", "b", "c", "d"};
for (i = 0; i < #exampleTable; i++) {
    print(i);
    print(" : ");
    println(exampleTable[i]);
}

/*
Command-Line Output:
0 : a
1 : b
2 : c
3 : d
*/

/* Manually indexed table printed to the command-line. */
testVar = 5;
exampleTableV2 = {"test":testVar, "hello":"b", "world":90, 0:1};
println(exampleTableV2["test"]);
println(exampleTableV2["hello"]);
println(exampleTableV2["world"]);
println(exampleTableV2[0]);

/*
Command-Line Output:
5
b
90
1
*/
```

A.3 Conditional statements

```
if (true) {
  println("This statement is always reached");
}

test = "a";
if (test == "a") {
  println("test is equal to a!");
}

if (test == "b") {
  println("test equals b!");
} else {
  println("test isn't equal to b...");
}

if (test == "b") {
```

```
  println("test now equals b?");
} else if (test == "a") {
  println("test really is equal to a.");
} else {
  println(test);
}

/*
Command-line Output:
This statement is always reached
test is equal to a!
test isn't equal to b...
test really is equal to a.
*/
```

A.4 Creating a function

```
/* Basic function */
function sayHello() {
    println("Hello world!");
}

/* Basic function with a parameter */
function sayGoodbye(name) {
  println("Goodbye, " + name);
}

/* Basic function that returns a value */
function echo(message) {
  return message;
}

/* Printing the contents of a table nicely */
function print_table(table) {
    print("{");
    for (i = 0; i < #table; i++) {
        if (table[i] != null) {
            print(table[i]);

            if (i != #table - 1) {
                print(", ");
            }
        }
    }
    println("}");
}

sayHello();
sayGoodbye("world");
println(echo("Is there an echo?"));

test = {"a", "b", "c"};
print_table(test);

/*
Command-line Output:
Hello world!
Goodbye, world
Is there an echo?
{a, b, c}
*/
```

A.5 Processing a text file

```
data = read("textFile.txt");

/* Parse files into seperate lines */
lines = {};
line = "";
for (i = 0; i < #data; i++) {
  if (data[i] == "\n") {
        lines.append(line);
        line = "";
    } else {
        line += data[i];
    }
```

```
}

/* Sort and print lines */
lines.sort();
for (i = 0; i < #lines; i++) {
    println(lines[i]);
}
```

A.6 Reading piped input from the command-line

```
/* Read and print all piped in input */
while (true) {
  str = input();

    if (str == eof) {
      break;
    }

    println(str);
}
```