Computing Studies

Software Design and Development

Reading & Writing Haggis Pseudocode

Buckhaven High School

Version 1

Contents

How to use this booklet

This booklet has been written to aid covering the following content in National 4 and National 5 Computing.

	National 4 	National 5 
Computational Constructs	Exemplification and implementation of the following constructs: • expressions to assign values to variables • expressions to return values using arithmetic operations (+,-,*,/,^) • execution of lines of code in sequence demonstrating input - process - output • use of selection constructs including simple conditional statements • iteration and repetition using fixed and conditional loops	Exemplification and implementation of the following constructs: • expressions to assign values to variables • expressions to return values using arithmetic operations (+,-,*,/,^,mod) • expressions to concatenate strings and arrays using the operator • use of selection constructs including simple and complex conditional statements and logical operators • iteration and repetition using fixed and conditional loops • pre-determined functions (with parameters
Data Types and Structures	string numeric (integer) variables graphical objects	string, character numeric (integer and real) boolean variables 1-D arrays
Algorithm Specification		Exemplification and implementation of algorithms including • input validation
Design notations	• graphical to illustrate selection and iteration • other contemporary design notations	• Pseudocode to exemplify programming constructs • other contemporary design notations

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Sept 2013

What is Haggis?

> *Definition - Haggis is a standardised design methodology used by the Scottish Qualification Authority (SQA) in place of a programming language for the purpose of asking coding questions in assessments or exams.*

Haggis is very similar to a programming language in that is has strictly defined syntax and rules. The inflexibility of Haggis syntax is not a usual feature of pseudocode as users would usually write pseudocode algorithms in natural language. This inflexibility is a necessary evil as the purpose of Haggis to set a standard across Scotland and therefore ensure that both staff and pupils are well prepared for exam questions.

This guide will help staff prepare their pupils for the new exams by explaining the ins and outs of Haggis syntax in reference to the following sections:

- Assigning values to variables
- Expressions to output data
- Expressions using arithmetic operators
- Expressions to concatenate strings
- Selection constructs including simple/complex conditions and logical operators
- Iteration and Repetition using fixed and conditional loops
- Pre-defined functions with parameters

Note that Haggis syntax only applies to the final refinement of a problem. In your pseudocode's main algorithm you should outline a problem that requires further refinement by using < >.

For example, the algorithm below shows two completed "Haggis" lines and four lines that require further refinement.

```
Line 1    RECEIVE numberOfItems FROM (INTEGER) KEYBOARD
Line 2    <Calculate the total cost of purchases>
Line 3    <Get valid type of customer>
Line 4    SET vatTotal TO 0.175*totalCost
Line 5    <Calculate final cost>
Line 6    <Display purchase details>
```

This guide may also be given to pupils as a reference document to help them interpret pseudocode.

It is important to note that pupils will never be expected to write Haggis code in an exam. They will always be given the option "using pseudocode or a programming language of your choice" when answering coding questions.

Formatting rules of Haggis?

1. **Keywords**
   All Haggis command words should be capitalised.
   SET
   FOR
   WHILE etc

2. **Line Numbers**
   Haggis uses a numbering system for lines of code and refinements. Lines should be numbered as shown below using a capital L and a single space before each number.
   Line 1
   Line 2
   Line 3

   A refinement of line 2 would be written as:
   Line 2.1
   Line 2.2
   Line 2.3

3. **Indentation**
   The beginning and end of some constructs (REPEAT..UNTIL, IF..END IF) should be highlighted by indenting the code between. For example,

   Line 1 REPEAT
   Line 2 SET total = total + 5
   Line 3 UNTIL total =100

   Ensure code does not look like this,
   Line 1 REPEAT
   Line 2 SET total = total + 5
   Line 3 UNTIL total =100
   To avoid confusion, staff should use tab markers or a table to ensure code is clearly lined up.

4. **Variable Names**
   Simple variable names (one word) should be written in lower case. For example,
   total
   surname

   Where the user wishes to use a longer variable name (two or more words) the second word should be emphasised with a capital letter. For example,
   firstName
   secondNumber

5. **Data**
   Where a numeric value is used in Haggis the number on its own is enough.
   SET number TO 973

   The use of text is indicated by using “”.
   SET name TO “Greg”

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Sept 2013

Assigning Values to Variables

The command words **SET** & **TO** are used to assign values to variables.

Numeric assignment example:

SET number TO 973

String assignment example:

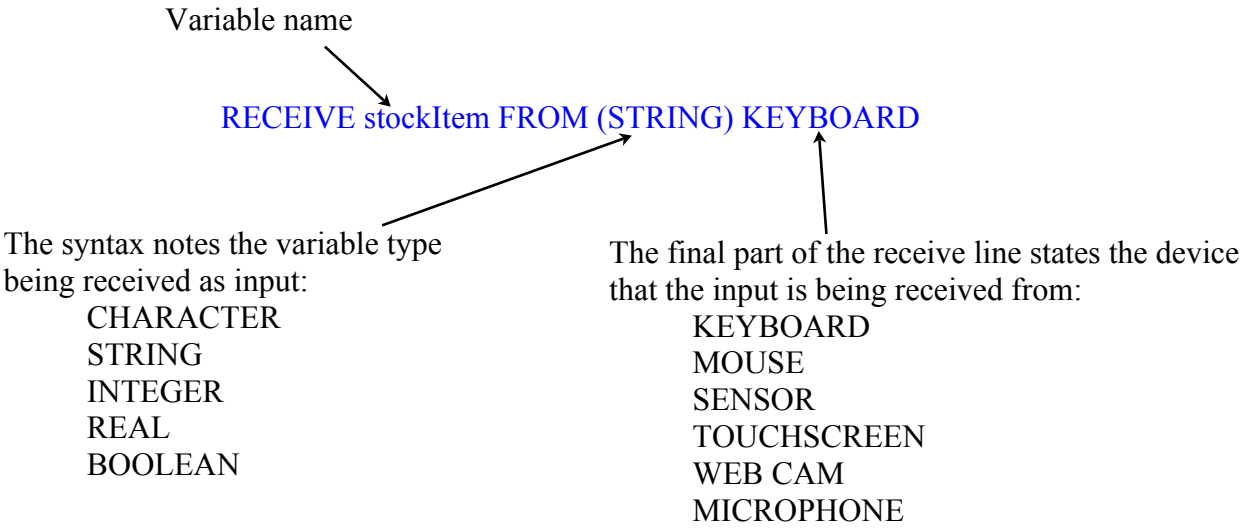
SET name TO “Greg”

Array assignment examples:

SET names[3] TO “Clare”

SET names[loop] TO “John”

Variables may also be assigned values as part of an input statement - **RECEIVE & FROM**



Note, due to the level of programming at N4/N5, the majority of questions will probably receive simple input from a keyboard.

Expressions to Output Data

The command words **SEND** & **TO** are used to output data to a variety of devices. A few examples are listed below.

Numeric	SEND 23 TO DISPLAY
Variable	SEND total TO DISPLAY
String	SEND “The total is:” TO DISPLAY
Alternative	SEND on TO MOTOR
	SEND “print this text” TO PRINTER

Expressions Using Arithmetic Operators

Arithmetic operators (+,-,/,*,^,mod) are normally used with a **SET** or **SEND** :

```
SET number TO 911+34
SET number TO 34/7
SET number TO 23-7
SET number TO 2^2
SEND 34*78 TO DISPLAY
SEND mod(34.5) TO DISPLAY
SET total TO num1 + num2
```

Arithmetic operators may also be combined in statements:

```
SET number TO (45+5)/2
```

Arithmetic operators may occasionally be seen in conditional statements:

```
IF numOne + numTwo >= 100 THEN SEND "Pass" TO DISPLAY
```

Expressions to Concatenate Strings

Text, number & variables may be concatenated with an ampersand (**&**) and enclosed by **[ ]** brackets:

```
SET errorMessage TO ["Device failed due to fault number" & faultNumber]
SEND ["Device failed due to fault number" & faultNumber] TO DISPLAY

SEND ["Player" & playerNumber & "'s score is" & playerScore] TO TOUCHSCREEN
```


Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Sept 2013

Selection Constructs Including Simple Conditions and Logical Operators

The command words **IF** and **THEN** are used to make decisions.

An **IF** statement with a simple condition can take two forms:

Single Line

```
IF temperatureNow >= 100 THEN SEND "Water is in a gaseous form at this temp" TO DISPLAY
```

Extended Statement

```
IF temperatureNow >= 100 THEN
    SEND "Water is in a gaseous form at this temp" TO DISPLAY
    SEND "This is called steam" TO DISPLAY
END IF
```

An **ELSE** statement may be used to show what should take place should a condition be false.

```
IF temperatureNow >= 100 THEN
    SEND "Water is in a gaseous form at this temp" TO DISPLAY
    SEND "This is called steam" TO DISPLAY
ELSE
    SEND "Water is a liquid or a solid at this temp" TO DISPLAY
END IF
```

Selection Constructs Including Complex Conditions and Logical Operators

Logical operators (**AND**, **OR**, **NOT**) may be used to create complex conditions

```
IF temperatureNow > 0 AND temperatureNow < 100 THEN
    SEND "Water is in a liquid at this temp" TO DISPLAY
    SEND "Lower the temperature to form a solid" TO DISPLAY
END IF

IF temperatureNow <= 0 OR temperatureNow >= 100 THEN SEND "Water is not a liquid at this
temp" TO DISPLAY

IF NOT(temperatureNow > 0 AND temperatureNow < 100) THEN
    SEND "Water is not a liquid at this temp" TO DISPLAY
END IF
```

Iteration and Repetition Using Fixed Loops

Code may be repeated a fixed number of times using **FOR**, **FROM**, **TO**, **DO**, **END FOR**.

```
FOR counter FROM 1 to 10 DO
    GET nextInput FROM (REAL) KEYBOARD
    SET totalCost TO totalCost + nextInput
END FOR
```

The variable used in the loop ("counter" in the above example) may also be used in the pseudocode. A **STEP** may also be introduced (for example - STEP 2) to control how the loop counts.

The example below stores 10 names in an array in reverse order.

```
FOR counter FROM 1 to 10 STEP -1 DO
    SEND ["Please enter the name of user number" & counter] TO DISPLAY
    RECEIVE nextUserName FROM (STRING) KEYBOARD
    SET nameList[counter] TO nextUserName
END FOR
```

Alternatively, a fixed loop may also be written using the **REPEAT**, **TIMES**, **END REPEAT** commands:

```
REPEAT 10 TIMES
    RECEIVE nextValue FROM (REAL) KEYBOARD
END REPEAT
```

Additionally a **FOR EACH** loop may be used for structures with a set length,

```
FOR EACH character FROM "This is a test"
    IF <character does not equal a letter of the alphabet>  THEN
        SEND [character & " is not a letter"] TO DISPLAY
    END IF
END FOR EACH
```

or

```
FOR EACH time FROM runnersTimesArray
    SET totalTimes TO totalTimes + time
END FOR EACH
SET averageTime TO totalTimes / numberOfRunners
```

Iteration and Repetition Using Conditional Loops

Loops may also be ended with a pre or post condition using **WHILE**, **END WHILE** or **REPEAT**, **UNTIL**. These are formatted as shown below:

Pre-condition

```
RECEIVE pressureLevel FROM (REAL) SENSOR
WHILE pressureLevel < 0 OR pressureLevel > 200 DO
    IF pressureLevel  < 0 OR pressureLevel  > 200 THEN
        SEND "Error in Reading.  Reset Sensor!" TO DISPLAY
    END IF
    RECEIVE pressureLevel FROM (REAL) SENSOR
END WHILE
```

Post-condition

```
REPEAT
    RECEIVE pressureLevel FROM (REAL) SENSOR
    IF pressureLevel < 0 OR pressureLevel > 200 THEN
        SEND "Error in Reading.  Reset Sensor!" TO DISPLAY
    END IF
UNTIL pressureLevel >= 0 AND pressureLevel <= 200
```

Pre-Defined Functions with Parameters

Predefined functions may differ from language to language. As yet the only mentions of pre-defined functions in N5 are length() in the SQA Haggis document and move(), rotate() from the graphical question in the specimen paper.

A function or sub-program is noted by using brackets **( )**.

If an item of data or a variable is passed into the function it is placed inside the brackets.

```
MOVE(5)

ROTATE(90)

SET lengthOfWord TO length("computing")

SET temperature TO currentTemp("Dining Room")

SET AsciiCode TO ord(character)

SET maxValue TO findMax(shoesizeList[ ])
```

In some cases the brackets may be left blank. This may be relevant to calling a sub-program.

```
GetValidName()
```

A Few Worked Examples

Problem (2008-2009, Intermediate 2 Coursework task**)**

The manager of a school cafeteria wants to use a computer system to calculate how much each customer has to pay. Members of staff have to pay VAT on their purchases but pupils do not. If the customer is a member of staff then the program will calculate the VAT and add it to the total cost.

VAT is calculated using the formula:

$$VAT = 0.175 \times \text{total cost}$$

The system requires the following inputs:

- How many items the customer has to pay for
- The price of each item in pounds
- Whether the customer is a pupil or a member of staff (P for pupil and S for staff)

The output from the program should display the total cost of purchases, the type of customer, the amount of VAT to be paid and the final cost e.g.

```
Total cost of purchases: 2.38
Type of customer: S
VAT: 0.42
Final Cost: 2.80
```

Solution in Haggis Pseudocode (Algorithm and Refinements)

```
Line 1    RECEIVE numberOfItems FROM (INTEGER) KEYBOARD
Line 2    <Calculate the total cost of purchases>
Line 3    <Get valid type of customer>
Line 4    SET vatTotal TO 0.175 * totalCost
Line 5    <Calculate final cost>
Line 6    <Display purchase details>

Line 2.1      SET totalCost TO 0
Line 2.2      FOR loop FROM 1 TO numberOfItems DO
Line 2.3          RECEIVE itemPrice FROM (REAL) KEYBOARD
Line 2.4          SET totalCost TO totalCost + itemPrice
Line 2.5      END FOR

Line 3.1      REPEAT
Line 3.2          RECEIVE customerType FROM (CHARACTER) KEYBOARD
Line 3.3          IF customerType ≠ P AND customerType ≠ S THEN
Line 3.4              SEND "Please enter P or S" TO DISPLAY
Line 3.5          END IF
Line 3.6      UNTIL customerType = P OR customerType = S

Line 5.1      IF customerType = P THEN SET  finalCost = totalCost
Line 5.2      IF customerType = S THEN SET  finalCost = totalCost + vatTotal

Line 6.1      SEND ["Total cost of purchases:    " & totalCost] TO DISPLAY
Line 6.2      SEND ["Type of customer:           " & customerType] TO DISPLAY
Line 6.3      SEND ["VAT:                        " & vatTotal] TO DISPLAY
Line 6.4      SEND ["Final Cost:                 " & finalCost] TO DISPLAY
```

Note that 4 of the main algorithm steps are written in a less formal style using < >. This shows that these 4 steps require further refinement.

Problem (Guess Number, Standard Grade Credit Programming Task**)**

A program is required to prompt the user to guess a randomly-chosen whole number between 1 and 20.

The input should be validated. If the guess is incorrect, the user should be told if the target number is bigger or smaller. This process should continue until the target number is guessed correctly.

The user should then be told how many valid guesses were made.

An example of output is shown to the right. The output from your program may look different but must meet the specification.

```
I am thinking of a whole number between 1 and 20

What is the number?
83
Enter a whole number between 1 and 20
0
Enter a whole number between 1 and 20
4.9
Enter a whole number between 1 and 20
11
My number is smaller than your guess.
What is the number?
3
My number is bigger than your guess.
What is the number?
7
Correct. I was thinking of 7

The number of valid guesses was 3
```

Solution in Haggis Pseudocode (Algorithm and Refinements)

```
Line 1    SET chosenNumber TO <random whole number between 1 and 20>
Line 2    REPEAT
Line 3        SET numberOfGuesses TO numberOfGuesses + 1
Line 4        <Get validGuess from user>
Line 5        <Display appropriate message>
Line 6    UNTIL chosenNumber = validGuess
Line 7    SEND ["Correct. I was thinking of " & chosenNumber] TO DISPLAY
Line 8    SEND ["The number of valid guesses was " & numberOfGuesses] TO DISPLAY

Line 4.1      REPEAT
Line 4.2          RECEIVE validGuess FROM (INTEGER) KEYBOARD
Line 4.3          IF validGuess < 1 OR validGuess > 20 THEN
Line 4.4              SEND "Enter a whole number between 1 and 20" TO DISPLAY
Line 4.5          END IF
Line 4.6      UNTIL validGuess >= 1 AND validGuess <= 20

Line 5.1      IF validGuess < chosenNumber  THEN SEND "My number is bigger than your guess." TO
              DISPLAY
Line 5.2      IF validGuess > chosenNumber THEN SEND "My number is smaller than your guess." TO
              DISPLAY
```

Problem (2009-2010, Intermediate 2 Coursework task**)**

Greg wants a piece of software to display the tracks he has burned onto the CD-R along with the duration in seconds of each track. He also wants to display the total time of all the tracks on the CD-R.

The program should initially ask the user for the number of tracks to be listed. This should be validated. At least one track and no more than twenty can be burned onto a CD-R.

The program requires the following inputs:

- the number of tracks to be burned
- the title of each track
- the length in seconds of each track.

An example of the required output is shown below.

Supernatural Superserious	204 seconds
Another Way to Die	263 seconds
Jealous Guy	234 seconds
CD-R running time	701 seconds

Solution in Haggis Pseudocode (Algorithm and Refinements)

```
Line 1    SET totalRunningTime TO 0
Line 2    <Get valid numberOfTracks from user>
Line 3    FOR counter FROM 1 TO numberOfTracks  DO
Line 4        GetTitleAndLength()
Line 5        SET totalRunningTime TO totalRunningTime + trackLength[counter]
Line 6    END FOR
Line 7    <display track titles and track lengths>
Line 8    SEND ["CD-R running time          " & totalRunningTime] TO DISPLAY

Line 2.1    REPEAT
Line 2.2        RECEIVE numberOfTracks  FROM (INTEGER) KEYBOARD
Line 2.3        IF numberOfTracks  < 1 OR numberOfTracks  > 20 THEN
Line 2.4            SEND "Please enter number of tracks between 1 and 20" TO DISPLAY
Line 2.5        END IF
Line 2.6    UNTIL numberOfTracks >= 1 AND numberOfTracks <= 20

Line 4.1    RECEIVE trackTitle[counter]  FROM (STRING) KEYBOARD
Line 4.2    RECEIVE trackLength[counter]  FROM (REAL) KEYBOARD

Line 7.1    FOR counter FROM 1 TO numberOfTracks  DO
Line 7.2        SEND [trackTitle[counter] & trackLength[counter]] TO DISPLAY
Line 7.3    END FOR
```


Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Sept 2013