Lecture C4: Strings, IF-statement, bits and bytes

Response to 'Muddiest Part of the Lecture Cards'

(63 respondents, out of 72 students)

2) ***What is Little/Big Endian?*** (1 student)
I did not have time to cover that in class today. Will on Friday.

Endianness refers to which bytes (1 byte = 8 bits) are most significant in a multi-byte data type. In a big endian machine, the leftmost byte (the one with the lower address) is the most significant. In a little-endian machine, the righmost byte is the most significant. For example the 32 bit hexadecimal number 16#ABCDEF42# will on a big-endian machine be stored in 4 consecutive bytes in the following order: AB CD EF 42. The same 32 bit number would on a little-endian machine be saved in the following byte order: 42 EF CD AB. Examples of little-endian machines are the Intel x86. Examples of big-endian machines are: Motorola 68k and the PowerPC.

3) ***Could we get more practice using hexadecimal code?*** (1 student)
Yes, absolutely! Maybe already in tomorrows recitation.

4) ***What is FOR?*** (1 student)
Ada provides the FOR statement for definite iteration. Definite iteration is where the set of actions is performed a known number of times. The number might be determined by the program specification, or it might not be known until the program is executing, just before starting the iteration. The FOR statement will be covered in later lecture.

5) ***Binary numbers*** (3 students)
B inary numbers only use ones and zeros as their numbers. A binary number can be repsesented by any set of bits (binary digits). We will see example of this in recitation tomorrow. Just as an example, the decimal numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 are represented using 4 bits in the following way: 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100. Arithmetic in binary is much like arithmetic in other numberal systems:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (the 1 is a Carry/overflow)

6) ***Concatenation of strings*** (and other Stirng questions) (5 students)
The string concatenation operator **&** applied to 2 strings S1 and S2, concatenates, or "pastes together",

its two arguments. For example the statement

S3 := S1 & S2;

stores in S3 the concatenation of S1 and S2. The length of S3 must match the sum of the length of S1 and S2, if it does not, a constarint error will be raised. [Feldman page 436]

7) ***What are the Boolean operators, and when can XOR be useful?*** (2 students)
There are four Boolean operators: AND, OR, NOT, XOR.

The logical operator XOR (exclusive or) yields the result TRUE when one, and only one, of its operands is TRUE. XOR can be written using the other operators as: (A XOR B) = (A OR B) AND NOT(A AND B). XOR can for example be useful if you want to do a opinion poll, e.g., the following expression {Married XOR CollegeStudent) will evaluate to True if the person is either married **or** a College student, but **not** both.

The XOR operation is sometimes used as a simple mixing function in cryptography. XOR can also be useful if you want to invert some or all bits. Then put ones where you want to invert the bits and zeroes everywhere else. Binary values XOR:ed by themselves are always zero. This feature is in some machines used as an optimization when it comes to storing a zero in a register. It can be a faster operation to XOR a value with it self than it is to store a zero in the register.

8) ***What was the connection between the binary numbers and the ASCII-hello?*** (3 students)
Each ASCII symbol can be represented by its hexadecimal value. We need 8 bits to represent 2 hex numbers which is needed to represent one ASCII symbol.

For example what ASCII string is represented by the following bits:
01101000011001010110110001101100011011111 ?

i) group them into being groups of 8 digits: **01101000 01100101 01101100 01101100 01101111**

ii) we know that one hex number can be represented by 4 bits. Take a look at the first 8 bits: **01101000**. We split them into groups of 4: **0110 1000.** We now translate the binary 4-bit numbers into their hexadecimal counter parts resulting in: **0110 = 6** and **1000 = 8**. Resulting in the hexadecimal number **68.**

iii) we now take a look in the ASCII table and find that the hexadecimal number 68, represents 'h', thus we have the first letter of **01101000011001010110110001101100011011111**. If you continue the process you are going to find that the bits represent the string 'hello'.

9) ***Where can we find a table of conversions from letters to hex to ascii to binary? (and similar questions)*** (3 students)
For example on page 499 in Brookshear, or for example via this link.

10) ***Can we review the elsif statement in recitation?*** (1 student)
yes!

11) ***Can you translate the if-then-else into plain English?*** (and simlar questions) (3 students)
if GrossPay > 100.00 **then**
NetPay := GrossPay - Tax;
else
NetPay := GrossPay;
end if;

In plain English: If the Boolean expression 'GrossPay > 100.00' evaluates to **true**, then NetPay will be assigned the value of GrossPay minus Tax. Otherwise, if the Boolean expression 'GrossPay > 100.00' evaluates to **false**, then NetPay will instead be assigned the value of GrossPay.

12) ***Does Get_Line cut your string's array down to the entered size, or leave it the same?*** (3 students)
It does not change the size of the array, instead the positions in the array that do not get assigned new values get 'undefined' values.

13) ***Did not understand truth table*** (2 students)
Example will/can be given in tomorrows recitation.

14) ***Did not understand how we determined how lego car would behave*** (3 students)
Example will/can be given in tomorrows recitation.

15) ***What does Float mean?*** (1 student)
Take a look at yesterdays Mud-answer or try this alternative explaination taken from Feldman page 97: The standard data types in Ada represent familiar objects. For example, the data type **Float** is that subset of real numbers (in the mathematical sense) that can be represented on the computer. Every **Float** value in Ada is a real number; however, not all real numbers can be represented in Ada or in any programming language. Some real numbers are too large or too small or cannot be represented precisely owing to the finite size of a memory cell.More information can be found in Feldman Chapter 7, and Feldman section 3.5.

16) ***When using if-then statements there is a statement_before and at the end a statemet_after. Are they neccessary? if left out what happens?*** (1 student)
They are not neccessary. If they are left out, it only means that there was no Ada statements before the **if** statement, and no Ada statements after the **if** statement.

17) ***In testave example, why is there no clause 'if name "end" then exit or something to that effect? How does the program recognize 'end'?*** (1 student)

while Name(1..Namelen) /= "end" **loop**
... statements;
end loop;

translated into 'plain English' those statements means: As long as the characters on position 1 to 'Namelen' in the string with name 'Name' does not equal (the sign /= means 'not equal to') the string **"end"** do all the statements between '**loop**' and '**end loop**;'.

We will cover the details of the **while statement** in later lecture.

18) ***with: (Item => HowMany, Width => 3) Why specify width?*** (1 student)
It just makes the output on the screen more 'pretty'. Try to modify the code to execute the following statements and see the difference on the screen.

Put (Item => HowMany, Width => 3);

Put (Item => HowMany, Width => 20);

Put (Item => HowMany);

19) ***How are strings used? Are they used to assign values/cells for characters?*** (1 student)
Example of strings can/will be shown in tomorrows recitation.

String is a certain kind of array of characters.

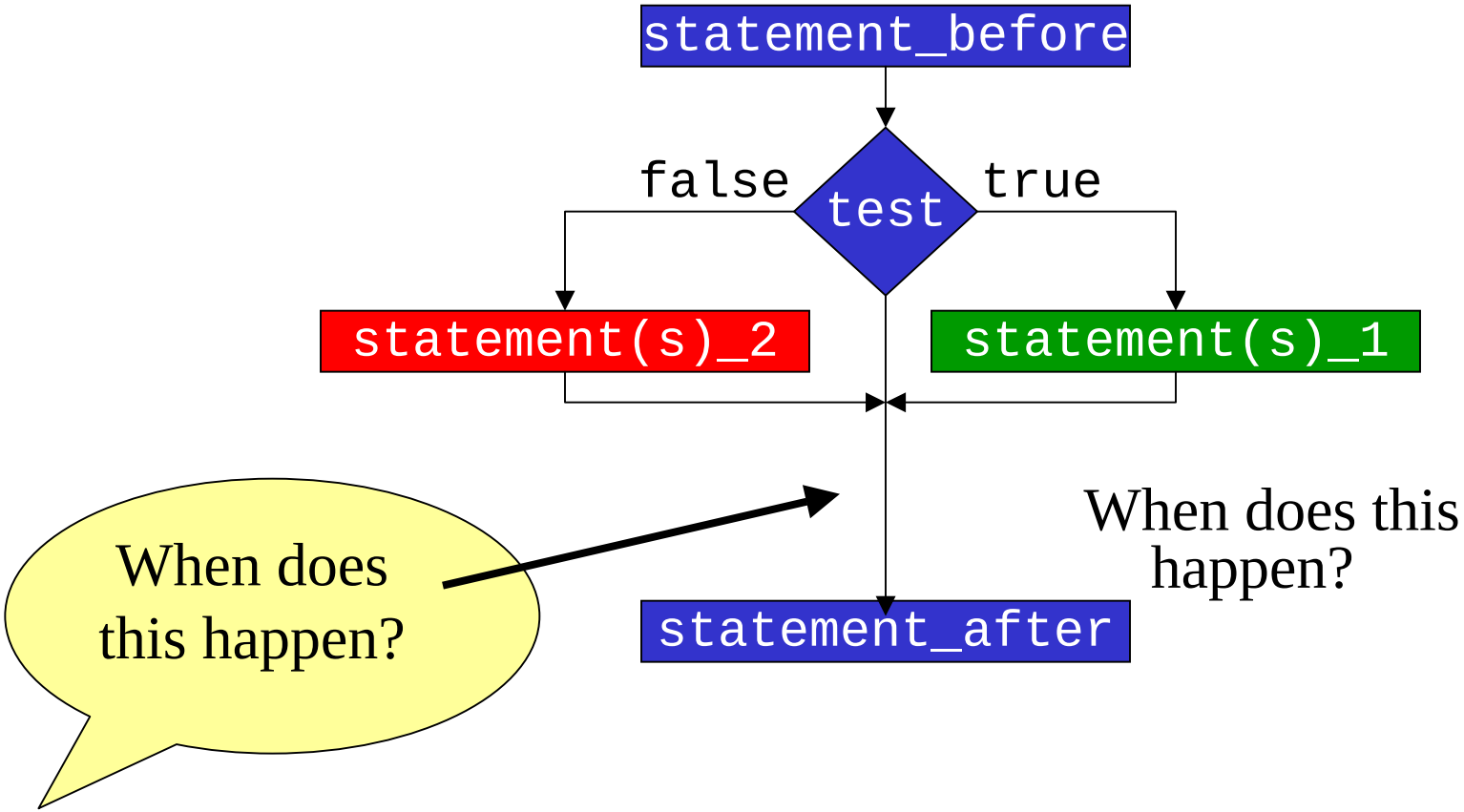
i) A string is an array of characters, with a subscript range that must be a subtype of **positive**.
ii) It is possible to assign or refer gto a part, or slice, of a string.
iii) string values can be compared and assigned like other Ada variables, but their lengths must match exactely.
iv) strings can be concatenated, or "pasted together", to form longer ones.

More information can be found in Feldman, Chapter 10.

20) ***What exactely was "response", how was it different from alphabet?*** (1 student)
'Response' was a copy of the array 'alphabet', meaning it was of the same size as 'alphabet' and had the same charcters in the same positions as was valid for the 'alphabet' array. The only difference between 'response' and 'alphabet' was the address in computer memory where they would reside.

21) ***When does the other case in an if-the-else structure occure?*** (1 student)
when the Boolean expresiion evaluates to **false**.

22) ***When does this happen?*** (1 student)



Directly after either 'statement(s)_1' or 'statement(s)_2' has finished executing, that is we are at the statement "**end if;**'

23) ***Can you alter a string so that it islonger than you originally declared it to be?*** (1 student)
Not a static string. There are dynamic strings which has this behaviour.

24) ***How is the Ada-code put into the robot?*** (1 student)
Ada code is converted into a language called NQC (Not Quite C) which is then converted into RCX code (the machine language understood by the robot). The final machine code is downloaded to the robot by means of an IR tower.

25) ***For if-then statements in e.g., "if test then ..." Does that mean if test is true? or do you have to specify a value (like if test=12 then ...)?***(1 student)
It means: If the Boolean expression 'test' evalutes to **true** then ...

26) ***Do you have all sample program savailable online?*** (1 student)
Not at the moment.

27) ***"No mud"*** (27 students)
Good :o)