Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

[Paulo Sousa / 311]

Team:

[Yen Huynh] - Id: [041 068 712] / [Mohammed Muhsin Shaik] - Id: [041 078 827]

Language Name [CoreX]

***This template is suggested (not mandatory) to answer A11 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **Language User Reference** |

**EXPLANATION**

*The purpose of this assignment is to invent a new computer language.*

* *This language can have the syntax and structure of your choosing.*
* *Option 1: Adapt the ‘Sofia language to be R compatible (see* <https://www.r-project.org/>*).*
* *Option 2: Define a* ***DSL*** *– Proper to solve specific problems (ex: science, economy, music, etc.)..*

*This is going to be a basic language. There's a lot of functionality that we'll be skipping over, while we implement the basics. You will need to tell me those basics, of course. In this document, I'm going to explain the steps of what to do with a bit of detail.*

* 1. **User Manual**

**Element 1: Name / Extension**

*[Name your language! We suggest you use one "word" related to your “Go-like” language or DSL]*

*[What is the filename extension of your language? For example, for C it is .c, and for Professor Paulo's “Sofia”* ***language*** *it is ".****sof****".]*

*Introduction:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Language:*** | ***Extension:*** | ***Advantages*** | ***Logo*** | ***How did I sell it?*** |
| *CoreX* | *“.cx”* | *Provide frameworks in the library.*  *Easy to install.*  *Efficiency and daily use purpose (data analysis)* | ***A logo with a blue and pink x  Description automatically generated*** | *Aim to the user's demands:*  *Efficiency?*  *Understandable?*  *Less complex?* |

1. *What is your language patterned after, or what is it similar to?*

***Answer:*** *It will be similar to the R programming language.*

1. *What languages are inspiring your choice? It's okay if you're following R closely.]*

**Answer:**  The main inspiration of coreX is R language because of its efficiency like library,

functioning, etc.

**Element 2 – Comments**

*[Comments: I want to make comments in your language. How do I write them?]*

**Answer:** We decided to make it simple by using the pound sign ‘#’ for commenting.

|  |  |
| --- | --- |
| **Single line** | **Multiple lines** |
| Only ‘#’ needed | Select the multiple lines on which you want to comment using the cursor and then use the key combination “control + shift + C” to comment or uncomment the selected lines. |

**Element 3 – Keywords**

*[Keywords: List the sequence of reserved / key words from your language]*

|  |  |
| --- | --- |
| **Keyword** | **Definition** |
| if/ else | Check if the value passes the requirements. |
| while/ repeat/ for | Use to iterate through the loop until it passes the requirements. |
| function | Provide it efficiently to reduce the space capacities of a computer. |
| in | Use for the loop |
| next | Use to skip the current iteration without loop termination. |
| break | Use to exit or terminate the loop. |
| TRUE/ FALSE | Boolean statement for returning true, false |
| NULL | Undefine value |
| Inf/ NaN | Inf:  + is.finite and is.infinite  NaN:  + Not a number |
| NA/ NA\_integer/ NA\_real/ NA\_complex/ NA\_character | Not available or represents a missing value |
| …, ..1, etc | Used to indicate a variable number of arguments in functions |

**Element 4 – Datatypes**

*[Datatypes: Define integers, real numbers (float points) and strings. Determine their ranges]*

*[Remember to define the number of bytes – and, if possible, range]*

|  |  |  |  |
| --- | --- | --- | --- |
| **Datatypes** | **Definition** | **Number of bytes** | **Ranges** |
| Integers | Whole numbers without decimal points. | 4 bytes (32 bits) | −231−231 to 231−1231−1 (approximately -2.147 billion to +2.147 billion |
| Real Numbers | Numbers with fractional parts are represented by using floating-point arithmetic. | 8 bytes (64 bits) | ±2.2250738585072014×10−308±2.2250738585072014×10−308 to ±1.7976931348623157×10308±1.7976931348623157×10308. |
| Strings | Sequences of character for textual data. | Depending on the length of the text | Limited by available memory, no fixed upper character limit |

**Element 5 – Variables**

*[Variables: How would a programmer define variables that can hold integer numbers (numbers with no decimal point), floating point numbers (numbers with a decimal point) or text (ie: strings in Java). This is element 1. Consider if you want to flag the variables in a special way, like SOFIA or BASIC, or not, like C or Java.]*

**Answer:** There are 3 ways to do it:

|  |  |  |
| --- | --- | --- |
| Equal to operator | Leftward Operator | Rightward Operator |
| Moha = “AClive” | Yen <- “AClive” | “AClive” -> Muhsin |

**Element 6 – Methods / Functions**

*[Variables: How would a programmer define methods]*

**Answer:** name <- function(parameter) {}

**Element 7 - Commands**

* ***Attribution / assignment****: How does your language let a programmer assign a value to a variable? (Will you allow casting? If so, how will it work?) How will your language handle math, and will it allow strings to be concatenated (merged)?*
* ***Selection****: How does your language do if-style logic? (Optional: Do you want to do some kind of switch/case as well?). You will need to explain how "conditionals" work in your language. How do you write Boolean operations, such as "or", "and", "not", and other conditions, such as less than, greater than, etc?*
* ***Interaction****: How will your code handle looping? (You can do one or more of a for-style loop, a while/do loop, etc.)*
* ***Input****: How does your program get input from the keyboard? (Strings are easiest.)*
* ***Output****: What would a programmer type to put output on the screen? What sort of variables or data will your code take?*
* ***Functions****: [Function definition: parameters and returning types]*
  + *What will be the syntax for making a function or subroutine?*
  + *How will it take parameters?*
  + *How will it return results?*
* **Most common statements:**

|  |  |  |
| --- | --- | --- |
| **Variables / Conditions / Operator** | **Iteration** | **Input / Output/ Functions** |
| **Define variable:**  x <- “Yen” / x = “Yen” / “Yen” -> x  **Casting? Yes, we have it.**  **Syntax:**  cast(data, formula, fun.aggregate)  **Parameters:**  **data:** represents dataset.  **formula:** represents the form in which data has to be reshaped.  **fun.aggregate:** represents aggregate function  **How does it work?** It is used to reshape the molten data which takes aggregate function and formula to aggregate the data accordingly.  **/------------------------------------------/**  **Conditions:**  if/ else(cond) {}  **/------------------------------------------/**  **Operator:**  **Comperator:**   |  |  |  | | --- | --- | --- | | a > b  a >= b | a == b  a != b | a < b  a <= b |   **Logical:**   |  |  |  | | --- | --- | --- | | && | || | ! |   **Miscellaneous:**   |  |  | | --- | --- | | **Operator** | **Description** | | : | Creates sequence. | | %in% | Find the element that belongs to a vector. | | %\*% | Matrix Multiplication | | for (var in seq) expr  while (cond) expr  repeat expr | **Input:**  readline()  **Output:**  print()  **/------------------------------------/**  **Functions:**  x <- function(param) {  return value (int, char)  } |

* **Explanation:**
* **cond:** A length-one logical vector that is not NA
* **var:** A syntactical name for a variable
* **seq:** An expression evaluating to vector, or a character vector. This can be a long vector.
* **expr:** An expression in a formal sense. This is either a simple expression or a so-called compound expression.

**Element 7 – Proper elements**

*[Include specific features / elements to be included in your language]*

* *What you could include / modify? Think about new datatypes / structures / commands, etc.*
* *Note: Do not share this info (it is supposed to be your proper elements in the language.*

**Answer:**

**Datatypes:** binary. Since we don’t have the computer-readable language in R studio, We decided to program a new datatype which is binary.

**How to define?**

x <- bi(01110000);

bi(): a function which converts binary to integer.

**Purpose:** It is designed to connect R and Assembly Language due to the complexity of Assembly.

|  |  |
| --- | --- |
| **Part**  **2** | **Language Comparison** |

**Comparing with C language**

**Differences**

|  |  |
| --- | --- |
| **R** | **C** |
| + Focus on data analytics.  + Easy to run.  + Not case sensitive.  + No need to define library.  + Build-in help terminal by using “?name”. | + Focus on software development.  + Complexity when compiles.  + Case sensitive.  + Define library.  + Not applicable. |

**Advantages / Disadvantages (in comparison with C)**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| + Easy to run.  + Not case sensitive.  + No need to define library. | -R language is a bit slower than C language since C is a compiled language.  -R is a higher-level statistical language so it is not as closely controllable on the hardware level as C language.  -R does not support multithreading and parallel programming as much as possible with C language |

**Comparing with another language[[1]](#footnote-2)**

**Language Name: Java**

**Differences**

|  |  |
| --- | --- |
| **R** | **Java** |
| Interpreted language, slower for general tasks but optimized for statistical computations | Compiled language with JIT, faster for general tasks |
| Easier for statisticians and data scientists, intuitive for data analysis | The steeper learning curve, especially for statistical tasks |
| Specialized in data analysis, statistical computing, and visualization | General-purpose, suitable for web, mobile, and enterprise applications |
| A rich ecosystem of packages for statistical analysis and data visualization | Extensive libraries for diverse applications |

**Advantages / Disadvantages (in comparison with this second language)**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Not complexity | Cannot program software |
| Focus on data analytics | Some functions are required for installing the package |
| No need to import packages | Cannot convert from binary to integer |
| Not case sensitive | Cannot do multiple-line comment |

|  |  |
| --- | --- |
| **Part**  **3** | **Architectural Questions** |

**Advantages**

*[What's the goal of your language? Are you trying to make something simple, fun, complicated? My personal language, Chambly, is based around being useful to scientists. (You can just make something up here, honestly. Think about it a little bit, have a little fun.)]*

**Answer:**

Our goal is to make an understandable language for programmers. A language that is not case sensitive, efficient, less complicated, and provides its curiosity to programmers.

**Strategy: C Implementation**

*[How your language can be implemented in C – ex: datatypes]*

* *In plain English, or maybe even some high-level pseudocode, how are you going to parse your language? You will be writing a compiler for your language, so these are some things you need to think about.*

***Note 1: C Datatypes***

*Remember that you are implementing your language in ANSI C. For this reason, you cannot create arbitrarily your language (from scratch). You need to use what is already provided by C Compiler. For this reason, think about using and defining the language obeying the datatypes.*

*[Your ideas about how to identify elements from language]*

* *Consider your "write to the console" command as an example. How will your compiler detect it?* *How will it sort out what to write to the console? What if there's some literal text (ie: "this is going to get printed") instead of variables?*

*[Your ideas about how to identify scope (ex: blocks between conditionals or functions)]*

* *How do you mark a block of code? If I use your loop logic, how do I control what portion of code gets looped through? In C, you might use { and }. In Python, the indentation is what matters. How does it work in your language?*

**Answer:**

**Pseudocode:**

Given the array T = [1, 2, 3, 4, 5, 6], the following pseudocode

n <- length(T)

for i <- 2 to n step 2 do

T[i] <- T[i-1]

end for

return(T)

results in:

T = [1, 1, 3, 3, 5, 5]

**My goal is to transform the pseudocode to R code:**

T = c(1,2,3,4,5,6)

n=length(T)

i=2

for (i in T ,by=2) {

T[i] = T[i-1]

end

}

return(T)

**How does the compiler detect when something is written in the software?**

**Answer:** There will be a run button to compile so we just need to click on it.

**How will it sort out what to write to the console?**

**Answer:** All of the commands have been stored inside the library, if the user makes a syntax error while programming it will display “Syntax error and do ?help for more”.

**What if there's some literal text (ie: "this is going to get printed") instead of variables?**

**Answer:** Our compiler does not support that command.

**How do you mark a block of code?**

**Answer:** Select the multiple lines on which you want to comment using the cursor and then use the key combination “control + shift + C” to comment or uncomment the selected lines.

**Loop:**

for (var in seq) expr

**Example:** for (i in 1:5) {print(i)}

**FINAL SUGGESTIONS**

*Here some ideas to think about your language....*

* *Don't make this assignment harder than it needs to be on yourself. Focus on making the syntax for your language that meets our requirements. Worry about extra features later.*
* *Don’t worry if your new language winds up having really difficult parts. You'll be allowed to change your language as you go along, as long as you make "patch notes" to explain those changes. We'll tell you about this later.*
* *There's a marking info in the Assignment Guide (CST8152\_Compilers\_242S--ASSAMG) that should steer you along for grades. Focus your efforts on where you'll get the best results.*
* *Finally, think about creating an “master-piece”: until now, you have used several languages. And if you have conditions to define yours, how it could be?*

**References**

“Comments in R.” *GeeksforGeeks*, GeeksforGeeks, 25 Nov. 2023, www.geeksforgeeks.org/comments-in-r/.

“R - Keywords.” *GeeksforGeeks*, GeeksforGeeks, 3 Apr. 2023, www.geeksforgeeks.org/r-keywords/.

“Melting and Casting in R Programming.” *GeeksforGeeks*, GeeksforGeeks, 12 June 2020, www.geeksforgeeks.org/melting-and-casting-in-r-programming/.

“R Operators.” *R Operators*, www.w3schools.com/r/r\_operators.asp. Accessed 16 May 2024.

Farm. “Pesudocode ----> R Code.” *Posit Community*, 12 Dec. 2022, forum.posit.co/t/pesudocode-r-code/154965.

ChatGPT for Element 4: Datatypes, PART 2

*[Include eventual references used here]*

|  |  |
| --- | --- |
|  | * ***NOTE****: Even if you use any AI tool (ex: ChatGPT), report here, including the references used.* |

Algonquin College

Summer, 2024

1. You can use any language (different from C). Ex: Java, Go, Python, etc. [↑](#footnote-ref-2)