Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Team:

[Aeham Alobidat] - Id: [040897798]

[Bradley Cliche] - Id: [041040049]

Language Name [Phoenix]

***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 ‘Julius’ language to be Julia compatible (see*[*https://julialang.org/*](https://julialang.org/)*).*
* *Option 2: Define a****DSL****– Proper to solve specific problems (ex: science, economy, music, etc.)..*
* *This is going to be a fairly 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 “Julia-like” language or DSL] 🡪 My new language name is Phoenix*

*[What is the filename extension of your language? For example, for C it is .c, and for Professor Paulo's “Julius”****language****it is ".****jul****".] My filename extension for this language going to be Nix*

*[What is your language patterned after, or what is it similar to? What languages are inspiring your choice? It's okay if you're following Julia closely.]*

*Following* Python

**Element 2 – Comments**

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

*I will use this Sign for the Comments ‘#/’*

*The Multi-line Comment*

**Element 3 – Keywords**

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

const

continue

else

elseif

end

for

function

global

if

return

struct

switch

try

type

using

while

**Element 4 – Variables and Datatypes**

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

* *How many bytes are you needing for your variables? This determines their ranges. (Chambly, for instance, has a special 64-byte integer. This is ridiculously huge for most purposes.)*

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

*floating numbers using the "Float64. This data type takes 8 bytes and that is the range -1.79769313486232e+308 to 1.79769313486232e+308*

*define strings using the "String" data type. The size of the string can be of any length and it generally uses 1 byte per character*

*we can define integers using the "Int64" data type. This data type takes 8 bytes of memory and the range of integers and that is the range -9223372036854775808 to 9223372036854775807*

**Element 5 – Variables and Datatypes**

*[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.]*

*variable that can hold an integer number (no decimal point) = x = 5*

*with annotation = x::Int64 = 55*

*variable that can hold a floating-point number (with a decimal point) = y = 3.14*

*with annotation = y::Float64 = 3.14*

*variable that can hold text (i.e: strings in Java) = str = "Hello, Julia"*

*with annotation = str::String = "Hello, Julia!"*

**Element 6 - Commands**

* ***Attribution****: 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)?*

In Phoenix, a variable is assigned a value using the assignment operator, which is typically the equals sign (=). For example

x = 5

Phoenix also allows casting, which refers to converting a value of one data type to another. This can be done by using casting functions or operators. For example

x = Int(5.0) # x will be 5

y = Float64(5) # y will be 5.0

z = string(5) # z will be "5"

Strings in Phoenix can also be concatenated, which means merging two or more strings into a single string. This is typically done using the string interpolation or the \* operator. Here's an example:

name = "John"

greeting = "Hello, " \* name \* "!"

println(greeting)

# Output: "Hello, John!"

* ***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?*

***Basically, like other language, I said that because most of the language are sharing the same strategy***

In Phoenix, like in many other programming languages, selection or conditional statements are used to perform different actions based on different conditions

if condition:

# code to be executed if the condition is true

For example, the following code checks if the variable x is greater than 5, and if it is, it prints "x is greater than 5"

x = 7

if x > 5:

print("x is greater than 5")

----------------------------------------------------------------------------------

else statement in combination

if condition:

# code to be executed if the condition is true

else:

# code to be executed if the condition is false

x = 3

if x > 5:

print("x is greater than 5")

else:

print("x is not greater than 5")

----------------------------------------------------------------------------------

if condition1:

# code to be executed if condition1 is true

elif condition2:

# code to be executed if condition1 is false and condition2 is true

else:

# code to be executed if condition1 and condition2 are false

You can also use these operators to chain multiple conditions together.

The "and"/"or"/not" operators

In Phoenix, there is no equivalent to a switch statement like in some other languages

x = 7

y = 4

z = 3

if (x > 5 and y < 10) or not z == 5:

print("x is greater than 5, y is less than 10 or z is not equal to 5")

* ***Interaction****: How will your code handle looping? (You can do one or more of a for-style loop, a while/do loop, etc.)*

Yes, I can do the loop styles

The for-style loop is used to iterate over a sequence of elements

for variable in sequence:

# code to be executed for each element in the sequence

end

*----------------------------------------------------------------------------------*

for number in numbers:

print(number)

end

*----------------------------------------------------------------------------------*

*The while-style loop is used to execute a block of code repeatedly as long as a certain condition is true.*

*while condition:*

*# code to be executed while the condition is true*

*end*

*----------------------------------------------------------------------------------*

*x = 0*

*do:*

*print(x)*

*x += 1*

*while x < 5*

*end*

* ***Input****: How does your program get input from the keyboard? (Strings are easiest.)*

In Phoenix, can get input from the keyboard by using the readline() function, which reads a line of text from the user and returns it as a string. The general syntax for the input() function is as follows:

name = readline()

println("Hello, $name!")

Additionally, Julia also provides readbytes(), readstring(), readline(), readlines() functions to read input from the keyboard.

And I could convert the input to other types like integers or floats by using parse()

***Output****: What would a programmer type to put output on the screen? What sort of variables or data will your code take?*

In Phoenix, can put output on the screen by using the println() and the print() function. The print() function takes one or more arguments and displays them on the screen.

----------------------------------------------------------------------------------

x::float65::= 5.1

println("The value of x is: ", x)

----------------------------------------------------------------------------------

x = 5

print("The value of x is: ")

print(x)

and for string

name = readline()

println("Hello, $name!")

* *The general syntax for the print() function is as follows:*
* ***Functions****: [Function definition: parameters and returning types]*
  + *What will be the syntax for making a function or subroutine?*

To generate a print function in my language will be as follows

function namefunction(ex1, ex2, ...)

# function code here

# return statement is optional

end

-----------------------------------------------------

function add\_numbers(a::Int, b::Int)

return a + b

end

* + *How will it return results?*

In my language, a function can return results using the return keyword followed by the value or expression that should be returned.

function add\_numbers(a::Int, b::Int)

return a + b

end

The function takes two integers as input and returns their sum by using the return keyword.

**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.*

**Float; Integer; Boolean; Strings; Hashes; Arrays; Symbols; Numbers**

New datatype

Improved Interoperability: it could be improved to make it easier to call other languages' libraries and to call Julia from other languages.

Additional data structures: additional data structures such as Trees or lists could be added to support more specific use cases.

* *Note: Do not share this info (it is supposed to be your proper elements in the language.*

|  |  |
| --- | --- |
| **Part**  **2** | **Examples** |

**Option 1: Julia-like**

**Hello World**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  # printing Hello World in PHX language  println("Hello World!")  print("Hello World!") |  |

**Sphere Volume Expression (or any other example)**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  function sphere\_volume(r::Float64)  return 4/3\*π\*r^3  end  r = 5.0  vol = sphere\_volume(r)  println("The volume of a sphere with radius $r is: $vol") |  |

*[TIP: See examples in the Lecture Notes –* ***Appendix 1****]*

**Option 2: DSL**

**[Your example here]**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here] |  |

|  |  |
| --- | --- |
| **Part**  **3** | **Architectural Aspects** |

**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.)]*

*My goal of this language to pass the course, not more then that. Because whatever I am going to do in this course I will forget about in the next few months.*

**Strategy: C Implementation**

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

* *In plain English, or maybe even some high-level pseudocodes, 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.*

The process of converting code written in Phoenix, into machine code that can be executed by a computer (also called "compilation") can be divided into several stages starting with :

The first step is to take your instructions and break them down into smaller, simpler parts. This is like taking a big puzzle and breaking it down into smaller pieces so you can work with them more easily. These smaller parts are called "tokens".

Once you have your tokens, the next step is to group them together to form bigger structures that make sense. This is like taking the puzzle pieces and grouping them together to form bigger shapes, like a car or a house. These bigger structures are called "syntax".

Finally, you need to make sure that the instructions make sense and that there are no errors. This is like making sure all the puzzle pieces fit together correctly and that there are no missing pieces

During this semester we going to learn about Lexical analysis, Syntax analysis, Semantic analysis, Code Generation, Optimization which each one of these stages is part of creating and identifying new language and processing

*[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?*

To detect the "write to the console" command in Julia, the compiler can look for specific keywords or function calls such as println() or print()

println("Hello World!")

If the argument is a literal text, such as "this is going to get printed" the compiler will simply output the text to the console. If the argument is a literal text, such as "this is going to get printed" the compiler will simply output the text to the console.

*[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?*

In my language. uses the begin and end keywords to indicate the beginning and end of a block. When a block of code starts with the keyword begin and ends with the keyword end, the compiler will understand that all the code in between these two keywords belongs to the same block.

the blocks of code are usually indicated by the indentation level, but the begin and end keywords can also be used to indicate the beginning and end of a block. This way, the compiler can understand which code belongs to which block and execute it accordingly.

For example,

begin

for i in 1:10

println(i)

end

In summary, in my language, the blocks of code are usually indicated by the indentation level, but the begin and end keywords can also be used to indicate the beginning and end of a block. This way, the compiler can understand which code belongs to which block and execute it accordingly.

**Basic ideas about C implementation**

*[Which structures or datatypes you imagine to use in your language implementation]*

* *What do you think is going to be really hard about this? What would be, in your opinion, the hardest part of parsing your own new language? You don't have to write an essay, a paragraph or two will be fine.*

important structure is the symbol table, which is used to keep track of the variables and their types. It would be used by the compiler to ensure that the variables are used correctly and to generate the appropriate machine code. In terms of data types, I would implement the basic data types such as integers, floating-point numbers, and strings, as well as more complex data types such as arrays and structures.

I think one of the hardest parts of parsing my own new language would be handling the different syntax rules and conventions of the language, such as the use of indentation and keywords. This would require a lot of attention to detail and a thorough understanding of the language's grammar and syntax to ensure that the parser can correctly interpret the source code.

***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.*

**Problems when using C implementation**

*[Your vision about main problems / difficulties when implementing a new language (ex: memory allocation, range of datatypes]*

As my experience from this course that I had it last semester. I think the following points will be a bit difficulty to do.

Syntax and Grammar

Error handling

Range of datatypes

**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 key at the end of****CST8152\_Compilers\_W23-A11-Specification****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**

*[Include eventual references used here]*

*ChatGPT*

Algonquin College

Fall, 2022