Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Name:

[Thomas Jones] - Id: [040953052]

Language Name [Viper]

***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 ‘BOA’ language to be Python compatible.*
* *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: Viper

Extension: .vip

The Viper language is a Python-like language used to solve simple and complex arithmetic. Easy to use functions and keywords help the user to convert everyday problems into code that is then solved by the compiler. The main idea behind the viper language is an easily comprehensible language that not only allows for easier arithmetic but still allows the user to use basic qualities of languages such as python including user input, loops, and many more functions.

**Element 2 – Comments**

Comments in Viper are very simple. For one-line comments a simple // will be used while any block commenting can be done using #/ ending in /#. An example of each can be shown below.

//This is a single line comment

#/This is a multiple

line comment /#

**Element 3 – Keywords**

A list of Major keywords can be seen below in Table 1.1

|  |  |
| --- | --- |
| TRUE | Boolean value, used for comparisons or = 1 |
| FALSE | Boolean value, used for comparisons or = 0 |
| IF | Used as conditionals statement. |
| ELSE | Used if options previous don’t apply to the conditional statement |
| WHILE | To create loops |
| TRY | Used to make try catch statements. Exception handling |
| RETURN | To exit a function and return a value |
| EXCEPT | To be used with exemptions to return a reason if exemptions occur |
| NULL | To assign a None or Null value |
| GLOBAL | To create globally accessible variables |
| IMPORT | To Import other libraries or modules |
| PI | To assign the value of Pi = 3.141 |
| LAMBDA | Creating small arithmetic functions within variables |
| RADICAL | Keyword used to find the radical of a variable: RADICAL (x) |
| FACTORIAL | Replaces the use of x!: FACTORIAL (x) |
| SUM | The summation of terms |
| MEAN | Calculates the mean value of terms |
| ABS | Converts a variable to its absolute value |

**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]*

The main data types that will be used in viper will include; String, int, float, double. Their byte sizes and ranges can be found in table 1.2 below

|  |  |  |
| --- | --- | --- |
| Name | Size | Range |
| String | 1 byte per character | User Determined |
| Int | 4 Bytes | -2,147,483,648 - 2,147,483,647 |
| Float | 4 Bytes | 3.4E +/- 38 |
| Double | 8 Bytes | 1.7E +/- 308 |

**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.]*

Data types will not require a specified declaration but can be cast to one before or after they have been initialized if the current variable value can be applied a specific data type. An example of declaration would be:

x = 6.2 //where x will be of type float and will equal 6.2

and an example of casting would be:

x = int(6.2) //where x will be of type int and will equal 6.

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

As stated above Viper will allow users to assign variables without specifying a datatype or by casting said variable using a specific datatype to ensure the variable is read properly. In terms of concatenation, a user can combine strings simply by using a + operator in a variable declaration or within print statements themselves. All basic (+, -, =, /, \*) and complex (^, Square Root, Radical) math operators will be implemented in such a way that it is easy for users to use them. For example, the square root of a variable can be calculated simply using SQRT(x).

* Selection:

If statements will work identically to Python. Multiple if statements can be made in a block and work based off logic operators including (a==b, a!=b, a</>b, etc..). If a user wants to use two conditional logical operators in one if statement, they can use Boolean operators like OR, AND, and NOT. An example of a Viper if statement can be seen below;

If b > a AND b!=a{

Print(b)

}

* Interaction:

The main loop in Viper will be while loops. While a specific occurrence or logical operation is true the compiler will continue to loop through the code stated in the while loop. The user can either use a normal while loop or use do while loops to ensure the code is run at least once. A user can also use break and continue statements within a loop to alter or completely stop its iteration.

* Input:

User input is taken very similarly to Python where the program stops once it hits an input statement in the code until the user hits enter after entering a value or string. That input will be saved in a variable as a string but can declared as a different data type within the input statement. An example of an input of a float can be seen below:

x = input\_float(“Input a float value: ”)

-Output:

The main way for Viper to display information on screen is through print statements. These statements will take all variables when printing them individually but when concatenating variables, the programmer must cast all variables to be of the same format. If the programmer would like to print a string followed by an integer, they must cast the integer as a string. Exemptions will be thrown if two data types are concatenated together.

-Functions:

Functions within Viper will be defined using the FUN syntax. An example of this would be:

Fun my\_function(){}

Where the block of text within the last set of parenthesis will be included in the function. Any parameters the programmer would like to pass into a function must be declared within the parenthesis after the function name. If a value would like to be returned from the calling of a function a return statement must be added within the function. This return statement can return any variable or assigned value from the function.

**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.*

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

**Option 1: Python-like**

**Hello World**

|  |  |  |
| --- | --- | --- |
|  | // VIPER Hello  Print(“Hello World”) |  |

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

|  |  |  |
| --- | --- | --- |
|  | // VIPER Example of sphere volume  FUN sphere\_Volume() {  r = input\_float(“Input a radius value: “)  Vol = 4.0 / 3.0 \* PI \* (r^3)  print(Vol)  }    //Alternatively you can use the built in Sphere Volume function  //SPHERE\_VOLUME(r) |  |

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

|  |  |
| --- | --- |
| **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.)]*

The main goal of Viper is to be a supplement to calculators for mathematicians and engineers allowing them to solve multiple equations while maintaining proper values through said calculations. One good example of the application of Viper could be in chemical engineering when you are working with multiple piping and volume equations where certainty and exact values are key in determining accurate and safe fluid and gas transport systems.

**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.*

*[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?*

My language will be parsed step by step in c code basically translating what is stated in Viper and breaking it down into c code. In terms of writing to console where Viper uses print statements, I will need to translate those print statements into printf statements. In parsing the code through c I can change the original Viper code and turn it into c code that can then be broken down and executed. Blocks of code will be defined using {} to allow for ease of translated and may be changed to use indentation in future changes to Viper.

*Parse code*

*if x = print(…)*

*x = printf(…)*

**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.*

***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]*

The hardest part of this will be in the translation of my code into c. With minimal experience in c language figuring out proper syntax and converting my written code to that will require a lot of patience. With the idea of Viper being a mathematical based language finding ways to convert specific keywords and equations into acceptable c code will be challenging. On top of this making sure that I am allocating enough data to each individual variable and data type depending on the required storage will be hard as well. While integers and floats may be a bit easier to implement, full strings and doubles will be much more challenging.

**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\_F22-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**

Corob-Msft, “Data Type Ranges,” *Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/en-us/cpp/cpp/data-type-ranges?view=msvc-170. [Accessed: 18-Sep-2022].

“Python Casting,” *w3schools*. [Online]. Available: https://www.w3schools.com/python/python\_casting.asp. [Accessed: 18-Sep-2022].

“Python Keywords,” *w3schools*. [Online]. Available: https://www.w3schools.com/python/python\_ref\_keywords.asp. [Accessed: 18-Sep-2022].

“Python User Input,” *w3schools*. [Online]. Available: https://www.w3schools.com/python/python\_user\_input.asp. [Accessed: 18-Sep-2022].

Algonquin College

Fall, 2022