Language Specification

Language Name [Bust]

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

* 1. **User Manual**

**Element 1: Name / Extension**

I would prefer to name my language as “Bust”. since I am a fan of Rust, I would like to name my language close to Rust. So, I changed the first character of Rust (R) to the first character of my name (B).

The extension is also very close to Rust, it would be “.bs”

**Element 2 – Comments**

I would like to use “//” as the sign of the start point of the comment

**Element 3 – Keywords**

***Answer:***

1. ***selective structure:***

if, else

1. ***loop structure:***

while

1. ***function definition and its functionalities:***

fn

return

1. ***Boolean Keyword:***

true, false

**Element 4 – Datatypes**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Data Type/Features*** | **Keyword** | **Range** | **Size** |
| **Integer** | i64 | -2,147,483,647 - 2,147,483,647 | 8 Bytes |
| **Character** | char | -128 - 127 | 1 Byte |
| **Float Point** | f64 | -2^128 ~ +2^128 | 8 Bytes |
| **String** | String | / | / |
| **Boolean** | bool | 1 | 1 Bytes |
| **Void** | NULL | / | 4 Bytes |

**Element 5 – Variables**

**There is no special way to flag a variable**

Variable can be defined by its data type keyword and its name with let.

The name of the variable can be only combined with digit and letter(uppercase or lowercase). The special character (space, sharp) and any keyword (int, if) are not allowed. Also, the name of the variable must begin with a letter.

For example:

let Int var = 1;

**Element 6 - Commands**

The value assignment would be performed by “=”.

The casting is allowed in my language. It would be performed by value casting itself with new data type.

For example: let float x = 10.1 x = int(x)

The language will perform the following basic arithmetic operations by:

plus (+), miner (-), multiplication (\*), division (/), module (%), power (^)

The concatenation of string is allowed, it can be perfozrmed by plus (+)

The conditional will be simply performed by if else keyword:

If expression

{

} else {}

The Boolean Operations would be performed by:

And (&&), or (||), not (!), less than (<), greater than (>), equal (==), not equal (! =)

The looping will be simply performed by while keyword:

while expression

{

The standard input will be simply performed by read\_line function:

read\_line(variable)

The standard output will be simply performed by print function:

print!(variable/literal value)

The syntax to make a function would be “fn”

It will take parameters after the function name. The in-take variable should be its datatype and data name.

It will optionally return the results. If there is a result need to be returned, then leave the datatype of return value after in-take variable.

For example:

fn myFunction (int64 var1, int64 var2) int64

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

**Option 1: Rust-like**

**Hello World**

|  |  |  |
| --- | --- | --- |
|  | fn main(){  print!(“Hello World!”);  } |  |

**Sphere Volume Expression**

|  |  |  |
| --- | --- | --- |
|  | fn main() {  let int pi = 3.141592653589793;  let float radius;  print!(“Radius of Sphere”)  read\_line(radius);  let float volume = (4.0/3.0) \* (pi \* radius ^ 3)  print!(volume)  } |  |