Arithmetic and Variables

Young reader, be weary of arithmetic in a language that refuses to use symbolic notation. Chances are you might have come across the unfortunate case of *Computato!* "Symbolic notation?" – you ask, unknown to the term. Why, young reader, look at a mathematical expression such as this:

$$a = 1 + 1$$

That's not quite so difficult for you, I assume. But this is not what you will find in *Computato*. You may think your so smart using plus signs and equals signs and that star thing for multiplication, and you never thought to consider how lucky you are to live in a world where these things exist. Well let this be a wakeup call for you to check your modern-arithmetic-privilege and put yourself in the shoes of a Roman lad who's just trying to do his math homework. Do you know what he saw?

A EST I PLVS I

There were no symbols! No notation of the like! To the people of Rome and Ancient Greece, mathematical equations were just sentences. Because you've chosen to learn this programming language, you have waived your right to a normal programming experience. Equations and calculation in *Computato* are a mix between modern and old (only because I can't yet figure out how to use Roman Numerals, but that will come eventually) and looks like this:

```
a est 1 plus 1
```

It's not so bad, it's fun! Hell is where you no longer can use + or – or * or / for math, but instead you use these:

```
+  plus
-  minus
*  mula
/  vide
%  mod
```

In case you don't know (I didn't when I started) *mod* is the modulus operator, meaning that is give you the remainder from a division operation (100 % 150 = 50). Also it doesn't follow order of operations, which canonically I'll pretend is because Romans didn't have order of operations but it's more so that I don't know how to make a parser for it. It will however obey parentheses, which if you remember are not parentheses but these things: <>. Trying to use symbolic notation in a *Computato* program will lead to a curse placed upon your descendants for a hundred generations (won't compile) so make sure that you use the words.

Some programming languages are nice enough to include operators to increment a variable without having to repeat the variable in the equation, i.e. x = x + 1 -> x += 1. Luckily, *Computato* has these operators too!

```
 plus  adde
minus  strahe
 mula  accu
 vide  divi
  mod  ?
```

There's no token for modulus because I forgot and just realized I didn't have one while writing this guide, so it'll be there like whenever I get around to it (never).

Now that you can do math, let's get on to variables! There are 3 types of native variables in *Computato* (since it is object oriented, you can created your own variable objects using the same syntax):

```
           string (Words)  verba
integer/double (Numbers)  numerus
    Boolean (True/False)  status
```

You will notice that there is no differentiation between the integer and double types like most languages. The numerus type can store numbers of any decimal place. If you're wondering whether or not this slows down the program do to memory issues, rest assured that everything is stored as strings in C++ anyways so you're not going to be creating some superfast shit here regardless.

The declaration of variables is fairly straightforward:

```
[variable type] [token] est [value]

numerus n est 2
```

So that's that. Changing the value of variables is equally easy, just call the variable and use 'est' again:

```
[token] est [value]

n est 3
```

For Booleans the value of true is $VERA while the value of false is $FALSUS the dollar sign before the definition is very important and leaving it off will cause a runtime error. Strings can be combined using the plus token:

```
verba v est |Salve| plus | Munde!|
```

There are also the comparator functions, which can be used with strings and Booleans (except the less than/greater than comparators, a string cannot be less than or greater than another, same for a Boolean):

```
  == (equals to)  sit
   < (less than)  minor
 > (greater than)  maior
!= (not equal to)  non
```

There is also the '!' modifier that negates a Boolean value, and it's the same in *Computato* because I couldn't think of a Latin equivalent that was one character. I'm keeping you on an honor system not to use '!' because if you did the hypothetical Roman from the first chapter would have difficulty understanding your code.

Arrays are not actually that bad. I'm personally pretty proud of how I implemented arrays, or as they are called in *Computato,* gammas. The declaration is thus:

```
gamma<[size]> [token] [variable type]
gamma<100> listus_numerus numerus
```

Gammas can be easily extended into multiple dimensions using the & operator (which is used as the comma elsewhere) like so:

```
gamma<10 & 10> ordinati numerus
```

Arrays as variables can be called and set exactly like any other type:

```
listus_numerus<69> est 5318008
LIBER~clamaLinea<ordinati<6 & 9>>
```

Give yourself a pat on the back, you are now a master when it comes to variable manipulation in a programming language nobody uses. If that doesn't depress you, try it out! Make a thing that does something, I don't know. Fibonacci seems to be something everyone does. Or make it display 1000 digits of pi. Where you expecting some programming tutorial here? I can't just give some tutorial every time, there's nothing here worth an entire program. You want a program? Make one, be the change you want to see. Also check out the examples from the Github, might help you if you are still confused, which is understandable.