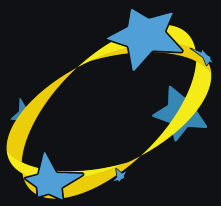
GLaDOS

Epitech Project GLaDOS for Promo 2026

Made by :

- ➔ Aurélien LECLERCQ
- ➔ Edgar DION
- ➔ Artyom TILLON
- ➔ Marius LACOMBLEZ
- ➔ Yanis BERKANE



What is the goal of the project ?

This project is part of the 3rd year epitech bachelor's curriculum.

The project aims to create a coding language and a virtual machine to execute it's compiled programs

---

Build guide

On Linux :

Requirements :

- ➔ **Git**
- ➔ **Make**

Build :

```
git clone git@github.com:EpitechPromo2026/B-FUN-500-PAR-5-2-glados-yanis.berkane.git
cd B-FUN-500-PAR-5-2-glados-yanis.berkane
make
```

That’s pretty much it, all you have to do now is run the program using `./glados`

Language Syntax

Syntax introduction

In this part of the documentation, we will go through the general syntax of the GLaDOS language.

Here's the full BNF of the language :

```
<syntax> ::= "" | <statement> <syntax>

<statement> ::= <item> ";" | <if-tree> | <while-statement> | <for-statement> | <function-definition>

<item> ::= <ternary> | <function-call> | <subscription> | <const> | <symbol> | "(" <item> ")"
<if-tree> ::= <if-statement> <else-statement>
<function-definition> ::= "<symbol>" "(" <comma-separated-symbol-list> ")" "=>" "(" <item> ")" | "<symbol>" "("
<comma-separated-symbol-list> ")" "=>" "{" <item> "}"

<code-block> = "(" <syntax> ")"
<condition> = <condition>

<if-statement> ::= "if" <condition> "then:" <code-block>
<else-statement> ::= "" | "else:" <code-block>

<function-statement> ::= "" | <syntax> | "return" <item> ";"
<comma-separated-item-list> ::= "" | <item> <comma-separated-item-list>

<ternary> ::= <item> "?" <item> ":" <item>

<int> ::= "" | <digit> <int>
<float> ::= <int> "." <int>
<bool> ::= "True" | "False"
<string> ::= " <ascii-char> "

<symbol> ::= <head-symbol-char>
<head-symbol-char> ::= <letter>
<tail-symbol-char> ::= "" | (<head-symbol-char> | "-" | "_" | <digit>)
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<letter> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q"
| "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
"j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
<complex-char-no-apostrophes> ::= "|" | " " | "!" | "#" | "$" | "%" | "&" | "(" | ")" | "*" | "+" | "," | "-" |
"." | "/" | ":" | ";" | ">" | "=" | "<" | "?" | "@" | "[" | "\" | "]" | "^" | "_" | "`" | "{" | "}" | "~"
<ascii-char> = <complex-char-no-apostrophes> | <letter> | <digit> | "\n" | "\t"

<infix-operator> ::= "+" | "-" | "*" | "/" | "%" | "&&" | "||" | "==" | "=" | "<=" | "<" | ">=" | ">" | "!="
<unary-operator> ::= "+" | "-" | "!"
<function-call> ::= <symbol> "(" <comma-separated-item-list> ")"
```

Arithmetic Operators

The GLaDOS language supports the following arithmetical operators:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Logical Operators

The GLaDOS language supports the following logical operators:

Operator	Description
&&	AND
||	OR
!=	NOT

Data Types

In the GLaDOS language, there are several different data types. These include :

➔ **int** - An integer value

```
<int> ::= 'Int64 MinValue' .. 'Int64 MaxValue'
```

➔ **char** - A single character

```
<char> ::= 'An ascii character'
```

➔ **string** - A string of characters

```
<string> ::= 'An array of ascii characters between double quotes'
```

➔ **bool** - A boolean value

```
<string> ::= 'An array of ascii characters between double quotes'
```

To declare a variable, use the following syntax :

```
<variable_definition> ::= “let” <identifier> “=” <expression> “;” | “var”
<identifier> “=” <expression> “;”
```

Variables are not typed, and can be assigned any value.

The value of a variable can be changed at any time.

Conditional Statements

In the GLaDOS language, you can express a conditional statement in the following way :

```
<if_statement> ::= “if” “(” <expression> “)” “then:” “(” <statement> “)”
```

The **if** keyword is followed by an expression, which is then followed by a block of statements. The block of statements is executed if the expression evaluates to **True**. If the expression evaluates to **False**, the block of statements is skipped.

You can also express an **if** statement with an **else** clause:

```
<if_statement> ::= "if" "(" <expression> ")" "then:" "(" <statement> ")" “else:” “(“ <statement> “)”
```

An expression must evaluate to a boolean value (True or False), if not, it will systematically evaluate to False.

Example :

```
let a = 1;
let b = 2;

if (a == b) then:
(
    print("a is equal to b");
)
print("a isn’t equal to b");
```

Output :

```
a isn't equal to b
```

Function definitions and expression

In the GLaDOS language, you can express a conditional statement in the following way :

```
helloWorld() => (
    print(“Hello, World!”);
);
```

Here, we only need the name of the function to declare it. It is then followed by a list of parameters between parentheses (none in this case), and finally by a block of statements.

```
fact (a) => (
    if (a == 1) then:
        return (1);
    else:
        return (fact(a - 1) * a);
);
```

This function is here to highlight the use of an argument in a function. The **fact** function takes an argument **a** and returns **1** if **a** is equal to 1, and returns the factorial of the given parameter otherwise.

Builtin functions

IO :

➔ **print -** Print data to the standard output with a newline.

```
print(data);
```

Bonus !

You can download our code's extension on Visual Studio Marketplace :

To do so, all you have to do is click here