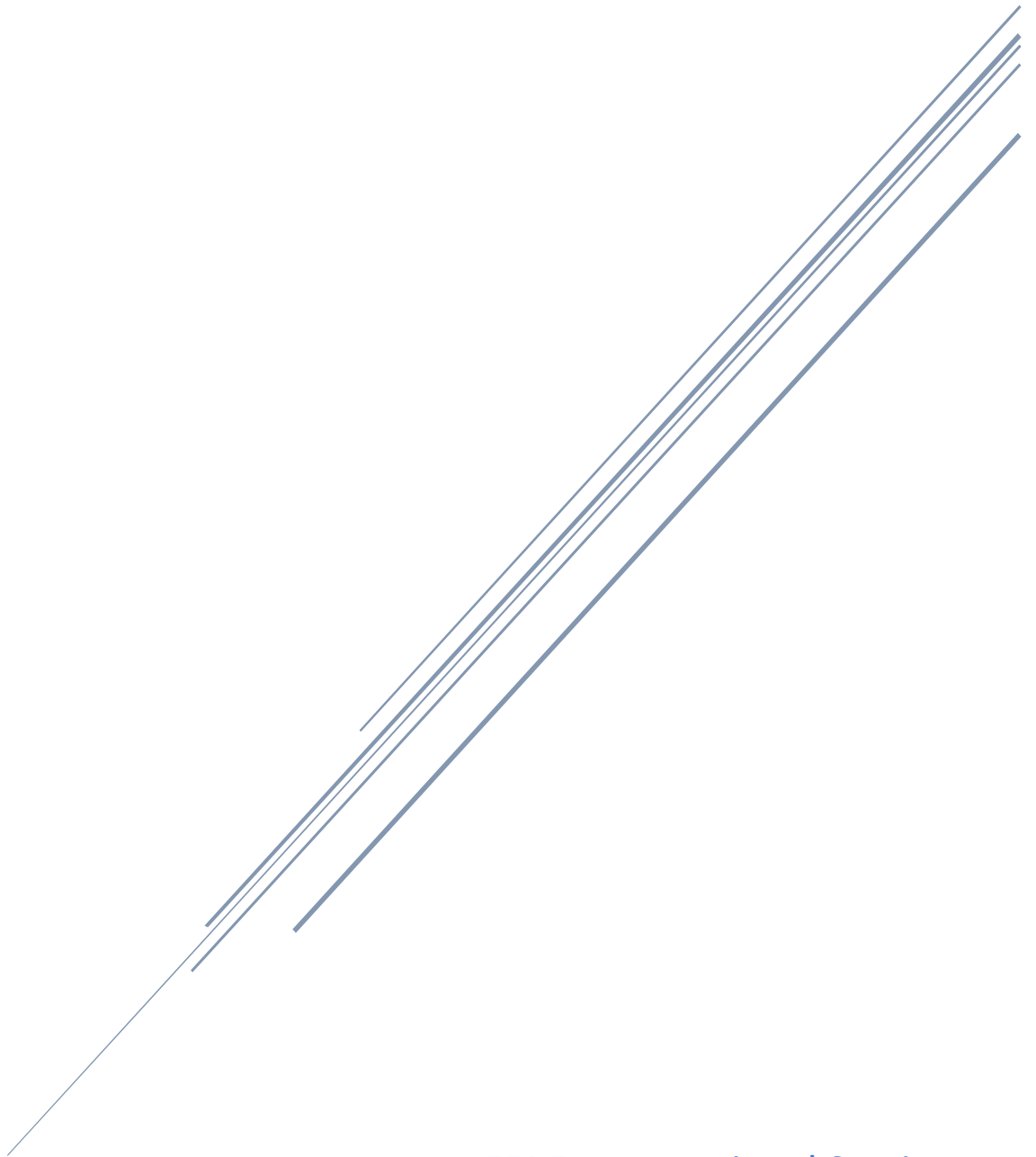


ALGORITHM INTERPRETER

Project 3



EFREI International Section
By Thibault LEPEZ & Camille BRIAND

I. Project goals

The main purpose of the project was to create an algorithm interpreter able to debug line by line in the console and to take inputs files. The subject gave us some bonus features as the support of floating numbers or error handling. We wanted to add as much functions as possible and we gave ourselves the challenge to be very memory efficient.

The memory management was one of our main concern. We constantly check memory leaks and memory allocation. Right now, our program is memory leak free and we are proud of that.

To have a rigorous code the organization of the work was very important in our project.

We wanted our program to be fancy and as user-friendly as possible so that took us a lot of work: to do so we made distinction between 12 different errors (allocation error, value assignation error, syntax error, structural error, illegal operation error...).

II. Work organization

For the project 2 we used Dropbox to work together and even if it's a very efficient way to exchange files the main problem was that we could not work on the same file without resulting in conflict when saving our work. Another problem with Dropbox was that our saves were the same so any mistake made was instantly uploaded in all of our computers.

Therefore, after hearing about GitHub, at a EFREI Linux meeting, we decided to change our organization.

We learned how to use GitHub in our association EFREI Linux. This service allowed us to upload our code only if we were sure of its quality. It also allowed us to work on the same file as long as we don't change the same part, which is much more convenient.

The only "problem", even if it's not really one, was that GitHub is much more easy and fast to use on Linux.

III. Programming and issues

We started the project directly after knowing about the subject. But at first, we thought it was mainly an algorithm to C language translator. After reading the subject we understood that it was an algorithm interpreter, as we both did some python programming we were inspired by the native python interpreter.

On firsts weeks, we started programming in Windows OS (Windows 10) as the major part of the class did so. But even if we were working with Windows we implemented the possibility to choose between UNIX based OS and Windows ones.

By working with Windows, we started the project with Dropbox. But during the project we learned how to use GitHub and this made us change our organization and we decided to work with LINUX.

When we compute our program the first time with LINUX it instantly crashes for memory reasons (core dumped, which means that we were trying to access an non allowed RAM "case"). At this moment, we understood that Windows was much more permissive with memory as we had no problems since then, this was also an explanation of why our second project this year was randomly crashing apparently for no reason.

We opened an issue in our GitHub to keep the process of our program (link: <https://github.com/Mr-Monster-0248/C-project-3/issues/1>).

Thanks to Mr. Klai who advised us to use Valgrind we manage to fix our memories issues. Valgrind took us about a week to understand how to use it. And finally, it became indispensable for our program as we wanted to have the most efficient one.

After solving memory issues, we started the calculus function. For no reason, we were a bit scared by this function which we were thinking will be huge and complicated in its algorithm. Mr. Busca advised us to use recursivity, and so did we but not in the way he was advising: we only used it for the calculus of expressions between parentheses. We were often saying *"Of course we know how to make our program able to perform $2 + 2$, but we do not know how to make $2 + 2 + 2$ work!"* since the multiplicity of elements in the expression was scaring us.

To do so we use many functions, the first one to find the highest priority operator and which return the position of the character. Then another function compute the operation between the numbers. And we do so while there is only one number left. Adding the parenthesis support was challenging. At first we had memory issues with the recursive call of the function (GitHub link: <https://github.com/Mr-Monster-0248/C-project-3/issues/3>).

We added the string concatenation very easily as we stock all our variables with strings. We just needed to recognize when to compute the strings operations.

The support of the boolean expression was built as the operation function. Once we recognize a comparison we compared both numbers or expression. The only problem is that we are not able for the moment to compute an operation inside a comparison as with $(2 + 2 = 4)$. But maybe by changing the recognition of an operation we should be able to compute this sort of expression.

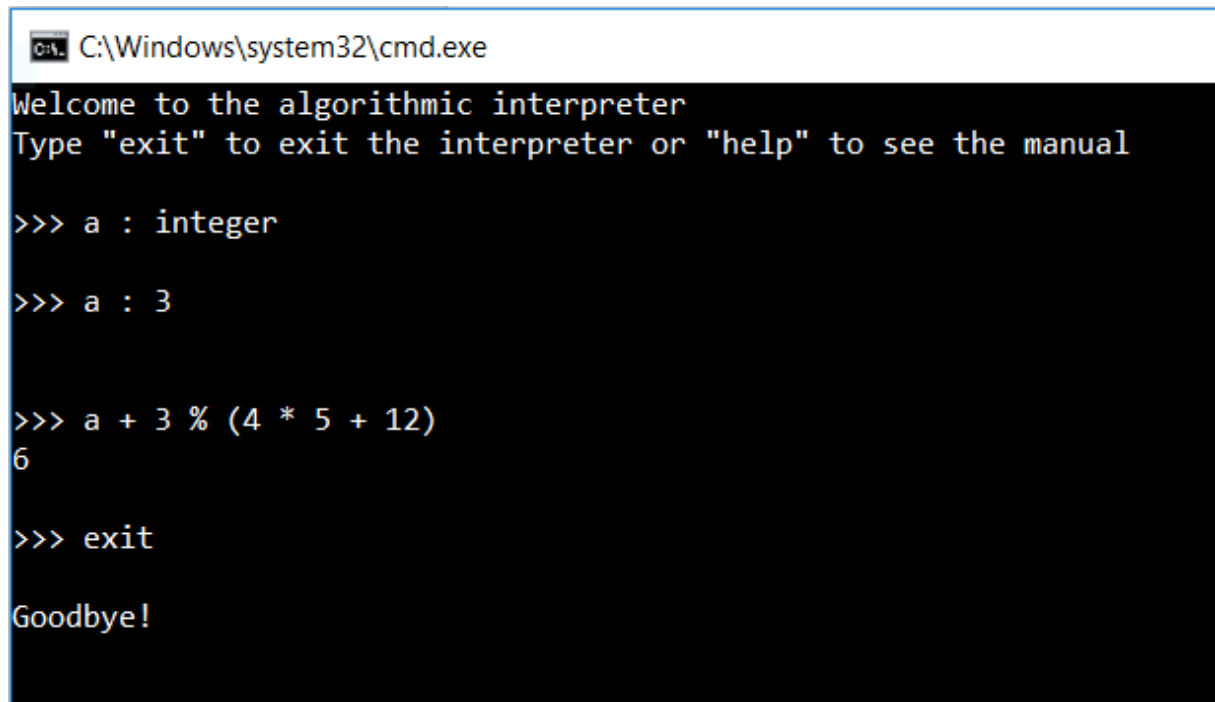
Our next challenge was to support variables, to do so we used structures. More precisely we use an array of structures to stock those variables. For the project defense, we weren't able to use those variable but the stocking was working. Finally, we manage to use variables, what prevented us of using them was simply that when we implemented a function to store new values in existing variables (knowing that a newly declared variable is always set to value *~undefined~*) we misunderstood the return of the string.h function strcmp().

IV. Conclusion

This project took us a lot of work (as you may see by looking at the numerous commits) and time. It was very interesting, mainly because it was challenging to implement such a complicated program in two months, but also because it is a program that may really be useful for us as students but also for programming beginners, and the goal of being as user-friendly as we could became a real thing for us.

Because we had to learn how to use GitHub, we also discovered the Markdown language to structure files in order to make their display fancier on Github website. (You may find a Markdown version of this report on our Github)

We will probably continue the project after the deadline. As we put it on GitHub with this link: [Github: C Programming language - Project 3](#), and we want to add the algorithmic language to C language translator feature, but first we need to finish to implement all the interpreter of this language.



A screenshot of a Windows command prompt window. The title bar at the top shows the icon for a command prompt and the text "C:\Windows\system32\cmd.exe". The main area of the window has a black background with white text. The text displayed is as follows:

```
Welcome to the algorithmic interpreter
Type "exit" to exit the interpreter or "help" to see the manual

>>> a : integer

>>> a : 3

>>> a + 3 % (4 * 5 + 12)
6

>>> exit

Goodbye!
```

Figure 1. The program running under Windows 10 (version 1.1.0)