TP Software Engineering

INTRODUCTION

Integers or real numbers are managed in a special way when programming. In C language for example, an int is mostly stored on 4 bytes and a float on 4 bytes. This implies that their size is limited. For example, a signed int cannot exceed 2,147,483,647 (232/2) if it is signed and 4,294,967,294 (232) if it is unsigned. Floats are stored differently from int. The data structure is more complex. A float has a mantissa and an exponent.

$1,2345 = 12345 \times 10^{-4}$ becomes once stored as a `float` :

Sign bit	Exhibitor							Mantisse																							
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1

Figure 1 Representation of the memory storage of a float in C language

This means that some calculations are not possible using these types of data.

Thus the following program :

```
int main (void) {
    float i = 123456789.0;
    printf ("i+1-i = %f\n", i+1-i);
    return EXIT_SUCCESS;
}
```

Returns an inconsistent result :

```
i+1-i = 0.000000
```

For most situations these types of data are sufficient and if we need more, there are the double and long types. However, what to do when the situation requires a numerical accuracy of 10-50 ready or when the significant number of digits of an integer is 100 or 1,000.

benjamin.maurice@telecomnancy.net
jeanaymeric@gmail.com

CONTENTS

HUGENUMBERCALCULATOR PROJECT SPECIFICATIONS

As you will have understood the idea is to build a calculator of very large numbers.

In order to allow the storage of these very large numbers (integers and reals), they will be stored using a chained list. Each of the numbers making up the number will be stored in one of the links of the chain. A small illustration will make things clearer :

This is how the integer 1,234,567 will be stored :

Figure 2 Illustration of the storage of a large number in a chained list

This is only a rough structure, it is of course up to you to define the exact structure allowing you, in addition to the numbers that make up the large number, to also store its sign. In the same way, the numbers may have to be stored upside down in the chained list in order to facilitate the programming of additions, subtractions, multiplication and division.

Moreover, for real numbers, it could be judicious to use a more complex structure. To store the number 10^{-100}, it is not useful to clutter up the memory by storing all 0's. However, this structure must be able to perform the $1-10^{-100}$ operation correctly.

SERVICE FUNCTIONS

Main functions

- FP1.1 : HugeNumberCalculator must allow for addition between two signed integers with no limit on the number of digits.
- FP1.2 : HugeNumberCalculator must allow the addition between two unsigned integers without limit of number of digits.
- FP1.3 : HugeNumberCalculator must allow the addition between two signed or unsigned real numbers without limit of number of digits.
- FP2.1 : HugeNumberCalculator must allow subtraction between two signed integers without limit of number of digits.
- FP2.2 : HugeNumberCalculator must allow subtraction between two unsigned integers without limit of number of digits.
- FP2.3 : HugeNumberCalculator must allow subtraction between two signed or unsigned real numbers with no limit on the number of digits.
- FP3.1 : HugeNumberCalculator must be able to multiply between two signed integers without limit of number of digits.
- FP3.2 : HugeNumberCalculator must be able to perform a multiplication between two unsigned integers without limit of number of digits.
- FP3.3 : HugeNumberCalculator must be able to perform a multiplication between two signed or unsigned integer real numbers without limit of number of digits.

- FP4.1 : HugeNumberCalculator must be able to perform a Euclidean division between two signed integers without limit of number of digits.
- FP4.2 : HugeNumberCalculator must be able to perform a Euclidean division between two unsigned integers without limit of number of digits.

Constraint functions

- FC1 : HugeNumberCalculator must be written in C language.
- FC2 : HugeNumberCalculator must be accompanied by Doxygen documentation written in English.
- FC3 : HugeNumberCalculator must be made up of source code written in English.
- FC4 : HugeNumberCalculator must be modular. The different libraries must be as independent as possible.
- FC5 : HugeNumberCalculator must be a console application.
- FC6 : HugeNumberCalculator must respect C code writing standards.

UML MODELING OF SERVICE FUNCTIONS

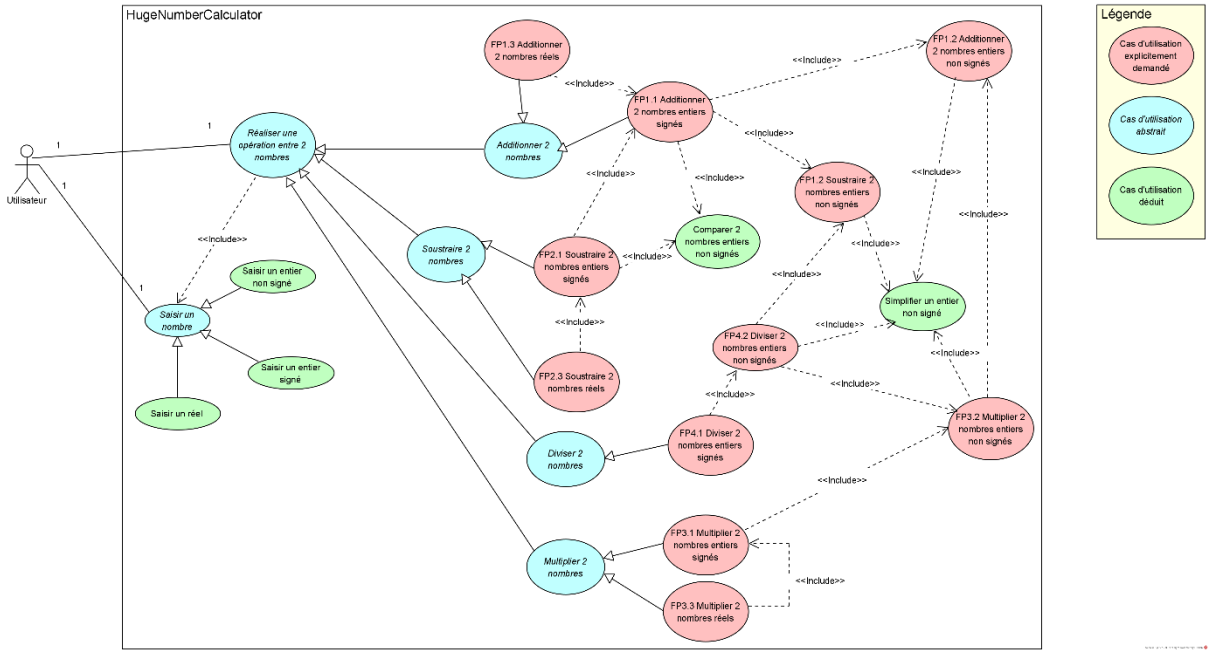

Figure 3 HugeNumberCalculator Use Case Diagram

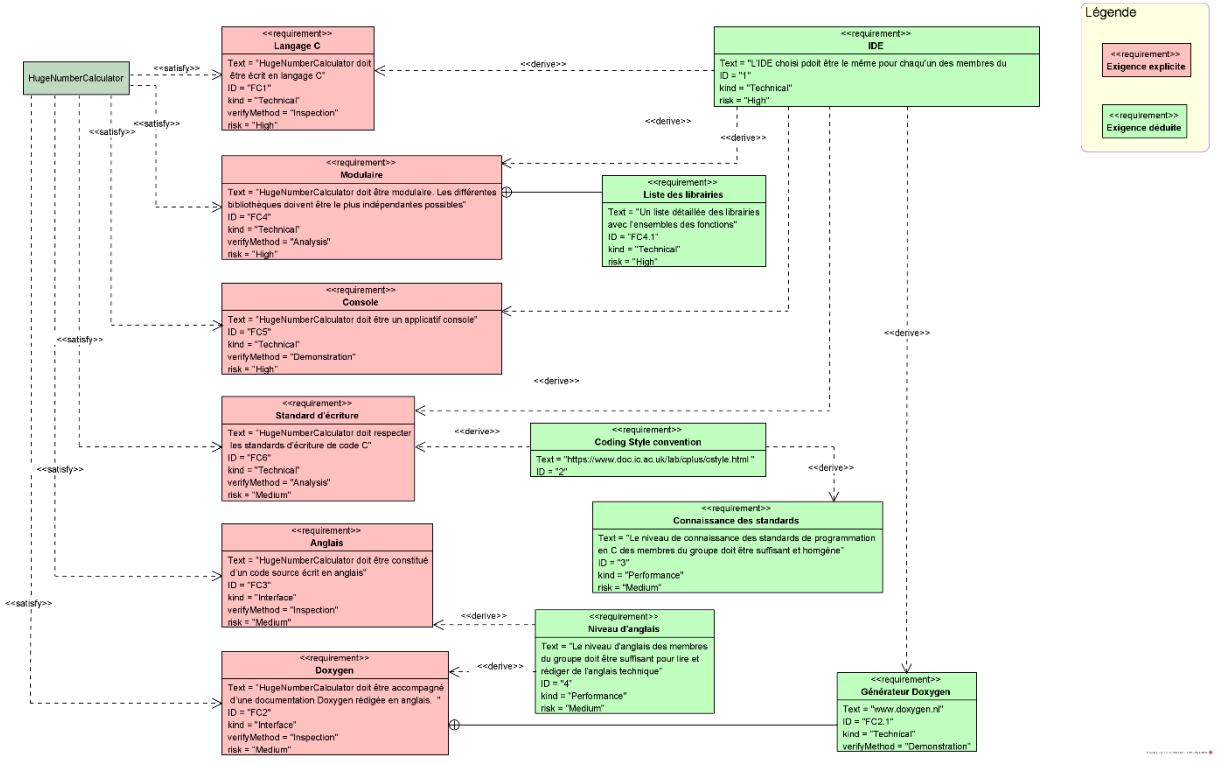

Figure 4 HugeNumberCalculator requirements diagram

WORK REQUIRED FOR TP1

A complete and detailed architecture of the HugeNumberCalculator programme. It must contain :

- A list of all .h and .c files.
- The relationships envisaged between the different .h
- The signature of the functions by .h
- The new data types defined

A division of labour. This division must contain :

- The method of work envisaged
- The tools used :
    - Choice of FDI
    - Choice of collaborative work tools
- The list of tasks to be carried out and its assignment to a member of the group.

RESOURCES:

Use case and requirement diagrams are available:

- Use Case Diagram.png
- Requirement Diagram.png
- HugeNumberCalculator.vpp.

The software used to build these diagrams is Visual Paradigm Community Edition. The software can be downloaded here :

https://www.visual-paradigm.com/download/community.jsp

Be careful to select the Community Edition version, which is free, the other version offers more functionalities, however it is not free of charge and is blocked after 30 days. In view of the time you have available, we do not recommend that you download this software during your work. Moreover, this software is a professional software, so its good use requires a rather important learning time. The images of the two diagrams are more than sufficient to do this. However, at the end of the practical work, it may be interesting for you to familiarise yourself with this modelling tool.