Algorithmics and C Programming, Project
Autumn 2011
LO27 and LO21

Calculation and arithmetic representation of big integers

Nicolas Gaud and Jonathan Demange

19 octobre 2011

1 Implementation and evaluation of the project

- Group of 1 or 2 persons of the same UV (LO27 or LO21) at most.
- Implementation in C language.
- **The project will be presented during a TP session in January 2011 : Wednesday, January the 4^{th}, Friday, January the 6^{th} and Monday, January the 9^{th}. Whatever your original TP group, you must register to present your project during one of these TP sessions.**
- The project will be documented in a written report that outlines (15 to 20 pages) used data structures, the selected algorithms and optimizations and possibly the encountered difficulties. No copy-paste of entire pages of source code.
- The source codes will be commented and the names of various authors of the project have to be precised at the beginning of the source file as well as a textual description specifying its purpose.
- The program will have a minimal GUI console. The GUI is not the core of the project, it should nevertheless be simple and user friendly.
- the report in **PDF** format and the source code have to be delivered later **January the 3^{rd} 2011 at 17h** by mail adressed to `nicolas.gaud@utbm.fr` for LO27 and `jonathan.demange@utbm.fr` for LO21, (Object entitled : LO[27/21] Project - Group : StudentName1Uppercase and StudentName2Uppercase) in a ZIP or TAR.GZ archive named in the following way :
  LO[27/21]_StudentName1Uppercase_StudentName2Uppercase.zip

Any delay or failure to comply with these guidelines will be penalized.

A special attention should be paid on the following points when writing the program : the program runs smoothly without bugs (it's better to not provide a feature rather than to provide it if it does not work, negative points), readability and clarity of the code (comments and indentation), complexity and efficiency of proposed algorithms, choice of data structures, modularity of the code (development of small functions, distribution in various files comprising functions consistently and appropriately named).

2 Project goal

The goal of this project aims at providing a library of features for handling arithmetic expressions using big integers and a program enabling a user to test this library in an interactive and practical way.

We can assume, to simplify the calculations and adopt the types of data indicated below, that the greatest integer typed unsigned long int on the computer where the program will be implemented is actually higher than 9999×9999. This assumption, usually correct, should be verified by the library functions.

3 Representation of integers using linked lists

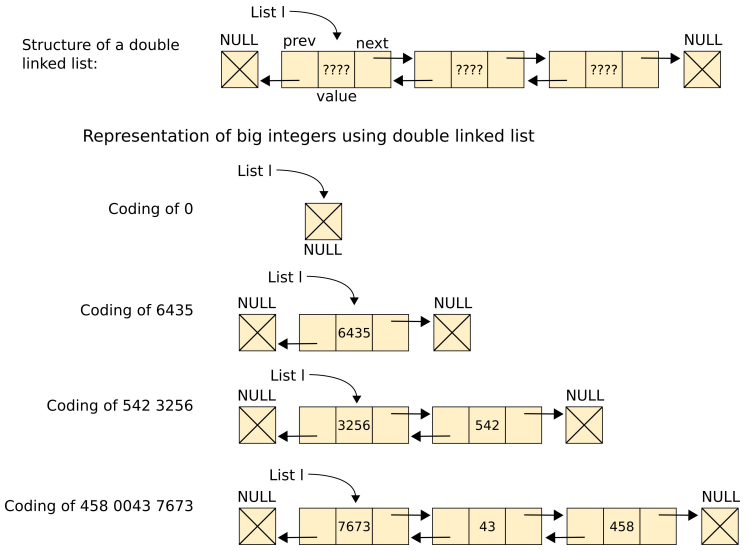

FIGURE 1 – Examples of integer representation using double linked lists

3.1 Representation of unsigned integers (greater or equals to zero)

An unsigned integer in arbitrary precision will be encoded in base 10000. Note that coding an integer in base 10,000 consists in firstly writing it in base 10, then gathering the digits four by four starting with the rightest. The "figures" in base 10 000 are thus the integers from 0 to 9999.

An integer n will be represented by a double linked list. Each item in this list will contain a "figure" related to the coding of n in base 10000, i.e, a number between 0 and 9999. To simplify the development of arithmetic operations, the coding will begin with the right.

Thus, the first element of the list corresponds to the number of "units" in base 10000, the value of n modulo 10000 (gather figures of the units, tens, hundreds and thousands in the coding of n in base 10). The empty list will represent the integer zero. The lists will be standardized : the last element of a list should contain a non-zero figure. Figure 1 details some examples of integer coding.

3.2 Representation of signed integers

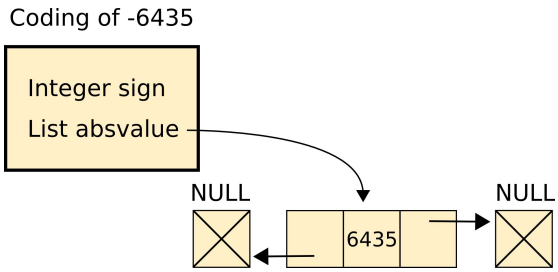

FIGURE 2 – Example of integer representation using a structure containing a double linked list corresponding to the absolute value and an integer for the sign

A signed integer in arbitrary precision will be represented by its sign and its absolute value, represented as above with a double linked list (cf figure 2).

4 Work to achieve

4.1 Library

A library written in C containing the types and functions described below.

Libraries will be implemented using at least two C files (header and source). The final archive will at least contain in a dedicated directory the following files :

biginteger.h the header file that contains the prototype of the functions, types, constants and variables provided by the library managing BigIntegers ;

biginteger.c the associated source file that contains the body of the various proposed functions.

dlinkedlist.h the header file that contains the prototype of the functions, types, constants and variables provided by the library managing doubly linked lists of integer ;

dlinkedlist.c the associated source file that contains the body of the various proposed functions.

Makefile makefile to compile the sources, generate libraries and executable. At least, 3 targets in this Makefile :

- **all** compiles everything, generates the libraries and executable.
- **lib** generates the binary code of the library libBigInteger.so or BigInteger.dll
- **clean** cleans the tmp files generated during the compilation process and the various binary files

main.c the main program.

A dynamic library may be compiled using the following command line :

$gcc -Wall -pedantic <source files> -o libBigInteger.so -shared -fpic

Types

- the type *BigInteger* enabling to code signed big integers ;
- the type *Boolean* enabling to represent a boolean.

Functions

- $newBigInteger : char* \rightarrow BigInteger$, build a new *BigInteger* from its string-based representation ;
- $printBigInteger : BigInteger \rightarrow void$, enabling to print a *BigInteger* ;
- $isNull : BigInteger \rightarrow boolean$, returns true if the specified *BigInteger* is null, false otherwise ;
- $signBigInt : BigInteger \rightarrow int$, returns a negative integer, zero, or a positive integer as the value of the specified *BigInteger* is lesser than, equal to, or greater than zero ;

$$signBigInt(e_1) \text{ is equal to } \begin{cases} 0 \text{ si } e_1 = 0 \\ 1 \text{ si } e_1 > 0 \\ -1 \text{ si } e_1 < 0 \end{cases}$$

- $equalsBigInt : BigInteger \times BigInteger \rightarrow Boolean$, true if the two specified *BigIntegers* are equals ; false otherwise ;
- $compareBigInt : BigInteger \times BigInteger \rightarrow int$, compare the two specified *BigIntegers* and returns a negative integer, zero, or a positive integer as the first *BigInteger* is lesser than, equal to, or greater than the second ;

$$compareBigInt(e_1, e_2) \text{ is equal to } \begin{cases} 0 \text{ si } e_1 = e_2 \\ 1 \text{ si } e_1 > e_2 \\ -1 \text{ si } e_1 < e_2 \end{cases}$$

- $sumBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the sum of the two specified *BigIntegers* ;
- $diffBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the difference between the two specified *BigIntegers*, $diffBigInt(e_1, e_2) \Leftrightarrow e_1 - e_2$;
- $mulBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the product between the two specified *BigIntegers* ;
- $quotientBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the quotient of the Euclidean division of the two specified *BigIntegers* ;
- $restBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the quotient of the Euclidean division of the two specified *BigIntegers* ;
- $gcdBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the greatest common divisor of the two specified *BigIntegers* ;
- $lcmBigInt : BigInteger \times BigInteger \rightarrow BigInteger$, computes the least common multiple of the two specified *BigIntegers* ;

– $factorial$: $unsigned\ long \rightarrow BigInteger$, computes the factorial of the specified unsigned long integer (*unsigned long int*), $n! = n \times (n-1) \times \cdots \times 3 \times 2 \times 1$;

– cnp : $unsigned\ long \times unsigned\ long \rightarrow BigInteger$, computes the combinatorial of the two specified unsigned long integers $C_n^p = \frac{n!}{p!(n-p)!}$.

4.2 Main program and graphical user interface

To provide a main program using the previous library and enabling a user to test all of its provided functionnalities in a simple and friendly way.

This program will be compile using the following command line :
$ gcc -Wall -pedantic -L<repertoire bibliothèque> main.c -o main.exe -lBigInteger

In running the program, the variable LD_LIBRARY_PATH have to specify the directory containing the library *BigInteger*.

5 The project deliverables

A ZIP archive or TAR.GZ
– the report in pdf format
– Makefile to build the library and get the executable associated to the file main.c
– source files of the libraries : BigInteger.c and BigInteger.h, DLinkedList.c and DLinkedList.h.
– the binary of the library libBigInteger.so
– the main program main.c