

Computer Programing

Tomasz Gorol

# **LONG INTEGERS**

tutor: Piotr Fabian, Ph. D

Gliwice, 25 April 2018

# 1 INTRODUCTION

The following program is an individually assigned project in the CP course.

## Long Integers

Write a program capable of operating on big integers (at least 50 decimal digits) with full precision. At least four basic operations (+, -, \*, /) should be implemented.

# 2 SOLUTION ANALYSIS

As the requirement is to operate on at least 50 decimal digits, the program must not rely on any common types of variables, such as int, long or even long long. It seems that the best option is to use int arrays.

User opens the program, chooses desired mathematical operation from the menu and provides input (2 numbers). Program operates only on absolute values, whereas signs of the numbers are processed separately. In the end, the solution is being printed onto the screen.

# 3 INTERNAL SPECIFICATION

## Important preprocessor definitions

```
#define SET_AMOUNT_OF_DIGITS_HERE 50
```

As the name says, this is the place to define how many digits we want the program to operate on. The only limitance here is the size of int array.

```
#define DEBUG 0
```

This stands for debugging mode. Change the value to 1 to enter the mode. The mode contains a series of various tests, checks etc.

## Structures

```
struct BigInt{  
    bool hasExceeded = false;  
    int value[DIGITS];  
    bool sign = true;  
};
```

The whole program relies on this structure.

Variable *hasExceeded* is false by default and turns true if the program tries to perform an operation that exceeds the amount of digits set with the preprocessor definition.

Integer array *value* stores all the digits of a number, of course and sign stands for the signs of the number (+ or -).

## Functions

*BigInt add(BigInt num1, BigInt num2, bool s)*

The function implement the operation of the columnar addition.

*BigInt subtract(BigInt num1, BigInt num2, bool s)*

The function implement the operation of the columnar subtraction.

*BigInt multiply(BigInt num1, BigInt num2)*

The function implement the operation of the columnar multiplication.

*BigInt times10(BigInt num)*

Multiplies a number 10 times by shifting digits to the left and adding a 0 in the end.

*BigInt divide(BigInt num1, BigInt num2)*

The function implement the operation of the columnar division.

*BigInt divide10(BigInt num)*

Divides a number by 10 by shifting digits to the right and cutting the last (0) out.  
Called only after if `times10()` was used.

*BigInt compare(BigInt num1, BigInt num2, int mode)*

Put `GET_SMALLER` or `GET_BIGGER` as mode to obtain desired result.

*BigInt chooseAddSub(BigInt num1, BigInt num2, int mode)*

This one is used to call for addition or subtraction but even before that, to put the numbers in a convenient order.

*bool isEqual(BigInt num1, BigInt num2)*

Returns true if the numbers equal each other.

*void initializeNumber(BigInt \*num)*

Marks every digits of a nuber as unused by putting a value that is not possible to obtain for the program during calculations.

*bool setNumber(BigInt \*num, string value)*

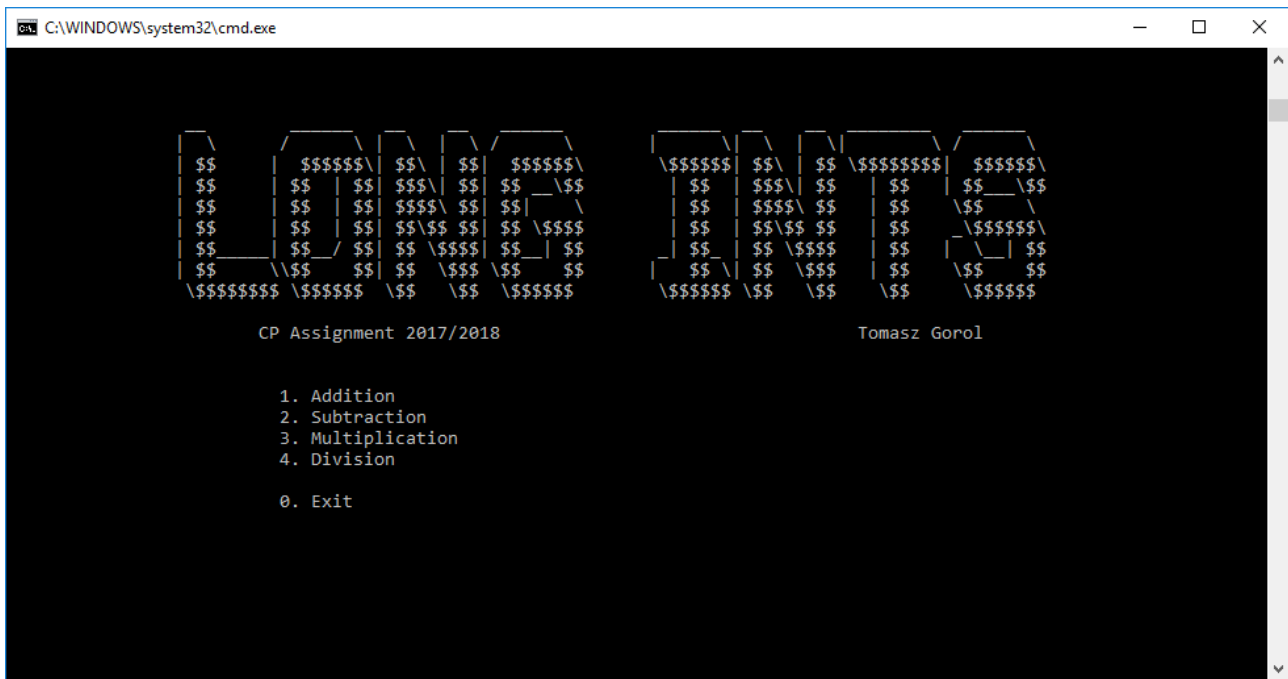
Very useful function which sets a value of a number basing on the provided string. If this string contains any 'minus' sign then the sign of the number is changed.

*void printNumber(BigInt num)*

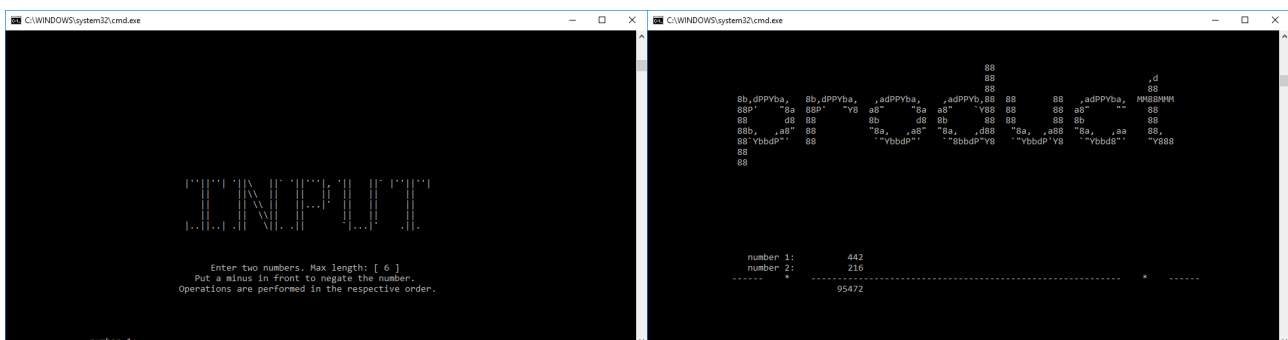
Prints the number onto the screen. Digits marked as unused remain unprinted.

## 4 EXTERNAL SPECIFICATION

The program is as simple to use as possible. User is welcomed by a nice-looking menu and is asked to choose one of five options (along with Exit).



As he chooses, program asks for the input numbers. After providing two numbers, chosen operation is performed on the numbers and solution is being printed onto the screen.



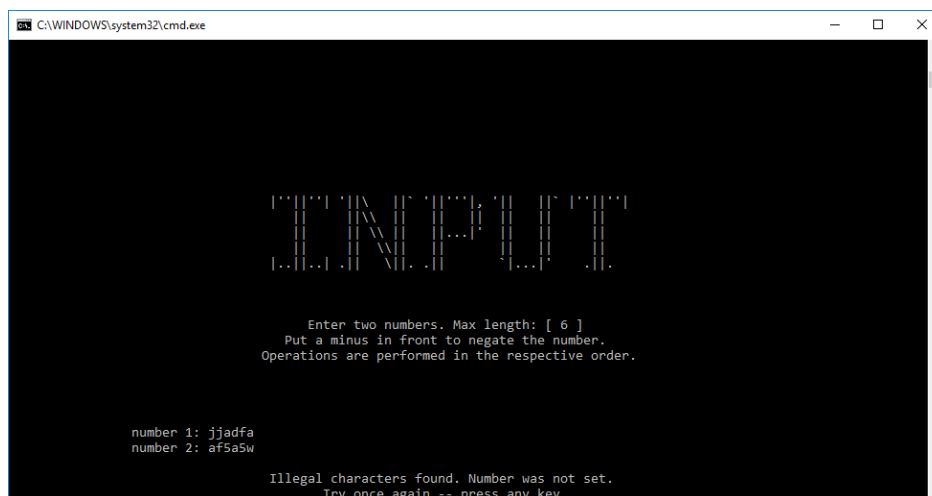
Once user has the result, he can press any key to move back to the main menu.

## 5 TESTING

The program was from the beginning written in the way, so that user won't make program fail. There are bunch of checks helping that. They consider input state mostly, for obvious reasons.

### Illegal characters

If user (by the mistake of course) tried to provide anything different but minus sign or number, program will notice this fact and inform user about it, asking kindly to try once again.



```
C:\WINDOWS\system32\cmd.exe

                                INPUT

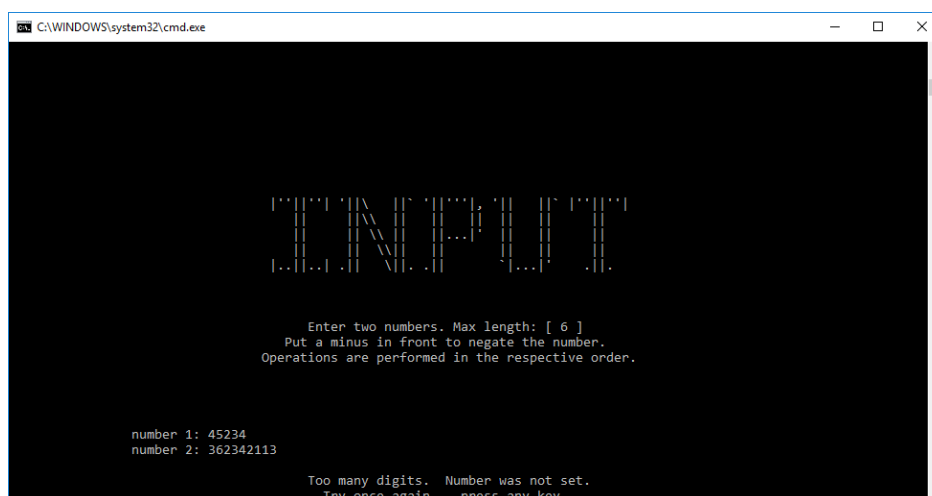
Enter two numbers. Max length: [ 6 ]
Put a minus in front to negate the number.
Operations are performed in the respective order.

number 1: jjadfa
number 2: af5a5w

Illegal characters found. Number was not set.
Try once again -- press any key.
```

### Too many digits

If user happened to provide more digits that the current capability of the program, it will inform about the fact and ask to try again.



```
C:\WINDOWS\system32\cmd.exe

                                INPUT

Enter two numbers. Max length: [ 6 ]
Put a minus in front to negate the number.
Operations are performed in the respective order.

number 1: 45234
number 2: 362342113

Too many digits. Number was not set.
Try once again -- press any key.
```

## Solution would have too many digits

Even assuming user provides proper amount of digits, program checks if the solution won't exceed the capability, which might be the case during multiplication. If so, the program will proceed to the solution panel, yet instead of the solution there will be an information about the problem and suggested fix.



```
C:\WINDOWS\system32\cmd.exe

      88
      88
      88
8b,dPPYba, 8b,dPPYba, ,adPPYba, ,adPPYb,88 88 88 ,adPPYba, MM88MMM
88P' "8a 88P' "Y8 a8" "8a a8" ^Y88 88 88 a8" "" 88
88 d8 88 8b d8 8b 88 88 88 8b 88
88b, ,a8" 88 "8a, ,a8" "8a, ,d88 "8a, ,a88 "8a, ,aa 88,
88`Ybbdp"" 88 ^"Ybbdp"" ^"8bbdp"Y8 ^"Ybbdp"Y8 ^"Ybbd8"" "Y888
88
88

number 1: 553562
number 2: 11234
----- * -----
EXCEEDED. Change settings to fit that many digits.
```

**It can be safely said that the program works exactly as planned and moreover, is errorproof.**