­

**Infinite Int Calculator Report**

**CAU 23-2**

**OOP Project02**

|  |  |
| --- | --- |
|  |  |
| Subject | Object Oriented Programming |
| Professor | Bong-Soo Sohn |
| Major | Computer Science and Engineering |
| Team | TEAM 07 |
| Members | 20204946 이규성  20203436 이병구  20203458 조영호 |

Contents

[(a) Summary - 3 -](#_Toc150097617)

[(b) Build and Run - 3 -](#_Toc150097618)

[Build - 3 -](#_Toc150097619)

[Run - 3 -](#_Toc150097620)

[Test environment - 3 -](#_Toc150097621)

[(c) Important Functionality - 4 -](#_Toc150097622)

[(d) how you implemented (important implementation issues) - 4 -](#_Toc150097623)

[inf\_int class - 4 -](#_Toc150097624)

[calculator class - 4 -](#_Toc150097625)

[(e) the result of SW system design - 5 -](#_Toc150097626)

[(f) execution results - 6 -](#_Toc150097627)

[(g) Applying object-oriented concepts and Learning OOP-related things - 6 -](#_Toc150097628)

[(h) Conclusion - 7 -](#_Toc150097629)

## (a) Summary

Presentation speaker: 이규성

Project github Link : <https://github.com/Tastypotato245/OOP_Project02>

Description: This program can do infinite integer calculations.

Feature:

* inf\_int can be made from normal int, string, and another inf\_int instance.
* Assignment overloading, copy constructor, destructor
* Comparison operation between inf\_int (==, !=, >, <)
* Three basic operation between inf\_int (+, -, \*)
* Calculation absolue value (abs())
* inf\_int can be printed (<<)

## (b) Build and Run

### Build

$ make

### Run

$ ./inf\_int 9999999999999999999 + 999999

$ ./inf\_int 12345678901234567890 \\* 123456789

$ ./inf\_int

Input: 54 \* 123456788888

Output: 6666666599952

Input: 2345234523525345 + 22233322

Output: 2345234545758667

Input: ^C (terminating)

Input: 0 (terminating)

### Test environment

* macOS 14.0 arm64 (clang 15.0.0)
* Debian GNU/Linux 11 aarch64 (g++ 10.2.1)

## (c) Important Functionality

Through the method of storing numbers in reverse order using a dynamically assigned array of characters, integers are stored and processed without limitation to the size of the numbers.

The algorithm was implanted to efficiently manage carry and borrow that can occur during calculation.

Positive and negative numbers are accurately classified through sign processing, and the sign of the operation result is determined. Can also be calculated for many numbers and signs.

## (d) how you implemented (important implementation issues)

### inf\_int class

We classified the cases.

* positive number + positive number
* positive number + negative number
* negative number + positive number
* negative number + negative number

A piece of paper with writing on it

Description automatically generated

And substitution is also classified. Then we implemented each operation.

### calculator class

We made two important method: Calculator::calculate, Calculator::infix\_to\_postfix.

Infix\_to\_postfix method changes infix to postfix by using stack and priority table between operators. Calculate method calculates postfix expressions and generates a result by using stack.

## (e) the result of SW system design

텍스트, 도표, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

Main knows queue, inf\_int. Calculators, and using inf\_int and Calculator with STL functions and Classes, we can implement inf\_int Calculators.

## (f) execution results

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

## (g) Applying object-oriented concepts and Learning OOP-related things

## Encapsulation and abstraction were implemented through an overloading of constructors, dissipators, and operators. Ensure the correct copying of objects through copy generators and substitution operators. The class member function improves control over access and manipulation of data.

## (h) Conclusion

We were able to write git and GitHub while working on a team project. And it helped me learn functions such as merge, issue, and pull requests of git and GitHub. In addition, implementing infinite integer operations through C++ caused many memory problems. It took a lot of time to find and fix the part where the problem occurred. In addition, the ability to structure and abstract programs has improved by using subject-oriented techniques.