OOP Assignment

(inf\_int Class)

Team #8

유용민

설지환

안지완

이의제

이주형

채승운

최동욱

Index.

* Introduction
* Concept of the project
* Required Components
* Specification of the functions
* Implementation
* Execution Results
* Conclusion

List of team members

|  |  |  |
| --- | --- | --- |
| Name | ID | Role |
| 유용민 |  |  |
| 설지환 |  |  |
| 안지완 |  |  |
| 이의제 |  |  |
| 이주형 |  |  |
| 채승운 |  |  |
| 최동욱 |  |  |

Introduction

Our team’s project is mainly about “making a class that can maintain an infinite length of number, having a basic operator of addition, subtraction, and multiplication.” As a basic ‘Integer class’ given by C++ can only maintain at most approximately 2 billion, we should create a customized integer class, named as ‘inf\_int.’ In this report, we will explain how we made a brief concept, specification, and implementation of this customized ‘inf\_int’ class.

Concept of the ‘inf\_int’

As we explained above, a basic integer class (a.k.a ‘int’) can only maintain at most 2 billion, which means that assigning above 2 billion to the ‘int’ will make an overflow of the program, leading to the fatal error to the program. To prevent this, we decided to use ‘string’ instead. A string type can maintain an infinite length of the string (actual length is 4.2billion, which is still close to infinite, 2^(4.2billion)). A brief abstraction of the class ‘inf\_int’ looks like this:

class inf\_int

{

private :

char\* digits;

unsigned int length;

bool thesign;

public :

inf\_int();

inf\_int(int);

inf\_int(const char\* );

inf\_int(const inf\_int&);

~inf\_int();

inf\_int& operator=(const inf\_int&);

friend bool operator==(const inf\_int& , const inf\_int&);

friend bool operator!=(const inf\_int& , const inf\_int&);

friend bool operator>(const inf\_int& , const inf\_int&);

friend bool operator<(const inf\_int& , const inf\_int&);

friend inf\_int operator+(const inf\_int& , const inf\_int&);

friend inf\_int operator-(const inf\_int& , const inf\_int&);

friend inf\_int operator\*(const inf\_int& , const inf\_int&);

friend ostream& operator<<(ostream& , const inf\_int&);

};

A customized class ‘inf\_int’ has a responsibility to not only operate basic arithmetic functions properly, but also perform comparison operator and I/O functions. Thus, we overrode the basic operators to get our ‘inf\_int” class as the parameters, declaring them as ‘friend’ functions to give them permission to access the internal data (digits, length, thesign). By doing this, operators will access the char-type digits and perform the functions by reading each digit. Now we need to specify the implementations.

Specifying each implementations

Constructors

Before explaining the implementation of the function, we first need to specify the structure of the internal data (digits, length, thesign). As explained above, we should store digits in char type. However, since we cannot figure out the exact length of the number before declaring it, we should not use the static array. Instead, we should use ‘dynamic allocation.’ This concept will be used in the constructor of the ‘inf\_int’ class, as the internal members are initialized when the constructor is called.

Four types of constructors are declared in the ‘inf\_int’ class : One that gets an integer as a parameter, one that gets a string, another as a copy constructor, and the other with none parameter.

Starts with the first (integer), we should beware that the structure of a string type, and the integer type is different. As we assumed that the length of the digits is infinite, we cannot simply cast the integer to the string type. If doing so, it won’t properly perform the arithmetic functions with another ‘inf\_int’ object having infinite length of digits. Considering this case, we should divide the integer with 10.

15491

-> 1549 … 1 ( Quotient | Remainder)

-> 154 … 9

-> 15 … 4

-> 1 … 5

-> 0 … 1