Object-Oriented Programming Lab #10

Department: 응용물리학과 Student ID: 2017103038 Name: 권인회

A. Code explanation & output analysis (Write the source code and results)

A-1. Listing 13.1

< 코드 >

```
#include <iostream>
#include <string>
int main() {
	// Declare a string object and initialize it
	std::string word = "fred";
	// Prints 4, since word contains four characters
	std::cout << word.length() << '₩n';
	// Prints "not empty", since word is not empty
	if (word.empty())
		std::cout << "empty₩n";
	else
		std::cout << "not empty₩n";
	// Makes word empty
	word.clear();
	// Prints "empty", since word now is empty
	if (word.empty())
		std::cout << "empty₩n";
	else
		std::cout << "not empty₩n";
	// Assign a string using operator= method
	word = "good";
	// Prints "good"
	std::cout << word << '₩n';
	// Append another string using operator+= method
	word += "-bye";
	// Prints "good-bye"
	std::cout << word << '₩n';
	// Print first character using operator[] method
	std::cout << word[0] << '₩n';
	// Print last character
	std::cout << word[word.length() - 1] << '₩n';
	// Prints "od-by", the substring starting at index 2 of length 5
	std::cout << word.substr(2, 5);
	std::string first = "ABC", last = "XYZ";
	// Splice two strings with + operator
	std::cout << first + last << '₩n';
	std::cout << "Compare " << first << " and ABC: ";
	if (first == "ABC")
		std::cout << "equal₩n";
	else
		std::cout << "not equal₩n";
	std::cout << "Compare " << first << " and XYZ: ";
	if (first == "XYZ")
```

```
                std::cout << "equal₩n";
        else
                std::cout << "not equal₩n";
}
```

< 실행결과 >

4

not empty

empty

good

good-bye

g

e

od-byABCXYZ

Compare ABC and ABC: equal

Compare ABC and XYZ: not equal

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 20004개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

이번엔 설명을 위에 적어둔 실행결과에 따라 하고자 한다. Class는 member를 가질 수 있고 바로 연산이 가능한 method도 가질 수 있다. Vector에서 다뤘던 것처럼 string object에도 포함된 method들이 있다. Element가 char인 vector를 string이라고 생각하면 편하다.

word라는 이름을 가진 string 객체를 "fred"로 초기화해준다. Word는 fred라는 문자열을 가진 string형이 되는 것이다. Length() method를 이용하여 word의 길이를 출력할 수 있다. 글자수가 4개이므로 4가 출력된다.

이후 첫번째 if-else문에서 empty() method를 이용하여 word 객체가 비어있는 경우 "empty"를 출력하고 아니면 "not empty"를 출력한다. 여기서, word에는 "fred"라는 문자열 값이 들어있으므로

비어있지 않아 not empty가 출력한다. 이후 clear() method를 이용하여 word 안의 내용을 모두 지운다. 그러면 비어있는 상태가 되어서 다시 if-else문을 구동하였을 때, empty가 출력된다. 그래서 차례대로 출력된 것을 볼 수 있다.

이후 word를 "good"으로 할당하는 것도 할 수 있음을 알 수 있고, 그 값도 바뀌어 그대로 "word" 문자열이 출력된다. +=연산자를 이용하여 word 객체에 새로운 문자열을 더해줄 수도 있다. "-bye"라는 문자열이 뒤에 붙어 word는 이제 "good-bye"라는 문자열을 가진 string 객체가 된다. 따라서, 그대로 출력된다.

Word[0]처럼 슬라이싱해서 출력도 가능하다. 이 경우 word에 들어있는 문자열을 하나하나 문자형으로 나눴을 때, index 0번의 문자형을 출력해준다. 따라서, g가 출력된다. 이후 코드는 word의 길이에 -1한 값을 index로 가지는 문자형을 출력해주므로, 가장 마지막 글자를 출력하는 것이다. 따라서, e가 출력된다.

Substr(a,b) method는 1번째 파라미터가 index, 2번째 파라미터가 길이를 뜻하며 a번째 index에서 길이 b만큼의 문자열을 뽑아내는 것이다. 따라서, good-bye에서 index 2번째인 o부터 길이 5만큼이므로 od-by가 출력되는 것이다. 그리고, 두 string 객체를 한 줄 코드에서 동시에 초기화할 수 있고, 서로 더해주는 것이 가능하다. 서로 더해주면 문자열이 그대로 이어진다. 그것을 ABCXYZ로 보여준다.

동등관계 비교도 가능하다. String 객체인 first에 "ABC"를 할당해주고 문자열 "ABC"와 같은지 물어보면 같다고 답해준다. 이것을 if-else문으로 구현해두었다. 그래서 처음엔 equal이라고 출력된다. 그 다음 if-else문은 first 객체와 문자열 내용이 다르기 때문에 not equal이 출력된다.

A-2. Listing 14.2

< 코드 >

```
#include <iostream>
#include <string>
#include <vector>
class Account {
public:
        // String representing the name of the account's owner
        std::string name;
        // The account number
        int id;
        // The current account balance
        double balance;
};
// Allows the user to enter via the keyboard information
// about an account and adds that account to the database.
void add_account(std::vector<Account>& accts) {
        std::string name;
        int number;
```

```
        double amount;
        std::cout << "Enter name, account number, and account balance: ";
        std::cin >> name >> number >> amount;
        Account acct;
        acct.name = name;
        acct.id = number;
        acct.balance = amount;
        accts.push_back(acct);
}
// Print all the accounts in the database
void print_accounts(const std::vector<Account>& accts) {
        int n = accts.size();
        for (int i = 0; i < n; i++)
                std::cout << accts[i].name << "," << accts[i].id
                << "," << accts[i].balance << '₩n';
}
void swap(Account& er1, Account& er2) {
        Account temp = er1;
        er1 = er2;
        er2 = temp;
}
bool less_than_by_name(const Account& e1, const Account& e2) {
        return e1.name < e2.name;
}
bool less_than_by_id(const Account& e1, const Account& e2) {
        return e1.id < e2.id;
}
bool less_than_by_balance(const Account& e1, const Account& e2) {
        return e1.balance < e2.balance;
}
// Sorts a bank account database into ascending order
// The comp parameter determines the ordering
void sort(std::vector<Account>& db,
        bool (*comp)(const Account&, const Account&)) {
        int size = db.size();
        for (int i = 0; i < size - 1; i++) {
                int smallest = i;
                for (int j = i + 1; j < size; j++)
                        if (comp(db[j], db[smallest]))
                                smallest = j;
                if (smallest != i)
                        swap(db[i], db[smallest]);
        }
}
// Allows a user interact with a bank account database.
int main() {
        // The simple database of bank accounts
        std::vector<Account> customers;
        // User command
        char cmd;
        // Are we done yet?
        bool done = false;
        do {
                std::cout << "[A]dd [N]ame [I]D [B]alance [Q]uit==> ";
                std::cin >> cmd;
```

```
                switch (cmd) {
                case 'A':
                case 'a':
                        // Add an account
                        add_account(customers);
                        break;
                case 'P':
                case 'p':
                        // Print customer database
                        print_accounts(customers);
                        break;
                case 'N':
                case 'n':
                        // Sort database by name
                        sort(customers, less_than_by_name);
                        print_accounts(customers);
                        break;
                case 'I':
                case 'i':
                        // Sort database by ID (account number)
                        sort(customers, less_than_by_id);
                        print_accounts(customers);
                        break;
                case 'B':
                case 'b':
                        // Sort database by account balance
                        sort(customers, less_than_by_balance);
                        print_accounts(customers);
                        break;
                case 'Q':
                case 'q':
                        done = true;
                        break;
                }
        } while (!done);
}
```

< 실행결과 예시 >

[A]dd [N]ame [I]D [B]alance [Q]uit==> a

Enter name, account number, and account balance: A 34 100.34

[A]dd [N]ame [I]D [B]alance [Q]uit==> a

Enter name, account number, and account balance: B 10 1323.00

[A]dd [N]ame [I]D [B]alance [Q]uit==> a

Enter name, account number, and account balance: C 88 55.05

[A]dd [N]ame [I]D [B]alance [Q]uit==> a

Enter name, account number, and account balance: D 33 423.50

[A]dd [N]ame [I]D [B]alance [Q]uit==> a

Enter name, account number, and account balance: E 11 7.27

[A]dd [N]ame [I]D [B]alance [Q]uit==> n

A,34,100.34

B,10,1323

C,88,55.05

D,33,423.5

E,11,7.27

[A]dd [N]ame [I]D [B]alance [Q]uit==> i

B,10,1323

E,11,7.27

D,33,423.5

A,34,100.34

C,88,55.05

[A]dd [N]ame [I]D [B]alance [Q]uit==> b

E,11,7.27

C,88,55.05

A,34,100.34

D,33,423.5

B,10,1323

[A]dd [N]ame [I]D [B]alance [Q]uit==> q

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 33136개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

이 예시는 Instance Variables 즉, 사용자 정의 class의 예시를 보여준다. className이 Account인 class를 정의해준다. Public: 을 표기해주는 이유는 이 class 안의 외부에서도 참조할 수 있도록 해주는거다. 안의 멤버변수로 string형 객체인 name, int형 변수 id, double형 변수 balance를 선언해준다.

파라미터로 Account class의 요소로 구성된 벡터를 받는 함수 add_account이다. 여기서는 name과 number와 amount를 cin을 통해 입력받고 Account class형의 method를 이용해서 Account의 정보를 넣어준다. (name, number, amount) 이후 이 Account형 acct를 파라미터로 받은 벡터에 push_back해서 저장해준다. 계좌를 등록하는 과정이다.

같은 파라미터를 받아 실행하는 함수 print_accounts가 있다. 이 벡터의 사이즈만큼 안에 벡터의 한 요소 안에 있는 Account class형의 method를 이용해서 그 class의 정보를 하나씩 출력한다. 따라서, 벡터 안에 들어있는 모든 요소 안에 들어있는 Account class형에 들어있는 모든 3개의 정보를 출력해주는 것이다.

Swap 함수는 Account class로 정의된 er1, er2를 레퍼런스 파라미터로 받고 두 변수를 교체한다. Less_than_by_name 함수도 Account class로 정의된 두 레퍼런스 파라미터를 받고 e1의 name member 값이 더 작은 경우 True가 반환된다. 마찬가지로, less_than_by_id도 마찬가지이고 less_than_by_balance의 경우도 앞의 함수와 같은 파라미터를 받고서 각각 e1의 id, balance가 작은 경우 true를 반환한다. 이 함수들은 이후 sort 함수의 파라미터 함수로 이용될 것이다.

sort함수는 Account class의 요소로 구성된 벡터와, boolean형으로 return되는 함수를 파라미터로 받는다. 이 벡터의 사이즈 길이 내에서 오름차순 정렬하는 코드이며 이전 실습에서 설명하였기에 sort 관련 코드 내용은 생략한다. 대신, 이 정렬 반복문 안에서 필요한 상황에 따라 파라미터로 받은 함수를 다르게 집어넣어 Account class로 정의된 요소의 어떤 멤버를 비교할 지 정할 수 있다. Name을 오름차순으로 정렬할 경우, less_than_by_name을 파라미터 함수로 넣고, 그 외에도 마찬가지이다. 이런식으로 앞서 정의한 swap 함수를 이용하여 Account class로 정의된 요소의 어떤 멤버를 기준으로 벡터를 오름차순으로 정렬해준다.

Main 함수를 살펴보면, customers라는 이름을 가진 Account class형 벡터를 선언해주고 cmd는 명령어를 인식하기 위해 문자형으로 선언한다. Done은 boolean형으로 반복분 탈출을 지시할 것이다. Do while문 내에서 cout에서 보여줬듯이 A,N,I,B,Q의 명령어를 cin으로 받을 것이다. 그럼 그에 따른 지시를 수행할 것이다. 당연히 switch문을 사용하여 명령어에 따른 지시를 수행하며, A나 a의 경우 add_account 함수를 실행, P나 p의 경우 print_accounts 함수를 실행, N이나 n의 경우 sort 함수에 member name을 기준으로 오름차순 하는 sort함수와 print_acoount함수를 차례로 실행, I나 i의 경우 함수에 member id를 기준으로 오름차순하는 sort함수와 print_account함수를 차례로 실행, B나 b의 경우 이전과 같은 상황에서 member balance를 기준으로 실행한다. Q나 q의 경우 Boolean done을 true로 바꿔주어 while문을 탈출하여 프로그램을 종료시킨다.

B. Exercises (Write the questions down on your answer sheet)

(pp. 430-435), Exercises 1-10
(write output analysis for all exercises)

1. Given the definition of the SimpleRational number class in Listing 14.5 (simplerational.cpp), complete the function named add:

SimpleRational add(SimpleRational r1, SimpleRational r2) { // Details go here }

that returns the rational number representing the sum of its two parameters. Recall that adding fractions is more involved than multiplying them; you must find a common denominator, adjust the fractions so they both have the same denominator, and then add just the numerators.

< 코드 >

```
SimpleRational add(SimpleRational r1, SimpleRational r2) {
        int a = r1.get_denominator();
        int b = r2.get_denominator();
        while (b != 0) {
                int r = a % b;
                a = b;
                b = r;
        }
        int c = r1.get_denominator() * r2.get_denominator() / a;
        return { (r1.get_numerator() * r2.get_denominator() / a) + (r2.get_numerator() *
r1.get_denominator() / a),
                c };
        }
```

< 전체 코드 >

```
#include <iostream>
#include <cstdlib>
// Models a mathematical rational number
class SimpleRational {
        int numerator;
        int denominator;
public:
        // Initializes the components of a Rational object
        SimpleRational(int n, int d) : numerator(n), denominator(d) {
                if (d == 0) {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // The default constructor makes a zero rational number
        // 0/1
        SimpleRational() : numerator(0), denominator(1) {}
        // Allows a client to reassign the numerator
        void set_numerator(int n) {
```

```
                numerator = n;
        }
        // Allows a client to reassign the denominator.
        // Disallows an illegal fraction (zero denominator).
        void set_denominator(int d) {
                if (d != 0)
                        denominator = d;
                else {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // Allows a client to see the numerator's value.
        int get_numerator() {
                return numerator;
        }
        // Allows a client to see the denominator's value.
        int get_denominator() {
                return denominator;
        }
};
// Returns the product of two rational numbers
SimpleRational multiply(SimpleRational f1, SimpleRational f2) {
        return { f1.get_numerator() * f2.get_numerator(),
        f1.get_denominator() * f2.get_denominator() };
}
SimpleRational add(SimpleRational r1, SimpleRational r2) {
        int a = r1.get_denominator();
        int b = r2.get_denominator();
        while (b != 0) {
                int r = a % b;
                a = b;
                b = r;
        }
        int c = r1.get_denominator() * r2.get_denominator() / a;
        return { (r1.get_numerator() * r2.get_denominator() / a) + (r2.get_numerator() *
r1.get_denominator() / a),
                c };
        }

void print_fraction(SimpleRational f) {
        std::cout << f.get_numerator() << "/" << f.get_denominator();
}
int main() {
        SimpleRational fract(1, 2); // The fraction 1/2
        std::cout << "The fraction is ";
        print_fraction(fract);
        std::cout << '₩n';
        fract.set_numerator(19);
        fract.set_denominator(4);
        std::cout << "The fraction now is ";
        print_fraction(fract);
        std::cout << '₩n';
        // Alternate syntax uses {} with constructor instead of ()
```

```
        SimpleRational fract1{ 1, 4 }, fract2{ 1, 6 };
        auto prod = add(fract1, fract2);
        std::cout << "The sum of ";
        print_fraction(fract1);
        std::cout << " and ";
        print_fraction(fract2);
        std::cout << " is ";
        print_fraction(prod);
        std::cout << '₩n';
}
```

< 실행결과 >

The fraction is 1/2

The fraction now is 19/4

The sum of 1/4 and 1/6 is 5/12

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 66640개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

만든 add 함수를 test해보기위해서 주어진 코드에서 main함수를 product에서 add로 수정하였다. 결과가 잘 나오는 것을 확인할 수 있다. 이전에 배운 최대공약수를 구하는 코드를 통해 최대공배수를 구하고, 최대공배수를 분모로 지정해주고 각 분수에서 그에 따른 분자의 값을 조정해주는 과정을 거쳐서 return해주었다.

2. Given the definition of the geometric Point class in Listing 15.6 (point.cpp), complete the function named distance:

double distance(Point r1, Point r2) { // Details go here }

that returns the distance between the two points passed as parameters.

< 코드 >

```
double distance(const Point& pt1, const Point& pt2) {
        double a = pt1.get_x() - pt2.get_x();
```

```
        double b = pt1.get_y() - pt2.get_y();
        double c = (a * a) + (b * b);
        return sqrt(c);
}
```

< 전체 코드 >

```
#include <iostream>
#include <cmath>
class Point {
        double x;
        double y;
public:
        Point(double x, double y) : x(x), y(y) {}
        double get_x() const { return x; }
        double get_y() const { return y; }
        void set_x(double x) { this->x = x; }
        void set_y(double y) { this->y = y; }
};

double distance(const Point& pt1, const Point& pt2) {
        double a = pt1.get_x() - pt2.get_x();
        double b = pt1.get_y() - pt2.get_y();
        double c = (a * a) + (b * b);
        return sqrt(c);
}

int main() {
        Point p1(3.0, 4.0), p2(0.0, 0.0);
        std::cout << distance(p1, p2) << '₩n';
}
```

< 실행결과 >

```
5



C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 72040개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...
```

< 설명 >

두 Point class 내의 member 함수를 이용하여 x와 y의 값을 가져와 그것을 이용하여 두 점 사이의 거리를 구하는 공식을 <cmath> 라이브러리의 sqrt를 이용하여 구현하였다. 간략하게 설명하자면 x값의 차의 제곱과 y값의 차의 제곱을 더한 뒤 제곱근 해줬다. 테스트 코드 구현결과 원하는 값이 나오는 것을 확인할 수 있다.

3. What is the purpose of a class constructor?

: 생성자는 class의 member variable을 적절한 default값으로 설정하여 초기화하거나 class를 사용하는데 파일 열기와 같은 필요한 설정이 필요한 경우 사용하는 것이 목적이다. 예를 들어 A라는 class를 A()라는 식으로 해도 A class 안에 정의되어있는 member variable 모두 어떤 값으로 초기화될 수 있도록 설정할 수 있다는 것이다. 특히 멤버 변수를 private로 선언했을 때 중요하기도 하다. 누가 객체를 만들 수 있는지 결정하며 객체를 초기화할 수 있다.

4. May a class constructor be overloaded?

: 생성자는 오버로딩이 가능하다. 따라서, parameter를 받은 경우, 안 받은 경우 따로 설정이 가능하다.

5. Given the definition of the SimpleRational number class in Section 14.3, complete the following method named reduce:

```
class SimpleRational {
        // Other details omitted here ...
        // Returns an object of the same value reduced
        // to lowest terms
        SimpleRational reduce() {
                // Details go here
        }
};
```

that returns the rational number that represents the object reduced to lowest terms; for example, the fraction 10/20 would be reduced to 1/2.

- Section 14.3에는 해당 내용이 없어 Section 14.5로 진행함.

< 코드 >

```
class SimpleRational {
        // Other details omitted here ...
        // Returns an object of the same value reduced
        // to lowest terms
        SimpleRational reduce() {
                int a = numerator;
                int b = denominator;
                while (b != 0) {
                        int r = a % b;
                        a = b;
                        b = r;
                }
```

```
                numerator /= a;
                denominator /= a;
                return { numerator , denominator };
        }
};
```

< 전체 코드 >

```
#include <iostream>
#include <cstdlib>
// Models a mathematical rational number
class SimpleRational {
        int numerator;
        int denominator;
public:
        // Initializes the components of a Rational object
        SimpleRational(int n, int d) : numerator(n), denominator(d) {
                if (d == 0) {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // The default constructor makes a zero rational number
        // 0/1
        SimpleRational() : numerator(0), denominator(1) {}
        // Allows a client to reassign the numerator
        void set_numerator(int n) {
                numerator = n;
        }
        // Allows a client to reassign the denominator.
        // Disallows an illegal fraction (zero denominator).
        void set_denominator(int d) {
                if (d != 0)
                        denominator = d;
                else {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // Allows a client to see the numerator's value.
        int get_numerator() {
                return numerator;
        }
        // Allows a client to see the denominator's value.
        int get_denominator() {
                return denominator;
        }
        SimpleRational reduce() {
                int a = numerator;
                int b = denominator;
                while (b != 0) {
                        int r = a % b;
                        a = b;
                        b = r;
```

```
                }
                numerator /= a;
                denominator /= a;
                return { numerator , denominator };
        }
};
void print_fraction(SimpleRational f) {
        std::cout << f.get_numerator() << "/" << f.get_denominator();
}
int main() {
        SimpleRational fract(1, 2); // The fraction 1/2
        std::cout << "The fraction is ";
        print_fraction(fract);
        std::cout << '₩n';
        fract.set_numerator(19);
        fract.set_denominator(4);
        std::cout << "The fraction now is ";
        print_fraction(fract);
        std::cout << '₩n';
        // Alternate syntax uses {} with constructor instead of ()
        SimpleRational fract1{ 10, 100 }, fract2{ 2, 6 };
        print_fraction(fract1);
        print_fraction(fract2);
        std::cout << '₩n';
        fract1.reduce();
        fract2.reduce();
        print_fraction(fract1);
        print_fraction(fract2);
}
```

< 실행결과 예시 >

The fraction is 1/2

The fraction now is 19/4

10/1002/6

1/101/3

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 78172개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

앞선 문제에서 분수의 덧셈을 했던 것처럼 분자와 분모의 최대공약수를 찾아서 그 값으로 나눠주

는 것이다. 그렇게되면 약분이 된다. 위 코드에서 a가 while문을 통해 최대공약수가 된다. 그리고 나눠준 값을 새로운 멤버값으로 넣어주면 된다. 그 이후 여기서 사용한 class형으로 return해주면 된다.

6. Given the definition of the SimpleRational number class in Section 14.3, complete the following free function named reduce:

// Returns a fraction to lowest terms SimpleRational reduce(SimpleRational f) { // Details go here }

that returns the rational number that represents the object reduced to lowest terms; for example, the fraction 10/20 would be reduced to 1/2.

- Section 14.3에는 해당 내용이 없어 Section 14.5로 진행함.

< 코드 >

```
SimpleRational reduce(SimpleRational f) {
        int a = f.get_numerator();
        int b = f.get_denominator();
        while (b != 0) {
                int r = a % b;
                a = b;
                b = r;
        }
        return { f.get_numerator() / a, f.get_denominator() / a };
}
```

< 전체 코드 >

```
#include <iostream>
#include <cstdlib>
// Models a mathematical rational number
class SimpleRational {
        int numerator;
        int denominator;
public:
        // Initializes the components of a Rational object
        SimpleRational(int n, int d) : numerator(n), denominator(d) {
                if (d == 0) {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // The default constructor makes a zero rational number
        // 0/1
        SimpleRational() : numerator(0), denominator(1) {}
        // Allows a client to reassign the numerator
        void set_numerator(int n) {
                numerator = n;
```

```
        }
        // Allows a client to reassign the denominator.
        // Disallows an illegal fraction (zero denominator).
        void set_denominator(int d) {
                if (d != 0)
                        denominator = d;
                else {
                        // Display error message
                        std::cout << "Zero denominator error₩n";
                        exit(1); // Exit the program
                }
        }
        // Allows a client to see the numerator's value.
        int get_numerator() {
                return numerator;
        }
        // Allows a client to see the denominator's value.
        int get_denominator() {
                return denominator;
        }
        void reduce() {
                int a = numerator;
                int b = denominator;
                while (b != 0) {
                        int r = a % b;
                        a = b;
                        b = r;
                }
                numerator /= a;
                denominator /= a;
        }
};
// Returns a fraction to lowest terms
SimpleRational reduce(SimpleRational f) {
        int a = f.get_numerator();
        int b = f.get_denominator();
        while (b != 0) {
                int r = a % b;
                a = b;
                b = r;
        }
        return { f.get_numerator() / a, f.get_denominator() / a };
}

void print_fraction(SimpleRational f) {
        std::cout << f.get_numerator() << "/" << f.get_denominator();
}
int main() {
        SimpleRational fract(1, 2); // The fraction 1/2
        std::cout << "The fraction is ";
        print_fraction(fract);
        std::cout << '₩n';
        fract.set_numerator(19);
        fract.set_denominator(4);
        std::cout << "The fraction now is ";
```

```
        print_fraction(fract);
        std::cout << '₩n';
        // Alternate syntax uses {} with constructor instead of ()
        SimpleRational fract1{ 10, 100 }, fract2{ 2, 6 };
        print_fraction(fract1);
        print_fraction(fract2);
        std::cout << '₩n';
        auto prod1 = reduce(fract1);
        auto prod2 = reduce(fract2);
        print_fraction(prod1);
        print_fraction(prod2);
}
```

< 실행결과 예시 >

The fraction is 1/2

The fraction now is 19/4

10/1002/6

1/101/3

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 84408개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

5번 문제와 같은 기능을 수행하지만 함수를 class 내에서 선언해주는 것이 아니라 밖에서 선언해주었다. return 형식만 살짝 수정해주고 테스트 코드만 수정해주면 되었다. 우선 return 형식이 void가 아니라 SimpleRational이라는 class 객체이므로 return 형식을 그에 맞게 { , } 형태로 변환해주었다. 그리고 parameter로 SimpleRational class 객체를 받아야 되기 때문에 테스트 코드도 fract1과 fract2를 parameter로 넣어주고 이 return 값을 다른 변수인 prod1 과 prod2에 넣고 이를 출력해주었다. 따라서 실행결과와 같이 약분된 결과가 나온 것을 확인할 수 있다. 약분 방법은 5번 문제와 같기 때문에 생략한다.

7. Given the definition of the geometric Point class in Section 14.2 add a method named distance:

```
class Point {
        // Other details omitted
        // Returns the distance from this point to the
        // parameter p
        double distance(Point p) {
                // Details go here
```

```
        }
};
```

that returns the distance between the point on whose behalf the method is called and the parameter p.

< 코드 >

```
class Point {
// Other details omitted
        // Returns the distance from this point to the
        // parameter p
        double distance(Point p) {
                double a = x - p.x;
                double b = y - p.y;
                double c = (a * a) + (b * b);
                return sqrt(c);
        }
};
```

< 전체 코드 >

```
#include <iostream>
// The Point class defines the structure of software
// objects that model mathematical, geometric points
class Point {
public:
        double x; // The point's x coordinate
        double y; // The point's y coordinate
        double distance(Point p) {
                double a = x - p.x;
                double b = y - p.y;
                double c = (a * a) + (b * b);
                return sqrt(c);
        }
};
int main() {
        // Declare some point objects
        Point pt1, pt2;
        // Assign their x and y fields
        pt1.x = 1.0; // Use the dot notation to get to a part of the object
        pt1.y = 3.0;
        pt2.x = -2.0;
        pt2.y = -1.0;
        // Print them
        std::cout << "pt1 = (" << pt1.x << "," << pt1.y << ")₩n";
        std::cout << "pt2 = (" << pt2.x << "," << pt2.y << ")₩n";
        std::cout << "distance = " << pt2.distance(pt1);
}
```

< 실행결과 예시 >

pt1 = (1,3)

pt2 = (-2,-1)

distance = 5

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 89972개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

class 내에서 함수를 정의해주었기 때문에 parameter도 Point class 객체를 하나 받는다. 그래서 method로 사용할 수 있는데 지정한 객체와 parameter에 들어있는 class 객체를 이용하여 서로 계산하는 것이다. 그래서 테스트 코드를 보면 pt2.distance(pt1); 으로 표기가 되어있는 것이다. pt2는 지정한 객체, pt1은 parameter에 있는 객체가 되는 것이다. 따라서, 함수에서 봤을 때 pt2에 들어있는 x값이 x, pt1에 들어있는 x값이 p.x로 역할을 해서 계산된 뒤에 거리가 return 되는 것이다. 두 점 사이의 거리를 구하는 것을 구현한 것은 2번문제와 동일하므로 생략한다.

8. Consider the following C++ code:

```
#include <iostream>
class IntPoint {
public:
        int x;
        int y;
        IntPoint(int x, int y) : x(x), y(y) {}
};
class Rectangle {
        IntPoint corner; // Location of the rectangle's lower-left corner
        int width; // The rectangle's width
        int height; // The rectangle's width
public:
        Rectangle(IntPoint pt, int w, int h) :
                corner((pt.x < -100) ? -100 : (pt.x > 100 ? 100 : pt.x),
                        (pt.y < -100) ? -100 : (pt.y > 100 ? 100 : pt.y)),
                width((w < 0) ? 0 : w), height((h < 0) ? 0 : h) {}
        int perimeter() {
                return 2 * width + 2 * height;
        }
        int area() {
                return width * height;
        }
        int get_width() {
                return width;
        }
        int get_height() {
```

```
                return height;
        }
        // Returns true if rectangle r overlaps this
        // rectangle object.
        bool intersect(Rectangle r) {
                // Details omitted
        }
        // Returns the length of a diagonal rounded to the nearest
        // integer.
        int diagonal() {
                // Details omitted
        }
        // Returns the geometric center of the rectangle with
        // the (x,y) coordinates rounded to the nearest integer.
        IntPoint center() {
                // Details omitted
        }
        bool is_inside(IntPoint pt) {
                // Details omitted
        }
};
int main() {
        Rectangle rect1(IntPoint(2, 3), 5, 7),
                rect2(IntPoint(2, 3), 1, 3),
                rect3(IntPoint(2, 3), 15, 3),
                rect4(IntPoint(2, 3), 5, 3);
        std::cout << rect1.get_width() << '₩n';
        std::cout << rect1.get_height() << '₩n';
        std::cout << rect2.get_width() << '₩n';
        std::cout << rect2.get_height() << '₩n';
        std::cout << rect3.get_width() << '₩n';
        std::cout << rect3.get_height() << '₩n';
        std::cout << rect4.get_width() << '₩n';
        std::cout << rect4.get_height() << '₩n';
        std::cout << rect1.perimeter() << '₩n';
        std::cout << rect1.area() << '₩n';
        std::cout << rect2.perimeter() << '₩n';
        std::cout << rect2.area() << '₩n';
        std::cout << rect3.perimeter() << '₩n';
        std::cout << rect3.area() << '₩n';
        std::cout << rect4.perimeter() << '₩n';
        std::cout << rect4.area() << '₩n';
}
```

- 코드에 문제가 있어 main 함수를 수정함. get_perimeter -> perimeter, get_area -> area

(a) What does the program print?

5

7

1

3

15

3

5

3

24

35

8

3

36

45

16

15

C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 18788개)이(가

) 종료되었습니다(코드: 0개).

이 창을 닫으려면 아무 키나 누르세요...

(b) With regard to a Rectangle object's lower-left corner, what are the minimum and maximum values allowed for the x coordinate? What are the minimum and maximum values allowed for the y coordinate?

: lower-left corner의 x좌표로 가능한 최소값은 -100이며, 최대값은 100이다. y좌표로 가능한 최소값은 -100이며, 최대값은 100이다. Rectangle 생성자 코드에 잘 적혀있다.

(c) What is a Rectangle object's minimum and maximum width?

: width로 가능한 최소값은 0이고, 최대값은 정해져있지 않아 int형의 최대범위인 2147483647일 것이다.

(d) What is a Rectangle object's minimum and maximum height?

: height로 가능한 최소값은 0이고, 최대값은 정해져있지 않아 int형의 최대범위인 2147483647일 것이다.

(e) What happens when a client attempts to create a Rectangle object with parameters that are outside the acceptable ranges?

: 간략하게 작성되어있는 if문 (a ? b : c)에 의해서 자동으로 값이 조절된다. x와 y 둘 다 -100보다 작은 값을 받은 경우 -100, 100보다 큰 값을 받은 경우 100으로 자동 설정된다. 따라서 width와 height도 0보다 작은 경우, 자동으로 0으로 설정된다.

(f) Implement the diagonal method.

```
int diagonal() {
		return sqrt((width * width) + (height * height));
	}
```

(g) Implement the center method

```
	// Returns the geometric center of the rectangle with
	// the (x,y) coordinates rounded to the nearest integer.
	IntPoint center() {
		return { (width / 2) + corner.x , (height / 2) + corner.y };
	}
```

(h) Implement the intersect method.

```
// Returns true if rectangle r overlaps this
	// rectangle object.
	bool intersect(Rectangle r) {
		if ((corner.x >= r.corner.x) && (corner.x <= r.corner.x + r.width) &&
(corner.y >= r.corner.y) && (corner.y <= r.corner.y + r.height))
			return true;
		else if ((corner.x + width >= r.corner.x) && (corner.x + width <= r.corner.x +
r.width) && (corner.y >= r.corner.y) && (corner.y <= r.corner.y + r.height))
			return true;
		else if ((corner.x >= r.corner.x) && (corner.x <= r.corner.x + r.width) &&
(corner.y + height >= r.corner.y) && (corner.y + height <= r.corner.y + r.height))
```

```
                return true;
            else if ((corner.x + width >= r.corner.x) && (corner.x + width <= r.corner.x +
r.width) && (corner.y + height >= r.corner.y) && (corner.y + height <= r.corner.y + r.height))
                return true;
            else return false;
```

}: 두 사각형이 한 점에서 만나는 경우는 문제에서 따로 정의해준게 없으므로 겹치는 것으로 판단하고 설정하여 true로 나타나게 하였다. 두 사각형 중 하나를 기준으로 삼아 사용하는 x의 좌표의 범위와 y의 좌표의 범위를 생각하고 다른 사각형의 4개의 점이 이 범위 내에 존재하는 경우 두 사각형이 겹치는 걸로 판단하여 true를 return 하게 구성하였다.

(i) Implement the is_inside method.

```
bool is_inside(IntPoint pt) {
            if ((pt.x >= corner.x) && (pt.x <= corner.x + width) && (pt.y >= corner.y) &&
(pt.y <= corner.y + height))
                return true;
            else return false;
        }
```

: 사각형이 사용하는 x의 좌표의 범위와 y의 좌표의 범위를 생각하고 파라미터로 받은 pt의 x좌표와 y좌표가 이 범위 안에 속하는지 if문을 이용하여 체크한 뒤 그럴 경우에 true를 return하고, 그렇지 않으면 false를 return 한다.

(j) Complete the following function named corner:

IntPoint corner(Rectangle r) { // Details go here };

that returns the lower-left corner of the Rectangle object r passed to it. You may not modify the Rectangle class.

< 코드 >

```
IntPoint corner(Rectangle r) {
        return { (r.center().x - (r.get_width() / 2)), (r.center().y - (r.get_height() /
2)) };
};
```

< 설명 >

Rectangle class에서 corner, width, height이 private: 내에 선언되어있어서 직접 참조하는 것이 불가능하다. 따라서 아래 public: 에 있는 함수 method를 이용해서 객체를 불러와야한다. width와 height를 불러올 수 있는 함수가 만들어져있고, corner를 이용한 함수를 살펴보니 center()가 있었다. 따라서 그 두 함수를 이용하여 corner.x, corner.y의 값이 나오도록 설정하였다.

----- 8번 문항 채점하시기 편하시도록 테스트 문항까지 아래 첨부합니다. (전체 코드) ------

```
#include <iostream>
class IntPoint {
public:
        int x;
        int y;
        IntPoint(int x, int y) : x(x), y(y) {}
};
class Rectangle {
        IntPoint corner; // Location of the rectangle's lower-left corner
        int width; // The rectangle's width
        int height; // The rectangle's width
public:
        Rectangle(IntPoint pt, int w, int h) :
                corner((pt.x < -100) ? -100 : (pt.x > 100 ? 100 : pt.x),
                        (pt.y < -100) ? -100 : (pt.y > 100 ? 100 : pt.y)),
                width((w < 0) ? 0 : w), height((h < 0) ? 0 : h) {}
        int perimeter() {
                return 2 * width + 2 * height;
        }
        int area() {
                return width * height;
        }
        int get_width() {
                return width;
        }
        int get_height() {
                return height;
        }
        // Returns true if rectangle r overlaps this
        // rectangle object.
        bool intersect(Rectangle r) {
                if ((corner.x >= r.corner.x) && (corner.x <= r.corner.x + r.width) &&
(corner.y >= r.corner.y) && (corner.y <= r.corner.y + r.height))
                        return true;
                else if ((corner.x + width >= r.corner.x) && (corner.x + width <= r.corner.x +
r.width) && (corner.y >= r.corner.y) && (corner.y <= r.corner.y + r.height))
                        return true;
                else if ((corner.x >= r.corner.x) && (corner.x <= r.corner.x + r.width) &&
(corner.y + height >= r.corner.y) && (corner.y + height <= r.corner.y + r.height))
                        return true;
                else if ((corner.x + width >= r.corner.x) && (corner.x + width <= r.corner.x +
r.width) && (corner.y + height >= r.corner.y) && (corner.y + height <= r.corner.y + r.height))
                        return true;
                else return false;
        }
        // Returns the length of a diagonal rounded to the nearest
        // integer.
        int diagonal() {
                return sqrt((width * width) + (height * height));
        }
        // Returns the geometric center of the rectangle with
        // the (x,y) coordinates rounded to the nearest integer.
        IntPoint center() {
```

```
                return { (width / 2) + corner.x , (height / 2) + corner.y };
        }
        bool is_inside(IntPoint pt) {
                if ((pt.x >= corner.x) && (pt.x <= corner.x + width) && (pt.y >= corner.y) &&
(pt.y <= corner.y + height))
                        return true;
                else return false;
        }
};
IntPoint corner(Rectangle r) {
        return { (r.center().x - (r.get_width() / 2)), (r.center().y - (r.get_height() /
2)) };
};
int main() {
        Rectangle rect1(IntPoint(2, 3), 5, 7),
                rect2(IntPoint(2, 1), 2, 2),
                rect3(IntPoint(0, 6), 10, 3),
                rect4(IntPoint(0, 0), 2, 3);
        std::cout << rect4.diagonal() << '₩n';
        auto prod = rect3.center();
        std::cout << prod.x << prod.y << '₩n';
        std::cout << rect4.is_inside(prod) << '₩n';
        std::cout << rect4.intersect(rect2) << '₩n';
        std::cout << corner(rect3).x << corner(rect3).y << '₩n';
}
```

9. Develop a Circle class that, like the Rectangle class above, provides methods to compute perimeter and area. The Rectangle instance variables are not appropriate for circles; specifically, circles do have corners, and there is no need to specify a width and height. A center point and a radius more naturally describe a circle. Build your Circle class appropriately.

< 코드 >

```
class IntPoint {
public:
        int x;
        int y;
        IntPoint(int x, int y) : x(x), y(y) {}
};
class Circle {
        IntPoint center; // Location of the rectangle's lower-left corner
        double radius ; // The rectangle's width
        double pi = 3.141592;
public:
        Circle(IntPoint pt, double r) : center(pt.x,pt.y), radius(r) {}
        double perimeter() {
                return 2 * pi * radius;
        }
        double area() {
                return pi * radius * radius;
        }
```

```
};
```

< 전체 코드 >

```
#include <iostream>

class IntPoint {
public:
	int x;
	int y;
	IntPoint(int x, int y) : x(x), y(y) {}
};
class Circle {
	IntPoint center;
	double radius ;
double pi = 3.141592;
public:
	Circle(IntPoint pt, double r) : center(pt.x,pt.y), radius(r) {}
	double perimeter() {
		return 2 * pi * radius;
	}
	double area() {
		return pi * radius * radius;
	}
};

int main() {
	Circle cir1(IntPoint(0, 0), 1),
		cir2(IntPoint(2, 1), 2),
		cir3(IntPoint(0, 6), 3),
		cir4(IntPoint(0, 0), 4);
	std::cout << cir1.perimeter() << std::endl;
	std::cout << cir1.area() << std::endl;
	std::cout << cir2.perimeter() << std::endl;
	std::cout << cir2.area() << std::endl;
}
```

< 실행결과 >

```
6.28318

3.14159

12.5664

12.5664



C:₩Users₩Administrator₩source₩repos₩Ex002₩Debug₩Ex002.exe(프로세스 146236개)이(

가) 종료되었습니다(코드: 0개).
```

이 창을 닫으려면 아무 키나 누르세요...

< 설명 >

원의 중심점을 받기 위해 위에서 사용한 IntPoint class를 그대로 가져왔다. Circle class에서 중심점을 IntPoint class 형으로 받고 반지름을 double형, pi를 double형으로 선언해주었다. 생성자로 선언해주고 둘레와 넓이를 return하는 method도 수학 공식에 따라 만들었다. 테스트 함수는 main 함수 안에 만들었다. 정상적으로 코드가 수행된다.

10. Given the Rectangle and Circle classes from questions above, write an encloses function:

// Returns true if rectangle rect is large enough to // completely enclose circle circ bool encloses(Rectangle rect, Circle circ) { // Details omitted }

so that it returns true if circle circ's dimensions would allow it to fit completely within rectangle rect. If circ is too big, the function returns false. The positions of rect and circ do not influence the result.

< 코드 >

```
bool encloses(Rectangle rect, Circle circ) {
        int width = rect.get_width();
        int height = rect.get_height();
        int radius = circ.get_radius();
        IntPoint center((width / 2) + rect.get_corner().x, (height / 2) +
rect.get_corner().y);
        if (width == height && (width / 2) == radius)
                if (center.x == circ.get_center().x && center.y == circ.get_center().y)
                        return true;
        return false;
}
```

< 전체코드 >

```
#include <iostream>

class IntPoint {
public:
        int x;
        int y;
        IntPoint(int x, int y) : x(x), y(y) {}
};
class Circle {
        IntPoint center; // Location of the rectangle's lower-left corner
        double radius ; // The rectangle's width
        double pi = 3.141592;
public:
        Circle(IntPoint pt, double r) : center(pt.x,pt.y), radius(r) {}
```

```
        double perimeter() {
                return 2 * pi * radius;
        }
        double area() {
                return pi * radius * radius;
        }
        double get_radius() {
                return radius;
        }
        IntPoint get_center() {
                return { center.x, center.y };
        }
};
class Rectangle {
        IntPoint corner; // Location of the rectangle's lower-left corner
        int width; // The rectangle's width
        int height; // The rectangle's width
public:
        Rectangle(IntPoint pt, int w, int h) :
                corner((pt.x < -100) ? -100 : (pt.x > 100 ? 100 : pt.x),
                        (pt.y < -100) ? -100 : (pt.y > 100 ? 100 : pt.y)),
                width((w < 0) ? 0 : w), height((h < 0) ? 0 : h) {}
        int perimeter() {
                return 2 * width + 2 * height;
        }
        int area() {
                return width * height;
        }
        int get_width() {
                return width;
        }
        int get_height() {
                return height;
        }
        IntPoint get_corner() {
                return { corner.x, corner.y };
        }
};
// Returns true if rectangle rect is large enough to
// completely enclose circle circ
bool encloses(Rectangle rect, Circle circ) {
        int width = rect.get_width();
        int height = rect.get_height();
        int radius = circ.get_radius();
        IntPoint center((width / 2) + rect.get_corner().x, (height / 2) +
rect.get_corner().y);
        if (width == height && (width / 2) == radius)
                if (center.x == circ.get_center().x && center.y == circ.get_center().y)
                        return true;
        return false;
}
int main() {
        Circle cir1(IntPoint(0, 0), 1),
                cir2(IntPoint(2, 1), 2);
        Rectangle Rect1(IntPoint(0, 0), 2, 2),
```

```
                Rect2(IntPoint(0, -1), 4, 4);
        std::cout << cir1.perimeter() << std::endl;
        std::cout << cir1.area() << std::endl;
        std::cout << cir2.perimeter() << std::endl;
        std::cout << cir2.area() << std::endl;
        std::cout << encloses(Rect1, cir1) << std::endl;
        std::cout << encloses(Rect2, cir2) << std::endl;
}
```

< 설명 >

편의를 위해 Circle class에서 반지름을 가져오는 get_radius 함수와 중심점을 가져오는 get_center 함수를 추가하였고, Rectangle class에서도 corner의 값을 가져오는 get_corner() 함수를 추가하였다. 이전에 정의하였던 Rectangle class 내의 center()함수도 사용할 수 있었지만, int형으로 변환되면서 정확하지 않은 값이 되어 새로 정의해주고 싶었다. 따라서 각 class내의 method를 이용해 필요한 값들을 가져왔다. 사각형의 너비, 높이, 중심점, 원의 반지름을 가져와서 int형 변수로 초기화해주었다. true가 되기 위해 사각형은 반드시 정사각형이어야 하므로 width와 height와 같은 경우, 또한 width의 반절이 원의 반지름과 같은 경우를 우선 처음 걸러주었다. 이후 사각형의 중심점과 원의 중심점이 같은 경우를 찾으면서 true를 return 하도록 하였다. 나머지는 전부 false이다.

C. Additional exercises (Write the questions down on your answer sheet)

C-1. Code explanation, write test codes for a csv file, & output analysis (Write the source code and results), csv example: diabetes.csv (https://www.kaggle.com/saurabh00007/diabetescsv)

```
void ReadCsv(std::string FileName, std::vector<std::vector<std::string>> &Data)
{
        std::ifstream ifs;

        ifs.open(FileName);
        if(!ifs.is_open()) return;

        std::string LineString = "";
        std::string Delimeter = ",";
        while(getline(ifs, LineString))
        {
                std::vector<std::string> RowData;
                unsigned int nPos = 0, nFindPos;
                do {
                        nFindPos = LineString.find(Delimeter, nPos);
                        if(nFindPos == std::string::npos)
                                nFindPos = LineString.length();

                        RowData.push_back(LineString.substr(nPos, nFindPos-nPos));
                        nPos = nFindPos+1;
                } while(nFindPos < LineString.length());
                Data.push_back(RowData);
        }

        ifs.close();
```

```
}
```

< 전체코드 >

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>

void ReadCsv(std::string FileName, std::vector<std::vector<std::string>>& Data)
{
	std::ifstream ifs;

	ifs.open(FileName);
	if (!ifs.is_open()) return;

	std::string LineString = "";
	std::string Delimeter = ",";
	while (getline(ifs, LineString))
	{
		std::vector<std::string> RowData;
		unsigned int nPos = 0, nFindPos;
		do {
			nFindPos = LineString.find(Delimeter, nPos);
			if (nFindPos == std::string::npos)
				nFindPos = LineString.length();

			RowData.push_back(LineString.substr(nPos, nFindPos - nPos));
			nPos = nFindPos + 1;
		} while (nFindPos < LineString.length());
		Data.push_back(RowData);
	}

	ifs.close();
}
void print_vector(const std::vector<std::vector<std::string>>& v)
{
	int len = v.size();
	if (len > 0)
	{
		for (auto x : v)
		{
			for (auto elem : x)
			{
				std::cout << std::setw(24) << elem << ' ';
			}
			std::cout << std::endl;
		}
	}
}
int main() {
	std::string filename = "diabetes.csv";
	std::vector<std::vector<std::string>> vec;
	ReadCsv(filename, vec);
```

```
        print_vector(vec);
}
```

< 설명 >

우선 실행결과는 매우 길기 때문에 생략하겠음.

교수님께서 제공하신 코드는 csv파일의 내용을 읽어와 2차원 string형 벡터에 저장하는 함수이다. ifstream을 이용하여 파일을 열어 ifs라는 ifstream형 변수에 넣어주고 만약 ifs에 파일이 안들어갔다면 아무것도 return 하지 않는다. 이후 string형 변수에 기준으로 삼을 것들을 설정해주고 getline함수와 while문을 이용해 ifs 안에 있는 내용을 LineString을 기준으로 한줄씩 가져온다. 이 한줄에서 쉼표를 기준으로 첫번째 벡터의 벡터에 넣어준다. 그 다음줄은 두번째 벡터의 벡터에 넣어주면서 2차원 벡터에 파일의 내용이 순서대로 들어가는 것이다. 이후 오류를 방지하기 위해 파일을 close()를 이용해 닫아준다.

이후는 본인이 가져온 코드이다. 파일의 내용을 2차원 벡터에 넣었기 때문에 2차원 벡터의 내용을 출력할 함수 print_vector를 만든다. 물론 강의자료를 참고하였다. 따라서 2차원 벡터의 내용을 순서대로 출력해주는 함수이다. 줄을 맞추기 위해 setw 함수를 사용하였다. (첫 행에서 가장 긴 단어를 기준하여 파라미터를 정하였음)

이제 main 함수에서 테스트 코드를 작성해보자 string 타입을 가진 filename에 파일명을 넣어주고 2차원 벡터도 선언해준다. ReadCsv 함수를 통해 filename으로 지정해준 파일의 내용을 선언해준 2차원 벡터에 가져온다. 이후 print_vector 함수를 통해 2차원 벡터의 내용을 출력해준다.