Object-Oriented Programming
Lab #05

Department: 화학공학과 **Student ID: 2019101074** **Name: 안용상**

1. 아래의 코드는 csv (comma-separated values) 파일을 읽어서 vector로 저장하는 함수이다. https://www.kaggle.com/saurabh00007/diabetescsv에서 csv을 다운로드 받아서 동작을 확인하고 (main 함수 작성), 함수의 동작을 설명하라.

```
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();
}
```

- 설명

> 💡 아래 코드는 캐글에서 받은 csv파일을 cpp파일이 있는 디렉토리에 저장했을때 정상 실행됩니다.

```
/*
##########################################
##########################################

ReadCsv는
파일이름을 문자열로 받으며
요소가 문자열인 2차원 벡터를 reference인자로 받는 함수이다

ReadCsv의 동작을 설명한뒤, main함수에서 직접 ReadCsv에 의해 담긴 2D벡터를
출력해서 확인해봤다.

##########################################
##########################################

ReadCsv동작,

파일핸들러를 선언한다.
```

```
파일핸들러에 파일이름을 전달하며 파일을 Open한다

파일이 성공적으로 열리지 않았을 때 함수를 즉시 종료하는 코드가 있다.
이는 오류처리와 불필요한 연산방지를 위한 것이다
파일이 존재하지 않거나, Access권한이 없는경우 오류처리를 할 수있다. 그리고 이때 조기에
종료함으로써 불필요한 연산을 방지한다.

파일핸들러로 조작하고있는 파일을 getline함수로 한줄씩 읽어올 것인데
그 읽은 줄을 담아둘 변수를 LineString = "";으로 초기화하고
한 줄에서 요소들을 구분할 구분자를 delimiter =","; 로 초기화한다
왜냐하면 csv가 comma separated values이기 때문이다.

while문을 돌면서 마지막 줄이 될때까지 getline으로 파일을 한줄한줄 읽어서 LineString에 담는다.
RowData라는 최종 2차원 벡터 Data에 행을 전달하기 위한 행컨테이너를 1차원 문자열벡터로 초기화한다.
그 뒤 아래와 같은 do문이 시작된다
do 문
문자열.find()로 delimieter를 찾을때 탐색의 시작점이 되어줄 nPos를 초기화한다
문자열에서 delimieter를 nPos의 위치부터 찾아 그 찾은 위치를 nFindPos에 담는다
이 때, std::string::npos가 nFindPos와 같은경우 즉, delimeter를 찾지 못한경우에
nFindPos를 읽어온 줄의 총 길이로 할당한다.
nPos부터 nFindPos의 구간 문자열을 잘라내어 RowData라는 1차원 행벡터에 psuh_back한다.
다음 delimeter를 찾기 시작할 위치를 정하기 위해
nPos를 직전에 찾아낸 delimeter의 위치였던 nFindPos에 1을 더한 값으로 할당한다.

do 문이 끝나면 while문으로 증가하는 nFindPos값이 LineString.length()보다
작지않을때까지 do 에 명시된 연산들을 반복한다.

nFindPos값이 LineString.length()값보다 같거나 커진경우에는 안에있는 do while문이 끝난다.
한 줄에 대한 delimeter구분 및 행컨테이너에 요소를 담는 작업을 다 끝냈으니
그 한줄에 대한 행컨테이너를 Data라는 2D벡터에 push_back해준다
위와같은 과정을 바깥의 while문으로 마지막 줄까지 반복한다.
그리고 함수가 끝난다.
reference인자로 받은 2D벡터 Data이므로 실제 Data로 넘겨준 벡터가 변경된다
*/
#include <iostream>
#include <stdio.h>
#include <vector>
#include <fstream>
#include <string>

void ReadCsv(std::string FileName, std::vector<std::vector<std::string>>& Data) {
  std::ifstream ifs; // 파일핸들러를 ifs로 초기화
  ifs.open(FileName); // 파일핸들러 ifs로 파일이름을 받아 연다.
  if (!ifs.is_open()) return; // 만약 열리지않을시, 불필요한 연산방지를 위해 조기종료
  std::string LineString = ""; //한줄한줄 읽을때 그 줄의 문자열을 담아둘 변수. 첫 초기화는 ""로.
  std::string Delimeter = ","; // CSV의 구분자를 정의
  while (getline(ifs, LineString)) { // ifs로 연 파일을 한줄씩 읽어 LineString에 담음.
    std::vector<std::string> RowData; // 1차원 벡터 RowData 선언
    unsigned int nPos = 0, nFindPos; // nPos : 구분자를 탐색할 시작위치(맨처음에는 0으로 초기화)
    // nFindPos : 찾아낸 구분자 위치
    do { // while 반복 전에 아래와 같은 코드를 먼저 실행
      nFindPos = LineString.find(Delimeter, nPos); // 구분자를 탐색시작위치부터 탐색해본다.
      if (nFindPos == std::string::npos) nFindPos = LineString.length();
      // 탐색시 구분자 발견에 실패한경우, nFindPos를 그 줄의 총길이로 초기화.
      // ( 이 경우에 바로아래 실행될 while문의 조건에 위배되어 while문 탈출)
      RowData.push_back(LineString.substr(nPos, nFindPos - nPos));
      // 1차원 문자열벡터에 찾아낼 위치 ~ 찾은 위치까지의 문자열을 슬라이싱해서 담아냄
      nPos = nFindPos + 1;
      // 그리고 다시 그 줄에서 그 다음에 나오는 구분자를 찾기위해, 찾기시작할 위치를
      // 찾아낸 위치 + 1로 초기화함.
    } while (nFindPos < LineString.length());// 찾아낸 위치가 그 줄의 문자열 총길이보다 작을때
    // 위의 do문에 써있는 코드를 실행함. 그렇지않으면 while문 탈출
    Data.push_back(RowData);
    // 2D문자열벡터 Data에 1D문자열벡터 RowData를 push_back함.
  }
  ifs.close();//메모리 누수를 방지하기위해 파일 핸들러를 닫고 열린파일테이블에서 해당 파일정보를 제거함.
}

int main() {
  std::vector<std::vector<std::string>> data;
  std::string FileName = "diabetes.csv"; // 파일이름을 프로젝트 폴더안에 저장한
  // 캐글에서 받은 파일이름으로 초기화
  ReadCsv(FileName, data); // ReadCsv에 인자로 FileName과 2D벡터를 전달하고, ReadCsv동작
```

```
  for (const auto& row : data) { //data를 한 행씩 for문을 돌림.
    for (const auto& element : row) { //각 한 행에서 요소들을 for문으로 추출.
      std::cout << element << " "; // 각 요소를 " " 로 구분하면서 출력.
    }
    std::cout << std::endl; // 각 행을 줄바꿈으로 구분하면서 출력을 계속.
  }
}
```

- 출력결과

```
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
6 148 72 35 0 33.6 0.627 50 1
1 85 66 29 0 26.6 0.351 31 0
8 183 64 0 0 23.3 0.672 32 1
1 89 66 23 94 28.1 0.167 21 0
.
..
...(너무 길어서 중략했습니다.)
..
.
10 101 76 48 180 32.9 0.171 63 0
2 122 70 27 0 36.8 0.34 27 0
5 121 72 23 112 26.2 0.245 30 0
1 126 60 0 0 30.1 0.349 47 1
1 93 70 31 0 30.4 0.315 23 0

C:\Users\qhfkd\Project1\x64\Debug\Project1.exe(프로세스 21788개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

---

2. 아래의 코드에서 IntPoint는 2차원 평면상의 점을 표현하는 클래스이며 x, y는 평면상의 좌표를 저장하는 멤버이며, Rectangle은 회전되지 않은 직사각형 표현하는 클래스로 왼쪽-코너점을 corner로, 폭과 높이를 각각 width와 height로 저장한다. 주석의 내용과 같이 동작하도록 코드를 완성하고 동작을 확인할 main 함수를 작성하라.

```
#include <iostream>
class IntPoint {
public:
  int x; // x 좌표
      int y; // y 좌표
  IntPoint(int x, int y): x(x), y(y) {}
};

class Rectangle {
  IntPoint corner; // 직사각형의 왼쪽-아래 코너점
  int width; // 직사각형의 폭
  int height; // 직시각형의 높이
public:
  Rectangle(IntPoint pt, int w, int h): corner(pt),
    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;
  }
  // 현재 인스턴스 사각형과 r이 겹쳐있다면 true, 그렇지 않으며 false
```

```
  bool intersect(Rectangle r) {
    // 코드 작성
  }
  // 대각선의 길이(int 형)를 반환
  int diagonal() {
    // 코드 작성
  }
  // 사각형의 중심점의 좌표를 IntPoint 형으로 반환
  IntPoint center() {
    // 코드 작성
  }
  // 현재 인스턴스 사각형의 내부(경계포함)에 pt가 있으면 true,
    // 그렇지 않으면 false
  bool is_inside(IntPoint pt) {
      // 코드 작성
  }
};
```

정답 작성:

```
#include <iostream>
#include <vector>
class IntPoint {
public:
  int x; // x 좌표
  int y; // y 좌표
  IntPoint(int x, int y) : x(x), y(y) {}
};

class Rectangle {
  IntPoint corner; // 직사각형의 왼쪽-아래 코너점
  int width; // 직사각형의 폭
  int height; // 직시각형의 높이
public:
  Rectangle(IntPoint pt, int w, int h) : corner(pt),
    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;
  }
  // 현재 인스턴스 사각형과 r이 겹쳐있다면 true, 그렇지 않으며 false
  bool intersect(Rectangle r) {
    bool is_left_side_in = (r.corner.x > corner.x) && (r.corner.x < corner.x + width);
    bool is_right_side_in = (r.corner.x + r.width > corner.x) && (r.corner.x + r.width < corner.x + width);
    bool is_top_side_in = (r.corner.y > corner.y) && (r.corner.y < corner.y + height);
    bool is_bottom_side_in = (r.corner.y + r.height > corner.y) && (r.corner.y + r.height < corner.y + height);

    if ((is_left_side_in || is_right_side_in) && (is_top_side_in || is_bottom_side_in)) {
      return true;
    }
    else {
      return false;
    }
  }
  // 대각선의 길이(int 형)를 반환
  int diagonal() {
    return std::sqrt(std::pow(width, 2) + std::pow(height, 2));
```

```
  }
  // 사각형의 중심점의 좌표를 IntPoint 형으로 반환
  IntPoint center() {
    IntPoint ip(corner.x + width / 2, corner.y + height / 2);
    return ip;
  }
  // 현재 인스턴스 사각형의 내부(경계포함)에 pt가 있으면 true,
  // 그렇지 않으면 false
  bool is_inside(IntPoint pt) {
    bool is_in_x = (pt.x <= corner.x + width) && (pt.x >= corner.x);
    bool is_in_y = (pt.y <= corner.y + height) && (pt.y >= corner.y);
    if (is_in_x && is_in_y) {
      return true;
    }
    else {
      return false;
    };
  }
};
void get_plot(std::vector<Rectangle> vec) {
  int count = 1;
  int minx, miny, maxy, maxx;
  for (Rectangle r : vec) {
    if (count == 1) {
      minx = r.get_corner().x;
      maxx = r.get_corner().x + r.get_width();
      miny = r.get_corner().y;
      maxy = r.get_corner().y + r.get_height();
    }
    else {
      minx = std::min(minx, r.get_corner().x);
      miny = std::min(miny, r.get_corner().y);
      maxx = std::min(maxx, r.get_corner().x + r.get_width());
      maxy = std::min(maxy, r.get_corner().y + r.get_height());
    }
  }
  for (int row = miny; row <= maxy; row++) {
    int scope = maxx - minx;
    int* arr = new int[scope];

    delete[] arr;
  }


}
int main() {
  IntPoint p1(10, 11);
  // 시작점이 x = 10, y = 11이며 너비,높이가 5와 7인 직사각형 생성
  Rectangle rec1(p1, 5, 7);

  IntPoint p2(15, 15);
  // 시작점이 x = 15, y = 15이며 너비,높이가 5와 7인 직사각형 생성
  Rectangle rec2(p2, 5, 7);

  IntPoint p3(13, 15);
  // 시작점이 x = 13, y = 15이며 너비,높이가 5와 7인 직사각형 생성
  Rectangle rec3(p3, 5, 7);

  std::cout <<"rec1는 rec2과 겹치는가?(겹치면 1, 안겹치면 0)"<< rec1.intersect(rec2)<< std::endl;
  std::cout << "rec1는 rec3과 겹치는가?(겹치면 1, 안겹치면 0)" << rec1.intersect(rec3) << std::endl;


  std::cout << "rec1의 대각선크기는?" << rec1.diagonal() << std::endl;
  std::cout << "rec1의 중심좌표" << "x : " << rec1.center().x << "y : " << rec1.center().y << std::endl;

  //테스트해볼 포인트를 IntPoint형태로 설정
  IntPoint testPoint(10, 13);
  std::cout << rec1.is_inside(testPoint);
};
```

3. 아래의 main 함수가 주석과 같이 동작하도록 Rational 클래스와 필요한 함수들을 정의하라.

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

int main () {
    Rational r1, r2(5), r3(2, 8), r4;
    Print(r1); // prints 0/1
    Print(r2); // prints 5/1
    Print(r3); // prints 1/4

    r4 = Mul(r2, r3); // r4 = r2*r3
    Print(r4); // prints 5/4
    r4 = r2.Add(r3); // r4 = r2+r3
    Print(r4); // prints 21/4

    if(r4.Equal(Rational{42, 8})) std::cout << "Equal" << std::endl;

    std::vector<Rational> v1;
    v1.push_back({1}); v1.push_back({3, 7});
    Print(v1); // prints 1/1, 3/7

    std::string s1 = "C++ programming", s2;
    s2 = NewString(s1); // s2: "***C++ programming***"
    std::cout << s2 << std::endl; // prints ***C++ programming***
}
```

정답작성

```
#include <iostream>
#include <vector>
#include <string>
/*
### 가장먼저 Ratinal클래스를 정의한다
*/
class Rational {
public: // 외부에서 접근 할 수 있는 클래스 멤버들을 정의한다.
    int first; // 분자를 담을 변수 first를 정의
    int second; // 분모를 담을 변수 second를 정의
    Rational(int x = 0, int y = 1) :first(x), second(y) {
        set_best(); // 생성자를 정의했다
        // 아무것도 생성자로써 인자를 받지못할때를 위해서 default로 분자를 0 분모를 1로 할당한다.
        // 그리고 무언가를 인자로 받으면 첫번째 인자를 분자로 두번째 인자를 분모로 할당한다.
    }
    void set_best() {// 기약분수형태로 만들어주는 set_best함수를 정의
        int new_first = first;
        int new_second = second;
        int smaller = (new_first < new_second) ? new_first : new_second;
        //분모와 분자중 더 작은 것을 smaller에 담는다.
        int divisor = 1; //나눌 최종수를 설정할 divisor변수를 1로 초기화한다
        int flag = 1; //공약수를 구해주는 while문을 멈추는 시점을 결정할 플래그.
        while (flag == 1) { //flag가 1로 설정될때 작동.
            for (int i = 2; i <= smaller; ++i) {
                // 2부터 순회하며 분자 분모 둘다 나머지가 0이되게 나누는 i를 찾는다.
                if (new_first % i == 0 && new_second % i == 0) {
                    // 분자 분모 둘다 나머지가 0이되게 나누는 i를 찾았을 때
                    // divisor에 그 i를 곱해주고, smaller는 i로 나눠준다.
                    divisor *= i;
                    smaller /= i;
                    break;// 그 후 그 smaller로 while문을 다시 돈다.
                }
            }
            flag = 0; // 둘다 0이되게하는 i를 찾지못하면 flag를 0으로바꾸고 다음번에
            // while문을 탈출.
        }
        //분자 분모를 divisor로 다시 모두 나눠줌.
        new_first /= divisor;
```

```
        new_second /= divisor;
        //해당 클래스 인스턴스의 실제 분자 분모 멤버에도 반영해줌.
        first = new_first;
        second = new_second;
    }
    // Rational 클래스받아 더하기 연산후 Rational 클래스를 반환해주는 Add함수.
    Rational Add(Rational r) {
        //공통분모를 맞춰주고 더해주는 연산.
        //(그 때의 분자분모를 new_first와 new_second에 담는다)
        int new_first = r.first * second + r.second * first;
        int new_second = r.second * second;
        // 새로운 Rational인스턴스를 공통분모계산을 통해 더하기연산을 해준 인스턴스로 할당.
        Rational new_r = Rational(new_first, new_second);
        // 그 Rational 인스턴스를 기약분수로 만들어주는 set_best()연산.
        new_r.set_best();
        //반환.
        return new_r;
    }
    bool Equal(Rational r) {
        //받은 유리수의 분자와, 연산주체인스턴스의 분모의 곱,
        //그리고 받은 유리수의 분모와 연산주체인스턴스의 분자의 곱의 차가 0이면
        //일치하면 같은 것으로 판단 후 true반환
        //그렇지 않을시 false반환
        if (r.first * second - r.second * first == 0) {
            return true;
        }
        else {
            return false;
        }
    }
};
void Print(Rational r) {
    // Rational클래스의 분자를 출력후 '/'를 연달아 출력후 분모를 출력 후 줄바꿈
    std::cout << r.first << '/' << r.second << std::endl;
};
void Print(std::vector<Rational> vec) {
    // Print함수의 오버로딩함수로
    // Rational클래스들을 요소로받는 벡터를 받아서 각각을 출력.
    // 몇번째 for문 순회인지를 알려주는 count변수에 1할당.
    int count = 1;
    // 벡터에서 Rational클래스들을 하나씩 뽑아냄.
    for (Rational r : vec) {
        // Rational클래스의 분자를 출력후 '/'를 연달아 출력후 분모를 출력 후 줄바꿈
        std::cout << r.first << '/' << r.second;
        //마지막 for문 순회가 아닐때 Comma를 붙혀서 출력.
        if (count != (int)vec.size()) {
            std::cout << ',' << ' ';
        }
        count += 1;
    }
    std::cout << std::endl; // 줄바꿈
}
//두 Rational클래스를 곱해주어 새로운 Rational인스턴스를 반환하는 함수
Rational Mul(Rational r1, Rational r2) {
    //분자끼리의 곱
    int first = r1.first * r2.first;
    //분모끼리의 곱
    int second = r1.second * r2.second;
    //서로 상호 곱해진 분자 분모로 새로운 Rational 인스턴스 r을 만듦.
    Rational r = Rational(first, second);
    //그 r을 기약분수로 바꿈.
    r.set_best();
    //r을 반환
    return r;
}
// 문자열을 받아서 ***를 양끝에 붙힌뒤 바뀐 문자열을  반환하는 함수.
std::string NewString(std::string s) {
    //***를 양끝에 붙힘.
    return "***" + s + "***";
};
int main() {
    Rational r1, r2(5), r3(2, 8), r4; // r1,r2,r3,r4선언
    Print(r1); // prints 0/1
```

```
    Print(r2); // prints 5/1
    Print(r3); // prints 1/4

    r4 = Mul(r2, r3); // r4 = r2*r3
    Print(r4); // prints 5/4
    r4 = r2.Add(r3); // r4 = r2+r3
    Print(r4); // prints 21/4
    // 동일한지 판단.
    if (r4.Equal(Rational{ 42, 8 })) std::cout << "Equal" << std::endl;

    std::vector<Rational> v1;
    //벡터에 생성자 인수들을 전달.
    v1.push_back({ 1 }); v1.push_back({ 3, 7 });
    Print(v1); // prints 1/1, 3/7

    std::string s1 = "C++ programming", s2;
    s2 = NewString(s1); // s2: "***C++ programming***"
    std::cout << s2 << std::endl; // prints ***C++ programming***
}
```

결과출력

```
0/1
5/1
1/4
5/4
21/4
Equal
1/1, 3/7
***C++ programming***

C:\Users\qhfkd\Project1\x64\Debug\Project1.exe(프로세스 22864개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```