

자료구조의 기초

## Lab 1. Basic Concepts

Taewhan Kim

# Lab Introduction

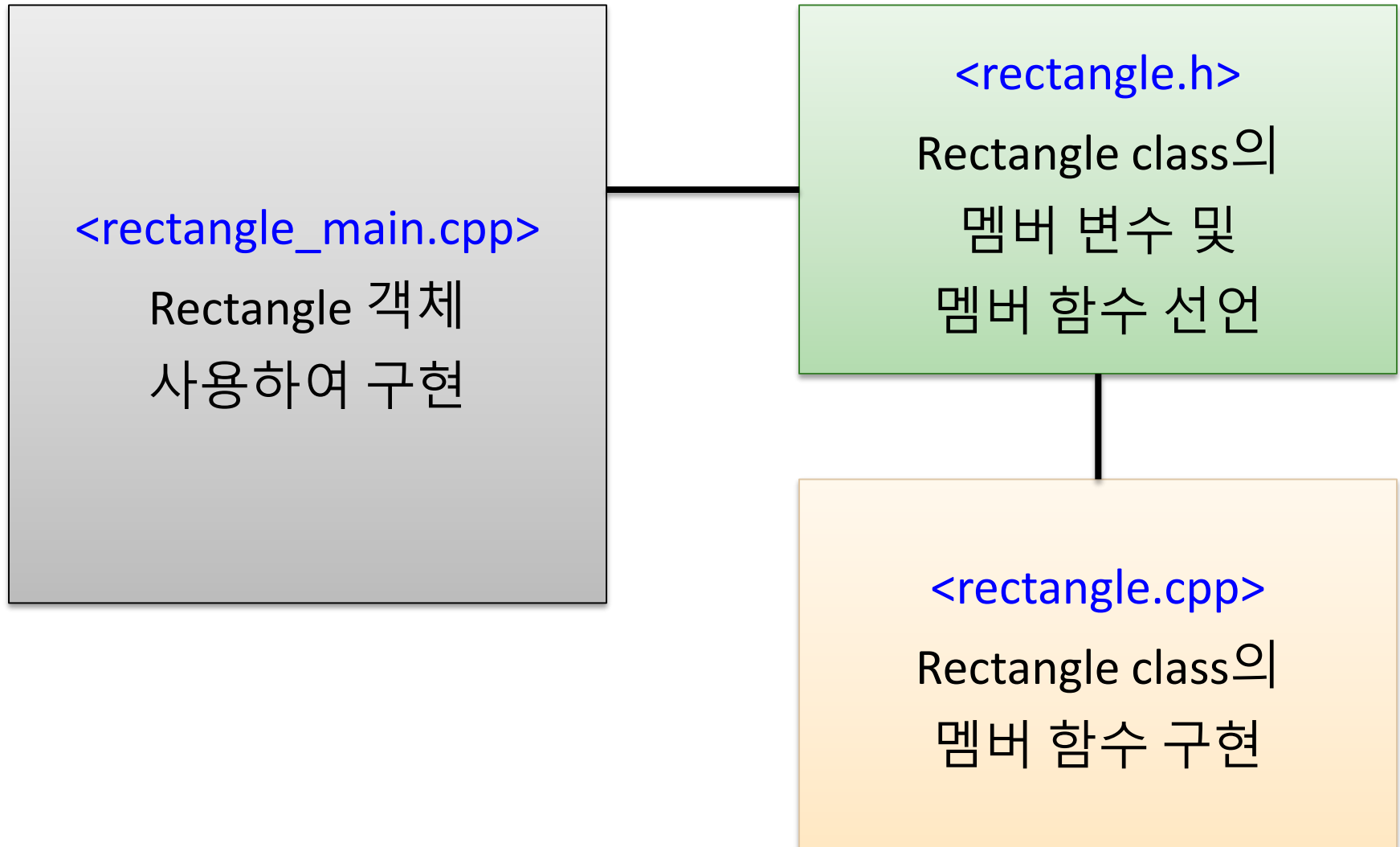
## ■ Visual Studio

- 모랩에 설치된 **Visual Studio 2017** 사용
- 개인 노트북에 설치된 Visual Studio 사용 가능

## ■ 출석

- **출석부에 서명 + eTL에 실습 코드 업로드**로 출석 체크
- 둘 중 하나라도 누락 시 **결석** 처리

# Rectangle Class 구조



# Header & Source

## rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

class Rectangle
{
public:
    Rectangle();
    ~Rectangle();
    int getWidth();
    int getHeight();
private:
    int xlow, ylow, width, height;
};
#endif
```

선언

## rectangle.cpp

```
#include "rectangle.h"

Rectangle::Rectangle() {
    xlow = 0;
    ylow = 0;
    width = 0;
    height = 0;
}

Rectangle::~Rectangle() {}

int Rectangle::getWidth() {
    return width;
}

int Rectangle::getHeight() {
    return height;
}
```

구현

# Rectangle Class 구현 – #ifndef~#endif

## rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

...

#endif
```

- #ifndef RECTANGLE\_H
  - 컴파일 시에 RECTANGLE\_H 정의 여부 체크
  - 정의가 된 경우
    - #endif 이후 코드 수행
  - 정의가 되지 않은 경우
    - #ifndef부터 #endif까지의 코드 수행 및 #endif 이후 코드 수행
- #define RECTANGLE\_H
  - RECTANGLE\_H 정의
- Header에 #ifndef~#endif 사용 이유
  - 프로그램 전체에서 Header를 한 번만 선언하여 중복 정의 방지

# Rectangle Class 구현 – Class 정의

## rectangle.h

```
class Rectangle
{
public:
    ...
private:
    int xlow, ylow,
        width, height;
};
```

## rectangle.cpp

```
#include "rectangle.h"

Rectangle::~~() {
    ...
}
```

- Header
  - Rectangle class 정의
  - 멤버 변수는 **private**으로 정의
  - 멤버 함수는 **public**으로 정의
- Source
  - Header 파일 불러오기 위해  
**#include "rectangle.h"** 입력
  - Rectangle class의 멤버 함수 구현
    - 멤버 함수 앞에 **Rectangle::** 추가

# Rectangle Class 구현 – Constructor

| rectangle.h                            | rectangle.cpp  |
|--|--|
| Rectangle();                           | <pre>Rectangle::Rectangle() {     xlow = 0;     ylow = 0;     width = 0;     height = 0; }</pre>                           |
| Rectangle(int x, int y, int w, int h); | <pre>Rectangle::Rectangle(int x, int y, int w, int h) {     xlow = x;     ylow = y;     width = w;     height = h; }</pre> |

- 객체 생성 시 멤버 변수 초기화 등의 기능 수행

# Rectangle Class 구현 – Getter, Setter

## rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

class Rectangle
{
public:
    int getWidth();
    int getHeight();
    void setWidth(int);
    void setHeight(int);
private:
    int xlow, ylow, width, height;
};
#endif
```

## rectangle.cpp

```
#include "rectangle.h"

int Rectangle::getWidth() {
    return width;
}

int Rectangle::getHeight() {
    return height;
}

void Rectangle::setWidth(int w) {
    width = w;
}

void Rectangle::setHeight(int h) {
    height = h;
}
```

- 멤버 변수 직접 접근 불가, Getter/Setter로 접근



# Rectangle Class 구현 – Operator Overloading

| rectangle.h  | rectangle.cpp   |
|--|---|
| <pre>#include &lt;iostream&gt;  using namespace std;  class Rectangle { public:     ...     int operator==(const Rectangle&amp;);     friend ostream&amp; operator&lt;&lt;         (ostream&amp;, const Rectangle&amp;); }</pre> | <pre>int Rectangle::operator==(const Rectangle &amp;s) {     if (this == &amp;s) return true;     if ((xlow == s.xlow) &amp;&amp; (ylow == s.ylow)         &amp;&amp; (width == s.width)         &amp;&amp; (height == s.height))         return true;     else return false; }  ostream&amp; operator&lt;&lt;(ostream&amp; os,     const Rectangle&amp; r) {     os &lt;&lt; "(" &lt;&lt; r.xlow &lt;&lt; ", "         &lt;&lt; r.ylow &lt;&lt; ")" &lt;&lt; endl;     os &lt;&lt; "Area : " &lt;&lt; r.width * r.height         &lt;&lt; endl;     return os; }</pre> |

- 다른 Class에 대해서도 동일한 Operator 사용 가능

# Rectangle Class 구현 – Test

## rectangle\_main.cpp

```
#include "rectangle.h"

int main() {
    Rectangle a;
    Rectangle c(0, 0, 3, 4);

    cout << a;
    cout << c;

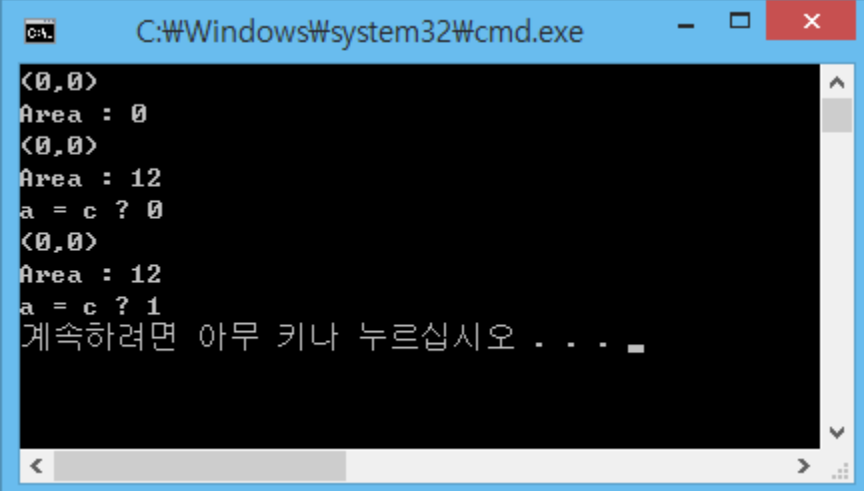
    cout << "a = c ? " << (a == c) << endl;

    a.setWidth(3);
    a.setHeight(4);

    cout << a;

    cout << "a = c ? " << (a == c) << endl;

    return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is as follows:

```
<0,0>
Area : 0
<0,0>
Area : 12
a = c ? 0
<0,0>
Area : 12
a = c ? 1
계속하려면 아무 키나 누르십시오 . . .
```

# TODO : Complex Class 구현

- 정의
  - 복소수 ( $a+bi$ )를 표현하는 Class
- 멤버 변수 (**private**로 정의)
  - `int re` : 실수부 ( $a$ )
  - `int im` : 허수부 ( $b$ )
- Constructor
  - 정수 두 개를 받아 하나는 실수부, 하나는 허수부로 설정
    - 예) `Complex(1, 3)`은  $1 + 3i$ 를 나타냄
  - Default Constructor :  $0 + 0i$ 
    - 예) `Complex()`는  $0 + 0i$ 를 나타냄

# TODO : Complex Class 구현

## ■ Operator Overloading

□ **Operator +** : 두 복소수의 합을 값으로 갖는 Complex 객체 반환

■ 예)  $(3 + 2i) + (5 + 3i) = (8 + 5i)$

□ **Operator -** : 두 복소수의 차를 값으로 갖는 Complex 객체 반환 (교환법칙 X)

■ 예)  $(3 + 2i) - (5 + 3i) = (-2 - i)$ ,  $(5 + 3i) - (3 + 2i) = (2 + i)$

□ **Operator \*** : 두 복소수의 곱을 값으로 갖는 Complex 객체 반환

■ 예)  $(3 + 2i) * (5 + 3i) = (9 + 19i)$

□ **Operator <<** :  $(a + bi)$  형태로 화면에 출력

■ 예)  $(1 + 2i)$ ,  $(-1 + 3i)$ ,  $(-5 + -2i)$ ,  $(4 + -9i)$

# TODO : Complex Class 구현

## ■ TODO

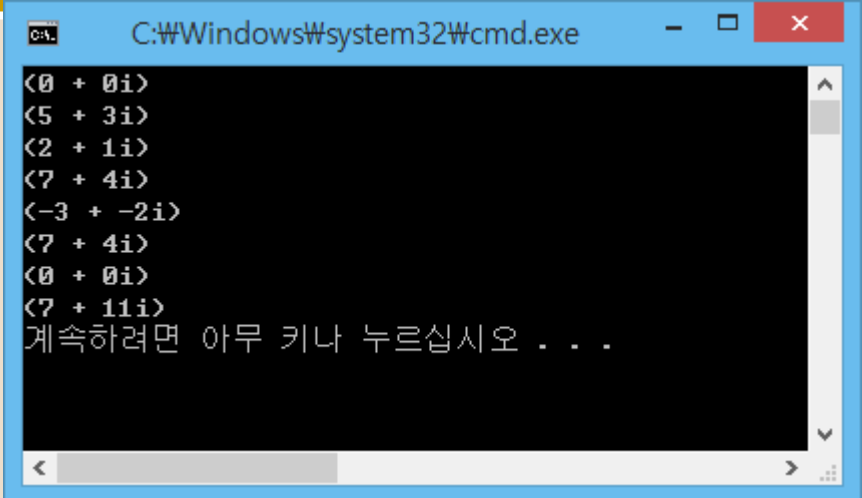
- complex.h에 선언된 Constructor 및 Operator overloading을 앞에 설명한 내용과 같이 complex.cpp에 구현

## ■ Test

### complex\_main.cpp

```
#include "complex.h"

int main() {
    Complex a;
    Complex b(5, 3);
    Complex c(2, 1);
    cout << a << b << c;
    cout << a + b + c << a - b + c << b - a + c;
    cout << a * b << b * c;
    return 0;
}
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the program, showing complex numbers in the format  $\langle \text{real} + \text{imaginary}i \rangle$ . The output is as follows:

```
<0 + 0i>
<5 + 3i>
<2 + 1i>
<7 + 4i>
<-3 + -2i>
<7 + 4i>
<0 + 0i>
<7 + 11i>
계속하려면 아무 키나 누르십시오 . . .
```

# Code Submission

## ■ 코드 제출

- 완료한 실습 코드를 다음과 같이 압축
  - 제출할 코드 : rectangle.h, rectangle.cpp, rectangle\_main.cpp, complex.cpp, complex\_main.cpp
  - 압축 파일명 : lab1\_홍길동\_2017-10000.zip
- 오늘 (2018년 3월 12일) **오후 11시**까지 eTL에 제출
- 제출된 코드는 따로 채점하지 않음

## ■ 출석

- **출석부에 서명 + eTL에 실습 코드 업로드**로 출석 체크
- 둘 중 하나라도 누락 시 **결석** 처리