



Computational Problem Solving II








CPET-321

Lab #2

RIT | College of
Engineering Technology

Assignments: Saturday 9/17 @ 11:59 PM

- zyBook

LAB #2	60 pts
Due: 09/17/2022, 11:59 PM EDT	
 Shown to students	
 18.1 (8) GHZ Lab: Student class	10 pts
 18.2 (8) GHZ LAB: Course Size	10 pts
 18.3 (8) GHZ LAB: Drop Student	10 pts
 18.4 (8) GHZ LAB: Find Student w/ Highest GPA	10 pts
 18.5 (8) GHZ LAB: Dean's List	10 pts
 18.6 (8) GHZ LAB: Print Student Roster	10 pts

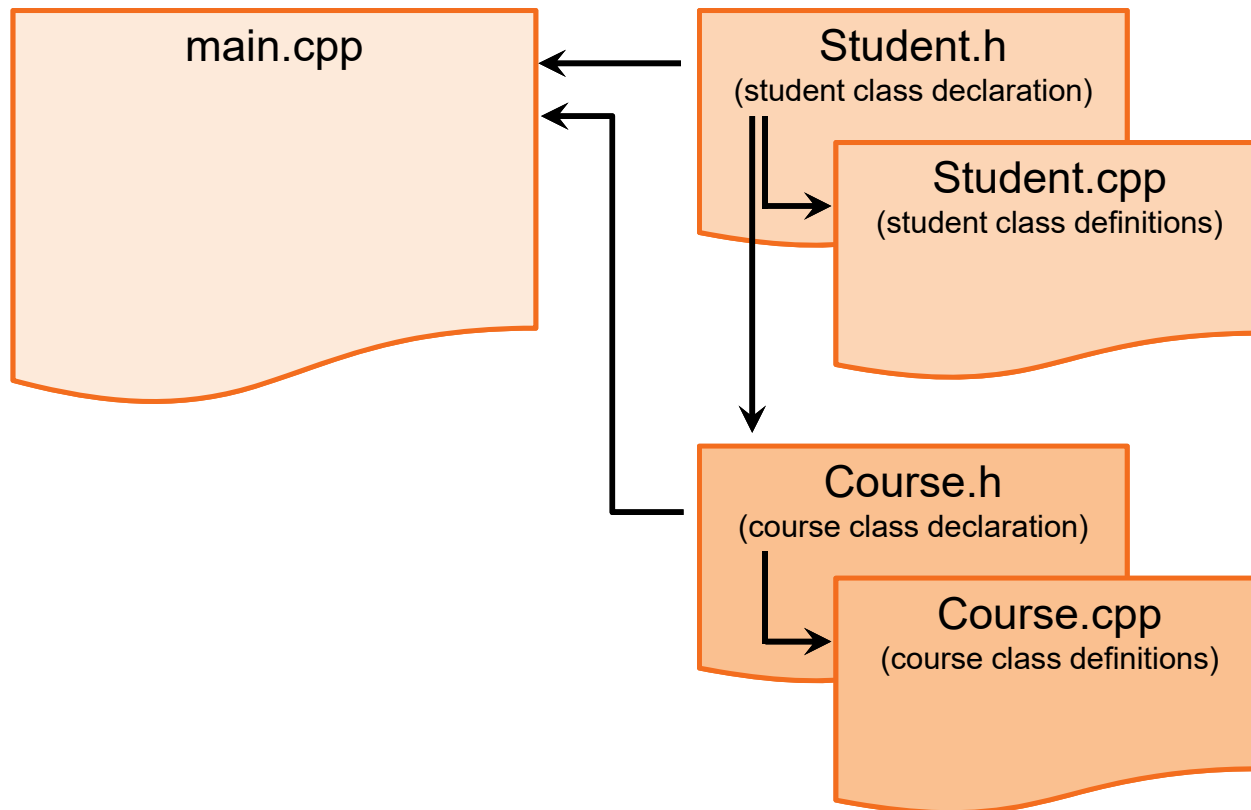
- Visual Studio

Rectangular Numbers:

- main_rec_nums.cpp
- rectangular.cpp
- rectangular.h

ZYBOOK CODE

Student / Course Class (LABS 18.1 – 18.6)



- 18.1 Student Class

- 18.2 Course Size
- 18.3 Drop Student
- 18.4 Find Student w/ Highest GPA
- 18.5 Dean's List
- 18.6 Print Student Roster

18.1 LAB: Student Class

In the Student.cpp file and Student.h file, build the Student class with the following specifications:

- Private data members
 - string name - Initialized in default constructor to "Louie"
 - double gpa - Initialized in default constructor to 1.0
- Default constructor
- Public member functions
 - SetName() - sets the student's name
 - GetName() - returns the student's name
 - SetGPA() - sets the student's GPA
 - GetGPA() - returns the student's GPA

Files:

- | | |
|---------------------|---------------------------|
| • main.cpp | Given |
| • Student.cpp | Partial, need to complete |
| • Student.h | Partial, need to complete |

Ex. If a new Student object is created, the default constructor sets name to "Louie" and gpa to 1. The output of GetName() and GetGPA() before and after calling SetName("Felix") and SetGPA(3.7) is:

```
Louie/1  
Felix/3.7
```



18.2 LAB: Course Size

Complete the Course class by implementing the CourseSize() member function, which returns the total number of students in the course.

Given classes:

- Class Course represents a course, which contains a vector of Student objects as a course roster. (Type your code in here.)
- Class Student represents a classroom student, which has three data members: first name, last name, and GPA.

Note: For testing purposes, different student values will be used.

Ex. For the following students:

```
Henry Bendel 3.6  
Johnny Min 2.9
```

the output is:

```
Course size: 2
```

Files:

- | | |
|---------------------|---------------------------|
| • main.cpp | Given |
| • Course.cpp | Partial, need to complete |
| • Course.h | Given |
| • Student.cpp | Given |
| • Student.h | Given |



18.3 LAB: Drop Student

Complete the `Course` class by implementing the `DropStudent()` member function, which removes a student (by last name) from the course roster. If the student is not found in the course roster, no student should be dropped.

Given classes:

- Class `Course` represents a course, which contains a vector of `Student` objects as a course roster. (Type your code in here.)
- Class `Student` represents a classroom student, which has three private data members: first name, last name, and GPA. (Hint: `getLast()` returns the last name field.)

Note: For testing purposes, different student values will be used.

Ex. For the following students:

```
Henry Nguyen 3.5  
Brenda Stern 2.0  
Lynda Robison 3.2  
Sonya King 3.9
```

the output for dropping Stern is:

```
Course size: 4 students  
Course size after drop: 3 students
```

Files:

- | | |
|----------------------------------|---------------------------|
| • <code>main.cpp</code> | Given |
| • <code>Course.cpp</code> | Partial, need to complete |
| • <code>Course.h</code> | Given |
| • <code>Student.cpp</code> | Given |
| • <code>Student.h</code> | Given |



18.4 LAB: Find Student with Highest GPA

Complete the Course class by implementing the FindStudentHighestGpa() member function, which returns the Student object with the highest GPA in the course. Assume that no two students have the same highest GPA.

Given classes:

- Class Course represents a course, which contains a vector of Student objects as a course roster. (Type your code in here.)
- Class Student represents a classroom student, which has three private data members: first name, last name, and GPA. (Hint: GetGPA() returns a student's GPA.)

Note: For testing purposes, different student values will be used.

Ex. For the following students:

```
Henry Nguyen 3.5
Brenda Stern 2.0
Lynda Robison 3.2
Sonya King 3.9
```

the output is:

```
Top student: Sonya King (GPA: 3.9)
```

Files:

- | | |
|---------------------|---------------------------|
| • main.cpp | Given |
| • Course.cpp | Partial, need to complete |
| • Course.h | Given |
| • Student.cpp | Given |
| • Student.h | Given |



18.5 LAB: Dean's List

A student makes the Dean's list if their GPA is 3.5 or higher. Complete the Course class by implementing the GetDeansList() member function, which returns a vector of students with a GPA of 3.5 or higher.

Given classes:

- Class Course represents a course, which contains a vector of Student objects as a course roster. (Type your code in here.)
- Class Student represents a classroom student, which has three data members: first name, last name, and GPA. (Hint: getGPA() returns a student's GPA.)

Note: For testing purposes, different student values will be used.

Ex. For the following students:

```
Henry Nguyen 3.5  
Brenda Stern 2.0  
Lynda Robison 3.2  
Sonya King 3.9
```

the output is:

```
Dean's list:  
Henry Nguyen (GPA: 3.5)  
Sonya King (GPA: 3.9)
```

Files:

- | | |
|---------------------|---------------------------|
| • main.cpp | Given |
| • Course.cpp | Partial, need to complete |
| • Course.h | Given |
| • Student.cpp | Given |
| • Student.h | Given |



18.6 LAB: Print Student Roster

Complete `Course.cpp` by implementing the `PrintRoster()` function, which outputs a list of all students enrolled in a course and also the total number of students in the course.

Given files:

- `main.cpp` - contains the main function for testing the program.
- `Course.cpp` - represents a course, which contains a vector of `Student` objects as a course roster. (Type your code in here)
- `Course.h` - header file for the `Course` class.
- `Student.cpp` - represents a classroom student, which has three data members: first name, last name, and GPA.
- `Student.h` - header file for the `Student` class.

Ex: If the program has 4 students enrolled in the course, the output looks like:

```
Henry Cabot (GPA: 3.5)
Brenda Stern (GPA: 2.1)
Jane Flynn (GPA: 3.9)
Lynda Robison (GPA: 3.2)
Students: 4
```

Files:

- | | |
|----------------------------------|---------------------------|
| • <code>main.cpp</code> | Given |
| • <code>Course.cpp</code> | Partial, need to complete |
| • <code>Course.h</code> | Given |
| • <code>Student.cpp</code> | Given |
| • <code>Student.h</code> | Given |



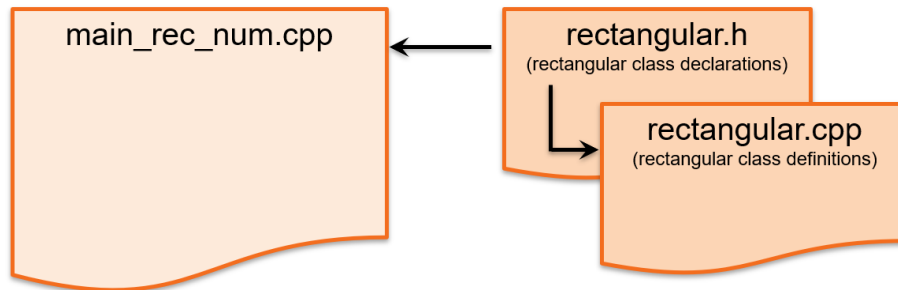
VISUAL STUDIO CODE

Rectangular Numbers – Specifications (1 of 3)

This project contains three files:

- ***main_rec_num.cpp***
- ***rectangular.cpp***
- ***rectangular.h***

Using the course example *Polar_Numbers* as a reference, implement the class ***Rectangular***. The declarations for this class should be code in ***rectangular.h*** and the definitions for this call should be coded in ***rectangular.cpp***



The code for ***main_rec_num.cpp*** has been provided.

Rectangular Numbers – Specifications (2 of 3)

The class Rectangular contains the following data members and member functions:

- Data Members – Two double variables (**real** and **imaginary**) representing the REAL and IMAGINARY components of the Rectangular number.
- **Rectangular()** – Default Constructor
- **Rectangular(double, double)** – Constructor with parameters for the REAL and IMAGINARY components of the number.
- **void setRectangularNumber(double, double)** – Mutator Function with parameters for the REAL and IMAGINARY components of the number.
- **void displayRectangularNumber()** – Accessor Function (this function has been implemented for you).
- **Rectangular operator + (Rectangular)** – Overload of the addition operator for Rectangular Numbers.
 - This function returns the sum (A+B) of the two Rectangular numbers.
- **Rectangular operator - (Rectangular)** – Overload of the subtraction operator for Rectangular Numbers.
 - This function returns the difference (A-B) of the two Rectangular numbers.

Rectangular Numbers – Specifications (3 of 3)

The class Rectangular contains the following member functions:

- **bool operator == (Rectangular)** – Overload of Equal (relational) operator for Rectangular Numbers.
 - This function returns true if the two Rectangular numbers are equal, otherwise it returns false.
 - Two Rectangular numbers are equal if the MAGNITUDE ($\sqrt{\text{REAL}^2 + \text{IMAGINARY}^2}$) of the first number is EQUAL to the MAGNITUDE of the second.
- **bool operator > (Rectangular)** – Overload of Greater-Than (relational) operator for Rectangular Numbers.
 - This function returns true if the first Rectangular number is greater than the second Rectangular number, otherwise it returns false.
 - The first Rectangular numbers is Greater-Than the second if the MAGNITUDE ($\sqrt{\text{REAL}^2 + \text{IMAGINARY}^2}$) of the first number is Greater-Than the MAGNITUDE of the second.
- **Rectangular operator+()** – Overload of Increment (unary) operators a Rectangular Number.
 - This function rotates (flips) the coordinates clockwise. For example, if the coordinates is in the 1st quadrant (i.e. $\text{REAL} > 0$ and $\text{IMAGINARY} > 0$), the results will be rotated to the 4th quadrant. If the coordinates are in the 4th quadrant (i.e. $\text{REAL} > 0$ and $\text{IMAGINARY} < 0$), the results will be rotated to the 3rd quadrant, etc.
 - If the coordinates are on the X or Y axis (i.e. $\text{REAL} = 0$ or $\text{IMAGINARY} = 0$) the coordinates are NOT rotated.
- **Rectangular operator-()** – Overload of Decrement (unary) operators a Rectangular Number.
 - This function rotates (flips) the coordinates counter-clockwise. For example, if the coordinates is in the 1st quadrant (i.e. $\text{REAL} > 0$ and $\text{IMAGINARY} > 0$), the results will be rotated to the 2nd quadrant. If the coordinates are in the 2nd quadrant (i.e. $\text{REAL} < 0$ and $\text{IMAGINARY} > 0$), the results will be rotated to the 3rd quadrant, etc.
 - If the coordinates are on the X or Y axis (i.e. $\text{REAL} = 0$ or $\text{IMAGINARY} = 0$) the coordinates are NOT rotated.

Rectangular Numbers – Outputs

The main() program (i.e. **main_rec_num.cpp**) has one input. Based on this input, 1 of 5 Section Tests will be selected. When fully implemented, the output for each test should be...

```
Microsoft Visual Studio Debug Console
1
Section #1 Test:
(A) 3.00 + j 4.00
(B) 4.00 + j 5.00
(C) -3.00 - j 4.00
(D) 8.00 - j 8.00
```

```
Microsoft Visual Studio Debug Console
2
Section #2 Test:
A + B = 7.00 + j 9.00
A - B = -1.00 - j 1.00
A + C = 0.00 + j 0.00
A - C = 6.00 + j 8.00
C + D = 5.00 - j 12.00
C - D = -11.00 + j 4.00
A + B - C + D = 18.00 + j 5.00
(A + B) - (C + D) = 2.00 + j 21.00
```

```
Microsoft Visual Studio Debug Console
3
Section #3 Test:
(A) 3.00 + j 4.00
(B) 3.00 + j 4.00
A is equal-to B
(A) 3.00 + j 4.00
(B) 4.00 + j 4.00
A is less-than B
(A) 3.00 + j 4.00
(B) 4.00 + j 3.00
A is equal-to B
(A) 3.00 + j 4.00
(B) 3.00 + j 3.00
A is greater-than B
```

```
Microsoft Visual Studio Debug Console
4
Section #4 Test:
Rotate Clock-Wise
(A) 8.00 + j 12.00
(B) 8.00 - j 12.00
(C) -8.00 - j 12.00
(D) -8.00 + j 12.00
(E) 8.00 + j 12.00
```

```
Microsoft Visual Studio Debug Console
5
Section #5 Test:
Rotate Counter Clock-Wise
(A) 8.00 + j 12.00
(B) -8.00 + j 12.00
(C) -8.00 - j 12.00
(D) 8.00 - j 12.00
(E) 8.00 + j 12.00
```

Rectangular Numbers – Code

- **displayRectangularNumber()**

```
void Rectangular::displayRectangularNumber()
{
    cout << fixed << setprecision(2);
    cout << setw(6) << real << " ";
    if (imaginary >= 0)
        cout << "+ j" << setw(6) << abs(imaginary) << endl;
    else
        cout << "- j" << setw(6) << abs(imaginary) << endl;
};
```

- *main_rec_num.cpp*

