COMP2123 Programming technologies and tools

Assignment 2 - Classes and Separate Compilation

Deadline: Dec 13, 23:00

Dear Students,

Hello! If you have any questions w.r.t. this assignment, please first visit our Assignment 2 Discussion Forum to see if you can find our answer there☺. If not, please post your questions in the Forum. If you would like to ask through email, please feel free to send email to Emily (**hzhang@cs.hku.hk**), and cc email to Dr. Cui (**heming@cs.hku.hk**). We are very happy to answer your questions.

- For all questions, please submit them through Moodle, and click the submit button before the deadline.
- Please double check your submission, check ALL of these to ensure that your assignment is submitted properly:
  - Check your email for acknowledgement from Moodle, keep it for record.
  - Check the assignment page again and check if the status is changed to "Submitted for grading".
  - Check the assignment page to see all files you have submitted.

Please contact us as soon as possible if you encounter any problem regarding the submission.

Enjoy this assignment and let us know if you need our help.

TA Emily and Dr. Cui

Please answer this question 1 on Moodle.

1. [35%] Consider the following program, `q1.h` and `q1.cpp`:

```
//q1.h
int addA (int i) {
    return i+100;
}
int b=2;
void MultiplyB(int i) {
    b = addA(i) * 5;
}
void InitialC() {
    MultiplyB(addA(b));
}
```

```
//q1.cpp
#include <iostream>
#include "q1.h"
using namespace std;
int main() {
    InitialC();
    cout << b << endl;
    return 0;
}
```

In an attempt to separate the function `MultiplyB()` and `InitialC()` from the main program, `q1.h` and `q1.cpp` are separated into four files, `q1a.h`, `q1a.cpp`, `q1b.h` and `q1b.cpp`.

```
//q1a.h
int addA(int i) {
    return i+100;
}
```

```
//q1b.h
void InitialC();
```

```
//q1a.cpp
#include <iostream>
#include "q1b.h"
using namespace std;
int main() {
    InitialC();
    cout << b << endl;
    return 0;
}
```

```
//q1b.cpp
#include "q1a.h"
#include "q1b.h"
int b=2;
void MultiplyB(int i) {
    b = addA(i) + 5;
}
void InitialC() {
    MultiplyB(addA(b));
}
```

In order to compile the program, the following commands are used in order:

```
1. g++ -c q1a.cpp
2. g++ -c q1b.cpp
3. g++ q1a.o q1b.o -o q1
```

a) The first command fails to compile anything. Suggest how `q1a.cpp` can be modified to fix the problem. Make sure that after the fix, the above commands should compile executable `q1` successfully. (You can only modify `q1a.cpp`)

b) Suppose the problem in part a) is fixed. In an attempt to further separate the function `MultiplyB()` from `q1b.cpp`, `q1b.cpp` is modified and a new file, `q1c.h` is created.

```
//q1b.cpp
#include "q1a.h"
#include "q1c.h"
void InitialC() {
    MultiplyB(addA(b));
}
```

```
//q1c.h
#include "q1a.h"
int b=2;
void MultiplyB(int i) {
    b = addA(i) + 5;
}
```

Compiling the program again with the above commands failed. Suggest how `q1a.h` can be modified to fix the problem. Make sure that after the fix, the above command should compile executable `q1` successfully. (You can only modify `q1a.h`)

c) Suppose the problem in part a) and part b) are fixed. A Makefile is created for building `q1a.exe`.

```
# Makefile
q1 : q1a.cpp q1b.cpp
     g++ -c q1a.cpp
     g++ -c q1b.cpp
     g++ q1a.o q1b.o -o q1
```

Suppose you have modified one of the header files (`q1a.h`, `q1b.h` or `q1c.h`) and try to compile it again with the command `make q1`. However, nothing is compiled. Explain why and suggest a way to fix the problem.

Please answer question 2 on Moodle.

2. [20%] Each of the following programs `Q2a.cpp`, `Q2b.cpp`, `Q2c.cpp` and `Q2d.cpp` cannot be compiled.

a. State what the error in each program is and

b. Suggest how the error can be fixed.

Please note the constraint on how each of the programs can be modified.

```
//Q2a.cpp
#include <iostream>
using namespace std;
class Integer{
public:
    int value;
};
int main(){
    Integer i, j;
    i.value = j.value = 1;
    if (i==j) cout << "Equal!" << endl;
    return 0;
} // Output: "Equal!"
```

Constraint: You should not modify the `main()` function.

```
//Q2b.cpp
#include <iostream>
using namespace std;
class Integer{
    int value;
    public:
        Integer(int i) { value = i; };
        void setValue(int i) const { value = i; };
        int getValue() const { return value; };
};
int main(){
    Integer a(1);
    a.setValue(2);
    cout << a.getValue() << endl;
    return 0;
} // Output: "2"
```

Constraint: You should not modify the `main()` function..

```
//Q2c.cpp
#include <iostream>
using namespace std;
class Integer{
    int value;
    public:
        Integer(int i) { value = i; };
        int getValue() { return value; };
};
Integer operator+(Integer & a, Integer & b){
    Integer i(a.value+b.value);
    return i;
}
int main(){
    Integer a(1), b(2);
    Integer c = a+ b;
    cout << c.getValue() << endl;
    return 0;
} // Output: "3"
```

Constraint: You should not change the visibility of the existing members in the `Integer` class.

```
//Q2d.cpp
#include <iostream>
using namespace std;
class Integer{
    int value;
    public:
        Integer(int i) { value = i; };
        int getValue() { return value; };
        friend Integer operator+(Integer & a, Integer & b){
            Integer i(a.value+b.value);
            return i;
        }
};

int main(){
    Integer a(1), b(2), c(3);
    Integer d = a + b + c;
    cout << d.getValue() << endl;
    return 0;
} // Output: 6
```

Constraint: You should not change the visibility of the existing members in the `Integer` class. You should not modify the `main()` function.

Question 3. [45%]

A **complex number** is a number that can be expressed in the form

$$a + bi$$

where a and b are real numbers and i is the imaginary unit, which satisfies the equation $i^2 = -1$. In this expression, a is the real part and b is the imaginary part of the complex number.

This question requires you to implement a class `Complex` that performs simple arithmetic operations on complex numbers. Your program should include two files

- `Complex.h` which includes the class definitions. **All member variables of the `Complex` class should be private.**
- `Complex.cpp` which includes the definitions of the member functions of the `Complex` class.

Please implement the following operations.

1 Constructors.

Implement the following two constructors

- `Complex()` - initializes `a` and `b` as `0`.
- `Complex(double real, double imagine).`
    - Initializes `a` as `real`, and `b` as `imagine`.
    - For example, `Complex c1(1.0, 5.0)` constructs a `Complex` object which real part is `1.0` (a `double` value) and the imaginary part is `5.0` (a `double` value).

Test case 1	**Result:**

`int main(){`
`  Complex c1;`
`  Complex c2(1.0, 5.0);`
`  return 0;`
`}`
	(Two complex objects created without error messages, there will be no printout)

2. Getters and Setters.

- `setReal()` function sets the real part of the `Complex` object.
- `setImaginary()` function sets the imaginary part of `Complex` object.
- `getReal()` returns the real part of the `Complex` object.
- `getImaginary()` returns the imaginary part of the `Complex` object.

Test case 2	**Output**

`int main(){`
`  Complex c1;`
`  cout << c1.getReal()<< " " << c1.getImaginary() << endl;`
`  c1.setReal(2.5);`
`  c1.setImaginary(5.5);`
`  cout << c1.getReal()<< " " << c1.getImaginary() << endl;`
`  return 0;`
`}`
	`0 0`
	`2.5 5.5`

3. Inserting a `Complex` object in an `ostream` object.

- Implement the necessary functions to enable `cout` of a complex object with an insertion operator.
- Depending on whether the imaginary part is positive or negative, the printout will follow this format:

`a+bi` if the imaginary part is positive.
`a-bi` if the imaginary part is negative.

Test case 3	**Output**

`int main(){`
`  Complex c1(1.0, 5.0);`
`  cout << "c1 = " << c1 << endl;`
`  Complex c2(2.0, -3.0);`
`  cout << "c2 = " << c2 << endl;`
`  return 0;`
`}`
	`c1 = 1+5i`
	`c2 = 2-3i`

4. Arithmetic operators (+, -): Addition operator.

- `Complex` objects are added by separately adding the real and imaginary parts of the summands. That is to say:

$$(a+bi)+(c+di)=(a+c)+(b+d)i$$ … Addition case 4.1a

Test case 4.1a	**Output**

`int main(){`
`  Complex c1(1.0, 5.0);`
`  Complex c2(2.0, -3.0);`
`  cout << "c1 = " << c1 << endl;`
`  cout << "c2 = " << c2 << endl;`
`  cout << "c1 + c2 = " << c1 + c2 << endl;`
`  return 0;`
`}`
	`c1 = 1+5i`
	`c2 = 2-3i`
	`c1 + c2 = 3+2i`

- Consecutive addition of `Complex` objects

Test case 4.1b	**Output**

`int main(){`
`  Complex c1(1.0, 5.0);`
`  Complex c2(2.0, -3.0);`
`  Complex c3(-5.5, 5.2);`
`  cout << "c1 + c2 + c3 = "`
`  << c1 + c2 + c3 << endl;`
`  return 0;`
`}`
	`c1 + c2 + c3 = -2.5+7.2i`

- Adding a `Complex` object with a `double` (The `double` is added to the real part of the `Complex` object).

Test case 4.2	**Output**

`int main(){`
`  Complex c1(1.0, 5.0);`
`  cout << "c1 + 3 = "  << c1 + 3.0 << endl;`
`  return 0;`
`}`
	`c1 + 3 = 4+5i`

- Adding a `double` with a `Complex` object (The `double` is added to the real part of the `Complex` object).

Test case 4.3	**Output**

`int main() {`
`  Complex c1(1.0, 5.0);`
`  cout << "1.6 + c1 = "  << 1.6 + c1 << endl;`
`  return 0;`
`}`
	`1.6 + c1 = 2.6+5i`

5. Handle the same tasks from test cases 4.1 to 4.4 for the subtraction operator. Subtraction is defined by:

$(a+bi)-(c+di)=(a-c)+(b-d)i$ … Subtraction case Test 5.1a

Test case		**Output**

5.1a	`int main(){`
	`  Complex c1(2.0, 3.0);`
	`  Complex c2(1.0, -5.0);`
	`  cout << "c1 = "  << c1  << endl;`
	`  cout << "c2 = "  << c2  << endl;`
	`  cout << "c1 - c2 = "  << c1 - c2  << endl;`
	`  return 0;`
	`}`
		`c1 = 2+3i`
		`c2 = 1-5i`
		`c1 - c2 = 1+8i`

5.1b	`int main() {`
	`  Complex c1(2.0, 3.0);`
	`  Complex c2(1.0, -5.0);`
	`  Complex c3(-2.5, 3.2);`
	`  cout << "c1 - c2 - c3 = "`
	`  << c1 - c2 - c3  << endl;`
	`  return 0;`
	`}`
		`c1 - c2 - c3 = 3.5+4.8i`

5.2	`int main() {`
	`  Complex c1(1.0, 5.0);`
	`  cout << "c1 - 3 = " << c1 - 3.0 << endl;`
	`  return 0;`
	`}`
		`c1 - 3 = -2+5i`

5.3	`int main() {`
	`  Complex c1(1.0, 5.0);`
	`  cout << "1.6 - c1 = " << 1.6 - c1 << endl;`
	`  return 0;`
	`}`
		`1.6 - c1 = 0.6-5i`

5.4	`int main() {`
	`  Complex c1(1.0, 5.0);`
	`  Complex c2(2.0, -3.0);`
	`  Complex c3 = 3.0 - c1 - 4.4 - c2 ;`
	`  cout << "c3 = " << c3 << endl;`
	`  return 0;`
	`}`
		`c3 = -4.4-2i`

6.1 The multiplication operator

The multiplication of two `Complex` objects is defined by the following formula:

$$(a+bi)*(c+di)=(ac-bd)+(bc+ad)i$$

A Complex object * A Complex object … Multiplication case 1

A Complex object * a double … Multiplication case 2

A double * A Complex object … Multiplication case 3

Test case 6.1	**Output**

`int main() {`
`  Complex c1(1.0, 5.0);`
`  Complex c2(2.0, -3.0);`
`  cout << "c1 * c2 = " << c1*c2 << endl;`
`  cout << "3.6 * c1 = " << 3.6*c1 << endl;`
`  cout << "c2 * 1.5 = " << c2*1.5 << endl;`
`  return 0;`
`}`
	`c1 * c2 = 17+7i`
	`3.6 * c1 = 3.6+18i`
	`c2 * 1.5 = 3-4.5i`

6.2 **The multiplication equal operator (*=)**

Test6.2 Run:	**Result:**

`int main() {`
`  Complex c1(1.0, 5.0);`
`  Complex c2(2.0, -3.0);`
`  cout << "c1 * c2 = " << c1 * c2 << endl;`
 	
`  cout << "c1 = " << c1 << endl;`
`  c1 *= c2 ;`
`  cout << "c1 = " << c1 << endl;`
 	
`  return 0;`
`}`
	`c1 * c2 = 17+7i`
	`c1 = 1+5i`
	`c1 = 17+7i`

Please use the test cases to test your implementation. Your implementation should ensure that each of those targets can be compiled successfully and the resulting client programs work correctly. **Please run all the test cases before your submit, and create more test cases to thoroughly test your program.**

Please submit `Complex.h` and `Complex.cpp` through Moodle.

-- END –