객체지향프로그래밍 II

Exercise for chapter 9 (week 3)

Class의 이해 및 활용 (교과서 9장, Part 1)

Exercise for :

1. Time Class Case Study
2. Class Scope and Accessing Class Members
3. Constructors with Default Arguments

박인규 교수

포함 전처리기 (preprocessor wrappers)

- 코드가 한번 이상 포함(include)되는 것을 방지
  - ✓ #ifndef XXX – “if XXX is not defined yet”

    : 만약 XXX가 이미 포함되어 있다면 이 코드를 건너 뛴다.
  - ✓ #define XXX

    : 이 코드가 나중에 다시 포함되지 않도록 라벨(label) XXX을 정의 (define)
  - ✓ #endif – #ifndef의 끝부분 (짝을 이룸)
- 만약 헤더가 이전에 포함되었다면
  - ✓ 라벨 XXX은 이미 정의 되었고, 따라서 헤더파일은 더 이상 포함되지 않는다.
- 다중 정의 (multiple definition) 에러를 방지

Ex 9-1: Preprocessor Wrappers 예제 (클래스 선언 Header 파일)

```cpp
// Fig. 9.1: Time.h
// Declaration of class Time.
// Member functions are defined in Time.cpp

// prevent multiple inclusions of header file
#ifndef TIME_H
#define TIME_H

// Time class definition
class Time
{
public:
   Time(); // constructor
   void setTime( int, int, int ); // set hour, minute and second
   void printUniversal(); // print time in universal-time format
   void printStandard(); // print time in standard-time format
private:
   int hour; // 0 - 23 (24-hour clock format)
   int minute; // 0 - 59
   int second; // 0 - 59
}; // end class Time

#endif
```

Preprocessor directive `#ifndef` determines whether a name is defined

Preprocessor directive `#define` defines a name (e.g., `TIME_H`)

Preprocessor directive `#endif` marks the end of the code that should not be included multiple times

Ex 9-1: Preprocessor Wrappers 예제 (클래스 구현 CPP 파일)

```
// Fig. 9.2: Time.cpp
// Member-function definitions for class Time.
#include <iostream>
using std::cout;

#include <iomanip>
using std::setfill;
using std::setw;

#include "Time.h" // include definition of class Time from Time.h

// Time constructor initializes each data member to zero.
// Ensures all Time objects start in a consistent state.
Time::Time()
{
   hour = minute = second = 0;
} // end Time constructor

// set new Time value using universal time; ensure that
// the data remains consistent by setting invalid values to zero
void Time::setTime( int h, int m, int s )
{
   hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
   minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
   second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
} // end function setTime
```

Ensure that **`hour`**, **`minute`** and **`second`** values remain valid

Ex 9-1: Preprocessor Wrappers 예제 (클래스 구현 CPP 파일)

```

// print Time in universal-time format (HH:MM:SS)
void Time::printUniversal()
{
   cout << setfill( '0' ) << setw( 2 ) << hour << ":"
      << setw( 2 ) << minute << ":" << setw( 2 ) << second;
} // end function printUniversal

// print Time in standard-time format (HH:MM:SS AM or PM)
void Time::printStandard()
{
   cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << ":"
      << setfill( '0' ) << setw( 2 ) << minute << ":" << setw( 2 )
      << second << ( hour < 12 ? " AM" : " PM" );
} // end function printStandard
```

Using **`setfill`** stream manipulator to specify a fill character

Ex 9-1: Preprocessor Wrappers 예제 (Driver 파일)

```cpp
// Fig. 9.3: fig09_03.cpp
// Program to test class Time.
// NOTE: This file must be compiled with Time.cpp.
#include <iostream>
using std::cout;
using std::endl;

#include "Time.h" // include definition of class Time from Time.h

int main()
{
   Time t; // instantiate object t of class Time

   // output Time object t's initial values
   cout << "The initial universal time is ";
   t.printUniversal(); // 00:00:00
   cout << "\nThe initial standard time is ";
   t.printStandard(); // 12:00:00 AM

   t.setTime( 13, 27, 6 ); // change time

   // output Time object t's new values
   cout << "\n\nUniversal time after setTime is ";
   t.printUniversal(); // 13:27:06
   cout << "\nStandard time after setTime is ";
   t.printStandard(); // 1:27:06 PM

   t.setTime( 99, 99, 99 ); // attempt invalid settings
```

Ex 9-1: Preprocessor Wrappers 예제 (Driver 파일)

```cpp

   // output t's values after specifying invalid values
   cout << "\n\nAfter attempting invalid settings:"
      << "\nUniversal time: ";
   t.printUniversal(); // 00:00:00
   cout << "\nStandard time: ";
   t.printStandard(); // 12:00:00 AM
   cout << endl;
   return 0;
} // end main
```

```
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM

After attempting invalid settings:
Universal time: 00:00:00
Standard time: 12:00:00 AM
```

클래스 스코프 (class scope)와 클래스 멤버로의 접근

- 점 (dot) 멤버 선택 연산자 (.)
  - ✓ 객체의 멤버에 접근 가능하다.
  - ✓ 객체의 이름(name) 이나 객체의 참조(reference)에 사용
    - 예) sunset.printUniversal( );

- 화살 (arrow) 멤버 선택 연산자 (->)
  - ✓ 객체의 멤버에 접근 가능하다.
  - ✓ 객체의 포인터에 사용한다.
    - 예) timePtr->printUniversal( )

Ex 9-2: 클래스 멤버로의 접근 예제

```cpp
// Fig. 9.4: fig09_04.cpp
// Demonstrating the class member access operators . and ->
#include <iostream>
using std::cout;
using std::endl;

// class Count definition
class Count
{
public: // public data is dangerous
   // sets the value of private data member x
   void setX( int value )
   {
      x = value;
   } // end function setX

   // prints the value of private data member x
   void print()
   {
      cout << x << endl;
   } // end function print

private:
   int x;
}; // end class Count
```

Ex 9-2: 클래스 멤버로의 접근 예제 (cont.)

```cpp

int main()
{
   Count counter; // create counter object
   Count *counterPtr = &counter; // create pointer to counter
   Count &counterRef = counter; // create reference to counter

   cout << "Set x to 1 and print using the object's name: ";
   counter.setX( 1 ); // set data member x to 1
   counter.print(); // call member function print

   cout << "Set x to 2 and print usin[illegible]";
   counterRef.setX( 2 ); // set data member x to 2
   counterRef.print(); // call member function print

   cout << "Set x to 3 and print using a pointer to an object: ";
   counterPtr->setX( 3 ); // set data member x to 3
   counterPtr->print(); // call member function print
   return 0;
} // end main
```

Using the dot member selection operator with an object

Using the dot member selection operator with a reference

Using the arrow member selection operator with a pointer

```
Set x to 1 and print using the object's name: 1
Set x to 2 and print using a reference to an object: 2
Set x to 3 and print using a pointer to an object: 3
```

생성자의 디폴트 인자 (default arguments)

- 데이터 멤버를 일관성 있게 초기화 한다.
  - ✓ 심지어 생성자의 인자가 주어지지 않은 경우 포함
- 모든 인자에 디폴트 인자가 지정된 생성자는 역시 디폴트 생성자 (default constructor)이다.
  - ✓ 어떠한 인자도 없이 호출 할 수 있다.
  - ✓ 클래스당 최대 한 개의 디폴트 생성자를 가진다.

Ex 9-3: 생성자의 디폴트 인자 예제 (Time.h)

```cpp
// Fig. 9.8: Time.h
// Declaration of class Time.
// Member functions defined in Time.cpp.

// prevent multiple inclusions of header file
#ifndef TIME_H
#define TIME_H

// Time abstract data type definition
class Time
{
public:
   Time( int = 0, int = 0, int = 0 ); // default constructor

   // set functions
   void setTime( int, int, int ); // set hour, minute, second
   void setHour( int ); // set hour (after validation)
   void setMinute( int ); // set minute (after validation)
   void setSecond( int ); // set second (after validation)
```

Prototype of a constructor with default arguments

Ex 9-3: 생성자의 디폴트 인자 예제 (Time.h)

```

   // get functions
   int getHour(); // return hour
   int getMinute(); // return minute
   int getSecond(); // return second

   void printUniversal(); // output time in universal-time format
   void printStandard(); // output time in standard-time format
private:
   int hour; // 0 - 23 (24-hour clock format)
   int minute; // 0 - 59
   int second; // 0 - 59
}; // end class Time

#endif
```

Ex 9-3: 생성자의 디폴트 인자 예제 (Time.cpp)

```cpp
// Fig. 9.9: Time.cpp
// Member-function definitions for class Time.
#include <iostream>
using std::cout;

#include <iomanip>
using std::setfill;
using std::setw;

#include "Time.h" // include definition of class Time from Time.h

// Time constructor initializes each data member to zero;
// ensures that Time objects start in a consistent state
Time::Time( int hr, int min, int sec )
{
   setTime( hr, min, sec ); // validate and set time
} // end Time constructor

// set new Time value using universal time; ensure that
// the data remains consistent by setting invalid values to zero
void Time::setTime( int h, int m, int s )
{
   setHour( h ); // set private field hour
   setMinute( m ); // set private field minute
   setSecond( s ); // set private field second
} // end function setTime
```

Parameters could receive the default values

Ex 9-3: 생성자의 디폴트 인자 예제 (Time.cpp)

```cpp

// set hour value
void Time::setHour( int h )
{
   hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
} // end function setHour

// set minute value
void Time::setMinute( int m )
{
   minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
} // end function setMinute

// set second value
void Time::setSecond( int s )
{
   second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
} // end function setSecond

// return hour value
int Time::getHour()
{
   return hour;
} // end function getHour

// return minute value
int Time::getMinute()
{
   return minute;
} // end function getMinute
```

Ex 9-3: 생성자의 디폴트 인자 예제 (Time.cpp)

```cpp

// return second value
int Time::getSecond()
{
   return second;
} // end function getSecond

// print Time in universal-time format (HH:MM:SS)
void Time::printUniversal()
{
   cout << setfill( '0' ) << setw( 2 ) << getHour() << ":"
      << setw( 2 ) << getMinute() << ":" << setw( 2 ) << getSecond();
} // end function printUniversal

// print Time in standard-time format (HH:MM:SS AM or PM)
void Time::printStandard()
{
   cout << ( ( getHour() == 0 || getHour() == 12 ) ? 12 : getHour() % 12 )
      << ":" << setfill( '0' ) << setw( 2 ) << getMinute()
      << ":" << setw( 2 ) << getSecond() << ( hour < 12 ? " AM" : " PM" );
} // end function printStandard
```

Ex 9-3: 생성자의 디폴트 인자 예제 (driver)

```cpp
// Fig. 9.10: fig09_10.cpp
// Demonstrating a default constructor for class Time.
#include <iostream>
using std::cout;
using std::endl;

#include "Time.h" // include definition of class Time from Time.h

int main()
{
   Time t1; // all arguments defaulted
   Time t2( 2 ); // hour specified; minute and second defaulted
   Time t3( 21, 34 ); // hour and minute specified; second defaulted
   Time t4( 12, 25, 42 ); // hour, minute and second specified
   Time t5( 27, 74, 99 ); // all bad values specified

   cout << "Constructed with:\n\nt1: all arguments defaulted\n  ";
   t1.printUniversal(); // 00:00:00
   cout << "\n  ";
   t1.printStandard(); // 12:00:00 AM

   cout << "\n\nt2: hour specified; minute and second defaulted\n  ";
   t2.printUniversal(); // 02:00:00
   cout << "\n  ";
   t2.printStandard(); // 2:00:00 AM
```

Initializing **`Time`** objects using 0, 1, 2 and 3 arguments

Ex 9-3: 생성자의 디폴트 인자 예제 (driver)

```

   cout << "\n\nt3: hour and minute specified; second defaulted\n  ";
   t3.printUniversal(); // 21:34:00
   cout << "\n  ";
   t3.printStandard(); // 9:34:00 PM

   cout << "\n\nt4: hour, minute and second specified\n  ";
   t4.printUniversal(); // 12:25:42
   cout << "\n  ";
   t4.printStandard(); // 12:25:42 PM

   cout << "\n\nt5: all invalid values specified\n  ";
   t5.printUniversal(); // 00:00:00
   cout << "\n  ";
   t5.printStandard(); // 12:00:00 AM
   cout << endl;
   return 0;
} // end main
```

Ex 9-3: 생성자의 디폴트 인자 예제 (실행 결과)

```
Constructed with:

t1: all arguments defaulted
  00:00:00
  12:00:00 AM

t2: hour specified; minute and second defaulted
  02:00:00
  2:00:00 AM

t3: hour and minute specified; second defaulted
  21:34:00
  9:34:00 PM

t4: hour, minute and second specified
  12:25:42
  12:25:42 PM

t5: all invalid values specified
  00:00:00
  12:00:00 AM
```

Invalid values passed to constructor, so object **`t5`** contains all default data

Ex 9-5: Complex class

9.5 ***(Complex Class)*** Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the form

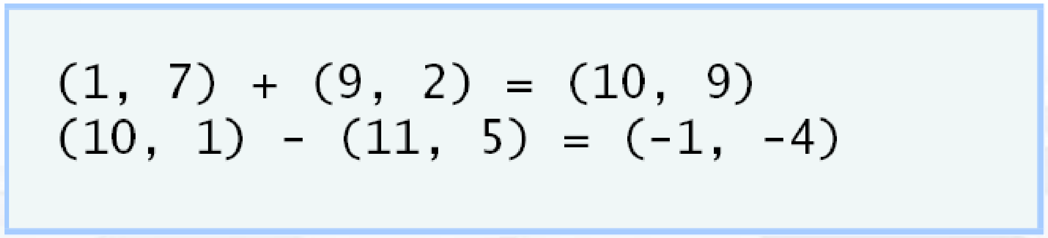
$$\text{realPart} + j\,\text{imaginaryPart}$$

Use double variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. **The constructor should contain default values in case no initializers are provided**. Provide public member functions that perform the following tasks:

a) Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.

b) Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.

c) Printing Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.



```
(1, 7) + (9, 2) = (10, 9)
(10, 1) - (11, 5) = (-1, -4)
```

HW #3 : Class의 이해 및 활용

- 예제 9-5의 기능을 완성하고 보고서 작성하기
- Deadline: 다음 실습 시간 전까지