

Exercise # 2 - Solution

(03-02-2012)

Exercise 1:

True or False?

- a. The specification of an abstract data type (ADT) should not mention implementation details. **True**
- b. If you do not use the reserved words public and private, all members of a C++ class are private. **True**
- c. A client is a software that declares and manipulates class members. **False**
- d. Some operations can be a combination of observers and constructors. **False**
- e. Implementing an abstract data type means choosing a concrete data representation of the abstract data. **False**

Fill in the blanks.

- a. In C++, a function contained within a class is called a **member function**.
- b. Protecting data from access by unauthorized functions is called **data hiding**.
- c. When each object of a class maintains its own copy of an attribute, the variable that represents the attribute is also known as a **data member**.
- d. Every class definition contains the keyword **class** followed immediately by the class's name.

Exercise 2:

Consider the following C++ class declaration and client code:

Class declaration

```
class SomeClass
{
public:
    void func1 (int n);
    int func2 (int n) const;
    void func3();
    SomeClass();
    SomeClass(int n);
private:
    int someInt;
};
```

Client code

```
SomeClass object1;
SomeClass object2(4);
int m;
int func3(int k);

object1.func1(3);
m = object2.func2(5);
```

- a. Give other possibilities of SomeClass class declaration.

```
class SomeClass
{
private:
    int someInt;
public:
```

```
class SomeClass
{
    int someInt;
public:
```

```

void func1 (int n);
int func2 (int n) const;
void func3();
SomeClass();
SomeClass(int n);
};

```

```

void func1 (int n);
int func2 (int n) const;
void func3();
SomeClass();
SomeClass(int n);
};

```

b. List all the identifiers that refer to data types (both built-in and programmer-defined).

void, SomeClass and int

c. List all the identifiers that are names of class members.

func1, func2, func3, SomeClass, and someInt

d. List all the identifiers that are names of class objects.

object1 and object2

e. List the names of all member functions that are allowed to inspect the private data.

func1, func2, func3, and SomeClass

f. List the names of all member functions that are allowed to modify the private data.

func1, func3, and SomeClass

g. Give the possible categories of each member function (constructor, observer, or transformer).

SomeClass: **constructor**

func1: **transformer**

func2: **observer**

func3: **transformer**

h. Which of the following client statements are allowed:

- `object1.someInt = object1.func2(7);` // **not allowed**
- `m = object2.func1(2);` // **not allowed**
- `object.func3();` // **not allowed**
- `if (object1 == object2) {` // **not allowed**
`func3(5);`
`}`
- `if (object1.someInt == object2.someInt) {` // **not allowed**
`func3(5);`
`}`
- `SomeClass arr[4];` // **allowed**
- `m = func3(m);` // **allowed**

i. Function *func2* gets an odd positive integer and returns the someInt'th next odd number. Write a specification and implementation of function *func2*.

Specification:

```
int func2(/* in */ int k) const;
// Precondition: k >= 0 and odd(k) where odd(x) means that x is an odd integer
// Postcondition: the someInt'th next odd number with respect to k is returned
```

Implementation:

```
int SomeClass::func2(/* in */ int k) const
// Precondition: k >= 0 and odd(k) where odd(x) means that x is an odd integer
// Postcondition: the someInt'th next odd number with respect to k is returned
{
    if ( k % 2 == 1) //another possible condition would be: (k > 0 && k % 2 )
    {
        return (k + 2*someInt);
    }
    else {
        return 0;
    }
}
```

- j. Design the data sets necessary to thoroughly test the *func2* function of the *SomeClass* class.

Test data:

1. a negative number
2. a positive even number
3. a positive odd number
4. zero

Exercise 3:

What is wrong in the following code?

```
class A
{
public:
    A() {};
    string s("abc");
};
int main() {
    A a;
    cout<<a.s<<endl;
    return 0;
}
```

Answer:

```
class A
{
public:
    A() { string s("abc");}
};
int main() {
    A a;
    cout<<a.s<<endl;
```

```
return 0;
}
```

Exercise 4:

```
#include<iostream>
using namespace std;
```

```
class count {
public:
    int count;
    Count(int c)
    {
        count = c;
    }

    Count()
    {
        count =0;
    }
};
```

```
void increment(Count c, int times)
```

```
{
    c.count++;
    times++;
}
```

```
int main() {
    Count myCount;
    int times = 0;

    for( int i=0; i<10;i++)
        increment(myCount, times);

    cout<<"myCount.count is: "<<myCount.count;

    cout<<"times is"<<times;
}
```

- a) What is the printout of the above code;

Answer: mycount is 0 and time is 0

- b) If the highlighted code in the exercise 4 is changed to:
void increment(Count &c, int times)

What is the printout?

Answer: mycount is 10 and time is 0

- c) If the highlighted code in the exercise 4 is changed to:
`void increment(Count &c, int ×)`

What is the printout?

Answer: mycount is 10 and time is 110

- d) Can you change the highlighted code in the exercise 4 is changed to:
`void increment(const Count c, int times)`

Answer: Error occurs. Because you cannot operate on constant object