B16 Object Oriented Programming

fwood@robots.ox.ac.uk (written by mosb)
http://www.robots.ox.ac.uk/∼fwood/teaching

Hilary 2015
Rev. February 5, 2015

- In a number of the questions, you are asked to write code. Your code need not be absolutely syntactically correct (though it would be good if it is), but must convey that you have understood the key ideas. If you are not sure about syntax, then add comments so that someone reading your "pseudo-code" can work out your intention.
- Questions marked [♣] are those that might be considered typical exam type questions, though again note that syntactically correct code would not be required in an exam, and that the stars indicate approximate level of difficulty and key concepts rather than length.

1. [♣] Object-oriented concepts: Explain the following concepts, and their significance, in the context of object-oriented programming:
   (a) Encapsulation.
   (b) Run-time Polymorphism.
2. **Classes:** You are to design a program for a mass-spring simulation. The program will make use of three classes: `Mass`, `Spring` and `World`. The `Mass` class, naturally, will have some intrinsic properties such as mass, position and velocity. Each mass will need to be able to update itself based on the force(s) applied to it. The `Spring` class will also have properties and will supply a force based on how far apart its ends are (as given by the mass positions). The physical system will comprise two masses joined by a spring, encoded in the `World` class. Design the class interfaces required for this program and explain (in words) how the classes will interact.

3. **Constructors:**

   A `MathVector` class is defined as follows:

```
class MathVector {
public:
    // constructors and other public interface goes here

private:
    // private data
    std::vector< double > elements;
    int length;
};
```

   (a) Design and code three constructors (using C++ or pseudo-code if you prefer):
      i. a default constructor taking a single integer indicating the size of the vector;
      ii. a constructor that takes a length integer and an array of doubles to initialise the values;
      iii. a copy constructor.

   (b) Give a example usage of each.

   (c) Explain the problem that would arise if we used dynamic memory allocation (that is, working with memory on the heap using C++-style `new` or C-style `malloc`) instead of using `std::vector<>`.

   (d) *[optional, not covered in lectures]* What mechanism would be used to overcome the problems arising in part (c)?

4. **Classes and function/operator overloading:**

   (a) For the `MathVector` class above, show how the square brackets operator (indexing operator) can be overloaded to return a given indexed element. Should this operator be a class member function, and if so will it be private or public?

   (b) Explain how the `*` operator can be overloaded to enable expressions such as $\alpha\mathbf{v}$, $\mathbf{v}\alpha$, and $\mathbf{u} * \mathbf{v}$ (which we take to mean the cross-product, which you need only implement for 3-vectors), where $\mathbf{u}$ and $\mathbf{v}$ are vectors and α is a scalar.

5. [♣] Inheritance: It is desired to write a class hierarchy for a set of graphics shapes, all of which have a position and orientation in 3D space, and which need to implement a method to draw themselves. The geometric objects are: Point, Line, Plane, Cube, Ellipsoid, Quadric, Cone, Cuboid.

   (a) Draw a likely class hierarchy.

   (b) Design the base class.

   (c) Explain why, by using a base class and hierarchy, the process of adding a new subclass, such as a Sphere, is greatly simplified.

6. **Patterns, Class design and the Standard Template Library**

   (a) Using C++'s Standard Template Library vector class template, define the private data and public interface of an `Image` class in which each pixel is a single byte (in C++ this is an `unsigned char`).
   Hint: the private data will need to store the image data itself and you could either use a vector of vectors of unsigned chars for this, or a single flat vector of unsigned chars; it will also need the height and width of the image; the interface will need methods to get and store pixel data, etc; the constructor should take the height and width as arguments and create the pixel data.

   (b) How could you change your class so that it could store a real number per pixel instead (i.e. `double`)?

7. [♣] Polymorphism: The box on the following page shows the definitions of a small C++ class hierarchy.

   (a) Draw a diagram showing the class hierarchy.

   (b) Determine what output the code will produce, describing carefully how you reach your conclusions.

   (c) Test a real implementation to verify your answer.

   (d) Identify which parameters are passed by value and which by reference. Taking account of the meaning of this, and what will be stored in the activation record when `doit1` and `doit2` are called, explain the result.

```
class Foo
{
public:
    void f() { std::cout << "Foo::f()" << std::endl; }
    virtual void g() { std::cout << "Foo::g()"
     << std::endl; }
};

class Bar : public Foo
{
public:
    void f() { std::cout << "Bar::f()" << std::endl; }
    virtual void g() { std::cout << "Bar::g()"
     << std::endl; }
};

void doit1(Foo f)
{
f.f();
f.g();
}

void doit2(Foo& f)
{
  f.f();
  f.g();
}

int main()
{
    Foo foo;
    Bar bar;
    doit1(foo);
    doit1(bar);
    doit2(foo);
    doit2(bar);

    return 0;
}
```

8. [♣] The box below shows the definitions of a set of C++ classes used for evaluating expressions.

   (a) Draw a diagram showing the class hierarchy.

   (b) Determine what output the following code will produce and explain clearly how you arrive at your answer. Hence or otherwise explain polymorphism.

```
Z a(2.5), b(3.0), c(10.0);
Y d(a, b);
Y e(d, c);

a.PrintValue();
d.PrintValue();
e.PrintValue();
```

   (c) Implement a class `Sum` and a class `Tan` that respectively compute a sum of arguments, and the tan of a single argument in radians.

```
class X {
public:
    virtual double Eval() { return 1.0; }
    void PrintValue() { cout << Eval() << endl; }
};

class Y : public X {
  public:
    Y(X& x1, X& x2) : arg1(x1), arg2(x2) {}
    double Eval() { return arg1.Eval() * arg2.Eval(); }
  private:
    X& arg1;
    X& arg2;
};

class Z : public X {
  public:
    Z(double v) : val(v) {}
    double Eval() { return -val; }
  private:
    double val;
};
```

9. **Design patterns and interfaces:** A class `Minimiser` is a class for finding the minimum of a real-valued, univariate function $f(x)$ (using a particular minimisation algorithm e.g. Newton-Raphson). Its public interface is defined as follows:

```
class Minimiser {
  public:
    Minimiser(const Func& f, const double startX);
    double Minimum() const;
    double MinimumLocation() const;
  private:
    Func _f;
    double _startX;
    double _minimum;
    double _minimumX;
};
```

The user supplies a class derived from the base class `Func` which can compute the value of the function at a point, and possibly its derivative at that point. The constructor first checks if the function can compute the derivative, and then finds the local minimum, starting from the x value `startX` (don't concern yourself with the details of this algorithm, other than to note that it must either use the user-supplied derivatives or compute them numerically).

(a) Design the interface for the base class `Func`.

(b) Write the skeleton of a class for the function e^x/x^2 and describe the implementation steps required to find the minimum of the function using the `Minimiser` class.