SF Natural Science Foundation Scholarship Examination

Hilary Term, 1998

COURSE 262 – COMPUTING

Friday, March 27, 1998
Exam Hall
9.30-12.30

John G. Carney

Attempt 4 Questions

1. a). What is operator overloading? Give an example of an in-built math operator in C++ that is overloaded and explain why it is overloaded.

b). Write a program in C++ that uses operator overloading to perform vector arithmetic. Include in your program a function to print vectors to the screen. Assume each vector consists of 5 integer values. Provide operations for add, subtract and multiply. Assume the usual C++ (left to right) operator precedence. The program should allow the programmer to write the following mainline:

```c++
main()
{
    vector4 = vector1 + vector2 + vector3;
```
a). What are C++ **template functions**? What advantages are there in using them?

b). Write a program in C++ that uses a template function to return the absolute value of any built-in data type passed to it.

c). Write a program in C++ that uses a Mileage class with 2 private members `city_name` and `miles_away` to model the distance of cities and towns from Dublin.

Include in your program a template function called `max()` that uses the `>=` operator to return the maximum of two values passed to it. The program should be written so that the template function can be used in the following way in the mainline:

```cpp
main()
{
    Mileage town1("Carlow", 50);
    Mileage town2("Waterford", 110);
    cout << "Of the first two towns, the following is farthest away: ";
    cout << max(town1, town2);
}```
3. a). What is the difference between private, protected and public inheritance in C++?

b). Consider the following equipment inheritance hierarchy for a company:

```
Equipment
   /       |
  Vehicle  Machine
   |       /   |
Car     Van
```

All vehicles have a registration number, colour, manufacturer, price, driver and top speed. Cars can be petrol or diesel but vans are always diesel. Machines have a function, price and make. All equipment has a location. Write a C++ program that best models the relationships between the objects in this hierarchy. Include constructor and destructor functions for each class implemented.

4. a). What are the differences between 'has-a' and 'is-a' relationships in object-oriented programming?

b). If you were asked to write a C++ program that models the solar system, what object-oriented relationships do you think would best model the relationships between:

- The solar system and its planets
- The planets and their moons
- Asteroid fields and asteroids
- The solar system and asteroid fields

Explain your reasoning.

c). Write a C++ program that models some very basic information about a solar system. Assume the solar system contains 7 planets, 20 moons, 2 asteroid fields and 1 sun. Each spherical body
(planet, moon or sun) contains a name, diameter, gravity and (x,y,z) co-ordinate. Each planet contains an atmosphere type, terrain type and water content. Each moon contains a body it is orbiting. Each asteroid field contains a specified number of asteroids, (x,y,z) co-ordinate of centre of field and estimated size of field. Include constructor and destructors for each object.

5. a. What is the difference between early (static) and late (dynamic) binding in C++?

b. What advantages are there in using C++ virtual functions and what is polymorphism?

c. Imagine you are part of a large software engineering team that develops graphics packages for PCs. Your project leader asks you to write the part of a graphics package that draws simple shapes (square, rectangle, triangle, and circle) on the screen. He is disorganised and busy and so only gives you the following information:

• Use this inheritance hierarchy:

```
      Shape
    /     \
   /       \       
Square     Rectangle     Triangle    Circle
```

• Each class contains basic information about a shape; position, size and colour.

• Each class has its own draw() member function.

• The classes are specified or initialised by the user who can instantiate many items of the same shape if he wishes.

• As each shape is created in your program, place a Shape * pointer to each new Shape object into an array called shape_array[].

• Include in your program a loop that iterates through shape_array[] sending draw messages to each object in the array to form a screen image.

• Assume the draw() functions for each shape are already written.

From this information write a graphics package in C++ that would best satisfy his requirements.