CSC2040 Data Structures, Algorithms and Programming Languages

Practical 3

21st October 2016

In this practical we will introduce the C++ class. In C++, typically, a class is declared in a Header file, and the class's member functions are implement in a Source file. This lets you share the class interface in other .cpp files whilst keeping the implementation separated.

Program 1

In this part of the practical we will look at the Person class introduced on Page 81 of the textbook. Download Person.h and Person.cpp from Queens Online. Study them closely and answer the following questions.

In Person.h there are two constructors. The first of these has a syntax:

```
Person(std::string first, std::string family, std::string ID, int birth) :
  given_name(first), family_name(family), ID_number(ID), birth_year(birth) { }
```

What does this do?

What is another way to write this constructor to achieve the same effect?

Write two C++ statements one declaring an object of this class that is initialised by the above parameterised constructor, and the other applying the `get_birth_year()` member function to this object.

Write two C++ statements one declaring an object of this class that is initialised by the parameter-less (i.e. default) constructor, and the other applying the `set_family_name(std::string family)` member function to this object with `family = "Smith"`.

Allocate an object of this class using **new** that is initialised by the parameterised constructor, and then apply the `get_ID_number()` member function to this object.

Allocate an object of this class using **new** that is initialised by the default constructor, and then apply the `set_birth_year(int birth)` member function to this object with `birth` being a suitable number.

Allocate an array of 385 objects of this class using **new,** and then apply the `set_given_name(std::string given)` member function to the 13th object with `given = "John"`.

In the above allocation of an array of objects, which one of the two constructors is used?

The following is an accessor function which is implemented in Person.h:

```
std::string get_given_name() const { return given_name; }
```

What is the word that describes this definition of a function?

What does the `const` mean in the above function declaration?

In this setter function

```
void set_given_name(const std::string& given) {
    given_name = given;
}
```

How would you describe the parameter `given`?

What is the return type of this function?

Can we make a change of the definition of this function as below? Why?

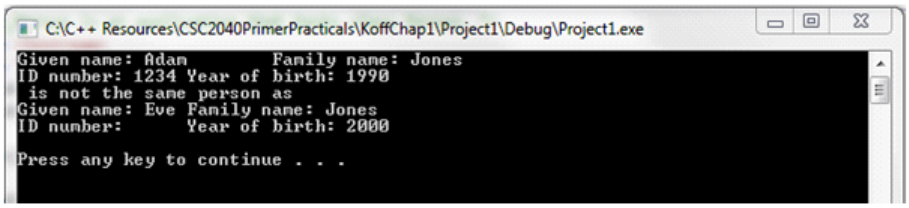
```
void set_given_name(const std::string& given) const {
    given_name = given;
  }
```

Program 2

Develop a Test program called TestPerson.cpp and type the following:

```
Person p1("Adam", "Jones", "1234", 0);
p1.set_birth_year(1990);
Person p2;
p2.set_given_name("Eve");
p2.set_family_name(p1.get_family_name());
p2.set_birth_year(p1.get_birth_year() + 10);
if (p1 == p2)
      cout << p1 << " is the same person as " << p2 << endl;
else
      cout << p1 << " is not the same person as " << p2 <<
endl;
```

Save, build and execute this program and examine the output. Now change the output operator << so that your output looks like this:



Program 3

Given two 1-D vectors (or arrays), `x`, `y`, of the same length `len`, their similarity can be measured using different methods. Two of the most common methods are: Euclidean Distance and Cosine Similarity. These are defined as follows:

Euclidean distance:

$$d(x,y) = \sqrt{(x[0]-y[0])^2+(x[1]-y[1])^2+ \ldots +(x[len-1]-y[len-1])^2}$$

Cosine similarity:

$$c(x,y) = p(x,y) / \left(\sqrt{x[0]^2+x[1]^2+\ldots+x[len-1]^2}\sqrt{y[0]^2+y[1]^2+\ldots+y[len-1]^2}\right)$$

where $p(x,y)$ is the inner (or dot) product between x and y:

```
p(x,y) = x[0]y[0] + x[1]y[1] + … + x[len-1]y[len-1]
```

Consider the following class named as `vecsim`, which calculates the above two similarity measures for two vectors, which are passed into an object through a parameterised constructor of the object:

```
class vecsim {
public:
      vecsim (double* v1, double* v2, int v_len);

      double Euclidean();
      double Cosine();

private:
      double* vec1, *vec2;
      int vec_len;
};
```

Implement the above class. Then, develop a Test program TestVecSim.cpp in which type the following:

```
double vector1[10] = {1.5, 3., 4.5, 6., 7.5, 9., 10.5, 12., 13.5, 15.};
double vector2[10] = {3., 4.5, 6., 7.5, 9., 10.5, 12., 13.5, 15., 16.5};

vecsim vs(vector1, vector2, 10);

cout << "Euclidean dis = " << vs.Euclidean() << endl;
cout << "Cosine sim = " << vs.Cosine() << endl;
```

Save, build and execute your program and examine the output.

Program 4

An extended `vecsim` class is shown below. The extended class includes an overloaded default (parameter-less) constructor, to allow for the declaration of an object without initialisers, and two overloaded distance functions, one for each distance type, to allow for the calculation of the similarity between two vectors which are passed into an object through the member function parameters.

```
class vecsim {
public:
      vecsim(double* v1, double* v2, int v_len);
      vecsim();

      double Euclidean();
      double Euclidean(double* v1, double* v2, int v_len);
      double Cosine();
      double Cosine(double* v1, double* v2, int v_len);

private:
      double* vec1, *vec2;
      int vec_len;
};
```

Extend your previous class to include the additional member functions. Then, in the Test program, following your previous testing statements, add the following:

```
vecsim vs2;

cout << "Euclidean dis = " << vs2.Euclidean(vector1, vector2, 10) << endl;
cout << "Cosine sim = " << vs2.Cosine(vector1, vector2, 10) << endl;
```

Save, build and execute your program and examine the output.

Practical Test 1

Practical Test 1 will take place during the practical class on Friday 28 October. This will be an open book test. Pen and paper can be used for working out/rough work. We will be using the Assignment Tool in Queens Online for you to upload the answers.