Started on	Friday, 9 October 2020, 12:50 AM
State	Finished
Completed on	Friday, 9 October 2020, 12:55 AM
Time taken	5 mins 10 secs
Marks	10.00/10.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

Suppose that you work with a derived object of type *D* that has multiple base classes *B1*, *B2* and *B3* (in this order). You work with this object through a base class pointer:

```
D d;
B2* b2_ptr = &d;
```

Following good practices of C++ programming, which **cast** should you use to convert the base class pointer (*b2_ptr*) to a valid derived class pointer?

Select one:

- a. D(b2_ptr); // C++ style cast
- b. dynamic_cast<D*>(b2_ptr); ✔ Correct!
- c. reinterpret_cast<D*>(b2_ptr);
- d. static_cast<D*>(b2_ptr);
- e. const_cast<D*>(b2_ptr);

This conversion requires information from a *vtable*, and so it calls for *dynamic_cast*.

The correct answer is: dynamic_cast<D*>(b2_ptr);

Question **2**

Correct

Mark 1.00 out of 1.00

Which of the creational **design patterns** creates new concrete objects by calling a *clone()/copy()* member function of an existing concrete object through a base class/inferface class pointer?

Select one:

- a. Visitor design pattern
- b. Template method design pattern
- c. Iterator design pattern
- d. Prototype design pattern ✔ Correct!
- e. Factory design pattern

Template method, iterator and visitor are not creational design patterns.Factory delegates creation of any supported type to a specific member function or its overrides.

Prototype design pattern uses existing objects as prototypes that can be clones to instantiate new objects of the same type as prototypes.

The correct answer is: Prototype design pattern

Question **3**

Correct

Mark 1.00 out of 1.00

Which of the following statements about interface classes is **true**?

Select one:

- a. They must inherit from a concrete implementation class.
- b. They allow for decoupling the use of an interface from a class instance realizing this interface. ✔ Correct!
- c. They contain virtual and non-virtual member functions.
- d. They contain only non-static data members.
- e. They may have private and public member functions.

Interface classes allow for **decoupling** the use of an interface from a class instance implementing ("*realizing*") this interface.

Typically, an interface class is a special class:

- All its member functions are pure virtual.
- A destructor is virtual and has a default implementation.
- It does not have static members.
- It does not have data members.
- It has no protected or private members.

The correct answer is: They allow for decoupling the use of an interface from a class instance realizing this interface.

Question **4**

Correct

Mark 1.00 out of 1.00

Which C++ keyword or specifier prevents any classes from inheriting from a given class?

As not all these keywords have been covered during lectures, your task is to visit https://en.cppreference.com/w/cpp and find out.

Select one:

- a. final ✔ Correct!
- b. overload
- c. abstract
- d. override
- e. virtual

The correct answer is: final

Question **5**

Correct

Mark 1.00 out of 1.00

A class *MyClass* inherits from an empty class, and has the following data members (in this order):

- float
- unsigned char
- bool

Assume that you compile the program on a Windows 10 64-bit platform.
What is the amount of **padding added at the end** of each object of this class?

Select one:

- a. 0 bytes
- b. 2 bytes ✔ Correct!
- c. 1 byte
- d. 4 bytes
- e. 6 bytes

Empty base class introduces 0 bytes, *float* starts at an offset 0 (divisible by 4), *unsigned char* starts at an offset 4 (divisible by 1), *bool* starts at an offset 5 (divisible by 1).

As all data members occupy 6 bytes, there must be **2 bytes of padding** to allow for the next *float* to start at the address 8 (divisible by 4).

The correct answer is: 2 bytes

Question **6**

Correct

Mark 1.00 out of 1.00

What is the most accurate (most precise) programming paradigm describing the **C** (not C++) programming language?

Select one:

- a. Template meta-programming
- b. Unstructured programming
- c. Procedural programming ✔ Correct!
- d. Generic programming
- e. Imperative programming

C uses procedural (structured, imperative) programming paradigm.

The correct answer is: Procedural programming

Question **7**

Correct

Mark 1.00 out of 1.00

Which of the **relationships** between two objects *Obj1* and *Obj2* fits the following description?

> *This relationship expresses a strong dependency between objects, where the object Obj1 is an integral part of the object Obj2 and cannot exist without it; deleting Obj2 results in destruction of the Obj1.*

In C++, this relationship is implemented as having a direct data member, and on the diagrams it is represented by a solid line with a filled-in diamond-shaped arrowhead.

Select one:

- a. Association
- b. Inheritance
- c. Dependency
- d. Realization
- e. Composition ✔ Correct!

This relationship is called **composition**.

The correct answer is: Composition

Question **8**

Correct

Mark 1.00 out of 1.00

Each answer below represents 3 values. In their respective order, they indicate the number of bytes that:

1. An empty base class *MyClass* contributes to the size of its derived class.
2. An empty class *MyClass* contributes to a class as a data member.
3. An interface class *MyClass* contributes to the size of a class that realizes this interface.

Choose the answer that correctly identifies **all three** values.

Select one:

- a. sizeof(void*) | 0 bytes | sizeof(void*)
- b. 1 bytes | 1 bytes | 0 bytes
- c. 1 byte | 0 bytes | sizeof(void*)
- d. 0 bytes | 1 byte | sizeof(void*) ✔ Correct!
- e. sizeof(MyClass) | sizeof(MyClass) | sizeof(MyClass)

0 bytes | 1 byte | sizeof(void*) or 0 bytes

1. An empty base class contributes 0 bytes to the size of its derived class.
2. An empty class contributes 1 byte to a class as a data member.
3. An interface class contributes 0 bytes or the size of a pointer (*vptr*) to the size of a class that realizes this interface.

The correct answer is: 0 bytes | 1 byte | sizeof(void*)

Question **9**

Correct

Mark 1.00 out of 1.00

Which of the statements about virtual inheritance is **not true**?

Select one:

- a. It requires the most derived class to create any virtual base class of its own base classes.
- b. It introduces vtable pointers into the object's memory layout.
- c. It allows for removing duplicates of base class data members.
- d. It does not affect the size of the object, compared to the non-virtual inheritance. ✔
  Correct; this statement is not true.
- e. It results in base class data members stored at the end, not at the beginning of the object.

Virtual inheritance may result in additional *vptr* pointers, removed duplicated data members, additional padding, and so overall it does impact the size of objects.

The correct answer is: It does not affect the size of the object, compared to the non-virtual inheritance.

Question **10**

Correct

Mark 1.00 out of 1.00

Which of the following statements about *vtables* is **true**?

Select one:

- a. A call to a static member function is resolved through vtable.
- b. A vptr and vtable are used for dynamic dispatch (typically, single dispatch). ✔
  Correct!
- c. A vtable contains only addresses of virtual functions.
- d. A vtable contains object's vptr.
- e. A derived object's vptr points at a vtable of its base class.

Single, double and multiple dispatch rely on virtual tables, and *vptr* and *vtable* facilitate such behaviour preparing requires data structures at the compilation step.

The correct answer is: A vptr and vtable are used for dynamic dispatch (typically, single dispatch).

◄ Templates - Practice Jump to... Quiz 3 ►