QuantNet

Review answers

Start date:	9 minutes ago
Complete date:	A moment ago
Question 1:	Which statement is true about classes and objects? ◯ Classes classify objects in separate groups. ◯ Objects state the objectives of a class. ✓ Classes describe the structure and behaviour of similar objects. An object is an instance of a class. ◯ Objects describe the structure and behaviour of similar classes. A class is an instance of an object.
Question 2:	What is encapsulation in the context of object-oriented programming? ✓ Bundling data with functionality that operates on that data. ◯ Hiding data from users. ◯ Compiling multiple classes in to one executable file. ◯ The process of writing a class.
Question 3:	Which statement is false about constructors? ✓ If we don't make a default constructor, then the system only creates one with a standard implementation when we didn't create any other constructors. ◯ If we don't make a copy constructor, then the system always creates one with a standard implementation. ◯ If we don't make a default constructor, then the system always creates one with a standard implementation. ◯ The copy constructor copies the state of an object and must accept a reference to the source object.
Question 4:	What statement is false about function name overloading? ◯ Overloading works with both member functions and global functions.

✓ Two functions can have the same name as long as the input arguments have different types.

○ Two functions can have the same name as long as the number of input arguments are different.
○ Two functions can have the same name as long as the output arguments have different types.

Question 5: Local variables can be defined at?

○ Only at the beginning of a function in both C and C++.
○ C and C++ don't have local variables.
○ Everywhere in a function in both C and C++.
✓ Only at the beginning of a function in C and everywhere in a function in C++.

Question 6: What statement is false about "pass by value" vs. "pass by reference"?

✓ To pass an argument by reference you need to declare the input parameter with a '*'.
○ Pass by value makes a copy of the argument.
○ Pass by value is less efficient than pass by reference for objects.
○ To pass an argument by reference you need to declare the input parameter with a '&'

Question 7: What statement is true about the use of #ifndef/#define/#endif statements in a class header file?

C++:

```
1  #ifndef MyClass_hpp
2  #define MyClass_hpp
3
4  class MyClass
5  {
6  };
7
8  #endif
```

○ The name of the #define must be the same as the header file.
✓ The #indef/#define/#endif statements are needed to ensure the class declaration can only be included once in each compilation unit.
○ The name of the #define must be the same as the class name.
○ The #indef/#define/#endif statements are needed to make the class known to the compiler. Else other files can't find the class that is declared.

declared.

Question 8: Which statement is true about data hiding?

- ○ Data hiding ensures the data cannot be changed.
- ○ Data hiding saves memory space.
- ○ Data hiding is mandatory in C++.
- ✓ Data hiding hides the internal data of a class from users of the class so the internal structure can be changed without affecting the users of a class.

Question 9: What statement is true about the following class?

C++:

```
1  // In the header file: "MyClass.hpp"
2  class MyClass
3  {
4    int m_data1;
5
6    MyClass();
7
8  private:
9    double m_data2;
10 }
11
12 // In the source file: "MyClass.cpp"
13 #include "MyClass.hpp"
14 MyClass::MyClass()
15 {
16 }
```

- ✓ This class declaration does not compile.
- ○ Client code can instantiate this class because it defines a default constructor.
- ○ The initial value of *m_data2* is 0.0.
- ○ *m_data1* is a public data member.

Question 10: Which statement is false about inline functions?

- ○ Inline functions can be executed faster than non inline functions.
- ✓ Functions declared as inline may not be compiled as regular function.
- ○ Member functions implemented within the class definition must also have the keyword *inline* to be compiled as inline.
- ○ The implementation of an inline function must be available at

	compile time. The function cannot be inlined when the function implementation is only available at link time.
Score:	7 (70.00%)
Pass/Fail:	Failed

Quiz ›

Contact us Advertise Terms and rules Privacy policy Help Home