Quiz >

Review answers

Complete date: A moment ago

Question 1: When writing a template class, were should you use the class name with the template type (*MyClass<T>*)?

- [x] In the name of the constructor and destructor.
- [x] In the source file of a template class before the scope operator.
- [x] When the template class is used as input argument.
- [ ] When the template class is used as output argument.

Question 2: What is the output of the following program?

C++:

```
1  class EA {};
2  class EB: public EA {};
3
4  void F()
5  {
6    throw EB();
7  }
8
9  int main()
10 {
11   try
12   {
13     F();
14   }
15   catch(EA&)
16   {
```

- (x) EA Exception Finished
- ( ) EB Exception Finished
- ( ) This program does not compile.
- ( ) EA Exception EB Exception Finished

Question 3: Which statement is false about generic programming?

◯ With generic programming, a generic data type is used in the code which at compile-time will be replaced by a specific type that is provided by the user of the generic code.
◯ Generic programming can be used as an alternative to polymorphic functions.
✓ You can't use generic programming and object-oriented programming at the same time.
◯ Generic programming enables us to create type-safe data structures for a certain type without creating a new class for each type.

Question 4: Which statement is false about composition and exceptions?

◯ When using composition, it is preferred that the parent object catches the exceptions thrown in the composite object and converts it to another exception object.
◯ When using composition, unhandled exceptions thrown in the composite object make that the client must know of the composite object.
◯ When using composition, the exceptions thrown in the composite object can be re-thrown with extra information.
✓ When using composition, it is preferred that the parent object does not catch the exceptions thrown in the composite object.

Question 5: Which statement is false about template classes?

◯ The template types are part of the class name.
◯ A template class is a description of a regular class.
✓ If a template class is defined as *template <typename T> class MyClass* and variables *a* and *b* are declared as *MyClass<int> a; MyClass<double> b;*, then variables *a* and *b* are of the same type.
◯ A class is an instance of a template class analog to that an object is an instance of a class.

Question 6: Which statement is false about iterators?

◯ In a loop that traverses a data structure, you cannot compare the current iterator with the end iterator using the < operator
✓ A *std::list<T>* supports a random access iterator.
◯ Iterators are a nested type of the data structure they iterate.
◯ Iterators are used to traverse data structures in a data structure independend way.

Question 7:	Which statement is false about exceptions? ◯ One *try* block can have many *catch* blocks. ✓ If a *try* block does not have any *catch* block, the exception will be passed to an outer *try* block or the system. ◯ If a *try* block does not have a matching catch block for the thrown exception, the exception will be passed to an outer *try* block or the system. ◯ Every type can be thrown. Not only primitive types but also your own class types.
Question 8:	What statement is true about catch handlers? ✓ A catch handler that catches all possible exceptions should catch "...": catch (...) {} ◯ A catch handler that catches all possible exceptions should use the default keyword: *catch default { }* ◯ A catch handler that catches all possible exceptions is not possible in C++. ◯ A catch handler that catches all possible exceptions should catch the Exception base class: catch (Exception& ex) {}
Question 9:	Which statement is false about STL? ✓ The STL library provides no functionality for networking. ◯ The STL library should be installed separately before you can use it. ◯ STL is a C++ library that uses templates for its implementation. ◯ The STL library provides among others various data structures, itrators, algorithms and allocators.
Question 10:	Which statement is false about sub-type polymorphism and parametric polymorphism? ✓ Parametric polymorphism does not work when the classes are not derived from a common base class. ◯ Parametric polymorphism is faster than inheritance polymorphism. ◯ Sub-type polymorphism is done at run-time while parametric polymorphism is done at compile-time. ◯ Sub-type polymorphism depends on virtual functions.
Score:	8 (80.00%)

Pass/Fail:	Passed

Quiz ›