

[Quiz](#) >

## Review answers

Start date: 16 minutes ago

Complete date: 1 minute ago

Question 1: Which statement is true about inheritance and aggregation?

- ☐ Inheritance is used when two class have a "has a" relationship.
- ☐ Aggregation specialises a general class.
- ☒ Inheritance is used when two classes have "is a kind of" (ISA/AKO) relation.
- ☐ Aggregation is used when two classes have "is a kind of" (ISA/AKO) relation.

Question 2: Which statement is false about sub-type polymorphism and parametric polymorphism?

- ☐ Sub-type polymorphism is done at run-time while parametric polymorphism is done at compile-time.
- ☐ Sub-type polymorphism depends on virtual functions.
- ☐ Parametric polymorphism is faster than inheritance polymorphism.
- ☒ Parametric polymorphism does not work when the classes are not derived from a common base class.

Parametric polymorphism assumes a class implements a certain function without the need for inheritance relations.

Question 3: What statement is true about the following code?

C++:

```
1 | A::A::B::B() { }
```

- ☐ This code does not compile because member functions in a namespace must be in a *namespace {}* block.
- ☐ This code implements function *B* of class *B* that is in namespace *A* that is nested in another namespace *A*.
- ☒ This code implements the default constructor of class *B* that is in namespace *A* that is nested in another namespace *A*.

☐ This code does not compile because you can't have two nested namespaces both called *A*.

Question 4: What statement is true about the following code?

C++:

```
1 namespace A::B
2 {
3     class MyClass
4     {
5     };
6 }
```

- ☐ This code defines a class in namespace *A* that is nested in namespace *B*.
- ☐ This code defines a class in the single namespace called *A::B*.
- ☐ This code does not compile.
- ☒ This code defines a class in namespace *B* that is nested in

Question 5: What is encapsulation in the context of object-oriented programming?

- ☒ Bundling data with functionality that operates on that data.
- ☐ The process of writing a class.
- ☐ Hiding data from users.
- ☐ Compiling multiple classes in to one executable file.

Question 6: Which statement is false about boost tuples?

- ☐ A tuple makes it easy to make a function that returns more than one value.
- ☐ The tuple `get<>()` member function can be used to set the element values.
- ☒ A tuple defined for *n* elements can contain zero till *n* values.
- A tuple defined with *n* elements always contains *n* elements. Elements not set will have the default value.
- ☐ A tuple is a fixed-sized collection of elements which can each have a different type.

Question 7: What is the correct syntax to create a boost shared pointer to an *MyClass* object?

- ☒ `boost::shared_ptr<MyClass> mc(new MyClass);`

- ☐ boost::shared\_ptr<MyClass> mc=new MyClass;  
☐ boost::shared\_ptr<MyClass\*> mc(new MyClass);  
☐ boost::shared\_ptr<MyClass> mc(MyClass());

Question 8: Which statement is false about aggregation?

- ☐ Aggregation is often combined with delegation to delegate functionality to another object.  
☐ With aggregation you create an object that consists of one or more other objects.  
☒ Aggregation is a special kind of inheritance.  
☐ Aggregation can hide the functionality of the aggregated object.

Question 9: Which statement is false about global variables?

- ☒ When referring a global variable that is in another file, we need to declare that variable as external in that other file.  
☐ Static variables within a function are global variables that are only accessible in that function.  
☐ Global variables are accessible by all functions even if they are in another source file.  
☐ Static global variables are only accessible by static global functions.

Question 10: Which statement is false about boost variants?

- ☐ A variant can contain one value of a given collection of types.  
☐ When retrieving a value from the variant, we can use the `get<T>()` global function.  
☒ A variant can contain one value of any type.  
A variant can only contain a value of the types that were specified as template arguments when creating the variant.  
☐ A variant is like a type-safe C union and boost variant can contain class types while a union can only contain the build-in data types.

Question 11: Which statement is false about iterators?

- ☒ A `std::list<T>` supports a random access iterator.  
Since `std::list<T>` is a linked list, it cannot randomly jump in the structure. It can only move one step forward or backward to it only support a bi-directional iterator.  
☐ Iterators are a nested type of the data structure they iterate.  
☐ In a loop that traverses a data structure, you cannot compare the current iterator with the end iterator using the `<` operator

☐ Iterators are used to traverse data structures in a data structure independent way.

Question 12: Which statement is true about destructors and inheritance?

- ☐ Destructors must be virtual so that the base class destructor is called when a derived class object is removed from memory.
- ☒ Destructors must be virtual so that derived class object in a base class variable will be deleted correctly.
- ☐ Destructors must be virtual because the canonical header file requires that.
- ☐ The system provided destructor is always called correctly.

Question 13: Which statement is false about an abstract class?

- ☐ Abstract classes enable us to work uniformly with a set of related classes.
- ☐ Abstract classes can contain member data.
- ☒ Abstract classes can only have function declarations, no function implementations.
- ☐ Abstract classes cannot be instantiated.

Question 14: Which two statements are true about looping statements?

- ☒ The *do...while* loop is executed as long an expression is true.
- ☐ The *while* loop is executed one or more times.
- ☐ The *repeat...until* loop is executed until an expression becomes true.
- ☒ The *for* loop can always be used to replace a *while* loop.

Question 15: Which of the operator declarations below is the best way to support the index operator for integer indices (*int*)?

- ☐ Type operator [] (int index) const;
- ☐ const Type& operator [] (int index) const; Type operator [] (int index);
- ☒ const Type& operator [] (int index) const; Type& operator [] (int index);
- ☐ Type& operator [] (int index) const;

Question 16: Which statement is false about classes and objects?

- ☐ Objects have state, behaviour and identity
- ☐ Classes have state and behaviour
- ☒ Classes have state, behaviour and identity

- ☒ Classes have state, behaviour and identity  
☐ Objects have state and behaviour

Question 17: Which statement is true about predicates?

- ☐ A predicate is one of the two main parts of a sentence, the other being the subject, which the predicate modifies.  
☐ A predicate is a (function object or global function) that changes an element.  
☒ A predicate is a functor (function object or global function) that returns a boolean.

Functors returning a boolean are used by certain algorithms to test if an element meets a certain criteria. These function objects are indeed called predicates.

- ☐ A predicate is a brand of dog food.

Question 18: Which statement is true about the following C code?

C++:

```
1 | int i,j;  
2 | for (i=0, j=4; i<j;)  
3 | {  
4 |     printf("%d, %d\n", i++, j--);  
5 | }
```

- ☐ This code does not compile because the comma is not supported in the loop initialisation expression.  
☒ This code executes the loop code two times.  
☐ This code is an infinite loop.  
☐ This code does not compile because the *for* statement is missing the post iteration expression.

Question 19: 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 20: Which statements are false about algorithms?

☒ Removing algorithms are a special kind of mutating algorithms.

Removing algorithms are a special kind of modifying algorithms, not mutating algorithms.

☐ Mutating algorithms change the order of elements but not the elements themselves.

☐ STL algorithms accept a start- and end-iterator instead of the complete container.

☒ Modifying algorithms can modify the elements of data structures and change the order.

Modifying algorithms do not change the order of the data in a container.

Score: 18 (90.00%)

Pass/Fail: Passed

[Quiz](#) >

[Contact us](#) [Advertise](#) [Terms and rules](#) [Privacy policy](#) [Help](#) [Home](#) 

© QUANTNET INC