Answered by 2015011308 唐适之

# Q&A List

The aim of the QA list is to ensure that the key knowledge in this course is delivered to you. We will summarize the good answers from our students, and compile them into a booklet for your future reference. Some questions are relatively easy to answer. You may just copy and paste from our slides. The underlined questions are extended and challenging, which need more background knowledge than what we have learned in class. Please try to find out the answers by yourself.

Assume the question No. is x and the student No. is y, then each student only needs to answer the following questions: {x | x % 5 == y % 5}.

Besides answering the required questions, each student is encouraged to propose other questions and the corresponding answers. Throughout this semester, students with enough GOOD new questions along with GOOD answers will be granted up to 5 bonus credits.

## Lecture 5

* Inline function

1. Why do we need C++ grammars like function-like macros and inline functions? Try to describe a typical situation where the two grammars are used.
2. What is wrong with the following code?

#define IS\_UPPER(C) ( (C) >=‘A’ && (C) <=‘Z’ )

while ( IS\_UPPER ( s=getchar() ) )

1. Please give an example showing why brackets “()” are important in avoiding errors in function-like macros.

If not using brackets,

**#define inc(a) a + 1**

inc**(1) \*** **2** **==** **1** **+** **1** **\*** **2** **==** **3**

while if using brackets,

**#define inc(a) (a + 1)**

inc**(1) \*** **2** **== (1** **+** **1) \*** **2** **==** **4**

1. Please state the main advantages of inline functions over function-like macros.
2. Is the keyword ***inline*** below necessary? Please explain this grammar.

class A {

int x\_;

public:

inline int setX(int x) { x\_ = x; }

inline int getX() { return x\_; }

};

1. Where is the keyword ***inline*** supposed to be added? Is it in declarations or definitions? Please try to explain why.
2. Can we add keyword ***inline*** for every function? Please state the rule for setting inline functions (e.g., from google coding style).
3. How is ***inline*** related to the C++ ***class*** grammar?

Functions defined in a class definition block will be inlined by default, but functions only declared in the class definition block have to be inlined manually.

* Namespace

1. Why do we need ***namespace***? Please try to describe some situations where namespaces are needed.
2. Please enumerate all the ways of creating a namespace and referring to the names in the namespace using some examples.
3. What is the scope (作用域) of “***using namespace***”? Are you supposed to use it in header files? Please explain the reason.

* Static

1. Please state the lifetime (生命期) of static variables in detail. How is ***static*** related to the C++ ***class*** grammar?
2. Where are static variables stored? Are they stored on the heap or stack?

They are stored in static memory, neither in the heap nor the stack.

1. Can we access a static variable defined in another source file directly? What should we do to do so? Please give an example.
2. Please state the concept of internal linkage (内部链接), and list all the different types of variables with the internal linkage.
3. Can we define two static variables with the same name in two different source files?
4. What are the differences between static and non-static member functions?
5. Please list all the ways to access static member functions and static member variables, preferably with examples.

Suppose we have the class below:

class A

**{**

public**:**

**static int** x**;**

**static void** f**();**

**void** g**();**

**};**

**int** A**::**x**;**

We can access them as below:

1. Through “::”:

A**::**x

or

A**::**f()

1. Directly access them in member functions:

**void** A**::**g**()**

**{**

x**;**

// or

f**();**

**}**

1. Please give an example of object counting (对象计数). Please state the necessity of object counting and try to describe a typical situation of using object counting.

* Singleton

1. Please give a complete example of singleton design pattern.
2. Please explain the necessity of singleton and try to describe some typical situations of using singletons.

# Lecture 6

* Reference

1. State the basic concept of reference.
2. Is it allowed to declare a reference variable without defining it? Is it allowed to change the value of a reference variable?
3. Please try to explain the advantages of ***reference*** over ***pointer***.
4. A reference can avoid accessing to NULL pointer.
5. When changing a function from passing a copy to passing a reference, the function call statement doesn’t have to be modified.
6. When access to the object referred to, we don’t have to use “\*” when using reference. We can access it like a normal variable instead.
7. Programmers won’t modify the address a reference refer to by accident.
8. Can we return a reference of a local variable in a function? Why? Please state the advantages of returning references rather than values in functions.
9. How to use references as member variables? Please give a complete example.
10. Can references be used as elements in ***std*** containers such as ***std::vector***? How about pointers?
11. Please list some ways of returning multiple values simultaneously from functions.
12. Please state the differences between passing references (引用传参) and passing values (值传参) as function arguments.
13. When passing a value, it will create a new block of memory in stack and copy the object into it. When passing a reference, it will just create a reference that refers to the original object. The former is time wasting and may use to much of the stack memory. But passing a value allow users to modify it in the function but not affect the original value.
14. When using polymorphism, passing a value is not allowed, for it will cause slicing.
15. Please state the least privilege principle preferably with examples.
16. What is the difference between rvalue references and lvalue references? Please state the basic concept and usage of rvalue references.

* Copy constructor

1. Please give an example of copy constructor.
2. How is ***reference*** related to the C++ ***class*** grammar?
3. Please list the situations where copy constructors are called. What if you do not define a copy constructor by yourself?

When constructing an object with an existing object, it will call the copy constructor. One can achieve this by explicitly passing a value (T for type T), or an L-value reference (T& for type T) to the constructor, or using implicit or explicit type converting.

This includes the situation of defining a new local variable or passing a value to a function parameter.

When no constructor is defined explicitly, a default constructor that copy every bytes of the object will be defined. This will cause error when there are resources opened in the object, such as pointers pointing to blocks of memory.

1. What does the compiler-generated copy constructor do? What is the fatal disadvantage of depending on the compiler-generated copy constructor? Please give an example of the fatal error.
2. Why should we avoid copy constructors as much as possible? What should we do to ensure that copy constructor cannot be called?
3. Please state the differences between move constructors and copy constructors.

# Lecture 7

* Composition
  1. Please state the connotation (内涵) of composition. Please try to describe a car using composition preferably with an example. And the example should be at least with the declaration and definition of constructors and destructor.
  2. Please list the different ways (i.e., public, protected, private) of composition and discuss their safety.
  3. Please state the differences between aggregation (聚集) and borrow (借用) in composition. Please give examples for the two different ways.

In aggregation, the parent object takes charge of the lifetime of the child object. The parent constructs and destructs the child.

In borrowing, the parent just saves a reference to the child. The child is constructed and destructed by other objects.

* 1. Please state the basic concept and usage of aggregate initialization.
  2. Please state the order of constructing and destructing member objects. Please try to explain why the order is like that.
  3. Please give an example showing how the constructor initializer list (构造函数初始化列表) is used in composition.
  4. Is it recommended that class members be initialized by the constructor initializer list (构造函数初始化列表) in the same order as that they are declared in the class? Why? In which order are the members actually initialized? As the order in the constructor initializer list or their declaration order?
* Inheritance

1. Please state the connotation (内涵) of inheritance. Please give an example of inheritance at least with the declaration and definition of constructors and destructor.
2. Please state the differences between public, protected and private inheritance preferably with a form (表格).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Public members in base class | Protected members in base class | Private members in base class |
| Public inheritance | Will be public in derived class | Will be protected in derived class | Won’t inherit |
| Protected inheritance | Will be protected in derived class | Will be protected in derived class | Won’t inherit |
| Private inheritance | Will be private in derived class | Will be private in derived class | Won’t inherit |

1. Please state the order of constructing and destructing of a derived class, and try to explain the order.
2. Which functions cannot be inherited from the base class? Please try to explain the reason.
3. Please state the differences between public inheritance and private inheritance.
4. When shall we use private inheritance?
5. How to re-declare a private member function inherited from base class to be public?

use “using” statement. For example:

class A

**{**

public**:**

**void** f**();**

**};**

class B **:** private A

**{**

public**:**

**using** A**::**f**;**

**};**

1. Please state the base concept of incremental development.
2. Please state the concept, usage and rules of multiple inheritance.

* Code reuse

1. Please list the common ways of code reuse, and try to compare between these ways.