OOP HW1

Notes

2026 年 4 月 24 日

1 C++ OOP and Namespaces Questions

Question 8

Which concept of OOP is false for C++?

- A. Code can be written without using classes
- **B. Code must contain at least one class**
- C. A class must have member functions
- D. At least one object should be declared in code

解析：正确选项是 **B**。C++ 是一门多范式（Multi-paradigm）语言。它不仅支持面向对象，还完全向下兼容 C 语言的面向过程编程。你可以写一个只有 `main()` 函数和几个普通计算程序的 `.cpp` 文件，里面不定义任何类，代码依然能完美编译和运行。因此，“代码必须包含类”这个说法在 C++ 中是绝对错误的。

Question 18

Which feature allows open recursion, among the following?

- **A. Use of this pointer**
- B. Use of pointers
- C. Use of pass by value
- D. Use of parameterized constructor

解析：正确选项是 **A**。“开放递归（Open Recursion）”是面向对象中的一个术语，指的是在类的内部，一个成员函数去调用另一个成员函数。在 C++ 编译器底层，这种类内部的函数相互调用，实际上都是通过隐藏的 `this` 指针（例如隐式执行了 `this->anotherFunction()`）来完成的。这也是实现多态时，基类调用虚函数能正确路由到子类实现的关键机制。

Question 21

Which among the following is false, for a member function of a class?

- A. All member functions must be defined
- B. Member functions can be defined inside or outside the class body
- **C. Member functions need not be declared inside the class definition**
- D. Member functions can be made friend to another class using the friend keyword

解析：正确选项是 **C**。C++ 的语法严格要求，类的所有行为规范必须在类的 `{}` 内部声明清楚。你可以把成员函数的**具体实现（函数体）**写在类的外面（使用作用域解析符 `ClassName::`），但你**必须**在类的大括号里面先写上它的**声明（函数原型）**。如果不声明，编译器将无法识别该函数属于这个类。

Question 32

Abstraction principle includes_____

- A. Use abstraction at its minimum
- B. Use abstraction to avoid longer codes
- **C. Use abstraction whenever possible to avoid duplication**
- D. Use abstraction whenever possible to achieve OOP

解析：正确选项是 **C**。抽象的核心目的不仅是隐藏细节，更是为了提炼共性。当你的代码中出现多处类似的逻辑时，软件工程的 DRY (Don't Repeat Yourself) 原则建议你将这些重复的部分“抽象”成一个公共的函数、类或基类。这能大幅提高代码的复用性和可维护性。

Question 33

Encapsulation and abstraction differ as _____

- **A. Binding and Hiding respectively**
- B. Hiding and Binding respectively
- C. Can be used any way
- D. Hiding and hiding respectively

解析：正确选项是 **A**。这是一道非常经典的区分题：**封装（Encapsulation）**侧重于**绑定（Binding）**，它把数据（属性）和操作数据的方法捆绑成一个不可分割的整体；而**抽象（Abstraction）**侧重于**隐藏（Hiding）**，它向外界隐藏内部复杂的运作机制，只暴露出最简单、必要的接口。

Question 34

In terms of stream and files _____

- **A. Abstraction is called a stream and device is called a file**
- B. Abstraction is called a file and device is called a stream
- C. Abstraction can be called both file and stream
- D. Abstraction can't be defined in terms of files and stream

解析：正确选项是 **A**。在 C++ 的 I/O 系统中，“流（Stream）”是一个纯粹的**逻辑抽象**概念，代表数据的流动过程（如 `cin`, `cout`）。而“文件（File）”或者显示器、键盘等，则是具体存在的**物理设备（Device）**。我们通过操作“流”这个抽象层，来实现对物理设备的统一数据读写。

Question 37

Which among the following is not a level of abstraction?

- A. Logical level
- B. Physical level
- C. View level
- **D. External level**

解析：正确选项是 **D**。这道题的背景源自计算机科学中经典的数据抽象三层架构：1. **物理层（Physical level）**：数据在底层的实际存储方式；2. **逻辑层（Logical level）**：数据的结构与关系；3. **视图层（View level）**：用户界面上展示的特定数据子集。“External level（外部层）”通常在数据库领域的某些特定标准中作为 View level 的同义词，但在这种标准的三层架构单选题中，它被视为非标准层级或干扰项。

Question 40

Identify the correct statement.

- **A. Namespace is used to group class, objects and functions**
- B. Namespace is used to mark the beginning of the program
- C. A namespace is used to separate the class, objects
- D. Namespace is used to mark the beginning & end of the program

解析：正确选项是 **A**。命名空间（Namespace）的作用类似于文件系统中的“文件夹”。在大型工程中，为了防止不同的模块产生命名冲突（Name Collision），我们使用命名空间将相关的类、对象、变量和函数包裹并分组在一起，实现隔离和管理。

Question 41

What is the use of Namespace?

- A. To encapsulate the data
- **B. To structure a program into logical units**
- C. Encapsulate the data & structure a program into logical units
- D. It is used to mark the beginning of the program

解析：正确选项是 **B**。这与第 40 题逻辑一致。类的主要目的是封装数据和方法（Encapsulate the data），而命名空间则是在更高的宏观维度上，将程序的代码架构划分为不同的“逻辑单元”或“模块”（例如 C++ 标准库的所有内容都被组织在 `std` 这个逻辑单元里）。

Question 47

What will be the output of the following C++ code?

```
#include <iostream>
using namespace std;
namespace extra {
    int i;
}
void i() {
    using namespace extra;
    int i;
    i = 9;
    cout << i;
}
int main() {
    enum letter { i, j };
    class i { letter j; };
    ::i();
    return 0;
}
```

- **A. 9**
- B. 10
- C. compile time error
- D. 11

解析：正确选项是 **A**。这道题考察的是作用域解析和变量覆盖（Shadowing）。在 `main` 函数中，`::i()` 前面的 `::` 告诉编译器忽略局部作用域里的枚举 `i` 和类 `i`，直接调用全局函数 `void i()`。在函数 `i()` 内部，虽然引入了 `extra` 命名空间，但紧接着声明的局部变量 `int i;` 优先级更高，直接屏蔽（Shadow）了命名空间里的 `i`。因此，`i = 9` 修改的是局部变量，最终输出 9。

Question 54

What will be the output of the following C++ code?

```
#include <iostream>
using namespace std;
namespace {
    int var = 10;
}
int main() {
    cout << var;
}
```

- **A. 10**
- B. Error
- C. Some garbage value
- D. Nothing but program runs perfectly

解析：正确选项是 **A**。代码中使用的是**匿名命名空间（Unnamed Namespace）**。匿名命名空间中的变量和函数自动具有内部链接属性（类似于 `static` 全局变量），它们的作用域被严格限制在当前文件中。在当前文件的 `main()` 函数中，可以直接通过变量名 `var` 访问它而无需任何前缀，因此程序正常输出 10。