CPP Final Fa24 @NJUSE

> by Fradow

一、简答题（12 × 5'）

1. 简述 C++ 的内存管理机制。栈区、堆区、静态存储区的区别？
2. C++ 中数组和指针的区别？如何用指针操作数组元素？
3. 虚函数如何实现多态？如何确保调用正确的函数？
4. C++ 中隐式类型转换和显式（强制）类型转换的区别？
5. 拷贝构造函数与移动构造函数的区别？何时调用拷贝构造函数？
6. 在 C++ 中，什么是 RAII？举出例子，这种约定在保证异常安全性方面的作用？
7. 什么是“对象切片”？给出实例，原因及解决方法？
8. 内联函数与宏的区别？内联函数的优点和缺点？
9. C++ 如何解决“菱形继承”问题？
10. 成员初始化列表的作用？相较于在构造函数体中赋值有何优势？
11. `constexpr` 的作用？与传统常量 `const` 有何不同？
12. Lambda 函数与函数对象的区别？

二、代码阅读题（10 × 3'）

> 与原题代码仅大意相同

正确则写结果；错误则写原因。代码均默认已有 `#include <iostream>`。

1

```
class Base {
 public:
  Base(int x) : x(x) {}
  ~Base() {
    std::cout << "Base destructor\n";
  }
  void show() {
    std::cout << x << '\n';
  }
 private:
  int x;
};

class Derived : public Base {
 public:
  explicit Derived(int x, int y) : Base(x), y(y) {}
  ~Derived() {
    std::cout << "Derived destructor\n";
  }
  void show() {
    std::cout << y << '\n';
```

```
  }
 private:
  int y;
};

int main() {
  Base *b = new Derived(1, 2);
  b->show();
  delete b;
  return 0;
}
```

2

```
void foo(int &&x) {
  std::cout << x << '\n';
}

int main() {
  int x = 10;
  foo(std::move(x));
  std::cout << x << '\n';
  return 0;
}
```

3

```
class A {
 public:
  A() {
    std::cout << "Constructor" << '\n';
  }
  ~A() {
    std::cout << "Destructor" << '\n';
  }
  void foo() {
    throw 1;
  }
};

int main() {
  try {
    A a;
    a.foo();
  } catch (...) {
    std::cout << "Exception" << '\n';
  }
  return 0;
}
```

4

```
void foo(int x, int y = 2) {
  std::cout << x + y << '\n';
}

void foo(int x, double y = 3.0) {
  std::cout << x + y << '\n';
}

int main() {
  foo(1);
  return 0;
}
```

5

```
int main() {
  char *str = "hello";
  str[0] = 'H';
  std::cout << str << '\n';
}
```

6

```
int main() {
  int x = 10;
  int y = 20;
  int * const p = &x;
  *p = y;
  std::cout << *p << '\n';
  return 0;
}
```

7

```
class Base {
 public:
  Base() {
    std::cout << "Base constructor" << '\n';
  }
  virtual ~Base() {
    std::cout << "Base destructor" << '\n';
```

```
  }
  virtual void show() const {
    std::cout << "Base show" << '\n';
  }
};

class Derived : public Base {
 public:
  Derived() {
    std::cout << "Derived constructor" << '\n';
  }
  ~Derived() override {
    std::cout << "Derived destructor" << '\n';
  }
  void show() override {
    std::cout << "Derived show" << '\n';
  }
};

int main() {
  Base *b = new Derived();
  b->show();
  delete b;
  return 0;
}
```

8

```
class FileError {};
class NotFoundError : public FileError {};

void foo() {
  throw NotFoundError();
}

int main() {
  try {
    foo();
  } catch (FileError &e) {
    std::cout << "FileError" << '\n';
  } catch (NotFoundError &e) {
    std::cout << "NotFoundError" << '\n';
  }
  return 0;
}
```

9

```
class Animal {
 public:
  void walk() {
    std::cout << "walk" << '\n';
  }
};

class Bird : protected Animal {
 public:
  void fly() {
    walk();
    std::cout << "fly" << '\n';
  }
};

int main() {
  Bird bird;
  bird.fly();
  bird.walk();
  return 0;
}
```

10

```
class Vehicle {
  public:
    virtual ~Vehicle() = default;
    virtual void drive() {
      std::cout << "Vehicle driving" << '\n';
    }
};

class Car : public Vehicle {
  public:
    void drive() override {
      std::cout << "Car driving" << '\n';
    }
};

template <typename T>
void foo(T t) {
  t.drive();
}

int main() {
  Car car;
  foo(car);
  return 0;
}
```

三、编程题（10'）

实现二维矩形类 `Rectangle2D` 满足以下要求：

1. 含整型成员变量 `length`，`width`。
2. 重载乘法操作符 `*`，实现长、宽分别与同一个整型相乘 ，返回一个新的 `Rectangle2D`。
3. 重载重定向操作符 `<<`，向输出流输出 `Rectangle2D` 的属性，格式为 `[Rectangle: Length = X, Width = Y]`。

实现三维长方体类 `Cuboid3D` 满足以下要求：

1. 继承 `Rectangle2D`，含整型成员变量 `height`。
2. 重载乘法操作符 `*`，实现长、宽、高分别与同一个整型相乘 ，返回一个新的 `Cuboid3D`。
3. 重载重定向操作符 `<<`，向输出流输出 `Cuboid3D` 的属性，格式为 `[Cuboid: Length = X, Width = Y, Height = Z]`。