复习材料：Namespaces 和 Nested Classes (第七章)

1. Namespaces

Namespaces 是 C++ 提供的一种机制，用来避免名字冲突，特别是在大型项目中，可能会有多个函数、类或者变量使用相同的名称。通过将这些符号放入不同的命名空间，可以保证在不同的作用域内使用相同的名称而不冲突。

作用：

- 避免命名冲突
- 提供逻辑上的模块化

命名空间语法：

```
namespace myNamespace {
    int myVariable;
    void myFunction() {
        // 函数实现
    }
}
```

使用命名空间中的元素可以通过以下方式：

```
myNamespace::myFunction();
```

也可以使用 `using` 语句：

```
using namespace myNamespace;
myFunction();
```

2. 示例：使用 Namespaces

在定义 `Pressure` 枚举类和相关操作符时，我们将它们放入 `enumerations` 命名空间中：

```
namespace enumerations
{
    enum class Pressure {
        Lo,
        Med,
        Hi,
        Pop
    };

    std::ostream& operator<<(std::ostream&, const Pressure&);
    Pressure& operator++(Pressure&);
    Pressure operator++(Pressure&, int);
}
```

在 `main` 函数中，使用 `using` 语句来访问命名空间中的元素：

```
using enumerations::Pressure;

int main() {
    Pressure pressure = Pressure::Lo;
    for (int i = 0; i < 4; i++) {
        std::cout << pressure << " ";
        ++pressure;
    }
    return 0;
}
```

3. Nested Classes

Nested Classes（嵌套类） 是指将一个类的定义放在另一个类的内部，这样可以将内部类的生命周期限制在外部类的作用域内，增强类之间的关联性和封装性。

嵌套类的使用场景包括：

- 将实现类隐藏在外部类的封装内
- 增强类的层次结构，逻辑上将它们组织在一起

嵌套类的定义：

```
class OuterClass {
public:
    class NestedClass {
    public:
        void display() {
            std::cout << "I am a nested class!" << std::endl;
        }
    };
};
```

嵌套类的对象可以通过外部类来访问：

```
OuterClass::NestedClass nestedObj;
nestedObj.display();
```

4. Studio 练习与问题总结

Q2 - 使用命名空间：
在 `Pressure` 枚举类中，我们通过 `enumerations` 命名空间来管理这些枚举的范围。在 `main` 函数中，使用 `using` 关键字来引入特定的命名空间成员，例如 `Pressure`，避免了每次使用时都要加上 `enumerations::` 的前缀。

```
using enumerations::Pressure;
Pressure pressure = Pressure::Lo;
```

Q3 - 嵌套类的使用：
嵌套类通常用于将实现类隐藏在外部类的封装内部，从而提供更高的封装性。在本练习中，可以通过嵌套类来定义内部的复杂操作逻辑，并仅对外暴露有限的接口。

5. 重点总结：Namespace 与 Nested Classes

1. **Namespace**：通过命名空间将符号（函数、类、变量等）分区，避免名称冲突。
    - 可以使用 `namespace` 关键字定义命名空间。
    - `using` 语句可以简化命名空间的使用。
2. **Nested Class**：嵌套类用于将一个类的定义包含在另一个类中，用于增强逻辑上的封装和关联性。
    - 嵌套类的生命周期由外部类控制。

考题示例

考题示例 1:
“What is a namespace in C++ and why would you use it?”

答案:
命名空间（ `namespace` ）用于将符号分区，避免命名冲突。特别是在大型项目中，可能有多个函数、类或变量使用相同的名称，通过将它们放入不同的命名空间中，可以确保它们不会相互冲突。

考题示例 2:
“What is the syntax for defining and using a nested class in C++?”

答案:
嵌套类是在一个类的内部定义的类，其生命周期由外部类控制。语法如下：

```
class OuterClass {
public:
    class NestedClass {
        void display() {
            std::cout << "I am a nested class!" << std::endl;
        }
    };
};
```

代码与问题总结示例

```
namespace enumerations
{
    enum class Pressure {
        Lo,
        Med,
        Hi,
        Pop
    };

    std::ostream& operator<<(std::ostream&, const Pressure&);
    Pressure& operator++(Pressure&);
    Pressure operator++(Pressure&, int);
}

int main() {
    using enumerations::Pressure;
    Pressure pressure = Pressure::Lo;

    for (int i = 0; i < 4; i++) {
        std::cout << pressure << " ";
        ++pressure;
    }

    return 0;
}
```

通过使用 `enumerations` 命名空间，我们将枚举 `Pressure` 的作用域限制在命名空间内，使得程序更加模块化和清晰。