# Week6

## a. What is relation between pointers and references?

在C++中，指针 (pointers) 和引用 (references) 本质上是高度相关的概念。从编译器最终生成的机器码层面来看，它们几乎是相同的。引用可以被视为指针的一种扩展，或者更贴切地说，引用通常是指针的“伪装”。它是在指针之上的一层语法糖 (syntax sugar)，旨在让代码更易于阅读和理解。所有能用引用实现的功能，也都能用指针来完成。

## b. Are references themselves new variables? Do they occupy memory?

不，引用本身并不是一个新的变量。它们通常不占用独立的内存空间，也没有自己的存储位置。引用只是存在于我们的源代码中；当你编译代码时，并不会为引用创建一个新的变量实体。例如，如果你有一个变量 `a` 和一个指向 `a` 的引用 `ref`，编译后只会存在变量 `a`。

## c. What is difference between pass by value and pass by reference?

这两种方式的核心区别在于函数参数的传递方式是否会影响原始变量。

- **传值 (Pass by value)**：当你以传值方式将变量传递给函数时，实际上是该变量值的一个**副本 (copy)**被传入函数。函数内部会创建一个全新的变量来存储这个副本。因此，在函数内部对这个参数的任何修改，都不会影响到函数外部的原始变量。
- **传引用 (Pass by reference)**：当你以传引用方式传递变量时，你允许函数直接操作原始变量本身。这样，函数内部对参数的修改会直接反映在原始变量上。

## d. How does the compiler treat a reference?

编译器在处理引用时，会根据上下文将其视为对原始变量的直接操作。

- 在作为别名 (alias) 的简单场景下，如果一个引用 `ref` 指向变量 `a`，那么源代码中对 `ref` 的操作在编译后会被直接转换为对 `a` 的操作。
- 在作为函数参数的场景下，引用在底层的实现与指针是完全相同的。编译器会生成与传递指针时一样的机器码，只是在语法层面为开发者隐藏了指针的复杂性。

## e. Explain the advantages of using a reference over using a pointer?

在可以选择的情况下，使用引用通常比使用指针有以下优势：

- **语法更简洁**：使用引用可以显著减少代码量和语法上的修饰。你不需要像使用指针那样，在传递参数时使用取地址运算符 (`&`)，在操作时也无需使用解引用运算符 (`*`)。
- **可读性更高**：由于语法更简洁，使用引用的代码看起来更加干净、清晰，也更容易阅读和理解。

## f. Can you create an uninitialized reference first and later use it to reference a variable?

不可以。C++的规则要求，一个引用在被声明时**必须**立即进行初始化，使其指向一个已存在的变量。你不能创建一个未初始化的引用。此外，一旦一个引用被初始化，它就不能再被修改为指向另一个不同的变量。

## g. Why do you need a class instead of a bunch of variables?

当需要表示一个复杂的实体（例如游戏中的玩家）时，使用一堆零散的变量会带来许多问题：

- **混乱且无组织**：这些变量没有被组织在一起，只是散落在代码中，显得非常混乱和无组织。
- **难以维护**：如果需要表示多个实体（例如两个玩家），就必须复制所有变量，这使得代码难以维护和理解。
- **函数签名冗长**：如果要编写一个操作这些数据的函数，你需要将所有相关的变量作为参数传入，这会产生大量的代码，也变得难以维护。

使用 `class` 可以将所有相关的变量（数据）整合到一个单一的类型中，从而极大地清理代码结构，并让所有变量集中在一个地方。

## h. What does a class consist of?

一个 `class` 是一个将**数据 (data)**和**功能 (functionality)**组织在一起的方式。它主要由两部分组成：

- **数据成员**：即类中定义的变量。
- **成员函数 (Methods)**：即定义在类内部的、用于操作这些数据成员的函数。

## i. What is the purpose of introducing classes in C++?

引入 `class` 的主要目的并不是为了提供任何你原本无法实现的新功能。其核心目的是为了让程序员的生活更轻松。`class` 本质上是一种语法糖 (syntactic sugar)，它提供了一种强大的工具来**组织代码**，使其**更干净、更易于维护**。

## j. What is the technical difference between a class and a struct?

从技术上讲，`class` 和 `struct` 之间只有**一个**微小的区别，这个区别在于其成员的默认可见性 (visibility)。

- 在 `class` 中，如果你不指定任何可见性修饰符，其所有成员默认为 `private` (私有)。
- 在 `struct` 中，其所有成员默认为 `public` (公开)。

除此以外，两者在技术上没有其他任何区别。

## k. Why do we keep the use of the struct in C++?

C++ 保留 `struct` 关键字的唯一原因是为了**维持对 C 语言的向后兼容性 (backwards compatibility)**。C 语言没有 `class` 的概念，但它有 `struct` (结构体)。如果 C++ 完全移除 `struct` 关键字，那么大量的 C 代码将无法在 C++ 编译器中正常编译，从而破坏了两者之间的兼容性。