**1. How is this Object-Oriented Programming (OOP) in C++? Where is the class?**

In C++, **struct** and **class** are almost identical, with one key difference:

* By default, members of a struct are **public**.
* By default, members of a class are **private**.

Since your struct node contains both **data members** (data and next) and **a constructor**, it's actually behaving like a **class** in an Object-Oriented Programming (OOP) manner.

In C++, unlike C, a struct can **have member functions**, **constructors**, and **access specifiers** (public, private, protected). So yes, you **can** use public inside a struct! That’s why your struct behaves just like a class, making it object-oriented.

**2. Difference between these two statements:**

node \* y = new node(2, nullptr);

node x = node (3, nullptr);

**✅ node \* y = new node(2, nullptr);**

* This dynamically allocates memory for a **new object of type node** on the **heap**.
* new returns a **pointer** to the allocated object (node\*), which is stored in y.
* Since y is a pointer, you must use y->data or y->next to access members.

**✅ node x = node(3, nullptr);**

* This **directly creates an object** x of type node on the **stack**.
* Memory is allocated **statically**; x is an **actual object** rather than a pointer.
* You access members using x.data or x.next because x is not a pointer.

**3. Memory Allocation**

| **Statement** | **Memory Type** | **Object or Pointer** |
| --- | --- | --- |
| node \* y = new node(2, nullptr); | **Heap** (Dynamic) | Pointer (y) to object |
| node x = node(3, nullptr); | **Stack** (Automatic) | Actual object (x) |

**4. Valid Operations**

**✅ For y (Heap Allocation, Pointer)**

cout << y->data << endl; // Access member via pointer

delete y; // Free memory manually

**✅ For x (Stack Allocation, Direct Object)**

cout << x.data << endl; // Access member directly

// No need for delete, memory is automatically freed when out of scope

**5. When to Use Heap vs Stack?**

* **Heap Allocation (new)** is useful when objects need to live beyond function scope.
* **Stack Allocation (Normal Object)** is simpler but limited to the function’s lifetime.

Since C only uses structs **without constructors or member functions**, these extra features in C++ might seem surprising!

Hope this helps! Let me know if anything needs further explanation. 🚀