Constructor Inheriting

In this lesson, we'll discuss how constructor calls are initiated when we call an object in the main section.

WE'LL COVER THE FOLLOWING

Constructor calls

When we call a constructor, a series of constructor calls may be triggered. This guarantees that each base object is properly initialized. The sequence of constructor calls starts with the base class and ends with the most derived class.

Example

Let's check how the series of calls is initiated when an object is called:

```
struct A{};
struct B: A{};
struct C: B{};
C c;              // A -> B -> C
```

Inheriting base class constructors

By using the declaration, a class inherits all constructors of its direct base class.

> Only the default constructor is inherited, the copy and move constructor will not be inherited.

```
class Account{
   public:

        Account(double amount{};
};

class BankAccount: public Account{
   public:
        using Account::Account;
};

BankAccount bankAccount(100.0);
```

The derived class inherits all the characteristics of the constructors:

- `public`, `protected`, and `private` access specifiers.
- `explicit` and `constexpr` declarations.

Default arguments for parameters of the constructor of a base class will not be inherited.

The derived class gets an additional constructor, having one parameter for the default argument. Constructors with the same parameters as constructors in the derived class will not be inherited.

> By inheriting constructors from the base class, there is the danger of forgetting to initialize the attributes of the derived class.

In the next lesson, we'll look at an example of constructor inheriting.