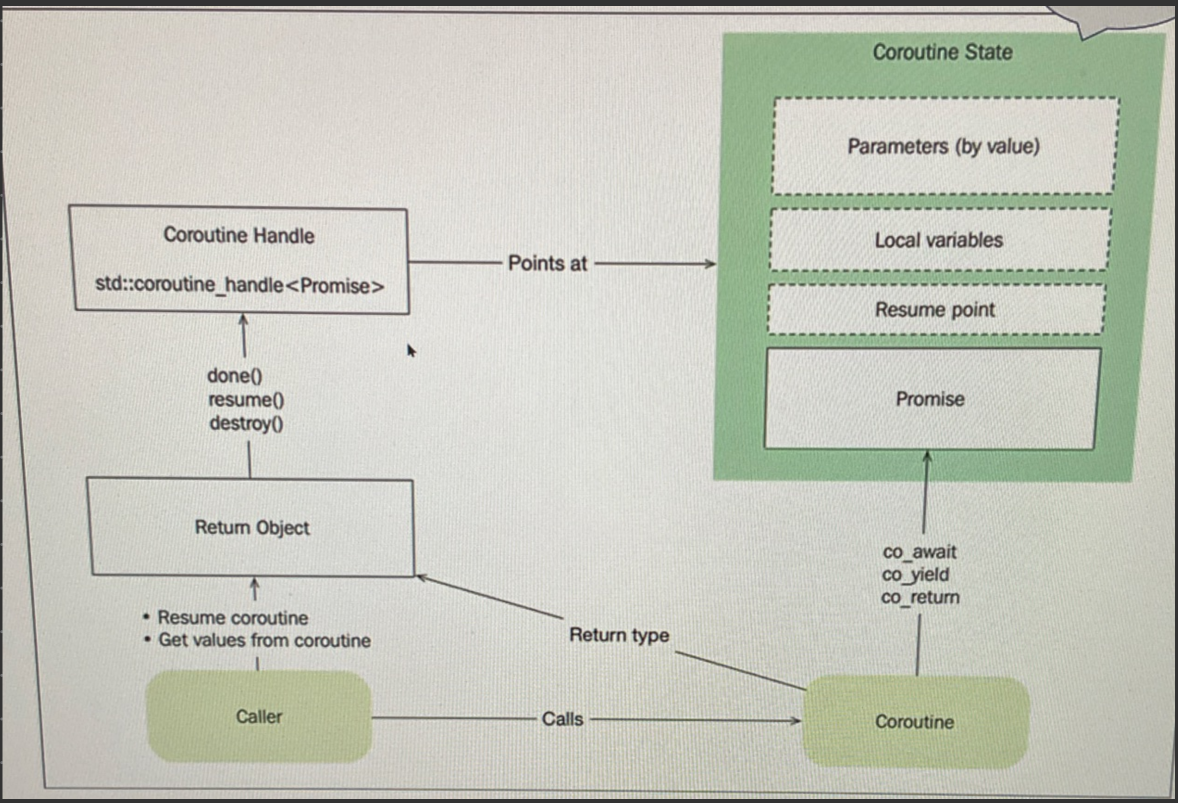
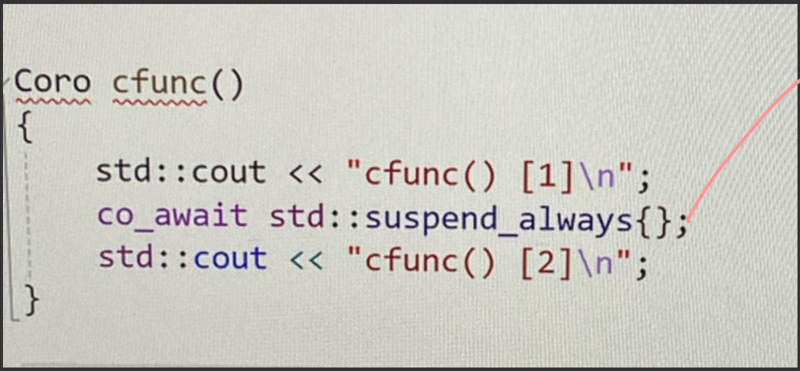
***Coroutine Handle:** • Derleyici coroutine state'i oluştururken, içindeki promise nesnesini de oluşturur.

promise nesnesini oluştururken, aynı zamanda coroutine state'i gösteren, Coroutine Handle nesnesini de oluşturur.

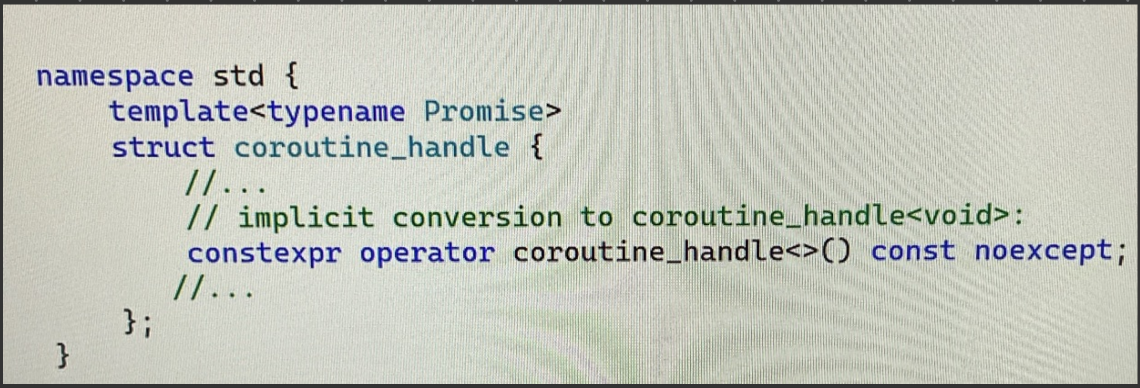
• Coroutine interface nesnesinin, coroutine state ile ilişkilendirilmek için, coroutine interface'in veri elemanı olarak, coroutine handle!
• Bu sınıfı biz yazmıyoruz!





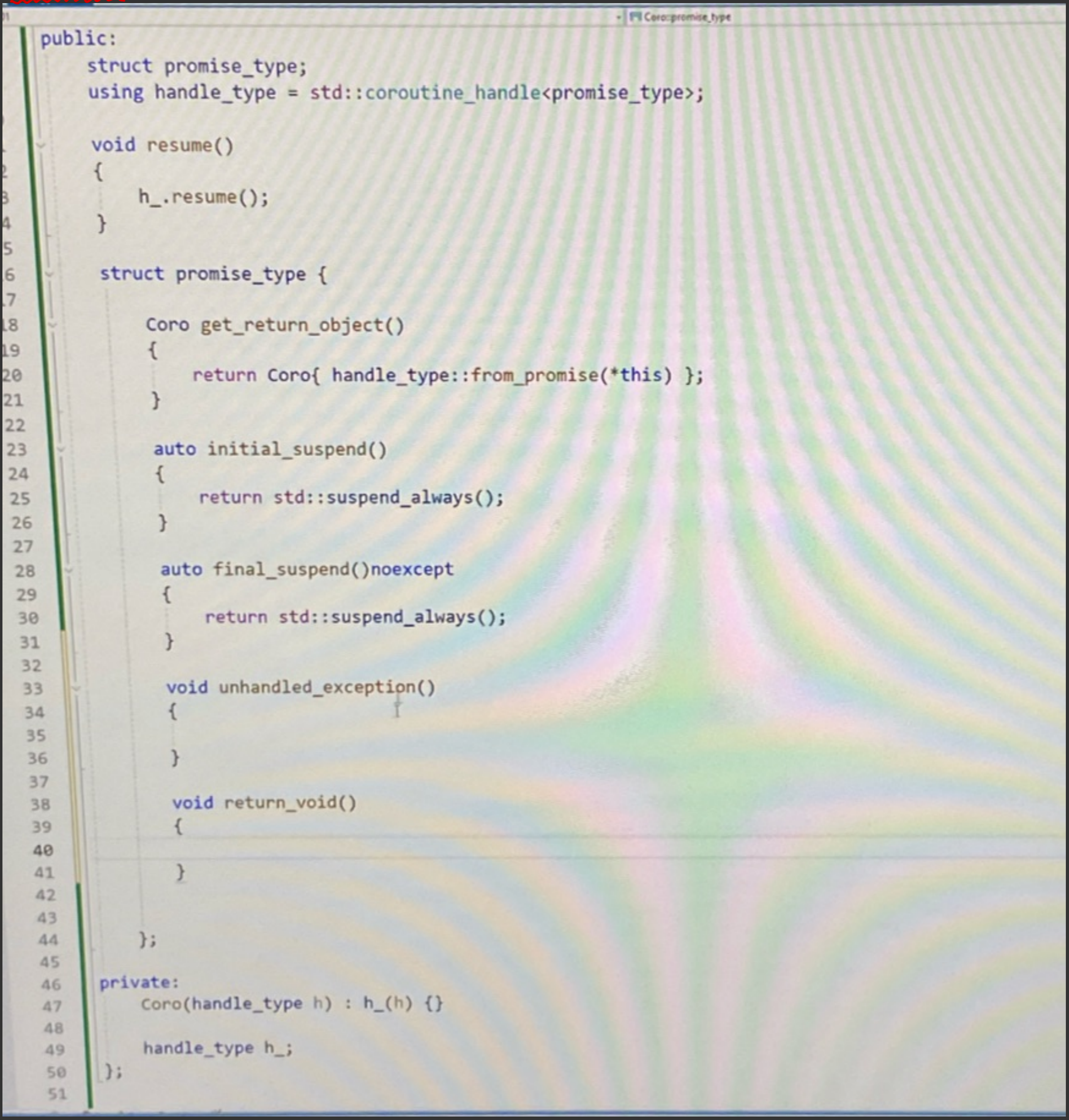
```cpp
Coro cfunc()
{
    std::cout << "cfunc() [1]\n";
    co_await std::suspend_always{};
    std::cout << "cfunc() [2]\n";
}
```

suspend edildikten sonra resume edebilmek için coroutine handle'a ihtiyaç duyarız!



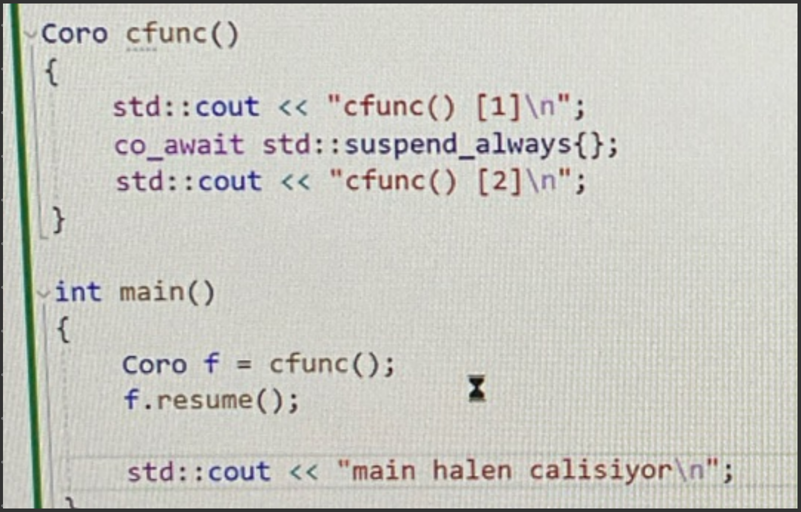
```cpp
namespace std {
    template<typename Promise>
    struct coroutine_handle {
        //...
        // implicit conversion to coroutine_handle<void>:
        constexpr operator coroutine_handle<>() const noexcept;
        //...
    };
}
```

Coro Interface:

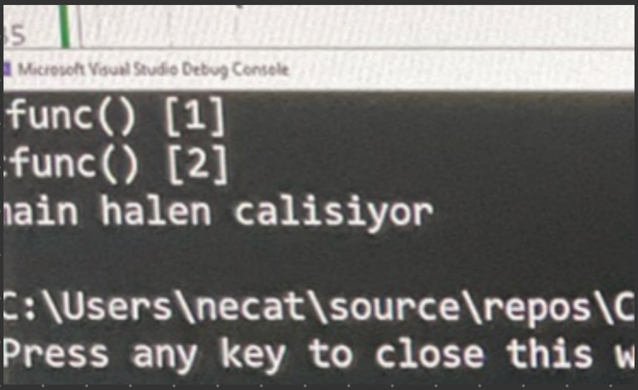


Coro::promise_type

```cpp
public:
    struct promise_type;
    using handle_type = std::coroutine_handle<promise_type>;

    void resume()
    {
        h_.resume();
    }

    struct promise_type {

        Coro get_return_object()
        {
            return Coro{ handle_type::from_promise(*this) };
        }

        auto initial_suspend()
        {
            return std::suspend_always();
        }

        auto final_suspend()noexcept
        {
            return std::suspend_always();
        }

        void unhandled_exception()
        {

        }

        void return_void()
        {

        }


    };

private:
    Coro(handle_type h) : h_(h) {}

    handle_type h_;
};
```



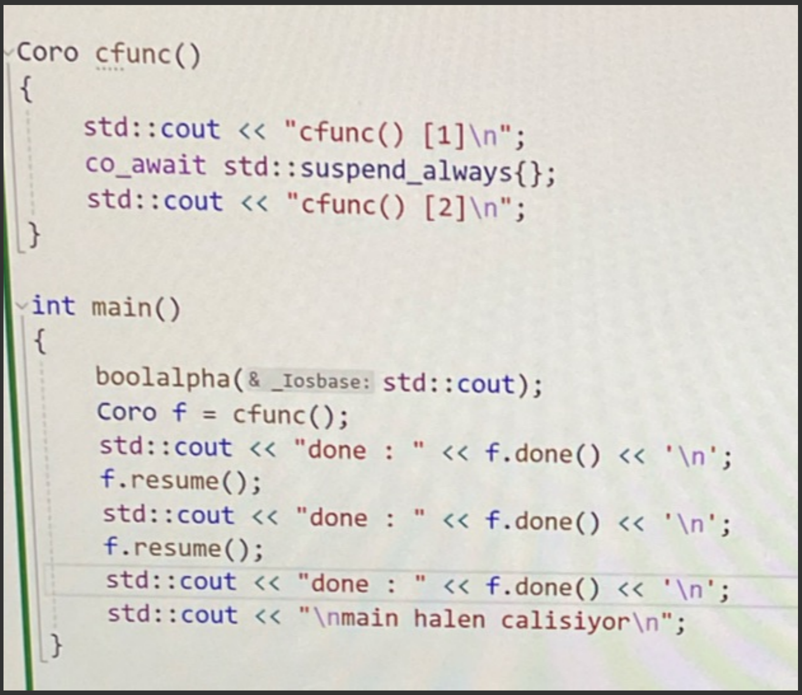
```cpp
Coro cfunc()
{
    std::cout << "cfunc() [1]\n";
    co_await std::suspend_always{};
    std::cout << "cfunc() [2]\n";
}

int main()
{
    Coro f = cfunc();
    f.resume();

    std::cout << "main halen calisiyor\n";
```



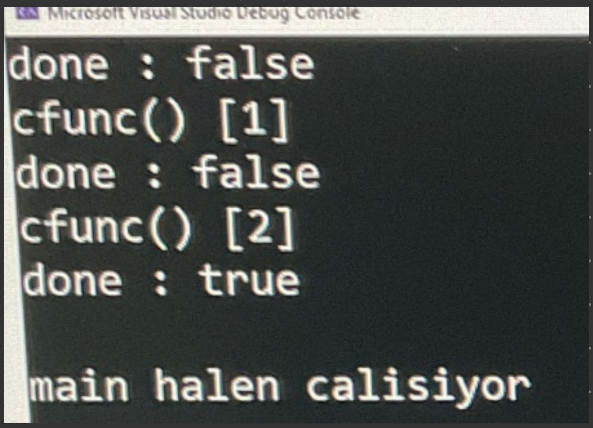
Microsoft Visual Studio Debug Console

```
func() [1]
func() [2]
ain halen calisiyor

:\Users\necat\source\repos\C
Press any key to close this w
```

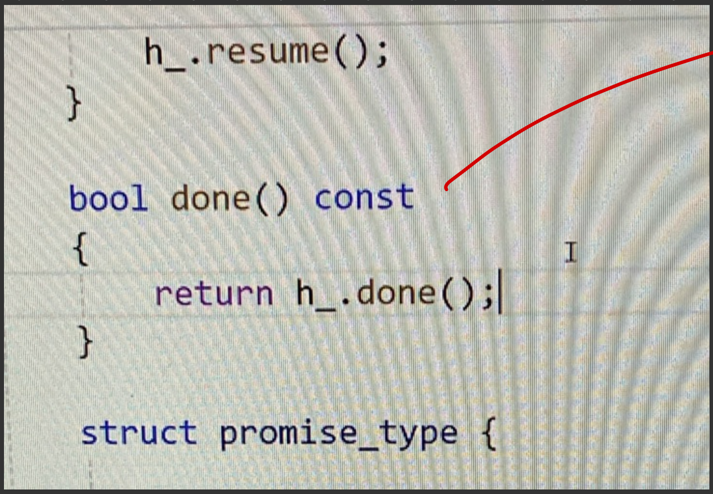
*Coroutine Handle:: Done Function:



```cpp
Coro cfunc()
{
    std::cout << "cfunc() [1]\n";
    co_await std::suspend_always{};
    std::cout << "cfunc() [2]\n";
}

int main()
{
    boolalpha(& _Iosbase: std::cout);
    Coro f = cfunc();
    std::cout << "done : " << f.done() << '\n';
    f.resume();
    std::cout << "done : " << f.done() << '\n';
    f.resume();
    std::cout << "done : " << f.done() << '\n';
    std::cout << "\nmain halen calisiyor\n";
}
```



```
done : false
cfunc() [1]
done : false
cfunc() [2]
done : true

main halen calisiyor
```



```cpp
        h_.resume();
    }

    bool done() const
    {
        return h_.done();
    }

    struct promise_type {
```

Coro interface'e done adında bir function ekledim. Coroutine handle'ın done functionı çağırıyor! → coro func bittiyse true bitmediyse false