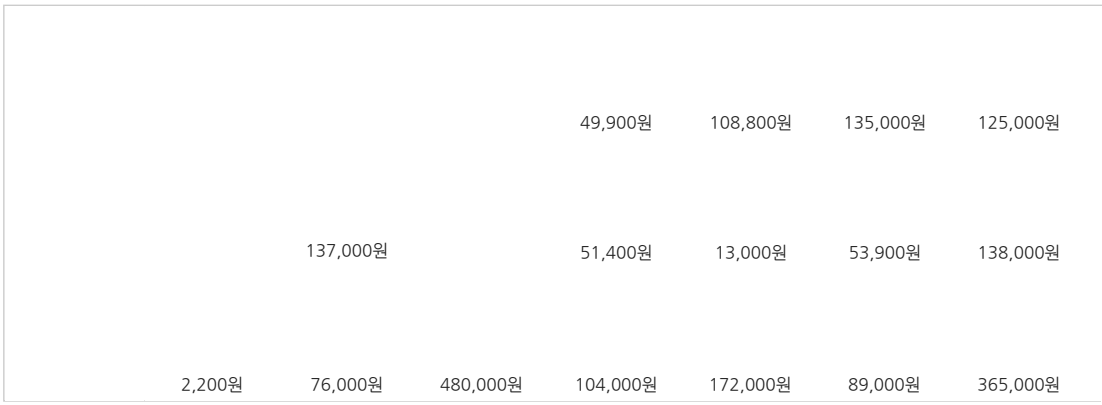
진리는어디에/C++20

# [C++20] 코루틴(Coroutine) - done()

kukuta 2021. 3. 30. 00:05



[진리는어디에] - [C++20] 코루틴(Coroutine)

[진리는어디에] - [C++20] 코루틴(Coroutine) - co_await

[진리는어디에] - [C++20] 코루틴(Coroutine) - co_yield

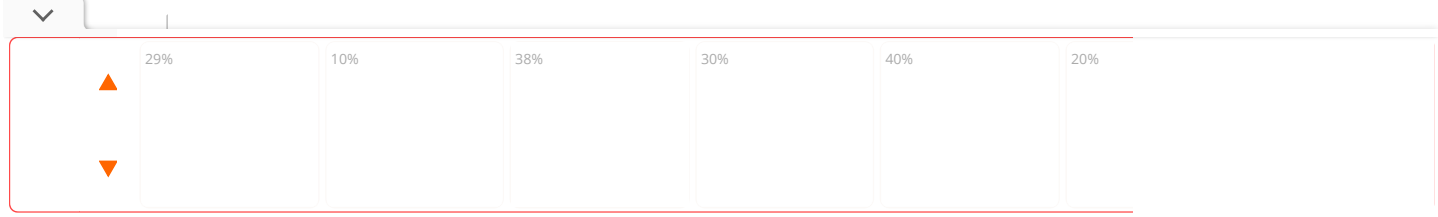
이전 강의들을 통해 C++20에서 코루틴을 생성하는 법, 코루틴을 중단하고 호출자로 돌아 오는 방법, 코루틴에서 호출자로 값을 리턴하는 방법을 알아 보았습니다.

이번 장에서는 일정한 범위의 숫자를 생성해내는 Range라는 코루틴 클래스를 만들어 보면서 코루틴의 끝을 탐지하는 방법과, Range 클래스에 iterator를 추가하여 일반 stl 컨테이너 처럼 ranged for를 이용할수 있도록 해보겠습니다

간략한 설명을 위해 예제 코드는 최대한으로 줄이고 있으나 본 포스트의 맨 아래에 전체 코드를 첨부 했으므로 이해가 부분적인 코드로 이해가 가지 않는 부분은 맨 밑의 전체 코르를 참고 부탁 드립니다.

## std::coroutine_handle<Promise>::done

coroutine_handle의 done 함수는 코루틴의 수행이 완료 되면(promise_type의 final_suspend 에서 suspend 된 상태) true를 리턴하고, 그렇지 않은 경우 false를 리턴 합니다. 우리는 resume() 이후 제어권이 호출자에게 넘어 올때 마다 done() 을 체크하여 코루틴이 완료되었는지 그렇지 않은지 알아 낼 수 있습니다.

```
    };
    // 생략..
    std::coroutine_handle coroutine;
};

Range<int> CreateRange(int start, int end)
{
    for(int i=first; i<=last; i++)
    {
        co_yield i;
    }
}

int main()
{
    Range<int> range = CreateRange(10, 20);

    while(true)
    {
        range.coroutine.resume();

        if(true == range.coroutine.done())
        {
            break;
        }
        std::cout << range.coroutine.promise().value << std::endl;
    }
}
```

자..이제 매번 루프 안에서 done()을 체크하는 것이 아닌 ranged for를 이용해 코드를 좀 더 간단히 쓸수 있도록 Range 클래스에 iterator를 추가 하겠습니다.

먼저 Range 클래스가 iteration 을 하기 위해서는 begin()과 end()가 있어야 합니다. 그리고 Range 클래스 안에는 iterator 클래스가 정의되어있어야 합니다. 그럼 대략 아래와 같은 구조가 됩니다. 반복자를 만드는 것은 어렵지 않은 작업이므로 하나하나 설명을 하지 않고 개략적인 인터페이스가 이렇다..정도만 보여 드립니다. 본문 마지막에 전체 소스 코드를 첨부 했으니 살펴 보시면 쉽게 이해가 되실겁니다.

```
template <class T>
class Range
{
public:
    struct iterator
    {
    public:
        // 생략. 자세한 코드는 전체 코드 참고
    };

    iterator begin()

    std::default_sentinel_t end()
    // 생략. 자세한 코드는 전체 코드 참고
}
;
```

위와 같이 Range 클래스 내에 iteration을 위한 반복자의 정의를 마치게 되면 이제 다음과 같이 ranged for 문법을 사용할 수 있습니다.

```
int main()
{
    Range<int> range = CreateRange(10, 20);
    for (auto value : range)
    {
        std::cout << value << std::endl;
    }
}
```

이상 C++20에서 부터 사용가능한 코루틴에 대한 전반적인 기능들을 살펴 보았습니다.

29% 10% 38% 30% 40% 20%

이만 긴글 여기까지 읽어 주셔서 감사합니다.

부록 1. 전체 코드

```cpp
#include <iostream>
#include <coroutine>

template <class T>
class Range
{
public:
    struct promise_type
    {
        T value;
        Range get_return_object()
        {
            return Range{ std::coroutine_handle<promise_type>::from_promise(*this) };
        }

        auto initial_suspend() { return std::suspend_always{}; }
        void return_void() { return ; }
        auto final_suspend() { return std::suspend_always{}; }
        void unhandled_exception() { std::exit(1); }

        std::suspend_always yield_value(const T& t)
        {
            value = t;
            return {};
        }
    };

    struct iterator
    {
    public:
        explicit iterator(std::coroutine_handle<promise_type> coroutine)
            : coroutine(coroutine)
        {
        }

        const T& operator* () const
        {
            return coroutine.promise().value;
        }

        iterator& operator++() {
            coroutine.resume();
            return *this;
        };

        bool operator == (std::default_sentinel_t) const
        {
            return !coroutine || coroutine.done();
        }
    private :
        std::coroutine_handle<promise_type> coroutine;
    };

    Range(std::coroutine_handle<promise_type> coroutine) : coroutine(coroutine)
    {
    }

    ~Range()
    {
        if (coroutine)
        {
            coroutine.destroy();
        }
    }

    iterator begin()
    {
        if (coroutine)
        {
            coroutine.resume();
```

29% 10% 38% 30% 40% 20%

```
    }

    std::coroutine_handle<promise_type> coroutine;
};

Range<int> CreateRange(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        co_yield i;
    }
}

int main()
{
    Range<int> range = CreateRange(10, 20);
    /*
    while(true)
    {
        range.coroutine.resume();

        if (true == range.coroutine.done())
        {
            break;
        }
        std::cout << range.coroutine.promise().value << std::endl;
    }
    */

    for (auto value : range)
    {
        std::cout << value << std::endl;
    }
}
```

공감 구독하기

태그 #c++20, #C/C++, #iterator

'진리는어디에/C++20' Related Articles

NO IMAGE

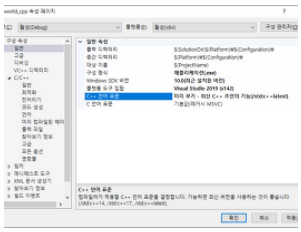
[C++20] Ranges

NO IMAGE

[C++20] 코루틴(Coroutine) - 활용

NO IMAGE

[C++20] 코루틴(Coroutine) - co_yield



[C++20] 컴파일


유익한 글이었다면 공감(♥) 버튼 꾹!! 추가 문의 사항은 댓글로!!

이름 암호 Secret

여러분의 사랑과 관심이 한 사람을 살립니다. 댓글 한번만 남겨 주세요. 관심 받고 싶습니다!! 하앍하앍!!

댓글달기

29% 10% 38% 30% 40% 20%


29%
10%
38%
30%
40%
20%