2025/11/02 19:01 1/2 Operating

## **Operating**

The newer and more robust way of processing your chains is through the process called operating.

## C++ Workflow

## **Using a Lambda**

You can easily operate on your chain via a C++ lambda and this is how you do it:

```
Chain->Operate([](const FChain::FCursor& Cursor, FMyTrait Trait)
{
    ...
});
```

The type of cursor here must match the type of the chain used. Note that you're not allowed to acquire a reference to the trait while processing a non-solid chain, only its copy. So in order to operate on a solid chain, you could do something like this:

```
SolidChain->Operate([](const FSolidChain::FCursor& Cursor, FMyTrait& Trait)
{
    ...
});
```

Now you can change the properties (fields) of the trait directly, without copying involved.

## Concurrency

Solid Chains also support a special type of operating - a multi-threaded one. The function to call is explicitly named with a Concurrently prefix and accepts two more arguments: the number of tasks to utilize and the minimum number of slots (Entity places) per each such task. For example:

```
SolidChain->OperateConcurrently([](const FSolidChain::FCursor& Cursor,
FMyTrait& Trait)
{
    ...
}, 4, 32);
```

2025/11/02 19:01 2/2 Operating

From:

http://turbanov.ru/wiki/ - Turbopedia

Permanent link:

http://turbanov.ru/wiki/en/toolworks/docs/apparatus/operating?rev=1630185133

Last update: 2021/08/28 21:12

