On naming things: The tension between naming something for what it is, what it does, or how it is used

devblogs.microsoft.com/oldnewthing/20241014-00

October 14, 2024

There is a tension in the problem of naming things: Do you name something for what it is? Do you name it for what it does? Or do you name it for how it is used?

Previously, we saw `std::type_identity`, which is named after what it is. One reaction was that the class should have been named something like `std::non_deduced`, which names it after how it is used: To prevent template type deduction.

```
template<typename...Args>
void enqueue(
    std::function<void(std::non_deduced_t<Args>...)> const& work,
    Args...args)
{
    enqueue([=] { work(args...); });
}
```

We have the opposite problem with `std::in_place`: The `in_place` types are named after how they are used, rather than what they are. They are tags that are used by some constructors of variant-like types to indicate what they should hold.

- `in_place`: Hold the primary thing (as opposed to nothing, or the alternate thing)
- `in_place_type<T>`: Hold the thing of type `T`
- `in_place_index<I>`: Hold the thing at index `I`

But when we used it as a tag type in our example, it was used not to indicate what the class itself should hold, but rather what the class should hold a reference to. Perhaps it could be named after what it is: `std::type_tag<T>` and `std::index_tag<I>`.

Though what about `std::in_place`? Maybe we leave that one alone?

And then we have `std::monostate`, which is named after what it is, rather than how it is used. In other languages, this type goes by the name `unit`, which to me feels like a name chosen with category-theory-colored glasses. (Also, the name `unit` could be misinterpreted as having to do with systems of measurement.)

Bonus chatter: Note that none of the examples are named after what they do, because none of them do anything!