33 - CONST in C++ & Mutable

segunda-feira, 24 de fevereiro de 2025 07:39

- Const
  - visibility for classes and structs to ake the code easier and cleaner
  - like a promisse that we give that something will not change
  - If we keep this promise we cn inprove erformance

```cpp
int* const_pointer_33 = new int;
// const before or after the *
const int* const_pointer_33_2 = new int; // can't change the content of the pointer but i can reassign
int* const const_pointer_33_2 = new int; // can change the content of the pointer but i cant reassign
const int* const const_pointer_33_2 = new int; // can't change the content of the pointer but i cant reassign
```

Mutable

- Something that can change
- One usage is to be able to modify members in a const method without reverting everything.
  - Mar it as mutable and you can change them
- Also aply to lambdas
  - We can deine lambdas as mutable so we can change variables passed by value in that specific scope. Otherwise it's not allowe, just passing by referece which will change the value later on
  - This way prevent us from making a copy inside the scope
  - Not very usefull

```cpp
// can pass variables by reference [&x34], by value [x34] or everything [=] or [&]
auto f34 = [=]() mutable
{
    // When passing by value, it becomes private... and w can't increment it
    // x34++;

    // We can defien the lambda as mutable, saying that the values we pass b value,
        // we can change them
    x34++;

    std::cout << "Hello " << x34 << std::endl; // 9
};

std::cout << "Hello " << x34 << std::endl; // Still 8

// call the lambda
f34();
```

```cpp
class Entity34
{
    private:
        std::string m_Name;
        mutable int m_DebugCount = 0;


    public:
        // Second const -> says taht we re not allow to modify class members
        // In some situations, we want
        const std::string& GetName() const
        {
            m_DebugCount++;
            std::cout << m_DebugCount << std:endl;
            // m_Name = "Hugo"; // Not allowed because of the second const
            return m_Name;
        }
};
```