20220423-书&GEB&C++

1.过程描述

1.1 Complications一书

1.2 GEB

Course 6:

1）takeaways

2）自己的一点想法：

1.3 C++

1）string_view

2）noexcept关键字

1.过程描述

1.1 Complications一书

> Medical care is about our life and death, and we've always needed doctors to help us understand what is happening and why, and what is possible and what is not. In the increasingly tangled web of experts and expert systems, a doctor has an even greater obligation to serve as a knowledgeable guide and confidant. Maybe machines can decide, but we still need doctors to heal

机器、算法也许可以在某些方面扮演医生的角色，但患者对于陪伴、倾诉、信任的需求却是很难从冷冰冰的机器身上得到满足的。

> Error experts, therefore, believe that it's the process, not the individuals in it, that requires closer examination and correction.

在进行错误反省时，应该更关注导致错误的过程或流程，而不是某个最终为错误负责的个人。

> The statistics may say that someday I will sever someone's main bile duct, but each time I go into a gallbladder operation I believe that with enough will and effort I can beat the odds. This isn't just professional vanity. It's a necessary part of good medicine, even in superbly "optimized" systems.Operations like that lap chole have taught me how easily error can occur, but they've also showed me something else: effort does matter; diligence and attention to the minutest details can save you. This may explain why many doctors take exception to talk of "systems

> problems," "continuous quality improvement," and "process re-engineering." It is the dry language of structures, not people.

即使统计规律表示人总会在足够多的实践中犯下或多或少的错误，但这并不妨碍医生认真对待每一次治理以尽可能“对抗”错误的发生。

1.2 GEB

Course 6:

1）takeaways

- agent-based model
- the concept of emergence
- Some deep questions are not favorable to our survival, so we may just ignore them and live.But some people do want to transcend them.

2）自己的一点想法：

- 人类作为一个系统，并不能完全了解自己的各个组件的状态，比如我们不知道自己当前的血氧浓度是多少，头痛是由于那部分导致的。是否有某种理论对这类系统进行定义？

1.3 C++

1）string_view

std::string_view比std::string的性能要高很多，因为每个std::string都独自拥有一份字符串的拷贝，而std::string_view只是记录了自己对应的字符串的指针和偏移位置。当我们在只是查看字符串的函数中可以直接使用std::string_view来代替std::string。

C++ 复制代码

```cpp
#include <iostream>
#include <chrono>
#include <string>
#include <string_view>

class Timer
{
private:
    std::string title;
    std::chrono::high_resolution_clock::time_point m_start, m_stop;
public:
    Timer(const std::string& title) :title(title)
    {
        m_start = std::chrono::high_resolution_clock::now();
    }
    ~Timer()
    {
        stop();
    }
    void stop()
    {
        m_stop = std::chrono::high_resolution_clock::now();
        std::chrono::milliseconds ms = std::chrono::duration_cast<std::chrono::milliseconds>(m_start - m_stop);
        std::cout << title << " " << (ms.count()) * 0.001 << "s\n";
    }
};

void FunctionWithString(const std::string& string)
{

}

void FunctionWithString(const std::string_view& stringView)
{

}

int main()
{
    {
        Timer timer("std::string");
        for (int i = 0; i < 1000000; i++)
        {
            std::string name = "Yang xunwu";
```

```
            std::string firstName = name.substr(0, 4);
            std::string lastName = name.substr(4, 9);
            FunctionWithString(firstName);
            FunctionWithString(lastName);
        }
    }

    {
        Timer timer("std::string_view");
        for (int i = 0; i < 1000000; i++)
        {
            const char* name = "Yang Xunwu";
            std::string_view firstName = std::string_view(name, 4);
            std::string_view lastName = std::string_view(name + 4, 9);
            FunctionWithString(firstName);
            FunctionWithString(lastName);
        }
    }
}
```

2）noexcept关键字

该关键字告诉编译器，函数中不会发生异常,这有利于编译器对程序做更多的优化。如果在运行时，noexecpt函数向外抛出了异常（如果函数内部捕捉了异常并完成处理，这种情况不算抛出异常），程序会直接终止，调用std::terminate()函数，该函数内部会调用std::abort()终止程序。

C++ 复制代码

```
无条件的noexcept
void swap(Type& x,Type& y) noexcept
{
    x.swap(y);
}

有条件的noexcept
void swap(Type& x,Type& y) noexcept(x==y)
{
    x.swap(y);
}

```