20220413-C++

1.过程描述

Timer使用示例 C++ 复制代码

```cpp
class Timer
{
public:
    Timer()
    {
        std::chrono::high_resolution_clock::now();
    }
    ~Timer()
    {
        Stop();
    }
    void Stop()
    {
        auto endTimepoint = std::chrono::high_resolution_clock::now();
        auto start = std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
        auto end= std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();

        auto duration = end - start;
        double ms = duration * 0.001;

        cout << duration << "us ("<<ms<<"ms)\n";

    }
private:
    std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTimepoint;
};

int main()
{
    int value = 0;
    {
        Timer timer;
        for (int i = 0; i < 1000000; i++)
        {
            value += 2;
        }
    }
    cout << value << endl;
    __debugbreak();

}
```

unique_ptr和shared_ptr的Timer对比 C++ 复制代码

```cpp
class Timer
{
public:
    Timer()
    {
        m_StartTimepoint=std::chrono::high_resolution_clock::now();
    }
    ~Timer()
    {
        Stop();
    }
    void Stop()
    {
        auto endTimepoint = std::chrono::high_resolution_clock::now();
        auto start = std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
        auto end= std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();

        auto duration = end - start;
        double ms = duration * 0.001;

        cout << duration << "us ("<<ms<<"ms)\n";

    }
private:
    std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTimepoint;
};

int main()
{
    struct Vector2
    {
        float x, y;
    };

    cout << "Make shared\n";
    {
        array<shared_ptr<Vector2>, 1000> sharedPtrs;
        Timer timer;
        for (int i = 0; i < sharedPtrs.size(); i++)
        {
            sharedPtrs[i] = make_shared<Vector2>();
```

```
42          }
43      }
44
45      cout << "New shared\n";
46      {
47          array<shared_ptr<Vector2>, 1000> sharedPtrs;
48          Timer timer;
49          for (int i = 0; i < sharedPtrs.size(); i++)
50          {
51              sharedPtrs[i] = shared_ptr<Vector2>(new Vector2());
52          }
53      }
54
55      cout << "Make unique\n";
56      {
57          array<unique_ptr<Vector2>, 1000> uniquePtrs;
58          Timer timer;
59          for (int i = 0; i < uniquePtrs.size(); i++)
60          {
61              uniquePtrs[i] =make_unique<Vector2>();
62          }
63      }
64
65      __debugbreak();
66
67  }
```

structured bind&multiple return C++ 复制代码

```cpp
tuple<string,int,int> CreatePerson()
{
    return { "Cherno",24,56 };
}//事实上用struct更好

pair<string, int> CreatePerson2()
{
    return { "Cherno",24 };
}

int main()
{
    /*
    auto person = CreatePerson();
    string& name=get<0>(person);
    int age = get<1>(person);
    int weight = get<2>(person);

    string name;
    int age;
    int weight;
    tie(name, age, weight) = CreatePerson();
    */

    auto[name, age, weight] = CreatePerson();//只有C++17才能compile
}
```

optional C++ 复制代码

```cpp
#include <iostream>
#include <fstream>
#include <optional>
#include <string>
std::optional<std::string> ReadFileAsString(const std::string& filepath)
{
    std::ifstream stream(filepath);
    if (stream)
    {
        std::string result;
        //read file
        stream.close();
        return result;
    }
    return {};
}

int main()
{
    std::optional<std::string> data = ReadFileAsString("data.txt");
    std::string value = data.value_or("Not present");
    std::cout << value << std::endl;
    if (data.has_value())
    {
        std::cout << "File read successfully\n";
    }
    else
    {
        std::cout << "File could not be opened\n";
    }
    std::cin.get();
}
//无法正确读取文件，但不知道bug在哪
```

2.结果输出

今天主要看了Cherno的几个C++视频，主要讲的都是一些比较新的C++特性，没实际使用的话确实学起来没啥感觉。这部分视频估计就看到这里，后面剩余的十几二十个视频可能等实际编程遇到具体问题时再看了。C++的语言学习暂时告一段落，之后打算先用4天时间完成TCP/IP编程，然后继续算法和数据结构的学习，这部分目前还没什么较好的学习思路，感觉看书有点云里雾里（主要也是没啥耐心），可能主要找一门网课跟着学一学。在这之后用一周时间做一个C++的项目，尽可能温习一下学习过的相关

特性。The Cherno的游戏引擎教程挺好，但是感觉体量过于大了，要真正搞起来估计没一两个月完成不了。在完成项目之后，要么开启java的学习，要么继续学习像计算机系统、体系机构之类的基础大件，或者统计、机器学习、离散数学、组合数学之类的知识。Anyway，四月份的计划大致如下（之前的计划看来已经不可持续，过高估计了自己的学习能力跟激情，也因为疫情导致困在宿舍学习效率不高，说来惭愧）。

原先的计划：

任务项	时间	预期	碎片时间关注
C++ prime	3月24–26	掌握C++基本语法知识	• The Cherno c++视频 • 计算机图形学 • 游戏引擎架构 • 即兴演讲+结构化写作
计算机网络	3月27–28	看完全书	
计算机网络习题	3月29	利用习题检验并回顾	
数据结构c语言版	3月30–4月3	看完全书	
数据结构与算法分析	4月4–4月10	看完全书	
TCP/IP网络编程	4月11–12	看完全书	• 数学之美 • 大图景 • 复杂性思维
C++深度学习框架	4月13–15	看完全书	
C和指针	4月16–18	看完全书	
计算机网络自顶向下	4月19–20	看完全书	

新计划：

任务项	时间	预期	碎片时间关注
TCP/IP网络编程	4月14–17	阅读全书并完成全书代码的抄写	算法导论阅读 通信原理 计算机网络
C++项目	4月18–24	完成一到两个C++项目实践	
数据结构及算法网课	4月25–30	完成一门数据结构及算法课程的学习	

目前具体做什么C++项目还没确定，当然难度不可能太大，考虑到目前也只完成基础知识的学习，太难的话容易劝退。后续几天重点关注一下有哪些值得上手的开源项目。数据结构及算法在学习完之后可能还是得在Leetcode上多刷题才能检验学习成果。目前感觉下来java跟C#可能会是下个月的学习重点，因为这两门语言也比较重要。有了C++的基础应该会比较好上手一点；此外，由于之后工作主要跟通信相关，所以这块也得提前打好一些基础，免得到时候手忙脚乱的。

目前上海的疫情还没稳定下来，学校里还时不时冒出几个阳性，距离解封遥遥无期。因为个人还是比较喜欢安静的学习环境，很容易受到外界干扰，所以在寝室学习效率确实不高，这点要努力克服。理想的状况是五月份能解封回家，回家之后我也不想只局限于干巴巴的学习之上，还是希望能在正式入职之前做些比较有意义、比较cool的事情，比如做做视频、学学吉他或者做些公益活动，以后估计很难有时间跟精力去从事一些工作以外的事情了。所以，珍惜时间，提升自我，确保每一天都有所进步吧。