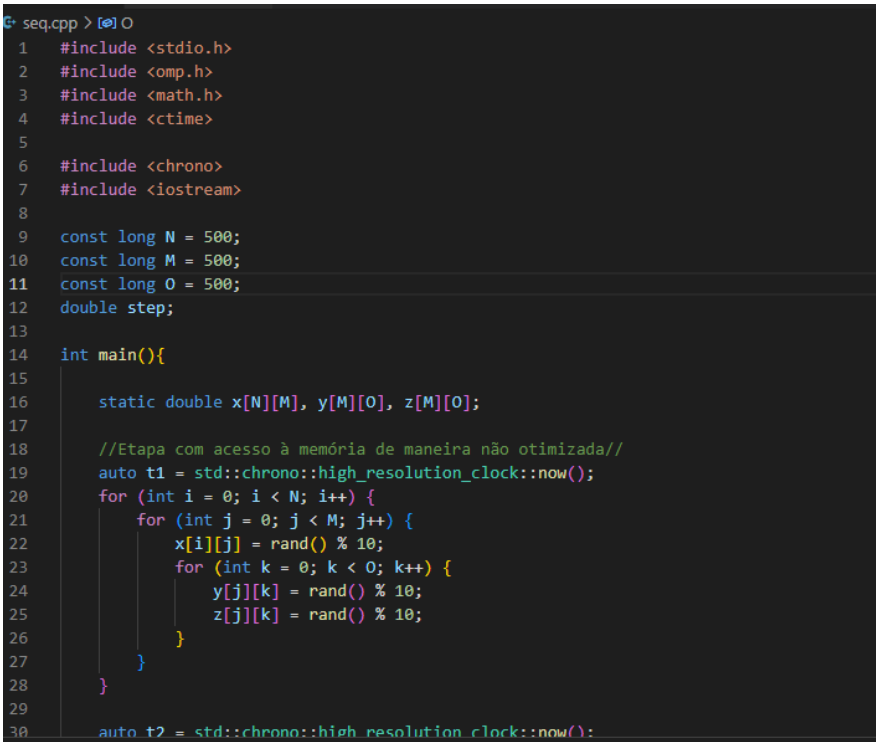
# Exercício 8 - Vidal Elias de Carvalho Neto

Utilizando como base o código do projeto final, realizei a otimização da compilação para diminuir o tempo em alguns dos casos, segue o código com a otimização de blocagem;



```
seq.cpp > O
#include <stdio.h>
#include <omp.h>
#include <math.h>
#include <ctime>

#include <chrono>
#include <iostream>

const long N = 500;
const long M = 500;
const long O = 500;
double step;

int main(){

    static double x[N][M], y[M][O], z[M][O];

    //Etapa com acesso à memória de maneira não otimizada//
    auto t1 = std::chrono::high_resolution_clock::now();
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            x[i][j] = rand() % 10;
            for (int k = 0; k < O; k++) {
                y[j][k] = rand() % 10;
                z[j][k] = rand() % 10;
            }
        }
    }

    auto t2 = std::chrono::high_resolution_clock::now();
```

```
auto t2 = std::chrono::high_resolution_clock::now();

//Otimização da memória utilizando blocagem//
auto t3 = std::chrono::high_resolution_clock::now();

for (int jj = 0; jj < M; jj = jj + 2) {
    for (int kk = 0; kk < O; kk = kk + 2) {
        for (int i = 0; i < N; i = i + 1) {
            for (int j = jj; j < jj + 2 - 1; j = j + 1) {
                x[i][j] = rand() % 10;
                for (int k = kk; k < kk + 2 - 1; k = k + 1) {
                    y[j][k] = rand() % 10;
                    z[j][k] = rand() % 10;
                }
            }
        }
    }
}

auto t4 = std::chrono::high_resolution_clock::now();

auto duration = (std::chrono::duration_cast<std::chrono::microseconds>(t2-t1).count());
auto duration2 = (std::chrono::duration_cast<std::chrono::microseconds>(t4-t3).count());


float time = (float)duration/1000000;
float time2 = (float)duration2/1000000;
```

Resultados:

```
C:\Users\ACER\Desktop\Arq2>g++ seq.cpp -o seq.exe

C:\Users\ACER\Desktop\Arq2>seq.exe
Tempo de processamento =5.53659segundos.
Tempo de processamento com otimizacao da memoria =2.01126segundos.

C:\Users\ACER\Desktop\Arq2>g++ seq.cpp -o1 -o seq.exe

C:\Users\ACER\Desktop\Arq2>seq.exe
Tempo de processamento =5.14555segundos.
Tempo de processamento com otimizacao da memoria =1.97862segundos.

C:\Users\ACER\Desktop\Arq2>g++ seq.cpp -o2 -o seq.exe

C:\Users\ACER\Desktop\Arq2>seq.exe
Tempo de processamento =6.09135segundos.
Tempo de processamento com otimizacao da memoria =2.00101segundos.

C:\Users\ACER\Desktop\Arq2>g++ seq.cpp -o3 -o seq.exe

C:\Users\ACER\Desktop\Arq2>seq.exe
Tempo de processamento =5.17436segundos.
Tempo de processamento com otimizacao da memoria =1.99378segundos.
```

| | -O | -O1 | -O2 | -O3 |
|---|---|---|---|---|
| Sem blocagem | 5,536s | 5,145s | 6,091s | 5,174s |
| Com blocagem | 2,011s | 1,978s | 2,001s | 1,993s |

Podemos observar que com exceção da otimização o2, todos eles apresentaram um tempo menor em todos os casos, porém não com uma grande relevância, talvez devido a grandeza dos dados utilizada.