**OpenMP 编程**

姓名：郁万祥 学号：2013852

问题一、

分别实现课件中的梯形积分法的Pthread、OpenMP版本，熟悉并掌握

OpenMP编程方法，探讨两种编程方式的异同。

具体实现见源码文件：1\_pthread、1\_omp

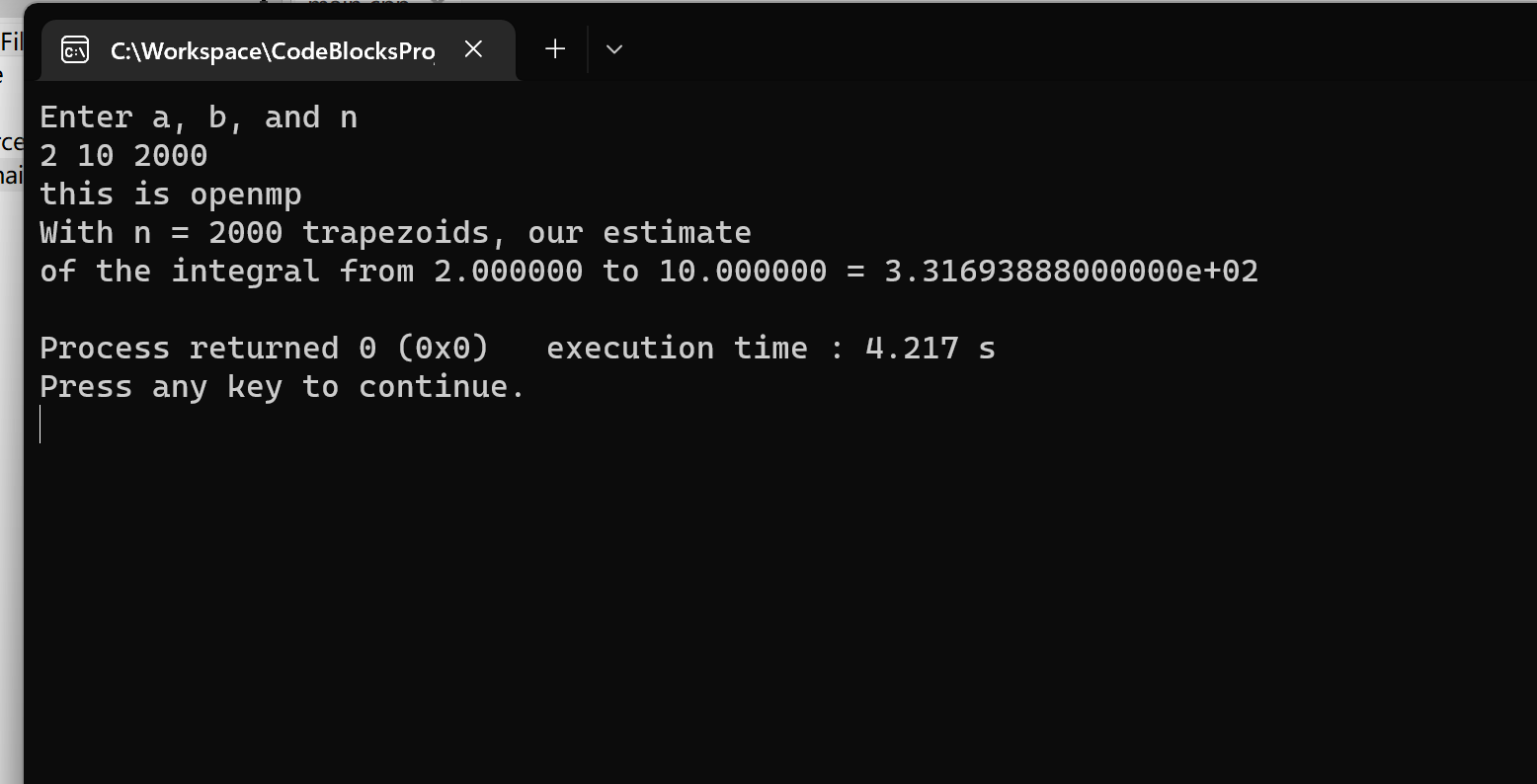
注意，对于使用omp编程的程序，需要编译并链接到-fopenmp以启用OpenMP。具体实现过程：

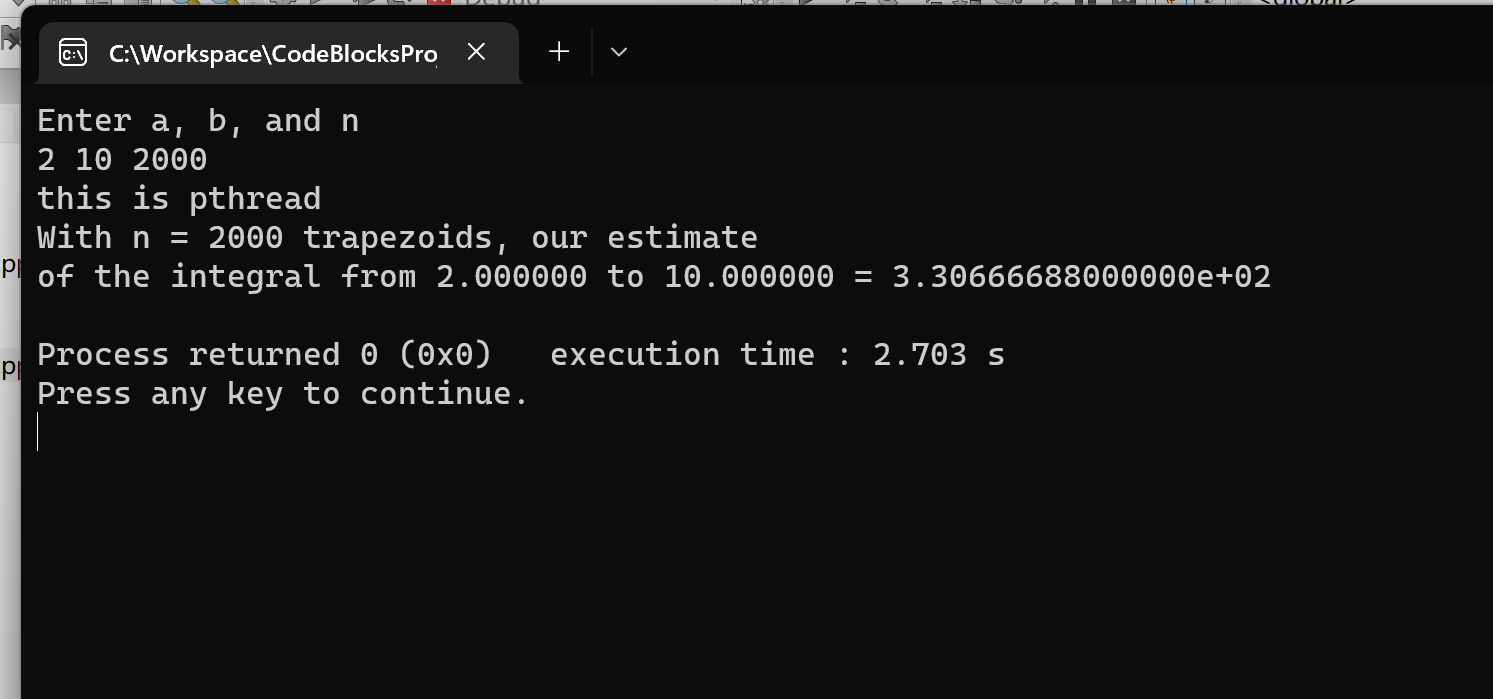
Setting->compiler->Compiler settings->other options里输入-fopenmp；

Setting->compiler->linker settings->other linker options里输入-lgomp -lpthread；

Setting->compiler->linker settings->Linker library里添加libgomp.dll.a文件：如D:\TDM-GCC\lib\gcc\mingw32\5.1.0\libgomp.dll.a。

实现结果：





比较异同：

OpenMP：

OpenMP是根植于编译器的，更偏向于将原来串行化的程序，通过加入一些适当的编译器指令(compiler directive)变成并行执行，从而提高代码运行的速率。这样做是在没有OpenMP支持编译时，代码仍然可以编译。因此，OpenMP的代码更易于扩展。创建线程等后续工作需要编译器来完成。OpenMP,不需要指定数量，在有循环的地方加上代码，修改设置文件极客。OpenMP非常方便，因为它不会将软件锁定在事先设定的线程数量中，但是相对的查错更难也更麻烦。

Pthread：

而Pthread是一个库，所有的并行线程创建都需要我们自己完成。Pthread仅在有多个处理器可用时才对并行化有效，并且仅在代码针对可用处理器数进行了优化时才有效。Pthread在程序启动时创建一束线程，将工作分配到线程上。然而，这种方法需要相当多的线程指定代码，而且不能保证能够随着可用处理器的数量而合理地进行扩充。

问题二、

对于课件中“多个数组排序”的任务不均衡案例进行OpenMP编程实现（规模

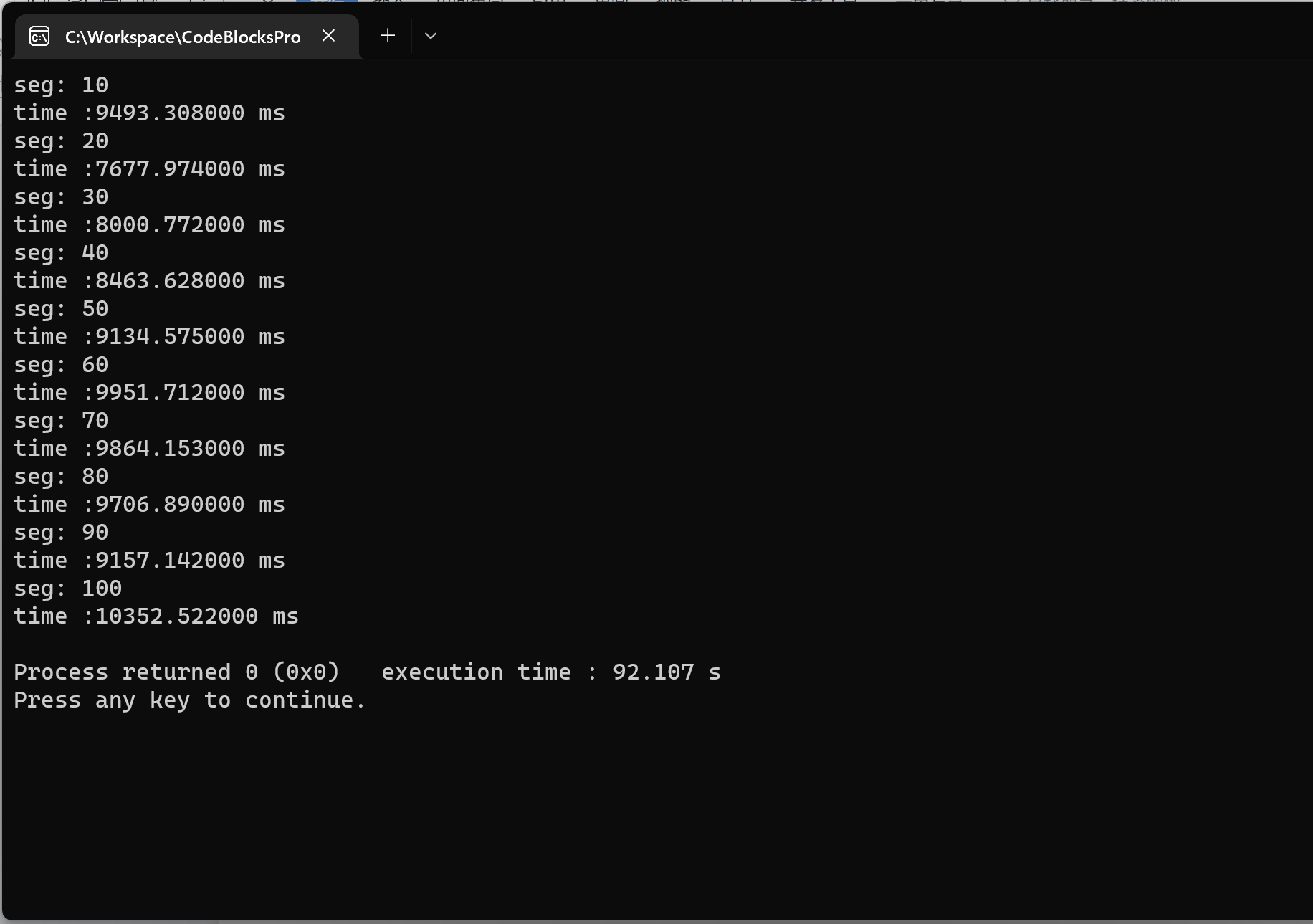
可自己调整)，并探索不同循环调度方案的优劣。提示：可从任务分块的大小、

线程数的多少、静态动态多线程结合等方面进行尝试，探索规律。

具体实现见源码文件：2\_omp1、2\_opm2

1. 通过任务分块的大小入手：

指定线程数目为4，通过改变粗颗粒的分配大小seg从10到100，没次增长10，进行更改任务块的大小，得出程序运行结果：



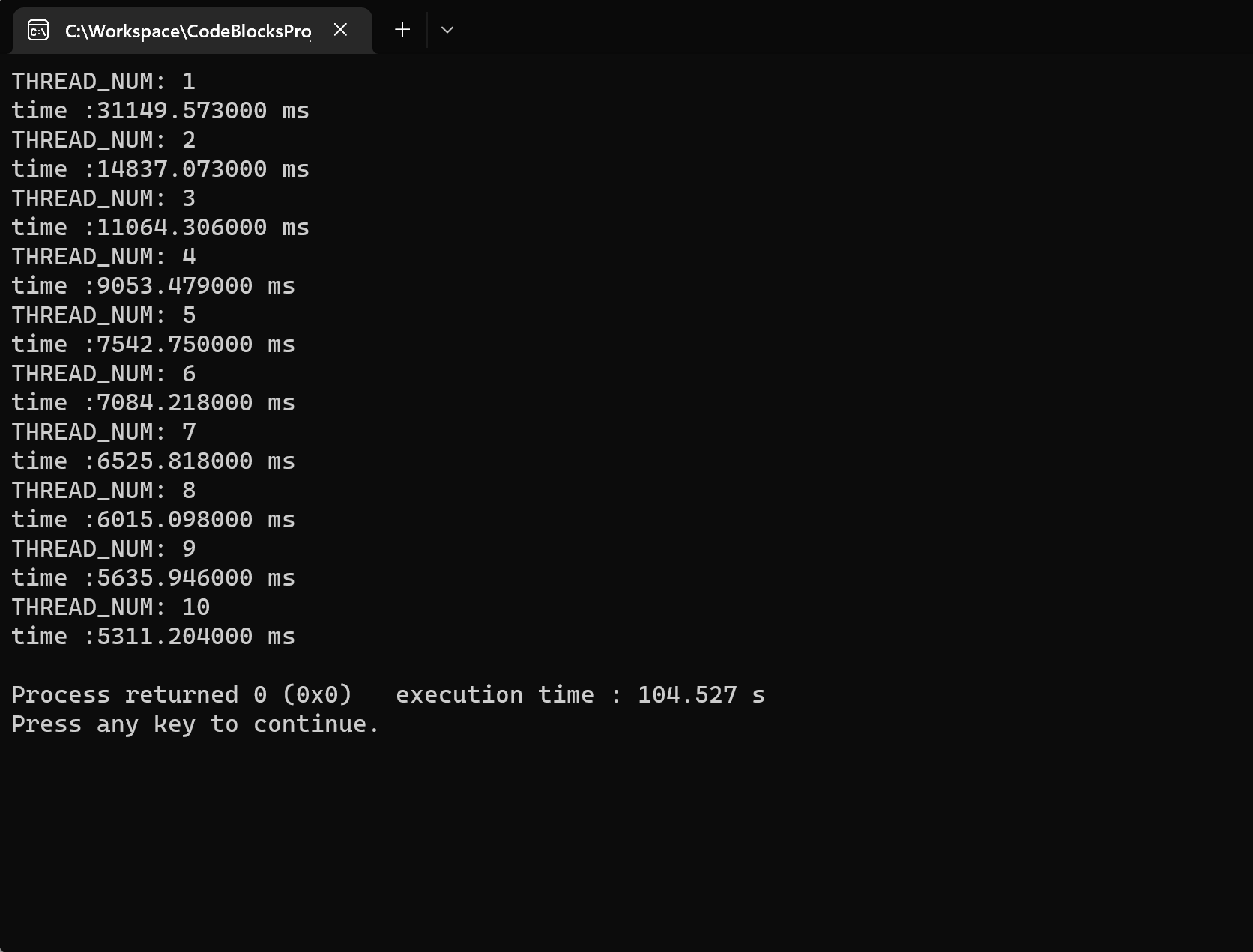
整理数据：

我们发现，通过改变任务块的大小，对于运行时间虽然有影响，但是影响不大，并且没有规律的影响，只不过，在此次实验结果中，当每次的分配数量为20时，程序运行时间最少，效率最高。

1. 通过线程数的多少入手：

基于上面的实验，我们设定每次分配数量恒为20，通过改变线程数目进行实验，线程数目我们从1到10，每次增加1。

实验结果：



整理数据

我们发现，在设定的线程数目的范围区间内，

1. 线程数目越多，程序运行时间越小；
2. 但是，值得说明的是，线程数目对程序运行效率的影响程度并不是不变的；
3. 在线程数目比较少的时候，线程数目对于程序运行效率的改善还比较客观；
4. 但是，当线程数目超过4的时候，影响就不是很大了；
5. 因此考虑到实际情况，对于不同规模的问题，应该选择合适的线程数目。

附加题：

实现高斯消去法解线性方程组的OpenMP编程，与SSE/AVX编程结

合，并探索优化任务分配方法。

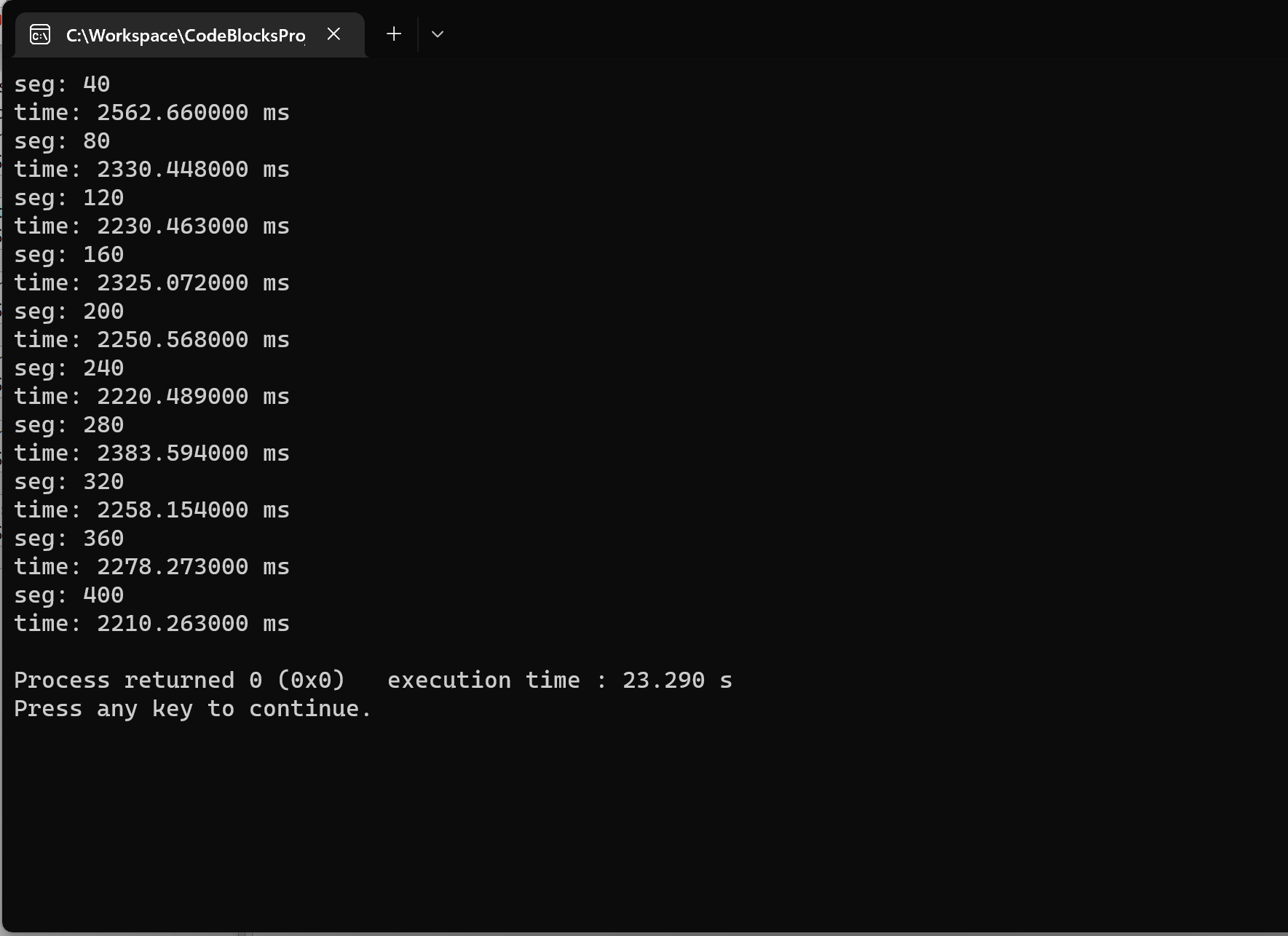
具体实现见源码文件：3\_omp

实验过程：

首先，我们通过第上次实验发现，其实对于较小的矩阵规模，多线程的效果并不明显，会导致，无论任务划分如何，效率都还不错，因此对于任务划分的探索也比较局限，所以在进行高斯并行化实验的过程中，增加矩阵整体的大小，同时相应的，对于任务的分配也应该有所增加，因此实验数据如下：

矩阵规模固定为2048\*2048，线程数固定为4，对于任务的划分，我们设定seg从40到400，每次增加40，来实现任务划分从每次40到每次400的目的，探索高斯消去中任务划分的优化。

实验结果：



结果分析：

当前情况下，我们发现，当任务划分比较大的时候，运行的效率比较高，其中当任务划分为240时，本次实验的效率最高，所以，其实任务的合适划分需要根据问题的规模进行敲定，问题规模越大，任务划分的优化效果更加明显，达到效率最高所需要的任务划分一般也尽量比较大。