電子三丙 107360734 *Name:* 游鎮遠

1. Coding *independent threads* can speed up the running of programs. Let us conduct an experiment to experience the features and skill of *multithreading*. You need to *code two different styles of programs* to perform a "matrix multiplication" in the same programming language (e.g. C language), as follows.

$$C_{35\times35} = A_{35\times60} \times B_{60\times35}$$
$$\left[a_{ij}\right] = 3.5i - 6.6j, \quad \left[b_{ij}\right] = 6.6 + 8.8i - 3.5j$$

- In the ***first*** program, you just code it following the *traditional* ***for-looping*** skill.
- In the ***second*** program, you need trying to code it by using the new ***multithreading*** skill.

Q1: Point out the *major parts* in the threaded program to highlight its differences with *for-loops*.

for迴圈的運行方式是循環執行，執行緒是同時動作，像是Rows的執行緒就是每一列的計算同時動作；Cells則是每一個元素的計算同時動作。

在程式中的顯著差異就是在執行緒的寫法中，要創建執行緒，pthread_t threads[ ]和pthread_create函式可以用來建立新的執行緒，執行緒在建立之後，就會以同時動作的方式執行。再以pthread_join等待所有執行緒的結束，結束後將所有同時計算出的結果組合在一起輸出。

Q2: Record your *experimental results* at least *3 rounds* execution in the below table, and state how you can count the running time of programs in ms.

Coding Skill	Lines of Code	Execution Time (ms)			Average Execution Time
		1-round	2-round	3-round	
【A】 ***For-loops***	43	8302	8173	8207	8227.3
【B1】 *by cells* ***Multithread***	77	6761	6687	6717	6721.6
Differences 【A - B1】	-34	1585	1486	1490	1520.3
【B2】 *by rows* ***Multithread***	57	7570	7717	7397	7561.3
Differences 【A – B2】	-14	732	456	810	666

電子三丙 107360734 *Name*: 游鎮遠

Q3: State your discovering and comments on this exercise of coding threaded programs.

完成這次的作業之後，我學到了如何寫執行緒的程式，也驚覺到，原來寫一個需要重複動作的程式，除了以前只會的for迴圈之外，還能有更好的做法，就是讓它同時執行。但畢竟這是我第一次寫執行緒的程式，在實作的過程中也是跌跌撞撞。

這次的作業需要寫三個程式，一開始，我是從for迴圈的程式下手，畢竟是相對簡單的一個部分，所以很快做完了。再來，我選擇做Rows的執行緒程式，也就是同時執行每一列的計算，我發現且參考了網路上的教學，使用pthread的函式庫，可以輕鬆地建立執行緒來做計算，再將A和B矩陣改上作業所要求的aij與bij算法就行，與for迴圈比較呢，是確實比較快，但我是測試到使用200*200的矩陣之後才比較看得出速度的差異，可以差到快一秒（我使用time.h寫入程式直接計算main()的執行時間）。

接著，來到了Cells的部分，每個執行緒計算一個元素，以200*200的矩陣來看，就是同時進行200*200條執行緒（雖然執行緒不會同時建構，但還是這樣講比較好懂）！除了看起來就比Rows難之外，其程式的寫法和Rows比較，確實複雜許多，原本在Rows的寫法，我只需考慮A矩陣的列向量，再乘上B矩陣的每一行就好，因為是每條執行緒算一個列，但到了Cells變成A矩陣列和行都要考慮，再去乘上對應的B矩陣元素，光是這點，在相乘時就複雜很多了，甚至還要將值先丟進暫存的data裡，所以Cells的部分我與同學討論了很久。

而在我兢兢業業地以為我終於做完的時候，我按下了執行，結果執行時間需要快17秒…，原本以為是我用200*200的矩陣相乘會產生太多的執行緒導致overhead過高，但我改成10*10測試，一樣跑得比for迴圈和Rows還慢，因為在執行初期會卡頓很久才顯示出答案，所以我又反覆修改了很久，但還是找不到原因。

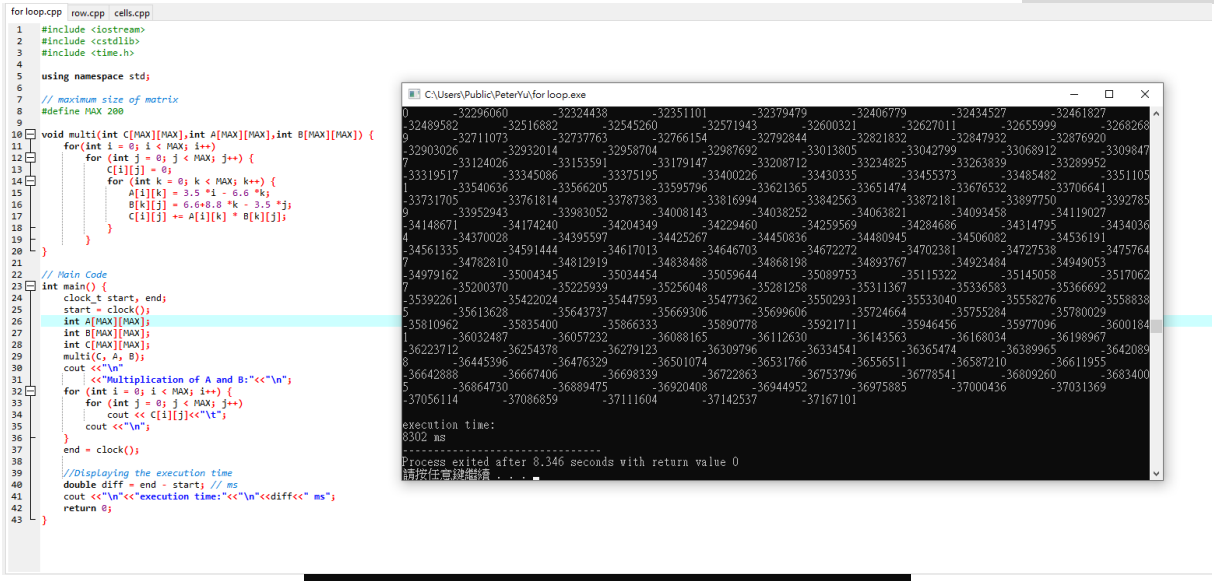
結果，就在我一籌莫展之際，想說放在別的編譯與執行環境跑看看好了，就把我的程式傳給同學，讓他在他的電腦幫我試看看（我們都是用Dev C++），竟然就奇蹟般地能用！完全不會卡頓，執行時間只要6秒左右，比for迴圈快了一秒多，也比Rows快了快一秒，於是我才確定原來是環境問題，而這問題也在我重灌Dev C++之後消失了…，可能是我以前有改到什麼環境參數吧。

總之，我其實滿感謝有這次機會，讓我學習到了執行緒程式的寫法，相較於一直循環的for迴圈動作方式，執行緒能同時處理多個計算，讓程式執行的效率變好，將CPU的計算能力最大化，這與上課聽老師講理論完全是不同的體驗，果然凡事都是要親自做過才會懂呢，而要不是這次要交這份作業，我可能一輩子也不會碰到這塊領域，所以我很珍惜這次機會。

* Note that you need to take some pictures of the running screen on LCD, and appendix your two *source codes* in your report.

For-loops :

```
#include <iostream>
#include <cstdlib>
#include <time.h>

using namespace std;

// maximum size of matrix
#define MAX 200

void multi(int C[MAX][MAX],int A[MAX][MAX],int B[MAX][MAX]) {
      for(int i = 0; i < MAX; i++)
            for (int j = 0; j < MAX; j++) {
                  C[i][j] = 0;
                  for (int k = 0; k < MAX; k++) {
                        A[i][k] = 3.5 *i - 6.6 *k;
                        B[k][j] = 6.6+8.8 *k - 3.5 *j;
                        C[i][j] += A[i][k] * B[k][j];
            }
            }
}

// Main Code
int main() {
   clock_t start, end;
   start = clock();
      int A[MAX][MAX];
      int B[MAX][MAX];
      int C[MAX][MAX];
      multi(C, A, B);
      cout <<"\n"
      <<"Multiplication of A and B:"<<"\n";
      for (int i = 0; i < MAX; i++) {
            for (int j = 0; j < MAX; j++)
                  cout << C[i][j]<<"\t";
            cout <<"\n";
      }
      end = clock();

   //Displaying the execution time
      double diff = end - start; // ms
      cout <<"\n"<<"execution time:"<<"\n"<<diff<<" ms";
      return 0;
}
```

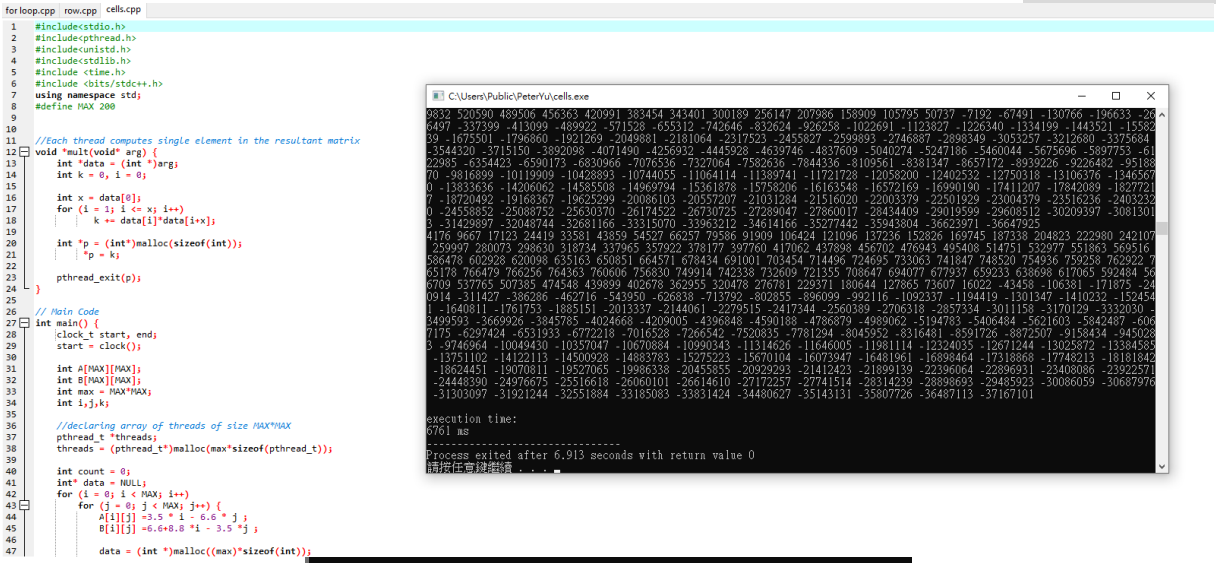


Cells Multithread :

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
#include<stdlib.h>
#include <time.h>
#include <bits/stdc++.h>
using namespace std;
#define MAX 200


//Each thread computes single element in the resultant matrix
void *mult(void* arg) {
	int *data = (int *)arg;
	int k = 0, i = 0;

	int x = data[0];
	for (i = 1; i <= x; i++)
		k += data[i]*data[i+x];

	int *p = (int*)malloc(sizeof(int));
		*p = k;

	pthread_exit(p);
}

// Main Code
```

```
int main() {
    clock_t start, end;
   start = clock();

      int A[MAX][MAX];
      int B[MAX][MAX];
      int max = MAX*MAX;
   int i,j,k;

      //declaring array of threads of size MAX*MAX
      pthread_t *threads;
      threads = (pthread_t*)malloc(max*sizeof(pthread_t));

      int count = 0;
      int* data = NULL;
      for (i = 0; i < MAX; i++)
            for (j = 0; j < MAX; j++) {
                  A[i][j] =3.5 * i - 6.6 * j ;
                  B[i][j] =6.6+8.8 *i - 3.5 *j ;

                  data = (int *)malloc((max)*sizeof(int));
                  data[0] = MAX;

                  for (k = 0; k < MAX; k++) {
                        data[k+1] = A[i][k];
                        data[k+MAX+1] = B[k][j];
            }

                        //creating threads
                        pthread_create(&threads[count++], NULL, mult, (void*)(data));
            }

   cout << "\n"
     << "Multiplication of A and B:" << "\n";

      for (i = 0; i < max; i++) {
         void *k;
      //join threads
         pthread_join(threads[i], &k);
         int *p = (int *)k;
         printf("%d ",*p);
         if ((i + 1) % MAX == 0)
               printf("\n");
      }
      end = clock();

   //Displaying the execution time
   double diff = end - start; // ms
      cout <<"\n"<<"execution time:"<<"\n"<<diff<<" ms";
   return 0;
}
```

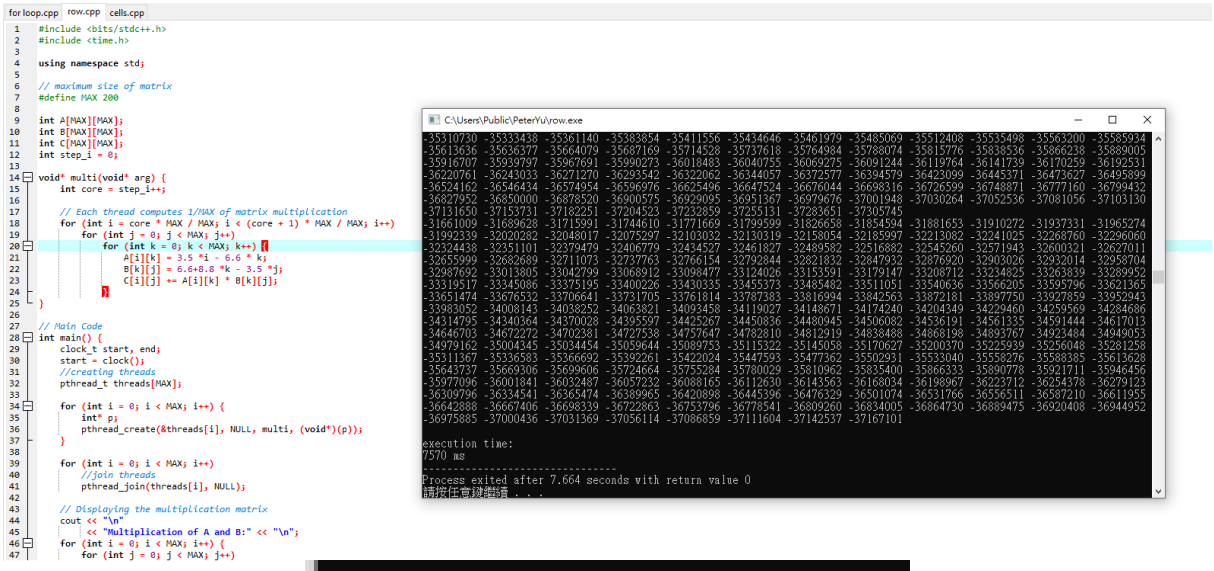
電子三丙 107360734 *Name*: 游鎮遠

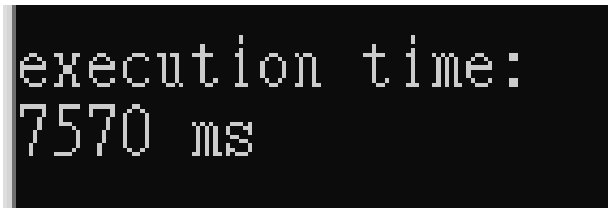


Rows Multithread :

```
#include <bits/stdc++.h>
#include <time.h>

using namespace std;

// maximum size of matrix
#define MAX 200

int A[MAX][MAX];
int B[MAX][MAX];
int C[MAX][MAX];
int step_i = 0;

void* multi(void* arg) {
   int core = step_i++;

   // Each thread computes 1/MAX of matrix multiplication
   for (int i = core * MAX / MAX; i < (core + 1) * MAX / MAX; i++)
    for (int j = 0; j < MAX; j++)
          for (int k = 0; k < MAX; k++) {
                A[i][k] = 3.5 *i - 6.6 * k;
                B[k][j] = 6.6+8.8 *k - 3.5 *j;
                C[i][j] += A[i][k] * B[k][j];
          }
}

// Main Code
```

```
int main() {
   clock_t start, end;
   start = clock();
   //creating threads
   pthread_t threads[MAX];

   for (int i = 0; i < MAX; i++) {
    int* p;
    pthread_create(&threads[i], NULL, multi, (void*)(p));
   }

   for (int i = 0; i < MAX; i++)
    //join threads
    pthread_join(threads[i], NULL);

   // Displaying the multiplication matrix
   cout << "\n"
      << "Multiplication of A and B:" << "\n";
   for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
           cout << C[i][j] << " ";
    cout << "\n";
   }
   end = clock();

   //Displaying the execution time
      double diff = end - start; // ms
      cout <<"\n"<<"execution time:"<<"\n"<<diff<<" ms";
   return 0;
}
```





Due date: **June 17, 2021.**