



**MATERIA: COMPUTO EN LA NUBE**

**ALUMNO: FRANCISCO MEDELLIN ZERTUCHE**

**MATRICULA: A01794044**

**PROFESOR TITULAR: EDUARDO ANTONIO CENDEJAS  
CASTRO**

**TAREA: 1. Programación de una solución paralela**

## 1. Introducción:

En las ciencias computacionales el concepto de paralelismo hace referencia a técnicas computacionales basadas en un principio "dividir un gran problema en varios y resolverlos al mismo tiempo". En otras palabras, es el uso simultaneo de múltiples recursos computacionales para resolver un problema.

En este ejercicio se llevó acabo el diseño e implementación de un algoritmo paralelo mediante el uso del lenguaje de programación C++ y la librería OpenMP. El objetivo principal del algoritmo es la suma paralela entre arreglos.

## 2. Liga de Repositorio:

Dentro del repositorio para la materia cloud computing, se encuentra la carpeta "M2 Tarea1 ProgramacionParalela" donde se encuentran los archivos de esta actividad.

[Link de la carpeta de tareas en el repositorio de clase.](#)

## 3. Ejecuciones del proyecto y explicación del código y resultados:

### Explicación del código:

- Primero declaramos las librerías necesarias.
- Definimos las constantes **N** (cantidad de elementos a manejar en los arreglos), **chunk** (tamaño que tendrán los grupos de los arreglos para que cada hilo los opere), **mostrar** (Cantidad de datos a imprimir).

```
3
4  #include "pch.h"
5  #include <iostream>
6  #include <omp.h>
7
8  #define N 1000
9  #define chunk 100
10 #define mostrar 10
11
12 void imprimeArreglo(float *d);
13
14 int main()
15 {
```

- Declaración de 3 arreglos (a, b, c). Los dos primeros usados para asignar valores aleatorios dentro de un ciclo for. El arreglo "c" almacena el resultado de la suma de los dos primeros.

```

16      std::cout << "Sumando Arreglos en Paralelo!\n";
17      float a[N], b[N], c[N];
18      int i;
19
20      for (i = 0; i < N; i++)
21      {
22          a[i] = i * 10;
23          b[i] = (i + 3) * 3.7;
24      }
25      int pedazos = chunk;

```

- Después definimos la instrucción for que se realizara en paralelo usando la librería OpenMP. Este ciclo es donde se realiza la sumatoria en paralelo.

```

27      #pragma omp parallel for \
28      shared(a, b, c, pedazos) private(i) \
29      schedule(static, pedazos)
30
31      for (i = 0; i < N; i++)
32          c[i] = a[i] + b[i];
33

```

- Luego se imprime la suma en paralelo para comprobar los resultados, imprimiendo tanto los valores de a y b y sus sumas en "c". Se invoca a la función ImprimeArreglo().

```

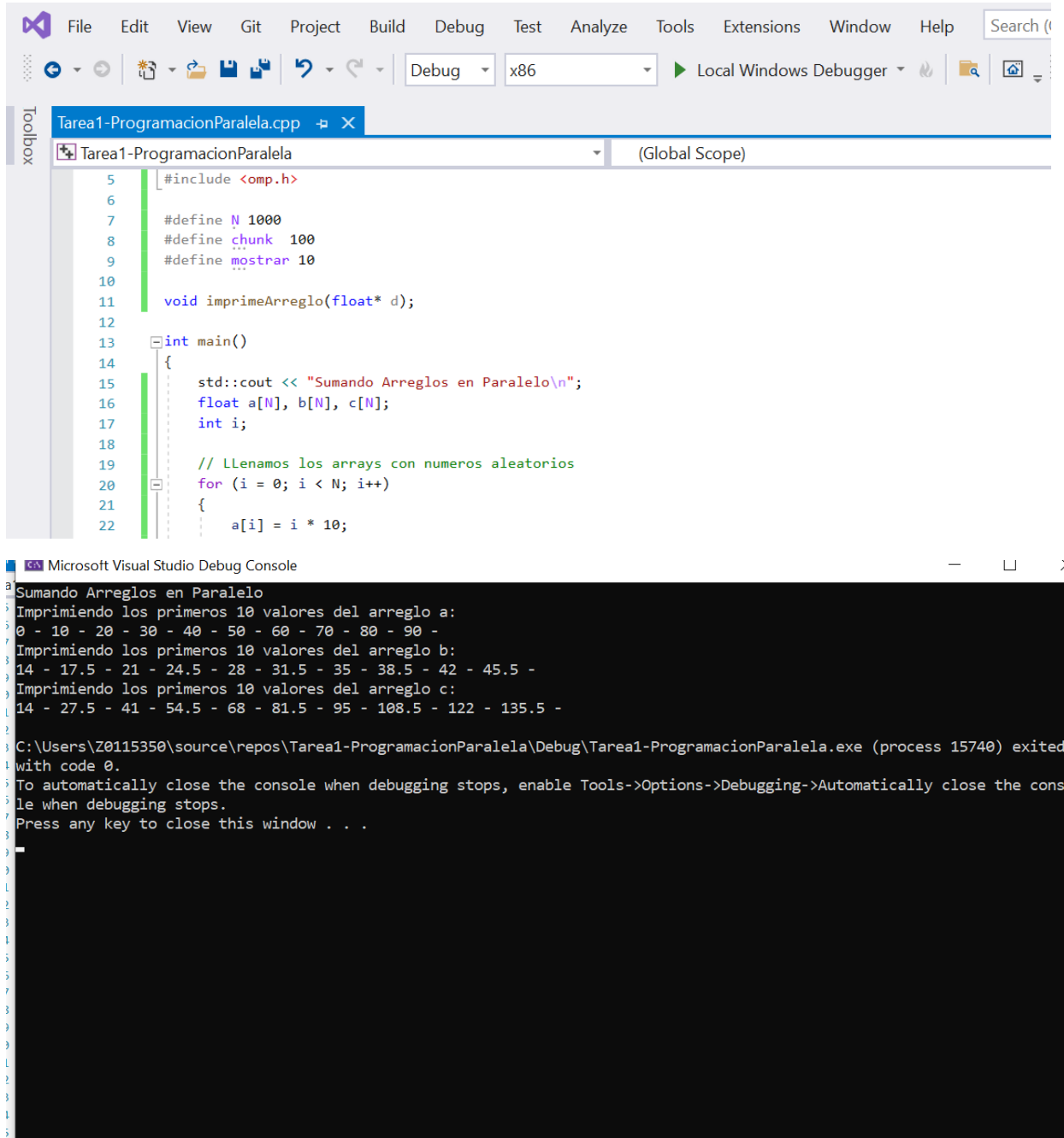
34      std::cout << "Imprimiendo los primeros " << mostrar << " valores del arreglo a: " << std::endl;
35      imprimeArreglo(a);
36      std::cout << "Imprimiendo los primeros " << mostrar << " valores del arreglo b: " << std::endl;
37      imprimeArreglo(b);
38      std::cout << "Imprimiendo los primeros " << mostrar << " valores del arreglo c: " << std::endl;
39      imprimeArreglo(c);
40  }
41
42  void imprimeArreglo(float *d)
43  {
44      for (int x = 0; x < mostrar; x++)
45          std::cout << d[x] << " - ";
46      std::cout << std::endl;
47  }

```

## Ejecución y resultados del código:

- En la primera ejecución use los valores: N=100, chunk=100, mostrar = 10.

Nótese como las sumas de cada elemento de los arreglos a y b las sumas se muestran correctamente. Ejemplo la suma de  $a[i]+b[i]$  es correcta.



```
#include <omp.h>

#define N 1000
#define chunk 100
#define mostrar 10

void imprimeArreglo(float* d);

int main()
{
    std::cout << "Sumando Arreglos en Paralelo\n";
    float a[N], b[N], c[N];
    int i;

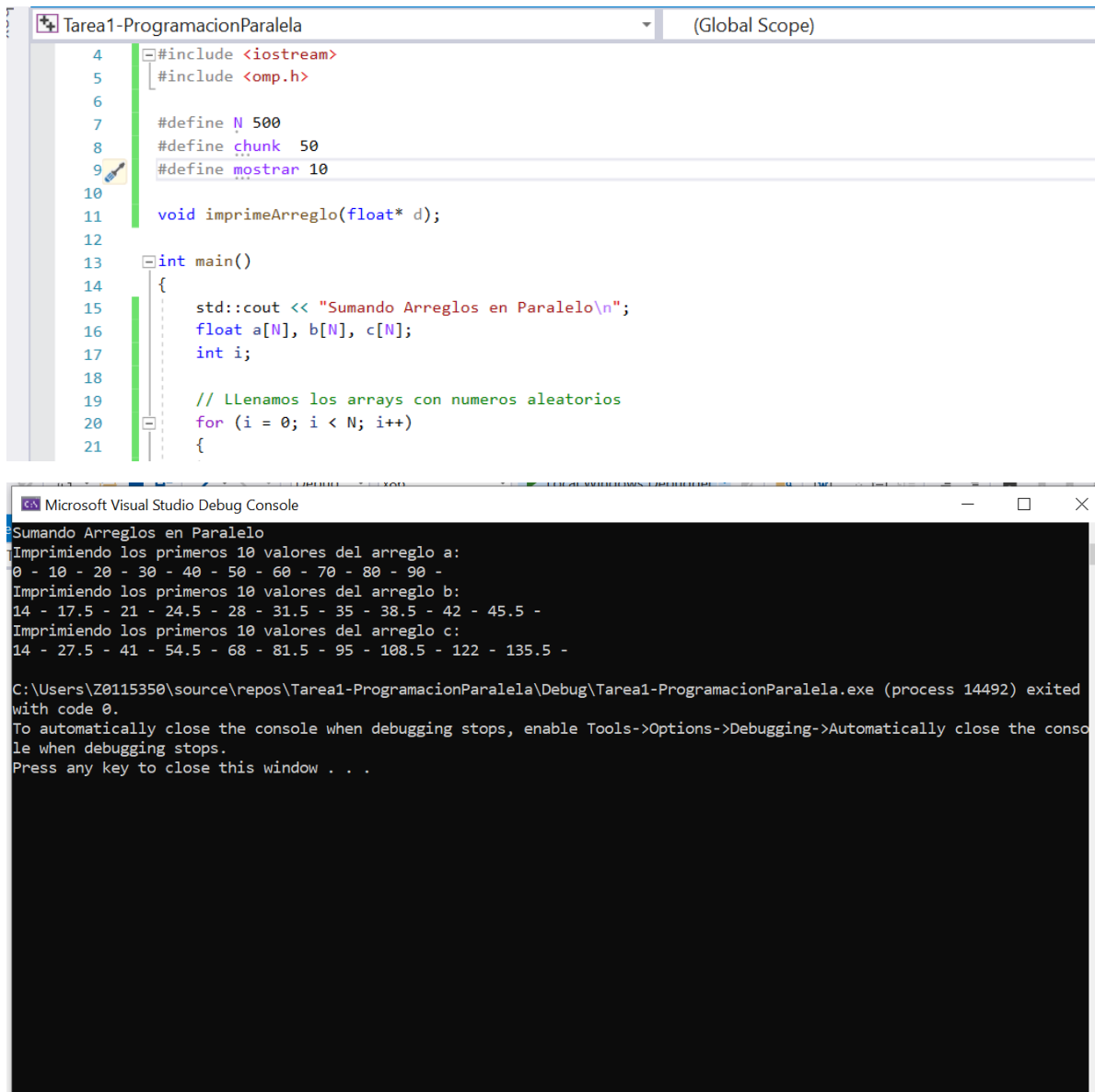
    // Llenamos los arrays con numeros aleatorios
    for (i = 0; i < N; i++)
    {
        a[i] = i * 10;
```

Microsoft Visual Studio Debug Console

```
Sumando Arreglos en Paralelo
Imprimiendo los primeros 10 valores del arreglo a:
0 - 10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 -
Imprimiendo los primeros 10 valores del arreglo b:
14 - 17.5 - 21 - 24.5 - 28 - 31.5 - 35 - 38.5 - 42 - 45.5 -
Imprimiendo los primeros 10 valores del arreglo c:
14 - 27.5 - 41 - 54.5 - 68 - 81.5 - 95 - 108.5 - 122 - 135.5 -
C:\Users\Z0115350\source\repos\Tarea1-ProgramacionParalela\Debug\Tarea1-ProgramacionParalela.exe (process 15740) exited
with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .
```

- En la primera ejecución use los valores: N=500, chunk=50, mostrar = 10.

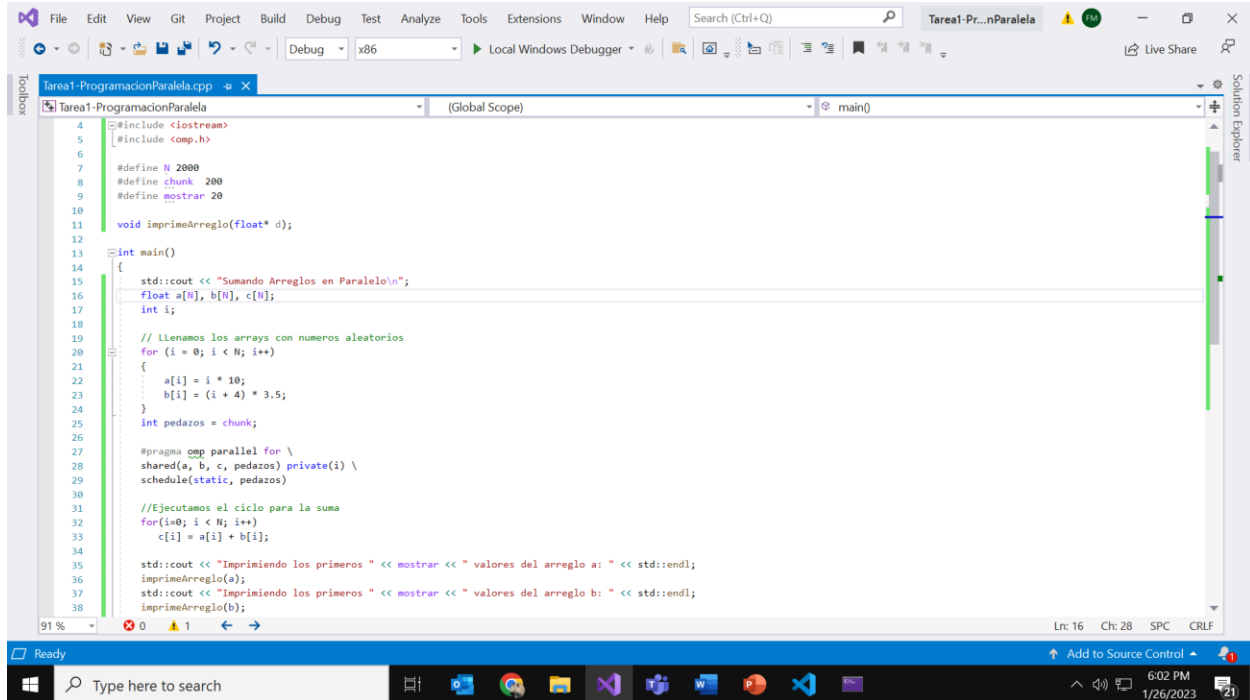
Pude percibir que el tiempo fue un poco mayor debido a los 500 elementos de cada arreglo, pero el incrementar el chunk ayudo a la velocidad de ejecución. Nótese como las sumas de cada elemento de los arreglos a y b las sumas se muestran correctamente. Ejemplo la suma de  $a[i]+b[i]$  es correcta.



```
Tarea1-ProgramacionParalela (Global Scope)
4  #include <iostream>
5  #include <omp.h>
6
7  #define N 500
8  #define chunk 50
9  #define mostrar 10
10
11 void imprimeArreglo(float* d);
12
13 int main()
14 {
15     std::cout << "Sumando Arreglos en Paralelo\n";
16     float a[N], b[N], c[N];
17     int i;
18
19     // Llenamos los arrays con numeros aleatorios
20     for (i = 0; i < N; i++)
21     {
```

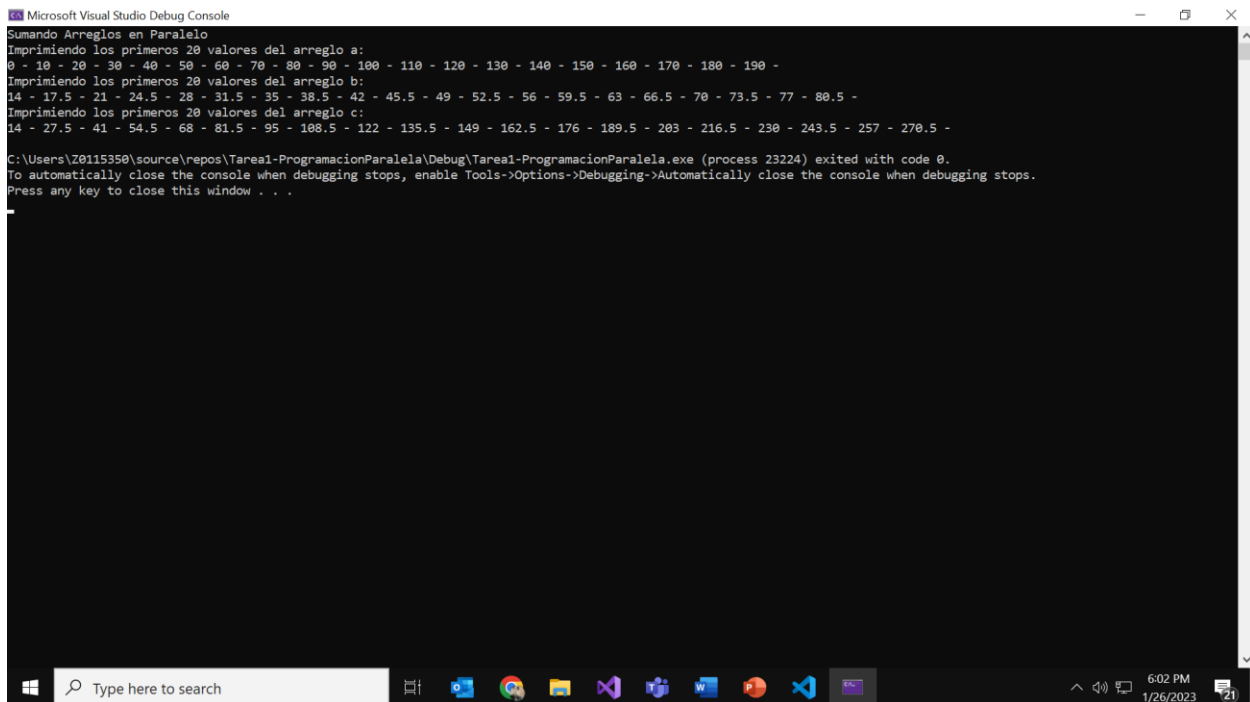
```
Microsoft Visual Studio Debug Console
Sumando Arreglos en Paralelo
Imprimiendo los primeros 10 valores del arreglo a:
0 - 10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 -
Imprimiendo los primeros 10 valores del arreglo b:
14 - 17.5 - 21 - 24.5 - 28 - 31.5 - 35 - 38.5 - 42 - 45.5 -
Imprimiendo los primeros 10 valores del arreglo c:
14 - 27.5 - 41 - 54.5 - 68 - 81.5 - 95 - 108.5 - 122 - 135.5 -

C:\Users\Z0115350\source\repos\Tarea1-ProgramacionParalela\Debug\Tarea1-ProgramacionParalela.exe (process 14492) exited
with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .
```



The screenshot shows the Visual Studio IDE with the file 'Tarea1-ProgramacionParalela.cpp' open. The code is in C++ and implements a parallel summation of array elements using OpenMP. The code includes headers for `<iostream>` and `<omp.h>`, and defines constants for the number of elements (`N = 2000`), the number of chunks (`chunk = 200`), and the number of threads to display (`mostrar = 20`). The `main` function initializes two arrays, `a` and `b`, with random values. It then uses a parallel `for` loop to calculate the sum of the elements in array `a` and store the result in array `c`. The code uses `#pragma omp parallel for` to parallelize the loop, with `shared(a, b, c, pedazos)` and `private(i)` clauses. The `schedule(static, pedazos)` clause is used to schedule the iterations. The code also includes a function `imprimeArreglo` to print the first 20 elements of an array. The output of the program is shown in the Debug Console window below the code editor.

```
4 #include <iostream>
5 #include <omp.h>
6
7 #define N 2000
8 #define chunk 200
9 #define mostrar 20
10
11 void imprimeArreglo(float* d);
12
13 int main()
14 {
15     std::cout << "Sumando Arreglos en Paralelo\n";
16     float a[N], b[N], c[N];
17     int i;
18
19     // Llenamos los arrays con numeros aleatorios
20     for (i = 0; i < N; i++)
21     {
22         a[i] = i * 10;
23         b[i] = (i + 4) * 3.5;
24     }
25     int pedazos = chunk;
26
27     #pragma omp parallel for \
28     shared(a, b, c, pedazos) private(i) \
29     schedule(static, pedazos)
30
31     //Ejecutamos el ciclo para la suma
32     for(i=0; i < N; i++)
33     {
34         c[i] = a[i] + b[i];
35     }
36
37     std::cout << "Imprimiendo los primeros " << mostrar << " valores del arreglo a: " << std::endl;
38     imprimeArreglo(a);
39     std::cout << "Imprimiendo los primeros " << mostrar << " valores del arreglo b: " << std::endl;
40     imprimeArreglo(b);
41 }
```



The screenshot shows the Microsoft Visual Studio Debug Console window. The output of the program is displayed, showing the sum of the elements in array `a` and the first 20 values of arrays `a` and `b`. The output is as follows:

```
Sumando Arreglos en Paralelo
Imprimiendo los primeros 20 valores del arreglo a:
0 - 10 - 20 - 30 - 40 - 50 - 60 - 70 - 80 - 90 - 100 - 110 - 120 - 130 - 140 - 150 - 160 - 170 - 180 - 190 -
Imprimiendo los primeros 20 valores del arreglo b:
14 - 17.5 - 21 - 24.5 - 28 - 31.5 - 35 - 38.5 - 42 - 45.5 - 49 - 52.5 - 56 - 59.5 - 63 - 66.5 - 70 - 73.5 - 77 - 80.5 -
Imprimiendo los primeros 20 valores del arreglo c:
14 - 27.5 - 41 - 54.5 - 68 - 81.5 - 95 - 108.5 - 122 - 135.5 - 149 - 162.5 - 176 - 189.5 - 203 - 216.5 - 230 - 243.5 - 257 - 270.5 -
C:\Users\Z0115350\source\repos\Tarea1-ProgramacionParalela\Debug\Tarea1-ProgramacionParalela.exe (process 23224) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

#### 4. Reflexión de la programación paralela:

La programación paralela ha sido la causa de grandes avances tecnológicos y es una gran técnica se requiere mejorar el rendimiento de un algoritmo. Pero no siempre es la mejor opción ya que no todo se puede ejecutar en paralelo hay tareas que requieren como entrada la salida de otras o tareas que necesitan acceder a los mismos recursos de memoria. También están los costes que se generan al implementar estos algoritmos. Aunque al final del día si hay muchas tareas que pueden ser optimizadas con paralelización.