Miguel Angel Aguilera
A00642541

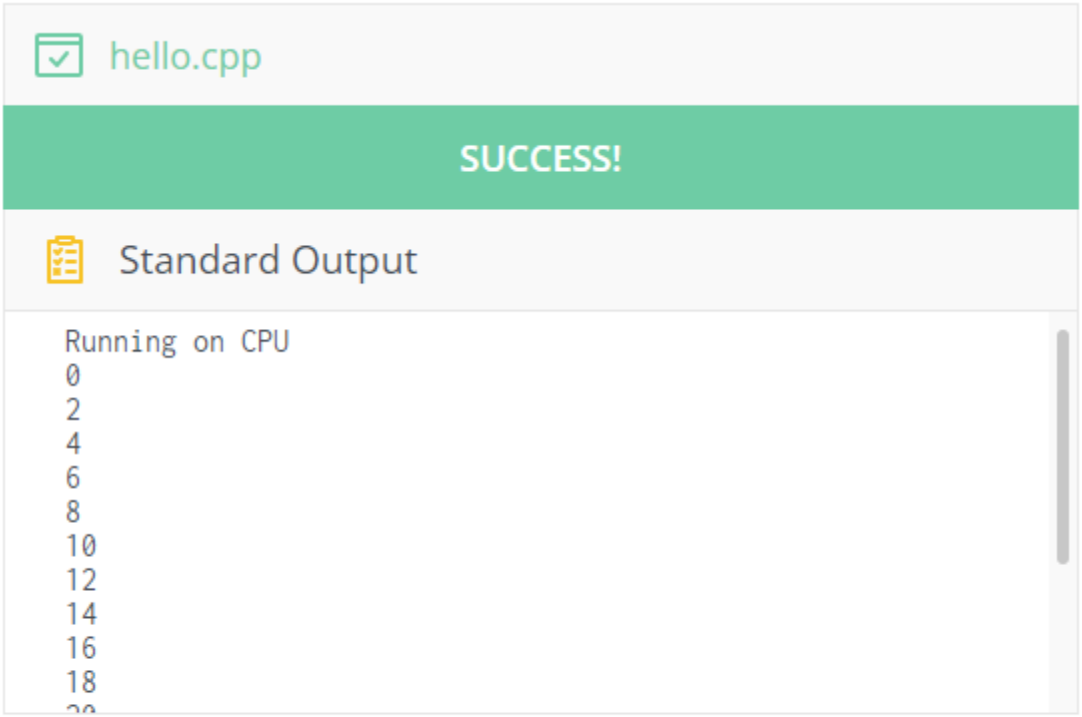
Tarea 1. Programación de una solución paralela

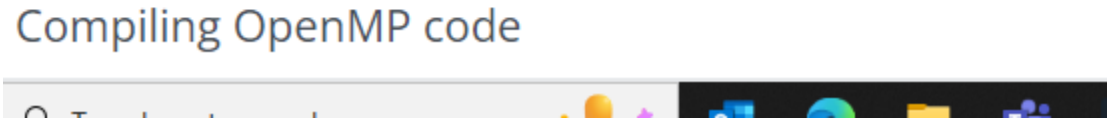
Introducción:

La programación paralela ha revolucionado la forma de ejecutar grandes tareas computacionales. Al dividir una gran lista de tareas en partes más pequeñas y manejables que pueden ejecutarse simultáneamente, la programación paralela permite un procesamiento más rápido y eficiente de grandes cantidades de datos. Con los avances tecnológicos y la creciente demanda de hardware, más rápidos y potentes, se ha convertido en una herramienta vital para resolver problemas en una amplia gama de campos, desde las simulaciones científicas y los modelos financieros hasta la predicción meteorológica y el diseño de videojuegos. En el mundo tecnológico actual, que avanza a gran velocidad, se ha convertido en un factor clave de innovación, ya que proporciona la potencia de cálculo necesaria para resolver algunos de los problemas más complejos a los que se enfrenta la humanidad.

```
#include "pch.h"
#include <iostream>
#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;

    for (i=0; i < N; i++)
    {
        a[i] = i * 10;
        b[i] = (i + 3)* 3.7;
    }
    int pedazos = chunk;
}

#pragma omp parallel for \
shared (a, b, c, pedazos) private (i) \
schedule (static, pedazos)

for (i=0; i<N; i++)
c[i] = a [i] + b [i];

std::cout << "imprimiendo los primeros" <<mostrar <<"valores del arreglo a:" <<std::endl;
imprimeArreglo(a);
std::cout << "imprimiendo los primeros" <<mostrar <<"valores del arreglo b:" <<std::endl;
imprimeArreglo(b);
std::cout << "imprimiendo los primeros" <<mostrar <<"valores del arreglo c:" <<std::endl;
imprimeArreglo(c)

void imprimeArreglo(float *d)
{
```

```
    for (int x=0; x <mostrar; x++)
    std::cout << d [x] << "-";
    std::cout << std::endl;
}
}
```





Liga GitHub:

https://github.com/MikerAguilera/Cloud-Computing/projects?query=is%3Aopen

Explicación de código:

En las primeras líneas del código se incluyen las librearías para ejecutarlo. La principal desde mi punto de vista es <omp> ya que nos ayudará a ejecutar de manera paralela una parte del código.

Después definimos el tamaño de los arreglos. Y <chunk> que nos ayuda a definir el número de elementos.

Al marcar la variable <i> evitamos el race condition y <pedazos> la operaciones que ejecutaremos.

La última función del void despliega los valores de los arreglos.

Conclusión:

La programación paralela ayuda en definitiva a resolver problemas de manera mucho más rápida principalmente porque el uso de varios procesadores están involucrados. El hecho de dividir tareas en partes independientes, permite que la ejecución de un código se mucho más acelerada y ante una mayor complejidad de problemas, esto es crucial.

Existen limites tanto tecnológicos como físicos al momento de buscar la optimización completa de una tarea, sin embargo este tipo de acercamiento a la solución de problemas, permiten darnos un respiro. Este tipo de tecnología permite poder cambiar nuestra manera de abordar un desarrollo, donde la optimización no se encuentra en el hardware, si no en la manera en la que se programa.