

Laboratorio No. 5

Definiciones

Cláusulas o directivas	Funcionalidad
private	La cláusula private permite que cada hilo dentro de la región paralelizada tenga una copia de una variable. Es importante entender que al utilizar esta cláusula, las copias de las variables no estarán inicializadas, es decir, no tendrán el valor de la variable original. La sintaxis consiste en encerrar entre paréntesis la lista de variables separadas por comas, para declararlas como privadas (Private clause, s.f.).
shared	Esta cláusula declara que uno o más elementos de una lista serán compartidos entre las tareas generadas por la estructura (o construct) en donde fue generada. Las variables compartidas no se duplican para realizar cada tarea, sino que todas las tareas acceden a la misma instancia de la variable, y por ello, se da correctamente la sincronización (Shared clause, s.f.).
firstprivate	Cada subproceso debe tener su propia instancia de la variable que necesite utilizar, a diferencia de la cláusula private, las copias de las variables sí están inicializadas, es decir, tienen el valor de la variable original. Esto es porque la variable ya existe antes de la construcción paralela (TylerMSFT, s.f.).
barrier	Esta cláusula permite sincronizar todos los hilos en un equipo, y todos los equipos paran de hacer su tarea (o ejecución) al llegar a la barrera, esto hasta que todos los hilos ejecutan la tarea. Esto permite coordinar fases de ejecución en un programa paralelo, para que no se ejecute una sección más avanzada en el código sin que se completen las tareas previas (TylerMSFT, 2022).
critical	Esta directiva restringe la ejecución de una porción de código, para que esta solo se pueda realizar por un único hilo (Critical construct, s.f.).
atomic	La directiva atomic asegura que una locación específica de memoria es accedida atómicamente, esto permite que solo un hilo pueda manejarla, y que no se produzcan valores indeterminados en la variable porque múltiples hilos realizaron operaciones sobre ella. Además, se emplea atomic sobre una única línea de código más que sobre una porción de

código (*Atomic Construct*, s.f.).

Programación

2.

```
#include <omp.h>

int main()
{
    const int N = 1000000;
    //se utiliza este tipo de dato para que se muestre el resultado de la suma correctamente
    long long int result = 0;
    double start_time, end_time;

    //capturamos el tiempo de ejecución antes de que comience el bloque paralelo
    start_time = omp_get_wtime();

    //usamos la directiva for y la cláusula reduction con el operador + por la suma
    //el for inicia en 1 y termina en N para sumar los primeros N numeros naturales
    #pragma omp parallel for reduction(+ : result)
    for (int i = 1; i <= N; i++)
    {
        result += i;
    }

    //capturamos el tiempo de ejecución cuando termina el bloque paralelo
    end_time = omp_get_wtime();

    printf("La suma es: %lld tiempo de ejecución: %f s\n", result, end_time - start_time);
}
```

estire 4\cursos\microprocesadores\OpenMP\ejemplosClase> gcc -fopenmp for_paralelo.c -o for_par
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./for_par
La suma es: 500000500000 tiempo de ejecución: 0.005000 s
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase>

3.

```
int main()
{
    printf("\nEl factorial de 10 es %d", num, factorial(num));

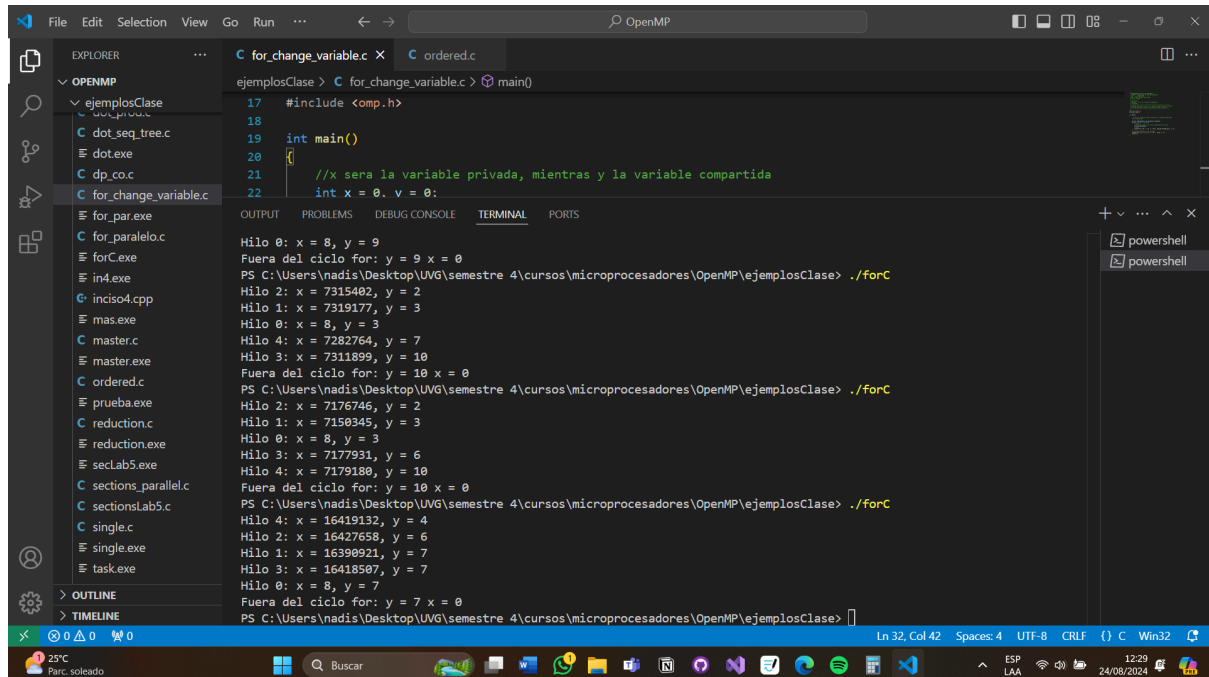
    #pragma omp section
    {
        printf("\nEl fibonacci de %d es %d", n, fib(n));
    }

    #pragma omp section
    {
        printf("\nEl numero mayor del arreglo es %d", nMax);
    }
}
```

PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> gcc -fopenmp sectionsLab5.c -o secLab5
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./secLab5
El fibonacci de 9 es 34
El numero mayor del arreglo es 215
El factorial de 10 es 3628800
Tiempo de ejecución: 0.001000 s
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase>

4.

Comportamiento sin directiva atomic

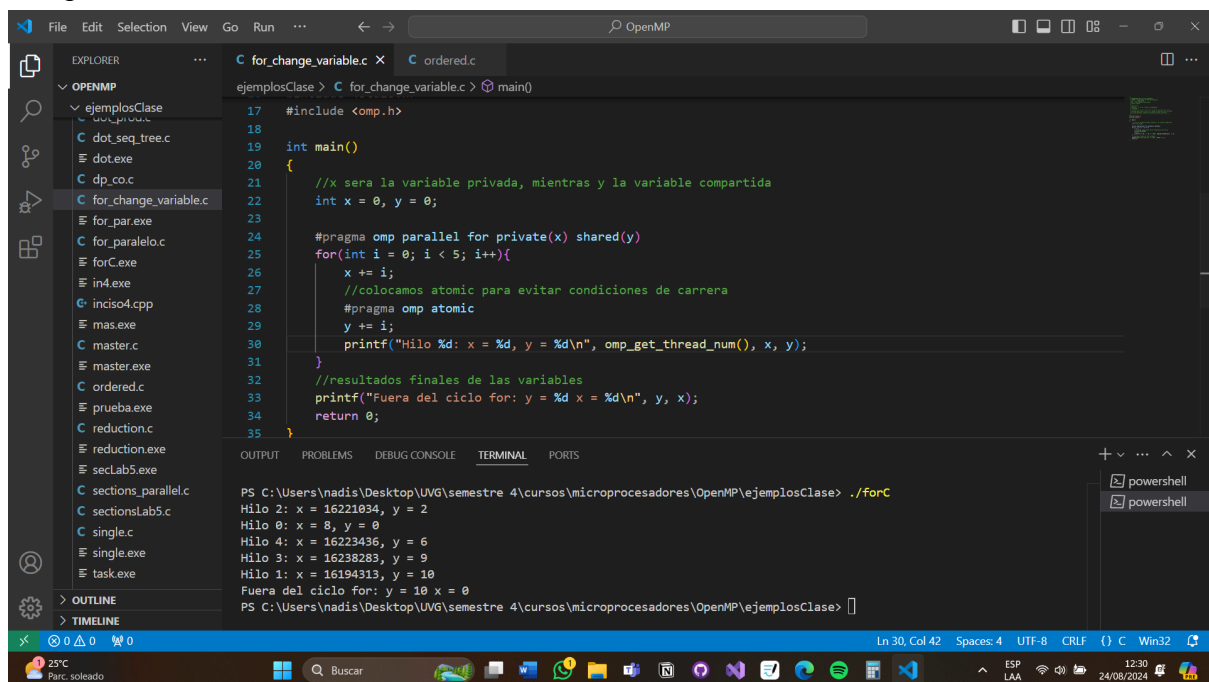


```
File Edit Selection View Go Run ... OpenMP
EXPLORER
  OPENMP
    ejemplosClase
      C_dot_seq_tree.c
      dot.exe
      dp.co.c
      C_for_change_variable.c
      for_par.exe
      for_paralelo.c
      forC.exe
      in4.exe
      inciso4.cpp
      mas.exe
      master.c
      master.exe
      ordered.c
      prueba.exe
      reduction.c
      reduction.exe
      seclab5.exe
      sections_parallel.c
      sectionsLab5.c
      single.c
      single.exe
      task.exe
  OUTLINE
  TIMELINE
  C_for_change_variable.c
    17 #include <omp.h>
    18
    19 int main()
    20 {
    21     //x sera la variable privada, mientras y la variable compartida
    22     int x = 0, y = 0;
    23
    24     for(int i = 0; i < 5; i++){
    25         y += i;
    26         //colocamos atomic para evitar condiciones de carrera
    27         #pragma omp atomic
    28         x += i;
    29         printf("Hilo %d: x = %d, y = %d\n", omp_get_thread_num(), x, y);
    30     }
    31
    32     //resultados finales de las variables
    33     printf("Fuera del ciclo for: y = %d x = %d\n", y, x);
    34     return 0;
    35 }
```

OUTPUT

```
Hilo 0: x = 8, y = 9
Fuera del ciclo for: y = 9 x = 0
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./forC
Hilo 2: x = 7315402, y = 2
Hilo 1: x = 7319177, y = 3
Hilo 0: x = 8, y = 3
Hilo 4: x = 7282764, y = 7
Hilo 3: x = 7311899, y = 10
Fuera del ciclo for: y = 10 x = 0
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./forC
Hilo 2: x = 7176746, y = 2
Hilo 1: x = 7150345, y = 3
Hilo 0: x = 8, y = 3
Hilo 3: x = 7177931, y = 6
Hilo 4: x = 7179180, y = 10
Fuera del ciclo for: y = 10 x = 0
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./forC
Hilo 4: x = 16419132, y = 4
Hilo 2: x = 16427658, y = 6
Hilo 1: x = 16390921, y = 7
Hilo 3: x = 16418507, y = 7
Hilo 0: x = 8, y = 7
Fuera del ciclo for: y = 7 x = 0
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase>
```

Comportamiento utilizando atomic



```
File Edit Selection View Go Run ... OpenMP
EXPLORER
  OPENMP
    ejemplosClase
      C_dot_seq_tree.c
      dot.exe
      dp.co.c
      C_for_change_variable.c
      for_par.exe
      for_paralelo.c
      forC.exe
      in4.exe
      inciso4.cpp
      mas.exe
      master.c
      master.exe
      ordered.c
      prueba.exe
      reduction.c
      reduction.exe
      seclab5.exe
      sections_parallel.c
      sectionsLab5.c
      single.c
      single.exe
      task.exe
  OUTLINE
  TIMELINE
  C_for_change_variable.c
    17 #include <omp.h>
    18
    19 int main()
    20 {
    21     //x sera la variable privada, mientras y la variable compartida
    22     int x = 0, y = 0;
    23
    24     #pragma omp parallel for private(x) shared(y)
    25     for(int i = 0; i < 5; i++){
    26         x += i;
    27         //colocamos atomic para evitar condiciones de carrera
    28         #pragma omp atomic
    29         y += i;
    30         printf("Hilo %d: x = %d, y = %d\n", omp_get_thread_num(), x, y);
    31     }
    32
    33     //resultados finales de las variables
    34     printf("Fuera del ciclo for: y = %d x = %d\n", y, x);
    35     return 0;
    36 }
```

OUTPUT

```
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./forC
Hilo 2: x = 16221034, y = 2
Hilo 0: x = 8, y = 0
Hilo 4: x = 16223436, y = 6
Hilo 3: x = 16238283, y = 9
Hilo 1: x = 16194313, y = 10
Fuera del ciclo for: y = 10 x = 0
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase>
```

Explicación del los resultados:

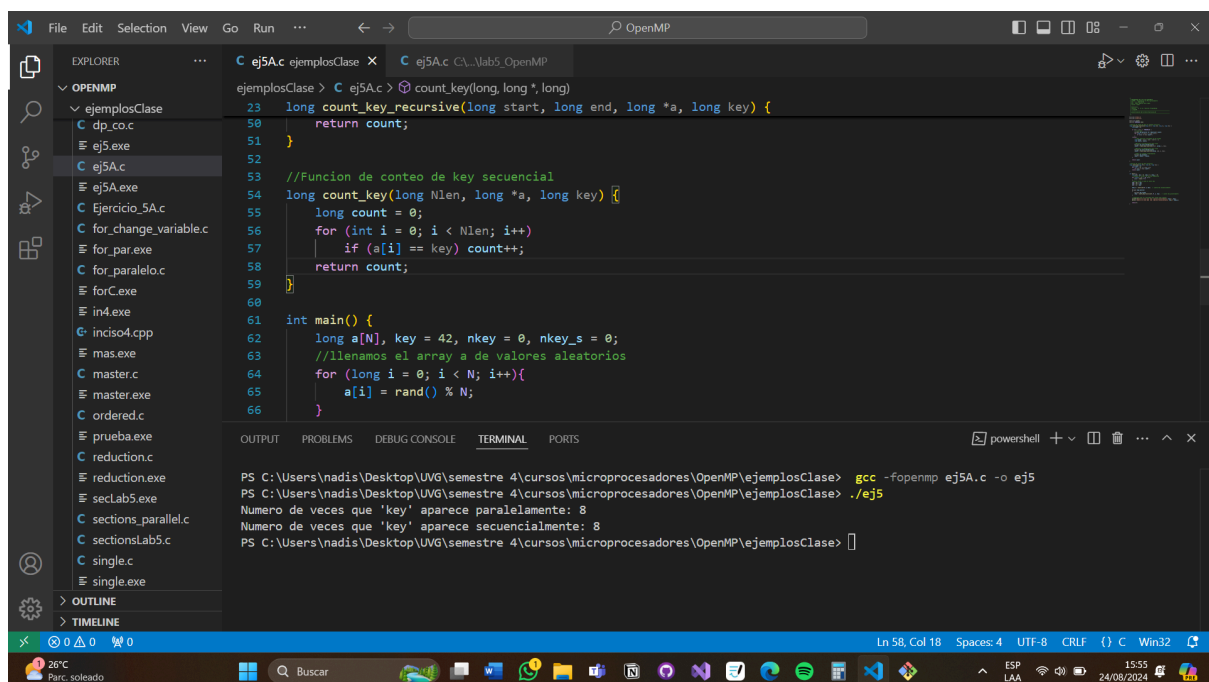
Cada hilo tiene su propia copia de x, entonces, cada hilo realiza operaciones en su propia versión de x y no hay interferencia entre hilos para esta variable. Es por esto que el valor de x puede parecer aleatorio o inusual cuando se imprime en consola, los hilos están modificando su copia local de x y no están sincronizados entre sí. Al final se observa que el valor de x es

cero x, esto es porque esta variable es privada para cada hilo y no se mantiene entre iteraciones ni se combina de ninguna manera.

La variable y es compartida entre todos los hilos. Cuando los hilos actualizan y sin protección (con la directiva atomic), pueden ocurrir condiciones de carrera. El uso de #pragma omp atomic asegura que la operación $y += i$ sea atómica, lo que previene interferencias entre hilos y asegura que y se actualice correctamente.

En conclusión, las variables privadas no afectan a otros hilos, pero sus resultados no se sincronizan, en cambio, las variables compartidas necesitan sincronización para evitar condiciones de carrera y obtener los resultados esperados.

5.



```
ejemplosClase > C ej5Ac > count_key(long, long *, long)
23 long count_key_recursive(long start, long end, long *a, long key) {
50     return count;
51 }
52
53 //Funcion de conteo de key secuencial
54 long count_key(long Nlen, long *a, long key) {
55     long count = 0;
56     for (int i = 0; i < Nlen; i++)
57         if (a[i] == key) count++;
58     return count;
59 }
60
61 int main() {
62     long a[N], key = 42, nkey = 0, nkey_s = 0;
63     //llenamos el array a de valores aleatorios
64     for (long i = 0; i < N; i++){
65         a[i] = rand() % N;
66     }
```

```
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> gcc -fopenmp ej5Ac.c -o ej5
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase> ./ej5
Numero de veces que 'key' aparece paralelamente: 8
Numero de veces que 'key' aparece secuencialmente: 8
PS C:\Users\nadis\Desktop\UVG\semestre 4\cursos\microprocesadores\OpenMP\ejemplosClase>
```

Referencias

private Clause. (s.f.). <https://www.openmp.org/spec-html/5.2/openmpsu37.html>

shared Clause. (s.f.). <https://www.openmp.org/spec-html/5.2/openmpsu36.html>

TylerMSFT. (s.f.). *Cláusulas de OpenMP*. Microsoft Learn.

<https://learn.microsoft.com/es-es/cpp/parallel/openmp/reference/openmp-clauses?view=msvc-170#firstprivate>

TylerMSFT. (2022, May 17). *OpenMP directives*. Microsoft Learn.

<https://learn.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-directives?view=msvc-170>

critical Construct. (s.f.). <https://www.openmp.org/spec-html/5.0/openmpsu89.html>

atomic Construct. (s.f.). <https://www.openmp.org/spec-html/5.0/openmpsu95.html>