Pregunta 1:

Programa 1:

Bienvenido | Programa1_1.c 1 | Programa1_2.c 1

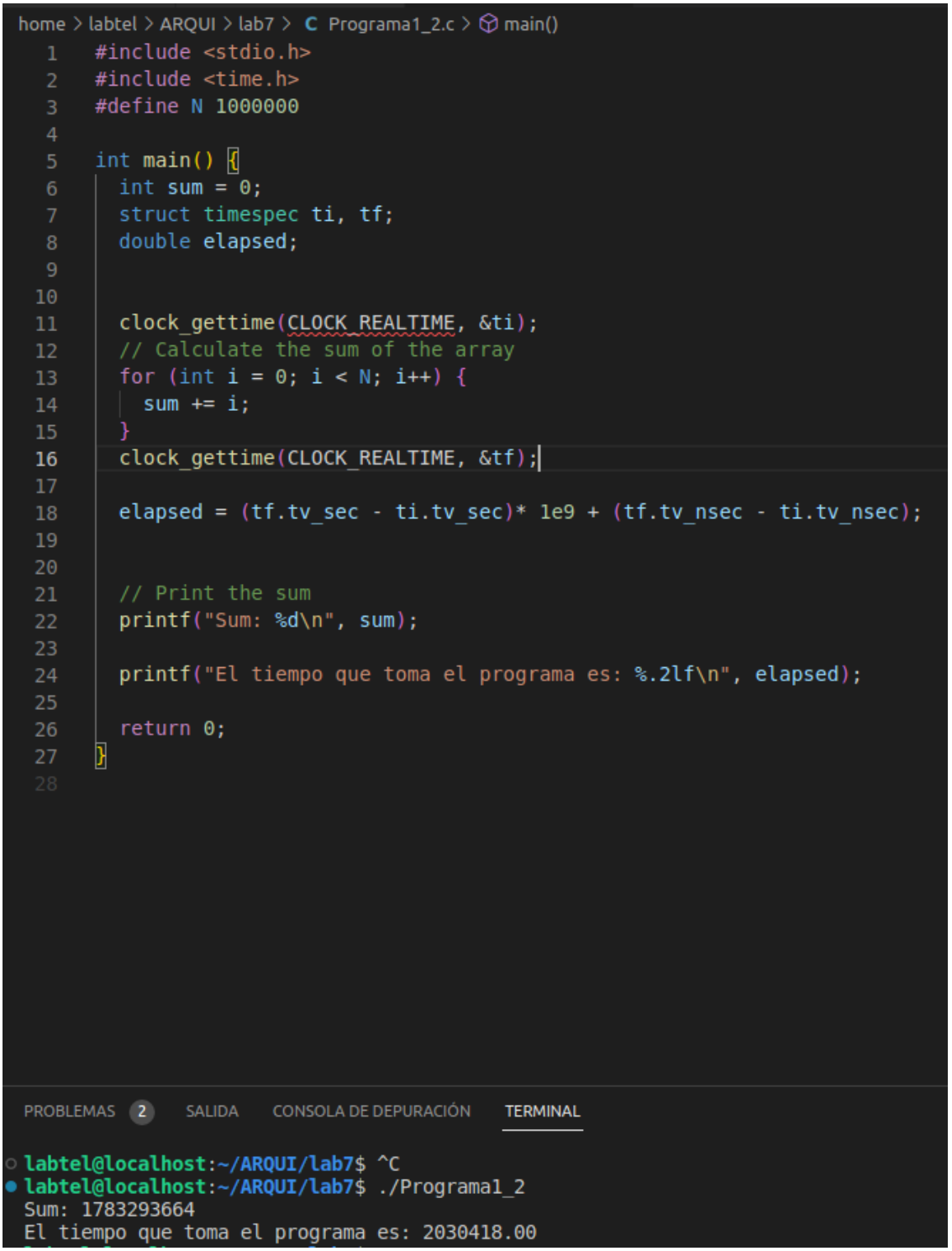
home > labtel > ARQUI > lab7 > Programa1_1.c > main()

```c
#include <stdio.h>
#include <time.h>

#define N 1000000

int main() {

  int sum = 0;
  int array[N];

  struct timespec ti, tf;
  double elapsed;


  // Initialize the array
  for (int i = 0; i < N; i++) {
    array[i] = i;
  }

  clock_gettime(CLOCK_REALTIME, &ti);
  // Calculate the sum of the array
  for (int i = 0; i < N; i++) {
    sum += array[i];
  }
  clock_gettime(CLOCK_REALTIME, &tf);

  elapsed = (tf.tv_sec - ti.tv_sec)* 1e9 + (tf.tv_nsec - ti.tv_nsec);


  // Print the sum
  printf("Sum: %d\n", sum);

  printf("El tiempo que toma el programa es: %.2lf\n", elapsed);


  return 0;
}

```

PROBLEMAS 2 | SALIDA | CONSOLA DE DEPURACIÓN | TERMINAL

```
labtel@localhost:~$ cd ARQUI
labtel@localhost:~/ARQUI$ cd lab7
labtel@localhost:~/ARQUI/lab7$ gcc Programa1_1.c -o Programa1_1
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1564607.00
```

Programa 2:

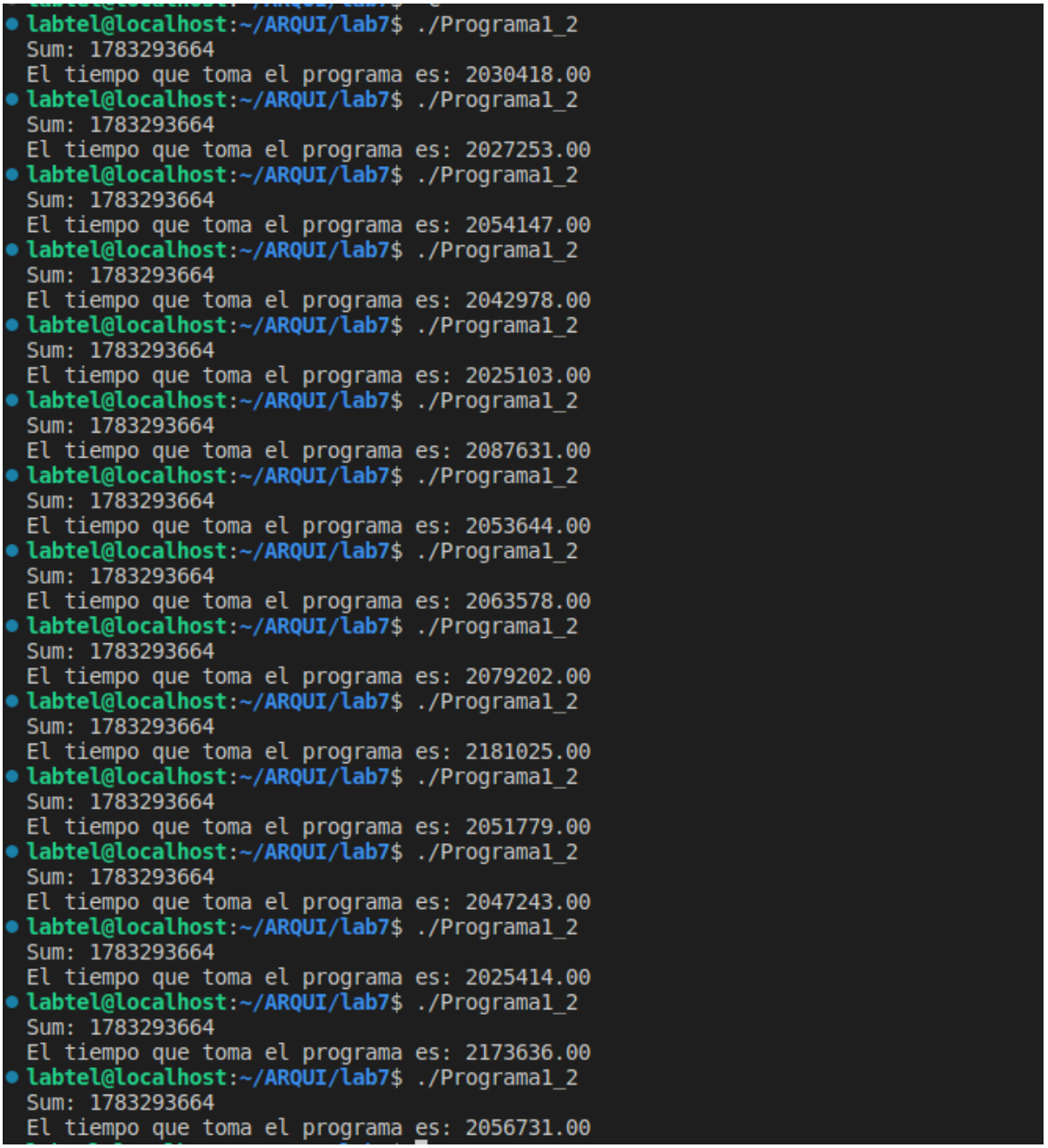

home > labtel > ARQUI > lab7 > Programa1_2.c > main()
#include <stdio.h>
#include <time.h>
#define N 1000000
int main() {
int sum = 0;
struct timespec ti, tf;
double elapsed;
clock_gettime(CLOCK_REALTIME, &ti);
// Calculate the sum of the array
for (int i = 0; i < N; i++) {
sum += i;
}
clock_gettime(CLOCK_REALTIME, &tf);
elapsed = (tf.tv_sec - ti.tv_sec)* 1e9 + (tf.tv_nsec - ti.tv_nsec);
// Print the sum
printf("Sum: %d\n", sum);
printf("El tiempo que toma el programa es: %.2lf\n", elapsed);
return 0;
}
PROBLEMAS 2
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
labtel@localhost:~/ARQUI/lab7$ ^C
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2030418.00

Pregunta 2:

Programa 1:

```
labtel@localhost:~/ARQUI/lab7$ gcc Programa1_1.c -o Programa1_1
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1564607.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1884896.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1974087.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1729238.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1839847.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1800844.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1804220.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 2013790.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1770032.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1904305.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1908530.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1849451.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1856008.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1691412.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_1
Sum: 1783293664
El tiempo que toma el programa es: 1802266.00
labtel@localhost:~/ARQUI/lab7$
```

Programa 2:


labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2030418.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2027253.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2054147.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2042978.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2025103.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2087631.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2053644.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2063578.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2079202.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2181025.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2051779.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2047243.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2025414.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2173636.00
labtel@localhost:~/ARQUI/lab7$ ./Programa1_2
Sum: 1783293664
El tiempo que toma el programa es: 2056731.00

Pregunta 3:

Tiempo de ejecución Programa 1 (ns)	Tiempo de ejecución Programa 2 (ns)
1564607.00	2030418.00
1884896.00	2027253.00
1974087.00	2054147.00
1729238.00	2042978.00
1839847.00	2025103.00
1800844.00	2087631.00
1804220.00	2053644.00
2013790.00	2063578.00
1770032.00	2079202.00
1904305.00	2181025.00
1908530.00	2051779.00
1849451.00	2047243.00
1856008.00	2025414.00
1691412.00	2173636.00
1802266.00	2056731.00

Pregunta 4:

```
// Initialize the array
for (int i = 0; i < N; i++) {
  array[i] = i;
}

clock_gettime(CLOCK_REALTIME, &ti);
// Calculate the sum of the array
for (int i = 0; i < N; i++) {
  sum += array[i];
}
clock_gettime(CLOCK_REALTIME, &tf);
```

La **localidad espacial** en el programa 1, se puede apreciar en la inicialización de los elementos del arreglo “array” en donde los elementos se inicializan de forma secuencial. Por ejemplo:

caso i = 0 array[0] = 0
caso i = 1 array[1] = 1
caso i = 2 array[2] = 2

.
.
.
.

caso i = N-1 array[N-1] = 999999

De esta manera, la memoria Caché accedió a los elementos del arreglo no solo a uno en particular, sino que también a los elementos adyacentes o vecinos (los elementos se acceden en bloque) que se encuentran en la memoria principal. Cabe destacar que en la suma de los elementos también hay un aprovechamiento de localidad espacial debido al empleo del arreglo “array”

La **localidad temporal** en el programa 1, se presenta en la parte en la que se calcula la suma de los elementos del arreglo mediante el comando for. Se puede observar que el bucle actualiza el valor de la variable “sum” en cada iteración.

caso i = 0 sum = sum + array[0]
caso i = 1 sum = sum + array[1]
caso i = 2 sum = sum + array[2]

.
.
.
.

caso i = N-1 sum = sum + array[N-1]

Entonces, esta variable es utilizada un pequeño instante después de haber sido utilizada o sea es muy probable que se vuelva a emplear en un futuro muy cercano.

Pregunta 5:

```
clock_gettime(CLOCK_REALTIME, &ti);
// Calculate the sum of the array
for (int i = 0; i < N; i++) {
  sum += i;
}
clock_gettime(CLOCK_REALTIME, &tf);

elapsed = (tf.tv_sec - ti.tv_sec)* 1e9 + (tf.tv_nsec - ti.tv_nsec);


// Print the sum
```

La **localidad temporal** en el programa 2, se presenta cuando se calcula la suma de los elementos del arreglo ya que el valor de la suma se actualiza en cada iteración.

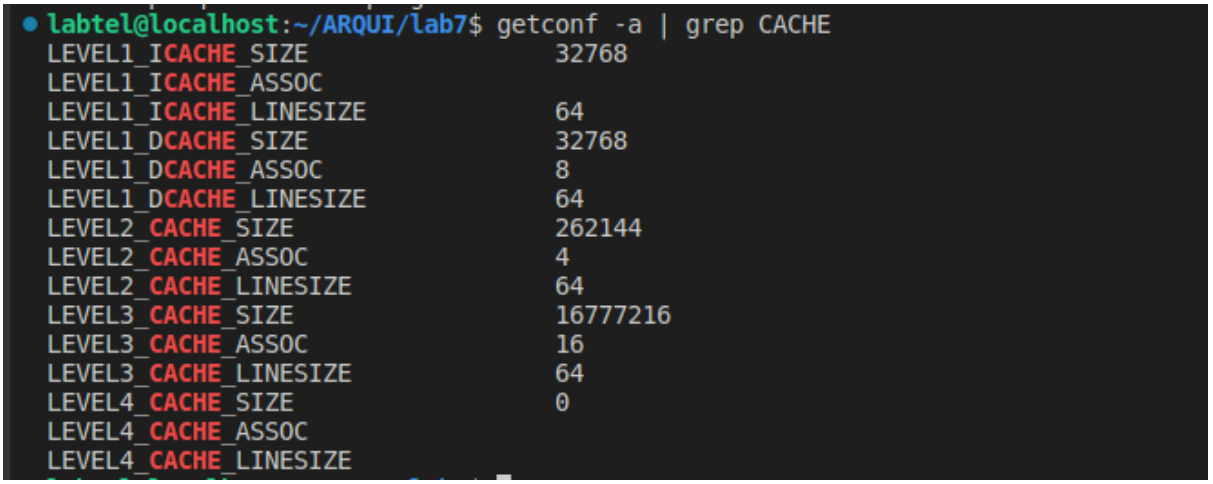
caso i = 0 sum = sum + 0
caso i = 1 sum = sum + 1
caso i = 2 sum = sum + 2

.
.
.
.

caso i = N-1 sum = sum + (N-1)

De esta manera, se observa que la variable “sum” se emplea en un tiempo futuro muy cercano.

Pregunta 6:



El tamaño del bloque en el nivel 1 es 64 Bytes.

Pregunta 7:

Respuesta en base al programa 1: En un primer momento, la memoria Caché puede estar o bien vacía o puede contener datos pero no los empleados, por lo que se puede predecir que en ese instante se producirá un Miss por parte de la memoria Caché. Luego, se producirá la localidad espacial ya que estamos inicializando el arreglo y se accederán a esos elementos en bloque, por lo que hay mucha probabilidad de haber Hits por parte de la memoria Caché. Sin embargo, después no es posible saber con exactitud ya que hay una cantidad de elementos enteros considerable (1 000 000) y cada uno pesa 4 Bytes, por lo que podemos predecir que en algún momento la memoria Caché se podría llenar y luego habría un Miss en Caché; y posteriormente, acceder a los elementos del arreglo en bloque que se necesitan en la memoria principal, por lo que se necesita tener información más detallada, específicamente hablando de las características del procesador y, por ende, de la memoria Caché para predecir el momento con exactitud cuando se produzca un Hit o Miss en la Caché.

Respuesta en base al programa 2: En primera instancia, la memoria Caché se puede encontrar vacía o contener datos ya guardados debido al uso de la PC, pero no los que vamos a emplear, por ello, se producirá, en ese instante, un Miss en memoria Caché. Posteriormente, ya guardada la variable en Caché, se deduce que habrán Hits por parte de la memoria Caché ya que la computadorá emplea el principio de localidad temporal.

Pregunta 8:

Luego de haber analizado los incisos anteriores en conjunto, se puede concluir que los mejores tiempos de ejecución son del programa 1, esto debido a que en el programa 1 hay un buen aprovechamiento de la localidad espacial respecto al arreglo “array” ya que en un primer momento se trabaja con la inicialización de los elementos del arreglo todos en tipo int. Asimismo, debido a la suma de elementos del arreglo, al emplear dicho arreglo, se aprovecha la localidad espacial respecto al arreglo “array” en la suma y también se aprovecha localidad temporal respecto a la variable “sum”. Debido a ello, se trabaja con mayor eficiencia aprovechando ambas localidades.

Por otro lado, si bien es cierto que se aprovecha una alta localidad temporal, al ser una variable escalar “sum”, la función carece de localidad espacial respecto a la variable “sum”. Asimismo, al ser también una variable escalar “i”, entonces podemos concluir que la propia función tiene una pobre localidad espacial pero alta localidad temporal, por lo que se podría implementar con más eficiencia también aprovechando localidad espacial.

Finalmente, concluimos que el programa 1 es más eficiente que el programa 2, por el aprovechamiento en ambas localidades.

Pregunta 9:

El tipo de dato para el arreglo “array” es int. Si se cambiara al tipo de dato char, se necesitaría la cuarta parte de la memoria para almacenar la misma cantidad de números (ya que 1 dato tipo char pesa 1 Byte y el int 4 Bytes). Sin embargo, el tiempo de ejecución podría ser mayor que cuando es int ya que para realizar la operación suma tendría que hacer una conversión a int por lo que demoraría más.

Asimismo, si se emplea tipo long en lugar de int, las operaciones de suma con números mucho más pesados generaría una mayor demora que cuando los elementos sean declarados en int

Por otro lado, si el tipo de dato es short, el resultado podría verse afectado (si bien es cierto emplea menos memoria ya que cada uno pesa 2 Bytes, su rango de valores es más limitado que int y para el valor de los numeros empleados se necesitan elementos más pesados que short) ya que si se utiliza podría haber un mal cálculo de la suma en el resultado final.

En conclusión, no esperaría resultados similares en el tiempo de ejecución tanto para char ni long ya que serían mayores que int y mucho menos del tipo de dato short ya que realizaría un mal cálculo de la suma.