Taller Inicial de Compilación, Ejecución y Uso de Optimizadores

Integrantes:

Adel Alvarez - 2191932
Yonathan Camilo Benítez Mancipe - 2204133

1.Compilelo usando usando gcc y ejecutelo.

Usando gcc

```
[ycbenitezm@guane09 taller]$ gcc jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m12.735s
user    0m0.000s
sys     0m0.001s
```

2.Use las opciones -01, -02 y -03 generando un ejecutable, por ejemplo de cada manera: suejecutable01.exe ¿Que observa en el comportamiento del código tanto al compilarlo como al ejecutar?

Usando -01

```
[ycbenitezm@guane09 taller]$ gcc -o1 jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m4.919s
user    0m0.000s
sys     0m0.001s
```

Usando -02

```
[ycbenitezm@guane09 taller]$ gcc -o2 jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m4.828s
user    0m0.000s
sys     0m0.001s
```

Usando -03

```
[ycbenitezm@guane09 taller]$ gcc -o3 jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m4.737s
user    0m0.000s
sys     0m0.001s
```

RTA: Se observa una ligera disminución en el tiempo de ejecución a medida que se aumenta el nivel de optimización, lo que es consistente con la expectativa general de que los niveles más altos de optimización deberían producir un código más eficiente y, por lo tanto, tiempos de ejecución más rápidos.

3. ¿Que pasa si usa las opciones -O, -O0, -Ofast -Og y Oz?

Usando -O

```
[ycbenitezm@guane09 taller]$ gcc -O jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m4.214s
user    0m0.002s
sys     0m0.000s
```

Usando -O0

```
[ycbenitezm@guane09 taller]$ gcc -O0 jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

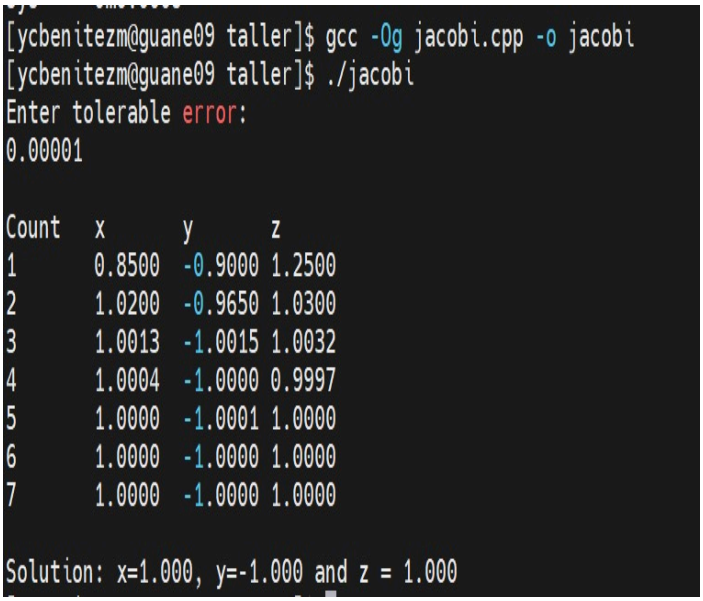
```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m4.110s
user    0m0.001s
sys     0m0.000s
```

Usando Ofast

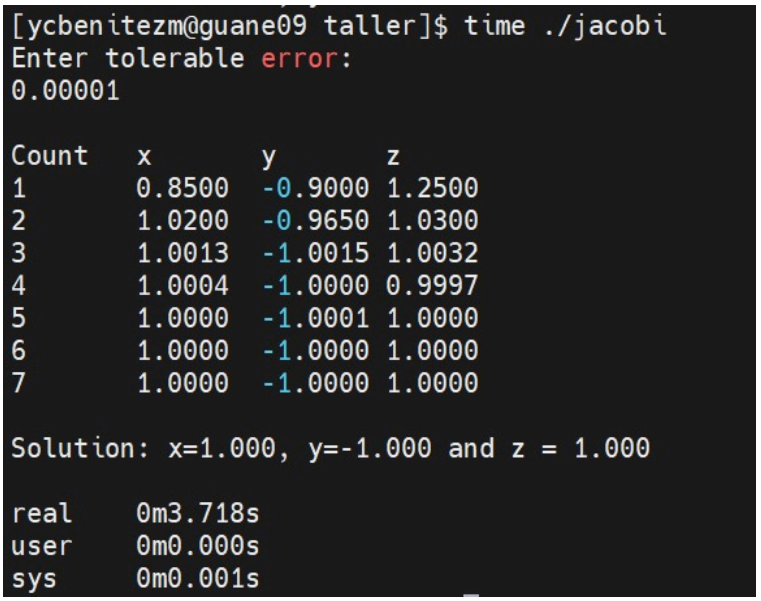
```
[ycbenitezm@guane09 taller]$ gcc -Ofast jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0033
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0033
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m3.559s
user    0m0.001s
sys     0m0.000s
```

Usando Og

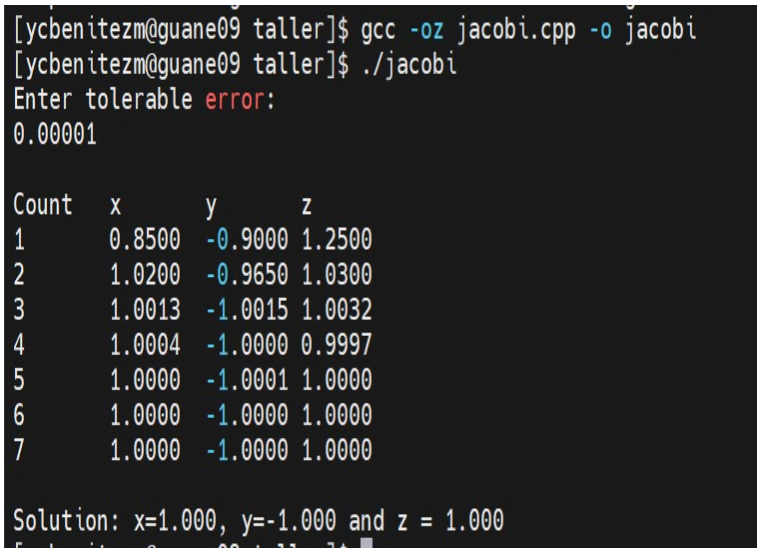


```
[ycbenitezm@guane09 taller]$ gcc -Og jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

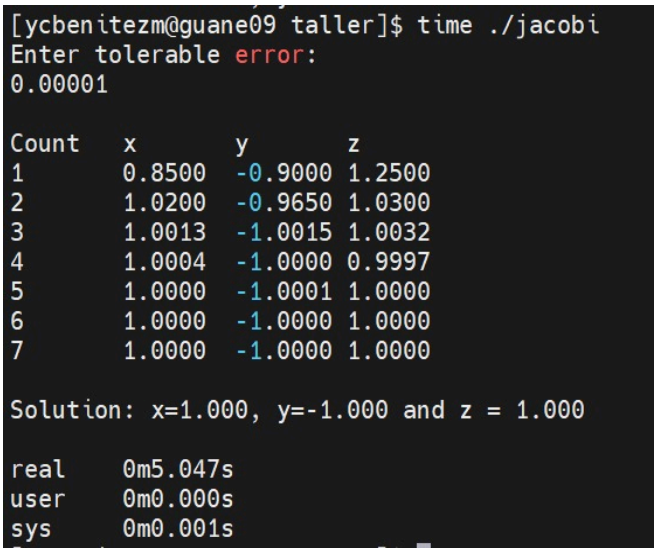


```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m3.718s
user    0m0.000s
sys     0m0.001s
```

Usando -Oz



```
[ycbenitezm@guane09 taller]$ gcc -oz jacobi.cpp -o jacobi
[ycbenitezm@guane09 taller]$ ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```



```
[ycbenitezm@guane09 taller]$ time ./jacobi
Enter tolerable error:
0.00001

Count   x       y       z
1       0.8500  -0.9000 1.2500
2       1.0200  -0.9650 1.0300
3       1.0013  -1.0015 1.0032
4       1.0004  -1.0000 0.9997
5       1.0000  -1.0001 1.0000
6       1.0000  -1.0000 1.0000
7       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000

real    0m5.047s
user    0m0.000s
sys     0m0.001s
```

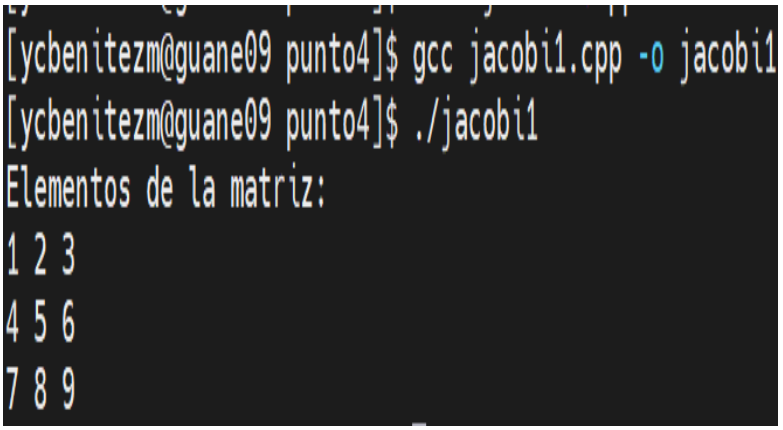
RTA:
-O: Este es el nivel de optimización por defecto.
Se observa que es ligeramente más rápido que algunos niveles específicos como -O0 y -Oz, pero más lento que otros como -Ofast y -Og.
los resultados muestran que el nivel de optimización -Ofast produce el tiempo de ejecución más rápido en este caso particular, seguido por -Og.

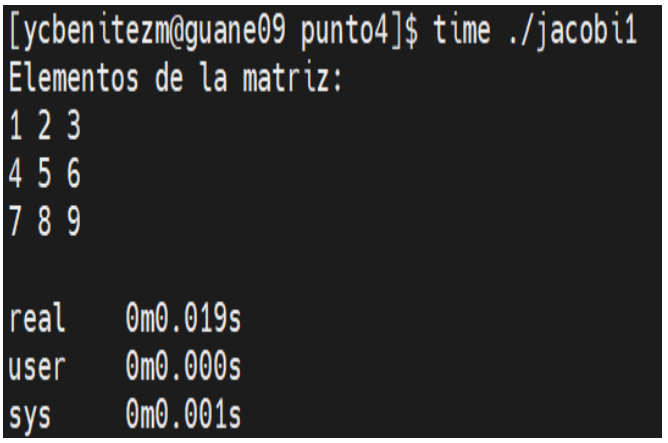
4. Busqué por internet un código de ejemplo simple que use punteros y repita los puntos del 1 al 3. NO PUEDEN HABER DOS CODIGOS IGUALES POR PAREJAS DEL CURSO POR LO QUE DEBEN PONERSE DE ACUERDO CON SUS COMPAÑEROS DE NO REPETIR.

Este es el codigo que empleamos para punteros:

```c
#include <stdio.h>

int main() {
    int matriz[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int *puntero = &matriz[0][0]; // Puntero apuntando al primer elemento de la matriz

    printf("Elementos de la matriz:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", *(puntero + i * 3 + j)); // Acceso al elemento usando aritmética de punteros
        }
        printf("\n");
    }

    return 0;
}
```

- Compilelo usando usando gcc y ejecutelo.
  Usando gcc





- Use las opciones -01, -02 y -03 generando un ejecutable, por ejemplo de cada manera: suejecutable01.exe ¿Que observa en el comportamiento del código tanto al compilarlo como al ejecutar?

Usando -O1

```
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
```

```
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.003s
user    0m0.000s
sys     0m0.001s
```

Usando -O2

```
[ycbenitezm@guane09 punto4]$ gcc -o2 jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.010s
user    0m0.001s
sys     0m0.000s
```

Usando -O3

```
[ycbenitezm@guane09 punto4]$ gcc -o3 jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.007s
user    0m0.000s
sys     0m0.001s
```

Con -O1: El tiempo de ejecución es el más rápido, indicando que la optimización nivel 1 produce un código más eficiente para este programa en particular.

Con -O2: El tiempo de ejecución es ligeramente más lento que con -O1, lo que podría ser atribuido a la aplicación de optimizaciones adicionales que, en este caso particular, no brindan beneficios significativos.

Con -O3: Aunque es más rápido que -O2, el tiempo de ejecución es un poco más lento que -O1. Esto puede indicar que algunas de las optimizaciones agresivas aplicadas en -O3 no son tan efectivas para este código en particular.

- ¿Que pasa si usa las opciones -O, -O0, -Ofast -Og y Oz?

  Usando -O

```
[ycbenitezm@guane09 punto4]$ gcc -O jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.008s
user    0m0.001s
sys     0m0.000s
```

  Usando -O0

```
[ycbenitezm@guane09 punto4]$ gcc -O0 jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.004s
user    0m0.000s
sys     0m0.001s
```

Usando -Ofast

```
[ycbenitezm@guane09 punto4]$ gcc -Ofast jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.005s
user    0m0.001s
sys     0m0.000s
```

Usando -Og

```
[ycbenitezm@guane09 punto4]$ gcc -Og jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.011s
user    0m0.000s
sys     0m0.001s
```

Usando -Oz

```
[ycbenitezm@guane09 punto4]$ gcc -oz jacobi1.cpp -o jacobi1
[ycbenitezm@guane09 punto4]$ ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9
[ycbenitezm@guane09 punto4]$ time ./jacobi1
Elementos de la matriz:
1 2 3
4 5 6
7 8 9

real    0m0.006s
user    0m0.000s
sys     0m0.001s
```

-O: Este nivel de optimización produce el tiempo de ejecución más rápido, con un tiempo real de 0.008 segundos. Esto sugiere que las optimizaciones generales proporcionadas por -O son beneficiosas para este código.

-O0: Desactivar la optimización con -O0 resulta en un tiempo de ejecución aún más rápido, con un tiempo real de 0.004 segundos. Esto puede ser sorprendente, ya que -O0 generalmente produce código menos eficiente. Sin embargo, en este caso, puede indicar que las optimizaciones no son necesarias o incluso perjudiciales para este código simple.

-Ofast: Este nivel de optimización produce un tiempo de ejecución similar al de -O0, con un tiempo real de 0.005 segundos. Aunque se esperaría que fuera más rápido que -O0, el código generado puede no aprovechar completamente las optimizaciones permitidas por -Ofast en este caso particular.

-Og: Este nivel de optimización, centrado en la facilidad de depuración, resulta en un tiempo de ejecución más lento que los otros niveles, con un tiempo real de 0.011 segundos. Esto se debe probablemente a que -Og no realiza optimizaciones agresivas que podrían acelerar el código.

-Oz: Este nivel de optimización, diseñado para minimizar el tamaño del código, produce un tiempo de ejecución similar al de -O, con un tiempo real de 0.006 segundos. Aunque se espera que -Oz produzca un código más pequeño a costa de un posible rendimiento, en este caso, el tiempo de ejecución es comparable al de -O.