Objetivos de la Guía

- Recordar el uso arreglos
- Recordar el uso de vectores
- Reforzar sobre la complejidad algorítmica.

> A - Javier y los arreglos locos

Javier esta loco por los arreglos, de hecho le gusta mucho recorrer arreglos y aplicar operaciones sobre los elementos de estos.

Por eso, Javier corre para hacer varios programas al respecto.

* Parte 1: Sumar el mismo elemento

Javier primero quiere realizar un programa que reciba una secuencia de arreglos y quiere que a cada elemento de arreglos se le sume el mismo elemento.

Por ejemplo, si se recibe el arreglo `[1, 3, 2, 4]`. Entonces, al elemento en la primera posición se le debe sumar 1, al elemento en la segunda posición se le debe sumar 3, al elemento en la tercera posición se le debe sumar 2, finalmente al último elemento se le debe sumar 4. Dando como resultado el siguiente arreglo `[2, 6, 4, 8]`.

Para esto, Javier te dice que se ingresará por pantalla un entero n que consiste en la cantidad de elementos del arreglo. Luego, en la siguiente línea se encontrarán n enteros separados por un espacio. Javier quiere que se vea por pantalla el nuevo arreglo.

* Input:

```
5
1 2 1 0 5
```

* Output:

```
2 4 2 0 10
```

Tip 1

Declarar un vector se realiza con la siguiente línea `vector< T > v;`, dónde `T` corresponde al tipo de dato.

Tip 2

Para leer n enteros recuerda que se puede hacer lo siguiente:

```
// esto te permite generar un arreglo de tamaño n
vector< int > v(n);
// almacenamos directamente en el arreglo el elemento leido por pantalla
for(int i = 0; i < n; i++) {
  cin >> v[i];
}
```

* Parte 2: Comparar!

Javier también le gustan comparar cosas. El toma dos arreglos (`a` y `b`, ambos de mismo tamaño. Quiere crear un tercer arreglo (`c`) del mismo tamaño en el cual en la posición `i` de ese arreglo almacene el resultado de `a > b`.

Para esto, Javier te dice que leerás por pantalla un entero n, que corresponde al tamaño de ambos arreglos. Luego, le sigue una línea con n números, correspondientes a los elementos del arreglo `a`. Finalmente, le sigue otra línea con n números, correspondiente a los elementos del arreglo `b`.

Se debe mostrar por pantalla los elementos del arreglo `c`.

* Input:

```
10
1 2 1 3 5 6 4 7 10 10
1 0 1 4 4 4 5 8 1 1
```

* Output:

```
0 1 0 0 1 1 0 0 1 1
```

Tip 1

Recuerda que la operacion > entrega un resultado booleano: verdadero (1) o falso (0).

Tip 2

Las posiciones en que el elemento de `a` es mayor al elemento de `b` son: 1, 4, 5, 8, 9.
Por eso en el arreglo resultado tiene un 1 en esas posiciones.

> B - Marcelo y Gabriel están complicados

Marcelo y Gabriel están complicados con el concepto de complejidad algorítmica. Para entenderlo mejor escriben ciertos códigos y quieren determinar su complejidad.

* Parte 1: Conceptos rapidos

Marcelo y Gabriel te piden que calcules aproximadamente gracias a la complejidad cuanto se demorara tu código aproximadamente en función de n. Y si es que podrían ser usados para un $n = 10000$.

- Código 1

```
for(int i = 0; i < n; i++) {
  sum++;
}
```

- Código 2

```
for(int i = 0; i < n; i++) {
  if(a > b) {
    sum++;
  }
}
```

- Código 3

```
for(int i = 0; i < n; i++) {
  for(int j = 0; j < n; j++) {
    sum++;
  }
}
```

Tip 1

Calcule cuantas veces se repiten los ciclos.

Tip 2

Recuerda que un segundo es aproximadamente 100000000 calculos.

* Parte 2: Hablando con Javier

Marcelo y Gabriel se enteran de los ejercicios realizados anteriormente sobre los arreglos con Javier. Por esto, te piden que determines cuanto se demorarían aproximadamente en función de los tamaños de lo arreglos.