2. EJERCICIOS RESUELTOS.

2.1. Calcular el número de operaciones elementales del siguiente fragmento de programa C++.
(Es posible que un algoritmo realice las mismas operaciones en todos los casos (por la ausencia de ramas condicionales o por su equivalencia en operaciones)).

```
1:   for ( i=0; i<n; i++ )          1,1,2 (1 salto)
2:        A[i]++                    4 ops= 2 accesos,1 asig, 1 suma
```

Peor caso = Mejor caso = Caso medio

$$T_{ciclo} = T_{cuerpo} + T_{incremento} + T_{salto} + T_{cond} = 4 + 2 + 1 + 1 = 8$$

$$T = T_{ini} + T_{cond} + T_{salto} + \sum_{i=0}^{n-1} T_{ciclo} = 1 + 1 + 1 + \sum_{i=0}^{n-1} 8 = 8n + 3$$

for (i=0 ; i<n ; i++)
A[i]++ ⟶ A[i] = A[i] + 1

Vemos que no existen casos diferenciados, por tanto c.mejor = c.medio = c.peor

T(n) = inicialización + condición + $\sum$ ciclo + salto
T_{ciclo} = cuerpo + incremento + salto + condición

$T_{ciclo} = 4 + 2 + 1 + 1 = 8$
$T(n) = 1 + 1 + \sum_{i=0}^{n-1} (8) + 1$
$T(n) = 8n + 3$

2.2. Calcular el número de operaciones elementales (OE) del procedimiento sumador utilizando como tamaño de la entrada el valor del argumento.

procedimiento sumador(n)	funcion factorial(n)

procedimiento sumador(n)
i=1:
s=0;
mientras i ≤2*n **hacer**
s=s*factorial(i);
i=i+1;
fmientras
fprocedimiento
	funcion factorial(n)
	aux=1:
	i=n;
	mientras i >0 **hacer**
	aux=aux*i;
	i=i-1;
	fmientras
	devolver aux
	ffunción

$T_{factorial}$ = asignacion + asignación + n veces · (condición + (asignacion + multiplicación + asignación + resta) + salto) + salto + devolver

$T_{factorial} = 2 + 6n + 2 = 6n + 4$

$T_{sumador}$ = asignación + asignación + $\sum_{i=1}^{2n}$((comparación + multiplicación) + + asignacion + multiplicacion + llamada + $T_{factorial}$ + asignacion + suma + salto) + + comparación + multiplicación + salto

$$T_{sumador} = 2 + \sum_{i=1}^{2n} (8 + 6i + 4) + 3 = 5 + 2n(8+4) + \sum_{i=1}^{2n} (6i) = 24n + 5 + \sum_{i=1}^{2n} 6i$$

$$T_{sumador} = 24n + 5 + 6\frac{(2n+1)(2n-1+1)}{2} = 24n + 5 + 3(2n+1)2n = 24n + 5 + 6n(2n+1)$$

$$T_{sumador} = 24n + 5 + 12n^2 + 6n \; ; \quad T_{sumador} = 12n^2 + 30n + 5$$

2.3. Calcular el número de **operaciones elementales (OE)** del procedimiento de ordenación por inserción. Realizar el análisis del caso peor.

```
1.   procedimiento inserción(T[1..n]
2.      para i← 2 hasta n hacer
3.         x ← T[i]
4.         j ← i-1
5.         mientras j>0 AND x < T[j] hacer
6.            T[j+1] ← T[j]
7.            j ← j-1
8.         fmientras
9.         T[j+1]  ← x
10.     fpara
11.  fprocedimiento
```

Línea	**Operación: nº de veces**
2	Asignación inicial:1 vez (y salto)
2	Condición bucle: **n** veces
2	Incremento de **i**: **n-1** veces (y salto)
3,4,9	**n-1** veces
5	Condición del bucle: $\sum_{i=2}^{n} i$ veces
6,7	$\sum_{i=2}^{n}(i-1)$ veces

Complejidad del algoritmo en el caso peor:

$$T(n) = \text{inicialización} + \text{condición} + \sum_{i=2}^{n}\Big(\text{asignación} + \text{acceso} + \text{asignación} + \text{resta} + T_{mientras} + \text{suma} + \text{acceso} + \text{asignación} + \text{incremento} + \text{salto} + \text{condición}\Big) + \text{salto}$$

$$T_{mientras} = \text{condición} + \sum_{j=i-1}^{1}\Big(\text{suma} + \text{acceso} + \text{asignación} + \text{acceso} + \text{asignación} + \text{resta} + \text{salto} + \text{condición}\Big) + \text{salto}$$

$$T_{mientras} = 4 + \sum_{j=i-1}^{1}(11) + 1 = 5 + \sum_{j=i-1}^{1} 11$$

$$T(n) = 2 + \sum_{i=2}^{n}\Big(4 + \big(5 + \sum_{j=i-1}^{1} 11\big) + 3 + 2 + 2\Big) + 1$$

$$T(n) = 3 + \sum_{i=2}^{n}\Big(14 + \sum_{j=i-1}^{1} 11\Big) = 3 + \sum_{i=2}^{n}\big(14 + 11(i-1)\big) = 3 + \sum_{i=2}^{n}\big(14 + 11i - 11\big)$$

$$T(n) = 3 + 3(n-1) + 11\,\frac{(n+2)(n-2+1)}{2}; \quad T(n) = 3 + 3n - 3 + 11\,\frac{n^2 - n + 2n - 2}{2};$$

$$T(n) = 3n + 11\,\frac{n^2 + n - 2}{2}; \quad T(n) = \frac{11}{2}n^2 + \frac{17}{2}n - 11$$

2.4. Obtener la complejidad del procedimiento de ordenación por inserción por el método de la **instrucción característica**. Realizar el análisis del caso peor.

```
1.   procedimiento inserción(T[1..n])
2.      para i← 2 hasta n hacer
3.         x ← T[i]
4.         j ← i-1
5.         mientras j>0 AND x < T[j] hacer
6.            T[j+1] ← T[j]
7.            j ← j-1
8.         fmientras
9.         T[j+1]  ← x
10.     fpara
11.  fprocedimiento
```

- Utilizamos como instrucción característica la **comparación** del bucle **mientras** en la línea 5. Ésta instrucción se ejecuta ***i*** veces en cada iteración del bucle **para**

Complejidad del algoritmo en el caso peor:

El peor caso sería cuando el bucle mientras realiza todas las iteraciones posibles, es decir, que se realizan todas las comparaciones de la condición, desde i=2 hasta n, como dicta el bucle para.

$$T_{i.critica} = \sum_{i=2}^{n} i = \frac{(n+2)(n-2+1)}{2}$$

$$\boxed{\sum_{i=p}^{q} i = \frac{(q+p)\cdot(q-p+1)}{2}}$$

$$T_{i.critica} = \frac{n^2 - n + 2n - 2}{2} = \frac{n^2}{2} + \frac{n}{2} - 1$$

2.5. ALGORITMO DE BÚSQUEDA SECUENCIAL.

Para determinar el tiempo de ejecución, calculamos primero el número de operaciones elementales (OE) que se realizan:

líneas	`int BusquedaSecuencial (int T[ ],int n,int valor)`	nº OE	
	`{`		
1.	`int i=1;`	1	asignación
2.	`while (T[i] != valor && i<n) {`	4	Cond.Bucle(2comp., 1 acceso vector, 1 lógica)
3.	`i=i+1;`	2	incremento y asignación
4.	`}`		
5.	`if (T[i]==valor)`	2	1 condición y 1 acceso al vector
6.	`return i;`	1	si la condición se cumple
7.	`else return 0;`	1	cuando la condición es falsa.
	`}`		

Como observamos, dependiendo de la ejecución del bucle while, obtendremos un número de operaciones u otro, por tanto nos encontramos con una situación de varios casos:

caso peor: Cuando el bucle se ejecuta todas las veces posibles.

$T(n)$ = asignación + T_{while} + T_{if}

T_{while} = condición + $\sum_{i=1}^{n}$(asignación + suma + salto + condición) + salto

T_{if} = condición + devuelve ← [En este ejemplo da igual el if o el else ya que ambos son el mismo número de OE, si no usar el que mayor OE tenga]

$T_{while} = 4 + \sum_{i=1}^{n}(3+4)+1 = 5 + \sum_{i=1}^{n} 7 = 5 + (7n - 1 + 1) = 7n + 5$

$T_{if} = 2 + 1 = 3$

$T(n) = 1 + (7n+5) + 3 = 7n + 9$

caso medio: Cuando el bucle se ejecuta la mitad de las veces posible.

$T(n)$ = asignación + T_{while} + T_{if}

T_{while} = condición + $\sum_{i=1}^{n/2}$(asignación + suma + salto + condición) + salto

T_{if} = condición + devuelve ← [En este ejemplo da igual el if o el else ya que ambos son el mismo número de OE, si no usar el que mayor OE tenga]

$T_{while} = 4 + \sum_{i=1}^{n/2}(3+4)+1 = 5 + \sum_{i=1}^{n/2} 7 = 5 + (\frac{7}{2}n - 1 + 1) = \frac{7}{2}n + 5$

$T_{if} = 2 + 1 = 3$

$T(n) = 1 + (\frac{7}{2}n+5) + 3 = \frac{7}{2}n + 9$

caso mejor: Donde el bucle while solo ejecuta la condición y salta a la siguiente instrucción, ya que no se cumple la condición.

$T(n)$ = asignación + condición + salto + condición(if) + devuelve

$T(n) = 1 + 4 + 1 + 2 + 1 = 9$

2.6. ALGORITMO DE ORDENACIÓN POR BURBUJA

Para obtener el tiempo de ejecución, calcularemos primero el número de operaciones elementales (OE) que se realizan:

líneas	`void burbuja(int T[],int n)`	nº OE	
	`{`		
	`int i, j;`		
	`int aux;`		
1)	`for (i = 1; i < n - 1 ; i++) {`	2,2,1	en cada iteración del bucle (ciclo i): (2condición+2incremento+1salto)
2)	`for (j = n; j > i ; j-- ) {`	1,2,1	en cada iteración del bucle (ciclo j): 1condición+ 2incremento+ 1salto
3)	`if (T[j] < T[j-1]) {`	4	1 resta, 2 accesos a un vector, 1 comparación
4)	`aux = T[j] ;`	2	4) a 6) sólo se ejecutan si se cumple la condición y realizan un total de 9 OE
5)	`T[j] = T[j-1] ;`	4	
6)	`T[j-1] = aux ;`	3	
	`}`		
	`} // bucle j`	1,1,1	por la salida del bucle(inicialización+condición+salto)
	`} // bucle i`	1,2,1	por la salida del bucle(inicialización+condición+salto)
	`}`		

El tiempo del algoritmo será el de ejecución de la única instrucción que tiene, el bucle para i:

observando el algoritmo, nos damos cuenta que en función de las veces que se ejecuten las instrucciones del bloque if, tendremos diferentes casos:

caso mejor: no se cumple la condición del if

$$T(n) = \text{inicialización}1 + \text{condición}1 + \sum_{i=1}^{n-1} \left(T_{bucle2} + \text{incremento} + \text{salto} + \text{condición} \right) + \text{salto}$$

$$T_{bucle2} = \text{inicialización}2 + \text{condición}2 + \sum_{j=n}^{i} \left(\text{condicion}3 + \text{decremento} + \text{salto} + + \text{condición}2 \right) + \text{salto}$$

$$T_{bucle2} = 1 + 1 + \sum_{j=n}^{i} (4+2+1+1) + 1 = 3 + \sum_{j=n}^{i} 8 = 3 + 8(i+1) = 8i + 11$$

$$T(n) = 1 + 2 + \sum_{i=1}^{n-1} ((8i+11) + 2 + 1 + 2) + 1 = 4 + \sum_{i=1}^{n-1} (8i + 17)$$

$$T(n) = 4 + 16(n-1) + \sum_{i=1}^{n-1} 8i = 16n - 12 + 8\frac{(n-1+1)(n-1-1+1)}{2} = 16n - 12 + 4n(n-1)$$

$$T(n) = 16n - 12 + 4n^2 - 4n = 4n^2 - 12n - 12 = 2n^2 - 6n - 6$$

<u>caso peor:</u> Las instrucciones del bloque if se ejecutan siempre

$$T(n) = \text{inicialización}1 + \text{condición}1 + \sum_{i=1}^{n-1}\left(T_{bucle2} + \text{incremento} + \text{salto} + \text{condición}\right) + \text{salto}$$

$$T_{bucle2} = \text{inicialización}2 + \text{condición}2 + \sum_{j=n}^{i}\left(T_{if} + \text{decremento} + \text{salto} + \text{condición}2\right) + \text{salto}$$

$$T_{if} = \text{condición}3 + \text{asignacion} + \text{acceso} + \text{acceso} + \text{asignación} + \text{acceso} + \text{resta} + \text{acceso} + \text{resta} + \text{asignación}$$

$$T_{if} = 4 + 2 + 4 + 3 = 13$$

$$T_{bucle2} = 1 + 1 + \sum_{j=n}^{i}(13 + 2 + 1 + 1) + 1 = 3 + \sum_{j=n}^{i} 17 = 3 + 17(i+1) = 17i + 20$$

$$T(n) = 1 + 2 + \sum_{i=1}^{n-1}((17i + 20) + 2 + 1 + 2) + 1 = 4 + \sum_{i=1}^{n-1}(17i + 25)$$

$$T(n) = 4 + 25(n-1) + \sum_{i=1}^{n-1} 17i = 25n - 21 + 17\frac{n(n-1)}{2} = \frac{17}{2}n^2 - \frac{17}{2}n + 25n - 21$$

$$T(n) = \frac{17}{2}n^2 + \frac{33}{2}n - 21 = \frac{17}{6}n^2 + \frac{11}{2}n - 7$$

<u>caso medio:</u> cuando el bloque if se ejecuta la mitad de veces

$$T(n) = \text{inicialización}1 + \text{condición}1 + \sum_{i=1}^{n-1}\left(T_{bucle2} + \text{incremento} + \text{salto} + \text{condición}\right) + \text{salto}$$

$$T_{bucle2} = \text{inicialización}2 + \text{condición}2 + \sum_{j=n}^{i}\left(T_{if} + \text{decremento} + \text{salto} + \text{condición}2\right) + \text{salto}$$

$$T_{if} = \text{condición}3 + \text{asignacion} + \text{acceso} + \text{acceso} + \text{asignación} + \text{acceso} + \text{resta} + \text{acceso} + \text{resta} + \text{asignación}$$

$$T_{if} = 4 + (2 + 4 + 3)/2 = 4 + \frac{9}{2}$$

$$T_{bucle2} = 1 + 1 + \sum_{j=n}^{i}\left(\left(4 + \frac{9}{2}\right) + 2 + 1 + 1\right) + 1 = 3 + \sum_{j=n}^{i} 25/2 = \frac{25}{2}i + \frac{31}{2}$$

$$T(n) = 1 + 2 + \sum_{i=1}^{n-1}\left(\frac{25}{2}i + \frac{31}{2} + 2 + 1 + 2\right) + 1 = 4 + \sum_{i=1}^{n-1}\left(\frac{25}{2}i + \frac{41}{2}\right)$$

$$T(n) = 4 + \frac{41}{2}n - \frac{41}{2} + \sum_{i=1}^{n-1}\frac{25}{2}i = \frac{41}{2}n - \frac{33}{2} + \frac{25}{2}\,\frac{n(n-1)}{2}$$

$$T(n) = \frac{25}{4}n^2 - \frac{25}{4}n + \frac{41}{2}n - \frac{33}{2} = \frac{25}{4}n^2 + \frac{57}{4}n - \frac{33}{2}$$