1. INTRODUCCIÓN.

- El **tiempo de ejecución de un algoritmo** va a ser una **función** que mide el **número de operaciones** elementales (las que el ordenador realiza en un tiempo acotado) que realiza el algoritmo para un tamaño de entrada dado.
- Consideraremos **operaciones elementales** (contabilizándolas como 1 operación elemental) las siguientes:
    - Operaciones **aritmética**s básicas (+,-,*,/,%,…) y **operaciones relacionales** (<, >, |=, ==,…).
    - Asignaciones a variables de tipo predefinido.
    - Saltos:
        - Llamadas a funciones o procedimientos
        - Retorno desde funciones o procedimientos
    - Comparaciones lógicas (AND, OR, NOT..)
    - Accesos a estructuras indexadas básicas (vectores, matrices…)
- **EJEMPLOS**:

indice = -1;	1 Operación: asignación
salir = false;	1 Operación: asignación
i=0;	1 Operación: asignación
A[i] = 10;	2 Operaciones: asignación + acceso (en la i)
i = j + 1;	2 Operaciones: asignación + suma
A[i] = A[i] + k;	4 Operaciones: asignación + 2 accesos(2i) + operación suma
indice == -1;	1 operación relacional

- **Estructuras condicionales:** El tiempo de ejecución de la sentencia "**Si** C **entonces** S1 **otro** S2 **fsi**" es en el caso peor **T = T(C) + max{T(S1),T(S2)}**

1:	**Si** (x == 10) **entonces**	1 op: comparación
2:	i = 17	1 op: asignación (caso verdadero)
3:	**otro**	
4:	i= i+1	2 ops: asignación + suma (caso falso)
6:	**fsi**	

Operaciones del caso verdadero = *Tcond+Tcuerposi = 1+1 =2*

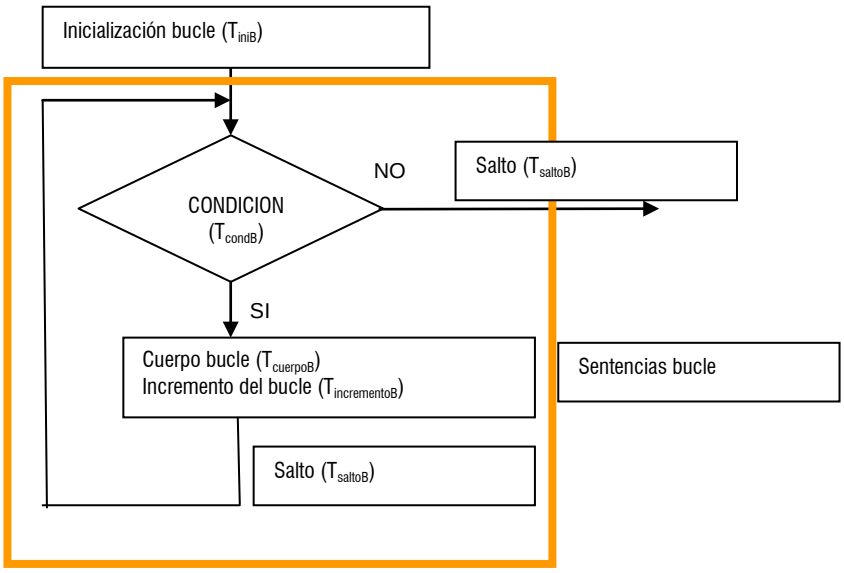
Operaciones del caso falso = *Tcond* + *Tcuerpono = 1+2=3*

1:	**Si** (x == 10) **entonces**	1 op: comparación
2:	i = 7	1 op: asignación (caso verdadero)
3:	**fsi**	

Operaciones del caso verdadero = *Tcond+Tcuerposi* = 1+1 =2

Operaciones del caso falso = *Tcond* = 1

- **Estructuras repetitivas:**



Para el **bucle B** tendríamos un tiempo total de ejecución:

$$T_{cicloB} = T_{cond} + T_{cuerpoB} + T_{incrementoB} + T_{salto}$$
$$T_B = T_{ini} + T_{cond} + T_{salto} + \sum_{(contador=vi;vf)} T_{cicloB}$$

- **Mientras:** El tiempo de ejecución del bucle "**mientras** C **hacer** S **fmientras**" es en el caso peor **T = T(C) + (nº iteraciones)*(T(S) + T(C))**
  (tanto T(C) como T(S) pueden variar en cada iteración, y por tanto habrá que tenerlo en cuenta para su cálculo).

1:	i=0;	1 op (inicialización)
2:	**mientras** (i<n)	1 op: Comparación:
3:	A[i] = 0;	2 op: asignación + acceso
4:	i=i+1;	2 op: asignación + suma
5:	**fmientras**	1 op: salto al comienzo
6:		1 op: salto condicional del final

Tiempo del cuerpo del bucle: $T_{cuerpo} = 2+2 = 4$
Tiempo del 1 ciclo: $T_{ciclo} = T_{cuerpo} + T_{salto} + T_{cond} = 4 + 1 + 1 = 6$
Tiempo total: $T_{ini} + T_{cond} + T_{salto} + \sum_{(i=0;n-1)} T_{ciclo} = 1+1+1+ \sum_{(i=0;n-1)} 6 = \mathbf{3+6n}$

- **Repetir:** El tiempo de ejecución del bucle "**repetir** S **hasta** C" es en el caso peor **T = (nº iteraciones)*(T(S) + T(C)).** Ahora consideraremos que siempre se realiza una evaluación del cuerpo del bucle y una evaluación de la condición. Por cada ciclo se ejecuta un salto, una evaluación del cuerpo del bucle y una evaluación de la condición.
  (tanto T(C) como T(S) pueden variar en cada iteración, y por tanto habrá que tenerlo en cuenta para su cálculo).

```
1:  i=0;                    1 op (inicialización)
2:   repetir
3:          A[i] = 0;       2 ops: asignación + acceso
4:          i=i+1;          2 ops: asignación + suma
5:  hasta   (i==n)          2 ops: 1 Comparación y 1 salto al comienzo
```

Tiempo del cuerpo del bucle: $T_{cuerpo} = 2+2 = 4$

Tiempo del 1 ciclo: $T_{ciclo} = T_{cuerpo} + T_{cond} + T_{salto} = 4 + 1 + 1 = 6$

Tiempo total:

$$T_{ini} + T_{cuerpo} + T_{cond} + \sum_{(i=1;n-1)} T_{ciclo} = 1+4+1+ \sum_{(i=1;n-1)} 6 = \mathbf{6+6(n-1)=6n}$$

Observar que se ha incrementado el número de instrucciones fijo (independiente del número de ejecuciones del cuerpo del bucle) en una evaluación del cuerpo del bucle y hemos cambiado el recorrido del índice de modo que hace una repetición menos. Esto no es exactamente equivalente al caso anterior ya que añade siempre las instrucciones necesarias para una evaluación del cuerpo del bucle.

- **Para:** (*for*): para el caso de C: `for( <inicialización>; <condición>; <incremento> ):`
  El bucle para de C++ es equivalente a:

```
<inicialización>
while ( <condición> ){
    … // Cuerpo del bucle
    <incremento>
}
```

  Con lo que el tiempo se calcula como en el caso del bucle mientras.

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

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

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

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)
        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
```

$$OE_{factorial}(n) = 2 + n(1+4+1) + 1 + 1 = 6n + 4$$

- Como en el procedimiento sumador dentro del bucle la llamada a factorial depende de la variable de control del bucle, debemos considerarlo a la hora de contar las OE del bucle. Así nos quedará:

$$OE_{sumador}(n) = 2 + \sum_{i=1}^{2n}\left(2 + 5 + 1 + OE_{factorial}(i)\right) + 2 + 1 = 5 + \sum_{i=1}^{2n}\left(8 + 6i + 4\right) =$$

$$5 + 24n + 6\frac{2n(2n+1)}{2} = 12n^2 + 30n + 5$$

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]	**Línea**	**Operación: nº de veces**
2.	**para** i← 2 **hasta** n **hacer**	2	Asignación inicial:1 vez (y salto)
3.	x ← T[i]	2	Condición bucle: **n** veces
4.	j ← i-1	2	Incremento de **i**: **n-1** veces (y salto)
5.	**mientras** j>0 AND x < T[j] **hacer**	3,4,9	**n-1** veces
6.	T[j+1] ← T[j]	5	Condición del bucle: $\sum_{i=2}^{n} \boldsymbol{i}$ veces
7.	j ← j-1	6,7	$\sum_{i=2}^{n}(\boldsymbol{i}-1)$ veces
8.	**fmientras**		
9.	T[j+1] ← x		
10.	**fpara**		
11.	**fprocedimiento**		

Complejidad del algoritmo en el caso peor:

$$\boldsymbol{OE(n)=T(n)=T_{b\_para}}$$

$$\boldsymbol{T_{b\_para}} = T_{ini} + T_{cond} + T_{salto} + \sum_{i=2}^{n} T_{ciclopara} = 1 + 1 + 1 + \sum_{i=2}^{n} T_{ciclopara} = 3 + \sum_{i=2}^{n} T_{ciclopara}$$

$$\boldsymbol{T_{ciclopara}} = T_{cuerpopara} + T_{incremento} + T_{salto} + T_{cond} = T_{cuerpopara} + 2 + 1 + 1 = \boldsymbol{T_{cuerpopara}} + 2$$

$$T_{cuerpopara} = T_{asig(3,4)} + \boldsymbol{T_{b\_mientras}} + T_{asig(9)} = 4 + \boldsymbol{T_{b\_mientras}} + 3 = 7 + \boldsymbol{T_{b\_mientras}}$$

$$\boldsymbol{T_{b\_mientras}} = T_{cond} + T_{salto} + \sum_{j=i-1}^{1} T_{ciclomien.} = 4 + 1 + \sum_{j=i-1}^{1} T_{ciclomien.} = 4 + \sum_{j=i-1}^{1} T_{ciclomien.}$$

$$\boldsymbol{T_{ciclomientras}} = T_{cuerpomientras} + T_{salto} + T_{cond} = T_{asig(6,7)} + 1 + 4 = 6 + 1 + 4 = \boldsymbol{11}$$

Por tanto:

$$\boldsymbol{OE(n) = T(n) = 3 + \sum_{i=2}^{n} \left(2 + 7 + 4 + \sum_{j=i-1}^{1} 11 \right) = 3 + \sum_{i=2}^{n} \left(13 + 11(i-1) \right)}$$

$$\boldsymbol{= 3 + \sum_{i=2}^{n} (13 + 11i - 11)}$$

$$\boldsymbol{= 3 + \sum_{i=2}^{n} 2 + \sum_{i=2}^{n} 11i = 3 + 2(n-1) + 11\frac{(n+2)(n-1)}{2} = \frac{11}{2}n^2 + \frac{15}{2}n - 10}$$

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:

$$\boldsymbol{IC(n)} = \sum_{i=2}^{n} i = \frac{(n+2)(n-1)}{2} = \frac{n(n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1$$

IC no es el número de operaciones elementales del algoritmo, sólo permite decir que el algoritmo, en el caso peor, tiene un comportamiento cuadrático (IC(n) $\in \Theta(n^2)$ como veremos en el tema siguiente)

2.5. ALGORITMO DE BÚSQUEDA SECUENCIAL.

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

líneas		nº OE	
	`int BusquedaSecuencial (int T[ ],int n,int valor)`		
	`{`		
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.
	`}`		

El tiempo de ejecución del algoritmo será:

$T_{BSecuencial}(n) = T_{Asig(1)} + T_{Bucle(2)} + T_{Si(5)}$ Para calcularlo lo haremos por partes:

$T_{Asig(1)} = 1$

$T_{Si(5)} = T_{condSi} + T_{cuerpoSi} = 1 + $ máx/mín/medio$(T_{return(6ó7)}) = 2+1=3$

$T_{Bucle(2)} = T_{condB} + T_{saltoB} + \sum_{(i=1;?)} T_{cicloBucle}$

$T_{cicloBucle} = T_{condB} + T_{cuerpoB}$ (=0sólo instrucción de incremento del bucle)$+ T_{incrementoB} + T_{saltoCicloB} = 4+2+1=7$

$T_{Bucle(2)} = T_{condB} + T_{saltoB} + \sum_{(i=1;?)} T_{cicloBucle} = 4+1+\sum_{(i=1;?)} 7 = 5+7\sum_{(i=1;?)}$

$\underline{\boldsymbol{T_{BSecuencial}(n)}} = T_{Asig(1)} + T_{Bucle(2)} + T_{Si(5)} = 1+5+7\sum_{(i=1;?)} +3 = \mathbf{9+7\sum_{(i=1;?)}}$

2.5.1. Caso mejor.

En el **caso mejor** para el algoritmo, se efectuará la línea (1) y la línea (2) que supone 4 OE Tras ellas la función acaba ejecutando las líneas (5) a (7). En consecuencia, el ciclo del bucle no se ejecuta nunca:

$\sum_{(i=1;0)} = 0$ y por tanto,

$$\mathbf{T_{BSecuencial}\ (n)=}\ 9+7\sum_{(i=1;0)} = \mathbf{9}$$

2.5.2. Caso peor.

En el caso peor sucede cuando el valor no se encuentra en el vector. Se efectúa la línea (1), el bucle se repite n veces hasta que se cumple la segunda condición, después se efectúa la condición de la línea (5) y la función acaba al ejecutarse la línea (7). Cada iteración del bucle está compuesta por las líneas 2 (4OE) y 3(2OE), junto con una ejecución adicional de la línea (2) que es la que ocasiona la salida del bucle. En consecuencia, el ciclo del bucle se ejecuta n veces:

$\sum_{(i=1;n)}=n$ y por tanto,

$T_{BSecuencial}\ (n)= 9+7\sum_{(i=1;n)} = 7n+9$

2.5.3. Caso medio.

En el caso medio el bucle se ejecutará un nº de veces entre 1 y n, y suponemos que cada una de ellas tiene la misma probabilidad de suceder. En consecuencia, el ciclo del bucle se ejecuta n/2 veces:

$\sum_{(i=1;n/2)} = n/2$ y por tanto,

$$T_{BSecuencial}\ (n)= 9+7\sum_{(i=1;n/2)} =9+7(n/2)= \mathbf{(7/2)n+9}$$

2.5.4. Conclusión:

Su caso mejor se da cuando el elemento está en la primera posición del vector O(n). El caso peor se produce cuando el elemento no está en el vector, y el caso medio ocurre cuando consideramos equiprobables cada una de las posiciones en las que puede encontrarse el elemento dentro del vector (incluyendo la posición especial 0, que indica que el elemento a buscar no se encuentra en el vector) pero ambos son de O(n)..

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		nº OE	
	`void burbuja(int T[],int n)`		
	`{`		
	`  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:

$T_{Burbuja}(n) = T_{Bi}(n)$. Para calcularlo lo haremos por partes:

$T_{cuerpoSi} = 2+4+3=9$ (si se cumple la condición sumarlo!!)
$T_{condSi} = 4$
$T_{Si} = T_{condSi} + T_{cuerpoSi} = 4+9=13$
$T_{cuerpoBj} = T_{Si} = T_{condSi} + T_{cuerpoSi} = 4 + 9=13$
$T_{cicloBj} = T_{condBj} + T_{cuerpoBj} + T_{incrementoBj} + T_{saltoCicloBj} = 2 + T_{cuerpoBj} + 2 + 1 = 5 + T_{cuerpoBj}$

$T_{Bj} = T_{iniBj} + T_{condBj} + T_{saltoBj} + \sum_{(j=n;i)} T_{cicloBj} = 1 + 1 + 1 + \sum_{(j=n;i+1)} T_{cuerpoBj} = 3 + \sum_{(j=n;i+1)} T_{cuerpoBj}$

$T_{cuerpoBi} = T_{Bj} = 3 + \sum_{(j=n;i+1)} T_{cuerpoBj}$

$T_{cicloBi} = T_{condi} + T_{cuerpoBi} + T_{incrementoBi} + T_{saltoCicloBj} = 2+3+\sum_{(j=n;i+1)} T_{cuerpoBj} + 2 + 1 =$
$= 8 + \sum_{(j=n;i+1)} T_{cuerpoBj} = 8 + \sum_{(j=n;i+1)} (T_{condSi} + T_{cuerpoSi}) = 8 + (n-i)(4 + T_{cuerpoSi})$

$T_{Bi} = T_{iniBi} + T_{condBi} + T_{saltoBi} + \sum_{(i=1;n-1)} T_{cicloBi} =$
$= 1+2+1+\sum_{(i=1;n-1)} (8+(n-i)(4+ T_{cuerpoSi})) = 3+8(n-1)+ \sum_{(i=1;n-1)} (n-i)(4+ T_{cuerpoSi})$
$= 3+8n-8+ n(n-1)(4+ T_{cuerpoSi}) - (4+ T_{cuerpoSi}) \sum_{(i=1;n-1)} i =$
$= 8n-5+ n(n-1)(4+ T_{cuerpoSi}) - (4+ T_{cuerpoSi})(n(n-1)/2) =$
$= 8n-5+ (4+ T_{cuerpoSi}) n(n-1)/2 = 8n-5+ 2n^2-2n+ T_{cuerpoSi}\, n(n-1)/2 =$
$= 2n^2+6n-5+ n(n-1)/2 T_{cuerpoSi}$

$$\mathbf{T_{Burbuja}(n) = 2n^2+6n-5+ n(n-1)/2 T_{cuerpoSi}}$$

2.6.1. Caso mejor.

En el caso mejor para el algoritmo la condición de la línea (3) será siempre falsa, y no se ejecutarán nunca las líneas (4), (5) y (6).

$T_{cuerpoSi} = 0$

$$\mathbf{T_{Burbuja}(n) = } 2n^2+6n-5+ n(n-1)/2 T_{cuerpoSi} = \mathbf{2n^2+6n-5}$$

2.6.2. Caso peor

En el caso peor, la condición de la línea (3) será siempre verdadera, y las líneas (4), (5) y (6) se ejecutarán en todas las iteraciones.

$T_{cuerpoSi}=9$

$\mathbf{T_{Burbuja}\ (n)=}\ 2n^2+6n-5+n(n-1)/2T_{cuerpoSi}=2n^2+6n-5+\ 9n(n-1)/2=\ \mathbf{13/2\ n^2+3/2n-5}$

2.6.3. Caso medio.

En el caso medio, la condición de la línea (3) será verdadera con probabilidad 1/2. Así, las líneas (4), (5) y (6) se ejecutarán en la mitad de las iteraciones del bucle más interno, y por tanto realiza un total de OE:

$T_{cuerpoSi}=9(1/2)$

$\mathbf{T_{Burbuja}\ (n)=}\ 2n^2+6n-5+n(n-1)/2T_{cuerpoSi}=2n^2+6n-5+\ (9/2)n(n-1)/2=\ \mathbf{(17/4)\ n^2+(15/4)n-5}$

2.6.4. Conclusión:

Sus casos mejor, peor y medio se producen respectivamente cuando el vector está inicialmente ordenado de forma creciente, decreciente y aleatoria pero Como los tiempos de ejecución en los tres casos son polinomios de grado 2, la complejidad del algoritmo es cuadrática $O(n^2)$, independientemente de qué caso se trate.