**Tema 12: Esquemas típicos de operación con formaciones.**

En este tema veremos:

* Esquemas típicos de operación con vectores (recorrido, búsqueda, inserción y ordenación).
* Centinelas y matrices orladas.

1. **Esquema de recorrido.**

# RECORRIDO DE VECTORES DE UNA DIMENSIÓN

El esquema de recorrido consiste en realizar cierta operación con todos o parte de los elementos de una formación(vector). Este recorrido se puede realizar con vectores de cualquier dimensión. La forma más general del esquema de recorrido sería:

Iniciar operación;

**while**(queden elementos sin tratar) {

Elegir uno de ellos y tratarlo;

}

Completar operación;

Otra manera de recorrer una formación:

Iniciar operación;

**for**(**int** i = 0; i < N; i++){

Tratar v[i];

}

Completar operación;

## RECORRIDO DE VECTORES DE DOS O MAS DIMENSIONES

Para realizar el recorrido de un vector de “x” dimensiones, haría falta “x” cantidad de **for** anidados unos dentro de otros. Por ejemplo:

Recorrido de matriz de dos dimensiones:

**for**(**int** i = 0; i < LongitudPrimeraDimension; i++){

**for** (**int** j = 0; j < LongitudSegundaDimension; j++){

Matriz[i][j] = 1;

}

}

Recorrido de matriz de tres dimensiones:

**for**(**int** i = 0; i < LongitudPrimeraDimension; i++){

**for** (**int** j = 0; j < LongitudSegundaDimension; j++){

**for**(**int** k = 0; i < LongitudPrimeraDimension; k++){

Matriz[i][j][k] = 1;

}

}

}

## RECORRIDO NO LINEAL

Mientras que en el recorrido lineal el índice usado para controlar el bucle nos delimita claramente el elemento a procesar en cada iteración. En el recorrido no lineal, la posición del vector a tratar se elige mediante la realización de ciertos cálculos, por tanto, esta posición no es tan predecible. Un ejemplo puede ser la construcción de un cuadrado mágico en el que la suma de los números de cada fila, columna y diagonal principal es siempre la misma.

Para cuadrados de lado impar se puede construir rellenando las casillas

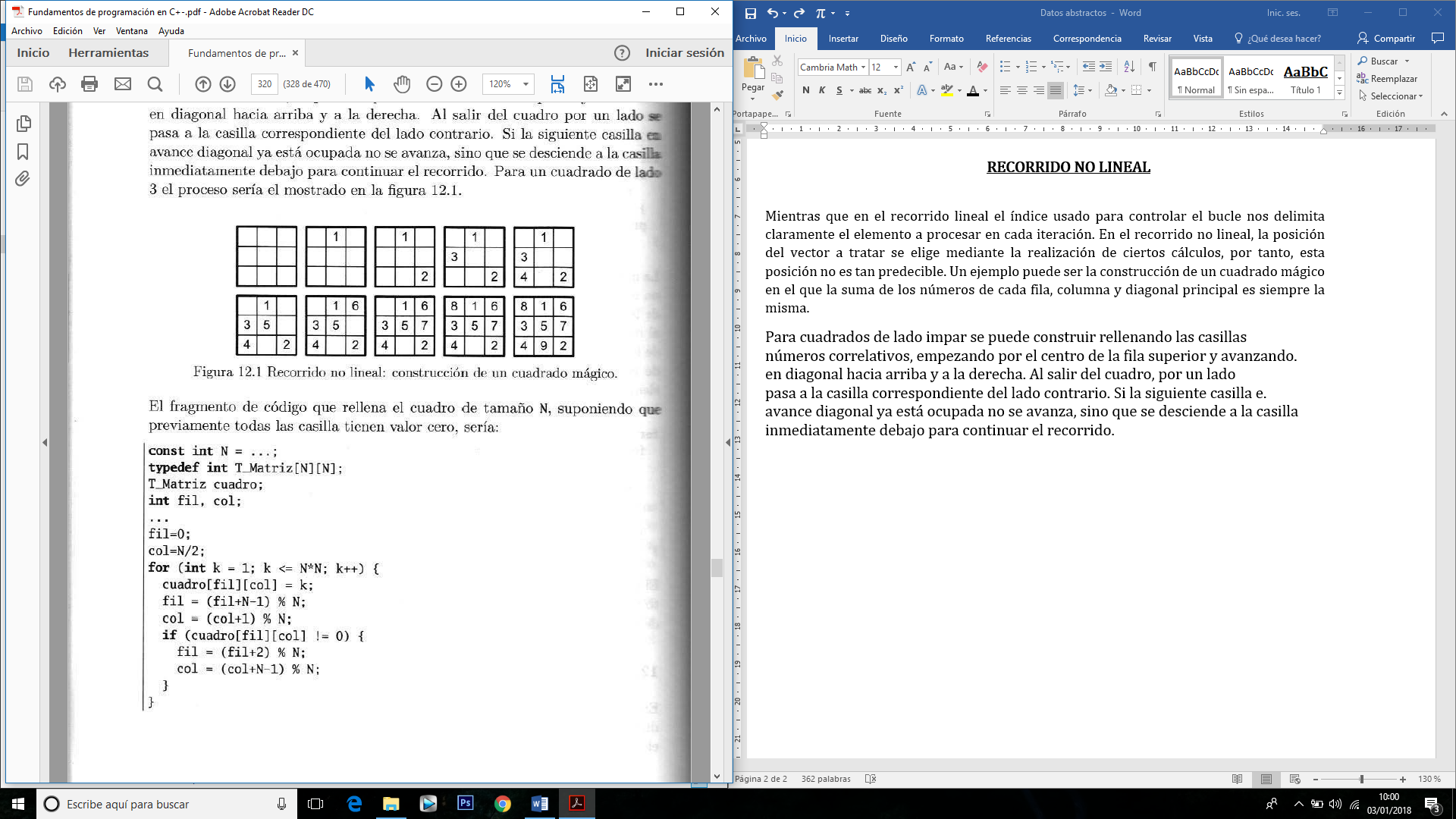
números correlativos, empezando por el centro de la fila superior y avanzando.

en diagonal hacia arriba y a la derecha. Al salir del cuadro, por un lado

pasa a la casilla correspondiente del lado contrario. Si la siguiente casilla e.

avance diagonal ya está ocupada no se avanza, sino que se desciende a la casilla

inmediatamente debajo para continuar el recorrido.



1. **Búsqueda secuencial.**

En las operaciones de búsqueda secuencial se examinan uno a uno los elementos de la colección para tratar de localizar los que cumplen una cierta condición. Si queremos encontrar todos los valores posibles que cumplen la condición dentro del vector, entonces habrá que realizar un recorrido como los mostrados en la sección anterior. Es decir, podríamos hacer un recorrido con un bucle for y dentro de ese bucle poner las condiciones necesarias para cada posición que se analice). Por ejemplo:

SI NECESITAMOS LOCALIZAR TODOS LOS ELEMENTOS QUE CUMPLAN UNA CARÁCTERISTICA

**int** CantidadPares = 0;

**typedef** **int** vector[N];

vector Myvector;

**for**(**int** i = 0; i < N; i++){

if(Myvector[i]%2== 0){

cantidadPares++;

}

}

SI QUEREMOS PARAR DE BUSCAR CUANDO ENCONTRAMOS UN ELEMENTO CUANDO CUMPLA DETERMINADAS CARACTERÍSTICAS.

Este programa devuelve un número negativo si no encuentra ningún elemento que coincida con el criterio de búsqueda.

**int** Indice (T\_elemento buscado, **const** T\_Elemento v[], **int** N){

**int** pos = 0; 🡪 Partimos de la posición cero.

**while**(pos < N && v[pos] != buscado){

pos++;

}

**if** (pos >= N) {

Pos = -1;

}

**return** pos;

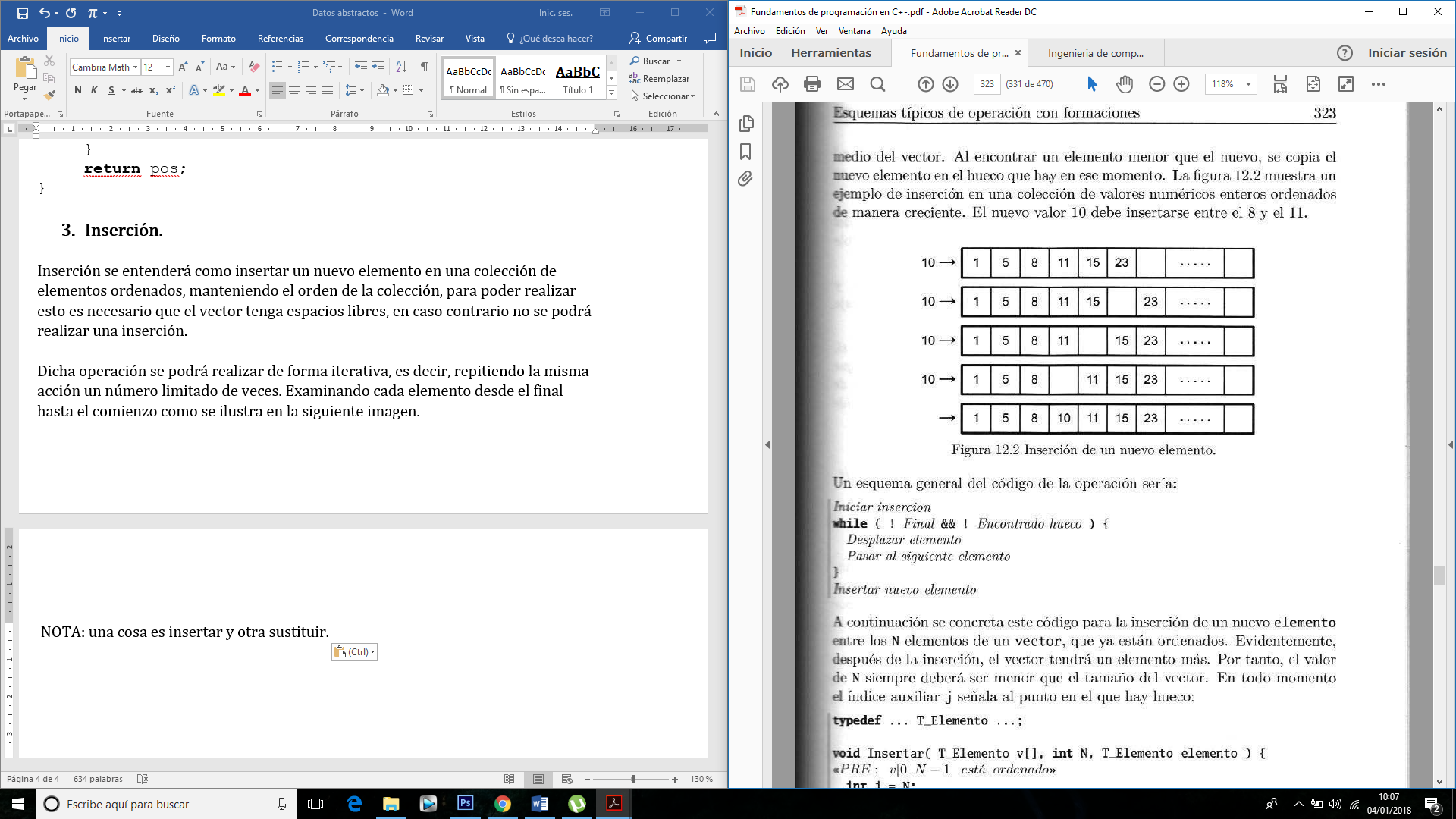
}

1. **Inserción.**

Inserción se entenderá como insertar un nuevo elemento en una colección de elementos ordenados, manteniendo el orden de la colección, para poder realizar esto es necesario que el vector tenga espacios libres, en caso contrario no se podrá realizar una inserción.

NOTA: Una cosa es insertar y otra sustituir.

Dicha operación se podrá realizar de forma iterativa, es decir, repitiendo la misma acción un número limitado de veces. Examinando cada elemento desde el final hasta el comienzo como se ilustra en la siguiente imagen.



El esquema general de código sería:

Iniciar inserción;

**while**(! Final && ! Encontrado hueco) {

Desplazar elemento;

Pasar al siguiente elemento;

}

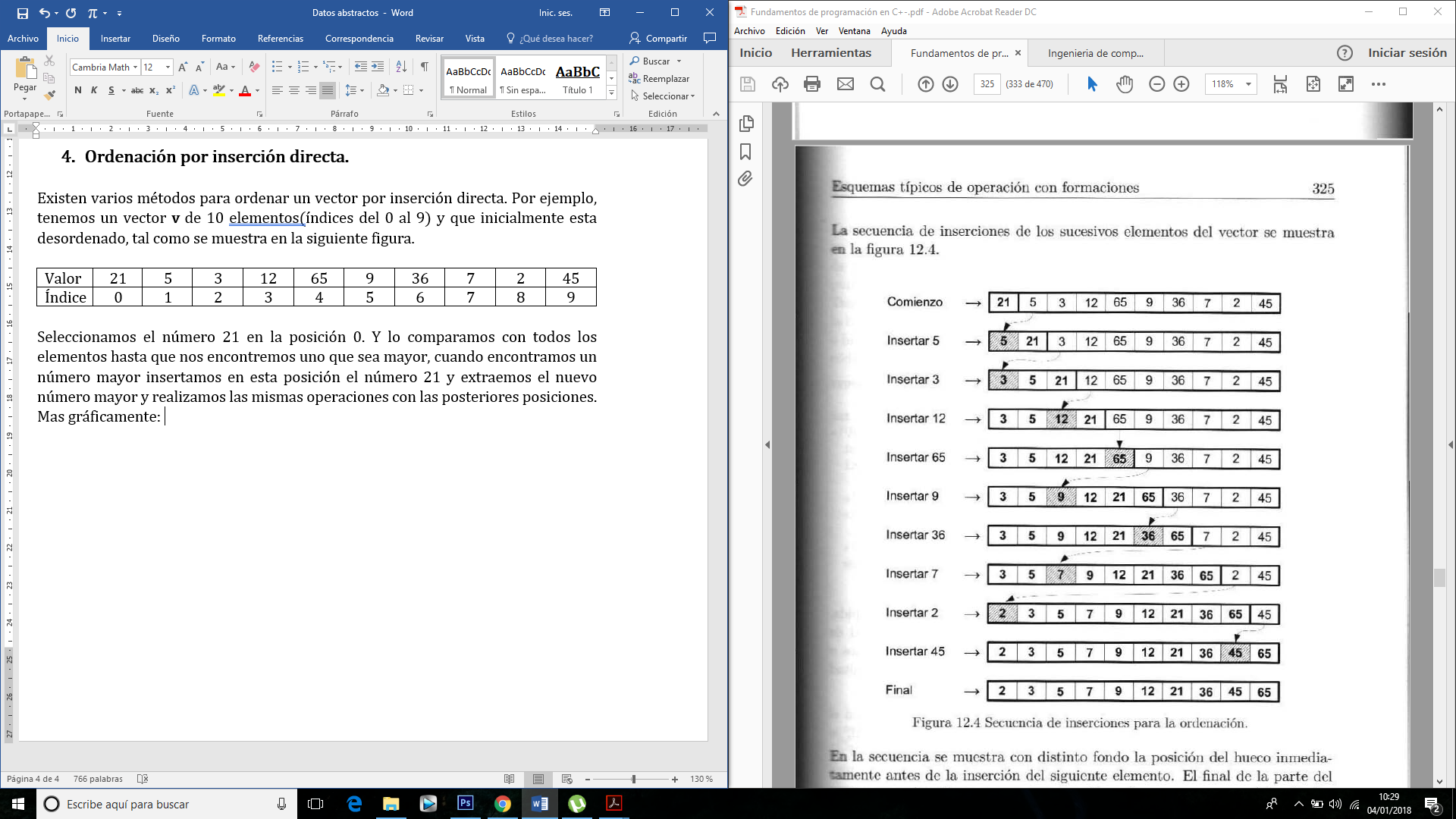
Insertar nuevo elemento;

1. **Ordenación por inserción directa.**

Existen varios métodos para ordenar un vector por inserción directa. Por ejemplo, tenemos un vector **v** de 10 elementos (índices del 0 al 9) y que inicialmente esta desordenado, tal como se muestra en la siguiente figura.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Valor | 21 | 5 | 3 | 12 | 65 | 9 | 36 | 7 | 2 | 45 |
| Índice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Seleccionamos el número 21 en la posición 0. Y lo comparamos con todos los elementos hasta que nos encontremos uno que sea mayor, cuando encontramos un número mayor insertamos en esta posición el número 21 y extraemos el nuevo número mayor y realizamos las mismas operaciones con las posteriores posiciones. Mas gráficamente:



El codigo para realizar lo que ilustra la anterior figura es el siguiente:

**void** ordenar( T\_elemento v[], int N) {

T\_Elemento valor;

**int** j;

**for** (**int** i = 1; i < N; i++) {

valor = v[i];

j = i;

**while**( j > 0 && valor < v[j-1]) {

v[j]= v [j-1];

j--;

}

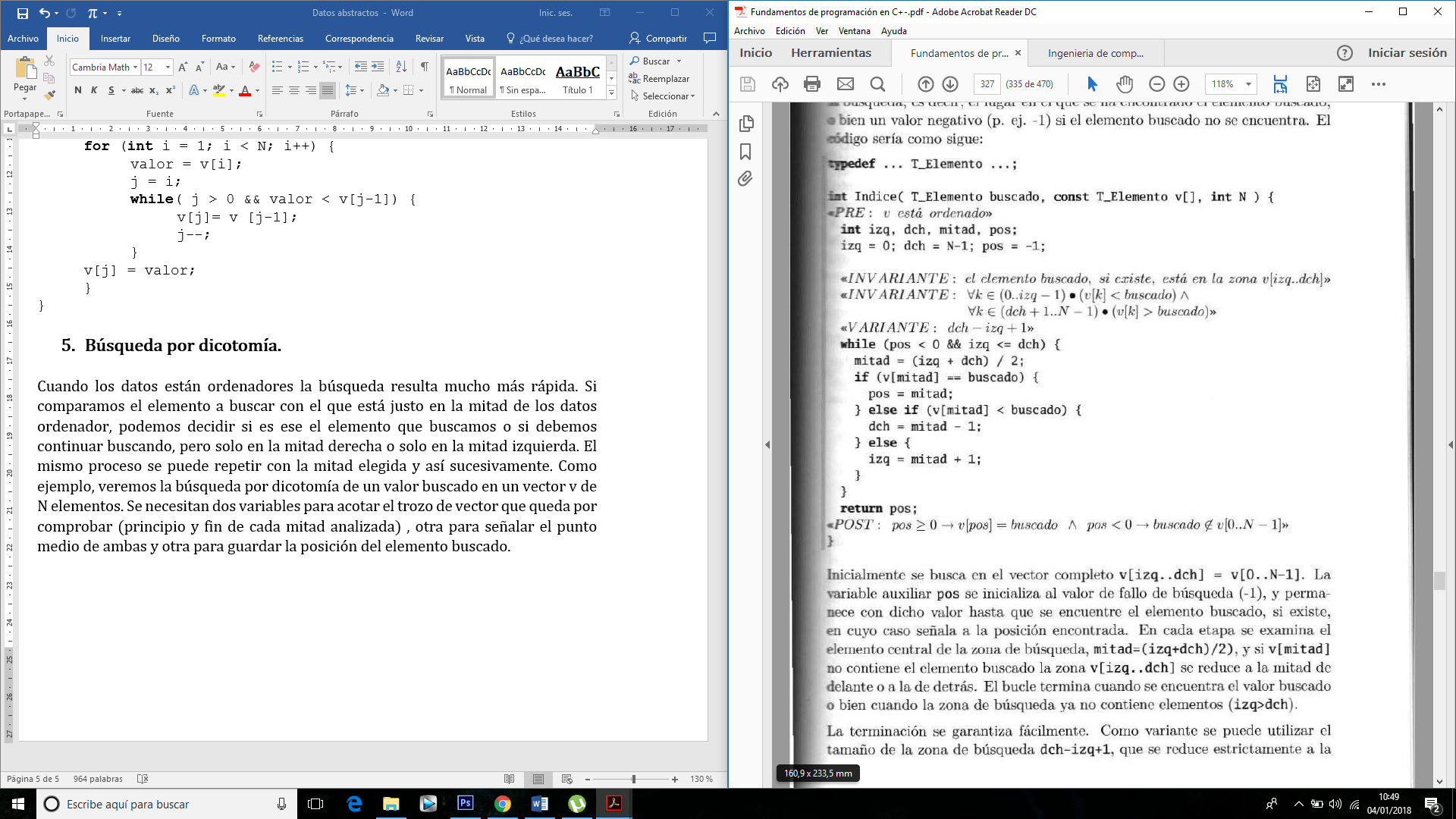
v[j] = valor;

}

}

1. **Búsqueda por dicotomía.**

Cuando los datos están ordenadores la búsqueda resulta mucho más rápida. Si comparamos el elemento a buscar con el que está justo en la mitad de los datos ordenador, podemos decidir si es ese el elemento que buscamos o si debemos continuar buscando, pero solo en la mitad derecha o solo en la mitad izquierda. El mismo proceso se puede repetir con la mitad elegida y así sucesivamente. Como ejemplo, veremos la búsqueda por dicotomía de un valor buscado en un vector v de N elementos. Se necesitan dos variables para acotar el trozo de vector que queda por comprobar (principio y fin de cada mitad analizada) , otra para señalar el punto medio de ambas y otra para guardar la posición del elemento buscado.

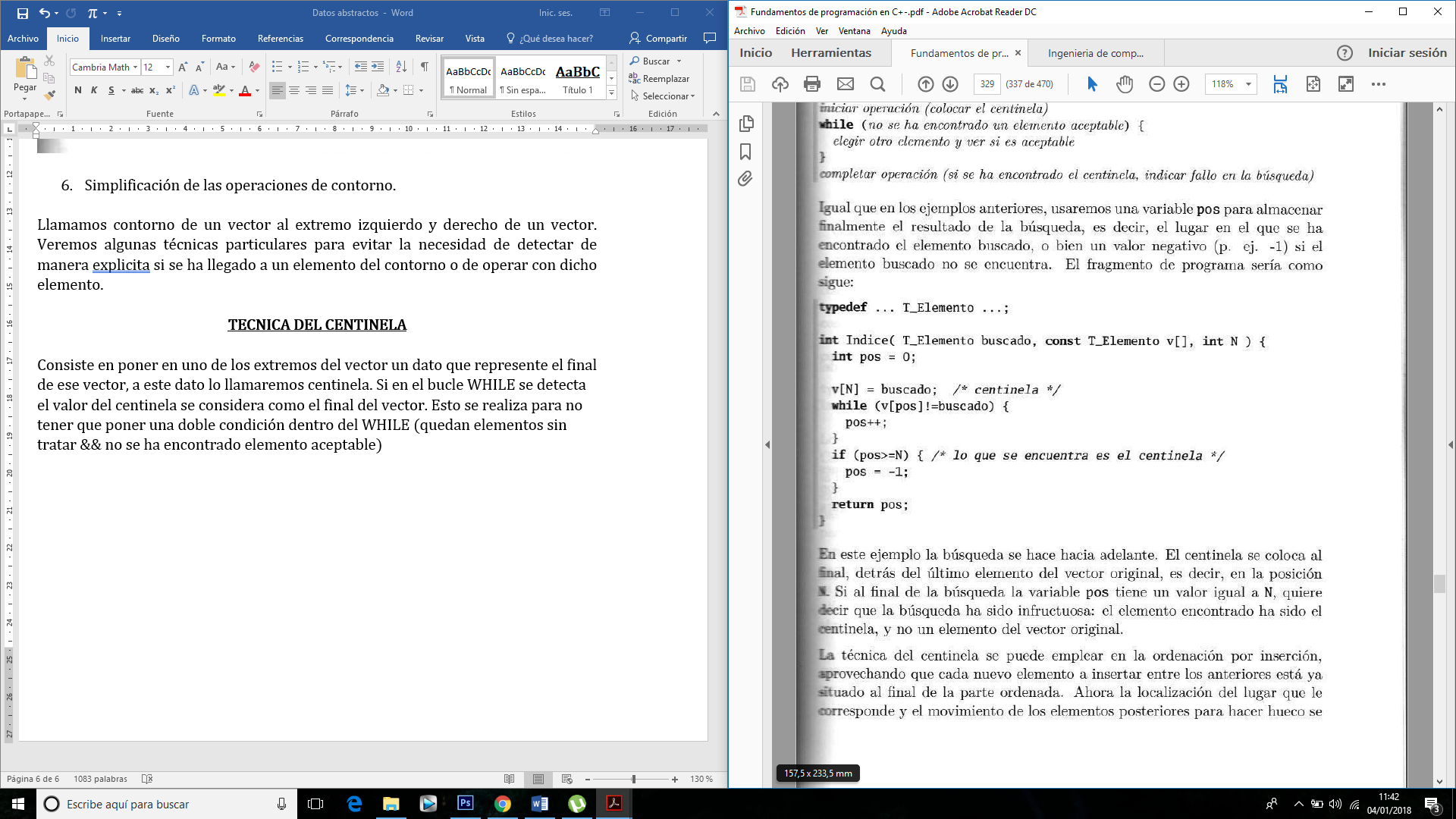


1. Simplificación de las operaciones de contorno.

Llamamos contorno de un vector al extremo izquierdo y derecho de un vector. Veremos algunas técnicas particulares para evitar la necesidad de detectar de manera explicita si se ha llegado a un elemento del contorno o de operar con dicho elemento.

### TECNICA DEL CENTINELA

Consiste en poner en uno de los extremos del vector un dato que represente el final de ese vector, a este dato lo llamaremos centinela. Si en el bucle WHILE se detecta el valor del centinela se considera como el final del vector. Esto se realiza para no tener que poner una doble condición dentro del WHILE (quedan elementos sin tratar && no se ha encontrado elemento aceptable)



### MATRIZ ORLADA

Consiste en usar la técnica del centinela en los extremos de la matriz, se añaden dos filas y dos columnas más a la matriz, y cada extremo de la matriz se le asigna el valor deseado para el centinela.

