Algoritmos y Estructura de datos

Clase 01
Estructura de datos y búsqueda

Profesor: Carlos Diaz

Contenido

- Introducción
- Definición de estructura de datos
- Operaciones en una estructura de datos
- Clasificación de estructura de datos
- Ejercicio
- Búsqueda lineal o secuencial
- Búsqueda Binaria
- Ejercicio

Introducción

- Tipo de dato: Esta definido por el conjunto de valores que representa y por el conjunto de operaciones que se pueden realizar con dicho tipo de dato.

Ejemplo:

En C++ el tipo de dato short int consta de 2 bytes y representa:

* Un entero sin signo entre 0 y 2^{16} o sea entre 0 y 65,536
* O un entero con signo entre -2^{15} y $2^{15}-1$ o sea entre −32,768 y 32,767

Y las operaciones que se pueden realizar con ellos son:

Suma, resta, multiplicación, división, etc.

Introducción

- Los tipos de datos se dividen en datos simples y datos estructurados.
- Datos simples: Son aquellos que al ser representados por la computadora ocupan solo “una casilla” de memoria, por ejemplo, un entero ocupa 4 bytes de memoria, es decir ocupa “una casilla” de 4 bytes.

Ejemplos de datos simples: enteros, reales, caracteres y booleanos.

- Datos estructurados: Son aquellos que están compuestos por otros tipos de datos, simples o estructurados. En este caso hace referencia a un “grupo de casillas” de memoria, por ejemplo un array de 5 enteros ocupa 20 bytes formado por 5 “casillas” de 4 bytes cada una.

Ejemplo de datos estructurados: arreglos (arrays), cadena de caracteres, etc.

Definición de estructura de datos

- Una estructura de datos es una colección de datos relacionados con funciones u operaciones que se pueden aplicar a dichos datos.

Ejemplos:

* Un vector es una colección de elementos en un orden específico, por lo general todos del mismo tipo. Se accede a los elementos utilizando un entero como índice.
* Una clase es una plantilla para la creación de una colección de objetos según un modelo predefinido. Las clases se utilizan como representación abstracta de conceptos, incluyen campos y operaciones que pueden consultar el valor de los campos o cambiar sus valores.

Operaciones en una estructura de datos

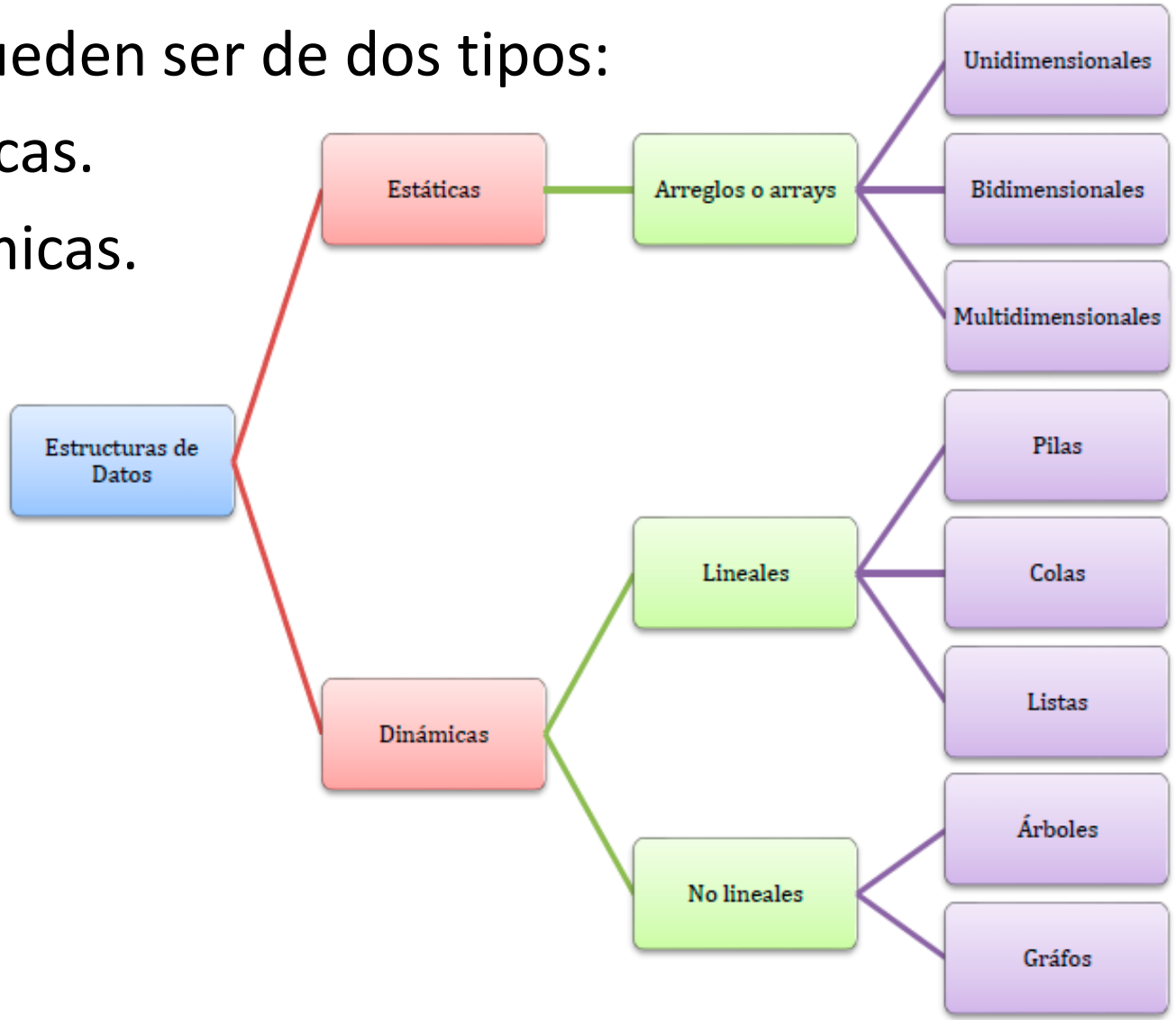
- Entre las más importantes tenemos:

* Inserción. Permite que se incluya un nuevo elemento a la estructura.
* Modificación. Permite cambiar parcial o totalmente el contenido de los elementos de la estructura.
* Eliminación. Permite suprimir elementos de la estructura.
* Búsqueda. Permite determinar si un elemento se encuentra o no en la estructura.
* Consulta. Permite obtener información de uno o más elementos de la estructura.
* Copia: Permite duplicar total o parcialmente una estructura.

Clasificación de estructura de datos

- Las estructuras de datos pueden ser de dos tipos:

* Estructuras de Datos Estáticas.
* Estructuras de Datos Dinámicas.



Clasificación de estructura de datos

- Estructuras de Datos Estáticas

* Tienen un número fijo de elementos que queda determinado desde la declaración de la estructura en el comienzo del programa.

* Las estructuras de datos estáticas, presentan dos inconvenientes:

1. La reorganización de sus elementos, si ésta implica mucho movimiento puede ser muy costosa. Ejemplo: insertar un dato en un arreglo ordenado.

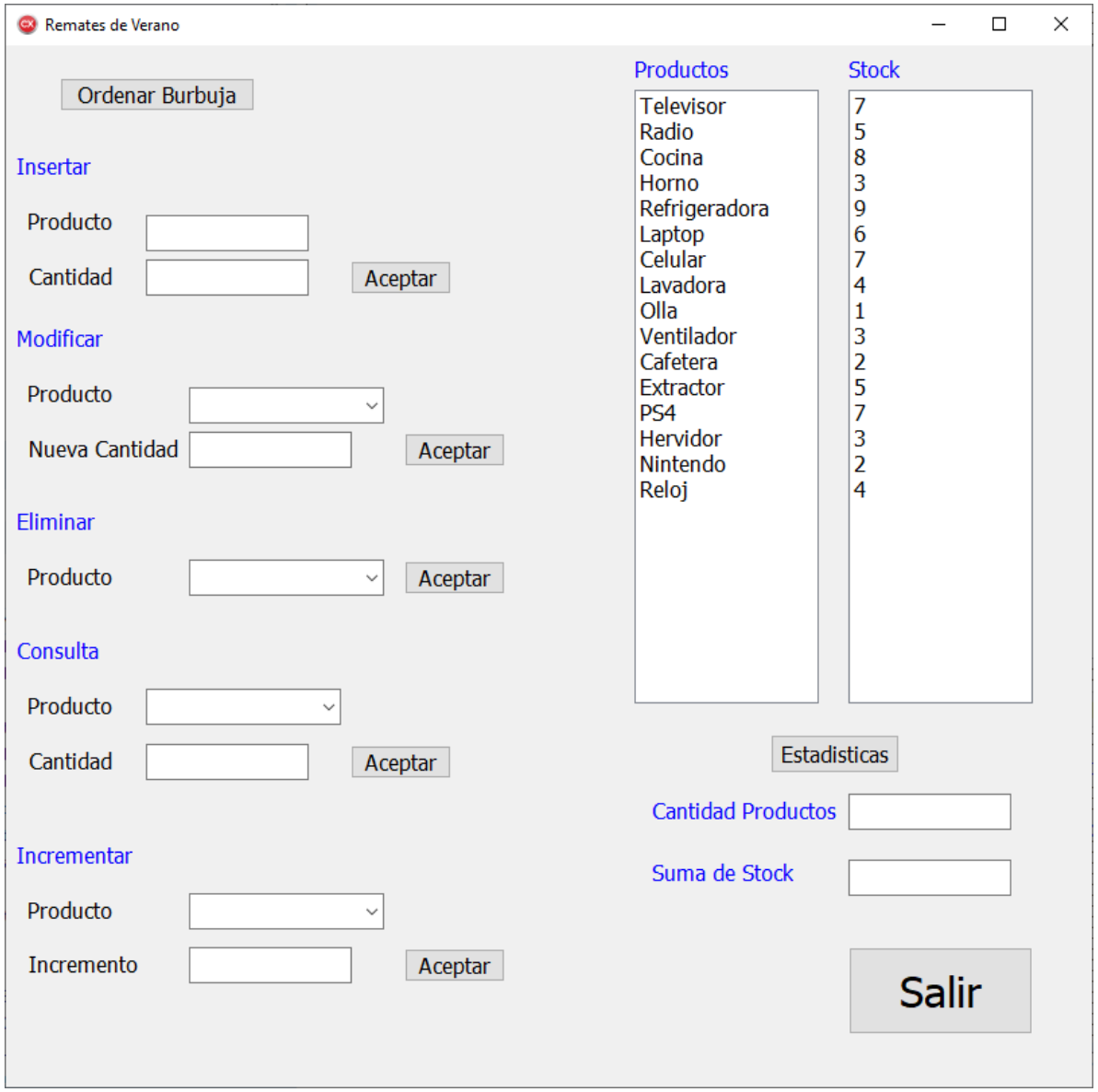
2. El tamaño ocupado en memoria es fijo, el arreglo podría llenarse y si se crea un arreglo de tamaño grande se estaría desperdiciando memoria.

Clasificación de estructura de datos

- Estructuras de Datos Dinámicas
* Al contrario de un arreglo, que tiene un número fijo de elementos, una estructura dinámica de datos se amplía y contrae durante la ejecución del programa.
* Este tipo de estructuras se pueden dividir en dos grupos según se ordenan sus elementos: Lineales y No Lineales.
- Estructuras de Datos Lineales
* En este tipo de estructuras los elementos se encuentran ubicados secuencialmente. Tenemos:
  1. Listas: podemos acceder (insertar y eliminar) por cualquier lado.
  2. Pilas: sólo tienen un único punto de acceso fijo a través del cual se añaden, se eliminan o se consultan elementos.
  3. Colas: tienen dos puntos de acceso, uno para añadir y el otro para consultar o eliminar elementos.
- Estructuras de Datos No Lineales
* Dentro de las estructuras de datos no lineales tenemos los árboles y grafos. En este tipo de estructuras los datos no se encuentran ubicados secuencialmente.

Ejercicio

- En el sistema de almacén mostrado, aplique las siguientes operaciones de estructura de datos: Insertar, modificar, eliminar, consultar, etc.
- Para este ejercicio utilice vectores de tamaño fijo de 25.



Tarea

- Modifique el ejercicio para que ahora utilice vectores de tamaño dinámico.

Búsqueda lineal o secuencial

- En una búsqueda secuencial, los elementos de una lista se exploran en secuencia, uno después de otro.
- El elemento que se busca se llama clave de búsqueda.
- El algoritmo de búsqueda secuencial compara cada elemento del array con la clave de búsqueda.
- Dado que el array no está en un orden prefijado, es probable que el elemento a buscar pueda ser el primer elemento, el último elemento o cualquier otro.
- De promedio, al menos el programa tendrá que comparar la clave de búsqueda con la mitad de los elementos del array.
- El método de búsqueda lineal funcionará bien con arrays pequeños o no ordenados.
- Ejemplo: Buscar 225 en el vector [13, 44, 75, 100, 120, 275, 325, 510]

Búsqueda lineal o secuencial

Clave

225

13	44	75	100	120	275	325	510

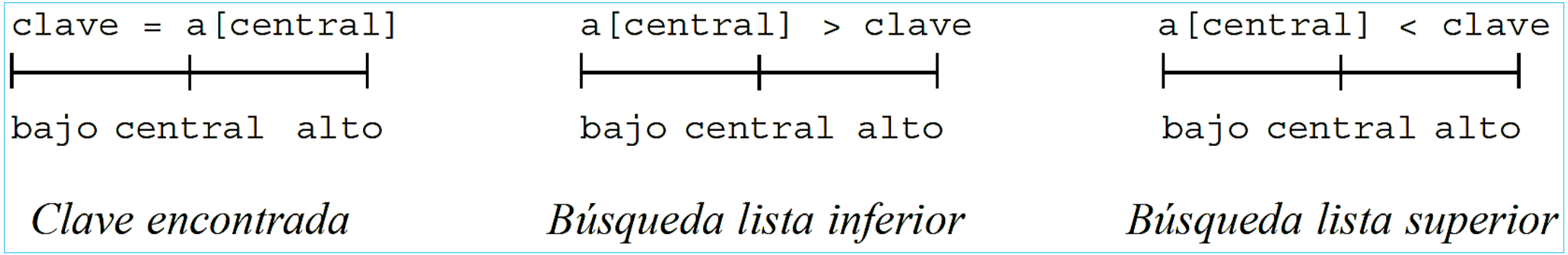
Búsqueda binaria

- La búsqueda secuencial se aplica a cualquier lista.
- Si la lista está ordenada, la búsqueda binaria proporciona una técnica de búsqueda mejorada.
- Localizar una palabra en un diccionario es un ejemplo típico de búsqueda binaria. Dada la palabra, se abre el libro cerca del principio, del centro o del final dependiendo de la primera letra de la palabra que busca. Se puede tener suerte y acertar con la página correcta; pero, normalmente, no será así y se mueve el lector a la página anterior o posterior del libro.
- Una idea similar se aplica en la búsqueda en una lista ordenada. Se sitúa el índice de búsqueda en el centro de la lista y se comprueba si nuestra clave coincide con el valor del elemento central.
- Si no se encuentra el valor de la clave, se sitúa en la mitad inferior o superior del elemento central de la lista.
- Ejemplo: Buscar 225 en el vector [13, 44, 75, 100, 120, 275, 325, 510]

Algoritmo de búsqueda binaria

- Suponiendo que la lista está en un array delimitado por índices bajo y alto, los pasos a seguir:
- 1. Calcular el índice del punto central del array:

  central = (bajo + alto)/2 (división entera)
- 2. Comparar el valor de este elemento central con la clave:



- Si a[central] < clave, la nueva sublista de búsqueda queda delimitada por:

  bajo = central+1 ... alto
- Si a[central] > clave, la nueva sublista de búsqueda queda delimitada por:

  bajo ... alto=central-1
- El algoritmo termina bien porque se ha encontrado la clave o porque el valor de bajo excede al de alto, lo que significa que la clave no se encuentra en la lista.

Búsqueda binaria

Clave

225

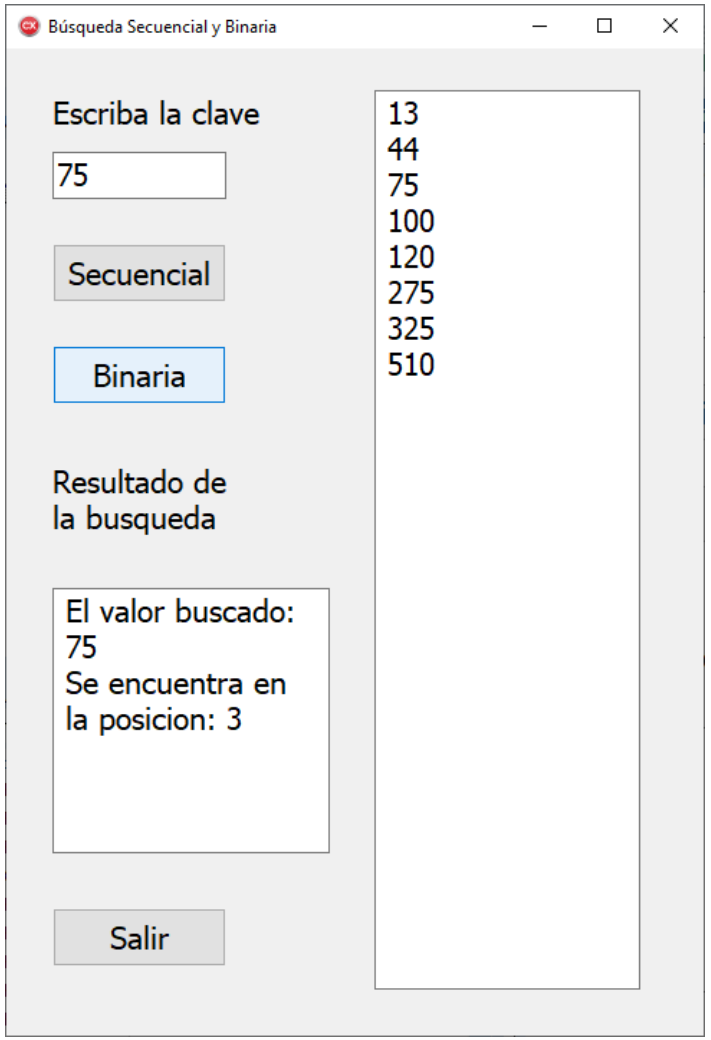
13	44	75	100	120	275	325	510

Codificación búsqueda binaria

```
int busquedaBin(int a[], int n, int clave)
{
	int central, bajo, alto;
	int valorCentral;
	bajo = 0;
	alto = n - 1;
	while (bajo <= alto)
	{
		central = (bajo + alto)/2; // índice de elemento central
		valorCentral = a[central]; // valor del índice central
		if (clave == valorCentral)
				return central; // encontrado, devuelve posición
		else if (clave < valorCentral)
				alto = central - 1; // ir a sublista inferior
		else
				bajo = central + 1; // ir a sublista superior
	}
return -1; //elemento no encontrado
}
```

Ejercicio

- Diseñe un formulario que muestre las búsquedas secuencial y binaria.



FIN