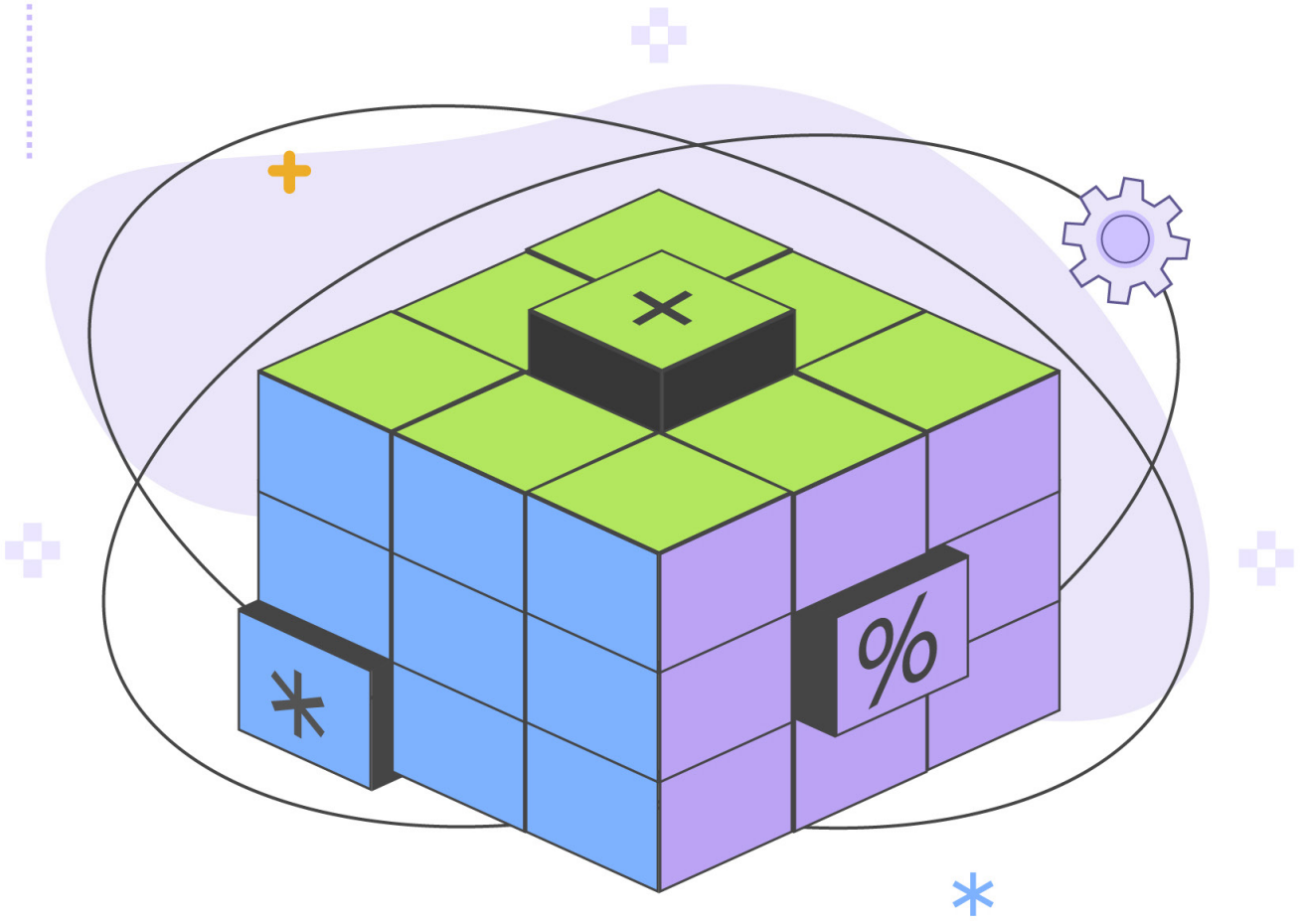
Capítulo 5

Estructuras de datos compuestas



Capítulo 5

Estructuras de datos compuestas

Objetivo

En este capítulo, exploramos el mundo de las estructuras de datos estáticas, como arreglos y matrices. Estas herramientas son fundamentales para almacenar y organizar datos de forma eficiente en nuestros programas. Aprenderemos a usarlas para resolver diversos problemas, desde el almacenamiento de información simple hasta la creación de estructuras complejas. Además, abordaremos las operaciones básicas como agregar, modificar, buscar y eliminar elementos.

Nos centraremos en estructuras de datos estáticas, que tienen un tamaño predefinido y que no puede cambiar durante la ejecución del programa.

Al finalizar este capítulo, estarás capacitado para utilizar las estructuras de datos estáticas y compuestas para modelar situaciones y resolver de manera eficiente muchos de los problemas que suelen surgir. Estarás en condiciones de seleccionar la estructura más adecuada para una gran diversidad de casos de uso.

Datos compuestos

Las estructuras de datos compuestas son estructuras que se construyen a partir de otras estructuras, por ejemplo de datos simples como los números, cadenas de texto, valores lógicos o caracteres.

Se utilizan para representar información más compleja, que se organiza y almacena en memoria de manera que se pueda acceder y manipular fácilmente.

En particular avanzaremos en los datos compuestos caracterizados por ser homogéneos e indexados.

Arreglos

El tipo de dato arreglo, es una colección ordenada e indexada de elementos, con las siguientes características:

- Todos los elementos son del mismo tipo. Esto le da la característica de ser un tipo de dato homogéneo.
- Los elementos pueden recuperarse en cualquier orden, se indica la posición que ocupan dentro de la estructura. Esto le da la característica de ser una estructura indexada.
- La memoria ocupada a lo largo de la ejecución del programa es fija. Esto le da la característica de ser una estructura estática.

Es importante destacar que se accede a cualquiera de los elementos de la colección a través del nombre del identificador de la variable y la posición que ocupa.

Los arreglos pueden ser de distintas dimensiones. La cantidad de dimensiones indica la cantidad de índices necesarios para acceder a un elemento.

En nuestra disciplina, el término "arreglo" es la traducción frecuentemente utilizada para referirse a un "array". Probablemente surgió como un intento de encontrar una palabra en español que capturara la idea de una colección ordenada de elementos, aunque "formación" podría ser una traducción más precisa desde un punto de vista etimológico (relacionado con "formar" una estructura).

[-] vectores

La forma más simple de un arreglo se denomina vector o arreglo unidimensional. Como su nombre lo indica, es una estructura que tendrá una única dimensión.

El siguiente gráfico muestra un vector de 6 elementos de números enteros.

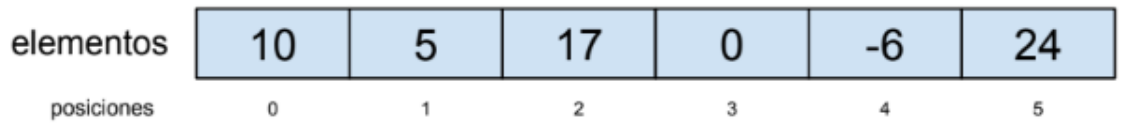


Fig.18 - vector.

Para utilizar este tipo de dato utilizaremos la sintaxis de la siguiente manera:

```
tipo [] vector = new tipo[capacidad];
```

El tipo de los elementos de un arreglo puede ser cualquiera de los que hemos visto hasta el momento, por ejemplo podríamos querer un vector de elementos enteros, para ello usaremos el tipo int. Para especificar la cantidad o "capacidad" de elementos usaremos una expresión del tipo ordinal, esto es, un tipo de dato en el cual sus valores se pueden contar y se les puede asignar un orden específico.

```
int [] vector = new int[10];
```

Tal como mencionamos con anterioridad, para acceder a los elementos de un vector debemos hacer referencia a su posición. Cada elemento del vector ocupa un lugar en la memoria de forma consecutiva, por lo tanto accederemos al lugar en la memoria a través del nombre de su variable y luego el desplazamiento indicado a través de su posición.

Por ejemplo si quisiéramos acceder al primer elemento del vector para guardar el valor en una variable, siempre del mismo tipo, lo podríamos hacer de la siguiente manera:

```
int variable = vector[0];
```

Se puede modificar el valor de cualquier elemento del vector respetando el tipo de dato asociado.

El siguiente ejemplo muestra la asignación de valores a cada unos de los elementos del vector:

```
//
int[] elementos = new int[6];

//
elementos[0] = 10;
elementos[1] =  5;
elementos[2] = 17;
elementos[3] =  0;
elementos[4] = -6;
elementos[5] = 24;
```

Hay que tener presente el tipo de dato asociado al vector, dado que el mismo será el que