

6.A. Introducción a las estructuras de almacenamiento.

Sitio: [Formación Profesional a Distancia](#)

Curso: Programación

Libro: 6.A. Introducción a las estructuras de almacenamiento.

Imprimido por: Iván Jiménez Utiel

Día: martes, 7 de enero de 2020, 22:31

Tabla de contenidos

[1. Introducción.](#)

1. Introducción.

Cuando el volumen de datos a manejar por una aplicación es elevado, no basta con utilizar variables. Manejar los datos de un único pedido en una aplicación puede ser relativamente sencillo, pues un pedido está compuesto por una serie de datos y eso simplemente se traduce en varias variables. Pero, ¿qué ocurre cuando en una aplicación tenemos que gestionar varios pedidos a la vez?

Lo mismo ocurre en otros casos. Para poder realizar ciertas aplicaciones se necesita poder manejar datos que van más allá de meros **datos simples** (números y letras). A veces, los datos que tiene que manejar la aplicación son **datos compuestos**, es decir, datos que están compuestos a su vez de varios datos más simples. Por ejemplo, un pedido está compuesto por varios datos, los datos podrían ser el cliente que hace el pedido, la dirección de entrega, la fecha requerida de entrega y los artículos del pedido.

Los datos compuestos son un tipo de estructura de datos, y en realidad ya los has manejado. **Las clases son un ejemplo de estructuras de datos que permiten almacenar datos compuestos**, y el [objeto](#) en sí, la [instancia](#) de una [clase](#), sería el dato compuesto. Pero, a veces, los datos tienen estructuras aún más complejas, y son necesarias soluciones adicionales.

Aquí podrás aprender esas soluciones adicionales. Esas soluciones consisten básicamente en la capacidad de poder manejar varios datos del mismo o diferente tipo de forma dinámica y flexible.

En otras ocasiones nos encontramos otro tipo de dificultades. Por ejemplo, ¿cómo almacenarías en memoria un listado de números del que tienes que extraer el valor máximo? Seguro que te resultaría fácil. Pero, ¿y si el listado de números no tiene un tamaño fijo, sino que puede variar en tamaño de forma dinámica? Entonces la cosa se complica.

Un listado de números que aumenta o decrece en tamaño es una de las cosas que aprenderás a utilizar aquí, utilizando estructuras de datos. Pasaremos por alto las clases y los objetos, pues ya los has visto con anterioridad, pero debes saber que las clases en sí mismas son la evolución de un tipo de estructuras de datos conocidas como datos compuestos (también llamadas registros). Las clases, además de aportar la ventaja de agrupar datos relacionados entre sí en una misma estructura (característica aportada por los datos compuestos), permiten agregar métodos que manejen dichos datos, ofreciendo una herramienta de programación sin igual. Pero todo esto ya lo sabías.

Las estructuras de almacenamiento, en general, se pueden clasificar de varias formas. Por ejemplo, atendiendo a si pueden almacenar datos de diferente tipo, o si solo pueden almacenar datos de un solo tipo, se pueden distinguir:

- Estructuras con capacidad de **almacenar varios datos del mismo tipo**: varios números, varios caracteres, etc. Ejemplos de estas estructuras son los arrays, las cadenas de caracteres, las listas y los conjuntos.
- Estructuras con capacidad de **almacenar varios datos de distinto tipo**: números, fechas, cadenas de caracteres, etc., todo junto dentro de una misma estructura. Ejemplos de este tipo de estructuras son las clases.

Otra forma de clasificar las estructuras de almacenamiento va en función de si pueden o no cambiar de tamaño de forma dinámica:

- **Estructuras cuyo tamaño se establece en el momento de la creación o definición** y su tamaño no puede variar después. Ejemplos de estas estructuras son los **arrays** y las matrices (arrays multidimensionales).
- **Estructuras cuyo tamaño es variable (conocidas como estructuras dinámicas)**. Su tamaño crece o decrece según las necesidades de forma dinámica. Es el caso de las listas, árboles, conjuntos y, como veremos también, el caso de algunos tipos de cadenas de caracteres. Serán tratadas en otra unidad de contenidos posterior.

Por último, atendiendo a la forma en la que los datos se ordenan dentro de la estructura, podemos diferenciar varios tipos de estructuras:

- **Estructuras que no se ordenan de por sí**, y debe ser el programador el encargado de ordenar los datos si fuera necesario. Un ejemplo de estas estructuras son los arrays.
- **Estructuras ordenadas**. Se trata de estructuras que al incorporar un dato nuevo a todos los datos existentes, este se almacena en una posición concreta que irá en función del orden. El orden establecido en la estructura puede variar dependiendo de las necesidades del programa: alfabético, orden numérico de mayor a menor, momento de inserción, etc. Serán tratadas en otra unidad de contenidos posterior.

Todavía no conoces mucho de las estructuras, y probablemente todo te suena raro y extraño. No te preocupes, poco a poco irás descubriéndolas. Verás que son sencillas de utilizar y muy cómodas.

Autoevaluación

El tamaño de las estructuras de almacenamiento siempre se determina en el momento de la creación.

¿Verdadero o falso?

- ☐ Verdadero.
- ☐ Falso.