TP 0 – Introducción

Alumno: Nuñez Souto, Aaron Agustín

1. Lo que la cita quiere dar a entender, es que para lograr escribir programas cada vez más eficientes, hay que tener conocimiento acerca de las diferentes estructuras de datos, para poder aplicar la que más se adapte a la resolución de un problema específico, planteado a la hora de realizar un programa.
2. Los 2 enfoques que se deben cumplir a la hora de diseñar algoritmos son:
3. *Diseñar un algoritmo que sea fácil de entender, codificar y depurar*: es un enfoque más orientado al área de la ingeniería de software, mediante el cual se busca que el algoritmo sea fácil de entender, de codificar y de depurar, para simplificar la labor de resolver un problema mediante la implementación de éste, ya sea mediante un código bien documentado, como de un código simple pero que responsa ante la problemática presentada, por mencionar unos ejemplos.
4. *Diseñar un algoritmo que haga un uso eficiente de los recursos* *de la computadora*: este enfoque hace referencia a que la implementación de un algoritmo debe utilizar los recursos de la computadora de forma tal que no los exija más allá de lo necesario para resolver un problema.
5. Una estructura de datos representa una colección de elementos, y cómo ésta se almacena en la memoria de la computadora, y/o en algún dispositivo de memoria secundaria. Esta forma de almacenamiento determina la manera en que los datos se pueden recuperar.

También se puede ver a una estructura de datos como una colección de valores, además de la relación que existe entre los mismos y las operaciones que podemos hacer sobre ellos. Una estructura de datos describe el formato en que los valores van a ser almacenados, cómo van a ser accedidos y modificados.

1. La eficiencia de un algoritmo debe medirse para determinar el uso que da a los recursos que utiliza. Dependiendo el uso de los recursos, se elige un algoritmo por sobre otro, siendo que el elegido debería ser aquel que puede llevar a cabo su funcionalidad con el menor uso posible de los recursos.
2. Hablamos de una “filosofía” en torno a las estructuras de datos, ya que para diseñar un programa que resulte ser eficiente, hay que plantearse en un inicio el analizar primero el problema, para determinar los objetivos de rendimiento que deben lograrse. Posterior a ello, se puede seleccionar aquella estructura de datos que se adapte a lo analizado previamente; aunque esta práctica no es llevada a cabo por todos los diseñadores de programas, dado que podemos encontrar programas deficientes que ignoran tal proceso de análisis y aplican estructuras de datos que, si bien tienen cierta familiaridad, no resulta ser la óptima para resolver el problema.
3. La relación entre el concepto de TDA (Tipo de Dato Abstracto) y el de estructura de datos, es que esta última se puede considerar como la implementación de un TDA. Esto se debe a que un TDA se define en términos de un tipo y un conjunto de operaciones en ese tipo. Éste no especifica cómo se implementa el tipo de datos, pero el comportamiento de cada operación está determinado por sus entradas y salidas. Por otra parte, los tipos de datos tienen una forma tanto lógica como física, siendo un TDA tal forma lógica; mientras que la implementación del tipo de datos como estructura de datos es su forma física.
4. La frase “el costo de una solución es la cantidad de recursos que consume la solución” quiere decir que a medida que se emplean más recursos para realizar una solución, el costo se incrementa, ya sea por el desgaste de los mismos, la pérdida de tiempo, entre otros aspectos.
5. Para seleccionar una estructura adecuada para resolver un problema se deben seguir los siguientes pasos:
6. Analizar el problema con el fin de determinar qué operaciones básicas deben ser apoyadas.
7. Cuantificar las limitaciones de recursos para cada operación.
8. Seleccionar la estructura de datos que mejor se adapte a estos requisitos.

Este enfoque analiza una vista centrada en los datos del proceso de diseño. La primera preocupación son los datos y las operaciones que se realizarán en ellos, la siguiente preocupación es la representación de esos datos y la última es la implementación de esa representación.

1. Cada estructura de datos tiene asociados costos y beneficios, por lo que generalmente es incorrecto decir que una estructura de datos es mejor que otra. Tal estructura requiere de una cierta cantidad de espacio para dato que almacena, una cierta cantidad de tiempo para realizar una operación básica y un cierto esfuerzo de programación; todo ello se consideraría el costo. Por otra parte, los beneficios que brinda cada estructura recalarían en el rendimiento del programa (dado que hay estructuras que se adecúan de mejor manera a ciertos problemas que otras; por ende, las que son adecuadas pueden llevar a cabo la tarea en menos tiempo) y el óptimo uso de los recursos requeridos para su ejecución.
2. A la hora de hablar de tipos de datos, encontramos que éstos se componen de una parte física y una parte lógica. En cuanto a la parte física, hallamos que está compuesta por los tipos de datos abstractos (TDA), dado que éstos están constituidos por el tipo y las operaciones del tipo de dato. Por otra parte, las estructuras de datos componen la parte lógica de un tipo de dato, ya que están compuestas por el espacio de almacenaje y las subrutinas. Esto se debe a que cuando implementamos un TDA, estamos lidiando con la forma física del tipo de dato asociado. Cuando usamos un TDA en otra parte del programa, no preocupamos de la forma lógica del tipo de dato asociado.
3. Así como utilizamos estructuras de datos a la hora de resolver problemas, también se pueden utilizar patrones de diseño, ya que incorpora y generaliza conceptos de diseño importantes para un problema concurrente. Una gran similitud entre ambos conceptos, es que éstos se adecúan para cierto tipo de problema en particular, es decir que una estructura de dato puede diferir de otra a la hora de encarar un problema, y lo mismo ocurre con los patrones de diseño, dado que resuelven una problemática en particular. Por otro lado, ambos buscan la eficiencia del programa, dado que un patrón o una estructura de dato bien elegidos permiten al programa ejecutarse de una forma más eficiente, en base a los recursos disponibles. Otra similitud es que ambos tienen costos y beneficios, tal como se ha mencionado previamente, lo cual implica que los *tradeoffs* sean posibles.
4. A la hora de desarrollar un programa nos centramos en la idea de resolver un problema, dado que es creado para llevar a cabo tal acción. Al escribir un programa, es una buena práctica considerar ciertos aspectos que guiarán el proceso de escritura. Uno de esos aspectos es el de las estructuras de datos, mediante las cuales se puede obtener un uso eficiente de los recursos del sistema, a la hora de encarar el problema planteado. Otro aspecto, relacionado con el anterior, es el de los patrones de diseño, los cuales incorporan y generalizan conceptos de diseño importantes para un problema concurrente. Por otra parte, al hablar de programas, hay que tener en cuenta el concepto de algoritmo, el cual hace referencia a un método o proceso que se emplea para resolver un problema. Un algoritmo cuenta con varias propiedades, de las cuales nos encontramos con que debe ser correcto (debe calcular la función deseada, convirtiendo cada entrada en la salida correcta), debe estar compuesto por una serie de pasos concretos (la acción descrita en cada paso es completamente entendida por la persona o máquina que debe realizar el algoritmo; a su vez, cada paso debe poder realizarse en un período de tiempo finito), no puede haber ambigüedad en cuanto a qué paso se realizará a continuación (se debe poder elegir qué paso sigue a continuación, pero el proceso de selección es inequívoco en el momento en que se realiza la elección), debe estar compuesto por un número finito de pasos y, por último, debe terminar (no puede entrar en un bucle infinito). Por ende, se puede decir que “un programa es una instancia de un algoritmo en un lenguaje de programación”.