**TIPOS DE DATOS**

Datos abstractos:

-Es una serie de objetos agrupados con una serie de operaciones.

-Son abstracciones matemáticas. Su definición no incluye ninguna forma de implementar operadores.

-Es una colección de datos

Pensamos en: Integers, Reals, Booleans

* Cada uno tiene operaciones como suma, resta etc.
* Pensando en ADTs tenemos: listas, colas, conjuntos, graphs
* Como cualquier otro tipo de dato, tiene operaciones asociadas como, por ejemplo: add, remove, contains.

¿Cómo implementarlo?

* Se puede implementar utilizando clases, porque permiten esconder los detalles de implementación.
* No es un concepto de OO.
* En el contexto OO puede ser lo mismo que una interfase o clase con algunas características.

¿Qué operaciones deben ir en el ADT?

* Es una decisión de diseño del programador.
* Hay que ser cuidadoso de no exponer los detalles de implementación.

**ARRAYS**

* Son objetos para un grado muy limitado.
* Se declara de acuerdo con el tipo de datos.
* Pueden ser considerados como instancias de una comprendida clase de array

¿Cómo declararlo?

Int [ ] a;

Int a[ ];

Int [ ] a = newInt[10];

Int [b] = {5,4,3,2}

* Se queda con el tamaño en que se declaró, es decir, no se puede modificar ni su tamaño ni sus valores.
* Se pueden acceder mediante índices.

LIMITACIONES DE LO ARRAYS:

* Cuesta mucho cambiar su tamaño, por ejemplo, se debe crear un nuevo array, copiando todo lo del array anterior hacia el nuevo con el nuevo tamaño.
* Lo datos en el array están todos juntos.

**MATRICES**

Una matriz puede ser imaginada como un array de dos dimensiones o un array de arrays. Ejemplo:

Int [][] nums = newInt[5][4]

**LINKED LIST (listas enlazadas)**

* Cada elemeto de la estructura es un nodo.
* Puede cambiarse su tamaño, ya que es una estructura dinámica.
* Un nodo contiene un campo de datos que hace referencia a otro nodo.
* El ultimo nodo apunta a “null”.
* Para incrementar el desempeño, especialmente cuando se inserta un nodo al final se puede añadir una referencia al último nodo (lista con dos finales).

Listas doblemente enlazadas:

* Su ventaja es que nos podemos devolver al nodo anterior.
* Cada nodo tiene dos referencias, la primera al nodo siguiente y la segunda al nodo anterior.
* Es un poco más compleja al tener nodos en las dos direcciones.

**LISTAS CIRCULARES**