Diseño

Algoritmos y estructuras de datos

Elías Cerdeira

¿Qué vamos a ver hoy?

- Breve introducción a diseño
- Diseñar un conjunto acotado
- Diseñar un buffer circular

¿Qué implica diseñar?

- **Elegir estructuras** concretas para almacenar la información
- Escribir los **algoritmos de las operaciones** especificadas
- Contemplar el **contexto de uso**

¿Cómo luce un diseño?

```
Modulo <Impl> implementa <TAD> {
        campo1: Type1
        campo2: Type2

        pred InvRep(i: <Impl>)
                { … }

        pred Abs(i: <Impl>, t: <TAD>)
                { … }

        proc Op1(inout i: <Impl>, in v: V)
                { … pseudocodigo … }
        proc Op1(in i: <Impl>): V
                { … pseudocodigo … }
}
```

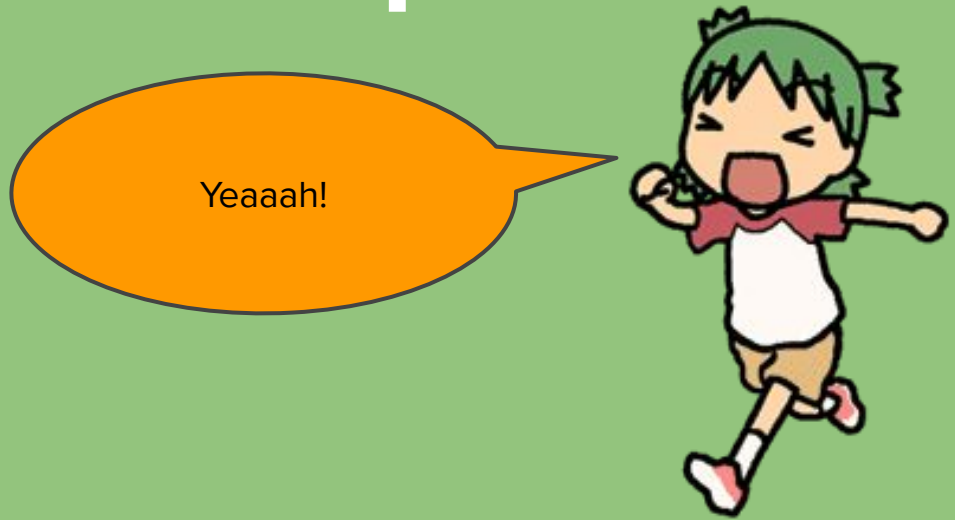
Representan el **estado** del módulo. Se manipulan con los algoritmos. Se utilizan tipos de **implementación**.

Restricciones sobre los conjuntos de valores **válidos** para las variables de estado.

Establece relación entre una **instancia** del **TAD** y una del **módulo**. Asocia variables con observadores.

Pseudocódigo que implementa la operación. Se pueden agregar **precondiciones**. Se utilizan operaciones de los TADs.

¡Vamos a ver el apunte!



¡A resolver ejercicios!

Diseñar conjunto acotado

Ejercicio 1. Quizás la forma más simple de implementar un conjunto acotado sea mediante un array de tamaño fijo, utilizando la siguiente estructura:

```
Modulo ConjAcotadoArr<T> implenenta ConjAcotado<T> {
    var datos: Array<T>
    var largo: int
}
```

En la variable *datos* guardaremos los elementos. Como el tamaño del arreglo es fijo, necesitamos otra variable, a la que llamamos *largo*, que indique cuántas posiciones del arreglo *datos* están siendo usadas.

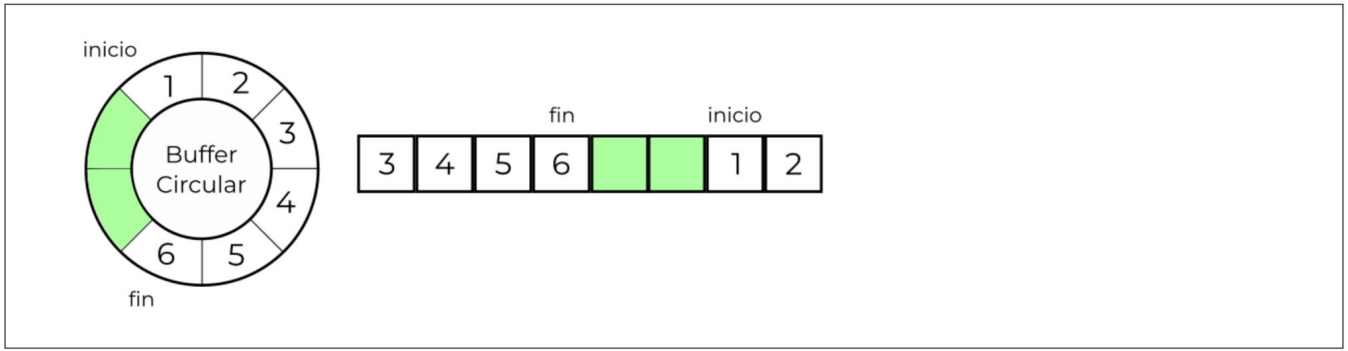
Con esta misma estructura, tenemos dos opciones: permitir que en el arreglo haya elementos repetidos o no permitirlo.

- Escriba el invariante de representación y la función de abstracción para ambos casos (con y sin repetidos)
- ¿Cuál es más eficiente? Cuándo usaría cada una de las dos versiones?
- Escriba los algoritmos para las operaciones de `agregar` un elemento y `sacar` un elemento para ambas versiones
- Respecto de la operación `sacar`, piense un algoritmo que no requiera generar un nuevo arreglo para reemplazar a *datos*, sino que se resuelva modificando alguna de sus posiciones

¡Intervalo!

Diseñar conjunto acotado

Ejercicio 4. Una forma eficiente de implementar el TAD `Cola` en su versión acotada (con una cantidad máxima de elementos predefinida), es mediante un *buffer circular*. Esta estructura está formada por un array del tamaño máximo de la cola (n) y dos índices (*inicio* y fin), que indican en qué posición empieza y en qué posición termina la cola, respectivamente. Al encolar un elemento, se lo guarda en la posición indicada por el índice *inicio* y se incrementa dicho índice. Al desencolar un elemento, se devuelve el elemento indicado por el índice fin y se incrementa el mismo. En ambos casos, si el índice a incrementar supera el tamaño del array, se lo reinicia a 0.



- Elija una estructura de representación
- Escriba el invariante de representación y la función de abstracción
- Escriba los algoritmos de las operaciones `encolar` y `desencolar`
- ¿Por qué tiene sentido utilizar un buffer circular para una cola y no para una pila?

¿Qué sigue?

- Con la clase de hoy pueden resolver **toda** la **guía 5**
- La clase que viene veremos cálculos de complejidad

¡Terminamos!

¡Hagan consultas!