Arreglos

Programación I - UNGS

Arreglos

- Un arreglo es una colección de valores donde:

Arreglos

- Un arreglo es una colección de valores donde:
  - Cada valor está identificado por un índice,

Arreglos

- Un arreglo es una colección de valores donde:
  - Cada valor está identificado por un índice,
  - Todos los valores son del mismo tipo,

Arreglos

- Un arreglo es una colección de valores donde:
  - Cada valor está identificado por un índice,
  - Todos los valores son del mismo tipo,
  - El largo de la colección es fijo.

Arreglos

- Un arreglo es una colección de valores donde:
  - Cada valor está identificado por un índice,
  - Todos los valores son del mismo tipo,
  - El largo de la colección es fijo.
- Los arreglos **son objetos**.

Declaración

- Se pueden definir arreglos de cualquier “tipo”.

Declaración

- Se pueden definir arreglos de cualquier “tipo”.
- El tipo de un arreglo se escribe como el tipo de los valores que contiene, seguido de corchetes ([ ]).

Declaración

- Se pueden definir arreglos de cualquier "tipo".
- El tipo de un arreglo se escribe como el tipo de los valores que contiene, seguido de corchetes ([ ]).
- Por ejemplo, el tipo de un arreglo de enteros es $\mathrm{int}[\,]$ y un arreglo de $\mathrm{doubles}$ es de tipo $\mathrm{double}[\,]$:

```
1 int[ ] cuenta;
2 double[ ] decimales;
```

Declaración

- Se pueden definir arreglos de cualquier "tipo".
- El tipo de un arreglo se escribe como el tipo de los valores que contiene, seguido de corchetes ([ ]).
- Por ejemplo, el tipo de un arreglo de enteros es $\mathrm{int}[\,]$ y un arreglo de $\mathrm{doubles}$ es de tipo $\mathrm{double}[\,]$:

```
1 int[ ] cuenta;
2 double[ ] decimales;
```

- ¿Cuánto vale un arreglo que no fue inicializado?

Declaración

- Se pueden definir arreglos de cualquier "tipo".
- El tipo de un arreglo se escribe como el tipo de los valores que contiene, seguido de corchetes ([ ]).
- Por ejemplo, el tipo de un arreglo de enteros es $\mathrm{int}[\,]$ y un arreglo de $\mathrm{doubles}$ es de tipo $\mathrm{double}[\,]$:

```
1 int[ ] cuenta;
2 double[ ] decimales;
```

- ¿Cuánto vale un arreglo que no fue inicializado?
- Las variables de tipo arreglo valen null hasta no ser inicializadas (como cualquier otra variable de tipo objeto).

Creación

- Para crear un objeto del tipo arreglo, usamos la palabra **new** en una expresión de la forma "**new** tipo[largo]".

Creación

- Para crear un objeto del tipo arreglo, usamos la palabra **new** en una expresión de la forma "**new** tipo[largo]".
- Por ejemplo:

```
1 cuenta = new int[4];
2 decimales = new double[largo];
```

Creación

- Para crear un objeto del tipo arreglo, usamos la palabra **new** en una expresión de la forma "**new** tipo[largo]".
- Por ejemplo:

```
1 cuenta = new int[4];
2 decimales = new double[largo];
```

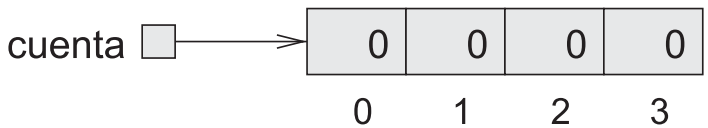
- cuenta referenciará a un arreglo de cuatro enteros y decimales a un arreglo de doubles del tamaño que indique largo al momento de ejecutarse la sentencia.

Creación

- Para crear un objeto del tipo arreglo, usamos la palabra **new** en una expresión de la forma "**new** tipo[largo]".
- Por ejemplo:

```
1 cuenta = new int[4];
2 decimales = new double[largo];
```

- cuenta referenciará a un arreglo de cuatro enteros y decimales a un arreglo de doubles del tamaño que indique largo al momento de ejecutarse la sentencia.
- Los elementos se inicializan con el valor por defecto del tipo correspondiente (0 para los números, false para boolean, null para objetos)



Acceder a los elementos

- Para acceder a un elemento del arreglo se debe escribir el nombre de la variable que referencia al arreglo seguido del índice entre corchetes ([i]).

Acceder a los elementos

- Para acceder a un elemento del arreglo se debe escribir el nombre de la variable que referencia al arreglo seguido del índice entre corchetes ($[\mathrm{i}]$).
- Por ejemplo, $\mathrm{cuenta}[0]$ es el primer elemento del arreglo, $\mathrm{cuenta}[1]$ el segundo y $\mathrm{cuenta}[\mathrm{i}]$ es el elemento en la posición i.

Acceder a los elementos

- Para acceder a un elemento del arreglo se debe escribir el nombre de la variable que referencia al arreglo seguido del índice entre corchetes ([i]).
- Por ejemplo, $\mathrm{cuenta}[0]$ es el primer elemento del arreglo, $\mathrm{cuenta}[1]$ el segundo y $\mathrm{cuenta}[\mathrm{i}]$ es el elemento en la posición i.
- Los elementos de un arreglo pueden ser utilizados como cualquier otra variable:

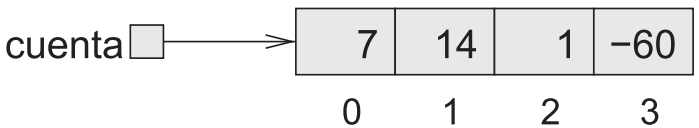
```
1 cuenta[0] = 7;
2 cuenta[1] = cuenta[0] * 2;
3 cuenta[2]++;
4 cuenta[3] -= 60;
```

Acceder a los elementos

- Para acceder a un elemento del arreglo se debe escribir el nombre de la variable que referencia al arreglo seguido del índice entre corchetes ($[\mathrm{i}]$).
- Por ejemplo, $\mathrm{cuenta}[0]$ es el primer elemento del arreglo, $\mathrm{cuenta}[1]$ el segundo y $\mathrm{cuenta}[\mathrm{i}]$ es el elemento en la posición i.
- Los elementos de un arreglo pueden ser utilizados como cualquier otra variable:

```
1 cuenta[0] = 7;
2 cuenta[1] = cuenta[0] * 2;
3 cuenta[2]++;
4 cuenta[3] -= 60;
```

- El resultado después de ejecutar el código anterior es el siguiente:



Copiar arreglos

- ¿Qué efecto tendría el siguiente código?

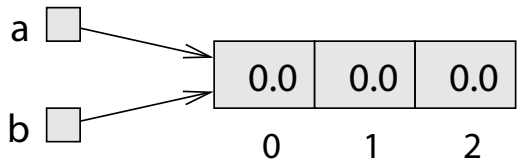
```
1 double[ ] a = new double[3];
2 double[ ] b = a;
```

Copiar arreglos

- ¿Qué efecto tendría el siguiente código?

```
1 double[ ] a = new double[3];
2 double[ ] b = a;
```

- Como con cualquier objeto, las variables contienen sólo referencias al arreglo, de modo que sólo se copiaría una referencia:

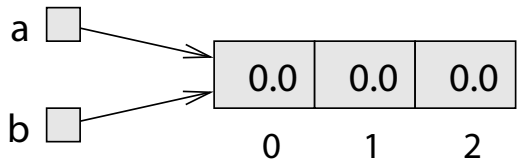


Copiar arreglos

- ¿Qué efecto tendría el siguiente código?

```
1 double[ ] a = new double[3];
2 double[ ] b = a;
```

- Como con cualquier objeto, las variables contienen sólo referencias al arreglo, de modo que sólo se copiaría una referencia:



- a y b ahora son dos nombres para el mismo objeto: es decir, hacen **aliasing**.

Copiar arreglos

- Para copiar un arreglo no tenemos otro remedio que crear uno nuevo y copiar elemento a elemento.

Copiar arreglos

- Para copiar un arreglo no tenemos otro remedio que crear uno nuevo y copiar elemento a elemento.
- Podemos utilizar la variable de instancia length que contiene el largo del arreglo:

```
1  double[ ] b = new double[a.length];
2  for(int i=0; i<a.length; i++)
3  {
4     b[i] = a[i];
5  }
```

Copiar arreglos

- Para copiar un arreglo no tenemos otro remedio que crear uno nuevo y copiar elemento a elemento.
- Podemos utilizar la variable de instancia length que contiene el largo del arreglo:

```
1 double[ ] b = new double[a.length];
2 for(int i=0; i<a.length; i++)
3 {
4   b[i] = a[i];
5 }
```

- También podemos usar el método de instancia $\mathrm{clone}()$ de los arreglos que hace esto mismo por nosotros:

```
1 double[ ] b = a.clone();
```

Consideraciones de eficiencia

- El concepto de “secuencia” es algo muy cotidiano.

Consideraciones de eficiencia

- El concepto de “secuencia” es algo muy cotidiano.
- Ejemplos:

Consideraciones de eficiencia

- El concepto de “secuencia” es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).

Consideraciones de eficiencia

- El concepto de “secuencia” es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).
  - Cartelera del cine (películas).

Consideraciones de eficiencia

- El concepto de "secuencia" es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).
  - Cartelera del cine (películas).
  - Lista de invitados a un evento (personas).

Consideraciones de eficiencia

- El concepto de “secuencia” es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).
  - Cartelera del cine (películas).
  - Lista de invitados a un evento (personas).
  - etc.

Consideraciones de eficiencia

- El concepto de "secuencia" es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).
  - Cartelera del cine (películas).
  - Lista de invitados a un evento (personas).
  - etc.
- Cuando tenemos que representar el concepto de secuencia en una computadora, tenemos varias alternativas.

Consideraciones de eficiencia

- El concepto de "secuencia" es algo muy cotidiano.
- Ejemplos:
  - Lista de compras del supermercado (productos).
  - Cartelera del cine (películas).
  - Lista de invitados a un evento (personas).
  - etc.
- Cuando tenemos que representar el concepto de secuencia en una computadora, tenemos varias alternativas.
- Lo que distingue a los arreglos de otras variantes (que veremos más adelante) es la **eficiencia** de sus operaciones.

Operaciones: Búsqueda por índice

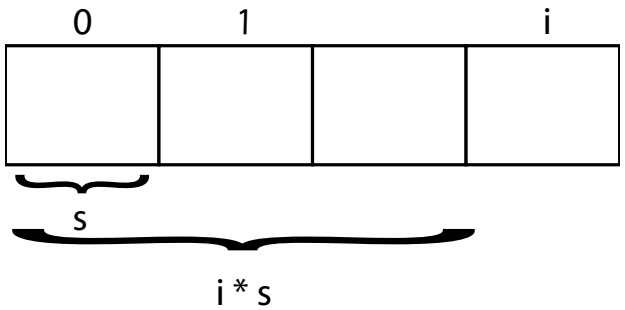
- Para buscar un elemento dado el índice del mismo, disponemos del operador [ ].

Operaciones: Búsqueda por índice

- Para buscar un elemento dado el índice del mismo, disponemos del operador [ ].
- Los elementos se guardan en memoria en forma consecutiva.

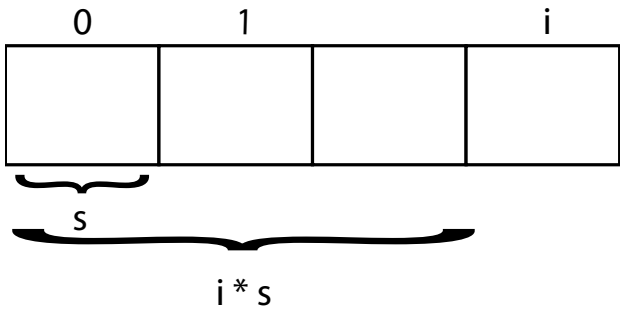
Operaciones: Búsqueda por índice

- Para buscar un elemento dado el índice del mismo, disponemos del operador [ ].
- Los elementos se guardan en memoria en forma consecutiva.
- Si cada elemento ocupa s bytes, encontrar el elemento en la posición i se resuelve yendo a buscarlo a la posición $i \times s$ después del inicio:



Operaciones: Búsqueda por índice

- Para buscar un elemento dado el índice del mismo, disponemos del operador [ ].
- Los elementos se guardan en memoria en forma consecutiva.
- Si cada elemento ocupa s bytes, encontrar el elemento en la posición i se resuelve yendo a buscarlo a la posición $i \times s$ después del inicio:



- Con lo cual, resolver esta operación tiene un tiempo de ejecución **constante**, independientemente del tamaño del arreglo.

Operaciones: Búsqueda por valor

- Si, en cambio, quisiéramos buscar la posición en la que se encuentra un valor (si está), no tenemos otro remedio que recorrer el arreglo entero.

Operaciones: Búsqueda por valor

- Si, en cambio, quisiéramos buscar la posición en la que se encuentra un valor (si está), no tenemos otro remedio que recorrer el arreglo entero.
- Por ejemplo, esta sería una implementación posible:

```
1  static int buscarValor(int[ ] arreglo, int val) {
2    for(int i = 0; i < arreglo.length; i++) {
3      if(arreglo[i] == val) {
4        return i; //Devuelve la posición
5      }
6    }
7    return -1; //Devuelve -1 si no se encontró
8  }
```

Operaciones: Búsqueda por valor

- El tiempo de ejecución de esta función se dice que es **lineal** porque toma un tiempo proporcional a la cantidad de elementos del arreglo.

Operaciones: Búsqueda por valor

- El tiempo de ejecución de esta función se dice que es **lineal** porque toma un tiempo proporcional a la cantidad de elementos del arreglo.
- Nótese que puede haber más de un elemento con el mismo valor. Esta función sólo devuelve el índice a la primera aparición.

Operaciones: Inserción

- Agregar un elemento nuevo a un arreglo, es imposible. La cantidad de elementos de un arreglo es fija.

Operaciones: Inserción

- Agregar un elemento nuevo a un arreglo, es imposible. La cantidad de elementos de un arreglo es fija.
- La única alternativa que tenemos es crear un nuevo arreglo de tamaño mayor al anterior, copiar todos los elementos y finalmente copiar el valor que queremos agregar.

Operaciones: Inserción

- Agregar un elemento nuevo a un arreglo, es imposible. La cantidad de elementos de un arreglo es fija.
- La única alternativa que tenemos es crear un nuevo arreglo de tamaño mayor al anterior, copiar todos los elementos y finalmente copiar el valor que queremos agregar.
- Nuevamente, si hiciéramos esto último, el tiempo de ejecución de esta operación sería lineal.

Operaciones: Eliminación

- Algo similar ocurre con el borrado de elementos: la cantidad es siempre la misma.

Operaciones: Eliminación

- Algo similar ocurre con el borrado de elementos: la cantidad es siempre la misma.
- Podemos de manera análoga crear un arreglo nuevo en que contenga todos los mismos elementos, menos el que queremos eliminar.

Operaciones: Eliminación

- Algo similar ocurre con el borrado de elementos: la cantidad es siempre la misma.
- Podemos de manera análoga crear un arreglo nuevo en que contenga todos los mismos elementos, menos el que queremos eliminar.
- Otra vez, el tiempo de ejecución de esta operación sería lineal.

Conclusión

- Los arreglos son una estructura de datos para almacenar una secuencia ordenada de elementos del mismo tipo.

Conclusión

- Los arreglos son una estructura de datos para almacenar una secuencia ordenada de elementos del mismo tipo.
- La cantidad de los elementos que almacena un arreglo se fija en su creación y esta cantidad no se puede modificar.

Conclusión

- Los arreglos son una estructura de datos para almacenar una secuencia ordenada de elementos del mismo tipo.
- La cantidad de los elementos que almacena un arreglo se fija en su creación y esta cantidad no se puede modificar.
- Acceder a un elemento dada la posición es una operación inmediata (tiempo constante ya que no depende del tamaño del arreglo).

Conclusión

- Los arreglos son una estructura de datos para almacenar una secuencia ordenada de elementos del mismo tipo.
- La cantidad de los elementos que almacena un arreglo se fija en su creación y esta cantidad no se puede modificar.
- Acceder a un elemento dada la posición es una operación inmediata (tiempo constante ya que no depende del tamaño del arreglo).
- Aunque otras operaciones pueden requerir tiempos más grandes (en general en función del tamaño del arreglo).

¿Preguntas?

¿Preguntas?