Programando con B4X

Tema 14 – Matrices - Arrays

Version 1.0, abril 2021

Autor original: Prokopis Leon

Traducido al español por José Miguel López

Tema 14 – Matrices - Arrays

⏱ **4h**

Lo que los estudiantes aprenderán
• Matrices unidimensionales (o Vectores) • Operaciones básicas con matrices • Elemento Máximo y Mínimo • Búsqueda lineal • Búsqueda binaria • Ordenación con Bubble Sort • Ordenación con Selection Sort

Un problema común en programación es la gestión de grandes cantidades de datos. Cuando un problema sólo necesita 6 o 7 variable, es fácil declararlas y usarlas. ¿Qué sucede cuando necesitamos usar muchos datos del mismo tipo a la vez? Por ejemplo, 100 estudiantes y 100 calificaciones; ¿cómo declarar 200 variables con nombres y calificaciones y cómo gestionarlas?

Una de las soluciones en ciencias de la computación consiste en usar arrays (o matrices, vectores). En general, un array se define como un conjunto de datos del mismo tipo que puede almacenarse en la memoria del ordenador de forma contigua. Así, el desarrollador puede localizarlas simplemente moviéndose de una ubicación a otra sin tener que emplear nombres diferentes.

En la imagen 1 puedes ver un ejemplo de array con las notas de 14 estudiantes para un tema. Todas las notas se colocan en posiciones contiguas y sólo usamos un nombre (Notas). Para referirse a cada posición dentro de la tabla el programador debe indicar el nombre del array (Notas) y después escribir entre paréntesis el número de la celda donde está el dato que desea consultar.

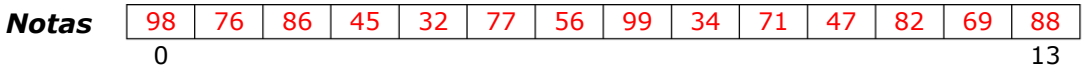


Imagen 1. Array llamado "Notas" con 14 huecos para notas

Así, por ejemplo, para mostrar el primer elemento del array tan sólo hay que escribir "Notas(0)", donde 0 es la primera posición del array.

Declaración de Arrays

Un array se declara igual que otras variables:

```
Private Notas(14) As Int
```

Aquí, Notas es el nombre del array y el número entre paréntesis nos dice cuántas posiciones (o tamaño) habrá en el array. Una vez que se declara un array con un tamaño determinado, no se puede cambiar en el código a menos que se vuelva a declarar con un nuevo tamaño.

Funciones sobre Arrays

Insertar elementos en un array

Para insertar un elemento en un array hay que asignar un valor al lugar correspondiente en el array. Por ejemplo:

```
Notas(0) = 89
```

Se podría seguir con todos los demás elementos así, pero es más fácil si usamos un proceso repetitivo para rellenar el array. En el caso del array Notas, en B4X podría ser así:

```
Private Notas(14)  As Int
For i = 0  To  13
  Notas(i) = Rnd(1,100)
Next
```

El código anterior rellena el array con números aleatorios de 1 a 100. Fíjate que la primera posición empieza en 0. La variable **i** indica en cada iteración

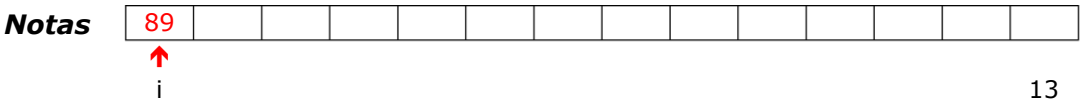


Imagen 2. Insertar elementos en un array

del bucle la posición del array en que insertaremos un valor y se le llama **índice** del array. Moviendo el índice i con una sentencia de repetición podemos acceder a cada posición del array.

Una segunda forma de insertar elementos es así:

```
Private Notas()  As Int
Notas = Array As Int(19,43,12,65,23,87,45,65,87,23,56)
```

El tamaño del array no se indica al declararlo, pero sí al realizar la inserción de los elementos con la **sentencia Array.**

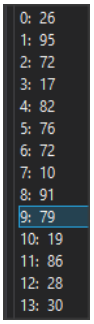
Es evidente que puedes tener arrays de cualquier tipo como, por ejemplo, cadenas de texto, decimales, etc. Pero **nunca puedes mezclar** los tipos en un array.

Mostrar los elementos de un array

Usando la función "Log" puedes imprimir un elemento de un array o el array al completo mediante una iteración.

```
Log(Notas(0)) ' Muestra el 1er elemento del Array Notas

For i = 0 to 13
  Log(i & ": " & Notas(i))
Next
```

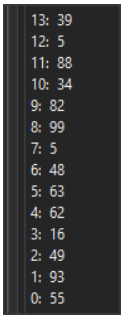


El ejemplo anterior usa un bucle For con un índice i para mostrar el valor que hay en la posición i del array.

Intentar acceder a una posición fuera del Array (en el ejemplo, la posición 14 o más alta) generaría un fallo del programa (excepción denominada "out-of-bounds index exception"), así que es muy importante que prestes atención a la posición a la que accedes y a los límites de un array.

Mostrar elementos en orden inverso

El siguiente código muestra los elementos de un array empezando por el final:




```
For i = 13 to 0 Step -1
  Log(i & ": " & Notas(i))
Next
```

En general, puedes modificar el índice como quieras en un bucle de repetición o usar los elementos del array que quieras.

Calcular la suma y la media de los elementos de un array

Las reglas aplicables a procesos repetitivos en los algoritmos se pueden aplicar en general a los arrays con pocos cambios. Así, la suma de los elementos de un array necesita del uso de una variable extra que guarde la suma (que llamaremos "suma") y de aplicar una repetición sobre el array.

```
Private intSuma As Int = 0
Private fltMedia as Float
For i = 0 to 13
  intSuma = intSuma + Notas(i)
Next
fltMedia = intSuma / 14
Log(intSuma & fltMedia)
```

Encontrar el máximo y el mínimo

Encontrar el elemento máximo y el mínimo en un array requiere del uso de variables adicionales llamadas, normalmente, **Máx** y **Mín**. Los valores iniciales de estas variables se fijan en el primer elemento del array y después se compara uno a uno cada elemento del array para ver si es mayor que **Máx** (o menor que **Mín**), actualizando sus valores en consecuencia.

```
Private intMáx, intMín  As Int
intMáx = Notas(0)
intMín = Notas(0)
For i = 0 To 13
   If intMáx < Notas(i)  Then
      intMáx = Notas(i)
   End If
   If intMín > Notas(i)  Then
      intMín = Notas(i)
   End If
Next
Log("Máximo = " & intMáx)
Log("Mínimo = " & intMín)
```

Algoritmos de búsqueda

Buscar un elemento en un array consiste en escanearlo para encontrar elemento que cumple una condición concreta.

En este tema veremos la búsqueda secuencial y la búsqueda binaria.

Búsqueda secuencial

Es el más simple, pero el más lento. Consiste en comprobar uno a uno todos los elementos del array para encontrar el elemento buscado. El siguiente código muestra las posiciones del array que contienen el valor clave.

```
'Buscar todas las posiciones que contienen un valor clave
For i = 0  To  999
   If Notas(i) = clave  Then
      Log("encontrado en la posición: " & i)
   End If
Next
```

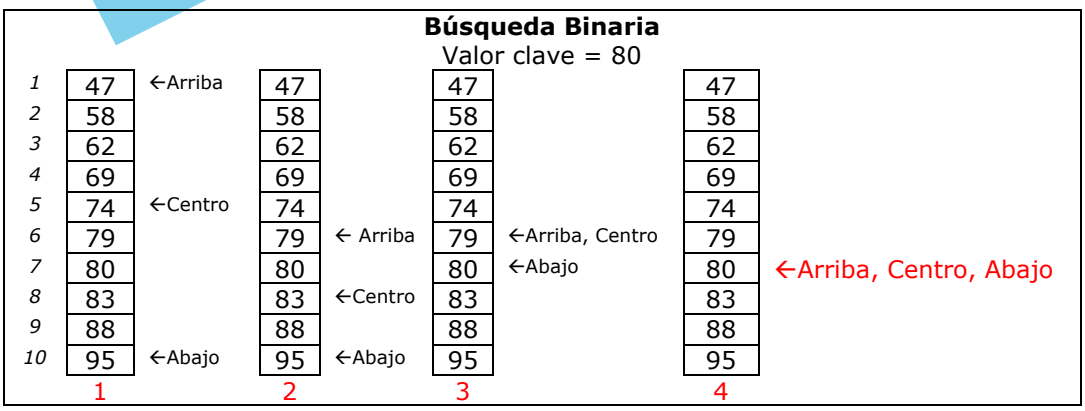
Si sólo hay que buscar la primera aparición de un valor clave, se puede usar una variable lógica llamada "encontrado" que pondremos a "true" cuando encontremos el elemento buscado.

```
Private encontrado  As Boolean  = False
i = 0
Do While Not (encontrado)  And  i <= 999
   If Notas(i) = clave  Then
      Log("encontrado en la posición : " & i )
      encontrado = True
   End If
   i = i + 1
Loop
If Not(encontrado)  then
   Log("No se ha encontrado")
End If
```

Búsqueda Binaria

La búsqueda binaria sólo se aplica a arrays ordenados. La filosofía básica del método consiste en comprobar el valor que hay en la mitad del array. Si el elemento que buscamos es menor que ese valor central, entonces buscamos en la mitad superior del array. Si es mayor, entones buscamos

en la mitad inferior del array. En el siguiente ejemplo buscamos el valor 80 en un array Notas con 10 valores ordenados de menor a mayor:



1. Inicialmente, las posiciones primera y última se guardan en las variables "Arriba" y "Abajo". El "centro" del array es el resultado de la división: (Arriba+Abajo)/2.
2. Comparamos el valor de Notas(Centro) con la clave y si es menor, a la variable "Abajo" se le pone el valor de "Centro" más 1 (porque ya hemos comprobado que el valor en "Centro" no es el que buscamos). Si fuera mayor, el valor de "Arriba" sería una posición más abajo de "Centro".
3. Repetimos los pasos anteriores hasta que encontremos el valor buscado o bien la posición de "Arriba" sea mayor que la de "Abajo" (eso implicaría que ya hemos recorrido todo el array y no hemos encontrado la clave buscada).

Ordenación

Existen varios algoritmos para ordenar un array. Veremos el método de la burbuja (Bubble Sort) y la ordenación por selección (Selection Sort).

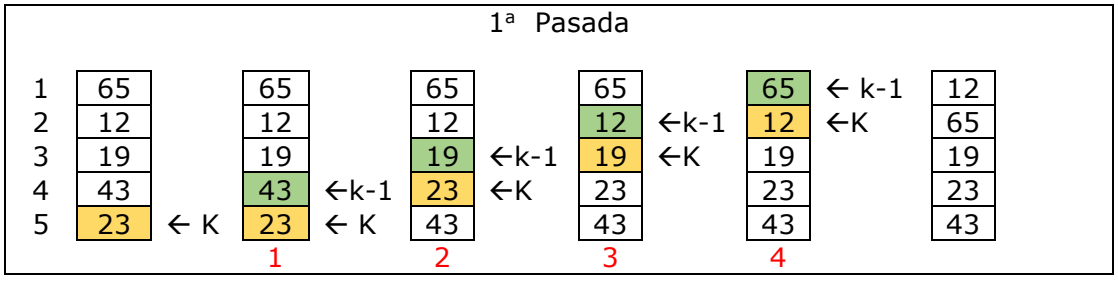
Ordenación por Burbuja

La ordenación por burbuja es un algoritmo sencillo en el que se recorre todo el array comparando los elementos adyacentes, intercambiándolos si están en el orden incorrecto. El array se recorre varias veces hasta que está ordenado. El nombre de "burbuja" se le puso porque los elementos más pequeños o más grandes suben como una burbuja hacia arriba.

Ejemplo de Ordenación por Burbuja

Ordenaremos un array "Notas" que contiene 5 enteros de menor a mayor.

```
Private Notas()  As Int
Notas = Array As Int(65,12,19,43,23)
```



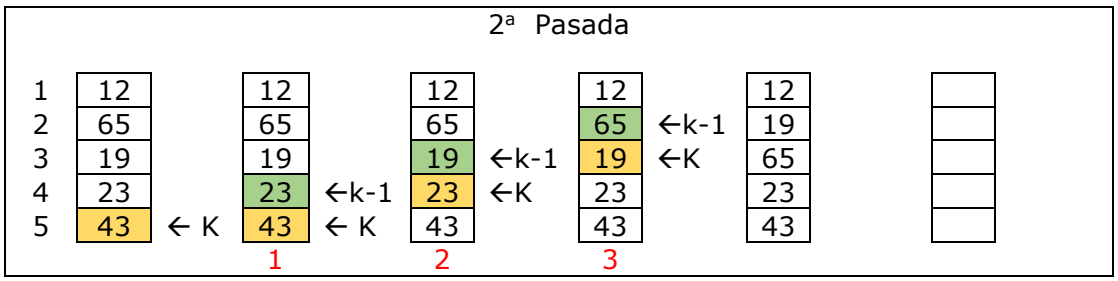
Al principio el algoritmo comienza por la última posición del array y compara cada elemento con el anterior secuencialmente:

1. La primera comparación se hace con los valores de las celdas 5 y 4, comprobando que Notas(5) es menor que Notas(4), con lo que se intercambian sus valores.
2. A continuación se comparan las celdas 3 y 4, comprobando que Notas(4) es mayor que Notas(3), con lo que no se hacen cambios en el array.
3. En las posiciones 3 y 2 tampoco se intercambian los valores.
4. En el último paso, las posiciones 1 y 2 se comparan y se intercambian sus valores de nuevo.

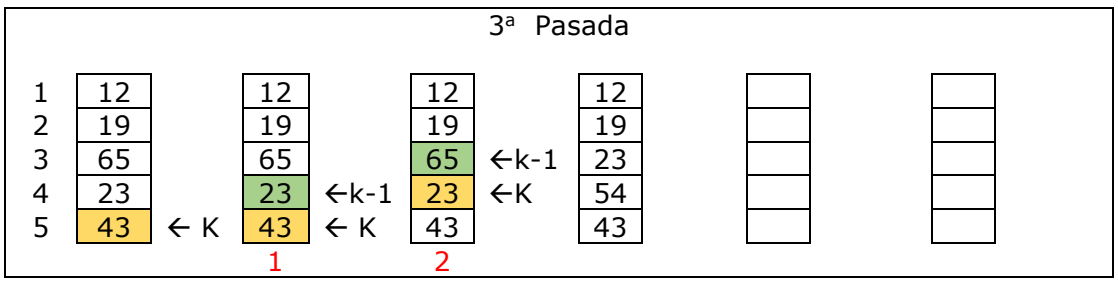
El primer paso se implementa con el siguiente código:

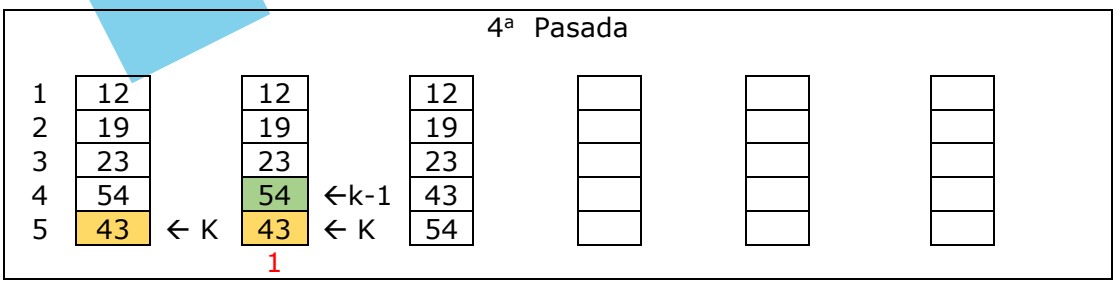
```
Private Notas()  As  Int
Private temp  As Int
Notas = Array As Int(19,43,12,65,23)
For k = 4  To  1  Step  -1
   If Notas(k)  <  Notas(k-1)  Then
      temp =  Grades(k)
      Notas(k) =  Notas(k-1)
      Notas(k-1) =  temp
   End If
Next
```

Ahora el elemento más pequeño está en la parte superior del array.



En la segunda pasada se aplica el mismo procedimiento, excepto que la primera celda no se compara porque el elemento más pequeño ya está en lo más alto del array.





Las pasadas continúan hasta que se orden por completo el array. Fíjate que en cada pasada se comprueban menos elementos, ya que en cada iteración el elemento más pequeño sube hacia arriba (burbuja).

En general, el número de pasadas que hay que hacer es el tamaño del array -1. En el ejemplo, para un array de 5 elementos, se hacen 4 pasadas. El código completo es el siguiente:

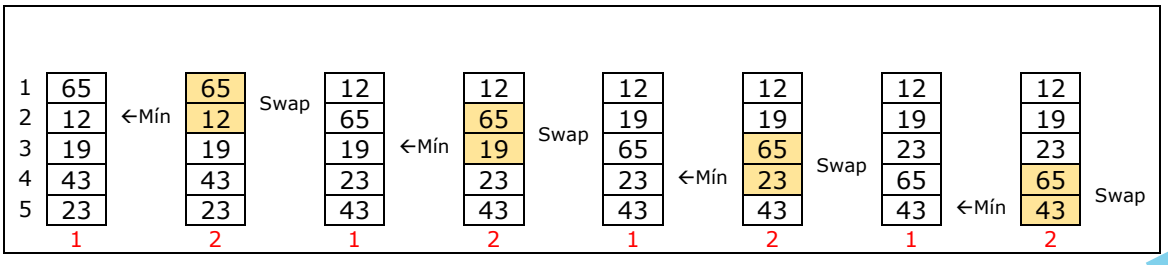
```
Private Notas()  As  Int
Private temp  As Int
Notas = Array As Int(19,43,12,65,23)
For i  = 1  to  4        'i  cuenta las pasadas
   For k = 4  To  i  Step  -1
      If Notas(k)  <  Notas(k-1)  Then
         temp =  Grades(k)
         Notas(k) =  Notas(k-1)
         Notas(k-1) =  temp
      End If
   Next
Next
```

Ordenación por Selección

El algoritmo divide el array en dos partes: una parte ordenada que se construye de izquierda a derecha (la izquierda es la primera posición del array) y otra parte que es un subarray con los demás elementos desordenados. El algoritmo busca el elemento más pequeño (o el más grande, según el orden que se desee) en el subarray desordenado, intercambiándolo (swapping) con el elemento más a la izquierda de la lista desordenada (dejándolo así en orden).

La implementación según los pasos indicados sería:

1. Elegir el elemento más pequeño
2. Intercambiar el mínimo con el primer elemento
3. Repetir los pasos 1 y 2 para todos los elementos del array



```
Private Notas()  As Int
Notas = Array As Int(65,12,19,43,23)
Private intMín, intMínPos  As Int

For k = 0  To  4
   intMín = Notas(k)
   intMínPos  = k
   For i = k  To  4    'Buscar el mínimo desde la posición k a la 5ª
      If intMín > Notas(i)  Then
         intMín = Notas(i)
         intMínPos  = i
      End If
   Next
   Notas(intMínPos) = Notas(k)
   Notas(k) = intMín
Next
```

Ejercicios

1.

a. Escribe un programa que rellene un array llamado **A(50)** con 50 números enteros aleatorios entre 1 y 100.
b. Calcule y muestre la suma de los elementos del array A(50) en posiciones pares.
c. Si la suma de los primeros 25 elementos del array es igual a la suma de los últimos 25 elementos, mostrar el mensaje "Son iguales".
d. Si A(1)=A(50), A(2)=A(49), A(3)=A(48)... A(25)=A(26), entonces se mostrará el mensaje "Array simétrico".
e. Encontrar el elemento máximo y la posición donde está.
f. Crea una subrutina que ordene el Array A.
g. Crea una subrutina que reciba un array y un entero y aplique la búsqueda binaria a ese número en el array. Deberá mostrar la posición del elemento o -1 si no lo encuentra.
h. Usando las dos anteriores subrutinas, ordena la tabla A, busca el número 67 y muestra un mensaje para indicar si se ha encontrado o no.

2.

a. Tenemos un array que guarda la temperatura media de cada día de un año que se llama **Temperatura(365)**. Supón que las temperaturas son números enteros entre 1 y 40ºC. Encuentra y muestra la frecuencia de cada temperatura.
b. En el array anterior **Temperatura**, encuentra la segunda mayor temperatura del año.