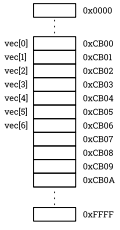
**UNIDAD 7 - ARREGLOS EN LENGUAJE C**

ARREGLOS:

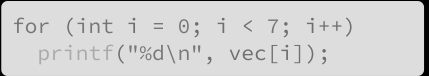
Los arreglos son un conjunto de posiciones de memoria contiguos, en donde se pueden almacenar valores del mismo tipo.

Tienen un nombre que debe respetar las características de los identificadores

Se puede acceder a cada valor mediante el nombre y el índice entre corchetes.



El índice puede ser una variable o cualquier expresión.



El primer elemento siempre es el índice “0”(cero)

DEFINICIÓN

Tipo identificador[tamaño];

Tipo se refiere a cualquier tipo de dato (int, char, float, etc.).

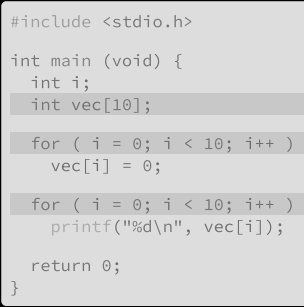
Identificador es el nombre que se usará para acceder a los elementos del arreglo. Debe cumplir los requisitos de cualquier identificador.

Tamaño es una constante que indica cuántos elementos tendrá el arreglo.



Esto define un arreglo llamado VEC de 100 enteros.

El estándar c99 determina que el tamaño máximo debe ser al menos 65536 elementos… pero el verdadero tamaño máximo depende de diversos factores relacionados a la memoria del programa (se verán más adelante en la carrera)



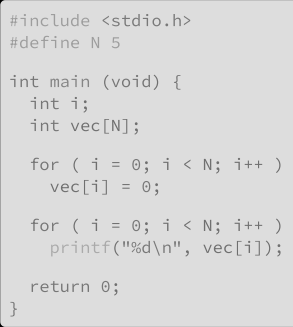
Si eventualmente se necesita un arreglo más grande, hay que cambiar el tamaño del arreglo... ...y el control de las sentencias repetitivas.

Si el programa tiene gran extensión, hacer estos cambios puede llevar a cometer errores.

Para evitarlos se puede utilizar la directiva de preprocesador #define

PREPROCESADOR:

Directiva #define



Se utiliza para definir constantes simbólicas

Antes de la compilación, el preprocesador reemplaza todas las constantes simbólicas por el valor que corresponde.

Los nombres de las constantes deben respetar las características de los identificadores, y se recomienda el uso de mayúsculas.

No deben usarse los punto y coma (;) en las directivas #define ya que luego, cuando se hagan los reemplazos podría haber errores de sintaxis.

ARREGLOS: INICIALIZACIÓN

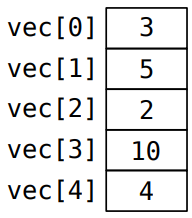
Así como las variables comunes se podían inicializar en la definición.



Los arreglos se pueden inicializar en la definición elemento por elemento usando llaves.

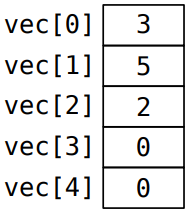


Los elementos entre llaves se asignarán en orden desde el índice cero.



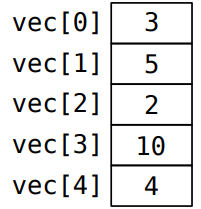
Si los valores para inicializar son menos que el tamaño del arreglo, el resto de los elementos son inicializados en cero.





Si los valores para inicializar son más que el tamaño del arreglo, el compilador da un error o warning dependiendo de si está la opción --pedantic-errors





Se puede inicializar todo el arreglo poniendo entre llaves un número cero



Estrictamente hablando de esta forma se inicializa el primer elemento en cero, y el resto de los elementos en cero.

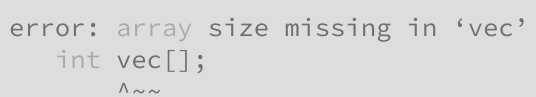
Se puede omitir el tamaño entre los corchetes, siempre y cuando se usen las llaves para inicializar.



pero el tamaño del arreglo siempre corresponderá a la cantidad de elementos inicializados entre llaves

Si se omiten tanto el tamaño entre corchetes como los inicializadores el compilador da error





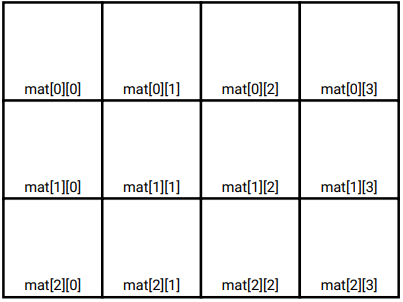
ARREGLO DE 2 DIMENSIONES

En C se pueden definir arreglos multidimensionales



El número entre corchetes a la izquierda determina la cantidad de filas.

El número entre corchetes a la derecha determina la cantidad de columnas.

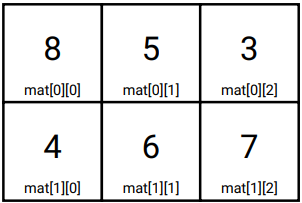


Para acceder a cualquier elemento hay que utilizar los índices de filas y columnas.

ARREGLO DE 2 DIMENSIONES- INICIALIZACIÓN

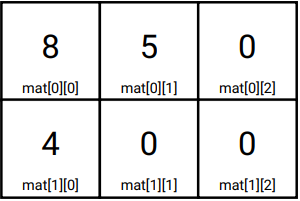
Como los arreglos de 1 dimensión, se utilizan llaves por cada fila y todas las filas también entre llaves.





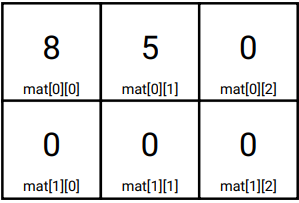
Se pueden inicializar de forma incompleta, donde los elementos faltantes se ponen en cero.





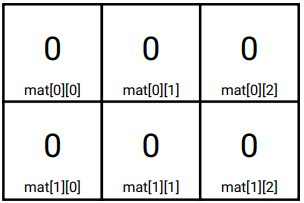
incluso pueden faltar filas.





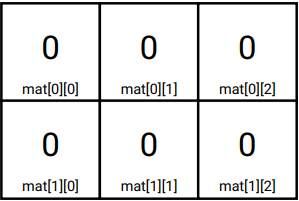
O todos los elementos, salvo el primero. Sirve para inicializar en cero todo el arreglo.





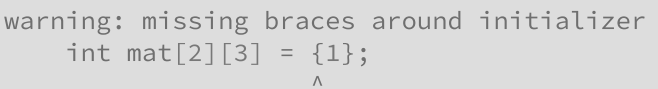
Se pueden omitir un par de llaves en el caso de ser cero…



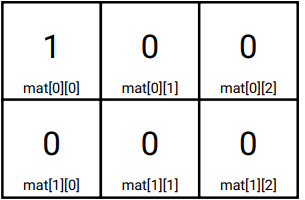


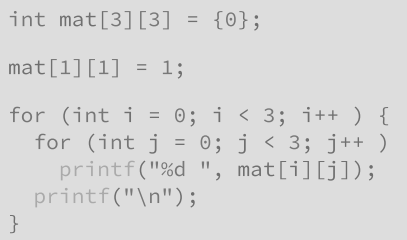
...pero da un warning si se usa otro número



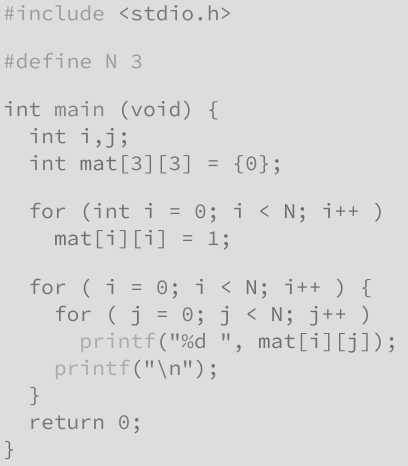


...de todas formas la inicialización es la esperada (pero no hay que dejar pasar los warnings)







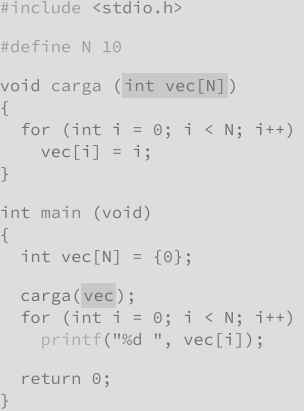


PASAJE DE ARREGLOS A FUNCIONES

A diferencia de las variables, al pasar los arreglos a las funciones, los cambios que se hagan sobre el arreglo, afectan al arreglo original.

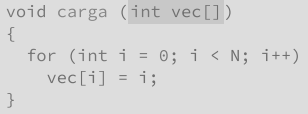
En el llamado no deben usarse los corchetes, sólo el nombre del arreglo

En la definición el arreglo, en la lista de parámetros, debe tener corchetes

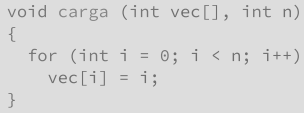




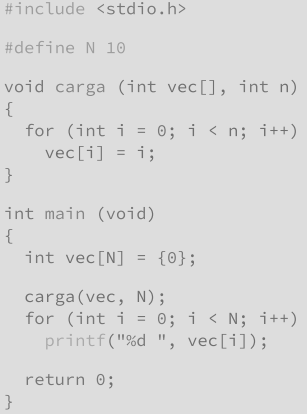
Por motivos que se ampliarán en la unidad 8, se puede definir el arreglo en la lista de parámetros sin ningún valor entre corchetes.



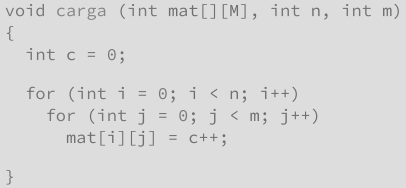
Una buena práctica de programación consiste en no incluir ningún valor "global" en la función.



Entonces, los parámetros de la función son: el arreglo, con corchetes sin valores, y la dimensión del mismo, pero como variable

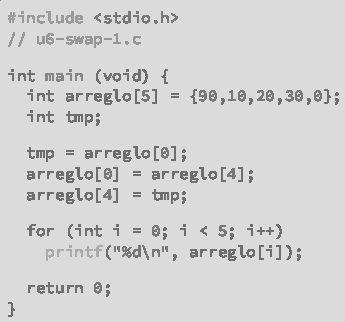


En el caso de arreglos de más dimensiones, el procedimiento es el mismo, con la salvedad que solo puede omitirse el primer límite del arreglo en la definición



**ALGORITMOS**

INTERCAMBIO

Para intercambiar dos elementos cualesquiera de un arreglo, hay que usar una variable temporal para no perder uno de los valores

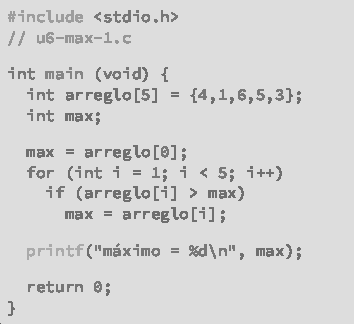
primero hay que salvar uno de los valores (tmp = arreglo[0];)

luego se puede pisar el elemento que contenía el valor salvado usando el nuevo elemento (arreglo[0] = arreglo[4];)

Finalmente, se recupera el valor salvado en la variable temporal colocándolo en la otra variable (arreglo[4] = tmp;)

BÚSQUEDA LINEAL O SECUENCIAL

Es un método para encontrar un valor en un arreglo. Por ejemplo, encontrar un máximo o un mínimo



Primero, se inicializa la variable max con el primer elemento del arreglo (max = arreglo[0];)

Luego se recorre el arreglo buscando algún valor que supere a max (for + if)

Cuando se lo encuentra se actualiza max (max = arreglo[i];)

En este caso, al final del bucle, la variable max almacena el máximo valor (y puede se mostrar)

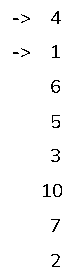
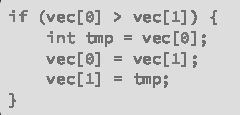
ORDENAMIENTO: MÉTODO DE LA BURBUJA

Consiste en comparar cada elemento con el siguiente e intercambiarlos si corresponde según el criterio de ordenamiento

Por ejemplo, si se propone ordenar de menor a mayor el

siguiente arreglo



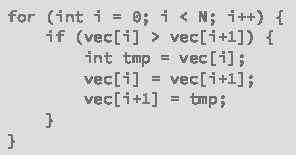
Se compara el elemento 0 con el 1, si el 0 es mayor que el 1, se intercambian

Se adelantan una posición ambos índices (se pasa del 4 al 6)

Se compara el elemento 1 con el 2, si el 1 es mayor que el 2, se intercambian

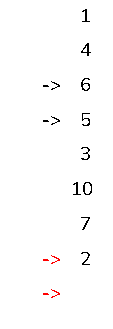
Se adelantan una posición ambos índices (se pasa del 6 al 5)

Esto se hace para todo el arreglo, así que se puede usar una variable para los índices de los elementos y un for para controlar esa variable.

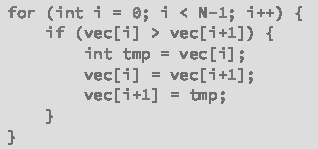


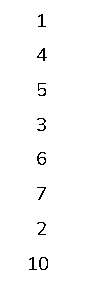
pero OJO!

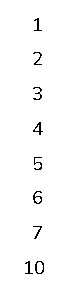
Al final del for hay un error lógico

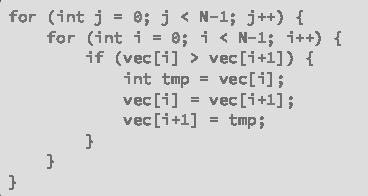


Para solucionar este error el for debe hacer una repetición menos

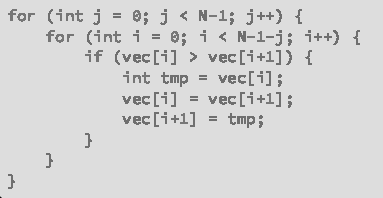


Así siempre el mayor queda al final del arreglo. Pero el arreglo no está ordenado

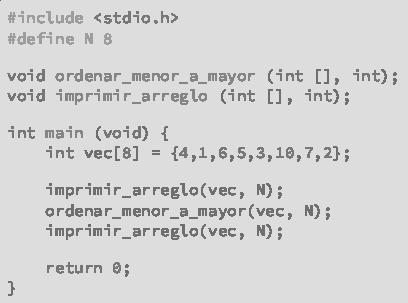
En el peor caso, donde el elemento más pequeño está en la última posición, se deben hacer N-1 repeticiones

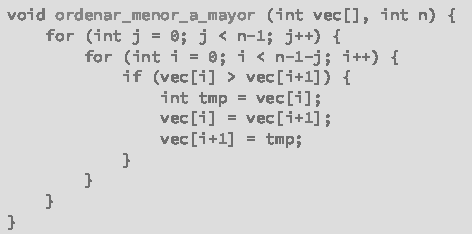


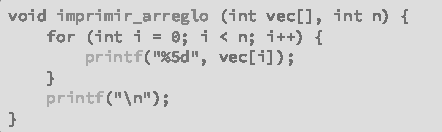


Se puede ver que en la primera pasada el elemento más grande del arreglo queda en su posición entonces en la segunda pasada, la última pregunta no hace falta, porque la respuesta nunca será verdad en la tercera pasada, las dos últimas preguntas no hace falta, porque las respuestas nunca serán verdad. Entonces se puede mejorar el algoritmo para que el alcance del lazo interno dependa del número de pasada (agregando -j al segundo for anidado)

Ejemplo con función:







ORDENAMIENTO: MÉTODO DE INSERCIÓN

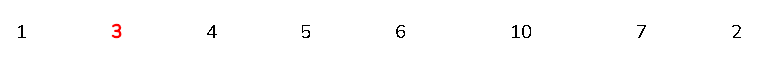
Consiste en ordenar cada elemento respecto de los elementos previamente ordenados

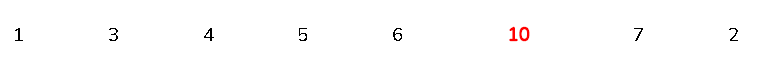
Supongamos un conjunto donde todos los elementos a la izq. del elemento en rojo están ordenados de menor a mayor

Buscaremos la posición del elemento rojo en la lista ordenada



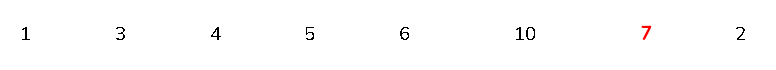
Se ordena a:



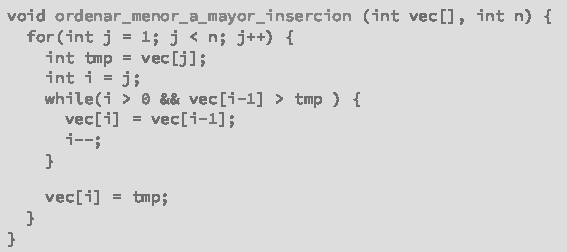
Se toma el 10:

Si se supone que el conjunto a la izq ya está ordenado, solo es necesario preguntar por el elemento más a la derecha del conjunto.

Si no es mayor el rojo se mueve a la derecha y se vuelve a comenzar



En código es:



ORDENAMIENTO: ORDENAMIENTO RÁPIDO (QUICKSORT)

La idea del QS es tomar el primer elemento del arreglo y colocarlo en su lugar final, o sea, en una posición en la que todos los elementos a su izq sean menores y todos los elementos a su derecha sean mayores.

Los dos grupos están desordenados, pero en cada uno se puede hacer lo mismo, tomar el primer elemento y colocarlo en su posición final.

Se repite esto con cada subconjunto que se va generando

¿Hasta cuándo hacemos esto?

Hasta que cada subconjunto tenga un solo elemento, entonces cada elemento estará en su lugar y el arreglo completo estará ordenado.

¿Cuál es la parte difícil?

Justamente determinar cuál es la posición final de un elemento aunque el arreglo esté desordenado.

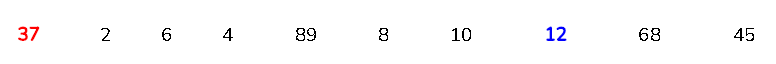
Supongamos el siguiente arreglo desordenado



1) Tomamos el primer elemento de la izq



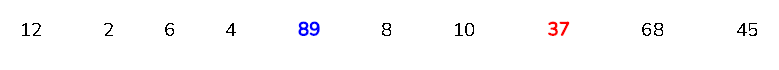
2) Buscamos desde la derecha el primer elemento que sea menor que el seleccionado



3) Intercambiamos estos elementos



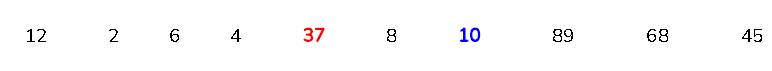
4) Desde la izq, pero comenzando con el elemento siguiente al recién colocado, se busca el primer elemento mayor al seleccionado



5) Intercambiamos estos elementos



6) Desde la derecha, pero comenzando con el siguiente elemento al recién colocado, buscamos el primer elemento menor al seleccionado



7) Intercambiamos estos elementos



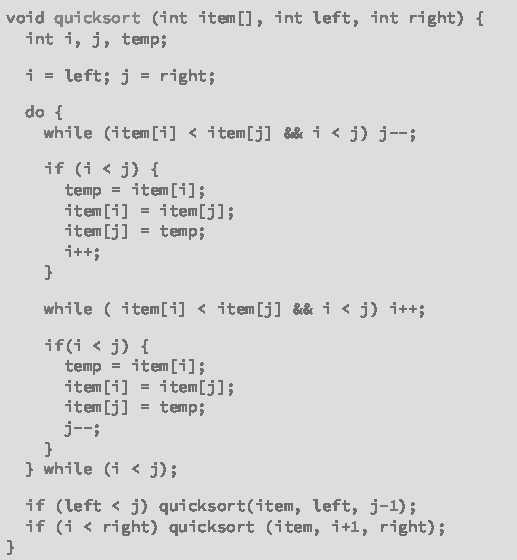
8) Desde la derecha, no hay elementos menores (el elemento que sigue es el seleccionado 37)

9) Desde la izquierda, no hay elementos mayores al elemento seleccionado 37

Ahora todos los elementos a la derecha del elemento seleccionado son mayores, y todos los elementos a la izquierda son menores, aunque estos subconjuntos no están ordenados.

Se puede hacer lo mismo con cada subconjunto usando una función recursiva.

La condición de corte será cuando el elemento seleccionado es el único del subconjunto



BÚSQUEDA BINARIA

A partir de un conjunto ordenado, se utiliza la estrategia divide y vencerás, para tratar de encontrar un elemento dentro del conjunto

Se selecciona un elemento central y se evalúa si es el elemento buscado, es mayor o es menor.

Se puede eliminar la mitad en la que no se encuentra el valor buscado, lo que lo hace muy veloz

