Práctica 7. TIPOS DE DATOS ESTRUCTURADOS
Arrays

OBJETIVOS:
- ✓ Conocer la importancia que tienen los arrays como estructuras de datos en el ámbito de la programación.
- ✓ Conocer los distintos tipos de arrays.
- ✓ Conocer las operaciones posibles a realizar sobre un array.
- ✓ Aprender alguno de los métodos de búsqueda y ordenación de estructuras de datos de tipo array.

Veamos antes de trabajar con ellas algunas definiciones con las que nos debemos familiarizar:

ARRAY.- estructura de datos constituida por un número fijo de elementos, todos ellos del mismo tipo y ubicados en direcciones de memoria físicamente contiguas.

ELEMENTOS DEL ARRAY.- también denominado componente, es cada uno de los datos que forman parte integrante de la array.

NOMBRE DE UNA ARRAY .- identificador utilizado para referenciar el array y los elementos que lo forman.

TIPO DE DATO DE UN ARRAY .- determina el tipo de dato que es común a todos y cada uno de los elementos o componentes que forman dicho array (numérico, carácter, etc.).

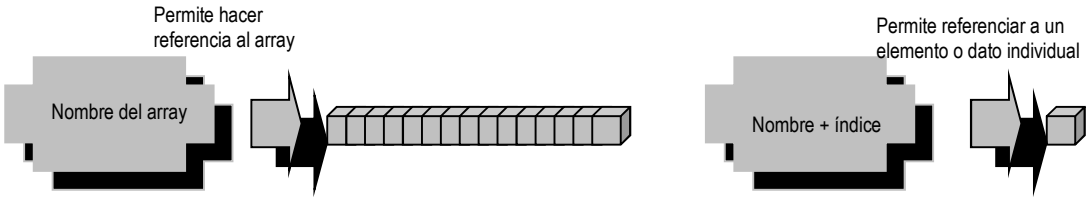
ÍNDICE .- normalmente es un valor numérico entero y positivo a través del cual podemos acceder directa e individualmente a los distintos elementos de un array, pues marca la situación relativa de cada elemento o componente dentro del mismo. En general, se puede utilizar como índice de un array un tipo ordinal (enteros y caracteres). Los tipos ordinales son aquellos que dado un elemento del dominio de ese tipo se encuentra definido su sucesor y/o predecesor.

TAMAÑO DE UNA ARRAY .- el tamaño o longitud de un array viene determinado por el número máximo de elementos que la forman, siendo el tamaño mínimo un elemento, y el tamaño máximo n elementos ($n > 0$).

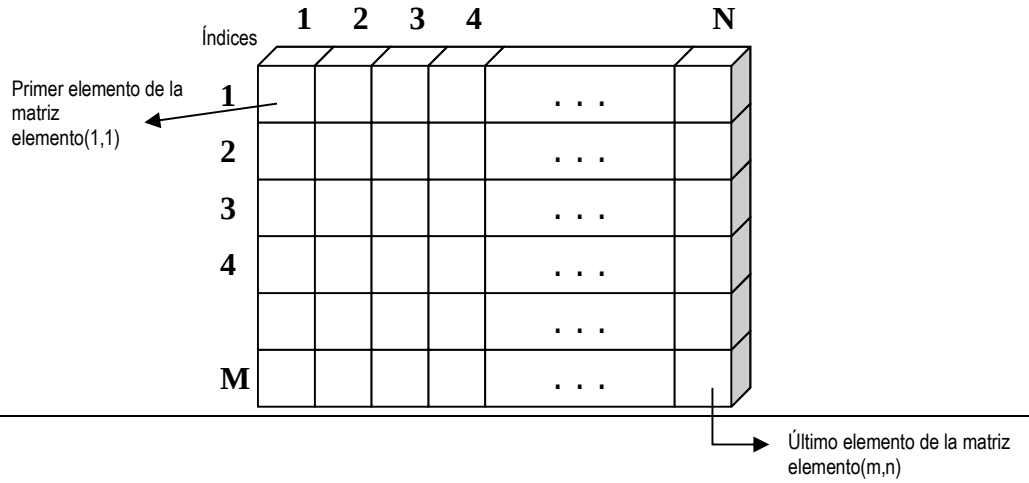
ACCESO A LOS ELEMENTOS O COMPONENTES DE UN ARRAY (ÍNDICE) .- los elementos o componentes de un array tratados individualmente son auténticos datos básicos del tipo correspondiente definidos para dicho array, reciben el mismo trato que cualquier otra variable de ese tipo, y tienen una denominación propia que les distingue del resto de elementos o componentes que constituyen dicha estructura de datos. Para acceder o referenciar un elemento en particular es suficiente con indicar el nombre del array seguido del índice (posición relativa que ocupa dicho elemento dentro del array) correspondiente a dicho elemento entre corchetes: nombre_array[indice].

DIMENSIÓN DEL ARRAY.- viene determinada por el número de índices que necesitamos para acceder a cualquiera de los elementos que lo forman.

Existen arrays de una dimensión (unidimensionales).



Y arrays de más de una dimensión. Como ejemplo de array n-dimensional podemos ver un array de 2 dimensiones. En este caso particular de 2 dimensiones se le denomina **matriz** o array bidimensional.



Ejercicio Resuelto 1. **Realiza un programa en C que lea 5 números reales y calcule su media, varianza y desviación típica. Media= ($\sum x_i$) / n Var = $\sum (x_i$ _ media)^2 Desv = var$^{(1/2)}$**

Para resolver este problema hay que almacenar los números en un array. La razón es la siguiente: cuando se calcula la media de una serie de números no es necesario almacenarlos, se pueden ir leyendo uno a uno y acumular su suma para luego calcular la media. Pero en este caso, sí es necesario guardar los números para poder calcular la correspondiente desviación respecto de la media de cada número.

```
#include <iostream>
#include <cmath>
using namespace std;

// declaración de la constante con el tamaño del vector de números reales
 const int TAM = 5;

// declaración de un nuevo tipo TVector
  typedef float TVector[TAM];

// cuerpo principal del algoritmo

int main()
{
   float   media, var, desv, suma;
   int     i;
   TVector vector;
   // inicializa la variable que acumula la suma de los cinco números
   suma = 0.0;

   // lee los cinco números y calcula su suma
   for (i=0; i < TAM; i++) {
      cout << "Introduce un numero: ";
      cin >> vector[i];
      suma = suma + vector[i];
   }

   // calcula y escribe la media
   media = suma / TAM;
   cout << "La media es " << media << endl;

   // calcula y escribe la varianza
   suma = 0;
   for (i=0; i < TAM; i++) {
      suma = suma + pow(vector[i] - media,2);
   }
   var = suma;
   cout << "la varianza es " << var << endl;

   // calcula y escribe la desviación típica
   desv = pow(var,1/2.0);
   cout << "la desviacion tipica es " << desv << endl;
}
```

Ejercicio Resuelto 2. **Realiza un programa en C en el que leamos un vector de enteros, obligando a que todos los enteros introducidos sean positivos. Luego ordenaremos el vector en modo ascendente y lo mostraremos ordenado. Tendremos un subprograma para leer el vector, otro para ordenarlo, y otro para imprimirlo.**

```
#include <iostream>
using namespace std;

// declaración de la constante con el tamaño del vector de números enteros
const int TAM = 10;

// declaración de un nuevo tipo TVector
typedef int TVector[TAM];

// Funciones cabecera
void leerVector(TVector vector);
void ordenaVector(TVector vector);
void imprimirVector(TVector vector);
;

// cuerpo principal del algoritmo
int main()
{
   TVector vector;
   leerVector(vector);
   ordenaVector(vector);
   imprimirVector(vector);
}

// Subprograma para leer el contentido del vector.
// Leeremos numeros enteros positivos
void leerVector(TVector vector)
{
    int i = 0;
    do {
       cout << "Introduzca el numero de la posición " << i << ": ";
       cin >> vector[i];
       if (vector[i] < 0)
          cout << "Error, debe introducir números positivos." << endl;
       else
          i++; // Pasamos a la siguiente posicion.
    } while (i < TAM);
}

// procedimiento que ordena el vector de forma ascendente.
void ordenaVector(TVector vector)
{
    int aux,i,j;
    for(i = 0; i < TAM-1; i++)
      for (j = (i+1); j<TAM ;j++)
       if(vector[j] < vector[i]) {
           aux = vector[i];
          vector[i] = vector[j];
          vector[j] = aux;
       }
}

// Imprime por pantalla los elementos del vector.
void imprimirVector(TVector vector)
{
    int i;

    cout << "Valores del vector: " << endl;
    for(i = 0; i < TAM; i++)
        cout << vector[i] << "  ";
    cout << endl;
}
```

Ejercicio Resuelto 3. **Realizar un programa en C con módulos que nos permitan leer los datos de un array bidimensional o matriz de 3 filas por 2 columnas y luego imprimir esta matriz por pantalla.**

```
#include <iostream>
using namespace std;

const int kFILAS = 3;
const int kCOLS  = 2;

typedef int TMatriz[kFILAS][kCOLS];

void leerMatriz(TMatriz matriz, int filas, int cols);
void escribirMatriz(TMatriz matriz, int filas, int cols);

int main()
{
    TMatriz matriz;

    leerMatriz(matriz, kFILAS, kCOLS);
    escribirMatriz(matriz, kFILAS, kCOLS);
}


// Procedimiento para leer los elementos de la matriz.
void leerMatriz(TMatriz matriz, int filas, int columnas)
{
   int i, j;

   for(i=0; i< filas; i++)
      for(j=0; j< columnas; j++) {
         cout << "Introduzca el elemento (" << i << "," << j << ") : ";
         cin >> matriz[i][j];
      }
}
// Procedimiento para imprimir la matriz.
void escribirMatriz(TMatriz matriz, int filas, int columnas)
{
   int i, j;

   for(i=0; i< filas; i++) {
      for(j=0; j< columnas; j++) {
         cout.width(5); // formatea salida rellenando con blancos según ancho indicado
         cout << matriz[i][j];
      }
      cout << endl;
   }
}
```

Ejercicio Resuelto 4. **Realiza un programa en C que dadas dos palabras, sustituya las letras minúsculas por mayúsculas, nos diga la longitud de cada palabra y nos indique cuál es la mayor en orden alfabético.**

```
#include <iostream>
using namespace std;

// declaración de la constante con el tamaño máximo de las palabras
const int TAM = 25;

// declaración de un nuevo tipo para las palabras
typedef char TPalabra[TAM];

// declaración de módulos
void Pasar_A_Mayusculas(char  cadena[]);

int main()
{
   TPalabra  palabra1, palabra2;
   int       compara;

   // leer las dos palabras introducidas por el usuario
   cout << "Introduce la primera palabra:";
   cin.getline(palabra1, TAM);
   cout << "Introduce la segunda palabra:";
   cin.getline(palabra2, TAM);

   // pasar a mayúsculas ambas palabras
   Pasar_A_Mayusculas(palabra1);
   Pasar_A_Mayusculas(palabra2);

   // mostrar la longitud de cada palabra
   cout << "la longitud de " << palabra1 << " es:" << strlen(palabra1) << endl;
   cout << "la longitud de " << palabra2 << " es:" << strlen(palabra2) << endl;

          // Comparar ambas palabras
   compara = strcmp(palabra1, palabra2);
   if (compara == 0)
      cout << "ambas palabras son iguales";
   else if (compara < 0)
      cout << palabra1 << " es menor que " << palabra2;
   else
      cout << palabra2 << " es menor que " << palabra1;

   cout << endl;
}

// Procedimiento que convierte una cadena de caracteres a mayúsculas
void Pasar_A_Mayusculas(char  cadena[])
{
   int  i;

   i = 0;
   while (cadena[i] != '\0') {
      cadena[i] = toupper(cadena[i]);
      i++;
   }
}
```

Ejercicio Resuelto 5. **Implementa un programa en C que solicite al usuario dos palabras y las compare según su orden alfabético. Para ello, primero se deberá validar que ambas palabras estén formadas exclusivamente por letras minúsculas ('a'..'z') o letras mayúsculas ('A'..'Z'). A continuación, se sustituirán en ambas palabras las letras mayúsculas por su correspondiente minúscula. Finalmente, se mostrarán por pantalla las 2 palabras y un mensaje indicando si son iguales, o cuál es la menor según el orden alfabético.**

```
#include <iostream>
using namespace std;

// Tamaño máximo de las cadenas
const int TAM = 20;

// Tipo de datos para las cadenas
typedef char TCadena[TAM];

bool esAlfabetica(TCadena);
void aMinusculas(TCadena);
void leerCadena(TCadena);

int main() {

      TCadena cad1, cad2;

      // Pide al usuario introducir dos cadenas
      leerCadena(cad1);
      leerCadena(cad2);

      // Convierte las cadenas a minúsculas
      aMinusculas(cad1);
      aMinusculas(cad2);

      // Muestra las cadenas en minúsculas
      cout << "Las dos palabras en minúsculas son:" << endl;
      cout << cad1 << endl << cad2 << endl;

      // Comprueba cuál es la menor cadena
      int result = strcmp(cad1,cad2);

      if(result<0) {
               cout << "La menor es " << cad1 << endl;
      } else if(result>0) {
               cout << "La menor es " << cad2 << endl;
      } else {
               cout << "Ambas cadenas con iguales" << endl;
      }
}

void leerCadena(TCadena cad) {
      do {
        cout << "Introduce una palabra: ";
        cin >> cad;

        if(!esAlfabetica(cad)) {
               cout << "Error, la palabra sólo debe contener letras" << endl;
        }
      } while(!esAlfabetica(cad));
}

bool esAlfabetica(TCadena cad) {
      bool alfabetica = true;
      int i=0;

      while(i<TAM && cad[i]!='\0') {
        if(!isalpha(cad[i])) {
               alfabetica = false;
               break;
        }
        i++;
      }

      return alfabetica;
}

void aMinusculas(TCadena cad) {
```

```
      int i=0;
      while(i<TAM && cad[i]!='\0') {
        cad[i] = tolower(cad[i]);
        i++;
      }
}
```

Ejemplo de ejecución:

Introduce una palabra: pizarra2
Error, la palabra sólo debe contener letras
Introduce una palabra: Pizarra
Introduce una palabra: boRRadOr
Las 2 palabras en minúsculas son:
pizarra
borrador
y la menor es: borrador