

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 el manejo de arrays bidimensionales.
- Aprender alguno de los métodos de búsqueda y ordenación más comunes en el tratamiento de información mediante el manejo de estructuras de datos de tipo array.

1. Arrays

En todas las prácticas realizadas hasta ahora hemos trabajado con tipos de datos simples como int, float o char. Estos datos sólo contienen un valor a la vez.

Sin embargo, **en esta práctica trabajaremos con tipos de datos estructurados**. Estos tipos de datos consisten en una colección de datos simples. Entre los tipos de datos estructurados encontramos arrays y registros. **Esta práctica se centra en arrays.**

 **El array es una estructura de datos en la que se almacena una colección de datos finita (debe determinarse el número de elementos), homogénea (todos los elementos deben ser del mismo tipo) y ordenada (se puede determinar cual es el n-esímo elemento del array).**

Los arrays se clasifican según su número de dimensiones en:

- Unidimensionales
- Bidimensionales
- Multidimensionales

Recordad que para referirse a un determinado elemento de un array deberemos utilizar un índice (encerrado entre corchetes) que especifique su posición relativa en el array. Por ejemplo, en un array unidimensional accederemos a un determinado elemento con `U[i]`, para un array bidimensional accederemos como `M[i][j]`.

🔗 **Como en cualquier tipo de variable, antes de utilizar un array debemos declararlo e inicializarlo.**

Un tipo especial de arrays unidimensionales son las cadenas de caracteres (string).

Para trabajar con cadenas de caracteres disponemos de una serie de funciones muy utilizadas:

Función	Descripción
<code>cin.getline(cadena, tamaño)</code>	Lectura de una cadena de caracteres por teclado hasta fin de línea.
<code>strcpy(cadena_destino, cadena_origen)</code>	Copia de cadenas.
<code>strcmp(cadena1, cadena2)</code>	Comparación alfabética de cadenas.
<code>strlen(cadena)</code>	Indica la longitud de la cadena

1.1 Definiciones

Antes de ponernos a trabajar con los tipos de datos estructurados debemos conocer algunas definiciones:

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 del array.

Nombre de un 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.

Índice

Es un valor numérico entero y positivo a través del cual podemos acceder directa e individualmente a los distintos elementos de un array.

Tamaño de un array

El tamaño o longitud del array viene determinado por el número máximo de elementos que la forman, siendo el tamaño mínimo un elemento.

2.1 Ejercicio resuelto 1. Repasa

- **Realiza un programa en C que lea 5 números reales y calcule su media y la desviación de cada número respecto a la media utilizando la fórmula**

$$d = x_i - \text{media}$$

2.1 Solución

Para resolver este problema es necesario que guardemos los números que se introducen por teclado en un array. Para calcular la media no es necesario almacenarlos porque podemos ir leyéndolos e ir acumulando la suma para después calcular la media. Pero en este caso, para poder calcular posteriormente la desviación respecto a la media debemos almacenar los números.

```
1  #include <iostream>
2  using namespace std;
3
4  //Constante con el tamaño del vector de números reales
5  const int TAM = 5;
6
7  //Declaración de un nuevo tipo TVector
8  typedef float TVector[TAM];
9
```

```
10  int main() {
11      TVector vector;
12      float  media, d, suma;
13      int     i;
14
15      //Inicializa la variable que acumula la suma de los
        cinco números
16      suma = 0.0;
17
18      //Lee los cinco números y calcula su suma
19      for (i=0; i < TAM; i++) {
20          cout << "Introduce un numero: ";
21          cin >> vector[i];
22          suma = suma + vector[i];
23      }
24
25      //Calcula y escribe la media
26      media = suma / TAM;
27      cout << "La media es " << media << endl;
28
29      //Calcula y escribe las desviaciones respecto de la
        media
30      for (i=0; i < TAM; i++) {
31          d = vector[i] - media;
32          cout << "numero " << vector[i] << "      d = " << d <<
        endl;
33      }
34  }
```

3. Ejercicio resuelto 2. Repasa

- Realiza un programa en C que lea un vector de enteros, obligando a que todos los números sean positivos. Posteriormente debe encontrar el elemento mayor.

Para la realización del programa utilizaremos tres módulos, uno para leer el vector, otro para imprimir el vector por pantalla y otro para calcular el elemento mayor.

3.1 Solución

```
1  #include <iostream>
2  using namespace std;
3
4  //Constante con el tamaño del vector de números enteros
5  const int TAM = 10;
6
7  //Declaración de un nuevo tipo TVector
8  typedef int TVector[TAM];
9
10 // Funciones cabecera
11 void leerVector(TVector vector);
12 void imprimirVector(TVector vector);
13 int mayorVector(TVector vector);
14
15 int main() {
16     TVector vector;
17     int mayor;
18
19     leerVector(vector);
20     imprimirVector(vector);
21     mayor = mayorVector(vector);
22
23     cout << "El elemento mayor del vector es : " << mayor <<
        endl;
24 }
25
26 //Función para leer el contenido del vector.
27 //Leeremos números enteros positivos
28 void leerVector(TVector vector) {
29     int i;
30     i = 0;
31     do {
32         cout << "Introduzca el numero de la posición " << i
            << ": ";
33         cin >> vector[i];
34         if (vector[i] < 0)
```

```
35         cout << "Error, debe introducir números  
    positivos." << endl;  
36     else  
37         i++; // Pasamos a la siguiente posición.  
38     } while (i < TAM);  
39 }
```

```
//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;
```

```
}
```

```
//Función que devuelve el mayor número del vector.
```

```
int mayorVector(TVector vector)
```

```
{
```

```
    int i, mayor;
```

```
    mayor = vector[0]; // Al principio el mayor será el  
    primer elemento.
```

```
    for(i = 1; i < TAM; i++)
```

```
        if(vector[i] > mayor)
```

```
            mayor = vector[i];
```

```
    return mayor;
```

```
}
```

4. Ejercicio resuelto 3. Repasa

- Escribe un programa en C que pida una cadena de caracteres (de máximo 15 caracteres) y devuelva la cadena escrita al revés.

4.1 Solución

Para resolver este problema podemos utilizar la función `strlen()` que nos indica la longitud de una cadena. Para utilizar esta función debemos incluir la librería `string.h`

```
1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  //Constante que utilizaremos para definir el tamaño de los
   arrays
6  const int TAM=15;
7  //Tipo de datos para definir las palabras que vamos a
   utilizar
8  typedef char TPalabra[TAM];
9
10 main(){
11     TPalabra palabra, palabra_auxiliar;
12     int i, j, longitud_palabra;
13
14     cout<<"Introduzca una palabra:";
15     cin.getline(palabra, TAM);
16
17     j = 0;
18     longitud_palabra = strlen(palabra);
19     for(i=longitud_palabra-1; i>=0; i--) {
20         palabra_auxiliar[j] = palabra[i];
21         j++;
22     }
23     palabra_auxiliar[j] = '\0';
24     cout<<"\nPalabra escrita al revés: " << palabra_auxiliar <<
        endl;
25 }
```

5. Ejercicio resuelto 4. Repasa

- Realiza un programa en C que lea dos palabras, sustituya las letras minúsculas por mayúsculas, nos diga la longitud de cada palabra y las muestre por pantalla en orden alfabético.
Las palabras las deberemos definir de un tamaño fijo, por ejemplo 25.

5.1 Solución

```
1  #include <iostream>
2  using namespace std;
3
4  //Constante con el tamaño máximo de las palabras
5  const int TAM = 25;
6
7  //Declaración de un nuevo tipo para las palabras
8  typedef char TPalabra[TAM];
9
10 //Declaración de módulos
11 void Pasar_A_Mayusculas(char  cadena[]);
12
13 int main() {
14     TPalabra  palabra1, palabra2;
15     int      compara;
16
17     //Lee las dos palabras introducidas por el usuario
18     cout << "Introduce la primera palabra:";
19     cin.getline(palabra1, TAM);
20     cout << "Introduce la segunda palabra:";
21     cin.getline(palabra2, TAM);
22
23     //Pasa a mayúsculas ambas palabras
24     Pasar_A_Mayusculas(palabra1);
25     Pasar_A_Mayusculas(palabra2);
26
27     //Muestra la longitud de cada palabra
28     cout << "La longitud de " << palabra1 << " es:" <<
```

```
    strlen(palabra1) << endl;
29     cout << "La longitud de " << palabra2 << " es:" <<
    strlen(palabra2) << endl;
30
31     //Compara ambas palabras y las muestra alfabéticamente
32     compara = strcmp(palabra1, palabra2);
33     if (compara == 0)
34         cout << "Ambas palabras son iguales";
35     else if (compara < 0)
36         cout << palabra1 << " - " << palabra2;
37     else
38         cout << palabra2 << " - " << palabra1;
39
40     cout << endl;
41 }
42
43 // Procedimiento que convierte una cadena de caracteres a
    mayúsculas
44 void Pasar_A_Mayusculas(char  cadena[]) {
45     int  i;
46
47     i = 0;
48     while (cadena[i] != '\0') {
49         cadena[i] = toupper(cadena[i]);
50         i++;
51     }
52 }
```

6. Ejercicio resuelto 5. Repasa

- Realizar un programa en C con módulos que nos permitan leer los datos de un array bidimensional o matriz de 5 filas por 4 columnas y luego imprima esta matriz por pantalla.

6.1 Solución

```
1  #include <iostream>
2  using namespace std;
3
4  //Constantes para especificar el tamaño de la matriz
5  const int kFILAS = 5;
```

```
6   const int kCOLS  = 4;
7
8   //Tipo de datos de la matriz
9   typedef int TMatriz[kFILAS][kCOLS];
10
11  //Declaración de los módulos
12  void leerMatriz(TMatriz matriz);
13  void escribirMatriz(TMatriz matriz);
14
15  int main() {
16      TMatriz matriz;
17
18      leerMatriz(matriz);
19      escribirMatriz(matriz);
20  }
21
22
23  // Procedimiento para leer los elementos de la matriz.
24  void leerMatriz(TMatriz matriz) {
25      int i, j;
26
27      for(i=0; i< kFILAS; i++)
28          for(j=0; j< kCOLS; j++) {
29              cout << "Introduzca el elemento (" << i << ", " <<
                j << "): ";
30              cin >> matriz[i][j];
31          }
32  }
33
34  // Procedimiento para imprimir la matriz.
35  void escribirMatriz(TMatriz matriz) {
36      int i, j;
37
38      for(i=0; i< kFILAS; i++) {
39          for(j=0; j< kCOLS; j++) {
40              cout.width(5); // formatea salida rellenando con
                blancos según ancho indicado
41              cout << matriz[i][j];
```

```
42     }  
43     cout << endl;  
44 }  
45 }
```

7. Ejercicio propuesto 1. Resuelve

- Realiza un programa que lea desde teclado una frase finalizada en punto. Tamaño máximo 80 caracteres. El programa deberá indicar cuantas mayúsculas y cuántas minúsculas se han introducido. Para ello, se deberá utilizar un módulo que devuelva el número de mayúsculas a partir del vector dado, y otro módulo que devuelva el número de minúsculas a partir del vector dado.

8. Ejercicio propuesto 2. Resuelve

- Escribir un programa que leyendo el día de la semana en que cae el primer día de un mes y el número de días que tiene ese mes, escriba el calendario de ese mes. Por ejemplo, si lee que el día de la semana es 3 (miércoles) y que el mes tiene 28 días, la respuesta que dé el programa será la siguiente: Utiliza para ello los módulos que creas adecuados:

```
MacBook-Pro-de-Rosana:pruebas rosanasatorre$ ./ej  
¿En qué día de la semana empieza el mes? 3  
¿Cuántos días tiene el mes? 28  
    1  2  3  4  5  
  6  7  8  9 10 11 12  
13 14 15 16 17 18 19  
20 21 22 23 24 25 26  
27 28  
MacBook-Pro-de-Rosana:pruebas rosanasatorre$
```

9. Ejercicio propuesto 3. Resuelve

- Escribe en C un programa que calcule la letra del DNI a partir del número de DNI. El programa deberá solicitar el número de DNI y mostrar la letra asociada a ese DNI. Es necesario calcular el resto de dividir el número de DNI entre 23.

La letra asociada al número vendrá dada por el valor del resto que podemos encontrar en la siguiente tabla:

T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

10. Ejercicio propuesto 4. Resuelve

- Implementa un programa en C que inicialmente visualice el siguiente menú:

```
MacBook-Pro-de-Rosana:pruebas rosanasatorre$ ./ej
1.- Introduce secuencia de números enteros
2.- Visualiza secuencia en orden descendente
3.- Buscar número
4.- Salir
Opción : █
```

Dependiendo de la opción de menú elegida, se deben realizar las siguientes acciones:

- **Opción 1:** se debe solicitar primero la cantidad de números a introducir y a continuación leer todos los números introducidos (almacenándolos en un array).
- **Opción 2:** se debe ordenar de forma descendente la secuencia de números (ordenar el array) y mostrar por pantalla dicha ordenación.
- **Opción 3:** se debe solicitar inicialmente un número a buscar, mostrándose por pantalla un mensaje que indique si dicho número está o no en la secuencia introducida y otro mensaje indicando el método de búsqueda utilizado.
- **Opción 4:** se debe finalizar la ejecución del programa.

Notas:

- Recuerda modularizar tu programa de la forma que creas más conveniente
- Declara una constante numérica en tu programa que establezca la cantidad máxima de números a introducir (por ejemplo, un máximo de 10 números)
- Puedes utilizar cualquier método de ordenación de tablas de los vistos en clase de teoría
- Implementa los métodos de búsqueda lineal y binaria, y utiliza el más eficiente cuando se elija la opción 3 del menú.

Ejemplo de ejecución:

```
// Se elige la opción 1...
```

```
Introduce la cantidad de números a introducir: 6
```

```
Introduce un número: 27
```

```
Introduce un número: 16
```

```
Introduce un número: 123
```

```
Introduce un número: 5
```

```
Introduce un número: 2003
```

```
Introduce un número: 98
```

```
// Se elige la opción 3...
```

```
Introduce el número que buscas: 5
```

```
El número buscado SI está en la lista
```

```
Se ha utilizado el método de búsqueda lineal
```

```
// Se elige la opción 2...
```

```
La lista de números ordenados de forma descendente es:
```

```
2003 ; 123 ; 98 ; 27 ; 16 ; 5
```

```
// Se elige la opción 3...
```

```
Introduce el número que buscas: 15  
El número buscado NO está en la lista  
Se ha utilizado el método de búsqueda binaria
```

```
// Se elige la opción 3...
```

```
Introduce el número que buscas: 16  
El número buscado SI está en la lista  
Se ha utilizado el método de búsqueda binaria
```

```
// Se elige la opción 4...
```

```
FIN DE LA EJECUCIÓN DEL PROGRAMA
```

11. Ejercicio propuesto 5. Resuelve

- Una sala de cine necesita que le diseñemos un programa en C para poder reservar los asientos de la sala. La sala se compone de 8 filas y cada una de ellas tiene 20 butacas. Del centro de la sala a la derecha encontramos las butacas pares, mientras que del centro a la izquierda encontramos los números impares.

Para realizar una reserva el programa deberá mostrar en primer lugar qué butacas están libres, y preguntar al cliente qué butacas desea. Cuando el cliente solicite una butaca, el programa deberá comprobar si esas butacas están libres, si lo están las reservará sino debe mostrar un mensaje de error.

Ejemplo de ejecución:

```

¿Quiere realizar una reserva?s
PATIO DE BUTACAS
19 17 15 13 11 9 7 5 3 1 2 4 6 8 10 12 14 16 18 20
1
2      * * * * *
3      * * *   * * * *   * * * *
4      * * * * * * * * * *
5 * * * * * * * * * * * * * * *
6      * * *           * * * * *
7      * * * * * * *   * * * *
8 * * * * * * *   * * *
¿Qué fila desea?2
¿Qué columna desea?15

Reserva realizada
¿Quiere realizar otra reserva?

```

12. Ejercicio Propuesto 6. Resuelve

- Se define como matriz zig-zag toda matriz en la que se van colocando los números en orden creciente conformando una especie de zigzag, por ejemplo:

1	6	7	12
2	5	8	11
3	4	9	10

Realizar un programa que solicite al usuario el número de filas y el número de columnas de la matriz y la rellene en zig-zag. A continuación debe mostrar la matriz resultante. La figura anterior representa una matriz de 3 filas y 4 columnas. Tanto el número de filas como el de columnas debe estar entre 1 y 15.

13. Errores de programación comunes

- **Olvidar que el primer índice en lenguaje C en un array es 0.** En muchas ocasiones se comete el error de empezar por 1.
- **Utilizar como índice un valor que no sea entero.** Los índices que se utilizan para acceder deben ser de tipo entero.
- **Acceder a una posición del array fuera de rango.** Cuando se accede a los elementos de un array no se hace ninguna comprobación. Suele ser un error frecuente acceder a posiciones que no pertenecen al array. Hay que asegurarse que siempre se utiliza como índice un valor comprendido entre 0 y *elementos-1*, siendo *elementos* el número de elementos del array.
- **Acceder a un array multidimensional de forma inadecuada.** Cuando se quiere, por ejemplo, acceder a la componente *i, j* de un array *M*, no se debe utilizar la expresión *M[i,j]* sino *M[i][j]*.
- **Olvidar contabilizar el carácter de fin de cadena de caracteres.** Cuando queramos reservar espacio para una cadena de caracteres hay que tener en cuenta que la cadena debe incluir al final el carácter nulo.
- **Usar comillas simples para representar cadenas de caracteres.** Las comillas simples se utilizan para representar caracteres individuales. Para las cadenas de caracteres se utilizan dobles comillas.
- **No cerrar bien una cadena de caracteres.** Nos debemos asegurar que las cadenas de caracteres utilizadas como valores constantes se encierran entre dobles comillas.
- **Copiar una cadena en otra que no tiene espacio suficiente.** Cuando queramos copiar una cadena en otra, nos tenemos que asegurar que la cadena destino tiene espacio suficiente para almacenar la cadena que se copia.