Temario:

- Arreglos estáticos unidimensionales
- Arreglos estáticos bidimensionales (matrices)
- Structs y arreglos de struct

Aclaración: se debe modularizar y reutilizar código cuando sea adecuado.

Ejercicios:

Ejercicio 1.

Identificar los errores, si los hay, de las siguientes declaraciones de arreglos. Para aquellos que son correctos especificar: la cantidad de elementos ocupados y los valores almacenados.

```
int a[4] = {8, 7, 6, 4, 3};
int b[] = {8, 7, 6, 4};
const int MAX = 4;
int c[MAX];
int d[5];
```

Ejercicio 2.

¿Qué salida produce el siguiente código?

```
int i, arreglo[10];
for (i = 0; i < 10; i++)
       arreglo[i] = 2*i;
for (i = 0; i < 10; i++)
       cout << arreglo[i] << " ";
cout << endl;
for (i = 0; i <= 10; i = i + 2)
       cout << arreglo[i] << " ";
```

Ejercicio 3.

Suponer que esperamos que los elementos de un arreglo estén ordenados de menor a mayor (de modo que cada elemento es menor o igual que el siguiente: a[0] <= a[1] <= a[2] ...)

Sin embargo, para estar seguros queremos probar el arreglo usando una función que informe si encuentra que algún elemento no está en orden (retornando su índice en caso de ser así, o retornando -1 si todo el arreglo está ordenado). Se supone que el siguiente código realiza esa prueba, pero contiene un error muy usual cuando se trabaja con arreglos, ¿en qué consiste?

```
int en_desorden (double arreglo[], int dimension)
    for (int indice = 0; indice < dimension; indice++) {
        if (a[indice] > a[indice + 1]) {
            return indice + 1;
    }
    return -1;
}
```

Ejercicio 4.

Cargar un arreglo con 20 números enteros ingresados por teclado de manera que el arreglo siempre se encuentre ordenado en forma ascendente. El arreglo se cargará por completo.
Hacer tres variantes, suponiendo que el usuario carga los datos de las siguientes formas:

a) El usuario ingresa los números en orden ascendente.
b) El usuario ingresa los números en orden descendente.
c) El usuario ingresa los números sin un orden en particular.

Ejercicio 5.

Implementar una función llamada *mas_uno* que tiene como parámetro un arreglo estático de enteros y que incrementa en uno el valor de cada elemento de dicho arreglo. Añadir otros parámetros que se necesiten.

Ejercicio 6.

Considerar la definición de función:

```
void dos(int a[], int cuantos) {
       for (int indice = 0; indice < cuantos; indice++) {
           a[indice] = 2;
       }
}
```

¿Cuales de las siguientes son llamadas de función aceptables?

```
int mi_arreglo[29];
dos(mi_arreglo, 29);
```

```
dos(mi_arreglo[], 29);
dos(mi_arreglo, 10);
dos(mi_arreglo, 55);
int tu_arreglo[100];
dos(tu_arreglo, 100);
dos(mi_arreglo[3], 29);
```

Ejercicio 7.

Dadas estas dos funciones y la descripción de cómo reciben los datos y cómo quedan al finalizar las funciones, ¿cuáles de los parámetros podrían declararse como constantes?.

```
int cargarNumeros(double arreglo[], int dimL) {
      //Precondición: el arreglo no tiene elementos útiles.
      //Poscondición: el arreglo contendrá los valores que haya deseado cargar el usuario.
}

void imprimir (int arreglo[], int dimL) {
      //Precondición: el arreglo contiene datos útiles.
      //Poscondición: se imprimieron todos los datos útiles del arreglo.
}
```

Ejercicio 8.

A pesar de que el programa a continuación no tiene errores de sintaxis, sí tiene varios otros problemas. ¿Cuáles son? ¿Cómo podrían corregirse?

```
const int MAX=100;
int cargar(float arreglo[], int dl){
    for (int i=0; i<MAX; i++){
        cout << "Número para almacenar: ";
        cin >> arreglo[i];
    }
    return dl;
}

int main(){
    float numeros[MAX];
```

```
    int cantidad = 0;
    cargar(numeros, cantidad);
}
```

Ejercicio 9.

Escribir una función int llamada *en_desorden* que reciba como parámetro un arreglo de tipo double y un int como dimensión lógica. Esta función determinará si el arreglo está en desorden, es decir, si viola la condición a[0] <= a[1] <= a[2] <= ...

La función devuelve -1 si los elementos no están en desorden; de lo contrario, devolverá el índice del primer elemento del arreglo que esté en desorden. Por ejemplo, considerá la declaración:

```
double a[10] = {1.2, 2.1, 3.3, 2.5, 4.5, 7.9, 5.4, 8.7, 9.9, 1.0};
```

En el arreglo anterior, a[2] y a[3] son el primer par que no está en orden. a[3] es el primer elemento en desorden, por lo que la función devuelve 3. Si el arreglo estuviera ordenado, la función devolvería -1.

Ejercicio 10.

Escribir una función llamada *cantidad_ocurrencias* que reciba como parámetro un arreglo de tipo int, un parámetro con la dimensión lógica del arreglo y un parámetro con un valor de tipo int. La función debe retornar la cantidad de ocurrencias del valor int encontradas en el arreglo.

Nota: como en todo algoritmo, se debe probar si la función hace lo esperado pasándole como argumentos diferentes casos posibles (por ejemplo: arreglo vacío, arreglo con un solo elemento, arreglo donde todos los elementos coinciden con el buscado, arreglo donde no se encuentra el elemento buscado, arreglo donde el elemento se encuentra al principio y al final, arreglo con repeticiones del elemento en posiciones consecutivas). Como ejemplo, se puede utilizar el siguiente programa para probar la función:

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

const int MAX = 10;

//Colocar acá la función cantidad_ocurrencias

int main() {
   int A[MAX] = {2,8,5,2,2,7,2,7,4,2};
   int B[MAX];
   int C[MAX] = {2};
   int D[MAX] = {2,2,2};
```

```
    int E[MAX] = {1,3,5,7,9};
    int dlA=10, dlB=0, dlC=1, dlD=3, dlE=5;
    cout << cantidad_ocurrencias(A, dlA, 2) << endl;
    cout << cantidad_ocurrencias(B, dlB, 2) << endl;
    cout << cantidad_ocurrencias(C, dlC, 2) << endl;
    cout << cantidad_ocurrencias(D, dlD, 2) << endl;
    cout << cantidad_ocurrencias(E, dlE, 2) << endl;
}
```

Ejercicio 11.

Escribir una función que, dado un arreglo de int y su dimensión lógica, invierta los elementos del arreglo.

Ejemplos: Si el arreglo contiene los elementos [1,2,3,4,5] deberá quedar [5,4,3,2,1].
Si el arreglo contiene [10,20] deberá quedar [20,10].
Si el arreglo contiene [5] deberá quedar [5].

Ejercicio 12.

Escribir un programa que permita al usuario gestionar los pagos de la cuota de un club.

a) Para ello, almacenar en un arreglo los números de DNI de los socios que pagaron. El club no tiene más de 200 socios.

b) Permitir al usuario buscar un DNI en el arreglo y, si el DNI está en el arreglo, imprimir "Cuota al día". Si no está, imprimir "Socio con deuda".

c) También se debe permitir al usuario eliminar un DNI del arreglo, en caso de haberlo ingresado erróneamente.

d) Finalmente, imprimir todo el arreglo.

Ejercicio 13.

Escribir un programa que permita al usuario ingresar números enteros positivos (los números ingresados no deben almacenarse). Al finalizar, informar la cantidad de veces que apareció cada dígito (del 0 al 9) en todos los números.

Ejemplo: Si el usuario ingresa los números 1156, 23, 73364, 988, 1003, 5, se debe informar que el 0 apareció 2 veces, el 1 apareció 3 veces, el 2 apareció 1 vez, el 3 apareció 4 veces, etc.

Ejercicio 14.

Escribir una variante del programa anterior en la que, en lugar de informar la cantidad de veces que apareció cada dígito, se informe si apareció o no cada dígito en todos los números ingresados. Analizar cómo puede resolverse este ejercicio evitando contabilizar la cantidad de dígitos.

Ejemplo: Si el usuario ingresa los números 44, 1, 710, 15, se debe informar que el 0 apareció, el 1 apareció, el 2 no apareció, el 3 no apareció, el 4 apareció, el 5 apareció, etc.

Ejercicio 15.

Dado un arreglo bidimensional de tipo float de 7 filas y 10 columnas, describir las sentencias para:

a) Asignar el valor 105 a la posición que se encuentra en la segunda fila, quinta columna.
b) Asignar en todas las posiciones de la cuarta fila el valor 1.5.
c) Imprimir todos los valores de la matriz, colocando cada fila en una nueva línea.
d) Permutar las columnas 3 y 5.
e) Sumar todos los elementos de las filas 2 y 6.
f) Imprimir todos los valores de la tercera columna.
g) Hallar en qué fila y columna se encuentra almacenado el mayor elemento.

Ejercicio 16.

Una matriz cuadrada A se dice que es simétrica si A(i,j) = A(j,i) para todo i, j. Escribir un programa que decida si una matriz de 3 x 3 dada es o no simétrica.

Ejercicio 17.

Dadas dos matrices de 4 x 4 leídas por teclado, informar la matriz resultante de la operación A + B.

Ejercicio 18.

Dado el siguiente programa, ¿qué hacen *funcion1* y *funcion2*? Indicar qué imprime el programa (sin ejecutarlo en máquina). Luego ejecutarlo para verificar si la salida en pantalla concuerda.

```
#include <iostream>

using namespace std;
const int N=3;
```

```
void funcion1(int matriz[][N]) {
   for (int i=0; i<N; i++) {
       for (int j=0; j<N; j++) {
           cout << matriz[i][j] << " ";
       }
       cout << endl;
   }
}

void funcion2(int m1[][N], int m2[][N], int resultado[][N]) {
   for(int i=0; i<N; i++) {
       for(int j=0; j<N; j++) {
           for(int k=0; k<N; k++) {
               resultado[i][j] += m1[i][k] * m2[k][j];
           }
       }
   }
}

int main() {
   int a[N][N] = { {2, 4, 1} , {2, 3, 9} , {3, 1, 8} };
   int b[N][N] = { {1, 2, 3} , {3, 6, 1} , {2, 4, 7} };
   int c[N][N] = { {0, 0, 0} , {0, 0, 0} , {0, 0, 0} };

   funcion1(a);
   cout << endl;
   funcion1(b);
   funcion2(a,b,c);
   cout << endl;
   funcion1(c);
}
```

Ejercicio 19.

Escribir un programa para ser utilizado en una ventanilla de venta de entradas de un pequeño teatro con capacidad para 50 personas. Las filas están identificadas de la 0 a la 4 y cada fila tiene asientos numerados de 1 a 10. Se debe registrar, por cada asiento, si está libre o no. Al iniciar el programa, todos los asientos están libres.

Cuando alguien desea comprar una entrada, se deben informar qué localidades están libres. Entonces el cliente deberá elegir su asiento (indicando fila y número de asiento) y éste se marcará como vendido. Si ya se vendieron todas las entradas, no se debe permitir elegir asiento.

Ejercicio 20.

Una oficina de servicio meteorológico desea informatizar la cantidad de lluvia que cayó en el año anterior (desde Enero hasta Diciembre). La información viene organizada por zonas: Mesopotamia, norte, centro, oeste y Patagonia. Utilizar una matriz como estructura principal para almacenar los datos de las lluvias. Utilizar también un arreglo de string para almacenar los nombres de las zonas que corresponden a cada índice en la matriz.

a) Permitir al usuario almacenar en la matriz los mililitros que llovió en cada mes, en cada una de las zonas.
b) Informar las precipitaciones totales en cada zona, en cada mes. Se debe mostrar nombre de la zona y cantidad de lluvia. Ejemplo:
   Mesopotamia:
   mes 1: 120 - mes 2: 230 - mes 3: 190 - mes 4: 250 ... etc.
c) Informar total y promedio de precipitaciones mensuales en todas las zonas. Ejemplo:
   Total del mes 1: 877 - Promedio del mes 1: 175.4
   Total del mes 2: 699 - Promedio del mes 2: 139.8
   ... etc.

Nota: una vez construida y verificada la función de carga del inciso a, es posible inicializar la matriz con datos de ejemplo, para efectuar pruebas evitando volver a cargar manualmente los datos en cada ejecución. Ejemplo:

```
{ {260, 25, 140, 80, 76, 84, 64, 79, 100, 130, 180, 220},
{96, 120, 89, 130, 76, 52, 66, 89, 118, 134, 146, 171},
{278, 300, 209, 160, 172, 102, 80, 98, 107, 150, 196, 242},
{152, 133, 96, 115, 94, 116, 78, 81, 80, 144, 138, 149},
{91, 121, 144, 162, 127, 103, 84, 73, 99, 108, 117, 102} };
```

Ejercicio 21.

El rey de un reino muy muy lejano está organizando festejos por su cumpleaños y desea ofrecer un banquete en cada una de las ciudades principales de su reino: Tuwin, Forella, Doriath y Aerilon. Los banquetes durarán 3 días en cada ciudad y se necesita contabilizar la cantidad de invitados que habrá cada día, para organizar todo.

a) Almacenar estos datos en una matriz, cargada por el usuario.
b) Informar cuántos invitados se esperan en total en cada ciudad (imprimiendo nombre de la ciudad y cantidad de invitados en el total de días).

c) Desde la ciudad Aerilon informan que habrá que restar 50 invitados que ya no asistirán el primer día.
d) Dado el nombre de una ciudad ingresada por teclado, informar la cantidad de personas que asistirán en cada uno de los tres días.

Ejercicio 22.

```
struct CuentaBancaria {
	double saldo;
	double tasa_interes;
	int plazo;
	char inicial1;
	char inicial2;
};

CuentaBancaria cuenta;
```

En los casos en que es posible indicarlo, ¿qué tipo tiene cada una de las siguientes variables? Indicar lo que no es correcto.

a) `cuenta.saldo`
b) `cuenta.tasa_interes`
c) `CuentaBancaria.plazo`
d) `cuenta_ahorros.inicial1`
e) `cuenta.inicial2`
f) `cuenta`

Ejercicio 23.

Considerando la siguiente definición de struct:

```
struct TipoZapato {
	string estilo;
	double precio;
};
```

¿Qué salidas producirá el siguiente código?

```
TipoZapato zapato1, zapato2;
zapato1.estilo = "Sandalia";
zapato1.precio = 9.99;
```

```
cout << zapato1.estilo << " $" << zapato1.precio << endl;
zapato2 = zapato1;
zapato2.precio = zapato2.precio / 9;
cout << zapato2.estilo << " $" << zapato2.precio << endl;
```

Ejercicio 24.

¿Qué error tiene el siguiente código? ¿Qué mensaje produce el compilador? ¿Cómo se podría solucionar?

```
bool mayoriaEdad(Persona p) {
    return p.edad >= 18;
}

int main() {
    struct Persona {
        string nombre;
        int edad;
    };

    Persona p;
    getline(cin>>ws, p.nombre);
    cin >> p.edad;
    if (mayoriaEdad(p))
        cout << "mayor de edad";
}
```

Ejercicio 25.

Realizar un algoritmo que almacene información de, como máximo, 500 libros en un arreglo estático. Un libro se define como un struct con los siguientes campos: titulo, autor, ISBN, editorial, cantidadHojas. El algoritmo finaliza luego de cargar el libro "El hobbit", el cual debe procesarse o cuando ya no quede espacio en el arreglo (lo que suceda primero). Finalmente, imprimir el arreglo resultante mostrando todos los datos de cada libro.

Ejercicio 26.

Escribir un programa que almacene en un arreglo los datos de los alumnos de una escuela luego de finalizado el año académico. Como máximo, puede haber 5000 alumnos. De cada alumno se debe guardar el nombre y la calificación obtenida en el primer, segundo y tercer trimestre del año académico. Imprimir el listado de nombres de alumnos mostrando un mensaje de "APTO" si el alumno supera o iguala la calificación de 5 para

todas las notas o "NO APTO" si no lo alcanza.
Utilizar un arreglo como campo del struct para almacenar las tres calificaciones.

Ejercicio 27.

La universidad posee información histórica sobre los estudiantes (no más de 1000) de una materia determinada. De cada uno almacena: nombre y apellido, legajo, cantidad de inasistencias a clase, calificación obtenida. Almacenar esta información en un arreglo de structs. Se pide:

a) Imprimir nombre y apellido de los alumnos que tuvieron más de 5 inasistencias.
b) Imprimir número de legajo de los alumnos cuya calificación promedio sea mayor o igual a la calificación promedio total (requiere calcular un promedio de las calificaciones de todos los alumnos en el listado).
c) Imprimir el número de legajo de aquellos alumnos que tienen promedio mayor o igual a nueve.
d) Dado el legajo de un alumno, eliminarlo del listado.

Nota: analizar si es posible reutilizar código en más de uno de los incisos.

Ejercicio 28.

Implementar un programa que almacene en un arreglo estático los datos de los ciudadanos que solicitaron un certificado de antecedentes en el Registro Nacional de las Personas. De cada ciudadano se conoce: nombre y apellido, fecha de nacimiento (tipo long: formato AAAAMMDD), dirección, número de documento, tipo de documento (tipo int: 1 -DNI-, 2 -Pasaporte-, 3 -Otro-) y sexo (tipo char: 'M', 'F').

a) Hacer un módulo de carga del arreglo, utilizando una condición de corte adecuada. Se debe validar el tipo de documento (1, 2 ó 3) y dejar al usuario en un bucle mientras ingrese un número inválido.
b) Imprimir el listado, con el siguiente formato de ejemplo:

Nombre y apellido: Juan Perez
Dirección: Levalle 132
Fecha de nacimiento: 27/10/1979
Tipo de documento: DNI
Número de documento: 27439221
Sexo: masculino

- La fecha debe mostrarse en formato dd/mm/aa. El tipo de documento debe mostrarse con su nombre. El género debe mostrar la palabra completa.

c) Implementar un módulo que, dado el arreglo original, cargue en otro arreglo los ciudadanos de sexo masculino únicamente. Imprimir este nuevo arreglo.
d) Hacer lo mismo que en el punto anterior pero obteniendo sólo las mujeres. Modularizar para no repetir código.

Ejercicio 29.

Implementar un sistema que permita administrar datos de clientes de un comercio para enviarles promociones. De cada cliente se tiene: nombre y apellido, DNI, e-mail, fecha de cumpleaños en formato DDMM, monto total de compras en el último año.

a) Realizar el módulo de carga. La misma finaliza con el nombre "z". Los clientes deben almacenarse ordenados por número de DNI.
b) Informar cuántos clientes cumplen años en cada uno de los 12 meses del año. Utilizar un arreglo auxiliar para contabilizarlos.
c) Informar el nombre del cliente con el mayor monto de compras.

Ejercicio 30.

Un municipio montará una exposición histórica sobre algunos protagonistas de las invasiones inglesas al Virreinato del Río de la Plata y requiere un sistema para organizar los artículos a exponer, que estarán divididos en dos salas diferentes: artículos de los invasores y artículos de los locales. No habrá más de 300 artículos en exposición. El programa debe mostrar al usuario un menú con opciones para realizar las siguientes operaciones repetidas veces:

a) Cargar datos de artículos en un arreglo estático. De cada artículo se tiene: descripción, protagonista al que pertenece, sala (“defensores” / “invasores”). Tener en cuenta los siguientes aspectos:
   - La sala no será ingresada por el usuario sino cargada automáticamente de acuerdo al protagonista:
   ● “Defensores” si el artículo pertenece a los protagonistas: Sobremonte, Pueyrredón ó Liniers.
   ● “Invasores” si el artículo pertenece a los protagonistas: Beresford, Popham ó Whitelocke.
   Precondición: el usuario sólo cargará artículos de los 6 protagonistas mencionados.
b) Leer de teclado una sala ("defensores" ó "invasores") e informar cantidad de artículos.
c) Dada una descripción de artículo, eliminar el artículo del arreglo, pero sólo si corresponde a los invasores. Informar si se realizó o no la eliminación.

Ejercicio 31.

Escribir un programa para gestionar las sucursales de una empresa.

a) Almacenar en un arreglo (máximo 2000 elementos) los datos de los empleados: legajo, nombre y DNI. En otro arreglo (máximo 20 elementos) almacenar la información de las sucursales: nombre de la sucursal y legajo del encargado. Cada vez que se ingrese un legajo del encargado de una sucursal, se debe validar si corresponde a un empleado existente. Si no existe, dejar al usuario en un bucle hasta que ingrese un número de legajo existente.

b) Dado el legajo de un empleado, informar su nombre.
c) Imprimir un listado de sucursales, mostrando nombre de sucursal y nombre del empleado encargado.

Nota: analizar si es posible reutilizar código en más de uno de los incisos.

Ejercicio 32.

En Junín se realizará la ronda preliminar del mundial de tango y es necesario registrar información de los participantes de la categoría de escenario. Escribir un programa que permita las siguientes operaciones:

a) Cargar los datos de parejas participantes en un arreglo (máximo 40 parejas), ordenado por número de inscripción. De cada pareja se almacenará: número de inscripción (int), nombre del integrante 1 (string), nombre del integrante 2 (string), nombre de la canción a usar en su presentación (string).
b) Cargar en otro arreglo (máximo 40 elementos) los datos de la música que se utilizará durante la competencia, almacenando de cada una: nombre de la canción (string), duración (expresada en segundos) (int).
   - Para la carga de los nombres de canciones (en ambos arreglos) se debe almacenar la primera letra en mayúscula y el resto en minúscula, independientemente de cómo lo haya ingresado el usuario. Nota: la función *toupper(char)* de C++ recibe un char y (si es una letra) lo retorna en mayúscula, así como la función *tolower(char)* recibe un char y lo retorna en minúscula.
c) Informar los números de parejas (puede haber más de una) que utilizan la canción de mayor duración.
d) Dado el nombre de una canción, eliminarla del arreglo, sólo si ninguna pareja la utiliza.
e) Dado el número de una pareja, indicar cuánto dura la canción que utilizará.