Objetivo de la práctica:
Declaración y utilización del tipo de dato estructurado array.

Conceptos básicos: vectores, matrices y array multidimensionales.

Un array es una secuencia de objetos del mismo tipo. Los objetos se llaman elementos del array y se numeran consecutivamente 0,1,2. El tipo de elementos almacenados en el array puede ser cualquier tipo de dato C++.

Adoptaremos el convenio de emplear **typedef** para crear este tipo de dato estructurado:

Vectores:

```
const int MAX = 20;
typedef int Vector20[MAX];  //define un vector de 20 enteros
Vector20  mi_vector; //declaro un vector de 20 enteros.
```

El acceso a los elementos de un vector se realiza mediante un índice. Los elementos están ordenados consecutivamente partiendo de 0. Ej. `mi_vector[3]=45;  //asigna el valor 45 a la cuarta componente`

matriz:

```
const int FIL = 10;
const int COL = 4;
typedef  int Matriz10x4[FIL][COL];
Matriz10x4  mi_matriz;
mi_matriz[1][3]=32;
```

Array multidimensional:

```
const int FILA1 = 10;
…..
const int FILAN =12;
typedef int Array_n_columnas[FILA1][FILA2]. . . .[FILAN];
Array_n_columnas   mi_array_n_columnas;
mi_array_n_columnas [1][2] . . .[10] = 20;
```

Los arrays **se pasan siempre por referencia** como argumentos de una función. La única diferencia con los tipos simples es que no se usa él '&' ya que basta simplemente el nombre del array.

EJERCICIOS

Utiliza los conocimientos aprendidos en las prácticas anteriores especialmente utilizando funciones.

(*)1. Realiza un programa que lea un vector de 10 elementos (enteros), aparezca un menú con las siguientes opciones:

1. Entrada del vector.
2. Visualización del vector.
3. Modificación de un elemento del vector.
4. Valor mínimo del vector.
5. Valor máximo del vector.
6. Suma de todos los elementos del vector.
7. Media de los elementos del vector.
8. Un elemento entero tecleado por pantalla está en el vector.

El vector está inicializado a 0.

(*)2. Escribir una función que permita calcular el cuadrado de los cien primeros números enteros y a continuación escriba una tabla que contenga dichos cien números.

3. Dados dos vectores {(a1,a2,a3) y (b1,b2,b3)}, calcular el producto escalar de ambos y su producto vectorial.

Producto escalar = Σ ai *bi y

Producto vectorial de dos vectores u(x,y,z) y v(x,y,z), obteniendo: w(x,y,z)

```
w[x] = u[y] * v[z] - u[z] * v[y];
w[y] = u[z] * v[x] - u[x] * v[z];
w[z] = u[x] * v[y] - u[y] * v[x]
```

(*)4. Realizar un programa que lea dos matrices de enteros de dimensión 5 x 5, y posteriormente devuelva la matriz producto de ellas.

Nota:

Producto A← m*p
B← p*n
C←m*n;
Donde Ci,j =ΣAik*Bkj desde k=1 hasta p.

(*)5. Escribir una función booleana en la que para cualquier matriz cuadrada, ya creada de números reales, podamos saber si es igual la suma de sus diagonales**.**

Ejemplo dada una matriz cuadrada 4x4 la suma de la diagonal (1,1)(2,2)(3,3)(4,4) coincide con (1,4)(2,3)(3,2)(3,1).

6. Un método para averiguar todos los números primos que hay entre el 1 y un entero N es el conocido como la Criba de Eratóstenes. Para ello, nos declaramos un vector de N booleanos al que inicializamos sus valores a cierto (true). Y procedemos de esta manera: partiendo del **índice** 2, cuyo contenido está a uno, voy multiplicando el **número** 2 por todos los números enteros hasta que el resultado sea mayor que N. Cada resultado de la multiplicación, **se usará como índice** para poner el elemento de la casilla a falso (false), pues ese número no puede ser primo. Acabado el 2, hacemos lo mismo con el 3, multiplicando por los siguientes enteros hasta que el resultado sea mayor que N y usando el resultado como índice del vector, cuyo contenido pondremos a falso.
Procedemos igual hasta que la multiplicación del entero que toca por su siguiente sea mayor que N.
Acabado este proceso los primos son los índices cuyo contenido se ha mantenido a cierto.
Implementa este método para calcular los primos existentes entre 1 y 1000. Para ello define una función a la que se le pase el vector y sea luego el programa principal quien lea el resultado.

Avanzado: Seguramente en tu implementación has hecho un recorrido sistemático por todos los elementos del vector buscando el siguiente primo para seguir aplicando la criba. Piensa que dado N, podemos saber el punto a partir del cual no vamos a poder encontrar más primos, viendo la propia dinámica de la criba. Escribe un algoritmo que se pare en dicho punto.

7. Dada una tabla de tipo real cuyas dimensiones son M, N leídas desde teclado, escribir un procedimiento que permita sumar el número de elementos positivos y el de negativos de una tabla T.

Aclaración.La matriz tiene de dimensión 10 x 9 , por teclado pedimos M y N que deben de ser inferiores al tamaño real de la matriz.

8. Implementa un programa que realice la suma de dos matrices enteras de dimensión 5 x 5. El programa debe pedir los componentes de las dos matrices, luego llamar a la función de suma, que devolverá el resultado en otra matriz. Luego se debe sacar la matriz resultado por pantalla.

(*)9. Implementa un programa que lea por teclado caracteres hasta que se pulse un carácter convenido que indique el final de la entrada, y devuelva el número de veces que se ha escrito cada una de las vocales.

Ej: entrada "2swe34ei rytoa2$"
Salida: 1 a, 2 e, 1 i, 1 o, 0 u (siendo '$' el carácter de final de entrada).
Utilizamos, cin ó cin.get(), tenemos que teclear salto de retorno y solamente cogerá la cadena de caracteres hasta que encuentre el primer centinela.

Para ello deberás crear una función a la que se le pase un array donde irás almacenando el número de ocurrencias de cada vocal. El programa principal debe leer ese array y sacar por pantalla el resultado.

Segunda parte

(*)10. Implementa un programa que declare un vector de 100 números enteros. El programa deberá entonces mostrar el siguiente menú:

1- Rellenar el vector con números aleatorios
2- Buscar un número con búsqueda lineal
3- Mostrar el vector
4- Salir

Los números aleatorios se deberán generar entre 0 y 1.000. Cada número se deberá insertar ordenado en el vector y podrán haber números repetidos. Para generar números aleatorios se utilizará la función `rand()` de la librería `stdlib.h`, que genera un número aleatorio entre 0 y RAND_MAX. Para generar números entre 0 y el valor que queramos se puede utilizar la siguiente expresión:

```
num_aleatorio = rand() * max / RAND_MAX
```

que genera números aleatorios entre 0 y max – 1.

(*)11. Realizar un programa que 'juegue' al Juego de la Vida. El Juego de la Vida consiste en una matriz con organismos distribuidos aleatoriamente. Los organismos se rigen por dos reglas:

1- Un organismo sobrevive si tiene entre 2 y 3 vecinos (llamamos vecinos a los organismos que están en las casillas adyacentes al organismo en cuestión, tanto adyacentes horizontal, vertical como diagonalmente).

2- Un organismo nace en una casilla desocupada, si esa casilla tiene tres vecinos.

En este juego utilizaremos una matriz de 8x8 casillas. Dibujaremos el tablero en pantalla utilizando el carácter * para representar un organismo. El programa dibujará el primer tablero y esperará a que se pulse una tecla para mostrar un nuevo ciclo de la vida.
NOTA: El calculo de la matriz correspondiente a un nuevo ciclo se realizará sobre una matriz auxiliar, y una vez acabado el cálculo se copiará sobre la matriz original.

Ejemplo:

Tablero inicial:

		*		*			
		*		*			
		*	*	*			

Primer ciclo de vida:

	*	*		*	*		
		*		*			
			*				