

Ejercicios del Tema 4

Vectores

1. Fibonacci

Escribe un programa que genere los 20 primeros términos de la serie de Fibonacci y los almacene en un vector.

```
#include <stdio.h>
#define N 20

int main ()
{
    int fib[N], i;
    fib[0] = 0;
    fib[1] = 1;
    //Genera los números y los almacena en el vector
    for (i = 2; i < N; ++i)
        fib[i] = fib[i-2] + fib[i-1];
    // Muestra el contenido del vector
    for (i = 0; i < N; ++i)
        printf("%i\n", fib[i]);
    return 0;
}
```

2. Producto escalar de vectores

Escribe un programa que realice el producto escalar de dos vectores. Use los dos vectores siguientes como ejemplo.

```
float v1[5] = {1, 34, 32, 45, 34};
float v2[5] = {12, -3, 34, 15, -5};
```

```
#include <stdio.h>
int main()
{
    float v1[5] = {1, 34, 32, 45, 34};
    float v2[5] = {12, -3, 34, 15, -5};
    float prod = 0;
    int i;

    for(i=0; i<5; i++)
        prod += v1[i] * v2[i];

    printf("El producto escalar es: %f\n", prod);

    return 0;
}
```

3. Distancia

- Escribe un programa que calcule la distancia entre dos puntos del espacio utilizando vectores de dimensión 2 para codificar los puntos.

```
#include <stdio.h>
#include <math.h>

int main()
{
    float p1[2], p2[2];
    float d;

    printf("Primer punto\n");
    scanf("%f %f", &p1[0], &p1[1]);

    printf("Segundo punto\n");
    scanf("%f %f", &p2[0], &p2[1]);

    d = sqrt(pow(p1[0] - p2[0], 2) + pow(p1[1] - p2[1], 2));

    printf("La distancia entre los puntos es %f\n", d);

    return 0;
}
```

- Realiza otra versión de este programa empleando una función dist como la indicada en el siguiente prototipo:

```
float dist(float p1[], float p2[]);
```

```
#include <stdio.h>
#include <math.h>

float dist(float p1[], float p2[]);

int main()
{
    float p1[2], p2[2];
    float d;

    printf("Primer punto\n");
    scanf("%f %f", &p1[0], &p1[1]);

    printf("Segundo punto\n");
    scanf("%f %f", &p2[0], &p2[1]);

    d = dist(p1, p2);

    printf("La distancia entre los puntos es %f\n", d);

    return 0;
}

float dist(float p1[], float p2[])
{
    return sqrt(pow(p1[0] - p2[0], 2) + pow(p1[1] - p2[1], 2));
}
```

4. Números primos

Realiza un programa que calcule los números primos comprendidos entre el 1 y el 300 y los almacene en un vector.

Cuando se complete el cálculo el programa debe mostrar el contenido del vector.

```
#include <stdio.h>

#define N 300

int main()
{
    int p, // Numero a comprobar
        i, // Indice de bucles
        iPrimo = 2, // Indice del vector primos, que indica la primera
                    // posicion libre
    primos[N] = {2, 3}; // Vector de primos, inicializo los dos
                        // primeros, y el resto quedan a 0

    _Bool esPrimo; // Booleano para indicar si es primo o no

    // Empiezo comprobando desde el 5
    for (p = 5; p <= N; p++)
    { // Inicialmente es primo
        esPrimo = 1;
        i = 0;
        // Con while comprobamos el resto de la division del numero p
        // con todos los primos anteriores
        while(esPrimo && i < iPrimo)
        { // Si alguno de los restos es cero
            if (p % primos[i] == 0)
                // el numero p deja de ser primo, y salgo del while
                esPrimo = 0;
            // En caso contrario sigo avanzando
            i++;
        }
        // Si salgo del bucle sin haber encontrado un resto igual a
        // cero, esPrimo sigue valiendo 1, lo que indica que se mantiene
        // la condicion de ser primo
        if (esPrimo)
        { //y por tanto debo almacenar ese numero en el vector
            primos[iPrimo] = p;
            //y avanzar el indice del vector
            ++iPrimo;
        }
    }
    // Muestro el resultado en pantalla
    for (i = 0; i < iPrimo; i++)
        printf ("%i ", primos[i]);
    printf ("\n");
    return 0;
}
```

5. Máximo, mínimo y promedio de una colección de números

Escribe un programa que calcule y muestre en pantalla el máximo, mínimo y promedio de una colección de 10 valores de tipo entero que se introducen por teclado.

```
#include <stdio.h>
```

```

#define N 10

void main()
{

    int vector[N], i;
    int min, max;
    float media;

    printf("Escribe %d números enteros.\n", N);
    // Rellena los valores del vector
    for (i = 0; i < N; i++)
        scanf("%d", &vector[i]);

    // Máximo
    max = vector[0]; //El maximo inicial es el primer elemento
    // Busca en el vector algún valor mayor
    for (i = 1; i < N; i++)
        if (vector[i] > max) max = vector[i]; // y lo sustituye si lo encuentra

    // Mínimo
    min = vector[0];
    for (i = 1; i < N; i++)
        if (vector[i] < min) min = vector[i];

    // Promedio
    media = 0.0;

    for (i = 0; i < N; i++)
        media += vector[i];
    media = media / N;

    printf("El máximo es %d, el mínimo es %d, y el promedio es %02f.\n",
        max, min, media);
}

```

Escribe otra versión de este programa empleando tres funciones de dos argumentos: `x[]` es un vector de dimensión por determinar, y `n` es la dimensión del vector `x[]`, definidas según los siguientes prototipos:

```

int maximo(int x[], int n);
int minimo(int x[], int n);
float promedio(int x[], int n);

```

```

#include <stdio.h>

#define N 10

int maximo(int x[], int n);
int minimo(int x[], int n);
float promedio(int x[], int n);

void main()
{

    int vector[N], i;
    int min, max;
    float prom;

    printf("Escribe 10 números enteros.\n");
    // Rellena los valores del vector
    for (i = 0; i < N; i++)

```

```

        scanf("%d", &vector[i]);
    // Calcula el minimo, maximo y promedio usando las funciones
    min = minimo(vector, N);
    max = maximo(vector, N);
    prom = promedio(vector, N);

    printf("El máximo es %d, el mínimo es %d, y el promedio es %02f.\n",
        max, min, prom);
}

int maximo(int x[], int n)
{
    int i, mx = x[0]; //El maximo inicial es el primer elemento
    // Busca en el vector algún valor mayor
    for (i = 1; i < n; i++)
        if (x[i] > mx) mx = x[i]; // y lo sustituye si lo encuentra
    return mx;
}

int minimo(int x[], int n)
{
    int i, mn = x[0];

    for (i = 1; i < n; i++)
        if (x[i] < mn) mn = x[i];
    return mn;
}

float promedio(int x[], int n)
{
    int i;
    float media = 0.0;

    for (i = 0; i < n; i++)
        media += x[i];
    return media / n;
}

```

6. Ordenamiento de vectores

- Escribe un programa que ordene de menor a mayor los elementos de un vector de 5 elementos. Los valores del vector serán introducidos por teclado, y el vector ordenado será mostrado por pantalla.

```

#include <stdio.h>

#define N 5

int main()
{
    int datos[N];
    int i, j, aux;
    //Primero pedimos los datos
    for(i = 0; i < N; i++)
    {
        printf("Dato %d: ", i + 1);
        scanf("%d", &datos[i]);
    }
    //Ahora lo ordenamos
    for(i = 0; i < N - 1; i++)
        { // Comparamos cada elemento con el siguiente

```

```

    for(j = i + 1; j < N; j++)
    {
        if (datos[i] > datos[j])
        {
            //Si es mayor intercambiamos el contenido de los dos
            //elementos
            aux = datos[i]; //Necesitamos una variable auxiliar de
                            //almacenamiento temporal
            datos[i] = datos[j];
            datos[j] = aux;
        }
    }
}

//sacamos el vector por pantalla
printf("Vector: ");
for(i = 0; i < 5; i++)
    printf("%d ", datos[i]);
printf("\n");

return 0;
}

```

- Realiza una versión del programa usando una función que implemente el ordenamiento del vector.

```

#include <stdio.h>

#define N 5

void ordena(int v[], int n);

int main()
{
    int datos[N];
    int i;
    // Primero pedimos los datos
    for(i = 0; i < N; i++)
    {
        printf("Dato %d: ", i + 1);
        scanf("%d", &datos[i]);
    }
    //ahora ordenamos: el vector pasa por referencia, no necesita asignación.
    ordena(datos, N);
    //sacamos el vector por pantalla
    printf("Vector: ");
    for(i = 0; i < N; i++)
        printf("%d ", datos[i]);
    printf("\n");

    return 0;
}

void ordena(int v[], int n)
{
    int i, j, aux;
    for(i = 0; i < n - 1; i++)
    {
        for(j = i + 1; j < n; j++)
        {
            if (v[i] > v[j])
            {
                aux = v[i];
                v[i] = v[j];
                v[j] = aux;
            }
        }
    }
}

```

```
}  
  }  
}
```