DOCUMENT_HEADING

10 files

(file list disabled)

t9\ejercicio1.c

```c
/*
  Diseña una función recursiva en C que calcule el factorial de un número.

*/
#include <stdio.h>

int factorial(int n) {
    if(n == 0) { // Caso base
        return 1;
    } else { // Caso recursivo
        return n * factorial(n - 1);
    }
}

int main() {
    int n;

    printf("Introduce un número para calcular su factorial: ");
    scanf("%d", &n);

    if(n < 0) {
        printf("No se puede calcular el factorial de un número negativo\n");
    } else {
        printf("El factorial de %d es %d\n", n, factorial(n));
    }

    return 0;
}
```

t9\ejercicio10.c

```c
/*
  Implementar un programa en C que use una función recursiva para ordenar una
lista con la técnica Quicksort.

*/
#include <stdio.h>

void quicksort(int arr[], int izquierda, int derecha) {
    if(izquierda >= derecha) { // Caso base: ya está ordenado
        return;
    }

    int pivote = arr[(izquierda + derecha) / 2];
    int i = izquierda, j = derecha;

    while(i <= j) {
```

```
            while(arr[i] < pivote) {
                i++;
            }

            while(arr[j] > pivote) {
                j--;
            }

            if(i <= j) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                i++;
                j--;
            }
        }

        quicksort(arr, izquierda, j);
        quicksort(arr, i, derecha);
    }

    int main() {
        int arr[] = {5, 3, 8, 1, 6, 2, 7, 4};
        int n = sizeof(arr) / sizeof(arr[0]);

        printf("Array original: ");
        for(int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");

        quicksort(arr, 0, n - 1);

        printf("Array ordenado: ");
        for(int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");

        return 0;
    }
```

t9\ejercicio2.c

```
    /*
      Escribe un programa en C que incluya una función recursiva que calcule la
    potencia de un número.

    */
    #include <stdio.h>

    double potencia(double base, int exponente) {
        if(exponente == 0) { // Caso base
            return 1;
        } else if(exponente > 0) { // Caso positivo
            return base * potencia(base, exponente - 1);
        } else { // Caso negativo
```

```
        return 1 / base * potencia(base, exponente + 1);
    }
}

int main() {
    double base, resultado;
    int exponente;

    printf("Introduce la base: ");
    scanf("%lf", &base);

    printf("Introduce el exponente: ");
    scanf("%d", &exponente);

    resultado = potencia(base, exponente);

    printf("%lf elevado a %d es %lf\n", base, exponente, resultado);

    return 0;
}
```

t9\ejercicio3.c

```
/*
  Implementa un programa que incluya una función recursiva que resuelva la
raíz cuadrada.

*/
#include <stdio.h>
#include <math.h>

double raiz_cuadrada(double num, double tolerancia, double aproximacion) {
    double diferencia = fabs(aproximacion * aproximacion - num);

    if(diferencia <= tolerancia) { // Caso base
        return aproximacion;
    } else { // Caso recursivo
        double nueva_aproximacion = (aproximacion + num / aproximacion) /
2.0;
        return raiz_cuadrada(num, tolerancia, nueva_aproximacion);
    }
}

int main() {
    double num, tolerancia, resultado;

    printf("Introduce un número: ");
    scanf("%lf", &num);

    printf("Introduce la tolerancia: ");
    scanf("%lf", &tolerancia);

    resultado = raiz_cuadrada(num, tolerancia, 1.0);

    printf("La raíz cuadrada de %lf es %lf\n", num, resultado);

    return 0;
```

```
}
```

t9\ejercicio4.c

```
/*
  Implementa un programa en C que incluya una función que calcule la suma de los componentes de un vector.

*/
#include <stdio.h>

int suma_vector(int vector[], int tamano) {
    int suma = 0;

    for(int i = 0; i < tamano; i++) {
        suma += vector[i];
    }

    return suma;
}

int main() {
    int tamano;

    printf("Introduce el tamaño del vector: ");
    scanf("%d", &tamano);

    int vector[tamano];

    printf("Introduce los elementos del vector:\n");
    for(int i = 0; i < tamano; i++) {
        scanf("%d", &vector[i]);
    }

    int suma = suma_vector(vector, tamano);

    printf("La suma de los elementos del vector es: %d\n", suma);

    return 0;
}
```

t9\ejercicio5.c

```
/*
  Escribe una función recursiva en C que calcule el máximo común divisor de dos números.

*/
#include <stdio.h>

int mcd(int a, int b) {
    if(b == 0) { // Caso base
        return a;
    } else { // Caso recursivo
```

```
        return mcd(b, a % b);
    }
}

int main() {
    int a, b;

    printf("Introduce dos números enteros: ");
    scanf("%d %d", &a, &b);

    if(a < 0 || b < 0) {
        printf("Los números introducidos deben ser positivos\n");
    } else {
        printf("El MCD de %d y %d es %d\n", a, b, mcd(a, b));
    }

    return 0;
}
```

t9\ejercicio6.c

```
/*
  Implementa un programa que incluya una función recursiva que calcule la
serie de Fibonacci de un número.

*/
#include <stdio.h>

int fibonacci(int n) {
    if(n == 0) { // Caso base
        return 0;
    } else if(n == 1) { // Caso base
        return 1;
    } else { // Caso recursivo
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main() {
    int n;

    printf("Introduce un número: ");
    scanf("%d", &n);

    if(n < 0) {
        printf("El número debe ser positivo\n");
    } else {
        printf("El %d-ésimo número de la serie de Fibonacci es %d\n", n,
fibonacci(n));
    }

    return 0;
}
```

t9\ejercicio7.c

```c
/*
  Implementa un programa en C que incluya una función recursiva que calcule la división entera.

*/
#include <stdio.h>

int division_entera(int dividendo, int divisor) {
    if(dividendo < divisor) { // Caso base
        return 0;
    } else { // Caso recursivo
        return 1 + division_entera(dividendo - divisor, divisor);
    }
}

int main() {
    int dividendo, divisor;

    printf("Introduce el dividendo: ");
    scanf("%d", &dividendo);

    printf("Introduce el divisor: ");
    scanf("%d", &divisor);

    if(divisor == 0) {
        printf("El divisor no puede ser cero\n");
    } else {
        int resultado = division_entera(dividendo, divisor);
        printf("%d / %d = %d\n", dividendo, divisor, resultado);
    }

    return 0;
}
```

t9\ejercicio8.c

```c
/*
  Implementa un programa en C que incluya una función que realice una búsqueda dicotómica en un array.

*/
#include <stdio.h>

int busqueda_dicotomica(int arr[], int n, int clave) {
    int izquierda = 0, derecha = n - 1;

    while(izquierda <= derecha) {
        int medio = (izquierda + derecha) / 2;

        if(arr[medio] == clave) {
            return medio;
        } else if(arr[medio] < clave) {
            izquierda = medio + 1;
        } else {
            derecha = medio - 1;
        }
```

```
    }

    return -1; // Si no se encuentra la clave, se devuelve -1
}

int main() {
    int arr[] = {1, 3, 4, 6, 8, 9, 11, 15};
    int n = sizeof(arr) / sizeof(arr[0]);
    int clave;

    printf("Introduce la clave a buscar: ");
    scanf("%d", &clave);

    int indice = busqueda_dicotomica(arr, n, clave);

    if(indice == -1) {
        printf("La clave no se encuentra en el array\n");
    } else {
        printf("La clave se encuentra en el índice %d del array\n", indice);
    }

    return 0;
}
```

t9\ejercicio9.c

```
/*
  Implementa un programa en C que incluya una función recursiva que solucione
el problema de las torres de Hanoi.

*/
#include <stdio.h>

void hanoi(int n, char torre_origen, char torre_destino, char torre_auxiliar)
{
    if(n == 1) { // Caso base
        printf("Mover disco 1 de la torre %c a la torre %c\n", torre_origen,
torre_destino);
        return;
    }

    hanoi(n - 1, torre_origen, torre_auxiliar, torre_destino);
    printf("Mover disco %d de la torre %c a la torre %c\n", n, torre_origen,
torre_destino);
    hanoi(n - 1, torre_auxiliar, torre_destino, torre_origen);
}

int main() {
    int n;

    printf("Introduce el número de discos: ");
    scanf("%d", &n);

    if(n < 1) {
        printf("El número de discos debe ser mayor o igual a 1\n");
    } else {
        hanoi(n, 'A', 'C', 'B');
    }
```

```
    return 0;
}
```