ejercicio 1:

```c
#include <stdio.h>

#define N 6

int main() {
    // Declaración de variables
    int vector[N];

    // Ingreso de datos
    printf("Ingrese 6 elementos para el vector:\n");
    for (int i = 0; i < N; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &vector[i]);
    }

    // Mostrar contenido del vector
    printf("\nContenido del vector:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");

    return 0;
}
```

ejercicio 2:

```c
#include <stdio.h>

#define N 7

int main() {
    // Declaración de variables
    int vector[N];
    int mayor, posicion;

    // Ingreso de datos
    printf("Ingrese 7 números enteros positivos:\n");
    for (int i = 0; i < N; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &vector[i]);

        // Verificar si es el mayor hasta el momento
        if (i == 0 || vector[i] > mayor) {
            mayor = vector[i];
            posicion = i + 1;
        }
    }

    // Mostrar el vector completo
    printf("\nContenido del vector:\n");
```

```
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }

    // Mostrar el mayor y su posición
    printf("\n\nEl mayor elemento es %d y se encuentra en la posición %d.\n", mayor, posicion);

    return 0;
}
```

ejercicio 3:

```
#include <stdio.h>

#define MAX_COMPRAS 10

int main() {
    // Declaración de variables
    float gastos[MAX_COMPRAS];
    float total = 0, mayorGasto = 0;

    // Ingreso de datos
    int numCompras;
    do {
        printf("Ingrese el número de compras (máximo 10): ");
        scanf("%d", &numCompras);
    } while (numCompras <= 0 || numCompras > MAX_COMPRAS);

    printf("Ingrese los gastos en compras:\n");
    for (int i = 0; i < numCompras; i++) {
        printf("Compra %d: $", i + 1);
        scanf("%f", &gastos[i]);

        // Cálculo del total
        total += gastos[i];

        // Verificación del mayor gasto
        if (i == 0 || gastos[i] > mayorGasto) {
            mayorGasto = gastos[i];
        }
    }

    // Mostrar resultados
    printf("\nMonto total gastado: $%.2f\n", total);
    printf("El mayor gasto fue: $%.2f\n", mayorGasto);

    return 0;
}
```

ejercicio 4:

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

```c
#define N 8

int main() {
    // Declaración de variables
    int numeros[N];
    int num1, num2;

    // Ingreso de datos
    printf("Ingrese 8 números enteros:\n");
    for (int i = 0; i < N; i++) {
        printf("Número %d: ", i + 1);
        scanf("%d", &numeros[i]);
    }

    // Ingreso de dos números adicionales
    printf("\nIngrese dos números adicionales:\n");
    printf("Número 1: ");
    scanf("%d", &num1);
    printf("Número 2: ");
    scanf("%d", &num2);

    // Verificación de pertenencia y conteo
    int conteo1 = 0, conteo2 = 0;
    int posicion1 = -1, posicion2 = -1;

    for (int i = 0; i < N; i++) {
        if (numeros[i] == num1) {
            conteo1++;
            posicion1 = i + 1;
        }
        if (numeros[i] == num2) {
            conteo2++;
            posicion2 = i + 1;
        }
    }

    // Mostrar resultados
    if (conteo1 > 0) {
        printf("\n%d pertenece a la secuencia y aparece %d veces en la posición %d.\n", num1, conteo1,
posicion1);
    } else {
        printf("\n%d no pertenece a la secuencia.\n", num1);
    }

    if (conteo2 > 0) {
        printf("%d pertenece a la secuencia y aparece %d veces en la posición %d.\n", num2, conteo2,
posicion2);
    } else {
        printf("%d no pertenece a la secuencia.\n", num2);
    }
```

```
    return 0;
}
```

ejercicio 5:

```
#include <stdio.h>

#define MAX_NUMEROS 20

int main() {
    // Declaración de variables
    int numeros[MAX_NUMEROS];
    int pares[MAX_NUMEROS], impares[MAX_NUMEROS];
    int cantidadNumeros = 0, cantidadPares = 0, cantidadImpares = 0;
    int numero;

    // Ingreso de datos
    printf("Ingrese números enteros (0 para finalizar o máximo 20 números):\n");

    do {
        printf("Número %d: ", cantidadNumeros + 1);
        scanf("%d", &numero);

        // Verificar si es par o impar
        if (numero != 0) {
            numeros[cantidadNumeros] = numero;

            if (numero % 2 == 0) {
                pares[cantidadPares] = numero;
                cantidadPares++;
            } else {
                impares[cantidadImpares] = numero;
                cantidadImpares++;
            }

            cantidadNumeros++;
        }
    } while (numero != 0 && cantidadNumeros < MAX_NUMEROS);

    // Mostrar vectores
    printf("\nNúmeros pares ingresados:\n");
    for (int i = 0; i < cantidadPares; i++) {
        printf("%d ", pares[i]);
    }

    printf("\n\nNúmeros impares ingresados:\n");
    for (int i = 0; i < cantidadImpares; i++) {
        printf("%d ", impares[i]);
    }

    return 0;
}
```

ejercicio 6:

```c
#include <stdio.h>

#define TAMANO 10

int main() {
    // Declaración de variables
    int arreglo[TAMANO];
    int maximo, minimo;
    int ocurrenciasMaximo = 0, ocurrenciasMinimo = 0;

    // Ingreso de datos
    printf("Ingrese 10 números enteros para el arreglo:\n");
    for (int i = 0; i < TAMANO; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &arreglo[i]);

        // Inicializar maximo y minimo con el primer elemento
        if (i == 0) {
            maximo = arreglo[i];
            minimo = arreglo[i];
        } else {
            // Actualizar maximo y minimo según el valor ingresado
            if (arreglo[i] > maximo) {
                maximo = arreglo[i];
            } else if (arreglo[i] < minimo) {
                minimo = arreglo[i];
            }
        }
    }

    // Contar ocurrencias de maximo y minimo
    for (int i = 0; i < TAMANO; i++) {
        if (arreglo[i] == maximo) {
            ocurrenciasMaximo++;
        } else if (arreglo[i] == minimo) {
            ocurrenciasMinimo++;
        }
    }

    // Mostrar resultados
    printf("\nValor máximo: %d\n", maximo);
    printf("Número de ocurrencias del valor máximo: %d\n", ocurrenciasMaximo);
    printf("Valor mínimo: %d\n", minimo);
    printf("Número de ocurrencias del valor mínimo: %d\n", ocurrenciasMinimo);

    return 0;
}
```

ejercicio 7:

```
#include <stdio.h>

#define NUM_CORREDORES 10

int main() {
    // Declaración de variables
    float tiempos[NUM_CORREDORES];
    float tiempoMedio = 0;
    int primerPuesto, segundoPuesto, ultimoPuesto;

    // Ingreso de datos
    printf("Ingrese los tiempos de los corredores:\n");
    for (int i = 0; i < NUM_CORREDORES; i++) {
        printf("Tiempo del corredor %d: ", i + 1);
        scanf("%f", &tiempos[i]);
        tiempoMedio += tiempos[i];
    }

    // Cálculo del tiempo medio
    tiempoMedio /= NUM_CORREDORES;

    // Encontrar primer, segundo y último puesto
    primerPuesto = segundoPuesto = ultimoPuesto = 0;

    for (int i = 1; i < NUM_CORREDORES; i++) {
        if (tiempos[i] < tiempos[primerPuesto]) {
            segundoPuesto = primerPuesto;
            primerPuesto = i;
        } else if (tiempos[i] < tiempos[segundoPuesto]) {
            segundoPuesto = i;
        }

        if (tiempos[i] > tiempos[ultimoPuesto]) {
            ultimoPuesto = i;
        }
    }

    // Mostrar resultados
    printf("\nCorredor del primer puesto: Corredor %d con tiempo %.2f\n", primerPuesto + 1,
tiempos[primerPuesto]);
    printf("Corredor del segundo puesto: Corredor %d con tiempo %.2f\n", segundoPuesto + 1,
tiempos[segundoPuesto]);
    printf("Corredor del último puesto: Corredor %d con tiempo %.2f\n", ultimoPuesto + 1,
tiempos[ultimoPuesto]);
    printf("Tiempo medio de la carrera: %.2f\n", tiempoMedio);

    return 0;
}
```

ejercicio 8:

```c
#include <stdio.h>

#define N 7

// Función para intercambiar dos elementos de un vector
void intercambiar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Función para ordenar un vector de menor a mayor (Método de Burbuja)
void ordenarVector(int vector[], int tamano) {
    for (int i = 0; i < tamano - 1; i++) {
        for (int j = 0; j < tamano - i - 1; j++) {
            if (vector[j] > vector[j + 1]) {
                intercambiar(&vector[j], &vector[j + 1]);
            }
        }
    }
}

int main() {
    // Declaración de variables
    int vector[N];

    // Ingreso de datos
    printf("Ingrese 7 números enteros positivos:\n");
    for (int i = 0; i < N; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &vector[i]);
    }

    // Ordenar el vector
    ordenarVector(vector, N);

    // Mostrar el contenido del vector ordenado
    printf("\nContenido del vector ordenado de menor a mayor:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");

    return 0;
}
```

ejercicio 9:

```c
#include <stdio.h>

#define N 8
```

```
// Función para intercambiar dos elementos de un vector
void intercambiar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Función para ordenar un vector de mayor a menor (Método de Burbuja)
void ordenarVectorDescendente(int vector[], int tamano) {
    for (int i = 0; i < tamano - 1; i++) {
        for (int j = 0; j < tamano - i - 1; j++) {
            if (vector[j] < vector[j + 1]) {
                intercambiar(&vector[j], &vector[j + 1]);
            }
        }
    }
}

int main() {
    // Declaración de variables
    int vector[N];

    // Ingreso de datos
    printf("Ingrese 8 números enteros positivos:\n");
    for (int i = 0; i < N; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &vector[i]);
    }

    // Mostrar el vector tal como fue ingresado
    printf("\nVector ingresado:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }

    // Ordenar el vector en forma decreciente
    ordenarVectorDescendente(vector, N);

    // Mostrar el vector ordenado en forma decreciente
    printf("\nVector ordenado en forma decreciente:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");

    return 0;
}
```

ejercicio 10:

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

```c
#define N 8

// Función para intercambiar dos elementos de un vector
void intercambiar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Función para ordenar los números positivos de mayor a menor
void ordenarPositivosDescendente(int vector[], int tamano) {
    for (int i = 0; i < tamano - 1; i++) {
        for (int j = 0; j < tamano - i - 1; j++) {
            if (vector[j] < vector[j + 1]) {
                intercambiar(&vector[j], &vector[j + 1]);
            }
        }
    }
}

// Función para ordenar los números negativos de menor a mayor
void ordenarNegativosCreciente(int vector[], int tamano) {
    for (int i = 0; i < tamano - 1; i++) {
        for (int j = 0; j < tamano - i - 1; j++) {
            if (vector[j] > vector[j + 1]) {
                intercambiar(&vector[j], &vector[j + 1]);
            }
        }
    }
}

int main() {
    // Declaración de variables
    int vector[N];

    // Ingreso de datos
    printf("Ingrese 8 números enteros (distintos de cero, positivos y negativos):\n");
    for (int i = 0; i < N; i++) {
        do {
            printf("Elemento %d: ", i + 1);
            scanf("%d", &vector[i]);
        } while (vector[i] == 0);
    }

    // Mostrar el vector tal como fue ingresado
    printf("\nVector ingresado:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", vector[i]);
    }

    // Separar y ordenar positivos en forma decreciente
```

```
    printf("\nPositivos ordenados en forma decreciente:\n");
    int positivos[N], cantidadPositivos = 0;
    for (int i = 0; i < N; i++) {
        if (vector[i] > 0) {
            positivos[cantidadPositivos] = vector[i];
            cantidadPositivos++;
        }
    }
    ordenarPositivosDescendente(positivos, cantidadPositivos);
    for (int i = 0; i < cantidadPositivos; i++) {
        printf("%d ", positivos[i]);
    }

    // Separar y ordenar negativos en forma creciente
    printf("\nNegativos ordenados en forma creciente:\n");
    int negativos[N], cantidadNegativos = 0;
    for (int i = 0; i < N; i++) {
        if (vector[i] < 0) {
            negativos[cantidadNegativos] = vector[i];
            cantidadNegativos++;
        }
    }
    ordenarNegativosCreciente(negativos, cantidadNegativos);
    for (int i = 0; i < cantidadNegativos; i++) {
        printf("%d ", negativos[i]);
    }

    printf("\n");

    return 0;
}


ejercicio 11:
#include <stdio.h>
#include <string.h>

#define MAX_LONGITUD 100

int main() {
    // Declaración de variables
    char cadena[MAX_LONGITUD];

    // Ingreso de datos
    printf("Ingrese una cadena de caracteres: ");
    fgets(cadena, sizeof(cadena), stdin);

    // Eliminar el salto de línea del final, si está presente
    size_t longitud = strlen(cadena);
    if (longitud > 0 && cadena[longitud - 1] == '\n') {
        cadena[longitud - 1] = '\0';
    }
```

```
    // Invertir la cadena
    for (int i = 0, j = longitud - 1; i < j; i++, j--) {
        char temp = cadena[i];
        cadena[i] = cadena[j];
        cadena[j] = temp;
    }

    // Mostrar la cadena invertida
    printf("Cadena invertida: %s\n", cadena);

    return 0;
}
```

ejercicio 12:

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

#define MAX_LONGITUD 100

// Función para eliminar los espacios en blanco de una cadena
void eliminarEspacios(char cadena[]) {
    int i, j;
    for (i = 0, j = 0; i < strlen(cadena); i++) {
        if (!isspace(cadena[i])) {
            cadena[j++] = cadena[i];
        }
    }
    cadena[j] = '\0';
}

// Función para verificar si una cadena es un palíndromo
int esPalindromo(char cadena[]) {
    int longitud = strlen(cadena);
    for (int i = 0, j = longitud - 1; i < j; i++, j--) {
        if (cadena[i] != cadena[j]) {
            return 0; // No es un palíndromo
        }
    }
    return 1; // Es un palíndromo
}

int main() {
    // Declaración de variables
    char texto[MAX_LONGITUD];

    // Ingreso de datos
    printf("Ingrese un texto para verificar si es un palíndromo: ");
    fgets(texto, sizeof(texto), stdin);
```

```
    // Eliminar el salto de línea del final, si está presente
    size_t longitud = strlen(texto);
    if (longitud > 0 && texto[longitud - 1] == '\n') {
        texto[longitud - 1] = '\0';
    }

    // Convertir el texto a minúsculas y eliminar espacios
    for (int i = 0; i < longitud; i++) {
        texto[i] = tolower(texto[i]);
    }
    eliminarEspacios(texto);

    // Verificar si es un palíndromo
    if (esPalindromo(texto)) {
        printf("El texto ingresado es un palíndromo.\n");
    } else {
        printf("El texto ingresado no es un palíndromo.\n");
    }

    return 0;
}


ejercicio 13:
#include <stdio.h>
#include <ctype.h>

#define MAX_LONGITUD 100

int main() {
    // Declaración de variables
    char texto[MAX_LONGITUD];

    // Ingreso de datos
    printf("Ingrese un texto para invertir las letras: ");
    fgets(texto, sizeof(texto), stdin);

    // Eliminar el salto de línea del final, si está presente
    size_t longitud = strlen(texto);
    if (longitud > 0 && texto[longitud - 1] == '\n') {
        texto[longitud - 1] = '\0';
    }

    // Invertir las letras (mayúsculas a minúsculas y viceversa)
    for (int i = 0; i < longitud; i++) {
        if (isalpha(texto[i])) { // Verificar si es una letra
            if (isupper(texto[i])) { // Convertir mayúscula a minúscula
                texto[i] = tolower(texto[i]);
            } else if (islower(texto[i])) { // Convertir minúscula a mayúscula
                texto[i] = toupper(texto[i]);
            }
        }
```

```
    }

    // Mostrar el texto con letras invertidas
    printf("Texto con letras invertidas: %s\n", texto);

    return 0;
}
```

ejercicio 14:

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

#define NUM_ALUMNOS 3
#define NUM_MATERIAS 5

// Función para verificar y corregir la primera letra del apellido
void corregirMayuscula(char apellido[]) {
    if (!isupper(apellido[0])) {
        apellido[0] = toupper(apellido[0]);
    }
}

// Función para calcular el promedio de un alumno
float calcularPromedio(float notas[]) {
    float suma = 0;
    for (int i = 0; i < NUM_MATERIAS; i++) {
        suma += notas[i];
    }
    return suma / NUM_MATERIAS;
}

int main() {
    // Declaración de variables
    char apellidos[NUM_ALUMNOS][50];
    float notas[NUM_ALUMNOS][NUM_MATERIAS];

    // Ingreso de datos
    for (int i = 0; i < NUM_ALUMNOS; i++) {
        printf("Ingrese el apellido del alumno %d: ", i + 1);
        scanf("%s", apellidos[i]);

        // Verificar y corregir la primera letra del apellido
        corregirMayuscula(apellidos[i]);

        printf("Ingrese las notas de las 5 materias para el alumno %d:\n", i + 1);
        for (int j = 0; j < NUM_MATERIAS; j++) {
            printf("Nota de materia %d: ", j + 1);
            scanf("%f", &notas[i][j]);
        }
    }
```

```
    // Calcular y mostrar el promedio de cada alumno
    for (int i = 0; i < NUM_ALUMNOS; i++) {
        printf("\nAlumno %d - Apellido: %s\n", i + 1, apellidos[i]);
        printf("Promedio: %.2f\n", calcularPromedio(notas[i]));
    }

    // Calcular y mostrar el promedio del curso
    float promedioCurso = 0;
    for (int i = 0; i < NUM_ALUMNOS; i++) {
        promedioCurso += calcularPromedio(notas[i]);
    }
    promedioCurso /= NUM_ALUMNOS;

    printf("\nPromedio del curso: %.2f\n", promedioCurso);

    return 0;
}
```

ejercicio 15:

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

#define MAX_LONGITUD 100

// Función para verificar si un carácter es una vocal
int esVocal(char c) {
    c = tolower(c); // Convertir a minúscula para simplificar la verificación
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
}

int main() {
    // Declaración de variables
    char texto[MAX_LONGITUD];

    // Ingreso de datos
    printf("Ingrese un texto: ");
    fgets(texto, sizeof(texto), stdin);

    // Eliminar el salto de línea del final, si está presente
    size_t longitud = strlen(texto);
    if (longitud > 0 && texto[longitud - 1] == '\n') {
        texto[longitud - 1] = '\0';
    }

    // Inicializar contadores de vocales
    int contadorA = 0, contadorE = 0, contadorI = 0, contadorO = 0, contadorU = 0;

    // Contar la cantidad de veces que se repite cada vocal en el texto
    for (int i = 0; i < longitud; i++) {
        if (esVocal(texto[i])) {
            switch (tolower(texto[i])) {
```

```
            case 'a':
                contadorA++;
                break;
            case 'e':
                contadorE++;
                break;
            case 'i':
                contadorI++;
                break;
            case 'o':
                contadorO++;
                break;
            case 'u':
                contadorU++;
                break;
        }
    }
}

// Mostrar resultados
printf("\nCantidad de veces que se repite cada vocal:\n");
printf("A: %d\n", contadorA);
printf("E: %d\n", contadorE);
printf("I: %d\n", contadorI);
printf("O: %d\n", contadorO);
printf("U: %d\n", contadorU);

return 0;
}
```

ejercicio 16:

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

#define MAX_LONGITUD_NOMBRE 50

// Estructura para almacenar información del empleado
struct Empleado {
    char nombre[MAX_LONGITUD_NOMBRE];
    char fechaIngreso[11]; // Se asume que la fecha se ingresa en formato "dd/mm/yyyy"
    float sueldo;
};

// Función para comparar dos fechas (dd/mm/yyyy)
int compararFechas(const char fecha1[], const char fecha2[]) {
    int dia1, mes1, anio1;
    int dia2, mes2, anio2;

    sscanf(fecha1, "%d/%d/%d", &dia1, &mes1, &anio1);
    sscanf(fecha2, "%d/%d/%d", &dia2, &mes2, &anio2);
```

```
    if (anio1 < anio2) {
        return -1;
    } else if (anio1 > anio2) {
        return 1;
    } else {
        if (mes1 < mes2) {
            return -1;
        } else if (mes1 > mes2) {
            return 1;
        } else {
            if (dia1 < dia2) {
                return -1;
            } else if (dia1 > dia2) {
                return 1;
            } else {
                return 0; // Las fechas son iguales
            }
        }
    }
}

int main() {
    // Declaración de variables
    struct Empleado empleados[3];

    // Ingreso de datos
    for (int i = 0; i < 3; i++) {
        printf("Ingrese nombre del empleado %d: ", i + 1);
        scanf("%s", empleados[i].nombre);

        printf("Ingrese fecha de ingreso del empleado %d (formato dd/mm/yyyy): ", i + 1);
        scanf("%s", empleados[i].fechaIngreso);

        printf("Ingrese sueldo del empleado %d: ", i + 1);
        scanf("%f", &empleados[i].sueldo);
    }

    // Encontrar el empleado más antiguo
    int indiceEmpleadoMasAntiguo = 0;
    for (int i = 1; i < 3; i++) {
        if (compararFechas(empleados[i].fechaIngreso,
empleados[indiceEmpleadoMasAntiguo].fechaIngreso) < 0) {
            indiceEmpleadoMasAntiguo = i;
        }
    }

    // Mostrar el nombre y sueldo del empleado más antiguo
    printf("\nEmpleado más antiguo:\n");
    printf("Nombre: %s\n", empleados[indiceEmpleadoMasAntiguo].nombre);
    printf("Sueldo: %.2f\n", empleados[indiceEmpleadoMasAntiguo].sueldo);

    return 0;
```

```
}
```

ejercicio 17:

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

#define MAX_LONGITUD_NOMBRE 50

// Estructura para almacenar información de una persona
struct Persona {
    char nombre[MAX_LONGITUD_NOMBRE];
    int dia, mes, anio; // Fecha de nacimiento (dd, mm, yyyy)
};

// Función para obtener la fecha actual
void obtenerFechaActual(int *dia, int *mes, int *anio) {
    time_t tiempo;
    struct tm *tiempoLocal;

    tiempo = time(NULL);
    tiempoLocal = localtime(&tiempo);

    *dia = tiempoLocal->tm_mday;
    *mes = tiempoLocal->tm_mon + 1; // El mes en la estructura tm comienza desde 0
    *anio = tiempoLocal->tm_year + 1900; // El año en la estructura tm es el año desde 1900
}

// Función para verificar y cargar la fecha de nacimiento
void cargarFechaNacimiento(struct Persona *persona) {
    int diaActual, mesActual, anioActual;
    obtenerFechaActual(&diaActual, &mesActual, &anioActual);

    do {
        printf("Ingrese la fecha de nacimiento de %s (formato dd mm yyyy): ", persona->nombre);
        scanf("%d %d %d", &persona->dia, &persona->mes, &persona->anio);

        if (persona->anio > anioActual ||
            (persona->anio == anioActual && persona->mes > mesActual) ||
            (persona->anio == anioActual && persona->mes == mesActual && persona->dia > diaActual)) {
            printf("Error: La fecha de nacimiento no puede ser posterior a la fecha actual. Intente
nuevamente.\n");
        }
    } while (persona->anio > anioActual ||
             (persona->anio == anioActual && persona->mes > mesActual) ||
             (persona->anio == anioActual && persona->mes == mesActual && persona->dia > diaActual));
}

// Función para calcular la edad a partir de la fecha de nacimiento
int calcularEdad(struct Persona *persona) {
    int diaActual, mesActual, anioActual;
```

```
    obtenerFechaActual(&diaActual, &mesActual, &anioActual);

    int edad = anioActual - persona->anio;
    if (mesActual < persona->mes || (mesActual == persona->mes && diaActual < persona->dia)) {
        edad--; // Aún no ha cumplido años en el presente año
    }

    return edad;
}

int main() {
    // Declaración de variables
    struct Persona personas[3];

    // Ingreso de datos
    for (int i = 0; i < 3; i++) {
        printf("Ingrese el nombre de la persona %d: ", i + 1);
        scanf("%s", personas[i].nombre);

        // Cargar la fecha de nacimiento verificando que sea válida
        cargarFechaNacimiento(&personas[i]);
    }

    // Modificación de la fecha a partir del nombre
    char nombreModificar[MAX_LONGITUD_NOMBRE];
    printf("\nIngrese el nombre de la persona para modificar la fecha: ");
    scanf("%s", nombreModificar);

    int indiceModificar = -1;
    for (int i = 0; i < 3; i++) {
        if (strcmp(personas[i].nombre, nombreModificar) == 0) {
            indiceModificar = i;
            break;
        }
    }

    if (indiceModificar != -1) {
        printf("Modificación de la fecha de nacimiento para %s:\n", nombreModificar);
        cargarFechaNacimiento(&personas[indiceModificar]);
    } else {
        printf("No se encontró a una persona con el nombre ingresado.\n");
    }

    // Mostrar los datos ingresados ordenados por edad
    printf("\nDatos ingresados ordenados por edad:\n");
    for (int i = 0; i < 3; i++) {
        printf("Nombre: %s, Fecha de Nacimiento: %02d/%02d/%04d, Edad: %d años\n",
               personas[i].nombre, personas[i].dia, personas[i].mes, personas[i].anio,
calcularEdad(&personas[i]));
    }

    return 0;
```

```
}
```