

Recopilación de Programas C++

Generado el 11/03/2025 a las 20:43:51

Programa: bloque5_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:44:27

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

// Mostrar el vector
cout << "\n■ Vector ingresado: [ ";
for(int i = 0; i < TAM; i++) {
    cout << numeros[i];
    if(i < TAM-1) cout << ", ";
}
cout << " ]\n";

// Mostrar el resultado con formato
cout << "\n■ Procesando suma...\n";
cout << "■ RESULTADO: La suma de todos los elementos es " << suma <<
"\n\n";

return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu - zsh - 80x24

DESCRIPCIÓN: Este programa almacena 5 números en un vector
              y calcula la suma de todos sus elementos.

Ingrese los valores del vector:
=====
Elemento [0]: 20
Elemento [1]: 20
Elemento [2]: 30
Elemento [3]: 30
Elemento [4]: 30

Vector ingresado: [ 20, 20, 30, 30, 30 ]

Procesando suma...
✓ RESULTADO: La suma de todos los elementos es 130

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque5_ejercicio2

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:44:51

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

        multiplicacion *= numeros[i]; // Acumulamos el producto de cada
elemento
    }

    // Mostrar el vector
    cout << "\n■ Vector ingresado: [ ";
    for(int i = 0; i < TAM; i++) {
        cout << numeros[i];
        if(i < TAM-1) cout << ", ";
    }
    cout << " ]\n";

    // Mostrar el cálculo paso a paso
    cout << "\n■ Cálculo: ";
    for(int i = 0; i < TAM; i++) {
        cout << numeros[i];
        if(i < TAM-1) cout << " x ";
    }
    cout << " = " << multiplicacion << "\n";

    // Mostrar el resultado con formato
    cout << "\n■ RESULTADO: La multiplicación de todos los elementos es "
<< multiplicacion << "\n\n";

    return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu — zsh — 80x24

DESCRIPCIÓN: Este programa almacena números en un vector
              y calcula el producto de todos ellos.

Ingrese los valores del vector:
=====
Elemento [0]: 230
Elemento [1]: 302
Elemento [2]: 302
Elemento [3]: 302
Elemento [4]: 320

Vector ingresado: [ 230, 302, 302, 302, 320 ]

Cálculo: 230 x 302 x 302 x 302 x 320 = -15014912

✓ RESULTADO: La multiplicación de todos los elementos es -15014912

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque5_ejercicio3

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:45:11

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

// Solicitar los elementos del vector
cout << "\n■ Ingrese los elementos del vector:\n";
cout << "=====\n";

for(int i = 0; i < n; i++) {
    cout << " Elemento [" << i << "]: ";
    cin >> numeros[i];
}

// Mostrar el vector con sus índices
cout << "\n■ VECTOR CON ÍNDICES:\n";
cout << "=====\n\n";

cout << "+-----+-----+\n";
cout << "|  ÍNDICE  |   VALOR   |\n";
cout << "+-----+-----+\n";

for(int i = 0; i < n; i++) {
    cout << " | " << setw(6) << i << " | " << setw(11) << numeros[i] <<
    " |\n";
}

cout << "+-----+-----+\n\n";
return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu - zsh - 80x24
Elemento [1]: 2
Elemento [2]: 3
Elemento [3]: 4
Elemento [4]: 5

■ VECTOR CON ÍNDICES:
=====

+-----+-----+
|  ÍNDICE  |   VALOR   |
+-----+-----+
|    0    |    1    |
|    1    |    2    |
|    2    |    3    |
|    3    |    4    |
|    4    |    5    |
+-----+-----+

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque5_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:45:26

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]


```

for(int i = 0; i < n; i++) {
    cout << " Elemento [" << i << "]: ";
    cin >> numeros[i];
}

// Mostrar el vector original
cout << "\n■ Vector original: [ ";
for(int i = 0; i < n; i++) {
    cout << numeros[i];
    if(i < n-1) cout << ", ";
}
cout << " ]\n";

// Mostrar el vector en orden inverso
cout << "\n■ Vector inverso: [ ";
for(int i = n-1; i >= 0; i--) {
    cout << numeros[i];
    if(i > 0) cout << ", ";
}
cout << " ]\n\n";

return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu - zsh - 80x24
VECTOR EN ORDEN INVERSO

DESCRIPCIÓN: Este programa almacena números en un vector
              y los muestra en orden inverso.

Ingrese el tamaño del vector (1-100): 3

Ingrese los elementos del vector:
=====
Elemento [0]: 1
Elemento [1]: 2
Elemento [2]: 3

Vector original: [ 1, 2, 3 ]

Vector inverso: [ 3, 2, 1 ]

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque5_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:45:26

Estado: Error en compilación

Error:

/Users/yoquelvisabreu/Desktop/c++/practica2/bloque5_ejercicio5.cpp:91:1: error: extraneous closing brace
(')'

91 | }

| ^

1 error generated.

Código Fuente:

[illegible]

```

        cout << "■■ Error: El tamaño debe estar entre 1 y " << MAX <<
        ".\n\n";
    } while(n <= 0 || n > MAX);

    // Solicitar los elementos del vector
    cout << "\n■ Ingrese los elementos del vector:\n";
    cout << "=====\n";

    for(int i = 0; i < n; i++) {
        cout << " Elemento [" << i << "]: ";
        cin >> numeros[i];

        // Verificar si el número actual es mayor
        if(numeros[i] > mayor) {
            mayor = numeros[i];
            posicion = i;
        }
    }

    // Mostrar el vector completo
    cout << "\n■ Vector ingresado: [ ";
    for(int i = 0; i < n; i++) {
        cout << numeros[i];
        if(i < n-1) cout << ", ";
    }
    cout << " ]\n";

    // Mostrar el resultado
    cout << "\n■ Analizando el vector...\n";
    cout << "■ RESULTADO: El mayor elemento es " << mayor << " (posición "
    << posicion << ")\n\n";

    return 0;
}

```

Programa: bloque6_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:45:45

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

        cin >> columnas;

        if(columnas <= 0 || columnas > MAX) {
            cout << "■■■ Error: El número de columnas debe estar entre 1 y " << MAX << ".\n\n";
        } while(columnas <= 0 || columnas > MAX);

        // Rellenar la matriz
        cout << "\n■ INGRESO DE DATOS DE LA MATRIZ (" << filas << "x" << columnas << "):\n";
        cout << "=====\n";

        for(int i = 0; i < filas; i++) {
            for(int j = 0; j < columnas; j++) {
                cout << "Elemento [" << i << "][" << j << "]: ";
                cin >> matriz[i][j];
            }
        }

        // Mostrar la matriz con formato mejorado
        cout << "\n■ MATRIZ RESULTANTE (" << filas << "x" << columnas << "):\n";
        cout << "=====\n\n";

        // Crear el encabezado de columnas
        cout << " ";
        for(int j = 0; j < columnas; j++) {
            cout << setw(5) << j << " ";
        }
        cout << "\n";

        // Línea separadora
        cout << " ";
        for(int j = 0; j < columnas; j++) {
            cout << "-----";
        }
        cout << "\n";

        // Mostrar los datos con índices de fila
        for(int i = 0; i < filas; i++) {
            cout << setw(2) << i << " | ";
            for(int j = 0; j < columnas; j++) {
                cout << setw(5) << matriz[i][j] << " ";
            }
            cout << "\n";
        }

        cout << "\n■ Matriz creada y mostrada con éxito.\n\n";
        return 0;
    }
}

```

Resultado de la Ejecución:



✎ Ingrese el número de filas (1-50): 2
✎ Ingrese el número de columnas (1-50): 1

📄 INGRESO DE DATOS DE LA MATRIZ (2x1):

=====

Elemento [0][0]: 1 1

Elemento [1][0]:

📄 MATRIZ RESULTANTE (2x1):

=====

	0
0	1
1	1

✅ Matriz creada y mostrada con éxito.

=====

Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque6_ejercicio2

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:46:09

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

cout << " ";
for(int j = 0; j < COLUMNAS; j++) {
    cout << setw(5) << j << " ";
}
cout << "\n ";

// Línea separadora
for(int j = 0; j < COLUMNAS; j++) {
    cout << "-----";
}
cout << "\n";

// Mostrar filas con índices
for(int i = 0; i < FILAS; i++) {
    cout << setw(2) << i << "| ";
    for(int j = 0; j < COLUMNAS; j++) {
        cout << setw(5) << matriz[i][j] << " ";
    }
    cout << "\n";
}

// Mostrar la diagonal principal con formato visual
cout << "\n■ DIAGONAL PRINCIPAL:\n";
cout << "=====\n\n";

for(int i = 0; i < FILAS; i++) {
    // Espacios en blanco antes del número (para visualizar la
    diagonal)
    for(int espacio = 0; espacio < i; espacio++) {
        cout << " ";
    }
    cout << "[" << matriz[i][i] << "]\n";
}


// Mostrar solo los elementos de la diagonal en forma de vector
cout << "\n■ Elementos de la diagonal: [ ";
for(int i = 0; i < FILAS; i++) {
    cout << matriz[i][i];
    if(i < FILAS - 1) cout << ", ";
}
cout << " ]\n\n";

return 0;
}

```


Resultado de la Ejecución:



 MATRIZ COMPLETA:

=====

	0	1	2
0	10	20	10
1	20	30	20
2	20	30	30


 DIAGONAL PRINCIPAL:

=====

[10]

[30]

[30]

 Elementos de la diagonal: [10, 30, 30]

=====

Ejecución completada

Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque6_ejercicio3

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:46:22

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

    cout << "■ DESCRIPCIÓN: Este programa crea una matriz de 2x2,\n";
    cout << "                                la llena con valores y copia su contenido a otra\n";
    cout << "matriz.\n\n";

    // Solicitar los elementos de la matriz
    cout << "■ INGRESO DE DATOS DE LA MATRIZ ORIGEN (2x2):\n";
    cout << "=====\n";

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            cout << " Elemento [" << i << "][" << j << "]: ";
            cin >> matriz1[i][j];
        }
    }

    // Mostrar matriz original
    cout << "\n";
    mostrarMatriz(matriz1, "■ MATRIZ ORIGINAL");

    // Copiar el contenido a la otra matriz
    cout << "■ Copiando matriz...\n\n";

    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            matriz2[i][j] = matriz1[i][j];
        }
    }

    // Mostrar la matriz copiada
    mostrarMatriz(matriz2, "■ MATRIZ COPIADA");

    // Mensaje de éxito
    cout << "■ Matriz copiada con éxito!\n\n";

    return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu — -zsh — 80x24

+-----+
0 |  20  20 |
1 |  20  20 |
+-----+

Copiando matriz...

■ MATRIZ COPIADA:
=====

    0      1
+-----+
0 |  20  20 |
1 |  20  20 |
+-----+

✓ Matriz copiada con éxito!

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque6_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:46:31

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 6: Ejercicio 4
 * Programa que crea una matriz preguntando al usuario el número de filas y
 * columnas,
 * la llena de números aleatorios, copia el contenido a otra matriz y la
 * muestra en pantalla.
 */
 * Autor: Yoquelyvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <cstdlib> // Para rand() y srand()
#include <ctime> // Para time()
using namespace std;

int main() {
    // Declaracion de variables
    int matriz1[50][50]; // Primera matriz
    int matriz2[50][50]; // Segunda matriz (copia)
    int filas, columnas;
    int i, j; // Variables para ciclos

    // Inicializar la semilla para los números aleatorios
    srand(time(NULL));

    // Pedir dimensiones de la matriz
    cout << "Digite el numero de filas: ";
    cin >> filas;

    cout << "Digite el numero de columnas: ";
    cin >> columnas;

    // Llenar la matriz con números aleatorios
    for(i = 0; i < filas; i++) {
        for(j = 0; j < columnas; j++) {
            matriz1[i][j] = rand() % 100; // Números aleatorios entre 0 y
99        }
    }

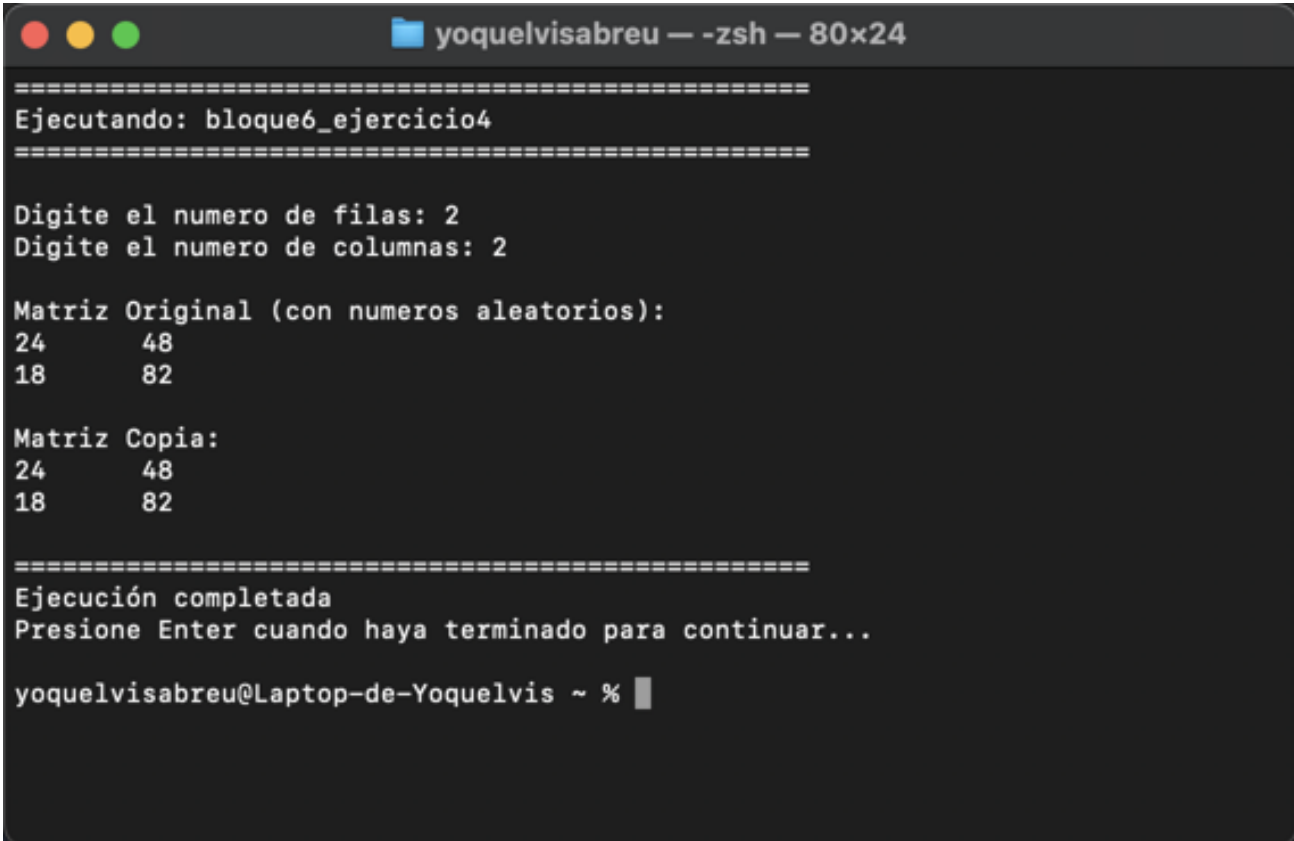
    // Copiar los datos a la segunda matriz
    for(i = 0; i < filas; i++) {
        for(j = 0; j < columnas; j++) {
            matriz2[i][j] = matriz1[i][j];
        }
    }

    // Mostrar la matriz original
    cout << "\nMatriz Original (con numeros aleatorios):\n";
    for(i = 0; i < filas; i++) {
        for(j = 0; j < columnas; j++) {
            cout << matriz1[i][j] << "\t";
        }
        cout << endl;
    }

    // Mostrar la matriz copia
    cout << "\nMatriz Copia:\n";
    for(i = 0; i < filas; i++) {
        for(j = 0; j < columnas; j++) {
            cout << matriz2[i][j] << "\t";
        }
    }
}
```

```
    cout << endl;  
}  
return 0;  
}
```

Resultado de la Ejecución:



A terminal window titled "yoquelvisabreu — -zsh — 80x24" displays the execution of a program. The output is as follows:

```
=====
Ejecutando: bloque6_ejercicio4
=====

Digite el numero de filas: 2
Digite el numero de columnas: 2

Matriz Original (con numeros aleatorios):
24      48
18      82

Matriz Copia:
24      48
18      82

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %
```

Programa: bloque6_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:46:48

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 6: Ejercicio 5
 * Programa que lee una matriz de 3x3 y crea su matriz traspuesta.
 * La matriz traspuesta es aquella en la que la columna i era la fila i
 * de la matriz original.
 * Autor: Yoquielvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
using namespace std;

int main() {
    // Declaración de variables
    int matriz[3][3]; // Matriz original
    int traspuesta[3][3]; // Matriz traspuesta
    int i, j; // Variables para ciclos

    // Pedir datos para la matriz
    cout << "Digite los elementos de la matriz 3x3:\n";
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            cout << "Digite el elemento [" << i << "][" << j << "]: ";
            cin >> matriz[i][j];
        }
    }

    // Crear la matriz traspuesta
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            traspuesta[j][i] = matriz[i][j];
        }
    }

    // Mostrar la matriz original
    cout << "\nMatriz Original:\n";
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            cout << matriz[i][j] << " ";
        }
        cout << endl;
    }

    // Mostrar la matriz traspuesta
    cout << "\nMatriz Traspuesta:\n";
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            cout << traspuesta[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Resultado de la Ejecución:



```
=====
Digite los elementos de la matriz 3x3:
Digite el elemento [0][0]: 30 30
Digite el elemento [0][1]: Digite el elemento [0][2]: 30 30
Digite el elemento [1][0]: Digite el elemento [1][1]: 30 30
Digite el elemento [1][2]: Digite el elemento [2][0]: 30 30
Digite el elemento [2][1]: Digite el elemento [2][2]: 30 30
```

Matriz Original:

```
30 30 30
30 30 30
30 30 30
```

Matriz Transpuesta:

```
30 30 30
30 30 30
30 30 30
```

```
=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
```

```
yoquelvisabreu@Laptop-de-Yoquelvis ~ %
```

Programa: bloque7_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:47:00

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 7: Ejercicio 1
 * Programa que pide al usuario una cadena de caracteres, verifica la
 * longitud de la cadena,
 * y si ésta supera a 10 caracteres la muestra en pantalla, caso contrario
 * no la muestra.
 */
* Autor: Yoquelyvis Abreu
* Fecha: Marzo 2024
*/

#include <iostream>
#include <string.h> // Para strlen()
using namespace std;

int main() {
    // Declaracion de variables
    char cadena[100]; // Arreglo para almacenar la cadena
    int longitud; // Para almacenar la longitud de la cadena

    // Pedir la cadena al usuario
    cout << "Digite una cadena de caracteres: ";
    cin.getline(cadena, 100, '\n');

    // Calcular la longitud de la cadena
    longitud = strlen(cadena);

    // Verificar si la longitud supera 10 caracteres
    if(longitud > 10) {
        cout << "\nLa cadena tiene " << longitud << " caracteres y es: " <<
cadena << endl;
    } else {
        cout << "\nLa cadena tiene " << longitud << " caracteres, no se
mostrara el contenido." << endl;
    }

    return 0;
}
```

Resultado de la Ejecución:



yoquelvisabreu — -zsh — 80x24

```
=====
Ejecutando: bloque7_ejercicio1
=====
```

```
Digite una cadena de caracteres: yoquelvis
```

```
La cadena tiene 9 caracteres, no se mostrara el contenido.
```

```
=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
```

```
yoquelvisabreu@Laptop-de-Yoquelvis ~ % █
```

Programa: bloque7_ejercicio2

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:47:20

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 7: Ejercicio 2
 * Programa que pide al usuario una cadena de caracteres, la almacena en un
 * arreglo
 * y copia todo su contenido hacia otro arreglo de caracteres.
 * Autor: Yoquielvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <string.h> // Para strcpy()
using namespace std;

int main() {
    // Declaracion de variables
    char cadenal[100]; // Arreglo original
    char cadena2[100]; // Arreglo de destino

    // Pedir la cadena al usuario
    cout << "Digite una cadena de caracteres: ";
    cin.getline(cadenal, 100, '\n');

    // Copiar el contenido de cadenal a cadena2
    strcpy(cadena2, cadenal);

    // Mostrar ambas cadenas para verificar
    cout << "\nCadena Original: " << cadenal << endl;
    cout << "Cadena Copiada: " << cadena2 << endl;

    return 0;
}
```

Resultado de la Ejecución:



```
=====
Ejecutando: bloque7_ejercicio2
=====
```

Digite una cadena de caracteres: yoquelvis1234

Cadena Original: yoquelvis1234

Cadena Copiada: yoquelvis1234

```
=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
```

yoquelvisabreu@Laptop-de-Yoquelvis ~ % █

Programa: bloque7_ejercicio3

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:47:41

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

    cout << "=====\n";
    cout << " Cadena 1: \"" << cadena1 << "\" (longitud: " <<
strlen(cadena1) << ")\n";
    cout << " Cadena 2: \"" << cadena2 << "\" (longitud: " <<
strlen(cadena2) << ")\n\n";

    // Comparar las cadenas
    cout << "■ COMPARANDO CADENAS...\n";
    cout << "=====\n";

    // Resultado de la comparación
    int resultado = strcmp(cadena1, cadena2);

    // Tabla de comparación
    cout << "\n+-----+-----+\n";
    cout << "| CRITERIO | RESULTADO | \n";
    cout << "+-----+-----+\n";

    // Fila 1: ¿Son iguales?
    cout << "| ¿Las cadenas son | ";
    if(resultado == 0) {
        cout << setw(20) << left << "■ SÍ son iguales" << " | \n";
    } else {
        cout << setw(20) << left << "■ NO son iguales" << " | \n";
    }

    // Fila 2: Orden alfabético
    if(resultado != 0) {
        cout << "| Orden alfabético | ";
        if(resultado > 0) {
            cout << setw(20) << left << "\"" + string(cadena2) + "\"
precede a \"" + string(cadena1) + "\" << " | \n";
        } else {
            cout << setw(20) << left << "\"" + string(cadena1) + "\"
precede a \"" + string(cadena2) + "\" << " | \n";
        }
    } else {
        cout << "| Orden alfabético | " << setw(20) << left << "No
aplica" << " | \n";
    }

    cout << "+-----+-----+\n\n";

    // Conclusión más clara
    cout << "■ CONCLUSIÓN: ";
    if(resultado == 0) {
        cout << "Ambas cadenas son exactamente iguales.\n\n";
    } else if(resultado > 0) {
        cout << "La cadena \"" << cadena2 << "\" es alfabéticamente mayor
que \"" << cadena1 << "\".\n\n";
    } else {
        cout << "La cadena \"" << cadena1 << "\" es alfabéticamente mayor
que \"" << cadena2 << "\".\n\n";
    }

    return 0;
}

```

Resultado de la Ejecución:



DATOS INGRESADOS:

=====

Cadena 1: "hola" (longitud: 4)

Cadena 2: "como estas" (longitud: 10)



COMPARANDO CADENAS...

=====

+-----+-----+	
CRITERIO	RESULTADO
+-----+-----+	
¿Las cadenas son	✗ NO son iguales
Orden alfabético	"como estas" precede a "hola"
+-----+-----+	



CONCLUSIÓN: La cadena "como estas" es alfabéticamente mayor que "hola".

=====

Ejecución completada

Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque7_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:47:51

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 7: Ejercicio 4
 * Programa que crea una cadena que tiene la frase "Hola que tal", luego
 * crea otra cadena
 * para preguntarle al usuario su nombre, por ultimo añade el nombre al
 * final de la primera
 * cadena y muestra el mensaje completo.
 *
 * Autor: Yoquielvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <string.h> // Para strcat()
using namespace std;

int main() {
    // Declaracion de variables
    char mensaje[100] = "Hola que tal "; // Mensaje inicial
    char nombre[50]; // Para almacenar el nombre del usuario

    // Pedir el nombre al usuario
    cout << "Por favor, digite su nombre: ";
    cin.getline(nombre, 50, '\n');

    // Añadir el nombre al final del mensaje
    strcat(mensaje, nombre);

    // Mostrar el mensaje completo
    cout << "\nMensaje completo: " << mensaje << endl;

    return 0;
}
```

Resultado de la Ejecución:



```
=====
Ejecutando: bloque7_ejercicio4
=====
```

Por favor, digite su nombre: yoquelvis

Mensaje completo: Hola que tal yoquelvis

```
=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
```

yoquelvisabreu@Laptop-de-Yoquelvis ~ % █

Programa: bloque7_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:48:16

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

sentidos).\n";
cout << "                                Ejemplos: ana, reconocer, anita lava la
tina\n\n";

// Solicitar la palabra
cout << "■ Ingrese una palabra o frase: ";
cin.getline(palabra, 100);

// Limpiar la palabra (quitar espacios y convertir a minúsculas)
limpiarCadena(palabra, palabraLimpia);

int longitud = strlen(palabraLimpia);

// Verificar si es un palíndromo
for(int i = 0; i < longitud/2; i++) {
    if(palabraLimpia[i] != palabraLimpia[longitud-1-i]) {
        esPalindromo = false;
        break;
    }
}

// Mostrar información del análisis
cout << "\n■ ANÁLISIS DE PALÍNDROMO:\n";
cout << "=====\n";
cout << " Texto original: \"" << palabra << "\"\n";
cout << " Texto procesado: \"" << palabraLimpia << "\" (sin espacios,
en minúsculas)\n";
cout << " Longitud: " << longitud << " caracteres\n\n";

// Visualización del palíndromo
cout << "■ VERIFICACIÓN:\n";
cout << "-----\n";

// Mostrar la palabra de izquierda a derecha
cout << " Normal: ";
for(int i = 0; i < longitud; i++) {
    cout << palabraLimpia[i] << " ";
}
cout << "\n";

// Mostrar la palabra de derecha a izquierda
cout << " Inversa: ";
for(int i = longitud-1; i >= 0; i--) {
    cout << palabraLimpia[i] << " ";
}
cout << "\n\n";

// Mostrar el resultado
cout << "■ RESULTADO: ";
if(esPalindromo) {
    cout << "■ \"" << palabra << "\" Sí es un palíndromo!\n\n";
} else {
    cout << "■ \"" << palabra << "\" NO es un palíndromo.\n\n";
}

return 0;
}

```

Resultado de la Ejecución:

```
yoquelvisabreu — -zsh — 80x24

es un palíndromo (se lee igual en ambos sentidos).
Ejemplos: ana, reconocer, anita lava la tina

Ingrese una palabra o frase: yoquelvis le gusta c++

ANÁLISIS DE PALÍNDROMO:
=====
Texto original: "yoquelvis le gusta c++"
Texto procesado: "yoquelvislegustac" (sin espacios, en minúsculas)
Longitud: 17 caracteres

VERIFICACIÓN:
-----
Normal:  y o q u e l v i s l e g u s t a c
Inversa: c a t s u g e l s i v l e u q o y

RESULTADO: ❌ "yoquelvis le gusta c++" NO es un palíndromo.

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %
```

Programa: bloque8_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:48:31

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

cout << "■ Edad: ";
cin >> corredor1.edad;
cin.ignore();

cout << "■ Sexo (Masculino/Femenino): ";
cin.getline(corredor1.sexo, 10, '\n');

cout << "■ Club: ";
cin.getline(corredor1.club, 30, '\n');

// Asignar categoría según la edad
if(corredor1.edad <= 18) {
    strcpy(corredor1.categoria, "Juvenil ■");
} else if(corredor1.edad <= 40) {
    strcpy(corredor1.categoria, "Senior ■");
} else {
    strcpy(corredor1.categoria, "Veterano ■");
}

// Limpiar pantalla y mostrar resultados
limpiarPantalla();
mostrarBanner();

// Mostrar los datos del corredor con formato mejorado
cout << "■ FICHA DEL CORREDOR\n";
cout << "=====\n\n";

cout << setfill(' ') << fixed;
cout << left << setw(15) << "■ Nombre:" << corredor1.nombre << endl;
cout << left << setw(15) << "■ Edad:" << corredor1.edad << " años" <<
endl;
cout << left << setw(15) << "■ Sexo:" << corredor1.sexo << endl;
cout << left << setw(15) << "■ Club:" << corredor1.club << endl;
cout << left << setw(15) << "■ Categoría:" << corredor1.categoria <<
endl;

cout << "\n■ ¡Registro completado con éxito! ■\n\n";

return 0;
}

```

Resultado de la Ejecución:








REGISTRO DE CORREDORES v1.0



FICHA DEL CORREDOR

=====

 Nombre: yoquelvis
 Edad: 12 años
 Sexo: masculino
 Club: naco
 Categoría: Juvenil 🌟

🌟 ¡Registro completado con éxito! 🌟

=====

Ejecución completada

Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque8_ejercicio2

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:49:15

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

cout << "■ REGISTRO DE CALIFICACIONES\n";
cout << "=====\n\n";
cout << "Por favor, ingrese los datos de los 3 estudiantes:\n\n";

// Pedir datos para los 3 alumnos
for(int i = 0; i < 3; i++) {
    cout << "■ ESTUDIANTE " << i+1 << " de 3\n";
    cout << "-----\n";

    cout << "■ Nombre: ";
    cin.ignore(i == 0 ? 0 : numeric_limits<streamsize>::max(), '\n');
    cin.getline(alumnos[i].nombre, 50, '\n');

    cout << "■ Edad: ";
    cin >> alumnos[i].edad;

    do {
        cout << "■ Promedio (0-10): ";
        cin >> alumnos[i].promedio;
        if(alumnos[i].promedio < 0 || alumnos[i].promedio > 10) {
            cout << "■ Error: El promedio debe estar entre 0 y 10\n";
        }
    } while(alumnos[i].promedio < 0 || alumnos[i].promedio > 10);

    // Verificar si este alumno tiene el mayor promedio
    if(alumnos[i].promedio > mayor_promedio) {
        mayor_promedio = alumnos[i].promedio;
        indice_mayor = i;
    }

    // Mostrar progreso
    mostrarProgreso(i + 1, 3);
}

// Limpiar pantalla y mostrar resultados
limpiarPantalla();
mostrarBanner();

// Mostrar los datos del alumno con el mejor promedio
cout << "■ ESTUDIANTE DESTACADO\n";
cout << "=====\n\n";

cout << setfill(' ') << fixed << setprecision(2);
cout << left << setw(15) << "■ Nombre:" <<
alumnos[indice_mayor].nombre << endl;
cout << left << setw(15) << "■ Edad:" << alumnos[indice_mayor].edad <<
" años" << endl;
cout << left << setw(15) << "■ Promedio:" <<
alumnos[indice_mayor].promedio << endl;

// Mostrar medalla según el promedio
cout << "\n■ ¡Felicitaciones! ";
if(alumnos[indice_mayor].promedio >= 9.0) cout << "¡Medalla de Oro!";
else if(alumnos[indice_mayor].promedio >= 8.0) cout << "¡Medalla de Plata!";
else cout << "¡Medalla de Bronce!";
cout << "\n\n";

return 0;
}

```

Resultado de la Ejecución:



REGISTRO DE ESTUDIANTES v1.0

🏆 ESTUDIANTE DESTACADO

=====

👤 Nombre: yoquelvis
🎂 Edad: 14 años
📊 Promedio: 10.00

🎉 ¡Felicitaciones! ¡Medalla de Oro! 🏅

=====

Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque8_ejercicio3

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:49:37

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```

int main() {
    int n;
    Empleado *empleados;
    int pos_mayor = 0;
    int pos_menor = 0;
    float mayor_salario = 0;
    float menor_salario = numeric_limits<float>::max();

    // Limpiar pantalla y mostrar banner inicial
    limpiarPantalla();
    mostrarBanner();

    cout << "■ REGISTRO DE EMPLEADOS\n";
    cout << "=====\n\n";

    // Pedir cantidad de empleados
    do {
        cout << "■ Número de empleados a registrar: ";
        cin >> n;
        if(n <= 0) {
            cout << "■ Error: Debe registrar al menos un empleado\n\n";
        }
    } while(n <= 0);

    // Crear el arreglo dinámico
    empleados = new Empleado[n];

    cout << "\n■ Por favor, ingrese los datos de los empleados:\n\n";

    // Pedir datos de los empleados
    for(int i = 0; i < n; i++) {
        cout << "■ EMPLEADO " << i+1 << " de " << n << "\n";
        cout << "-----\n";

        cin.ignore(i == 0 ? 0 : numeric_limits<streamsize>::max(), '\n');

        cout << "■ Nombre: ";
        cin.getline(empleados[i].nombre, 50, '\n');

        cout << "■ Cargo: ";
        cin.getline(empleados[i].cargo, 30, '\n');

        do {
            cout << "■ Salario: $";
            cin >> empleados[i].salario;
            if(empleados[i].salario <= 0) {
                cout << "■ Error: El salario debe ser mayor que 0\n";
            }
        } while(empleados[i].salario <= 0);

        // Verificar salarios extremos
        if(empleados[i].salario > mayor_salario) {
            mayor_salario = empleados[i].salario;
            pos_mayor = i;
        }
        if(empleados[i].salario < menor_salario) {
            menor_salario = empleados[i].salario;
            pos_menor = i;
        }

        // Mostrar progreso
        mostrarProgreso(i + 1, n);
    }

    // Limpiar pantalla y mostrar resultados
    limpiarPantalla();
    mostrarBanner();

    cout << "■ ANÁLISIS SALARIAL\n";
    cout << "=====\n\n";

    // Mostrar empleado con mayor salario
    cout << "■ SALARIO MAS ALTO\n";
    cout << "-----\n";
    cout << setfill(' ') << fixed << setprecision(2);
    cout << left << setw(15) << "■ Nombre:" << empleados[pos_mayor].nombre
    << endl;
    cout << left << setw(15) << "■ Cargo:" << empleados[pos_mayor].cargo
    << endl;

```

```

        cout << left << setw(15) << "■ Salario:" <<
formatearSalario(empleados[pos_mayor].salario) << endl;

        cout << "\n■ SALARIO MÁS BAJO\n";
        cout << "-----\n";
        cout << left << setw(15) << "■ Nombre:" << empleados[pos_menor].nombre
<< endl;
        cout << left << setw(15) << "■ Cargo:" << empleados[pos_menor].cargo
<< endl;
        cout << left << setw(15) << "■ Salario:" <<
formatearSalario(empleados[pos_menor].salario) << endl;

        // Mostrar diferencia salarial
        cout << "\n■ BRECHA SALARIAL\n";
        cout << "-----\n";
        cout << "Diferencia: " << formatearSalario(mayor_salario -
menor_salario) << endl;

        // Liberar memoria
        delete[] empleados;

        cout << "\n■ Análisis completado con éxito ■\n\n";

        return 0;
}

```

Resultado de la Ejecución:

```

yoquelvisabreu — -zsh — 80x24
🏆 SALARIO MÁS ALTO
-----
👤 Nombre:
👤 Cargo:  manuel 30
💰 Salario:  $20.00

📉 SALARIO MÁS BAJO
-----
👤 Nombre:
👤 Cargo:  manuel 30
💰 Salario:  $20.00

📈 BRECHA SALARIAL
-----
Diferencia: $0.00

✨ Análisis completado con éxito ✨

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque8_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:49:57

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

[illegible]

```
// Declaración de variables
int n;
Atleta *atletas;
int indice_mayor = 0;

// Explicación del programa
cout << "■ DESCRIPCIÓN: Este programa registra datos de atletas olímpicos\n";
cout << "y encuentra al atleta con mayor número de medallas.\n\n";

// Solicitar el número de atletas con validación
do {
    cout << "■ Ingrese el número de atletas a registrar: ";
    cin >> n;

    if(n <= 0) {
        cout << "■ Error: Debe registrar al menos un atleta.\n\n";
    }
} while(n <= 0);

// Crear el arreglo dinámico de atletas
atletas = new Atleta[n];

// Solicitar los datos de los atletas
cout << "\n■ REGISTRO DE DATOS DE ATLETAS:\n";
cout << "=====\n";

for(int i = 0; i < n; i++) {
    cout << "\n■ ATLETA #" << (i+1) << " de " << n << ":\n";
    cout << "-----\n";

    // Limpiar buffer para evitar problemas con getline
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << " ■ Nombre: ";
    cin.getline(atletas[i].nombre, 50);

    cout << " ■ País: ";
    cin.getline(atletas[i].pais, 50);

    do {
        cout << " ■ Número de medallas: ";
        cin >> atletas[i].medallas;

        if(atletas[i].medallas < 0) {
            cout << " ■ Error: El número de medallas no puede ser negativo.\n";
        }
    } while(atletas[i].medallas < 0);

    // Actualizar el atleta con mayor número de medallas
    if(i == 0 || atletas[i].medallas > atletas[indice_mayor].medallas) {
        indice_mayor = i;
    }

    // Mostrar progreso
    mostrarProgreso(i+1, n);
}

// Mostrar tabla con todos los atletas
limpiarPantalla();
mostrarBanner();

cout << "■ TABLA DE ATLETAS REGISTRADOS:\n";
cout << "=====\n";

cout << "+-----+-----+-----+-----+-----+\n";
cout << "| NUM | NOMBRE | PAÍS | MEDALLAS |\n";
cout << "+-----+-----+-----+-----+-----+\n";

for(int i = 0; i < n; i++) {
    cout << "| " << setw(3) << (i+1) << " | "
        << setw(22) << left << atletas[i].nombre << " | "
        << setw(22) << left << atletas[i].pais << " | "
        << setw(10) << left << atletas[i].medallas << " |\n";
}
```


Programa: bloque8_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:50:14

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 8: Ejercicio 5
 * Programa con 2 estructuras: una llamada promedio que tiene los campos
 * notal, nota2, nota3;
 * y otra llamada alumno que tiene los campos nombre, sexo, edad; hace que
 * la estructura
 * promedio esté anidada en la estructura alumno, pide todos los datos para
 * un alumno,
 * calcula su promedio, y por último imprime todos sus datos incluido el
 * promedio.
 */
* Autor: Yoquelyvis Abreu
* Fecha: Marzo 2024
*/

#include <iostream>
#include <string.h>
using namespace std;

// Definición de la estructura Promedio
struct Promedio {
    float notal;
    float nota2;
    float nota3;
    float promedio_final; // Para almacenar el promedio calculado
};

// Definición de la estructura Alumno con Promedio anidado
struct Alumno {
    char nombre[50];
    char sexo[10];
    int edad;
    Promedio calificaciones; // Estructura anidada
};

int main() {
    // Declaración de variables
    Alumno alumno1;

    // Pedir datos del alumno
    cout << "Digite los datos del alumno:" << endl;

    cout << "Nombre: ";
    cin.getline(alumno1.nombre, 50, '\n');

    cout << "Sexo (Masculino/Femenino): ";
    cin.getline(alumno1.sexo, 10, '\n');

    cout << "Edad: ";
    cin >> alumno1.edad;

    // Pedir las notas
    cout << "\nDigite las calificaciones del alumno:" << endl;

    cout << "Nota 1: ";
    cin >> alumno1.calificaciones.notal;

    cout << "Nota 2: ";
    cin >> alumno1.calificaciones.nota2;

    cout << "Nota 3: ";
    cin >> alumno1.calificaciones.nota3;
```

```

// Calcular el promedio
alumno1.calificaciones.promedio_final = (alumno1.calificaciones.nota1
+
alumno1.calificaciones.nota2 +
alumno1.calificaciones.nota3) /
3;

// Mostrar los datos del alumno
cout << "\n----- DATOS DEL ALUMNO -----" << endl;
cout << "Nombre: " << alumno1.nombre << endl;
cout << "Sexo: " << alumno1.sexo << endl;
cout << "Edad: " << alumno1.edad << " años" << endl;
cout << "Notas: " << alumno1.calificaciones.nota1 << ", "
<< alumno1.calificaciones.nota2 << ", "
<< alumno1.calificaciones.nota3 << endl;
cout << "Promedio: " << alumno1.calificaciones.promedio_final << endl;
return 0;
}

```

Resultado de la Ejecución:

```

=====
Digite los datos del alumno:
Nombre: manuel
Sexo (Masculino/Femenino): femenino
Edad: 20

Digite las calificaciones del alumno:
Nota 1: 1
Nota 2: 2
Nota 3: 30

----- DATOS DEL ALUMNO -----
Nombre: manuel
Sexo: femenino
Edad: 20 años
Notas: 1, 2, 30
Promedio: 11

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque8_ejercicio6

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:51:08

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 8: Ejercicio 6
 * Programa que utiliza las 2 estructuras del problema 5, pero ahora pide
 * datos
 * para N alumnos, calcula cuál de todos tiene el mejor promedio, e imprime
 * sus datos.
 */
* Autor: Yoquelyvis Abreu
* Fecha: Marzo 2024
*/

#include <iostream>
#include <string.h>
using namespace std;

// Definición de la estructura Promedio
struct Promedio {
    float notal;
    float nota2;
    float nota3;
    float promedio_final; // Para almacenar el promedio calculado
};

// Definición de la estructura Alumno con Promedio anidado
struct Alumno {
    char nombre[50];
    char sexo[10];
    int edad;
    Promedio calificaciones; // Estructura anidada
};

int main() {
    // Declaración de variables
    int n; // Cantidad de alumnos
    Alumno *alumnos; // Arreglo dinámico para almacenar los alumnos
    int pos_mejor = 0; // Posición del alumno con mejor promedio
    float mejor_promedio = 0; // Para almacenar el mejor promedio
    int i; // Variable para ciclos

    // Pedir cantidad de alumnos
    cout << "Digite el numero de alumnos: ";
    cin >> n;

    // Crear el arreglo dinámico para los alumnos
    alumnos = new Alumno[n];

    // Pedir datos para cada alumno
    for(i = 0; i < n; i++) {
        cout << "\nDigite los datos del alumno " << i+1 << ":" << endl;

        cin.ignore(); // Limpiar buffer
        cout << "Nombre: ";
        cin.getline(alumnos[i].nombre, 50, '\n');

        cout << "Sexo (Masculino/Femenino): ";
        cin.getline(alumnos[i].sexo, 10, '\n');

        cout << "Edad: ";
        cin >> alumnos[i].edad;

        // Pedir las notas
        cout << "\nDigite las calificaciones del alumno:" << endl;
```

```

        cout << "Nota 1: ";
        cin >> alumnos[i].calificaciones.nota1;

        cout << "Nota 2: ";
        cin >> alumnos[i].calificaciones.nota2;

        cout << "Nota 3: ";
        cin >> alumnos[i].calificaciones.nota3;

        // Calcular el promedio
        alumnos[i].calificaciones.promedio_final =
(alumnos[i].calificaciones.nota1 +
alumnos[i].calificaciones.nota2 +
alumnos[i].calificaciones.nota3) / 3;

        // Verificar si este alumno tiene mejor promedio
        if(alumnos[i].calificaciones.promedio_final > mejor_promedio) {
            mejor_promedio = alumnos[i].calificaciones.promedio_final;
            pos_mejor = i;
        }

        // Mostrar los datos del alumno con mejor promedio
        cout << "\n----- ALUMNO CON MEJOR PROMEDIO -----" << endl;
        cout << "Nombre: " << alumnos[pos_mejor].nombre << endl;
        cout << "Sexo: " << alumnos[pos_mejor].sexo << endl;
        cout << "Edad: " << alumnos[pos_mejor].edad << " años" << endl;
        cout << "Notas: " << alumnos[pos_mejor].calificaciones.nota1 << ", "
            << alumnos[pos_mejor].calificaciones.nota2 << ", "
            << alumnos[pos_mejor].calificaciones.nota3 << endl;

        cout << "Mejor Promedio: " <<
alumnos[pos_mejor].calificaciones.promedio_final << endl;

        // Liberar memoria
        delete[] alumnos;

        return 0;
    }
}

```

Resultado de la Ejecución:



Nota 3: 30

Digite los datos del alumno 3:

Nombre: jorge

Sexo (Masculino/Femenino): femenino

Edad: 20

Digite las calificaciones del alumno:

Nota 1: 23

Nota 2: 23

Nota 3: 23

----- ALUMNO CON MEJOR PROMEDIO -----

Nombre: yoquelvis 30 20

Sexo: femenino

Edad: 20 anos

Notas: 30, 30, 30

Mejor Promedio: 30

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

Programa: bloque8_ejercicio7

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:51:31

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 8: Ejercicio 7
 * Programa que define una estructura que indica el tiempo empleado por un
 * ciclista
 * en una etapa. La estructura tiene tres campos: horas, minutos y segundos.
 * Calcula el tiempo total empleado en correr todas las etapas.
 *
 * Autor: Yoquelyvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
using namespace std;

// Definición de la estructura Tiempo
struct Tiempo {
    int horas;
    int minutos;
    int segundos;
};

int main() {
    // Declaración de variables
    int n; // Número de etapas
    Tiempo *etapas; // Arreglo dinámico para almacenar los tiempos por
    etapa
    Tiempo total = {0, 0, 0}; // Tiempo total inicializado en cero
    int i; // Variable para ciclos

    // Pedir cantidad de etapas
    cout << "Digite el número de etapas: ";
    cin >> n;

    // Crear el arreglo dinámico para las etapas
    etapas = new Tiempo[n];

    // Pedir datos para cada etapa
    for(i = 0; i < n; i++) {
        cout << "\nEtapas " << i+1 << ":" << endl;

        cout << "Horas: ";
        cin >> etapas[i].horas;

        cout << "Minutos: ";
        cin >> etapas[i].minutos;

        cout << "Segundos: ";
        cin >> etapas[i].segundos;
    }

    // Calcular el tiempo total
    for(i = 0; i < n; i++) {
        total.horas += etapas[i].horas;
        total.minutos += etapas[i].minutos;
        total.segundos += etapas[i].segundos;
    }

    // Ajustar los minutos y segundos si son mayores o iguales a 60
    total.minutos += total.segundos / 60;
    total.segundos = total.segundos % 60;

    total.horas += total.minutos / 60;
    total.minutos = total.minutos % 60;
}
```

```

// Mostrar el tiempo total
cout << "\n----- TIEMPO TOTAL -----" << endl;
cout << "Horas: " << total.horas << endl;
cout << "Minutos: " << total.minutos << endl;
cout << "Segundos: " << total.segundos << endl;
cout << "Tiempo total: " << total.horas << "h "
                        << total.minutos << "m "
                        << total.segundos << "s" << endl;

// Liberar memoria
delete[] etapas;

return 0;
}

```

Resultado de la Ejecución:

```

=====
Ejecutando: bloque8_ejercicio7
=====

Digite el numero de etapas: 2

Etapa 1:
Horas: 10:20am
Minutos: Segundos:
Etapa 2:
Horas: Minutos: Segundos:
----- TIEMPO TOTAL -----
Horas: 10
Minutos: 0
Segundos: 0
Tiempo total: 10h 0m 0s

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...

yoquelvisabreu@Laptop-de-Yoquelvis ~ %

```

Programa: bloque8_ejercicio8

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 20:51:56

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

Código Fuente:

```
/*
 * Bloque 8: Ejercicio 8
 * Programa que define una estructura que sirve para representar a una
 persona.
 * La estructura contiene dos campos: el nombre de la persona y un valor de
 tipo lógico
 * que indica si la persona tiene algún tipo de discapacidad. Dado un vector
 de personas
 * rellena dos nuevos vectores: uno que contiene las personas sin
 discapacidad y otro
 * con las personas con discapacidad.
 */
 * Autor: Yoquielvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <string.h>
using namespace std;

// Definición de la estructura Persona
struct Persona {
    char nombre[50];
    bool discapacidad; // true si tiene discapacidad, false si no tiene
};

int main() {
    // Declaración de variables
    int n; // Número de personas
    Persona *personas; // Vector original de personas
    Persona *sin_discapacidad; // Vector para personas sin discapacidad
    Persona *con_discapacidad; // Vector para personas con discapacidad
    int contador_sin = 0; // Contador de personas sin discapacidad
    int contador_con = 0; // Contador de personas con discapacidad
    int i; // Variable para ciclos
    int opcion; // Para la selección de si tiene o no discapacidad

    // Pedir cantidad de personas
    cout << "Digite el numero de personas: ";
    cin >> n;

    // Crear los vectores dinámicos
    personas = new Persona[n];
    sin_discapacidad = new Persona[n]; // En el peor caso, todos sin
discapacidad
    con_discapacidad = new Persona[n]; // En el peor caso, todos con
discapacidad

    // Pedir datos para cada persona
    for(i = 0; i < n; i++) {
        cout << "\nPersona " << i+1 << ":" << endl;

        cin.ignore(); // Limpiar buffer
        cout << "Nombre: ";
        cin.getline(personas[i].nombre, 50, '\n');

        cout << "Tiene alguna discapacidad? (1=Si, 0=No): ";
        cin >> opcion;

        // Asignar el valor booleano según la opción
        if(opcion == 1) {
            personas[i].discapacidad = true;
        } else {
```



```

        personas[i].discapacidad = false;
    }
}

// Separar las personas en los dos vectores
for(i = 0; i < n; i++) {
    if(personas[i].discapacidad) {
        // Persona con discapacidad
        strcpy(con_discapacidad[contador_con].nombre,
personas[i].nombre);
        con_discapacidad[contador_con].discapacidad = true;
        contador_con++;
    } else {
        // Persona sin discapacidad
        strcpy(sin_discapacidad[contador_sin].nombre,
personas[i].nombre);
        sin_discapacidad[contador_sin].discapacidad = false;
        contador_sin++;
    }
}

// Mostrar las personas sin discapacidad
cout << "\n----- PERSONAS SIN DISCAPACIDAD -----" << endl;
if(contador_sin > 0) {
    for(i = 0; i < contador_sin; i++) {
        cout << i+1 << ". " << sin_discapacidad[i].nombre << endl;
    }
} else {
    cout << "No hay personas sin discapacidad." << endl;
}

// Mostrar las personas con discapacidad
cout << "\n----- PERSONAS CON DISCAPACIDAD -----" << endl;
if(contador_con > 0) {
    for(i = 0; i < contador_con; i++) {
        cout << i+1 << ". " << con_discapacidad[i].nombre << endl;
    }
} else {
    cout << "No hay personas con discapacidad." << endl;
}

// Liberar memoria
delete[] personas;
delete[] sin_discapacidad;
delete[] con_discapacidad;

return 0;
}

```

Resultado de la Ejecución:

```
yoquelvisabreu — -zsh — 80x24

16. 2
17. 2
18. 2
19. 2
20.
21. 3
22. 4
23.
24. 4
25. 4
26. 4
27. 4
28.
29. 4
30.

----- PERSONAS CON DISCAPACIDAD -----
No hay personas con discapacidad.

=====
Ejecución completada
Presione Enter cuando haya terminado para continuar...
f
yoquelvisabreu@Laptop-de-Yoquelvis ~ % f
```