

# Recopilación de Programas C++

Desarrollado por: Yoquelvis Jorge Abreu

## Acerca de esta herramienta:

Esta aplicación es un compilador y visualizador avanzado para programas C++, diseñado para facilitar el aprendizaje y la evaluación de ejercicios de programación. La herramienta automatiza el proceso de compilación, ejecución y documentación de programas C++.

## Características principales:

- Compilación automática de archivos C++ usando g++
- Ejecución interactiva en terminal con captura de pantalla
- Generación automática de informes en PDF con el código fuente y los resultados
- Análisis del código para identificar estructuras y características
- Interfaz intuitiva que permite procesar múltiples archivos secuencialmente

Los programas se compilan utilizando estándares modernos de C++ y se documentan incluyendo tanto el código fuente como una captura de pantalla de su ejecución, permitiendo una visualización completa de su funcionamiento.

Generado el 11/03/2025 a las 23:28:44

# Programa: bloque5\_ejercicio1

Compilador: g++  
Flags: -Wall -std=c++11  
Fecha: 2025-03-11 23:29:05  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SUMA DE ELEMENTOS EN UN VECTOR █
 * =====
 *
 * Bloque 5: Ejercicio 1
 * -----
 * Programa que define un vector de números y calcula la suma de sus
 * elementos.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_VECTOR = "\033[1;34m"; // Azul brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
    cout << "█      SUMA DE ELEMENTOS VECTOR      █\n";
    cout << "██████████ ██████████ ██████████ ██████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar el vector con formato mejorado
void mostrarVector(const int vector[], int tamanio, string titulo) {
    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Línea superior del vector
    cout << COLOR_VECTOR << "█";
    for(int i = 0; i < tamanio; i++) {
        cout << "██████████";
        if(i < tamanio-1) cout << "█";
    }
    cout << "█\n";

    // Índices del vector
    cout << "█";
    for(int i = 0; i < tamanio; i++) {
        cout << "[" << setw(2) << i << "] ";
        if(i < tamanio-1) cout << "█";
    }
    cout << "█\n";
}
```



```

        cout << COLOR_RESET;
    }

    // Calcular la suma de los elementos del vector
    for(int i = 0; i < tamanio; i++) {
        suma += numeros[i]; // Acumulamos cada elemento en la variable suma
    }

    // Mostrar el vector
    mostrarVector(numeros, tamanio, "■ Vector ingresado:");

    // Mostrar la animación de suma
    mostrarAnimacionSuma(numeros, tamanio, suma);

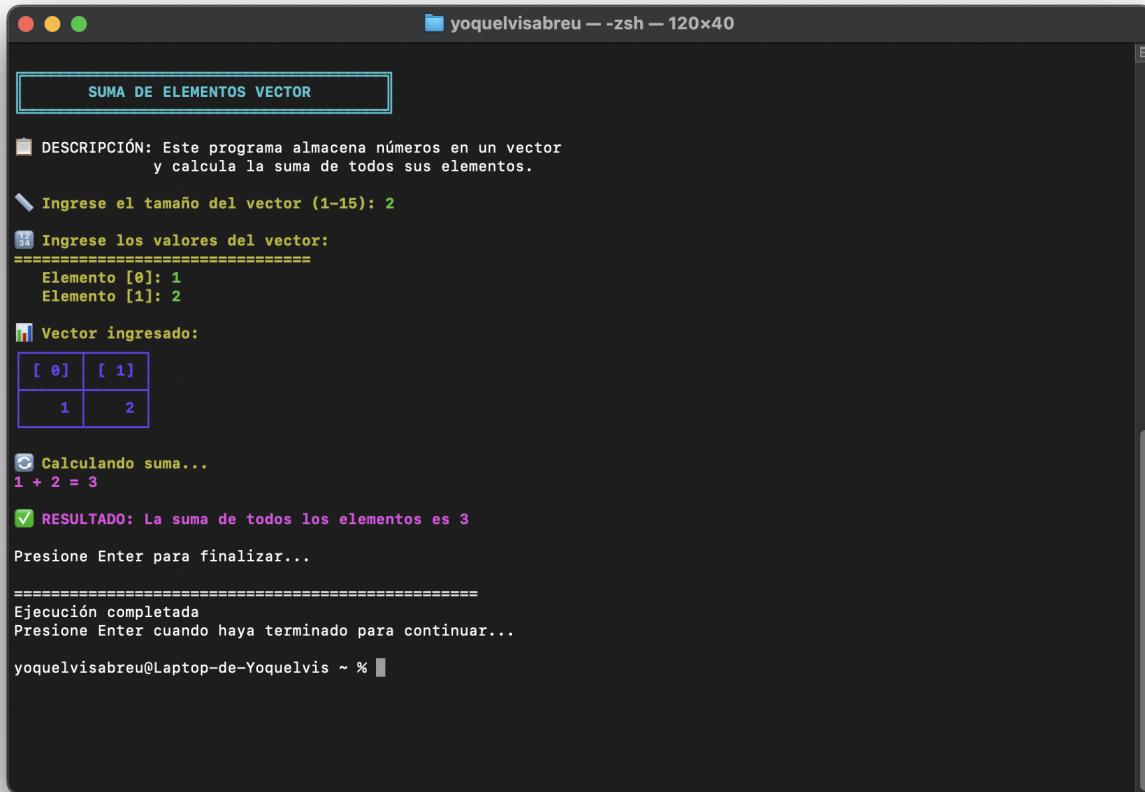
    // Mostrar el resultado con formato
    cout << "\n" << COLOR_RESULT << "■ RESULTADO: La suma de todos los
elementos es " << suma << COLOR_RESET << "\n\n";

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar...";
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**



The screenshot shows a terminal window titled "yoquelvisabreu -- zsh -- 120x40". The window displays the output of a C++ program. The program starts with a header "SUMA DE ELEMENTOS VECTOR". It provides a description: "Este programa almacena números en un vector y calcula la suma de todos sus elementos." It prompts the user to "Ingresar el tamaño del vector (1-15):" and receives the input "2". It then asks for "Ingresar los valores del vector:" and shows two rows of a 2x2 grid. The first row is labeled "[ 0 ]" and the second row is labeled "[ 1 ]". The first cell contains "1" and the second cell contains "2". The program then calculates the sum: "Calculando suma..." followed by "1 + 2 = 3". It concludes with a result message: "RESULTADO: La suma de todos los elementos es 3". Finally, it prompts the user to "Presione Enter para finalizar...". At the bottom, it says "Ejecución completada" and "Presione Enter cuando haya terminado para continuar...". The command prompt at the bottom is "yoquelvisabreu@Laptop-de-Yoquelvis ~ %".

# Programa: bloque5\_ejercicio2

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:29:17

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ MULTIPLICACIÓN DE ELEMENTOS EN UN VECTOR █
 * =====
 *
 * Bloque 5: Ejercicio 2
 * -----
 * Programa que define un vector de números y calcula la multiplicación
 * acumulada de sus elementos.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_VECTOR = "\033[1;34m"; // Azul brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout <<
    "██████████ █ MULTIPLICACIÓN DE ELEMENTOS EN VECTOR █\n";
    cout << COLOR_RESET;
}

// Función para mostrar el vector con formato mejorado
void mostrarVector(const int vector[], int tamanio, string titulo) {
    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Línea superior del vector
    cout << COLOR_VECTOR << "█";
    for(int i = 0; i < tamanio; i++) {
        cout << "██████████";
        if(i < tamanio-1) cout << "█";
    }
    cout << "█\n";

    // Índices del vector
    cout << "█";
    for(int i = 0; i < tamanio; i++) {
        cout << "[" << setw(2) << i << "] ";
        if(i < tamanio-1) cout << "█";
    }
}
```

```

cout << "■\n";
// Línea separadora
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << "■■■■■■■■■■";
    if(i < tamanio-1) cout << "■";
}
cout << "■\n";

// Valores del vector
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << " " << setw(4) << vector[i] << " ";
    if(i < tamanio-1) cout << "■";
}
cout << "■\n";

// Línea inferior del vector
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << "■■■■■■■■■■";
    if(i < tamanio-1) cout << "■";
}
cout << "■" << COLOR_RESET << "\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaracion de variables
    int tamanio;
    const int MAX = 15; // Limitamos para mejor visualización
    int numeros[MAX]; // Vector de hasta MAX numeros enteros
    int multiplicacion = 1; // Variable para almacenar la multiplicación
    acumulada

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa almacena números en un
vector\n";
    cout << "           y calcula el producto de todos ellos.\n\n";

    // Solicitar tamaño del vector
    do {
        cout << COLOR_HIGHLIGHT << "■ Ingrese el tamaño del vector (1-" <<
MAX << "): " << COLOR_INPUT;
        cin >> tamanio;
        cout << COLOR_RESET;

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

    cout << "\n"; // Salto de línea entre inputs

    // Solicitar valores al usuario
    cout << COLOR_HIGHLIGHT << "■ Ingrese los valores del vector:\n";
    cout << "=====*\n" << COLOR_RESET;

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

    // Calcular la multiplicación de los elementos del vector
    for(int i = 0; i < tamanio; i++) {
        multiplicacion *= numeros[i]; // Acumulamos el producto de cada
elemento
    }

    // Mostrar el vector
    mostrarVector(numeros, tamanio, "■ Vector ingresado:");
}

```

```

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

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

// Añadir instrucciones finales
cout << "Presione Enter para finalizar...";
cin.ignore();
cin.get();

return 0;
}

```

### **Resultado de la Ejecución:**

```

MULITPLICACIÓN DE ELEMENTOS EN VECTOR

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

✍ Ingrese el tamaño del vector (1-15): 5
=====
1234  Ingrese los valores del vector:
=====
Elemento [0]: 1
Elemento [1]: 2
Elemento [2]: 3
Elemento [3]: 4
Elemento [4]: 5

■ Vector ingresado:
[ 0] [ 1] [ 2] [ 3] [ 4]
[ 1]   2   3   4   5

1234  Cálculo: 1 × 2 × 3 × 4 × 5 = 120
✓ RESULTADO: La multiplicación de todos los elementos es 120
Presione Enter para finalizar...
=====
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 23:29:44  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ VISUALIZACIÓN DE VECTOR CON ÍNDICES █
 * =====
 *
 * Bloque 5: Ejercicio 3
 * -----
 * Programa que lee de la entrada estándar un vector de números y
 * muestra en la salida estándar los números del vector con sus índices
 * asociados.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_TABLE_HEADER = "\033[1;37;44m"; // Blanco brillante
sobre azul
const string COLOR_INDEX = "\033[1;34m"; // Azul brillante
const string COLOR_VALUE = "\033[1;35m"; // Magenta brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "█ VISUALIZACIÓN DE VECTOR CON ÍNDICES █\n";
    cout << "█\n";
    cout << COLOR_RESET;
}

// Función para mostrar el vector con sus índices en un formato de tabla
mejorado
void mostrarVectorConIndices(const int vector[], int tamanio) {
    cout << COLOR_HIGHLIGHT << "\n█ VECTOR CON ÍNDICES:\n";
    cout << "=====\\n\\n" << COLOR_RESET;

    // Encabezado de la tabla con color
    cout << COLOR_TABLE_HEADER << "█\n";
    cout << "█ ÍNDICE █ VALOR █\n";
    cout << "█\n";
    cout << COLOR_RESET << "\n";

    // Filas de la tabla
    for(int i = 0; i < tamanio; i++) {
        cout << "█ " << COLOR_INDEX << setw(10) << i << COLOR_RESET
            << "█ " << COLOR_VALUE << setw(11) << vector[i] <<
        COLOR_RESET << "█\n";
    }
}
```

```

        // Línea separadora excepto después de la última fila
        if(i < tamanio-1) {
            cout << "████████████████████████████████\n";
        }
    }

    // Línea inferior de la tabla
    cout << "████████████████████████████████\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaracion de variables
    const int MAX = 100; // Tamaño máximo del vector
    int numeros[MAX]; // Vector de números
    int n; // Tamaño real del vector

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa lee números introducidos por el
    usuario\n";
    cout << "                                y muestra el vector con sus índices
    asociados.\n\n";

    // Solicitar el tamaño del vector
    do {
        cout << COLOR_HIGHLIGHT << "■ Ingrese el tamaño del vector (1-" <<
MAX << "): " << COLOR_INPUT;
        cin >> n;
        cout << COLOR_RESET;

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

    cout << "\n"; // Salto de línea entre inputs

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

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

    // Mostrar el vector con sus índices usando la función mejorada
    mostrarVectorConIndices(numeros, n);

    // Añadir instrucciones finales
    cout << COLOR_HIGHLIGHT << "■ Vector mostrado con éxito" <<
COLOR_RESET << "\n\n";
    cout << "Presione Enter para finalizar... ";
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

```
yoquelvisabreu -- zsh -- 120x40
Elemento [7]: 80
Elemento [8]: 90
Elemento [9]: 100
VECTOR CON ÍNDICES:
=====


| ÍNDICE | VALOR |
|--------|-------|
| 0      | 10    |
| 1      | 20    |
| 2      | 30    |
| 3      | 40    |
| 4      | 50    |
| 5      | 60    |
| 6      | 70    |
| 7      | 80    |
| 8      | 90    |
| 9      | 100   |


✓ Vector mostrado con éxito
Presione Enter para finalizar...
=====
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 23:30:17  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ VECTOR EN ORDEN INVERSO █
 * =====
 *
 * Bloque 5: Ejercicio 4
 * -----
 * Programa que define un vector de números y muestra en la
 * salida estándar el vector en orden inverso—del último al primer elemento.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_VECTOR = "\033[1;34m"; // Azul brillante
const string COLOR_VECTOR_INVERSE = "\033[1;35m"; // Magenta brillante
const string COLOR_ARROW = "\033[1;31m"; // Rojo brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████\n";
    cout << "█     VECTOR EN ORDEN INVERSO     █\n";
    cout << "██████████ ██████████ ██████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar un vector con formato mejorado
void mostrarVector(const int vector[], int tamano, string titulo, string color) {
    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    cout << color;
    // Línea superior del vector
    cout << "█";
    for(int i = 0; i < tamano; i++) {
        cout << "███████";
        if(i < tamano-1) cout << "█";
    }
    cout << "█\n";

    // Índices del vector
    cout << "█";
    for(int i = 0; i < tamano; i++) {
        cout << "[" << setw(2) << i << "] ";
        if(i < tamano-1) cout << "█";
    }
}
```

```

}

cout << "■\n";

// Línea separadora
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << "■■■■■■■■";
    if(i < tamanio-1) cout << "■";
}
cout << "■\n";

// Valores del vector
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << " " << setw(4) << vector[i] << " ";
    if(i < tamanio-1) cout << "■";
}
cout << "■\n";

// Línea inferior del vector
cout << "■";
for(int i = 0; i < tamanio; i++) {
    cout << "■■■■■■■■";
    if(i < tamanio-1) cout << "■";
}
cout << "■" << COLOR_RESET << "\n";
}

// Función para mostrar una animación de inversión
void mostrarAnimacionInversion() {
    cout << "\n" << COLOR_ARROW;
    cout << "      ↓↓↓↓\n";
    cout << "      ■■■■■■■■\n";
    cout << "      ■ INVIRTIENDO... ■\n";
    cout << "      ↑↑↑↑↑↑↑\n" << COLOR_RESET;
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaracion de variables
    const int MAX = 100; // Tamaño máximo del vector
    int numeros[MAX]; // Vector de números
    int invertido[MAX]; // Vector para almacenar los números en orden
    inverso
    int n; // Tamaño real del vector

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa almacena números en un
vector\n";
    cout << "           y los muestra en orden inverso.\n\n";

    // Solicitar el tamaño del vector
    do {
        cout << COLOR_HIGHLIGHT << "■ Ingrese el tamaño del vector (1-"
        MAX << "): " << COLOR_INPUT;
        cin >> n;
        cout << COLOR_RESET;

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

    cout << "\n"; // Salto de línea entre inputs

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

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

```

```

    }

    // Crear vector invertido
    for(int i = 0; i < n; i++) {
        invertido[i] = numeros[n-1-i];
    }

    // Mostrar el vector original
    mostrarVector(numeros, n, "■ Vector original:", COLOR_VECTOR);

    // Mostrar animación de inversión
    mostrarAnimacionInversion();

    // Mostrar el vector en orden inverso
    mostrarVector(invertido, n, "■ Vector inverso:",
    COLOR_VECTOR_INVERSE);

    // Añadir instrucciones finales
    cout << "\n" << COLOR_HIGHLIGHT << "■ Inversión de vector completada
con éxito" << COLOR_RESET << "\n\n";
    cout << "Presione Enter para finalizar..." ;
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

Terminal Output:

```

Elemento [2]: 10
Elemento [3]: 9
Elemento [4]: 8
Elemento [5]: 7
Elemento [6]: 6
Elemento [7]: 5
Elemento [8]: 4
Elemento [9]: 3
Elemento [10]: 2
Elemento [11]: 1

■ Vector original:
[ 0 ] [ 1 ] [ 2 ] [ 3 ] [ 4 ] [ 5 ] [ 6 ] [ 7 ] [ 8 ] [ 9 ] [ 10 ] [ 11 ]
12   11   10    9    8    7    6    5    4    3    2    1

↓ ↓ ↓ ↓ ↓
INVERTIENDO...
↑ ↑ ↑ ↑ ↑

■ Vector inverso:
[ 0 ] [ 1 ] [ 2 ] [ 3 ] [ 4 ] [ 5 ] [ 6 ] [ 7 ] [ 8 ] [ 9 ] [ 10 ] [ 11 ]
1     2     3     4     5     6     7     8     9    10    11    12

✓ Inversión de vector completada con éxito
Presione Enter para finalizar...

=====
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 23:30:17

Estado: Error en compilación

Error:

```
/Users/yoquelvisabreu/Desktop/c++/practica2/bloque5_ejercicio5.cpp:194:1: error: extraneous closing brace
('}')
194 | }
| ^
1 error generated.
```

## Código Fuente:

```
/*
 * ── MAYOR ELEMENTO DEL VECTOR ──
 * =====
 *
 * Bloque 5: Ejercicio 5
 * -----
 * Programa que lee de la entrada estándar un vector de enteros y
 * determina el mayor elemento del vector.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
#include <limits> // Para usar INT_MIN
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_VECTOR = "\033[1;34m"; // Azul brillante
const string COLOR_MAX = "\033[1;31m"; // Rojo brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ MAYOR ELEMENTO DEL VECTOR ██████████\n";
    cout << "██████████\n";
    cout << COLOR_RESET;
}

// Función para mostrar un vector con formato mejorado
void mostrarVector(const int vector[], int tamano, int posicionMayor) {
    cout << COLOR_HIGHLIGHT << "\n■ Vector ingresado:" << COLOR_RESET <<
endl;

    // Línea superior del vector
    cout << COLOR_VECTOR << "■";
    for(int i = 0; i < tamano; i++) {
        cout << "██████████";
        if(i < tamano-1) cout << "■";
    }
    cout << "■\n";
}
```



```

}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    const int MAX = 100; // Tamaño máximo del vector
    int numeros[MAX]; // Vector de números
    int n; // Tamaño real del vector
    int mayor = numeric_limits<int>::min(); // Inicializamos con el menor
    valor posible
    int posicion = 0; // Posición del mayor elemento

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa encuentra el mayor elemento\n";
    cout << " dentro de un vector de números enteros.\n\n";

    // Solicitar el tamaño del vector
    do {
        cout << COLOR_HIGHLIGHT << "■ Ingrese el tamaño del vector (1-" <<
        MAX << "): " << COLOR_INPUT;
        cin >> n;
        cout << COLOR_RESET;

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

    cout << "\n"; // Salto de línea entre inputs

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

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

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

    // Mostrar el vector completo con formato mejorado, destacando el mayor
    mostrarVector(numeros, n, posicion);

    // Mostrar el proceso de búsqueda
    mostrarProcesoBusqueda(numeros, n, mayor, posicion);

    // Mostrar el resultado
    cout << "\n" << COLOR_HIGHLIGHT << "■ Resultado del análisis:" <<
    COLOR_RESET << endl;
    cout << COLOR_RESULT << "■ El mayor elemento es " << mayor << "
    (ubicado en la posición " << posicion << ")" << COLOR_RESET << "\n\n";

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar... ";
    cin.ignore();
    cin.get();

    return 0;
}

```

# Programa: bloque6\_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:30:50

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ MATRIZ PERSONALIZADA █
 * =====
 *
 * Bloque 6: Ejercicio 1
 * -----
 * Programa para llenar una matriz pidiendo al usuario el número de filas
 * y columnas. Posteriormente muestra la matriz en pantalla.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_ROW = "\033[1;35m"; // Magenta brillante
const string COLOR_COL = "\033[1;34m"; // Azul brillante
const string COLOR_VALUE = "\033[1;37m"; // Blanco brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████\n";
    cout << "█      MATRIZ PERSONALIZADA      █\n";
    cout << "██████████ ██████████\n\n" <<
COLOR_RESET;
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    const int MAX = 20; // Tamaño máximo reducido para mejor
    visualización
    int matriz[MAX][MAX]; // Matriz de tamaño máximo 20x20
    int filas, columnas; // Dimensiones reales

    // Explicación del programa
    cout << "█ DESCRIPCIÓN: Este programa crea una matriz con el
    tamaño\n";
    cout << "          que especifique el usuario y la muestra en
    pantalla.\n\n";

    // Solicitar dimensiones de la matriz con validación
    do {
```

```

        cout << COLOR_HIGHLIGHT << "■ Ingrese el número de filas (1-"
MAX << " " << COLOR_INPUT;
        cin >> filas;
        cout << COLOR_RESET;

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

    cout << "\n"; // Salto de línea entre inputs

    do {
        cout << COLOR_HIGHLIGHT << "■ Ingrese el número de columnas (1-"
<< MAX << " " << COLOR_INPUT;
        cin >> columnas;
        cout << COLOR_RESET;

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

    cout << "\n"; // Espacio adicional antes de ingresar datos

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

    for(int i = 0; i < filas; i++) {
        cout << "\n" << COLOR_ROW << "■ Fila [" << i << "]:" <<
COLOR_RESET << "\n";

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

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

    // Crear el encabezado de columnas con mejor formato
    cout << " ";
    for(int j = 0; j < columnas; j++) {
        cout << COLOR_COL << setw(4) << "Col " << j << " " << COLOR_RESET;
    }
    cout << "\n";

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

    // Mostrar los datos con índices de fila con mejor formato
    for(int i = 0; i < filas; i++) {
        cout << COLOR_ROW << " Fila " << setw(1) << i << " | " <<
COLOR_RESET;
        for(int j = 0; j < columnas; j++) {
            // Cambiar el color según si es un valor en la diagonal, para
destacarla
            if (i == j) {
                cout << COLOR_HIGHLIGHT;
            } else {
                cout << COLOR_VALUE;
            }
            cout << setw(7) << matriz[i][j] << " " << COLOR_RESET;
        }
        cout << "\n";
    }
}

```

```

        // Agregar una línea horizontal después de cada fila para mejorar
        la legibilidad
        if (i < filas - 1) {
            cout << "|";
            for(int j = 0; j < columnas; j++) {
                cout << "-----";
            }
            cout << "\n";
        }
    }

    cout << "\n" << COLOR_HIGHLIGHT << "■ Matriz creada y mostrada con
éxito." << COLOR_RESET << "\n\n";

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar...";
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

```

MATRIZ PERSONALIZADA

DECRIPCION: Este programa crea una matriz con el tamaño
que especifique el usuario y la muestra en pantalla.

Ingrese el número de filas (1-20): 2
Ingrese el número de columnas (1-20): 2
INGRESO DE DATOS DE LA MATRIZ (2x2):
=====

Fila [0]:
Elemento [0][0]: 1 2
Elemento [0][1]:
Fila [1]:
Elemento [1][0]: 2 1
Elemento [1][1]:
MATRIZ RESULTANTE (2x2):
=====

Col 0 Col 1
-----
Fila 0 | 1 2
         |
Fila 1 | 2 1

Matriz creada y mostrada con éxito.

Presione Enter para finalizar...
=====

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 23:31:22  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █■ DIAGONAL PRINCIPAL DE MATRIZ █■
 * =====
 *
 * Bloque 6: Ejercicio 2
 * -----
 * Programa que define una matriz de 3x3 y muestra la diagonal principal.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "████████████████████████████████████████████████\n";
    cout << "█■      DIAGONAL PRINCIPAL DE MATRIZ      █■\n";
    cout << "████████████████████████████████████████████████\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaracion de variables
    const int FILAS = 3;
    const int COLUMNAS = 3;
    int matriz[FILAS][COLUMNAS];

    // Explicación del programa
    cout << "█■ DESCRIPCIÓN: Este programa crea una matriz de 3x3 y\n";
    cout << "           muestra su diagonal principal.\n\n";

    // Solicitar los elementos de la matriz
    cout << "█■ INGRESO DE DATOS DE LA MATRIZ (3x3):\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 completa con formato mejorado
    cout << "\n█■ MATRIZ COMPLETA:\n";
    cout << "=====*\n\n";

    // Mostrar encabezados de columnas
}
```

## ***Resultado de la Ejecución:***

```
yoquelvisabreu -- zsh -- 120x40

DIAGONAL PRINCIPAL DE MATRIZ

[?] DESCRIPCIÓN: Este programa crea una matriz de 3x3 y
muestrala su diagonal principal.

[?] INGRESO DE DATOS DE LA MATRIZ (3x3):
=====
Elemento [0][0]: 3 3
Elemento [0][1]: Elemento [0][2]: 20 20
Elemento [1][0]: Elemento [1][1]: 20
Elemento [1][2]: 20
Elemento [2][0]: 20 20
Elemento [2][1]: Elemento [2][2]: 30 30

[?] MATRIZ COMPLETA:
=====
      0   1   2
-----
0 |   3   3   20
1 |   20  20   20
2 |   20   20   30

[?] DIAGONAL PRINCIPAL:
=====
[3]
[20]
[30]

[?] Elementos de la diagonal: [ 3, 20, 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 23:31:32

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ COPIADO DE MATRIZ █
 * =====
 *
 * Bloque 6: Ejercicio 3
 * -----
 * Programa que crea una matriz entera de 2x2, la llena de números
 * y luego copia todo su contenido hacia otra matriz.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "██████████ ██████████\n";
    cout << "█      COPIADO DE MATRIZ      █\n";
    cout << "██████████ ██████████\n\n";
}

// Función para mostrar una matriz con formato
void mostrarMatriz(int matriz[2][2], string titulo) {
    cout << titulo << ":\n";
    cout << string(titulo.length()+1, '=') << "\n\n";

    cout << "     0      1\n";
    cout << " +-----+\n";

    for(int i = 0; i < 2; i++) {
        cout << i << " | ";
        for(int j = 0; j < 2; j++) {
            cout << setw(4) << matriz[i][j] << " ";
        }
        cout << " |\n";
    }
    cout << " +-----+\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    int matriz1[2][2]; // Matriz original
    int matriz2[2][2]; // Matriz donde se copiará

    // Explicación del programa
```

```

cout << "■ DESCRIPCIÓN: Este programa crea una matriz de 2x2,\n";
cout << "           la llena con valores y copia su contenido a otra\n"
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 -- 120x40
[1] COPIADO DE MATRIZ

[2] DESCRIPCIÓN: Este programa crea una matriz de 2x2,
    la llena con valores y copia su contenido a otra matriz.

[3] INGRESO DE DATOS DE LA MATRIZ ORIGEN (2x2):
=====
Elemento [0][0]: 10 10
Elemento [0][1]:   Elemento [1][0]: 10 10
Elemento [1][1]:
[4] MATRIZ ORIGINAL:
=====

      0      1
+-----+
0 |  10  10 |
1 |  10  10 |
+-----+

[5] Copiando matriz...
[6] MATRIZ COPIADA:
=====

      0      1
+-----+
0 |  10  10 |
1 |  10  10 |
+-----+

[7] 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 23:31:50

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ MATRIZ CON NÚMEROS ALEATORIOS █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_ROW = "\033[1;35m"; // Magenta brillante
const string COLOR_COL = "\033[1;34m"; // Azul brillante
const string COLOR_VALUE = "\033[1;37m"; // Blanco brillante
const string COLOR_COPY = "\033[1;31m"; // Rojo brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
    cout << "█      MATRIZ CON NÚMEROS ALEATORIOS      █\n";
    cout << "██████████ ██████████ ██████████ ██████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar una matriz con formato mejorado
void mostrarMatriz(const int matriz[50][50], int filas, int columnas,
string titulo, string color) {
    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Encabezado de columnas
    cout << "      ";
    for(int j = 0; j < columnas; j++) {
        cout << COLOR_COL << setw(4) << "Col " << j << " " << COLOR_RESET;
    }
    cout << "\n";

    // Línea separadora
    cout << "      ";
}
```

```

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

        // Contenido de la matriz
        for(int i = 0; i < filas; i++) {
            // Índice de fila
            cout << COLOR_ROW << " Fila " << setw(1) << i << " | " <<
COLOR_RESET;

            // Valores de la fila
            for(int j = 0; j < columnas; j++) {
                cout << color << setw(7) << matriz[i][j] << " " << COLOR_RESET;
            }
            cout << "\n";

            // Separador entre filas (excepto después de la última)
            if(i < filas - 1) {
                cout << " | ";
                for(int j = 0; j < columnas; j++) {
                    cout << "-----";
                }
                cout << "\n";
            }
        }
    }

    // Función para mostrar una animación del proceso de copia
    void mostrarAnimacionCopia() {
        cout << "\n" << COLOR_COPY;
        cout << " ↓↓↓↓↓↓↓↓↓↓\n";
        cout << " █████████████████████████████████\n";
        cout << " █ COPIANDO DATOS... █\n";
        cout << " ██████████████████████████████\n";
        cout << " ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑\n" << COLOR_RESET;
    }

    int main() {
        // Limpiar pantalla y mostrar banner
        limpiarPantalla();
        mostrarBanner();

        // Declaración de variables
        const int MAX = 20;           // Tamaño máximo para mejor visualización
        int matriz1[50][50];         // Primera matriz
        int matriz2[50][50];         // Segunda matriz (copia)
        int filas, columnas;

        // Explicación del programa
        cout << "■ DESCRIPCIÓN: Este programa crea una matriz con números
aleatorios.\n";
        cout << "                   copia el contenido a otra matriz y muestra ambas
matrices.\n\n";

        // Pedir dimensiones de la matriz con validación
        do {
            cout << COLOR_HIGHLIGHT << "■ Ingrese el número de filas (1-" <<
MAX << "): " << COLOR_INPUT;
            cin >> filas;
            cout << COLOR_RESET;

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

        cout << "\n"; // Salto de línea entre inputs

        do {
            cout << COLOR_HIGHLIGHT << "■ Ingrese el número de columnas (1-" <<
MAX << "): " << COLOR_INPUT;
            cin >> columnas;
            cout << COLOR_RESET;

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

```

```

        }
    } while(columnas <= 0 || columnas > MAX);

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

    // Informar al usuario
    cout << "\n" << COLOR_HIGHLIGHT << "■ Generando matriz " << filas <<
    "x" << columnas << " con números aleatorios..." << COLOR_RESET << endl;

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

    // Mostrar la matriz original
    mostrarMatriz(matriz1, filas, columnas, "■ MATRIZ ORIGINAL:",
    COLOR_VALUE);

    // Mostrar animación de copia
    mostrarAnimacionCopia();

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

    // Mostrar la matriz copia
    mostrarMatriz(matriz2, filas, columnas, "■ MATRIZ COPIA:",
    COLOR_COPY);

    // Añadir instrucciones finales
    cout << "\n" << COLOR_HIGHLIGHT << "■ Proceso completado con éxito" <<
    COLOR_RESET << "\n\n";
    cout << "Presione Enter para finalizar...";
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

```
yoquelvisabreu -- zsh -- 120x40

DESPACIO: Este programa crea una matriz con números aleatorios,
copia el contenido a otra matriz y muestra ambas matrices.

Ingrese el número de filas (1-20): 2
Ingrese el número de columnas (1-20): 2
Generando matriz 2x2 con números aleatorios...

MATRIZ ORIGINAL:
Col 0 Col 1
-----
Fila 0 | 50    95
        |
Fila 1 | 14    23
        ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
        ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

COPIANDO DATOS...

MATRIZ COPIA:
Col 0 Col 1
-----
Fila 0 | 50    95
        |
Fila 1 | 14    23

✓ Proceso completado con éxito
Presione Enter para finalizar...
=====
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 23:32:23

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * ███ MATRIZ TRANSPUESTA █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_ROW = "\033[1;35m"; // Magenta brillante
const string COLOR_COL = "\033[1;34m"; // Azul brillante
const string COLOR_VALUE = "\033[1;37m"; // Blanco brillante
const string COLOR_TRANS = "\033[1;31m"; // Rojo brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "████████████████████████████████████████████████\n";
    cout << "█      MATRIZ TRANSPUESTA      █\n";
    cout << "████████████████████████████████████████████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar una matriz con formato mejorado
void mostrarMatriz(const int matriz[3][3], string titulo, string color) {
    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Encabezado de columnas
    cout << "    ";
    for(int j = 0; j < 3; j++) {
        cout << COLOR_COL << setw(4) << "Col " << j << " " << COLOR_RESET;
    }
    cout << "\n";

    // Línea separadora
    cout << "    ";
    for(int j = 0; j < 3; j++) {
        cout << "-----";
    }
    cout << "\n";
}
```



```

        mostrarMatriz(transpuesta, "■ MATRIZ TRANSPUESTA:", COLOR_TRANS);
        // Explicación visual de la transposición
        cout << "\n" << COLOR_HIGHLIGHT << "■ EXPLICACIÓN:" << COLOR_RESET <<
endl;
        cout << "El elemento [i][j] de la matriz original se convierte en\n";
        cout << "el elemento [j][i] de la matriz transpuesta.\n";
        cout << "Por ejemplo: matriz[0][2] = " << matriz[0][2] << " →
transpuesta[2][0] = " << transpuesta[2][0] << "\n";
        // Añadir instrucciones finales
        cout << "\n" << COLOR_HIGHLIGHT << "■ Transposición completada con
éxito" << COLOR_RESET << "\n\n";
        cout << "Presione Enter para finalizar..." ;
        cin.ignore();
        cin.get();

    return 0;
}

```

## **Resultado de la Ejecución:**

# Programa: bloque7\_ejercicio1

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:32:49

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ VERIFICADOR DE LONGITUD DE CADENAS █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_SUCCESS = "\033[1;92m"; // Verde claro brillante
const string COLOR_ERROR = "\033[1;91m"; // Rojo claro brillante
const string COLOR_INFO = "\033[1;94m"; // Azul claro brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ VERIFICADOR DE LONGITUD DE CADENAS ██████████\n";
    cout << "\n\n" <<
COLOR_RESET;
}

// Función para mostrar una cadena con formato visual
void mostrarCadena(const char* cadena, int longitud) {
    cout << COLOR_HIGHLIGHT << "\n█ ANÁLISIS DE LA CADENA:\n";
    cout << "===== \n" << COLOR_RESET;

    cout << COLOR_INFO << "█";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
    cout << "█\n";

    cout << "█ " << cadena << " █\n";

    cout << "█ ";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
    cout << "█ " << COLOR_RESET << "\n\n";
}
```

```

    cout << "■ Longitud: " << COLOR_HIGHLIGHT << longitud << " caracteres"
<< COLOR_RESET << "\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    char cadena[100]; // Arreglo para almacenar la cadena
    int longitud; // Para almacenar la longitud de la cadena

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa verifica si una cadena tiene más
de 10 caracteres\n";
    cout << "                                y la muestra solo si cumple esta condición.\n\n";

    // Pedir la cadena al usuario con formato mejorado
    cout << COLOR_HIGHLIGHT << "■ Digite una cadena de caracteres: " <<
COLOR_INPUT;
    cin.getline(cadena, 100, '\n');
    cout << COLOR_RESET;

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

    // Verificar si la longitud supera 10 caracteres
    if(longitud > 10) {
        cout << "\n" << COLOR_SUCCESS << "■ La cadena tiene más de 10
caracteres." << COLOR_RESET;
        mostrarCadena(cadena, longitud);
    } else {
        cout << "\n" << COLOR_ERROR << "■■■ La cadena tiene " << longitud
<< " caracteres, no se mostrará el contenido." << COLOR_RESET << "\n";
        cout << COLOR_INFO << " La cadena debe tener más de 10 caracteres
para ser mostrada." << COLOR_RESET << "\n";
    }

    // Añadir instrucciones finales
    cout << "\nPresione Enter para finalizar...";
    cin.get();
}

return 0;
}

```

### **Resultado de la Ejecución:**

VERIFICADOR DE LONGITUD DE CADENAS

DESCRIPCIÓN: Este programa verifica si una cadena tiene más de 10 caracteres y la muestra solo si cumple esta condición.

Digite una cadena de caracteres: yoquelvis

⚠ La cadena tiene 9 caracteres, no se mostrará el contenido.  
La cadena debe tener más de 10 caracteres para ser mostrada.

Presione Enter para finalizar...

=====

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 23:33:22

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ COPIA DE CADENAS DE CARACTERES █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_ORIGINAL = "\033[1;34m"; // Azul brillante
const string COLOR_COPY = "\033[1;35m"; // Magenta brillante
const string COLOR_INFO = "\033[1;37m"; // Blanco brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ COPIA DE CADENAS DE CARACTERES ██████████\n";
    cout << "██████████\n" << COLOR_RESET;
}

// Función para mostrar una cadena con formato visual
void mostrarCadena(const char* cadena, const char* titulo, string color) {
    int longitud = strlen(cadena);

    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Mostrar cadena con formato de tabla
    // Línea superior
    cout << color << "█";
    for(int i = 0; i < longitud; i++) {
        cout << "████";
    }
    cout << "█\n";

    // Índices
    cout << "█";
    for(int i = 0; i < longitud; i++) {
```



```

cout << "3. Se utilizó strcpy() para copiar cada carácter al segundo
arreglo.\n";
cout << "4. La copia incluye automáticamente el carácter nulo '\\0' al
final.\n\n";

// Verificación de la copia
bool sonIguales = (strcmp(cadena1, cadena2) == 0);
cout << COLOR_HIGHLIGHT << "■ VERIFICACIÓN: " << COLOR_RESET;

if(sonIguales) {
    cout << "Las cadenas son idénticas, la copia fue exitosa.\n\n";
} else {
    cout << "Error: Las cadenas no son idénticas.\n\n";
}

// Añadir instrucciones finales
cout << "Presione Enter para finalizar...";
cin.get();

return 0;
}

```

### **Resultado de la Ejecución:**

**CADENA ORIGINAL:**

0	1	2	3	4	5	6	7	8
y	o	q	u	e	l	v	i	s

Longitud: 9 caracteres  
Dirección de memoria: 0x16d24b224

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

**COPIANDO CARACTERES...**

**CADENA COPIADA:**

0	1	2	3	4	5	6	7	8
y	o	q	u	e	l	v	i	s

Longitud: 9 caracteres  
Dirección de memoria: 0x16d24b1c0

**EXPLICACIÓN DEL PROCESO:**

1. Se reservó espacio para dos arreglos de 100 caracteres.
2. Se leyó la cadena del usuario al primer arreglo.
3. Se utilizó strcpy() para copiar cada carácter al segundo arreglo.
4. La copia incluye automáticamente el carácter nulo '\0' al final.

**VERIFICACIÓN:** Las cadenas son idénticas, la copia fue exitosa.

Presione Enter para finalizar...

=====

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 23:34:00

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ COMPARADOR DE CADENAS █
 * =====
 *
 * Bloque 7: Ejercicio 3
 * -----
 * Programa que pide al usuario que digite 2 cadenas de caracteres,
 * e indica si ambas cadenas son iguales. En caso de no serlo,
 * indica cuál es mayor alfabéticamente.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_STRING1 = "\033[1;34m"; // Azul brillante
const string COLOR_STRING2 = "\033[1;35m"; // Magenta brillante
const string COLOR_TABLE = "\033[1;37;44m"; // Blanco brillante sobre azul
const string COLOR_RESULT = "\033[1;31m"; // Rojo brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "████████████████████████████████████████████████\n";
    cout << "█      COMPARADOR DE CADENAS      █\n";
    cout << "████████████████████████████████████████████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar una cadena con formato visual
void mostrarCadena(const char* cadena, const char* titulo, string color) {
    int longitud = strlen(cadena);

    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

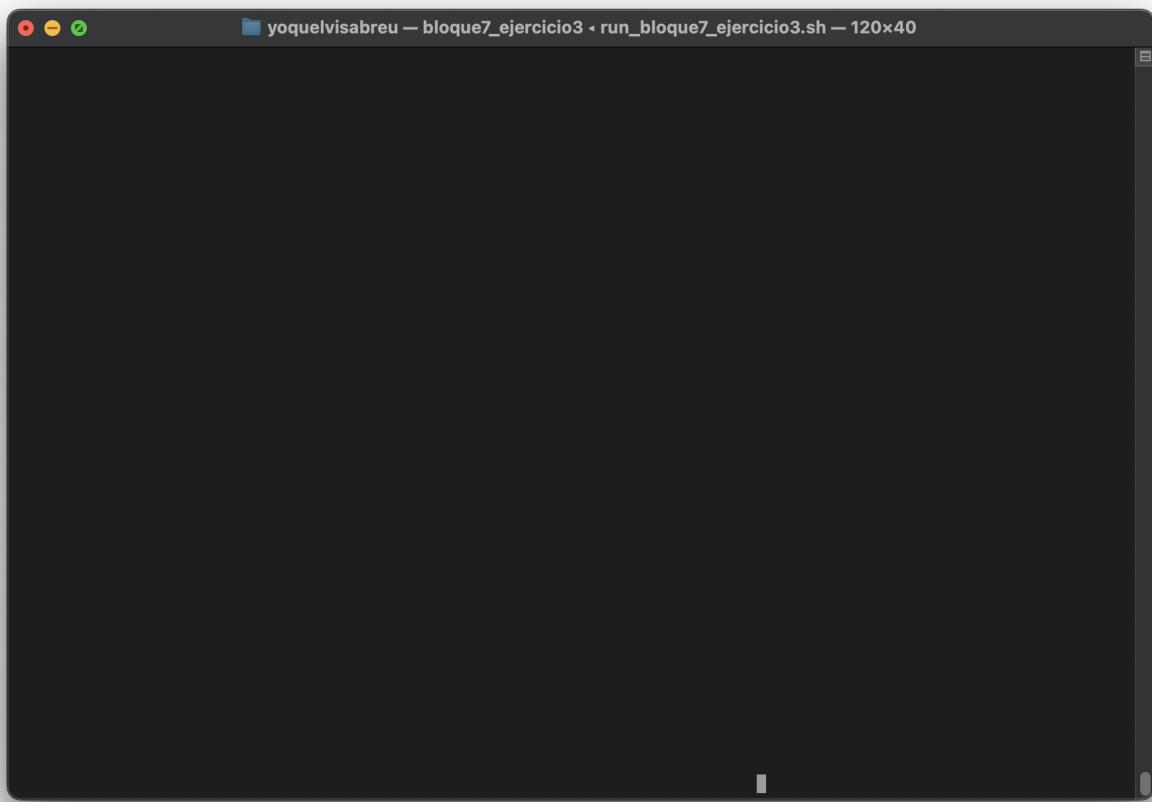
    // Mostrar cadena con formato de caja
    cout << color << "█";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
    cout << "█\n";

    cout << "█ " << cadena << " █\n";

    cout << "█ ";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
    cout << "█ " << COLOR_RESET << "\n";
}
```



### **Resultado de la Ejecución:**



# Programa: bloque7\_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:34:37

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ CONCATENACIÓN DE CADENAS █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */
#include <iostream>
#include <iomanip>
#include <string.h> // Para strcat()
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_STRING1 = "\033[1;34m"; // Azul brillante
const string COLOR_STRING2 = "\033[1;35m"; // Magenta brillante
const string COLOR_RESULT = "\033[1;31m"; // Rojo brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████\n";
    cout << "██      CONCATENACIÓN DE CADENAS      █\n";
    cout << "██████████ ██████████ ██████████\n\n" <<
COLOR_RESET;
}

// Función para mostrar una cadena con formato visual
void mostrarCadena(const char* cadena, const char* titulo, string color) {
    int longitud = strlen(cadena);

    cout << COLOR_HIGHLIGHT << "\n" << titulo << COLOR_RESET << endl;

    // Mostrar cadena con formato de caja
    cout << color << "█";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
    cout << "█\n";

    cout << "█ " << cadena << " █\n";

    cout << "█";
    for(int i = 0; i < longitud + 2; i++) cout << "█";
}
```

```

cout << "■" << COLOR_RESET << "\n";
// Información adicional
cout << " Longitud: " << longitud << " caracteres\n";
}

// Función para mostrar una animación del proceso de concatenación
void mostrarAnimacionConcatenacion() {
    cout << "\n" << COLOR_HIGHLIGHT;
    cout << "↓↓↓↓↓↓↓↓\n";
    cout << " CONCATENANDO CADENAS... " << "\n";
    cout << "↑↑↑↑↑↑↑↑↑↑↑↑↑↑\n" << COLOR_RESET;
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    char mensaje[100] = "Hola que tal"; // Mensaje inicial
    char nombre[50]; // Para almacenar el nombre del usuario

    // Explicación del programa
    cout << "■ DESCRIPCIÓN: Este programa concatena (une) el nombre del
    usuario\n";
    cout << "                                al final de un mensaje predefinido.\n\n";

    // Mostrar el mensaje inicial
    mostrarCadena(mensaje, "■ MENSAJE INICIAL:", COLOR_STRING1);

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

    // Mostrar el nombre ingresado
    mostrarCadena(nombre, "■ NOMBRE INGRESADO:", COLOR_STRING2);

    // Mostrar animación de concatenación
    mostrarAnimacionConcatenacion();

    // Añadir espacio antes de concatenar
    strcat(mensaje, " ");

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

    // Mostrar el mensaje completo
    mostrarCadena(mensaje, "■ MENSAJE COMPLETO:", COLOR_RESULT);

    // Explicación del proceso
    cout << "\n" << COLOR_HIGHLIGHT << "■ EXPLICACIÓN DEL PROCESO:" <<
    COLOR_RESET << endl;
    cout << "1. Se creó una primera cadena con el texto \"Hola que
    tal\".\n";
    cout << "2. Se obtuvo el nombre del usuario: " << nombre << ".\n";
    cout << "3. Se añadió un espacio en blanco a la primera cadena.\n";
    cout << "4. Se utilizó strcat() para unir las dos cadenas.\n";
    cout << "5. El resultado final combina ambos textos.\n\n";

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar...";
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**



# Programa: bloque7\_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:35:14

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * ────────── VERIFICADOR DE PALÍNDROMOS ──────────
 * =====
 *
 * Bloque 7: Ejercicio 5
 * -----
 * Programa que determina si una palabra es palíndroma.
 * (Se lee igual de izquierda a derecha que de derecha a izquierda)
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_NORMAL = "\033[1;34m"; // Azul brillante
const string COLOR_INVERSE = "\033[1;35m"; // Magenta brillante
const string COLOR_MATCH = "\033[1;32m"; // Verde brillante
const string COLOR_MISMATCH = "\033[1;31m"; // Rojo brillante
const string COLOR_RESULT = "\033[1;37m"; // Blanco brillante

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
    cout << "██████████ VERIFICADOR DE PALÍNDROMOS ██████████\n";
    cout << "██████████ ██████████ ██████████ ██████████\n" <<
COLOR_RESET;
}

// Función para convertir una cadena a minúsculas y sin espacios
void limpiarCadena(const char* original, char* limpia) {
    int j = 0;

    for(int i = 0; i < strlen(original); i++) {
        if(isalpha(original[i])) {
            limpia[j++] = tolower(original[i]);
        }
    }

    limpia[j] = '\0'; // Agregar terminador de cadena
}

// Función para mostrar la animación de verificación
void mostrarAnimacionVerificacion(const char* palabra, bool esPalindromo)
```

```

{
    int longitud = strlen(palabra);
    int mitad = longitud / 2;

    cout << "\n" << COLOR_HIGHLIGHT << "■ VERIFICACIÓN CARÁCTER POR
CARÁCTER:\n";
    cout << "=====\\n" << COLOR_RESET;

    // Mostrar la verificación de cada par de caracteres
    for(int i = 0; i < mitad; i++) {
        int j = longitud - 1 - i;

        cout << "Comparando: ";

        // Posición izquierda
        cout << "Posición " << setw(2) << i << " [";
        if(palabra[i] == palabra[j]) {
            cout << COLOR_MATCH << palabra[i] << COLOR_RESET;
        } else {
            cout << COLOR_MISMATCH << palabra[i] << COLOR_RESET;
        }
        cout << "] ";

        // Flecha de comparación
        if(palabra[i] == palabra[j]) {
            cout << COLOR_MATCH << "==" << COLOR_RESET;
        } else {
            cout << COLOR_MISMATCH << "!=" << COLOR_RESET;
        }

        // Posición derecha
        cout << " Posición " << setw(2) << j << " [";
        if(palabra[i] == palabra[j]) {
            cout << COLOR_MATCH << palabra[j] << COLOR_RESET;
        } else {
            cout << COLOR_MISMATCH << palabra[j] << COLOR_RESET;
        }
        cout << "] ";

        // Resultado de la comparación
        if(palabra[i] == palabra[j]){
            cout << COLOR_MATCH << "✓ Coincide" << COLOR_RESET;
        } else {
            cout << COLOR_MISMATCH << "✗ No coincide" << COLOR_RESET;
        }
    }

    cout << "\\n";
}

// Resultado final
cout << "\\n" << COLOR_HIGHLIGHT << "■ RESULTADO: " << COLOR_RESET;
if(esPalindromo) {
    cout << COLOR_MATCH << "■ Todas las comparaciones coinciden, es un
palíndromo.\\n" << COLOR_RESET;
} else {
    cout << COLOR_MISMATCH << "■ Al menos una comparación no coincide,
no es un palíndromo.\\n" << COLOR_RESET;
}
}

// Función para mostrar una cadena con formato visual
void mostrarTexto(const char* texto, const char* titulo, string color) {
    cout << COLOR_HIGHLIGHT << "\\n" << titulo << COLOR_RESET << endl;

    // Mostrar texto con formato de caja
    cout << color << "■";
    for(int i = 0; i < strlen(texto) + 2; i++) cout << "■";
    cout << "■\\n";

    cout << "■ " << texto << " ■\\n";

    cout << "■ ";
    for(int i = 0; i < strlen(texto) + 2; i++) cout << "■";
    cout << "■" << COLOR_RESET << "\\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();
}

```

```

// Declaracion de variables
char palabra[100];
char palabraLimpia[100];
bool esPalindromo = true;

// Explicación del programa
cout << "■ DESCRIPCIÓN: Este programa verifica si una palabra o
frase\n";
cout << "          es un palíndromo (se lee igual en ambos
sentidos).\n";
cout << "          Ejemplos: ana, reconocer, anita lava la
tina\n\n";

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

// 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" << COLOR_HIGHLIGHT << "■ ANÁLISIS DE PALÍNDROMO:\n";
cout << "===== \n" << COLOR_RESET;

// Mostrar textos con formato
mostrarTexto(palabra, "■ TEXTO ORIGINAL:", COLOR_NORMAL);
mostrarTexto(palabraLimpia, "■ TEXTO PROCESADO (sin espacios, en
minúsculas):", COLOR_NORMAL);

// Visualización del palíndromo
cout << "\n" << COLOR_HIGHLIGHT << "■ VISUALIZACIÓN BIDIRECCIONAL:\n";
cout << "-----\n" << COLOR_RESET;

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

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

// Mostrar verificación carácter por carácter
mostrarAnimacionVerificacion(palabraLimpia, esPalindromo);

// Mostrar el resultado final
cout << "\n" << COLOR_HIGHLIGHT << "■ CONCLUSIÓN: " << COLOR_RESET;
if(esPalindromo) {
    cout << COLOR_MATCH << "■ \\" << palabra << "\"" SÍ es un
palíndromo!" << COLOR_RESET << "\n\n";
} else {
    cout << COLOR_MISMATCH << "■ \\" << palabra << "\"" NO es un
palíndromo." << COLOR_RESET << "\n\n";
}

// Añadir instrucciones finales
cout << "Presione Enter para finalizar... ";
cin.get();

return 0;
}

```

## Resultado de la Ejecución:

```
yoquelvisabreu -- zsh -- 120x40
es un palíndromo (se lee igual en ambos sentidos).
Ejemplos: ana, reconocer, anita lava la tina

abc Ingrese una palabra o frase: test

📊 ANÁLISIS DE PALÍNDROMO:
=====
📝 TEXTO ORIGINAL:
test

✔ TEXTOS PROCESADOS (sin espacios, en minúsculas):
test

🕒 VISUALIZACIÓN BIDIRECCIONAL:
-----
Normal: t e s t
Inversa: t s e t

🔍 VERIFICACIÓN CARÁCTER POR CARÁCTER:
=====
Comparando: Posición 0 [t] == Posición 3 [t] ✓ Coincide
Comparando: Posición 1 [e] != Posición 2 [s] ✗ No coincide

❗️ RESULTADO: ✗ Al menos una comparación no coincide, no es un palíndromo.

✗ CONCLUSIÓN: ✗ "test" NO es un palíndromo.

Presione Enter para finalizar...

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

# Programa: bloque8\_ejercicio1

Compilador: g++  
Flags: -Wall -std=c++11  
Fecha: 2025-03-11 23:35:52  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE GESTIÓN DE CORREDORES █
 * =====
 *
 * Bloque 8: Ejercicio 1
 * -----
 * Este programa gestiona la información de corredores y determina
 * automáticamente su categoría de competición basada en la edad:
 *
 * █ Categorías:
 * - █ Juvenil: ≤ 18 años
 * - █ Senior: ≤ 40 años
 * - █ Veterano: > 40 años
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Definición de la estructura corredor
struct Corredor {
    char nombre[50];
    int edad;
    char sexo[10];
    char club[30];
    char categoria[20];
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "██████████ REGISTRO DE CORREDORES v1.0 ██████████\n";
    cout << "████████████████████████████████████████████████████████\n";
}

int main() {
    Corredor corredor1;

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

    // Pedir datos al usuario con formato mejorado
    cout << "█ Por favor, ingrese los datos del corredor:\n";
    cout << "===== \n\n";

    cout << "█ Nombre: ";
    cin.getline(corredor1.nombre, 50, '\n');
}
```

```

cout << "\n"; // Salto de línea entre inputs

// Validar la edad
bool edad_valida = false;
do {
    cout << "■ Edad: ";
    cin >> corredor1.edad;

    if(cin.fail()) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "\n■■ Error: Por favor ingrese un número válido para
la edad.\n\n";
    }
    else if(corredor1.edad <= 0 || corredor1.edad > 120) {
        cout << "\n■■ Error: La edad debe estar entre 1 y 120
años.\n\n";
    }
    else {
        edad_valida = true;
    }
} while(!edad_valida);

cin.ignore(); // Limpiar buffer después de leer la edad

cout << "\n"; // Salto de línea entre inputs

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

cout << "\n"; // Salto de línea entre inputs

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

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

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

// Añadir instrucciones finales
cout << "Presione Enter para finalizar... ";
cin.get();

return 0;
}

```

### **Resultado de la Ejecución:**

```
REGISTRO DE CORREDORES v1.0

FICHA DEL CORREDOR
=====
👤 Nombre: yoquelvis
🎂 Edad: 22 años
🚹 Sexo: Masculino
🏟 Club: leones
🏆 Categoría:Senior 💪

⭐ ¡Registro completado con éxito! ⭐

Presione Enter para finalizar...

=====
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 23:36:29

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE GESTIÓN ACADÉMICA █
 * =====
 *
 * Bloque 8: Ejercicio 2
 * -----
 * Este programa gestiona información de estudiantes y determina
 * automáticamente quién tiene el mejor rendimiento académico.
 * Permite registrar datos de 3 alumnos y muestra los detalles
 * del estudiante con el promedio más alto.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Definición de la estructura alumno
struct Alumno {
    char nombre[50];
    int edad;
    float promedio;
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "██████████ REGISTRO DE ESTUDIANTES v1.0 ██████████\n";
    cout << "██████████\n\n";
}

// Función para mostrar una barra de progreso
void mostrarProgreso(int actual, int total) {
    cout << "\n█ Progreso: [";
    int porcentaje = (actual * 20) / total;
    for(int i = 0; i < 20; i++) {
        if(i < porcentaje) cout << "█";
        else cout << "█";
    }
    cout << "] " << (actual * 100) / total << "%\n\n";
}

// Función para validar la edad (entre 6 y 100 años)
int pedirEdad(const char* mensaje) {
    int edad;
    bool valido = false;
    do {
        cout << mensaje;
```

```

        cin >> edad;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\n■■■ Error: Por favor ingrese un número válido para
la edad.\n\n";
        }
        else if(edad < 6 || edad > 100) {
            cout << "\n■■■ Error: La edad debe estar entre 6 y 100
años.\n\n";
        }
        else {
            valido = true;
        }
    } while(!valido);

    return edad;
}

// Función para validar el promedio (entre 0 y 10)
float pedirPromedio(const char* mensaje) {
    float promedio;
    bool valido = false;

    do {
        cout << mensaje;
        cin >> promedio;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\n■■■ Error: Por favor ingrese un número válido para
el promedio.\n\n";
        }
        else if(promedio < 0 || promedio > 10) {
            cout << "\n■■■ Error: El promedio debe estar entre 0 y
10.\n\n";
        }
        else {
            valido = true;
        }
    } while(!valido);

    return promedio;
}

int main() {
    Alumno alumnos[3];
    int indice_mayor = 0;
    float mayor_promedio = 0;

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

    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";

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

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

        cout << "\\n"; // Salto de línea entre inputs
        alumnos[i].edad = pedirEdad("■ Edad: ");

        cout << "\\n"; // Salto de línea entre inputs
        alumnos[i].promedio = pedirPromedio("■ Promedio (0-10): ");

        // Verificar si este alumno tiene el mayor promedio
    }
}

```

## **Resultado de la Ejecución:**

yoquelvisabreu — run bloque8\_ejercicio2.sh — 120x40

```
REGISTRO DE ESTUDIANTES v1.0

ESTUDIANTE DESTACADO
=====
Nombre: yoquelvis
Edad: 20 años
Promedio: 10.00

¡Felicitaciones! ¡Medalla de Oro! 🥇

RESUMEN DE TODOS LOS ESTUDIANTES:
=====



| NUM | NOMBRE    | EDAD | PROMEDIO | MEDALLA |
|-----|-----------|------|----------|---------|
| 1   | yoquelvis | 20   | 10.00    | 🥇       |
| 2   | yoquelvis | 20   | 10.00    | 🥇       |
| 3   | jorge     | 20   | 5.00     |         |



Presione Enter para finalizar...
=====

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

# Programa: bloque8\_ejercicio3

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:37:07

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE GESTIÓN DE PERSONAL █
 * =====
 *
 * Bloque 8: Ejercicio 3
 * -----
 * Este programa gestiona información de empleados y analiza
 * sus salarios para identificar los extremos salariales
 * en la empresa. Permite registrar N empleados y muestra
 * los detalles de quienes tienen el mayor y menor salario.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Definición de la estructura Empleado
struct Empleado {
    char nombre[50];
    char cargo[30];
    float salario;
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "████████████████████████████████████████████████\n";
    cout << "█      GESTIÓN DE RECURSOS HUMANOS v1.0      █\n";
    cout << "████████████████████████████████████████████████\n\n";
}

// Función para mostrar una barra de progreso
void mostrarProgreso(int actual, int total) {
    cout << "\n█ Progreso: [";
    int porcentaje = (actual * 20) / total;
    for(int i = 0; i < 20; i++) {
        if(i < porcentaje) cout << "█";
        else cout << "█";
    }
    cout << "] " << (actual * 100) / total << "%\n\n";
}

// Función para formatear el salario
string formatearSalario(float salario) {
    char buffer[50];
    sprintf(buffer, "$%,.2f", salario);
    return string(buffer);
}
```

```

// Función para validar el salario (mayor que cero)
float pedirSalario(const char* mensaje) {
    float salario;
    bool valido = false;

    do {
        cout << mensaje;
        cin >> salario;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\n■■■ Error: Por favor ingrese un número válido para el salario.\n\n";
        }
        else if(salario <= 0) {
            cout << "\n■■■ Error: El salario debe ser mayor que 0.\n\n";
        }
        else {
            valido = true;
        }
    } while(!valido);

    return salario;
}

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 con validación mejorada
    bool cantidad_valida = false;
    do {
        cout << "■■■ Número de empleados a registrar: ";
        cin >> n;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\n■■■ Error: Por favor ingrese un número válido.\n\n";
        }
        else if(n <= 0) {
            cout << "\n■■■ Error: Debe registrar al menos un empleado.\n\n";
        }
        else {
            cantidad_valida = true;
        }
    } while(!cantidad_valida);

    // 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 << "\\n"; // Salto de línea entre inputs

        cout << "■■■ Cargo: ";

```



```

        cout << setw(11) << left << formatearSalario(empleados[i].salario)
<< " █\n";
    }

    cout << "████████████████████████████████████████████████████████████████\n\n";

    // Liberar memoria
    delete[] empleados;

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

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar... ";
    cin.ignore();
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

The screenshot shows a terminal window titled "yoquelvisabreu — bloque8\_ejercicio3 · run\_bloque8\_ejercicio3.sh — 120x40". The window displays the output of a C++ program. The output is as follows:

```

GESTIÓN DE RECURSOS HUMANOS v1.0

ANÁLISIS SALARIAL
-----
🏆 SALARIO MÁS ALTO
-----
👤 Nombre: yoquelvis jorge
💼 Cargo: gerente
💰 Salario: $20.00

✍ SALARIO MÁS BAJO
-----
👤 Nombre: yoquelvis jorge
💼 Cargo: gerente
💰 Salario: $20.00

📝 BRECHA SALARIAL
-----
Diferencia: $0.00

LISTA COMPLETA DE SALARIOS
-----

```

NUM	NOMBRE	CARGO	SALARIO
1	yoquelvis	yoquelvis jorge gerente	\$20.00
2			\$20.00

★ Análisis completado con éxito ★  
Presione Enter para finalizar...■

# Programa: bloque8\_ejercicio4

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:37:45

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE REGISTRO DE ATLETAS █
 * =====
 *
 * Bloque 8: Ejercicio 4
 * -----
 * Programa que crea un arreglo de estructura llamada atleta para N atletas
 * que contiene: nombre, país, número de medallas, y devuelve los datos
 * (Nombre, país) del atleta que ha ganado el mayor número de medallas.
 *
 * Autor: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Definición de la estructura Atleta
struct Atleta {
    char nombre[50];
    char pais[50];
    int medallas;
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << "\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
    cout << "██      REGISTRO OLÍMPICO DE ATLETAS      ██\n";
    cout << "██████████ ██████████ ██████████ ██████████\n\n";
}

// Función para mostrar una línea de progreso
void mostrarProgreso(int actual, int total) {
    float porcentaje = (float)actual / total * 100;
    int barraLongitud = 30;
    int completo = (int)(porcentaje * barraLongitud / 100);

    cout << "\n█ Progreso: [";
    for(int i = 0; i < barraLongitud; i++) {
        if(i < completo) cout << "█";
        else cout << "█";
    }
    cout << "] " << fixed << setprecision(1) << porcentaje << "%\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();
```

```

// 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<streamsiz>::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\\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) << right << atletas[i].medallas << " | ";
    // Marcar al atleta con más medallas
    if(i == indice_mayor) {
        cout << " █";
    }
    cout << "\n";
}
cout << "-----+-----+-----+-----\n\n";
// Mostrar los datos del atleta con más medallas
cout << " █ ATLETA CON MAYOR NÚMERO DE MEDALLAS:\n";
cout << "=====|\n\n";
cout << " █ Nombre: " << atletas[indice_mayor].nombre << "\n";
cout << " █ País: " << atletas[indice_mayor].pais << "\n";
cout << " █ Medallas: " << atletas[indice_mayor].medallas << "\n\n";
// Mensaje de reconocimiento
cout << " █ i" << atletas[indice_mayor].nombre << " de "
    << atletas[indice_mayor].pais << " es el atleta más destacado";
if(atletas[indice_mayor].medallas == 1) {
    cout << " con 1 medalla! █\n\n";
} else {
    cout << " con " << atletas[indice_mayor].medallas << " medallas!
█\n\n";
}
// Liberar memoria
delete[] atletas;
return 0;
}

```

### **Resultado de la Ejecución:**

```
yoquelvisabreu -- zsh -- 120x40

REGISTRO OLÍMPICO DE ATLETAS

TABLA DE ATLETAS REGISTRADOS:
=====
+-----+-----+-----+-----+
| NUM | NOMBRE | PAÍS | MEDALLAS |
+-----+-----+-----+-----+
| 1 | juan | africa | 2 | 🏆
+-----+-----+-----+-----+

ATLETA CON MAYOR NÚMERO DE MEDALLAS:
=====
👤 Nombre: juan
🌍 País: africa
🏆 Medallas: 2

✨ ¡juan de africa es el atleta más destacado con 2 medallas! ✨

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

# Programa: bloque8\_ejercicio5

Compilador: g++

Flags: -Wall -std=c++11

Fecha: 2025-03-11 23:38:23

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE GESTIÓN DE CALIFICACIONES █
 * =====
 *
 * Bloque 8: Ejercicio 5
 * -----
 * Programa con 2 estructuras: una llamada promedio que tiene los campos
 * nota1, 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */
#include <iostream>
#include <string.h>
#include <iomanip>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_INFO = "\033[1;34m"; // Azul brillante
const string COLOR_ERROR = "\033[1;31m"; // Rojo brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante

// Definición de la estructura Promedio
struct Promedio {
    float nota1;
    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
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
    cout << "█      SISTEMA DE GESTIÓN ACADÉMICA      █\n";
    cout << "██████████ ██████████ ██████████ ██████████\n";
}
```

```

        cout << "████████████████████████████████████████████████████████████████████████████████\n\n" <<
COLOR_RESET;
}

// Función para validar una nota (0-10)
float pedirNota(const char* mensaje) {
    float nota;
    bool valido = false;

    do {
        cout << COLOR_HIGHLIGHT << mensaje << COLOR_INPUT;
        cin >> nota;
        cout << COLOR_RESET;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << COLOR_ERROR << "█ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
        }
        else if(nota < 0 || nota > 10) {
            cout << COLOR_ERROR << "█ Error: La nota debe estar entre 0 y
10.\n\n" << COLOR_RESET;
        }
        else {
            valido = true;
        }
    } while(!valido);

    return nota;
}

// Función para mostrar el nivel académico según el promedio
string obtenerNivelAcademico(float promedio) {
    if(promedio >= 9.0) return "Excelente █";
    else if(promedio >= 8.0) return "Muy Bueno █";
    else if(promedio >= 7.0) return "Bueno █";
    else if(promedio >= 6.0) return "Aprobado █";
    else return "Reprobado █";
}

// Función para mostrar un gráfico de barras para el promedio
void mostrarGraficoPromedio(float promedio) {
    int longitud = (int)(promedio * 3); // 3 caracteres por unidad

    cout << COLOR_INFO << "[ ";
    for(int i = 0; i < 30; i++) {
        if(i < longitud) {
            if(promedio < 6) cout << COLOR_ERROR << "█";
            else if(promedio < 8) cout << COLOR_HIGHLIGHT << "█";
            else cout << COLOR_RESULT << "█";
        } else {
            cout << COLOR_INFO << "█ ";
        }
    }
    cout << COLOR_INFO << "] " << setprecision(2) << fixed << promedio <<
"/10.0" << COLOR_RESET << "\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // Declaración de variables
    Alumno alumno1;

    // Explicación del programa
    cout << "█ DESCRIPCIÓN: Este programa registra los datos académicos de
un alumno,\n";
    cout << "                 calcula su promedio y muestra su nivel de
desempeño.\n\n";

    // Pedir datos del alumno con formato mejorado
    cout << COLOR_HIGHLIGHT << "█ DATOS PERSONALES DEL ALUMNO\n";
    cout << "=====*\n" << COLOR_RESET;

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

```



## **Resultado de la Ejecución:**

```
yoquelvisabreu@Laptop-de-Yoquelvis ~ % █ yoquelvisabreu --zsh-- 120x40

SISTEMA DE GESTIÓN ACADÉMICA

FICHA ACADÉMICA DEL ALUMNO
=====
=====

DATOS PERSONALES
=====
=====
Nombre: yoquelvis
Sexo: Femenino
Edad: 20 años

CALIFICACIONES
=====
=====
Nota 1: 10.00
Nota 2: 10.00
Nota 3: 10.00

RESULTADO FINAL
=====
=====
Promedio: 10.00
Nivel: Excelente 🏆

REPRESENTACIÓN GRÁFICA DEL PROMEDIO:
[██████████] 10.00/10.0

† Registro académico completado con éxito †

Presione Enter para finalizar...
=====
=====
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 23:39:01

Estado: Ejecución exitosa

Salida:

La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * █ SISTEMA DE RANKING ACADÉMICO █
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

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

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_INFO = "\033[1;34m"; // Azul brillante
const string COLOR_ERROR = "\033[1;31m"; // Rojo brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante
const string COLOR_SUCCESS = "\033[1;92m"; // Verde claro brillante

// Definición de la estructura Promedio
struct Promedio {
    float nota1;
    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
};

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "█ SISTEMA DE RANKING ACADÉMICO █\n";
    cout << "█\n";
    cout << COLOR_RESET << "\n\n" <<

```

```

}

// Función para validar una nota (0-10)
float pedirNota(const char* mensaje) {
    float nota;
    bool valido = false;

    do {
        cout << COLOR_HIGHLIGHT << mensaje << COLOR_INPUT;
        cin >> nota;
        cout << COLOR_RESET;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << COLOR_ERROR << "■■ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
        }
        else if(nota < 0 || nota > 10) {
            cout << COLOR_ERROR << "■■ Error: La nota debe estar entre 0 y
10.\n\n" << COLOR_RESET;
        }
        else {
            valido = true;
        }
    } while(!valido);

    return nota;
}

// Función para mostrar el nivel académico según el promedio
string obtenerNivelAcademico(float promedio) {
    if(promedio >= 9.0) return "Excelente ■";
    else if(promedio >= 8.0) return "Muy Bueno ■";
    else if(promedio >= 7.0) return "Bueno ■";
    else if(promedio >= 6.0) return "Aprobado ■";
    else return "Reprobado ■";
}

// Función para mostrar un gráfico de barras para el promedio
void mostrarGraficoPromedio(float promedio) {
    int longitud = (int)(promedio * 3); // 3 caracteres por unidad

    cout << COLOR_INFO << "[";
    for(int i = 0; i < 30; i++) {
        if(i < longitud) {
            if(promedio < 6) cout << COLOR_ERROR << "■";
            else if(promedio < 8) cout << COLOR_HIGHLIGHT << "■";
            else cout << COLOR_RESULT << "■";
        } else {
            cout << COLOR_INFO << "■";
        }
    }
    cout << COLOR_INFO << "] " << setprecision(2) << fixed << promedio <<
"/10.0" << COLOR_RESET << "\n\n";
}

// Función para mostrar una barra de progreso
void mostrarProgreso(int actual, int total) {
    cout << "\n■ Progreso: [";
    int porcentaje = (actual * 20) / total;
    for(int i = 0; i < 20; i++) {
        if(i < porcentaje) {
            cout << COLOR_SUCCESS << "■" << COLOR_RESET;
        } else {
            cout << "■";
        }
    }
    cout << "] " << (actual * 100) / total << "%\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // 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

// Explicación del programa
cout << "■ DESCRIPCIÓN: Este programa registra los datos académicos de
varios alumnos.\n";
cout << "           calcula sus promedios y muestra el alumno con
mejor desempeño.\n\n";

// Pedir cantidad de alumnos con validación
do {
    cout << COLOR_HIGHLIGHT << "■ Ingrese el número de alumnos a
registrar: " << COLOR_INPUT;
    cin >> n;
    cout << COLOR_RESET;

    if(cin.fail()) {
        cin.clear();
        cin.ignore(10000, '\n');
        cout << COLOR_ERROR << "■■■ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
    }
    else if(n <= 0) {
        cout << COLOR_ERROR << "■■■ Error: Debe registrar al menos un
alumno.\n\n" << COLOR_RESET;
    }
} while(n <= 0 || cin.fail());

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

// Pedir datos para cada alumno
for(int i = 0; i < n; i++) {
    // Limpiar pantalla y mostrar banner para cada alumno
    limpiarPantalla();
    mostrarBanner();

    cout << COLOR_HIGHLIGHT << "■ REGISTRO DE DATOS: ALUMNO " << (i+1)
<< " DE " << n << "\n";
    cout << "=====\\n" << COLOR_RESET;

    cin.ignore(i == 0 ? 1 : 10000, '\n'); // Limpiar buffer
adequadamente

    // Datos personales
    cout << COLOR_INFO << "■ DATOS PERSONALES\\n" << COLOR_RESET;

    cout << COLOR_HIGHLIGHT << " Nombre: " << COLOR_INPUT;
    cin.getline(alumnos[i].nombre, 50, '\n');
    cout << COLOR_RESET;

    cout << "\\n"; // Salto de línea entre inputs

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

    cout << "\\n"; // Salto de línea entre inputs

    // Validar la edad
    do {
        cout << COLOR_HIGHLIGHT << " Edad: " << COLOR_INPUT;
        cin >> alumnos[i].edad;
        cout << COLOR_RESET;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << COLOR_ERROR << " ■■■ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
        }
        else if(alumnos[i].edad <= 0 || alumnos[i].edad > 120) {
            cout << COLOR_ERROR << " ■■■ Error: La edad debe estar
entre 1 y 120 años.\n\n" << COLOR_RESET;
        }
    } while(alumnos[i].edad <= 0 || alumnos[i].edad > 120 ||
cin.fail());
}

```



```

    // Mostrar gráfico del promedio
    cout << COLOR_HIGHLIGHT << "■ REPRESENTACIÓN GRÁFICA DEL PROMEDIO:\n"
<< COLOR_RESET;
    mostrarGraficoPromedio(alumnos[pos_mejor].calificaciones.promedio_final);

    // Añadir mensaje de felicitación
    cout << COLOR_SUCCESS << "■ ¡Felicitaciones a " <<
alumnos[pos_mejor].nombre << " por obtener el mejor promedio!\n\n" <<
COLOR_RESET;

    // Resumen de todos los alumnos (tabla)
    cout << COLOR_HIGHLIGHT << "■ RESUMEN DE TODOS LOS ALUMNOS:\n";
    cout << "=====\\n\\n" << COLOR_RESET;

    cout << COLOR_INFO << "■■■■■\n";
    cout << "■ NUM ■ NOMBRE           ■ EDAD ■ PROMEDIO ■ NIVEL
■\\n";
    cout << "■■■■■" << COLOR_RESET << "\n";

    for(int i = 0; i < n; i++) {
        cout << "■ " << setw(5) << left << (i+1) << " ■ ";
        cout << setw(19) << left << alumnos[i].nombre << " ■ ";
        cout << setw(5) << left << alumnos[i].edad << " ■ ";
        cout << setw(8) << left << fixed << setprecision(2) <<
alumnos[i].calificaciones.promedio_final << " ■ ";

        // Destacar al mejor alumno
        if(i == pos_mejor) {
            cout << COLOR_SUCCESS << setw(8) << left << "■ MEJOR" <<
COLOR_RESET << " ■\\n";
        } else {
            cout << setw(10) << " " << "■\\n";
        }
    }

    cout << COLOR_INFO << "■■■■■" << COLOR_RESET << "\\n\\n";

    // Liberar memoria
    delete[] alumnos;

    // Añadir instrucciones finales
    cout << "Presione Enter para finalizar...";
    cin.ignore(10000, '\n');
    cin.get();

    return 0;
}

```

### **Resultado de la Ejecución:**

yoquelvisabreu — bloque8\_ejercicio6 · run bloque8\_ejercicio6.sh — 120×40

```
SISTEMA DE RANKING ACADÉMICO
● REGISTRO DE DATOS: ALUMNO 2 DE 2
=====
📝 DATOS PERSONALES
Nombre: jorge
Sexo (Masculino/Femenino): Mas
```

# Programa: bloque8\_ejercicio7

Compilador: g++  
Flags: -Wall -std=c++11  
Fecha: 2025-03-11 23:39:40  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## Código Fuente:

```
/*
 * ███ CRONÓMETRO DE CICLISMO ███
 * =====
 *
 * 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: Yoquelvis Abreu
 * Fecha: Marzo 2024
 */

#include <iostream>
#include <iomanip>
#include <limits>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_INFO = "\033[1;34m"; // Azul brillante
const string COLOR_ERROR = "\033[1;31m"; // Rojo brillante
const string COLOR_RESULT = "\033[1;35m"; // Magenta brillante
const string COLOR_SUCCESS = "\033[1;92m"; // Verde claro brillante

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

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout << "██████████ CRONÓMETRO DE CICLISMO ██████████\n";
    cout << "██████████\n";
    cout << COLOR_RESET;
}

// Función para pedir un valor de tiempo con validación
int pedirTiempo(const char* mensaje, int min, int max) {
    int valor;
    bool valido = false;

    do {
        cout << COLOR_HIGHLIGHT << mensaje << COLOR_INPUT;
        cin >> valor;
    } while (!valido);
}
```

```

        cout << COLOR_RESET;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << COLOR_ERROR << " █ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
        }
        else if(valor < min || valor > max) {
            cout << COLOR_ERROR << " █ Error: El valor debe estar entre "
<< min << " y " << max << ".\n\n" << COLOR_RESET;
        }
        else {
            valido = true;
        }
    } while(!valido);

    return valor;
}

// Función para formatear tiempo en formato HH:MM:SS
string formatearTiempo(const Tiempo& t) {
    char buffer[20];
    sprintf(buffer, "%02d:%02d:%02d", t.horas, t.minutos, t.segundos);
    return string(buffer);
}

// Función para mostrar una barra de progreso
void mostrarProgreso(int actual, int total) {
    cout << "\n█ Progreso: [";
    int porcentaje = (actual * 20) / total;
    for(int i = 0; i < 20; i++) {
        if(i < porcentaje) {
            cout << COLOR_SUCCESS << "█" << COLOR_RESET;
        } else {
            cout << "█";
        }
    }
    cout << "] " << (actual * 100) / total << "%\n\n";
}

// Función para mostrar una visualización gráfica del tiempo total
void mostrarGraficoTiempo(const Tiempo& tiempo) {
    // Convertir todo a segundos para la visualización
    int segundos_totales = tiempo.horas * 3600 + tiempo.minutos * 60 +
    tiempo.segundos;

    // Determinar escala: cada bloque representa 10 minutos (600 segundos)
    int bloques = segundos_totales / 600;
    if (segundos_totales % 600 > 0) bloques++; // Redondeando hacia arriba

    cout << COLOR_INFO << " Cada █ representa 10 minutos\n\n";
    cout << "[";
    for(int i = 0; i < bloques; i++) {
        if(i % 6 == 0 && i > 0) {
            cout << COLOR_RESULT << "█" << COLOR_INFO; // Marca cada hora
        }
        cout << COLOR_HIGHLIGHT << "█";
    }
    cout << COLOR_INFO << "] " << formatearTiempo(tiempo) << COLOR_RESET <<
"\n\n";
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // 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

    // Explicación del programa
    cout << " █ DESCRIPCIÓN: Este programa calcula el tiempo total empleado
    por un ciclista\n";
    cout << "
                                en varias etapas de una competición.\n\n";
}

```



## **Resultado de la Ejecución:**

yoquelvisabreu — bloque8\_ejercicio7 < run bloque8\_ejercicio7.sh — 120x40

CRONÓMETRO DE CICLISMO

REGISTRO DE TIEMPOS POR ETAPA

ETAPA	HORAS	MINUTOS	SEGUNDOS	TIEMPO
1	20	12	10	20:12:10
TIEMPO TOTAL				20:12:10

TIEMPO TOTAL EMPLEADO

DESGLOSE DE TIEMPO

Horas:	20
Minutos:	12
Segundos:	10
Tiempo total: 20:12:10	

REPRESENTACIÓN GRÁFICA DEL TIEMPO TOTAL:  
Cada █ representa 10 minutos

Presione Enter para finalizar...

# Programa: bloque8\_ejercicio8

Compilador: g++  
Flags: -Wall -std=c++11  
Fecha: 2025-03-11 23:40:18  
Estado: Ejecución exitosa  
Salida:  
La salida del programa se muestra en la captura de pantalla

## **Código Fuente:**

```

/*
 * ────────── CLASIFICADOR DE PERSONAS POR DISCAPACIDAD ──────────
 * =====
 * * 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: Yoquelvis Abreu
 * * Fecha: Marzo 2024
 */
#include <iostream>
#include <string.h>
#include <iomanip>
#include <limits>
using namespace std;

// Constantes para colores (ANSI escape codes)
const string COLOR_RESET = "\033[0m";
const string COLOR_TITLE = "\033[1;36m"; // Cyan brillante
const string COLOR_HIGHLIGHT = "\033[1;33m"; // Amarillo brillante
const string COLOR_INPUT = "\033[1;32m"; // Verde brillante
const string COLOR_INFO = "\033[1;34m"; // Azul brillante
const string COLOR_ERROR = "\033[1;31m"; // Rojo brillante
const string COLOR_SUCCESS = "\033[1;92m"; // Verde claro brillante
const string COLOR_WARNING = "\033[1;33m"; // Amarillo brillante
const string COLOR_WITH = "\033[1;35m"; // Magenta brillante
const string COLOR_WITHOUT = "\033[1;94m"; // Azul claro brillante

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

// Función para limpiar la pantalla
void limpiarPantalla() {
    #ifdef _WIN32
        system("cls");
    #else
        system("clear");
    #endif
}

// Función para mostrar el banner del programa
void mostrarBanner() {
    cout << COLOR_TITLE << "\n";
    cout <<
"██████████ ██████████ ██████████ ██████████ ██████████\n";
    cout << " █ CLASIFICADOR DE PERSONAS POR DISCAPACIDAD █\n";
    cout <<
"██████████ ██████████ ██████████ ██████████\n\n" <<
COLOR_RESET;
}

```

```

// Función para mostrar una barra de progreso
void mostrarProgreso(int actual, int total) {
    cout << "\n■ Progreso: [";
    int porcentaje = (actual * 20) / total;
    for(int i = 0; i < 20; i++) {
        if(i < porcentaje) {
            cout << COLOR_SUCCESS << "■" << COLOR_RESET;
        } else {
            cout << "■";
        }
    }
    cout << "] " << (actual * 100) / total << "%\n\n";
}

// Función para mostrar una lista de personas con formato
void mostrarListaPersonas(const Persona* personas, int cantidad, const
char* titulo, string color) {
    cout << COLOR_HIGHLIGHT << "\n■ " << titulo << " (" << cantidad <<
")\n";
    cout << "=====\\n" << COLOR_RESET;

    if(cantidad > 0) {
        cout << color <<
"■ # ■ NOMBRE\n";
        cout << "■\n";
        cout <<
"■\n";
        for(int i = 0; i < cantidad; i++) {
            cout << "■ " << setw(2) << left << (i+1) << " ■ ";
            cout << setw(35) << left << personas[i].nombre << " ■\n";
        }
        cout << "■\n";
        << COLOR_RESET << "\n";
    } else {
        cout << COLOR_WARNING << "■■■ No hay personas en esta categoría.\n";
        << COLOR_RESET;
    }
}

// Función para mostrar un gráfico de distribución
void mostrarGraficoDistribucion(int con_disc, int sin_disc, int total) {
    float porcentaje_con = ((float)con_disc / total) * 100;
    float porcentaje_sin = ((float)sin_disc / total) * 100;

    int barras_con = (int)(porcentaje_con / 2.5); // 40 barras total, cada
una 2.5%
    int barras_sin = (int)(porcentaje_sin / 2.5);

    cout << COLOR_HIGHLIGHT << "\n■ DISTRIBUCIÓN GRÁFICA:\n";
    cout << "=====\\n" << COLOR_RESET;

    cout << COLOR_WITH << "Con discapacidad      [ ";
    for(int i = 0; i < 40; i++) {
        if(i < barras_con) cout << "■";
        else cout << "■";
    }
    cout << "] " << fixed << setprecision(1) << porcentaje_con << "% (" <<
con_disc << " )\\n" << COLOR_RESET;

    cout << COLOR_WITHOUT << "Sin discapacidad      [ ";
    for(int i = 0; i < 40; i++) {
        if(i < barras_sin) cout << "■";
        else cout << "■";
    }
    cout << "] " << fixed << setprecision(1) << porcentaje_sin << "% (" <<
sin_disc << " )\\n" << COLOR_RESET;
}

int main() {
    // Limpiar pantalla y mostrar banner
    limpiarPantalla();
    mostrarBanner();

    // 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 opcion; // Para la selección de si tiene o no discapacidad

// Explicación del programa
cout << "■ DESCRIPCIÓN: Este programa clasifica a un conjunto de
personas en dos grupos\n";
cout << "                según si tienen o no algún tipo de
discapacidad.\n\n";

// Pedir cantidad de personas con validación
do {
    cout << COLOR_HIGHLIGHT << "■ Ingrese el número de personas a
registrar: " << COLOR_INPUT;
    cin >> n;
    cout << COLOR_RESET;

    if(cin.fail()) {
        cin.clear();
        cin.ignore(10000, '\n');
        cout << COLOR_ERROR << "■■ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
    }
    else if(n <= 0) {
        cout << COLOR_ERROR << "■■ Error: Debe registrar al menos una
persona.\n\n" << COLOR_RESET;
    }
} while(n <= 0 || cin.fail());

// 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(int i = 0; i < n; i++) {
    cout << "\n" << COLOR_INFO << "■ PERSONA " << i+1 << " DE " << n <<
":\n";
    cout << "-----\n" << COLOR_RESET;

    cin.ignore(i == 0 ? 1 : 10000, '\n'); // Limpiar buffer
    cout << COLOR_HIGHLIGHT << " Nombre: " << COLOR_INPUT;
    cin.getline(personas[i].nombre, 50, '\n');
    cout << COLOR_RESET;

    // Pedir si tiene discapacidad con validación
    do {
        cout << "\n" << COLOR_HIGHLIGHT << " ¿Tiene alguna
discapacidad?\n";
        cout << " " << COLOR_SUCCESS << "1) Sí\n";
        cout << " " << COLOR_WARNING << "0) No\n";
        cout << COLOR_HIGHLIGHT << " Opción: " << COLOR_INPUT;
        cin >> opcion;
        cout << COLOR_RESET;

        if(cin.fail()) {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << COLOR_ERROR << " ■■ Error: Debe ingresar un número
válido.\n\n" << COLOR_RESET;
        }
        else if(opcion != 0 && opcion != 1) {
            cout << COLOR_ERROR << " ■■ Error: Debe ingresar 0 (No) o
1 (Sí).\n\n" << COLOR_RESET;
        }
    } while((opcion != 0 && opcion != 1) || cin.fail());

    // Asignar el valor booleano según la opción
    personas[i].discapacidad = (opcion == 1);

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

// Separar las personas en los dos vectores

```

```

        for(int 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++;
            }
        }

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

        // Mostrar el resumen de la clasificación
        cout << COLOR_HIGHLIGHT << "■ RESUMEN DE LA CLASIFICACIÓN\n";
        cout << "=====\\n\\n" << COLOR_RESET;

        cout << COLOR_INFO << "Total de personas registradas: " <<
COLOR_HIGHLIGHT << n << COLOR_RESET << "\\n";
        cout << COLOR_WITH << "Personas con discapacidad: " << contador_con <<
COLOR_RESET << "\\n";
        cout << COLOR_WITHOUT << "Personas sin discapacidad: " << contador_sin
<< COLOR_RESET << "\\n";

        // Mostrar gráfico de distribución
        mostrarGraficoDistribucion(contador_con, contador_sin, n);

        // Mostrar las personas sin discapacidad
        mostrarListaPersonas(sin_discapacidad, contador_sin, "PERSONAS SIN
DISCAPACIDAD", COLOR_WITHOUT);

        // Mostrar las personas con discapacidad
        mostrarListaPersonas(con_discapacidad, contador_con, "PERSONAS CON
DISCAPACIDAD", COLOR_WITH);

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

        // Añadir instrucciones finales
        cout << "\\nPresione Enter para finalizar...";
        cin.ignore(10000, '\\n');
        cin.get();

        return 0;
    }
}

```

### **Resultado de la Ejecución:**

```
yoquelvisabreu — bloque8_ejercicio8 · run bloque8_ejercicio8.sh — 120x40

CLASIFICADOR DE PERSONAS POR DISCAPACIDAD

[?] RESUMEN DE LA CLASIFICACIÓN
=====
Total de personas registradas: 2
Personas con discapacidad: 0
Personas sin discapacidad: 2

[?] DISTRIBUCIÓN GRÁFICA:
=====
Con discapacidad [██████████] 0.0% (0)
Sin discapacidad [██████████] 100.0% (2)

[?] PERSONAS SIN DISCAPACIDAD (2)
=====
# | NOMBRE
---|---
1 | test
2 | test2

[?] PERSONAS CON DISCAPACIDAD (0)
=====
⚠No hay personas en esta categoría.

Presione Enter para finalizar...■
```