

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

// ===== DEFINICIONES DE FUNCIONES =====

void menuPrincipal();
void menuExpresiones();
void ejercicio1();
void ejercicio2();
void ejercicio3() // MODIFICADA
void ejercicio4() // MODIFICADA
void ejercicio5() // MODIFICADA
void ejercicio6();
void ecuacionSegundoGrado();
void imprimirResultado(bool resultado);

// Función auxiliar para imprimir el resultado booleano
void imprimirResultado(bool resultado) {
    if (resultado) {
        printf("Resultado: TRUE (Verdadero)\n");
    } else {
        printf("Resultado: FALSE (Falso)\n");
    }
}

// ===== FUNCIÓN MAIN =====

int main() {
    menuPrincipal();
    return 0;
}
```

```
}
```

```
// (Las funciones menuPrincipal y menuExpresiones se mantienen igual)
```

```
// ===== MENÚ PRINCIPAL =====
```

```
void menuPrincipal() {
```

```
    int op;
```

```
    do {
```

```
        system("cls");
```

```
        printf("=====\\n");
```

```
        printf("|| MENU PRINCIPAL ||\\n");
```

```
        printf("=====\\n");
```

```
        printf("1. Evaluacion de Expresiones\\n");
```

```
        printf("2. Ecuacion de Segundo Grado\\n");
```

```
        printf("3. Salir\\n");
```

```
        printf("-----\\n");
```

```
        printf("Ingrese una opcion: ");
```

```
        if (scanf("%d", &op) != 1) {
```

```
            while (getchar() != '\\n');
```

```
            op = 0;
```

```
        }
```

```
        switch (op) {
```

```
            case 1:
```

```
                menuExpresiones();
```

```
                break;
```

```
            case 2:
```

```
                ecuacionSegundoGrado();
```

```

        break;

case 3:
    printf("\nSaliendo del programa. ¡Hasta pronto!\n");
    break;

default:
    printf("\nOpcion invalida. Presione una tecla para continuar.\n");
    system("pause");
}

} while (op != 3);

}

```

// ===== SUBMENÚ DE EXPRESIONES =====

```

void menuExpresiones() {
    int op;

    do {
        system("cls");
        printf("=====|\n");
        printf("|| SUBMENU: EVALUACION DE EXPRESIONES ||\n");
        printf("=====|\n");
        printf("1. Ejercicio 1\n");
        printf("2. Ejercicio 2\n");
        printf("3. Ejercicio 3 (Aritmetico)\n");
        printf("4. Ejercicio 4\n");
        printf("5. Ejercicio 5\n");
        printf("6. Ejercicio 6\n");
        printf("7. Regresar al Menu Principal\n");
        printf("-----\n");
    }
}
```

```

printf("Ingrese una opcion: ");

if (scanf("%d", &op) != 1) {
    while (getchar() != '\n');
    op = 0;
}

switch (op) {
    case 1: ejercicio1(); break;
    case 2: ejercicio2(); break;
    case 3: ejercicio3(); break;
    case 4: ejercicio4(); break;
    case 5: ejercicio5(); break;
    case 6: ejercicio6(); break;
    case 7: break;
    default:
        printf("\nOpcion invalida. Presione una tecla para continuar.\n");
        system("pause");
}
} while (op != 7);
}

```

// ===== EJERCICIOS DE EXPRESIONES =====

```

void ejercicio1() {
    int i=4, j=2, k=8;
    bool expr;

    system("cls");
    printf("== Ejercicio 1: (3*j + 8/k) != i*k - j ==\n");
    printf("Valores iniciales: i=%d, j=%d, k=%d\n", i, j, k);
}

```

```

printf("\n--- Evaluacion ---\n");

// (3*2 + 8/8) != 4*8 - 2
// (6 + 1) != 32 - 2
// 7 != 30

printf("Paso 1: Sustitucion: (%d + %d) != %d - %d\n", 3*j, 8/k, i*k, j);
printf("Paso 2: Suma/Resta: (7) != 30\n");

expr = (3*j + 8/k) != i*k - j;
printf("\nResultado logico (7 != 30):\n");
imprimirResultado(expr); // TRUE

system("pause");
}

```

```

void ejercicio2() {
    int m=6, n=3, p=9;
    bool expr;

    system("cls");
    printf("== Ejercicio 2: m*(n+p)/2 >= p-n ==\n");
    printf("Valores iniciales: m=%d, n=%d, p=%d\n", m, n, p);

    printf("\n--- Evaluacion ---\n");
    // 6*(3+9)/2 >= 9-3
    // 6*12/2 >= 6
    // 72/2 >= 6
    // 36 >= 6

    printf("Paso 1: Sustitucion: 6*(3+9)/2 >= 9-3\n");
    printf("Paso 2: Operaciones: 72/2 >= 6\n");
    printf("Paso 3: Simplificacion: 36 >= 6\n");
}

```

```

expr = m*(n+p)/2 >= p-n;
printf("\nResultado logico (36 >= 6):\n");
imprimirResultado(expr); // TRUE

system("pause");
}

// =====
//           EJERCICIO 3 (MODIFICADO)
// Se cambia la condición booleana final para que sea FALSE.
// Resultado aritmético: (5 + 2*10) = 25.
// Condición original: (Resultado == 25) -> TRUE
// Condición MODIFICADA: (Resultado == 100) -> FALSE
// =====

void ejercicio3() {
    int a=5, b=2, c=10;
    int resultado_aritmetico;
    bool expr_booleana;

    system("cls");
    printf("== Ejercicio 3: (a + b*c) (Comprobacion logica) ==\n");
    printf("Valores iniciales: a=%d, b=%d, c=%d\n", a, b, c);

    printf("\n--- Evaluacion Aritmetica ---\n");
    printf("Paso 1: Sustitucion: (5 + 2*10)\n");
    printf("Paso 2: Multiplicacion: (5 + 20)\n");

    resultado_aritmetico = (a + b*c);
    printf("Resultado Aritmetico: %d\n", resultado_aritmetico); // 25
}

```

```

// CAMBIO AQUI: La expresión booleana se cambia a una condición falsa.

expr_booleana = (resultado_aritmetico == 100);

printf("\nComprobacion logica (Resultado == 100):\n");

imprimirResultado(expr_booleana); // FALSE

system("pause");

}

=====

//           EJERCICIO 4 (MODIFICADO)

// Se cambia el operador de igualdad '==' por el de desigualdad '!='.

// Operación:  $x/(y-1) + 3/y == x - y$ 

// Sustitución:  $14/(13-1) + 3/13 == 14 - 13$ 

// Evaluación:  $1 + 0 == 1 \rightarrow \text{TRUE}$ 

// Condición MODIFICADA:  $1 + 0 != 1 \rightarrow \text{FALSE}$ 

=====

void ejercicio4() {

    int x=14, y=13;

    bool expr;

    system("cls");

    // Se cambia el operador == por != en el encabezado

    printf("== Ejercicio 4:  $x/(y-1) + 3/y != x - y ==\n");

    printf("Valores iniciales: x=%d, y=%d\n", x, y);

    printf("\n--- Evaluacion ---\n");

    printf("Paso 1: Sustitucion:  $14/(13-1) + 3/13 != 14 - 13\n");

    printf("Paso 2: Division Entera:  $1 + 0 != 1\n");

    // CAMBIO AQUI: Se cambia el operador de '==' a '!='

    expr = x/(y-1) + 3/y != x - y;
}$$$ 
```

```

printf("\nResultado logico (1 != 1):\n");
imprimirResultado(expr); // FALSE

system("pause");
}

// =====
//           EJERCICIO 5 (MODIFICADO)

// Se cambia el operador de relación '<=' por '>'.

// Operación original: (u*v - 4) <= v + u/2

// Sustitución: (12*4 - 4) <= 4 + 12/2

// Evaluación: 44 <= 10 -> FALSE (ya era falso, se mantiene o se cambia a uno más obvio)

// Si el objetivo es que asome falso, y ya asomaba falso, solo se cambia el texto para reflejarlo,
// o si se quiere ver el efecto del cambio, se cambia a una expresión que antes era TRUE.

// **Si el original SÍ daba TRUE, se cambia a un operador que dé FALSE.**

// Revisando el código original: El original daba FALSE (44 <= 10).

// *Suponiendo que se quería un cambio para garantizar FALSE, vamos a cambiarlo a '==':*

// (u*v - 4) == v + u/2

// 44 == 10 -> FALSE

// *O simplemente cambiamos el valor de 4 en la operación a 40 para que el lado izquierdo sea
// 8, y lo comparamos con <=*

// (u*v - 40) <= v + u/2

// (12*4 - 40) <= 4 + 12/2

// (48 - 40) <= 10

// 8 <= 10 -> TRUE. No queremos TRUE.

// **Opción más simple: Cambiamos el operador a '==!**

// =====

void ejercicio5() {
    int u=12, v=4;
    bool expr;
}

```

```

system("cls");

// Se cambia el operador <= por == en el encabezado para garantizar FALSE
printf("== Ejercicio 5: (u*v - 4) == v + u/2 ==\n");
printf("Valores iniciales: u=%d, v=%d\n", u, v);

printf("\n--- Evaluacion ---\n");
// (12*4 - 4) == 4 + 12/2
// (48 - 4) == 4 + 6
// 44 == 10
printf("Paso 1: Sustitucion: (12*4 - 4) == 4 + 12/2\n");
printf("Paso 2: Operaciones: (48 - 4) == 4 + 6\n");
printf("Paso 3: Simplificacion: 44 == 10\n");

// CAMBIO AQUI: Se cambia el operador de '<=' a '=='
expr = (u*v - 4) == v + u/2;
printf("\nResultado logico (44 == 10):\n");
imprimirResultado(expr); // FALSE

system("pause");
}

void ejercicio6() {
    int q=7, r=5;
    bool expr;

    system("cls");
    printf("== Ejercicio 6: (q + r*2) <= q*r - 10 ==\n");
    printf("Valores iniciales: q=%d, r=%d\n", q, r);

    printf("\n--- Evaluacion ---\n");
}

```

```

// (7 + 5*2) <= 7*5 - 10
// (7 + 10) <= 35 - 10
// 17 <= 25

printf("Paso 1: Sustitucion: (7 + 5*2) <= 7*5 - 10\n");
printf("Paso 2: Operaciones: (7 + 10) <= 35 - 10\n");
printf("Paso 3: Simplificacion: 17 <= 25\n");

expr = (q + r*2) <= q*r - 10;
printf("\nResultado logico (17 <= 25):\n");
imprimirResultado(expr); // TRUE

system("pause");
}

// (La función ecuacionSegundoGrado se mantiene igual)

// ===== ECUACIÓN DE SEGUNDO GRADO (ax^2 + bx + c = 0)
=====

void ecuacionSegundoGrado() {
    float a, b, c, d, x1, x2;

    system("cls");
    printf("== ECUACION DE SEGUNDO GRADO (ax^2 + bx + c = 0) ==\n");
    printf("Ingrese el coeficiente a: ");
    if (scanf("%f", &a) != 1) { while (getchar() != '\n'); a = 0; }
    printf("Ingrese el coeficiente b: ");
    if (scanf("%f", &b) != 1) { while (getchar() != '\n'); b = 0; }
    printf("Ingrese el coeficiente c: ");
    if (scanf("%f", &c) != 1) { while (getchar() != '\n'); c = 0; }

    // Verificar si es una ecuacion de segundo grado (a != 0) para evitar division por cero.

```

```

if (a == 0) {

    printf("\nERROR: 'a' debe ser diferente de cero para ser una ecuacion de segundo
grado.\n");

} else {

    // Fórmula del Discriminante: d = b^2 - 4ac

    d = b*b - 4*a*c;

    printf("\nDiscriminante (d): %.2f\n", d);

    // La fórmula cuadrática es: $x = \frac{-b \pm \sqrt{d}}{2a}$

    // Evaluar el Discriminante para determinar el tipo de soluciones

    if (d < 0) {

        printf("\nCONCLUSION: El discriminante es negativo. La ecuacion no tiene soluciones
reales.\n");

    } else if (d == 0) {

        // Solución única real: $x = -b / 2a$

        x1 = -b / (2*a);

        printf("\nCONCLUSION: El discriminante es cero. Hay una unica solucion real (x1 =
x2):\n");

        printf("Solucion x1 = %.2f\n", x1);

    } else {

        // Dos soluciones reales distintas

        x1 = (-b + sqrt(d)) / (2*a);

        x2 = (-b - sqrt(d)) / (2*a);

        printf("\nCONCLUSION: El discriminante es positivo. Hay dos soluciones reales
distintas:\n");

        printf("Solucion 1 (x1): %.2f\n", x1);

        printf("Solucion 2 (x2): %.2f\n", x2);

    }

}

system("pause");

```

