# HECTOR JAVIER SALAZAR ALVAREZ

a)

Texto

Descripción generada automáticamente

resultado = a + b \* c;

En esta línea, la multiplicación b \* c se realiza primero debido a su mayor precedencia sobre la suma a +. El resultado es 10 + (5 \* 2), lo que da 20.

resultado = (a + b) \* c;

Aquí, el cálculo dentro de los paréntesis (a + b) se realiza primero, seguido por la multiplicación con c. Esto resulta en (10 + 5) \* 2, que es 30.

resultado = a / b - c;

La división a / b se efectúa antes de la resta - c debido a su mayor precedencia. Esto resulta en 10 / 5 - 2, que es 0.

resultado = a / (b - c);

En esta expresión, primero se realiza la resta dentro de los paréntesis (b - c), luego a se divide por el resultado. Esto resulta en 10 / (5 - 2), que es 3.

printf("a <= b: %s\n", (a <= b) ? "verdadero" : "falso");

Aquí se evalúa si a es menor o igual a b. Dado que 10 no es menor ni igual a 5, el resultado es "falso".

printf("a > b: %s\n", (a > b) ? "verdadero" : "falso");

Se evalúa si a es mayor que b. Dado que 10 es mayor que 5, el resultado es "verdadero".

logico1 = (a > b) && (b < c);

Se evalúan dos condiciones con el operador lógico && (AND). La primera condición (a > b) es verdadera, pero la segunda (b < c) es falsa. Como && requiere que ambas condiciones sean verdaderas, el resultado es "falso".

logico2 = (a > b) || (b < c);

Se evalúan dos condiciones con el operador lógico || (OR). La primera condición (a > b) es verdadera y la segunda (b < c) es falsa. Como || requiere que al menos una condición sea verdadera, el resultado es "verdadero".

resultado = a; resultado += b;

La variable resultado se inicializa con a y luego se le suma b usando el operador +=. Esto resulta en 10 + 5, que es 15.

resultado = a + b \* c - (a / b) + c;

Esta operación combina varios operadores. Primero se realiza la multiplicación b \* c, luego la división a / b, y después se suman y restan de acuerdo con la precedencia de los operadores. El resultado final es 10 + (5 \* 2) - (10 / 5) + 2, que es 17.

b)

Hacer una función que reciba un valor unsigned int v por parámetro y que lo multiplique por 32 usando corrimientos a la izquierda. Reportar: ¿qué pasa si el resultado ya no cabe en un unsigned int después de los corrimientos?:

Al momento de usar insigned int tienes como parámetros este usa todos los números que estén en base 2 para (2^32)-1 y al hacer desplazamientos cuando ya no cabe es que se reinicia y empieza desde 0 y otra vez empieza a hacer los recorrimientos. Por ejemplo si se usa 4294967296 que es (2^32). El recorrimiento lo empieza a hacer desde 0 y si se usa (2^32)+1 que es 4294967297 ahora lo hace con 1 y así sucesivamente.

d)

1. Descripción de lo que modifique en el codigo

Al intentar compilar y ejecutar el programa, me encontré con varios problemas. Los errores principales incluyeron problemas de compilación debido a constantes no declaradas y advertencias relacionadas con la conversión de cadenas en C++(esto al incluir la librería en mi compilador no lo podía compilar en c aunque fuera un c que solamente incluía una librería en c++ por lo que en eso hice algunas converciones). También había problemas con la inclusión y uso de archivos de encabezado (“.h”).

2. Pasos que Realicé para la Solución

2.1. Corrección en “screen.h”

- Definición de Constantes: Verifiqué y aseguré que “SCREEN\_NC”, “SCREEN\_NR”, “N\_COLORS”, y los índices de colores (“I\_NRM”, “I\_RED”, etc.) estuvieran correctamente definidos.

- Declaración de Funciones: Confirmé que las funciones “gotoxy”, “clearScreen”, y “setColor” estuvieran correctamente declaradas en el archivo de encabezado.

2.2. Corrección en “screen.c”

* Inclusión del Archivo de Encabezado: Aseguré que “screen.h” se incluyera al principio del archivo “screen.c” para que todas las definiciones y declaraciones fueran accesibles.
* Definición Condicional de Plataforma: Ajusté las directivas condicionales para manejar correctamente las diferencias entre Windows y Linux/Mac.
* Implementación de Funciones:
* “gotoxy(int x, int y)”: Implementé la función para mover el cursor en función de la plataforma, ajustando para Windows y Linux/Mac.
* “clearScreen(void)”: Implementé la función para limpiar la pantalla moviendo el cursor a cada posición y escribiendo un espacio.
* “setColor(int idxColor)”: Configuré la función para cambiar el color del texto según el sistema operativo.

2.3. Corrección en “main\_relojes.c”

* Cambio en la Definición de la Cadena de Caracteres: Ajusté la definición de “chars” a “const char chars[]” para evitar advertencias relacionadas con la conversión de una cadena literal.
* Uso de Constantes: Verifiqué que las constantes “N\_COLORS” y las usadas en “setColor” estuvieran correctamente definidas y referenciadas.

3. Compilación y Ejecución

* Compilación: Utilicé “g++” para compilar el programa.
* Ejecución:
* En Windows: Ejecuté el archivo “.exe” desde la línea de comandos.
* En Linux/Mac: Usé el comando adecuado para ejecutar el archivo compilado.

La ultima modificación que hice aparte que hice para modificar que compilara en Windows fue que puse los siguientes caracteres: “-\\\*#”

# evidencia de que los incisos de los códigos compilan:

a)

Texto

Descripción generada automáticamente

b)

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

c)

Interfaz de usuario gráfica

Descripción generada automáticamente

d)



e)

Fondo negro con letras blancas

Descripción generada automáticamente con confianza media

f)

Texto

Descripción generada automáticamente