Proyecto: #3 Reloj

Nombres y apellidos	Teoría	Participacion
Pablo Eduardo Ghezzi Barton	3	100%
Flavia Ailen Mañuico Quequejana	3	100%
Yamileth Rincón Tejeda	3	100%

Introducción:

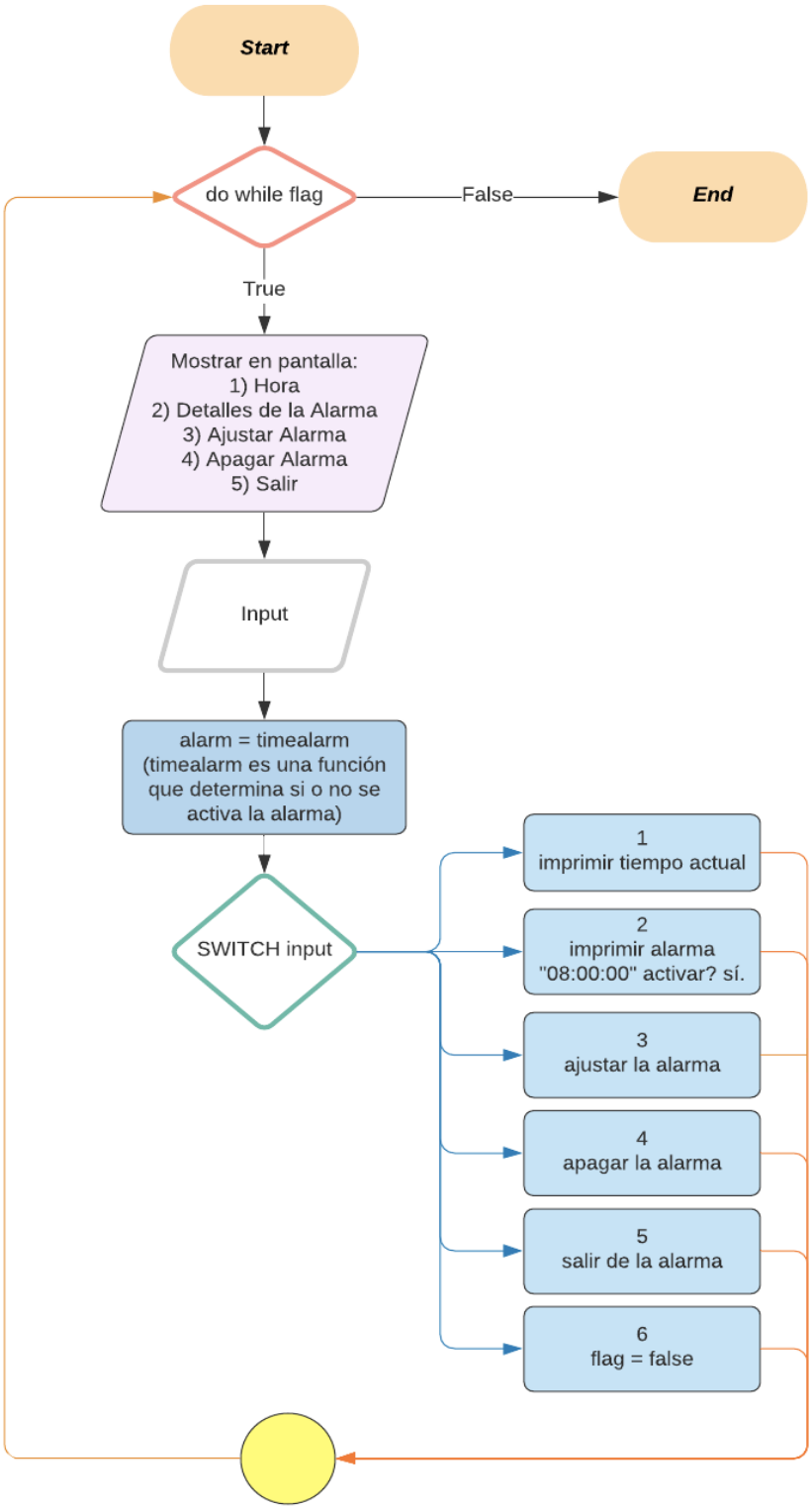
El problema se presenta con el trabajo de crear un reloj en C + +..
Las especificaciones de reloj son los siguientes:

- Debe indicar **Horas, Minutos y Segundos** basados en tiempos reales (se puede usar la librería ctime) y modificarlo.
- El reloj debe tener una variable **alarma** booleano que se inicia en **False.**
- Debe tener una función **prender la alarma**, que defina el booleano **alarma** a **True** a una hora determinada.
- Debe tener una función **apagar la alarma.**
- Debe tener una función **mostrar la alarma**, que imprima los datos de la alarma, como si está prendido o apagado.
- Debe correr el siguiente procedimiento:
    - Inicializar el reloj con la **hora actual** (*Esta se interpreta igual que la información de que la información de reloj debe ser actual, no que siempre debe estar activada debido a que impediría poner inputs a la consola*).
    - Imprimir la hora actual cuando se pida **y activar la alarma si la hora actual es mayor que las 8:00 am**. En caso contrario, **poner la alarma a las 8:00 am** (*Esto se interpreta como*).
    - Desactivar la alarma cuando se solicite e imprimir el mensaje correspondiente (*Esto se interpreta que cuando se activa o desactiva la alarma, se imprima un mensaje).*
    - (Extra) Dejar que el usuario cambie el horario de la alarma vía la interfaz principal.

Método:

(Nota: acá solo se enseña la estructura de control del main, con las funciones abstractas en el flujo. El código incluye comentarios en el funcionamiento del código mismo)

Como se debe interactuar, el sistema debe tener una interfaz definida por este flowchart, que define de forma abstracta el flujo de control del sistema.



Imágenes del código con comentarios:

```cpp
#include <iostream>

// La libreria en el centro del codigo
// ctime puedo conseguir el timpo actual
#include <ctime>

// iomanip manipula input y output para hacer que el tiempo
// impreso se vea bonito, haciedo que las horas
// tiene el estandard 00:00 formato
#include <iomanip>
```

-Librerías usadas en el código

```cpp
using namespace std;

// Retorna un puntero con objeto std::localtime para determinar
// el tiempo en este instante en forma de 24 horas
tm *tiempo()
{
    time_t result = time( _Time: nullptr);
    return localtime(&result);
}

// Usa el puntero now para determinar el tiempo en el instante
// Para imprimirlo de forma hh:mm:ss
void printTime()
{
    tm *ahora = tiempo();
    cout
            << setw( n: 2) << setfill( c: '0') << (ahora->tm_hour) << ':'
            << setw( n: 2) << setfill( c: '0') << (ahora->tm_min) << ':'
            << setw( n: 2) << setfill( c: '0') << (ahora->tm_sec) << endl;
    return;
}
```

-En la primera función se tiene el tiempo actual
-En la segunda función se imprime el tiempo actual

```
//Imprime la infomacion de la alarma
void printAlarm(bool &alarm, int *alarmtime)
{
    // impirme todo los numeros en alarmtime para
    // imprimir la hora que esta en el instante
    for (int i = 0; i < 3; i++)
    {
        cout << setw( n: 2) << setfill( c: '0') << alarmtime[i] << ':';
    }
    //formateo usando backspace para limpiar el ultimo ':'
    cout << '\b';
    cout << ' ' << '\t'
              << "Activar?: " << (alarm ? "Si" : "No")
              << endl;
    return;
}
```

-Se imprime la alarma “08:00:00 Activar? Sí”

```
// Usando referencias por objeto y un puntero al comienzo de un array
// La funcion actualiza variable alarm_time basado en el input del
// usuario
void setAlarm(int *alarm_time, bool &current)
{
    current = false;
    cout << "Establecre alarma(los tiempos no válidos se repetirán/ modelarán para que se ajusten) " << std::endl;
    cout << "Hour/Hora: ";
    cin >> alarm_time[0];
    cout << "Minute/Minuto: ";
    cin >> alarm_time[1];
    cout << "Second/Segundo: ";
    cin >> alarm_time[2];
    alarm_time[0] %= 24;
    alarm_time[1] %= 60;
    alarm_time[2] %= 60;
    return;
}
```

-Se establecerá una alarma

```
// Process la expresion booleana para ver si
// la alarma se activa o no
bool alarmOn(bool &alarm, int *alarm_time)
{
    tm *ahora = tiempo();
    // Esta expresion determina si alarma es mayor o igual a la alarma
    // set. Basado en eso determina el valor de alarm
    alarm = ((ahora->tm_hour > alarm_time[0]) ||
             ((ahora->tm_hour == alarm_time[0]) && ((ahora->tm_min > alarm_time[1]) ||
                                                    (ahora->tm_min == alarm_time[1]) && (ahora->tm_sec >= alarm_time[2]))));
    cout << "Alarma esta sonando!" << std::endl;
    return true;
}
```

-Se apagará la alarma

```
void alarmOff(bool &alarm, bool &current)
{
    //simplemente desactiva la alarma
    alarm = false;
    current = true;
}
```

-Se terminará el programa alarma

```cpp
int main(int argc, char *argv[])
{
    //Varibles booleanas:
    // - alarma determinado en si alarma esta predido o no
    // - flag para terminar el while loop
    // - current detemina si la alarma se debe quedar desactiva e.i no activarse
    bool alarm = false, flag = true, current = false;

    //Varibles int:
    // - array con {hh, mm, ss} para determinar que hora es en el momento
    // - input, la variable para el do while. Esta con su valor predeterminado 0
    int alarm_time[] = {8, 0, 0}, input;

    //Estructura de control pricipal basado en flow chart
    do
    {
        std::cout
            << "1) Time" << '\n'
            << "2) Alarm Details" << '\n'
            << "3) Set Alarm" << '\n'
            << "4) Turn off alarm" << '\n'
            << "5) Exit" << '\n'
            << ">> ";
        std::cin >> input;
        // solo chequea si alarm esta desactivado y si no se
        // desactivo esta alarma
        if (current)
            ;
        else if (!alarm)
            alarmOn(alarm, alarm_time);
        else if (input != 4)
            std::cout << "Alarma esta sonando!" << std::endl;
        switch (input)
        {
        case 1:
            printTime();
            break;
        case 2:
            printAlarm(alarm, alarm_time);
            break;
        case 3:
            setAlarm(alarm_time, current);
            break;
        case 4:
            if (!current)
            {
                alarmOff(alarm, current);
                std::cout << "Alarma Apagado" << std::endl;
            }
            break;
        case 5:
            flag = false;
            break;
        default:
            break;
        }
    } while (flag);
    // usando el flag deteremino si el while
    // continua o no
}
```

-Es la función principal (main), que se encargará de llamar a las funciones según la opción del usuario.

Resultados (descripción de partes relevantes del código, sus salidas y análisis de las mismas)

El código consta de 3 partes, el archivo main, que tiene el código principal que llamará a las funciones, el archivo clock.cpp que tiene las funciones desarrolladas y el archivo "clock.h" es el archivo que llama el archivo main por las funciones.

Usamos un switch para llamar a las funciones, según la opción del usuario, usamos la librería <ctime> para tener los datos de la hora actual y a la hora que se ajustará la alarma.

El código tiene una función llamada "setAlarm" (ajustar alarma), la cual se implementó a parte de lo que pedía el ejercicio, donde el usuario, puede escoger otra hora para la alarma.

Esta función devuelve la hora actual.

```
1) Time
2) Alarm Details
3) Set Alarm
4) Turn off alarm
5) Exit
>> 2
```

```
Alarma esta sonando!
```

```
Alarma Apagado
```

```
08:00:00        Activated?: No
```

```
Set Alarm (Invalid times will be repiared/ moded to fit)
Hour/Hora: 4
Minute/Minuto: 4
Second/Segundo: 3
```

Conclusiones:

El presente trabajo cumple algunas funciones las cuales posee un reloj convencional, solo que en este caso lo realizamos mediante un código.
El código utilizó funciones nombradas: *printTime* (muestra en pantalla la hora), *printAlarm* (muestra en pantalla la alarma), *setAlarm* (*extra: configura la alarma), *alarmOff* (apaga la alarma).

Por ejemplo, cuando empezamos a correr el código, automáticamente se nos muestra en pantalla 5 de las diferentes opciones que el usuario tiene por escoger: 1) Hora, 2) Detalles de la Alarma, 3) Establecer Alarma, 4) Apagar Alarma y 5) Salir. Si ingresamos el número 1, se imprime en pantalla la hora actual (hh:mm:ss) y el mensaje de "Alarma Sonando", si ingresamos el número 2, aparece el mensaje en pantalla "Alarma Sonando" y podemos ver la hora (por defecto) de la alarma la cual es 08:00:00, si luego ingresamos el número 3, en pantalla se muestra el mensaje "Alarma Sonando" y se pide las horas, minutos y segundos de la alarma que deseamos programar, y si luego ingresamos el número 4, la alarma se apaga por lo que ya no se imprime el mensaje "Alarma Sonando". Finalmente, si luego ingresamos el número 5, el programa termina.

- ***Funciones Extra:***

  Nuestro proyecto tiene como función extra dejar que el usuario cambie la hora de la alarma mediante la interfaz inicial, es decir, el usuario puede personalizar la alarma a la hora que desea. Esta función extra lleva el nombre de "*setAlarm*".