INFORME – Sistema de gestión de tareas semanales

En este informe se dará detalles de cómo funcionan las partes principales del proyecto, su lógica de funcionamiento y el como se relacionan una con otras.

librerías utilizadas:

1.- <iostream>: librería estándar de C++ utilizada para realizar operaciones de entrada y salida de datos.

2.- <string>: librería que permite trabajar con cadenas de tipo string, incluyendo funciones para manipular, comparar y convertir textos.

3.-<vector>: librería que proporciona la clase vector, que es una lista que crece o decrece automáticamente según sea necesario en el código.

4.- <unordered_map>: librería que permite utilizar un contenedor de tabla hash que almacena pares clave-valor. utilizado para almacenar element de forma rápida, accesible por una clave, siendo más eficiente que un map, ya que, si solo se busca por clave, no se necesita orden.

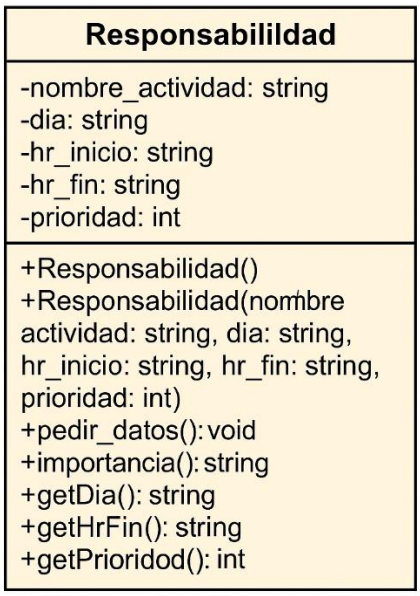
5.-<algorithm>: libreria que proporciona algoritmos estandar para trabajar con contenedores, incluyendo sort(), reverse(), find(), etc. entre otras. Permite ordenar, buscar o modificar datos de las selecciones.

6.-<fstream>: Permite leer y escribir en archivos desde el programa, usando clases como ifstream (leer) y ofstream (para escribir).

clase principal:

La clase principal que tiene el proyecto es la clase “Responsabilidad”. Esta clase es la que se encarga de crear las tareas que el usuario ingresa al programa. Esta clase tiene como atributos el nombre de la actividad, el día de esta, hora de inicio y fin de la actividad; y por último la prioridad de la tarea.

En cuanto a los métodos de la clase se tiene:



void pedir_datos() : es el método que se encarga de pedir los datos del usuario.
La primera parte del método tan solo pide el nombre de la actividad.
La siguiente parte del método valida el día, pidiendo al usuario que ingrese el día de la semana que se realizará la actividad y luego valida que sea un día correcto. El "for (auto& c: día) c= tolower(c);" convierte todo el texto a minúsculas para evitar problemas con mayúsculas y minúsculas. El if verifica que el día esté en la lista de de los días válidos, si estos días son válidos, sale del bucle while usando break.
la tercera parte válida la hora de inicio. Esta parte llama a la función hora_valida(hr_inicio) que es la encargada de validar si la hora está en el formato de 24:00 horas y si es válido.
de ser una hora válida, el bucle while termina, si no es válido, vuelve a pedir la hora después de mostrar el mensaje de error.
La cuarta parte válida la hora de finalización, que es muy similar a la 3ra parte ya descrita.
la quinta parte válida la prioridad ingresada, pidiendo la prioridad de la tarea descrita en 3 valores numéricos enteros: 1,2,3, donde el número más bajo es de mayor prioridad.
Con el if (cin.fail()) se verifica que el usuario no ingrese algo que no es un número, se limpia el error usando cin.clear() y se ignora los caracteres erróneos con cin.ignore(). luego se muestra el mensaje de error.
el siguiente Elseif (prioridad <1 || prioridad>3) verifica si el número está dentro del rango del 1 al 3.
si todas las entradas son válidas, el bucle while es roto con break.
Al final de todo se limpia la pantalla usando system("clear").

string importancia (): este método convierte el valor numérico de la prioridad en un texto que describe el nivel de importancia de la tarea, siendo 1(alto), 2 (media), 3 (baja).

void mostrar datos (): método que se encarga de mostrar los datos ingresados por el usuario.
Además, se tiene 2 constructores, siendo uno de estos, uno por defecto y otro con parámetros.
También aquí se encuentran los respectivos getters y setters.

funciones complementarias:

Validación de hora: bool hora_valida(const(string& hora)
Esta función es llamada dentro del método "pedir datos" de la clase, es básicamente una función auxiliar que ayuda al método a validar todos los datos respecto a las horas ingresadas. Se encarga de extraer la cadena de texto el cual introduce el usuario de la forma "00:00", también limitado al uso de hasta sólo las 23:59, el cual primero restringe la cadena que sea hasta de 5 espacios, de ahí extrayendo la cadena en dos partes hora y minutos usando el sub método hora.substraer, el cual solamente sustrae la cantidad de espacios que tiene la cadena que necesariamente precisa, de ahí mediante un bucle verifica si son números, para dar mayor seguridad y manejo de error, y de ahí pasa a unir las horas y minutos mediante la función stoi el cual transforma las cadenas de caracteres o strings en enteros, devolviendo así un valor numérico y devolviendo del método un valor booleano, porque se verifica si son valores y no cadena de texto.
Registro de actividad (.txt): void registrar_log(const string& mensaje)

Es una función el cual registra la creación y eliminación de tareas colocando su información dada a esa tarea y su prioridad, guardandola en un archivo de texto dentro de la carpeta donde se encuentra el programa.

Hora a entero: int hora_a_entero(const string& hora)

Esta función lo que hace es convertir la cadena de caracteres en un valor numérico para poder comparar con otro valor y así poder organizarlos **d**e menor a mayor, es decir, si introduzco una tarea el lunes a las 08:30 y luego otra tarea a las 09:40; esta función lo convierte a 0830 y 0940, teniendo los números 830 y 940 finalmente.

Mostrar actividad reciente (hash): mostrar_calendario_hash()

recorre el mapa unordered_map, ordena las tareas por prioridad y hora de finalización, luego las muestra día por día.

Usa la función sort() para ordenar las tareas de un día de acuerdo a la prioridad y hora, muestra un calendario semanal donde cada día tiene sus tareas ordenadas.

función principal:

En esta función principal se ejecuta la logia del programa.

Inicializa acción de variables:

- vector<Responsabilidad> lista_tareas:

Se crea un **vector de objetos** *Responsabilidad* llamado lista_tareas para almacenar todas las tareas que el usuario ingrese.

- unordered_map<string, vector<Responsabilidad>> tareas_por_dia;

Se crea un **mapa hash** (unordered_map) donde la **clave** es un **día de la semana** (string), y el **valor** es un **vector de tareas** (vector<Responsabilidad>), lo que permite almacenar tareas agrupadas por día.

- int opcion;

Se declara una variable opcion que almacenará la opción que el usuario elija en el menú del programa.

Bucles while:

- Se tiene un bucle do-while que se encarga de mostrar el menú al menos una vez y se asegura de terminar el programa solo si el valor 5 es ingresado.
- Luego de ese bule viene un 2do bucle while, que se encarga de verificar que los valores ingresados sean dentro del rango del 1 al 5. Si el usuario ingresa un carácter no numérico, el programa limpia el error de entrada con cin.clear() y cin.ignore(1000, '\n'), luego le pide al usuario que ingrese de nuevo. Si el número está fuera del rango (1-5), el programa vuelve a pedir que ingrese un valor correcto.

Switch case:

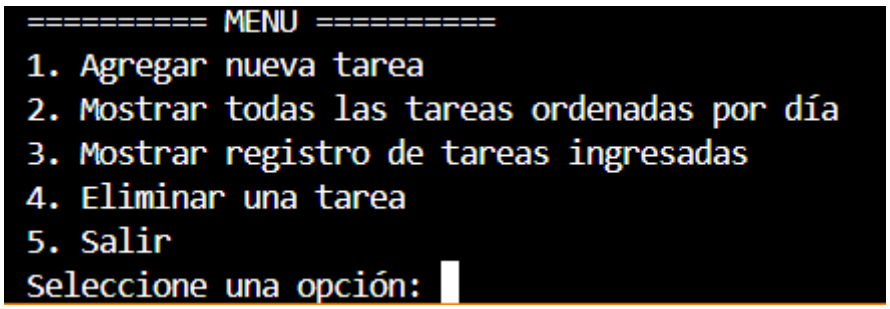
Dentro del switch case se tienen las acciones a realizarse:

- **case 1:** Crea un nuevo objeto Responsabilidad (una nueva tarea). Llama al método pedir_datos() de la clase Responsabilidad para que el usuario ingrese los datos de la

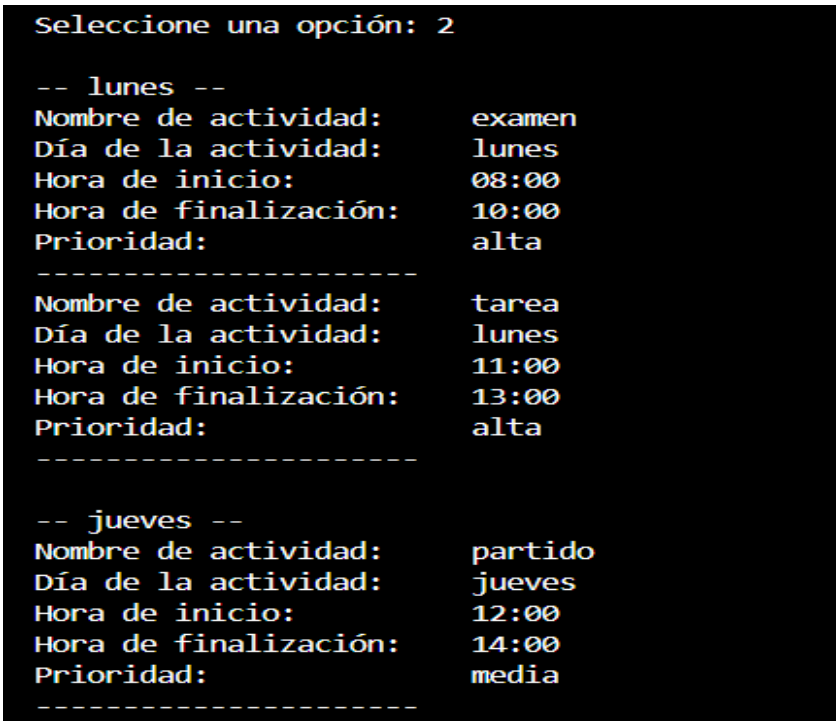
tarea. Para finalmente agregar la tarea a la lista lista_tareas y a la tabla hash tareas_por_dia (agrupada por el día de la semana). Registra la tarea en un archivo de log llamando a registrar_log().

- **case 2:** Si hay tareas registradas, las **muestras ordenadas** por día de la semana y hora de finalización. Si no hay tareas, muestra un mensaje de error.
- **case 3:** Llama a la función mostrar_calendario_hash() para mostrar todas las tareas agrupadas por día en un formato de tabla hash.
- **case 4:** Permite al usuario eliminar una tarea de la lista. Si la lista de tareas está vacía, muestra un mensaje indicando que no hay tareas para eliminar. Si hay tareas, le muestra una lista numerada de tareas y permite al usuario elegir cuál eliminar.
- **case 5:** Muestra un mensaje indicando que el programa está **saliendo**.
- **default:** Si el usuario ingresa un número no válido (que no corresponde a ninguna opción), muestra un mensaje de error.

La siguiente imagen ilustra el cómo el menú es generado en la terminal:



La siguiente imagen ilustra el formato en el que se muestran las tareas y actividades que se ingresen.



23:59
23 : 59