README - Sistema de Gestión de Tareas Semanales

Este proyecto es una **aplicación de consola en C++** diseñada para gestionar las tareas semanales de una persona. Su propósito es permitir al usuario agregar, eliminar y visualizar sus tareas de manera organizada, facilitando la planificación de su tiempo. A través de una interfaz en la terminal, el programa permite gestionar las tareas por día de la semana, con la posibilidad de asignarles una prioridad y horas de inicio y finalización.

Características del Proyecto:

1. **Gestión de Tareas**: El usuario puede ingresar detalles específicos de cada tarea, como el nombre de la actividad, el día de la semana en el que se llevará a cabo, la hora de inicio/finalización, y la prioridad de la tarea (alta, media o baja).
2. **Visualización Organizada**: Las tareas se muestran organizadas por día de la semana, lo que permite al usuario ver claramente su planificación semanal.
3. **Registro de Actividades**: Todas las acciones realizadas, como la adición o eliminación de tareas, se registran en un archivo de texto denominado `registro.txt`. Este archivo sirve como historial de las acciones realizadas durante la ejecución del programa.

   Se tiene también dentro del mismo programa una tabla hash que muestra las tareas ingresadas mientras el programa está corriendo.

4. **Validación de Entrada**: Para garantizar que el usuario ingrese datos válidos, se realiza una verificación de la entrada en varios puntos del programa:

- Se valida que las horas de inicio y finalización estén en el formato correcto de 24 horas (`HH:MM`).
- Se valida que el día de la semana ingresado sea uno de los días válidos (lunes, martes, etc.).
- Se valida que la prioridad sea un número entero entre 1 y 3, representando las prioridades alta, media y baja, respectivamente.

5. **Ordenamiento de Tareas**: Las tareas no solo se organizan por día, sino que también se ordenan por su **prioridad** y por **hora de finalización**. Las tareas de mayor prioridad se muestran primero, y entre las tareas con la misma prioridad, se ordenan por su hora de finalización.
6. **Interfaz de Consola**: El programa interactúa con el usuario a través de un menú simple en la terminal. Este menú permite al usuario realizar las operaciones deseadas, como agregar tareas, eliminar tareas o visualizar las tareas organizadas.
7. **Control de Entradas Incoherentes**: A través de la validación de entradas, el programa garantiza que el usuario no ingrese respuestas incoherentes o fuera de rango al momento de elegir una opción del menú, seleccionar un día o ingresar una actividad. Si el usuario comete un error, el programa lo notificará y le pedirá que ingrese la información correcta.

Uso:

A) Menú de Opciones

Al ejecutar el programa, empieza directamente en un menú interactivo el cual mostrará las siguientes opciones, este bucle o el menú estará ahí siempre y cuando no elijas una opción:

1. **Agregar nueva tarea**: Permite al usuario ingresar una nueva tarea con detalles como nombre, día, hora de inicio, hora de finalización y prioridad.
2. **Mostrar todas las tareas ordenadas por día**: Muestra las tareas de la semana organizadas por día.
3. **Mostrar tareas por día (tabla hash)**: Muestra las tareas utilizando una tabla hash que las organiza por día de la semana.
4. **Eliminar una tarea**: Permite al usuario eliminar una tarea específica de la lista.
5. **Salir**: Finaliza la ejecución del programa.

B) Ejemplo de Interacción:

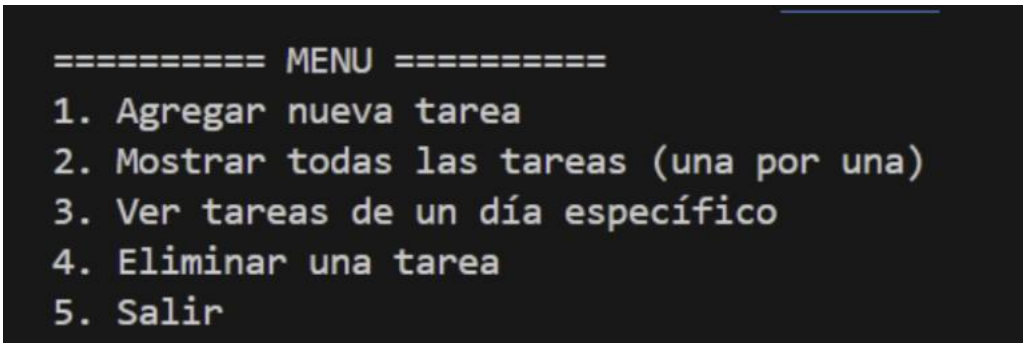

Ilustración 1 - menú interactivo

Lógica del Programa

1. Validación de Hora

La función Hora_valida es una función de tipo booleano el cual al ingresar el texto hora este reconoce cuando espacios tiene el cual debe tener un rango de 5 espacios de la matriz el cual si no supera los 5 espacios de memoria este devolverá un false, afectando al menú rehaciendo la pregunta de qué hora inicio o final le gustaría elegir, esta como una función fuera de una clase

- CPP

```
bool hora_valida(const string& hora) {
    if (hora.length() != 5 || hora[2] != ':') return false;
```

```
    // Verificación del formato y rango de hora
}
```

2. Registro de actividades:

La función `registrar_log` escribe un mensaje en un archivo de texto (`registro.txt`) para mantener un historial de las acciones realizadas, como la adición o eliminación de tareas, empieza a interactuar una vez se guarda una tarea en la semana, registrándolo directamente al archivo txt, gracias a la librería de fstream.

- CPP

```
void registrar_log(const string& mensaje) {
    ofstream archivo("registro.txt", ios::app); /* Guarda en la carpeta de texto en donde se acumula*/
    // Guardar mensaje en el archivo de registro
}
```

3. Clase Responsabilidad

La clase `Responsabilidad` representa una tarea y contiene los atributos:
- `nombre_actividad`// Nombre de la actividad.
- `dia`: //Día de la semana en que se realiza la actividad.
- `hr_inicio`: //Hora de inicio de la actividad.
- `hr_fin`: //Hora de finalización de la actividad.
- `prioridad`: //Prioridad de la actividad (alta, media, baja).

Métodos importantes:
- Constructor: //Inicializa los atributos de la actividad.
- pedir_datos**: //Solicita al usuario ingresar los detalles de la tarea, validando las entradas.
- mostrar.datos: //Muestra la información de la tarea en consola.
- importancia();:// Devuelve una cadena que representa la prioridad de la tarea.

* CPP
Dentro de la parte private se guarda los valores el cual objeto puede recibir y cambia con respecto al objeto necesario, también sus métodos es una forma de mostrar, pedirdatos, y hasta colocar la prioridad del objeto el cual se inicialice

```
class Responsabilidad {
private:
```

```
    string nombre_actividad;
    string dia;
    string hr_inicio;
    string hr_fin;
    int prioridad;

public:
    Responsabilidad();
    Responsabilidad(string nombre_actividad, string dia, string hr_inicio, string hr_fin, int
prioridad);
    void pedir_datos(); // Solicitar datos al usuario
    void mostrar() const; // Mostrar datos de la actividad
    string importancia() const; // Retornar prioridad
};
```

4. Mostrar Calendario (Tabla Hash)

Las tareas se almacenan en una tabla hash, utilizando un `unordered_map` donde la clave es el día de la semana y el valor es un vector de tareas. La función `mostrar_calendario_hash` organiza y muestra las tareas por día de la semana, ordenándolas por prioridad y hora de finalización, esta función de la tabla hash es almacenar las tareas las cuales han sido integradas, así como un monitor el cual guarda las tareas creadas, esto también es mediante vectores ya que cada tarea que se genere se guardara ahí.

* CPP

```
void mostrar_calendario_hash(const unordered_map<string, vector<Responsabilidad>>&
tareas_por_dia) {
    // Muestra las tareas organizadas por día utilizando una tabla hash
}
```

%Se utilizo chatgpt en el uso y la forma de colocar un hash para asi poder almacenar las tareas como en una lisa, el cual se preguntó cómo funciona y el dónde el calendario cumple con el objetivo de guarda la información.

5. Menú Principal y Opciones

El programa principal (`main`) ofrece un menú donde el usuario puede elegir entre varias opciones para interactuar con las tareas. Dependiendo de la opción seleccionada, el programa

ejecuta las funciones correspondientes, como agregar tareas, eliminar tareas o visualizar tareas ordenadas.

1. Menú de Opciones

El programa presenta un menú con cinco opciones:

1. **Agregar nueva tarea**: Permite al usuario ingresar los detalles de una nueva tarea, que incluye el nombre de la actividad, el día de la semana, las horas de inicio y finalización, y la prioridad (alta, media o baja).
2. **Mostrar todas las tareas ordenadas por día**: Muestra todas las tareas organizadas por día de la semana. Las tareas se ordenan primero por prioridad y luego por hora de finalización.
3. **Mostrar tareas por día (tabla hash)**: Muestra las tareas utilizando una estructura de tabla hash (`unordered_map`) que las agrupa por día de la semana.
4. **Eliminar una tarea**: Permite al usuario seleccionar una tarea previamente registrada y eliminarla tanto de la lista general de tareas como de la tabla hash.
5. **Salir**: Finaliza el programa.

2. Validación de Entradas

Para garantizar que las entradas sean correctas y evitar errores de ejecución:

- El programa valida que la opción seleccionada por el usuario esté dentro del rango de 1 a 5.
- Si el usuario ingresa un valor fuera de rango o un valor no numérico, el programa le solicita nuevamente una entrada válida.
- Se valida también que el día de la semana ingresado sea uno de los siete días válidos, y que las horas estén en formato de 24 horas.

3. Agregar Tarea
Cuando se selecciona la opción de agregar una tarea:

- El programa solicita al usuario que ingrese el nombre de la actividad, el día, las horas de inicio y finalización, y la prioridad.
- La tarea se guarda tanto en un **vector de tareas** (`lista_tareas`) como en un **unordered_map** (`tareas_por_dia`), donde las claves son los días de la semana y los valores son los vectores de tareas correspondientes a cada día.
- Se registra una entrada en un archivo de registro (`registro.txt`) para llevar un historial de las tareas agregadas.

4. Mostrar Tareas Ordenadas por Día

Cuando se selecciona la opción para mostrar todas las tareas:

- Las tareas se agrupan por día (lunes, martes, etc.) y se ordenan por prioridad y hora de finalización.
- Si no hay tareas registradas, el programa mostrará un mensaje indicándolo.

5. Mostrar Tareas por Día (Tabla Hash)
Al seleccionar esta opción:

- El programa usa la estructura de datos `unordered_map` para mostrar las tareas agrupadas por día de la semana.
- El formato de presentación es similar al anterior, pero esta vez se utiliza la tabla hash para organizar la información.

6. Eliminar Tarea
Cuando el usuario selecciona eliminar una tarea:

- El programa presenta una lista numerada de las tareas registradas, permitiendo al usuario elegir la tarea que desea eliminar.
- Si la tarea seleccionada existe, se elimina tanto de la lista de tareas (`lista_tareas`) como del `unordered_map` de tareas por día (`tareas_por_dia`).
- Se registra la eliminación de la tarea en el archivo de registro.

7. Salir
Cuando se selecciona esta opción:

- El programa termina su ejecución de manera segura y muestra un mensaje indicando que se está saliendo del programa.