Informe de la Arquitectura del Programa

Este programa implementa una estructura de datos para gestionar secciones académicas, incluyendo información sobre el nombre de la sección, los bloques de tiempo que la componen, el profesor asignado y los estudiantes inscritos. La arquitectura del programa se basa en los siguientes componentes principales:

1. Conjunto<T> (Conjunto.h):

- **Propósito:** Define una plantilla para representar un conjunto de elementos de cualquier tipo (T).
- **Estructura:**
  - Utiliza un std::vector<T> privado llamado elems para almacenar los elementos del conjunto.
  - Proporciona métodos públicos para:
    - insertar(const T &e): Inserta un elemento e en el conjunto solo si aún no pertenece a él.
    - pertenece(const T &e) const: Verifica si un elemento e pertenece al conjunto.
    - eliminar(const T &e): Elimina un elemento e del conjunto si existe.
- **Patrón de Diseño:** Implementa un concepto básico de conjunto, asegurando que no haya elementos duplicados.

2. Section (Section.h y Section.cpp):

- **Propósito:** Define una clase para representar una sección académica con sus atributos y funcionalidades.
- **Estructura (Section.h):**
  - Miembros públicos:
    - name: Un std::string para el nombre de la sección.
    - blocks: Un std::vector<int> que almacena la duración en horas de cada bloque de tiempo de la sección.
    - professor: Una estructura anidada Professor que contiene el name (string) y la ci (long, cédula de identidad) del profesor.
    - students: Un objeto de la clase Conjunto<long> para almacenar las cédulas de identidad de los estudiantes inscritos. El uso de Conjunto asegura que no haya estudiantes duplicados en la sección.
    - schedule: Un std::vector<BlockSchedule> para almacenar el horario detallado de cada bloque de la sección.
  - Métodos públicos:

        - Section(const std::string &name_, const std::vector<int> &blocks_): Constructor que inicializa el nombre y los bloques de la sección, y llama a initSchedule().
        - addStudent(long ci): Inserta la cédula de identidad (ci) de un estudiante en el conjunto de estudiantes de la sección.
        - setProfessor(const std::string &name, long ci): Asigna un profesor a la sección con su nombre y cédula de identidad.
        - initSchedule(): Inicializa el vector schedule basado en el vector blocks. Inicialmente, la hora de inicio de cada bloque se establece en 0.
    - Estructuras anidadas:
        - Professor: Contiene el nombre y la cédula de identidad del profesor.
        - BlockSchedule: Contiene la start_hour (int) y la duration (int) de un bloque de tiempo, y un método end_hour() que calcula la hora de finalización.
- **Implementación (Section.cpp):**
    - El constructor Section::Section inicializa los miembros y llama a initSchedule().
    - Section::addStudent utiliza el método insertar del objeto Conjunto para agregar estudiantes.
    - Section::setProfessor asigna los datos del profesor.
    - Section::initSchedule limpia el horario existente y lo redimensiona según la cantidad de bloques, inicializando la hora de inicio en 0 para cada bloque.

3. main.cpp:

- **Propósito:** Contiene la función principal (main) que orquesta la lectura de datos desde un archivo (datos.txt), la creación de objetos Section, la entrada del horario por parte del usuario y la presentación de la información de las secciones.
- **Funcionalidades:**
    - **Lectura de datos:** Abre y lee el archivo datos.txt línea por línea. Cada línea representa una sección y contiene información separada por punto y coma (;) en el siguiente formato: Nombre de la Sección; Duración de los Bloques (separados por coma); Nombre del Profesor, Cédula del Profesor; Cédulas de los Estudiantes (separados por coma).
    - **Procesamiento de líneas:** Para cada línea, se utiliza std::istringstream para extraer los diferentes campos. Se realiza un

proceso de "trimming" (eliminación de espacios en blanco al inicio y al final) para asegurar la correcta lectura de los datos.

- **Creación de objetos Section:** Se crea un objeto Section con el nombre y los bloques de tiempo analizados.
- **Asignación de profesor:** Se analiza la información del profesor y se utiliza el método setProfessor para asignarlo a la sección.
- **Adición de estudiantes:** Se analizan las cédulas de los estudiantes y se utiliza el método addStudent para agregarlos a la sección.
- **Entrada de horarios:** Itera sobre cada sección y luego sobre cada bloque de tiempo de esa sección, solicitando al usuario que ingrese la hora de inicio para cada bloque. Esta información se guarda en el vector schedule de la sección.
- **Presentación de la tabla:** Finalmente, el programa muestra una tabla formateada con la información de cada sección, incluyendo el nombre, el profesor y el horario de cada bloque.

Flujo General:

1. El programa comienza leyendo datos de secciones desde el archivo datos.txt.
2. Para cada sección leída, se crea un objeto Section y se inicializan sus atributos (nombre, bloques, profesor y estudiantes).
3. El programa solicita al usuario que ingrese la hora de inicio para cada bloque de cada sección.
4. Finalmente, se muestra una tabla resumen con la información de las secciones y sus horarios.

Compilación y Ejecución

1. Compilación:

Para compilar los archivos fuente, se debe utilizar un compilador de C++ que soporte el estándar C++11 (o superior), ya que se utilizan características como la inicialización uniforme y las plantillas. Abre una terminal o línea de comandos y navega hasta el directorio donde se encuentran los archivos Conjunto.h, Section.h, Section.cpp y main.cpp. Se usará el comando " g++ main.cpp Section.cpp -o horarios" para crear el ejecutable correcto a raiz de una funcion de "main.cpp".

2. Ejecución:

Una vez que la compilación sea exitosa y se haya generado el archivo ejecutable "horarios.exe", puedes ejecutar el programa desde la misma terminal o línea de comandos. De abrir el .exe directamente el programa se ejecutará y cerrará al instante porque no está diseñado para ello, obligatoriamente se tiene que abrir desde una terminal.

Acciones durante la ejecución:

1. **Lectura del archivo datos.txt:** Al iniciar la ejecución, el programa intentará abrir y leer el archivo llamado datos.txt que debe estar ubicado en el mismo directorio que el ejecutable. El formato esperado de este archivo es línea por línea, con la información de cada sección separada por punto y coma (;). Dentro de cada campo, las duraciones de los bloques y las cédulas de los estudiantes deben estar separadas por comas (,). Ejm:

   INFO-02002;2,2,2;Carlos
   Perez,11111111;31000000,31000001,31000002

2. **Entrada del horario por el usuario:** Después de leer las secciones del archivo, el programa iterará sobre cada sección y luego sobre cada bloque de tiempo definido para esa sección. Para cada bloque, solicitará al usuario que ingrese la hora de inicio (un número entero entre 0 y 23). El usuario deberá ingresar un número entero para cada solicitud y presionar Enter. El programa validará que la hora ingresada esté dentro del rango válido (0-23).
3. **Mostrar la tabla de horarios:** Una vez que el usuario haya ingresado los horarios para todos los bloques de todas las secciones, el programa mostrará una tabla formateada en la salida estándar (la consola). Esta tabla incluirá el nombre de la sección, el nombre del profesor y el horario de inicio y fin de cada bloque de tiempo. Finalmente, el programa terminará su ejecución con un código de retorno 0, indicando que se completó exitosamente. Si ocurre algún error durante la lectura del archivo (por ejemplo, si datos.txt no se encuentra), el programa mostrará un mensaje de error en la salida de error estándar (std::cerr) y terminará con un código de retorno 1.