## **Explanation of the Program**

**What it does:**FocusTime is a C++ console application that helps users track time spent on different tasks. Users can create tasks, start and stop timers for each task, and view how much time has been accumulated. The program saves all task data to a file so that time logs persist between sessions. This midterm is a MVP of the final project which will include more features/better usability.

**How it works:**

* When the program starts, it loads any saved tasks from a tasks.csv file.
* The main menu lets users:  
  + Add new tasks
  + Start and stop timers
  + View all tasks and their recorded time
  + Save all data to disk
* Timing is handled using the system clock to record the start and end times of each session.
* All tasks are stored in an array of pointers to dynamically allocated Task objects.
* On exiting, the program saves all tasks back to the CSV file to maintain a record for future sessions.

## **List of Concepts Used**

This project demonstrates all required C++ concepts:

**Data types and data sizes**

* string for task names
* int for durations
* long long for timestamps

**File types (.txt/.csv)**

* Tasks are saved to and loaded from tasks.csv

**Pointers**

* Tasks are dynamically allocated using new Task
* Stored in Task\* tasks[10] array
* Cleaned up with delete in the destructor to avoid memory leaks

**Arrays**

* Fixed-size array of pointers (Task\* tasks[10])

**Binary Search**

* Binary search implemented to locate tasks by name

**Use of strings**

* Task names handled as string

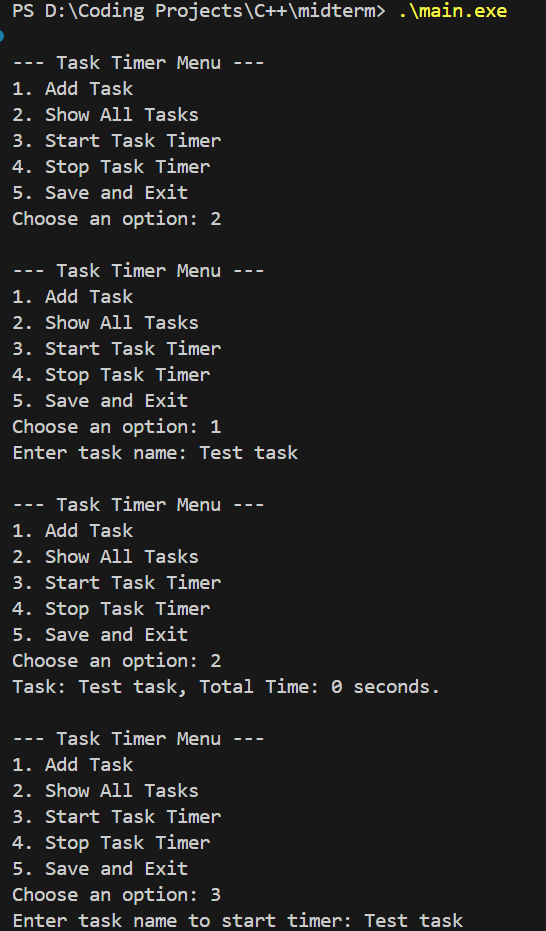
**File I/O**

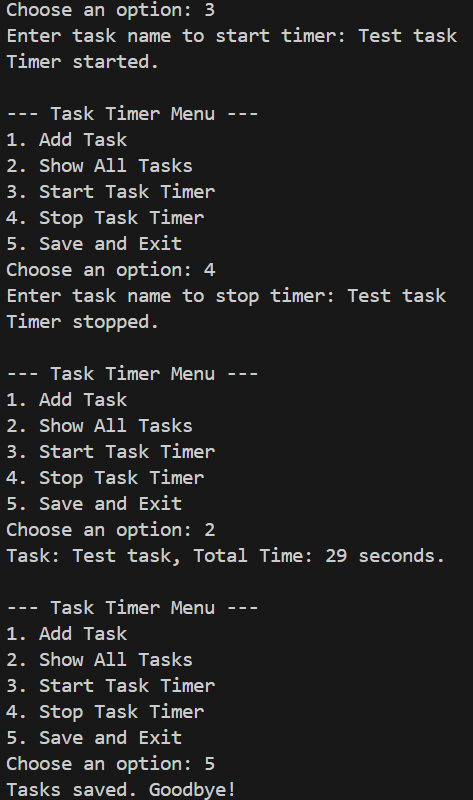
* ifstream to load saved tasks
* ofstream to save task data

**At least two classes**

* Task class handles individual task data and timing
* TaskManager class manages the collection of tasks

## **Screenshots of the Program Running**

****

****

Include:

* Program launch with menu
* Adding a task
* Starting and stopping the timer
* Showing all tasks with updated durations
* Exiting and verifying persistence

## **Challenges Faced and How They Were Solved**

**Managing dynamic memory**

* Challenge: Making sure that each Task object created with new was properly deleted to avoid memory leaks.
* Solution: Added a destructor in TaskManager that loops over all tasks and calls delete safely.

**Accessing private members**

* Challenge: While loading tasks from file, attempting to set startTime directly caused an error because startTime was private.
* Solution: Added a constructor overload in the Task class to initialize the total duration without needing to access private variables.

**Binary search requirements**

* Challenge: Implementing binary search in an array of pointers required careful handling of string comparisons and ensuring consistent sorting.
* Solution: Used standard string comparisons in the binarySearch() method, assuming tasks are added in sorted order or limiting use for small arrays.

**File I/O formatting**

* Challenge: Ensuring the data saved and loaded from the file was in a consistent format.
* Solution: Adopted a simple comma-separated format (CSV) to make reading and writing reliable.