XIAMEN UNIVERSITY MALAYSIA

Course Code	:	CST103
Course Name	:	Programming in language C
Lecturer	:	Prof. Li Xiaochao and Prof. Yang Chenhui
Academic Session	:	2023/09
Assessment Title	:	Lab Report and Presentation
Submission Due Date	:	2 / 1 / 2024

Prepared by :

Student ID	Student Name
CYS2309198	JIANG LETIAN

Date Received :

Feedback from Lecturer:

Mark:

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

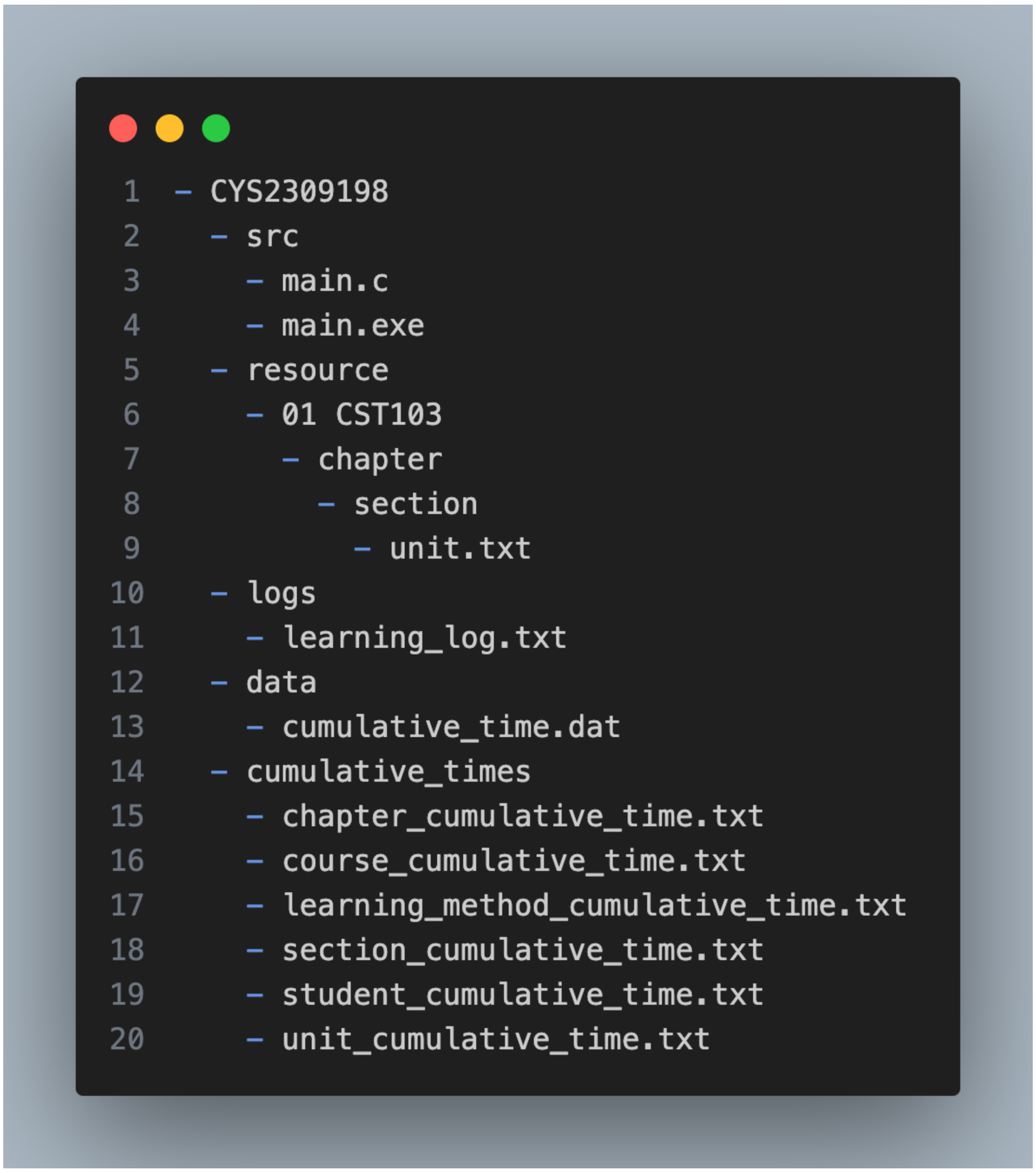
Signature: 江乐天

Date:2 / 1 / 2024

SUKPLTS system Lab report

Video link: CYS2309198

0: Folder structure



```
- CYS2309198
  - src
    - main.c
    - main.exe
  - resource
    - 01 CST103
      - chapter
        - section
          - unit.txt
  - logs
    - learning_log.txt
  - data
    - cumulative_time.dat
  - cumulative_times
    - chapter_cumulative_time.txt
    - course_cumulative_time.txt
    - learning_method_cumulative_time.txt
    - section_cumulative_time.txt
    - student_cumulative_time.txt
    - unit_cumulative_time.txt
```

1. Initial Goals:

1. Design a data representation: Design data structures to represent course content elements-course, chapter, section, and knowledge unit-along with their corresponding attributes such as learning times, CLOs, and student learning time.

2. Sustainability in the following study: Ensure the sustainability that allows it to add new course, chapter, section, unit to this software.
3. User-centric experience: Ensure that the software provides an intuitive and user-friendly interface for easy navigation and interaction.
4. Data Integrity and Persistence: Maintain data integrity and ensure the persistence of cumulative learning time records across different sessions.
5. Scalability and extensibility: Design the software to be adaptable and easily extensible for future enhancements or additional functionality.

2. Primary Design Ideas:

1. Modular Architecture: The KPLTS was structured around a modular design approach that emphasized the division of functionality into separate, reusable modules. Each module focused on specific tasks, promoting code maintainability and flexibility.
2. File-based data storage: The system used file-based storage to organize and manage learning materials. Courses, chapters, sections, and units were represented as directories and files within the resource folder, allowing easy access and manipulation of content.
   Learning materials were hierarchically organized:
   - Courses
     - ■ Chapters
       - ◆ Sections
         - ● Units

   This hierarchy facilitated systematic navigation and access to the learning content.
3. User-driven interactions: The user interface was designed to be interactive, allowing users (students or administrators) to navigate through the learning materials using a user-friendly command line interface. Users were prompted for input to make selections or perform actions within the system.
4. Time Tracking Mechanism: A time tracking mechanism was implemented to automatically record start and end times for learning sessions. This mechanism calculated the elapsed time to provide accurate learning time for each session.
5. Data Logging and Cumulative Analysis: The system logged detailed information about learning sessions, capturing essential data such as student IDs, learning methods, timestamps, and specific learning materials accessed. It then processed this data to generate cumulative reports, allowing for in-depth analysis of learning patterns and time spent on various educational components.

3. User View Functionalities and Usage:

Upon launching the Knowledge Point Learning Tracking System (KPLTS), users are presented with a main menu offering several functionalities:

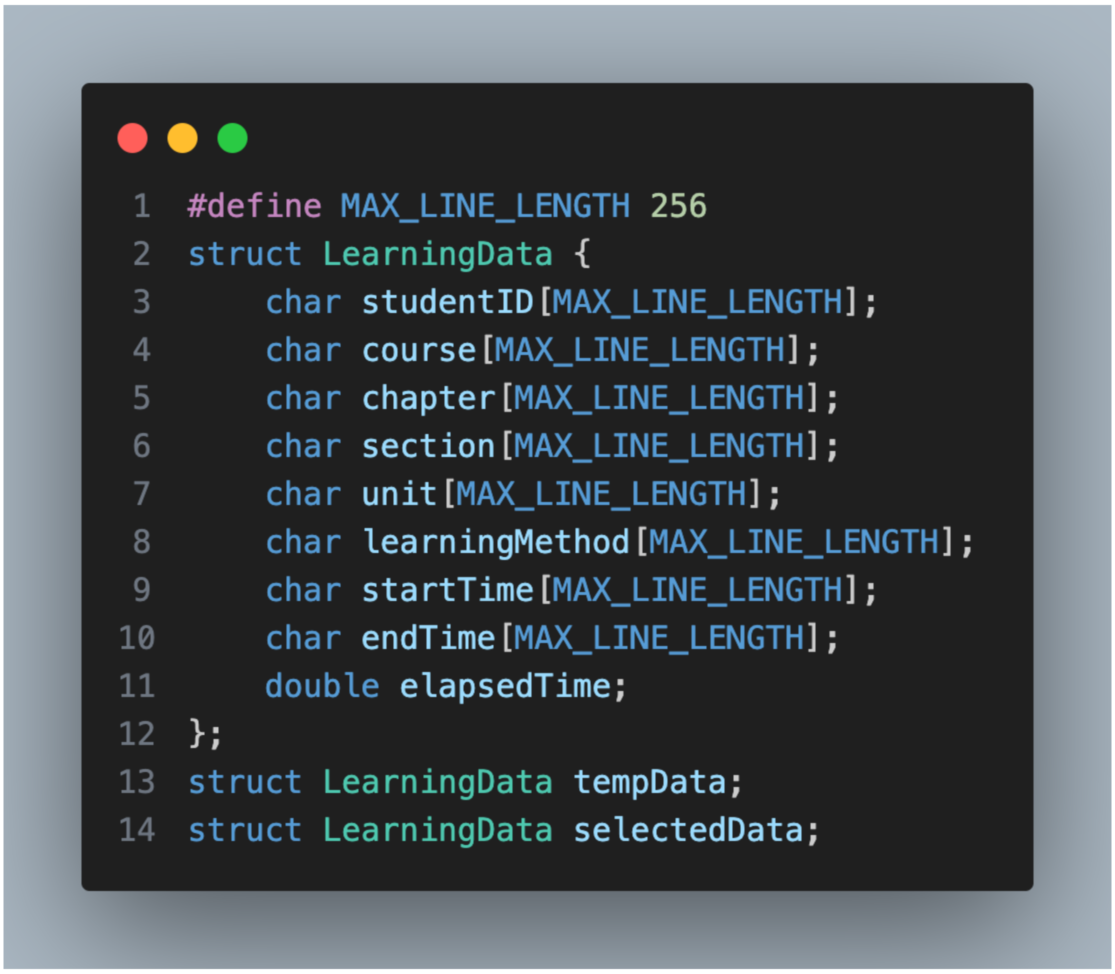
1. Start Learning: Allows users to initiate a learning session by selecting courses,

chapters, sections, and units to track learning time, then prompts the user to enter their student ID and learning method.

2. View All Units: Displays a structured view of available learning materials (courses, chapters, sections, and units) for exploration and selection.
3. Read Log: Provides access to a log file containing details of past learning sessions, including start time, end time, elapsed time, student ID, learning method, and specific learning materials accessed.
4. View Cumulative Learning Times: Provides options to view cumulative study times for various categories such as Students, Study Methods, Courses, Chapters, Sections, and Units.
5. Exit: Allows users to exit the system.

4. Programmer View: Software Structures and Modules

Data Structures:



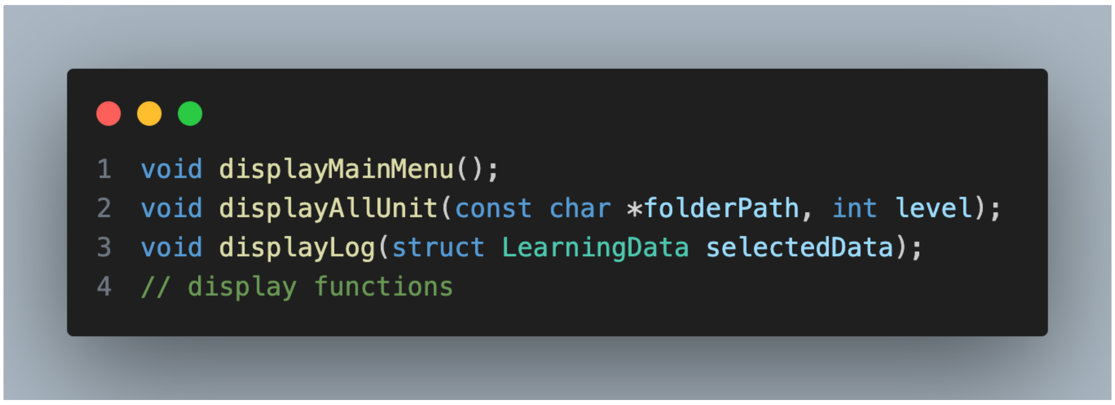
```c
#define MAX_LINE_LENGTH 256
struct LearningData {
    char studentID[MAX_LINE_LENGTH];
    char course[MAX_LINE_LENGTH];
    char chapter[MAX_LINE_LENGTH];
    char section[MAX_LINE_LENGTH];
    char unit[MAX_LINE_LENGTH];
    char learningMethod[MAX_LINE_LENGTH];
    char startTime[MAX_LINE_LENGTH];
    char endTime[MAX_LINE_LENGTH];
    double elapsedTime;
};
struct LearningData tempData;
struct LearningData selectedData;
```

Modular Architecture:
The Knowledge Point Learning Tracking System (KPLTS) was designed with a modular structure, featuring distinct modules responsible for specific functionalities:

1. Main Functionality Module: Contains the main function orchestrating user

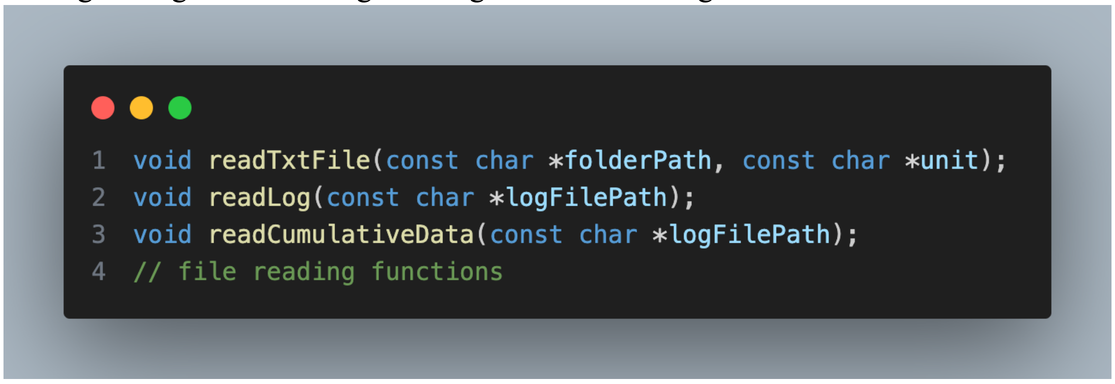
interactions and calling appropriate modules based on user inputs.

2. Display Module: Handles the display of menus, learning content, and summaries for user interaction.

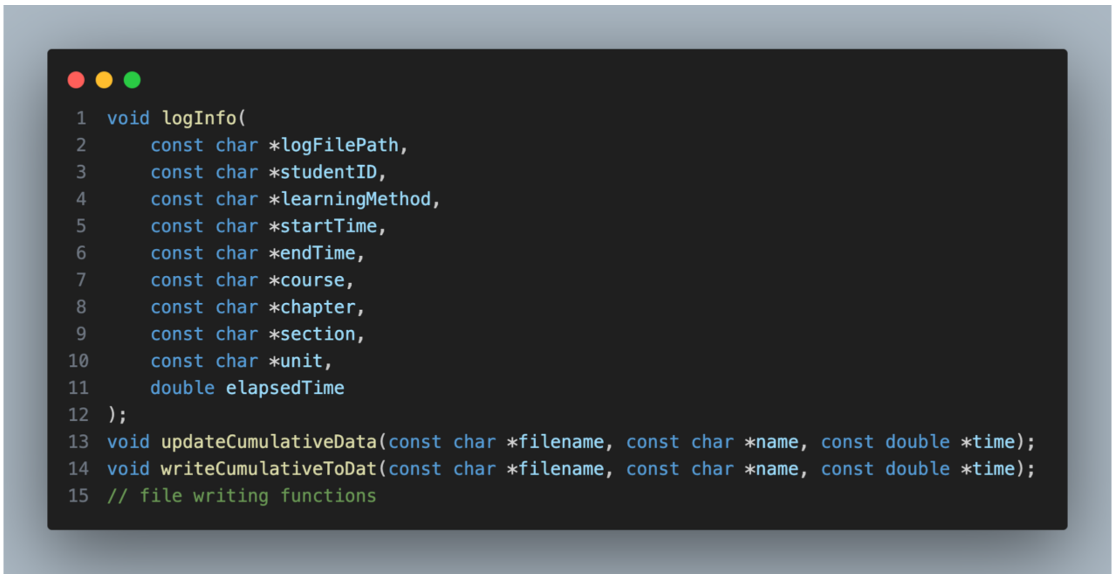


```c
void displayMainMenu();
void displayAllUnit(const char *folderPath, int level);
void displayLog(struct LearningData selectedData);
// display functions
```

3. File Operations Module: Manages file reading, writing, and directory handling for organizing and accessing learning materials and log files.

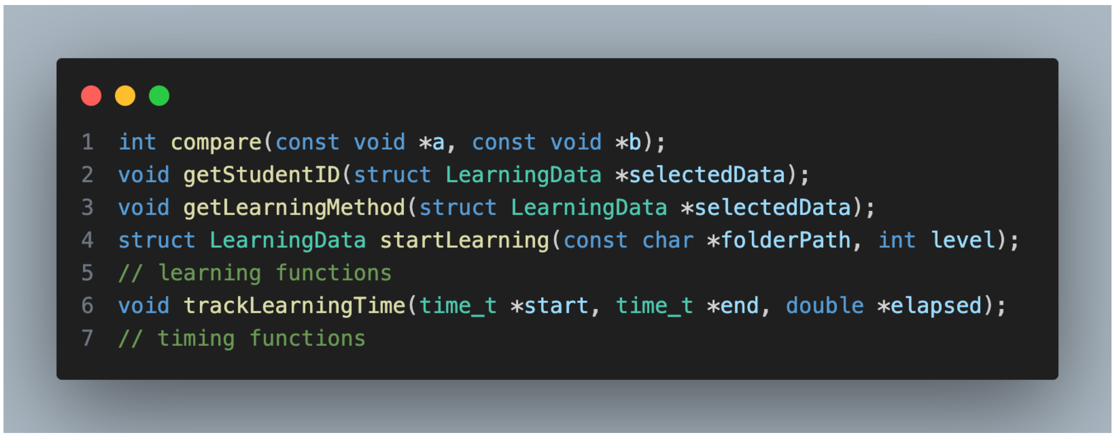


```c
void readTxtFile(const char *folderPath, const char *unit);
void readLog(const char *logFilePath);
void readCumulativeData(const char *logFilePath);
// file reading functions
```



```c
void logInfo(
    const char *logFilePath,
    const char *studentID,
    const char *learningMethod,
    const char *startTime,
    const char *endTime,
    const char *course,
    const char *chapter,
    const char *section,
    const char *unit,
    double elapsedTime
);
void updateCumulativeData(const char *filename, const char *name, const double *time);
void writeCumulativeToDat(const char *filename, const char *name, const double *time);
// file writing functions
```

4. Learning Tracking Module: Includes functionalities to track learning sessions, calculate elapsed time, and log session details.



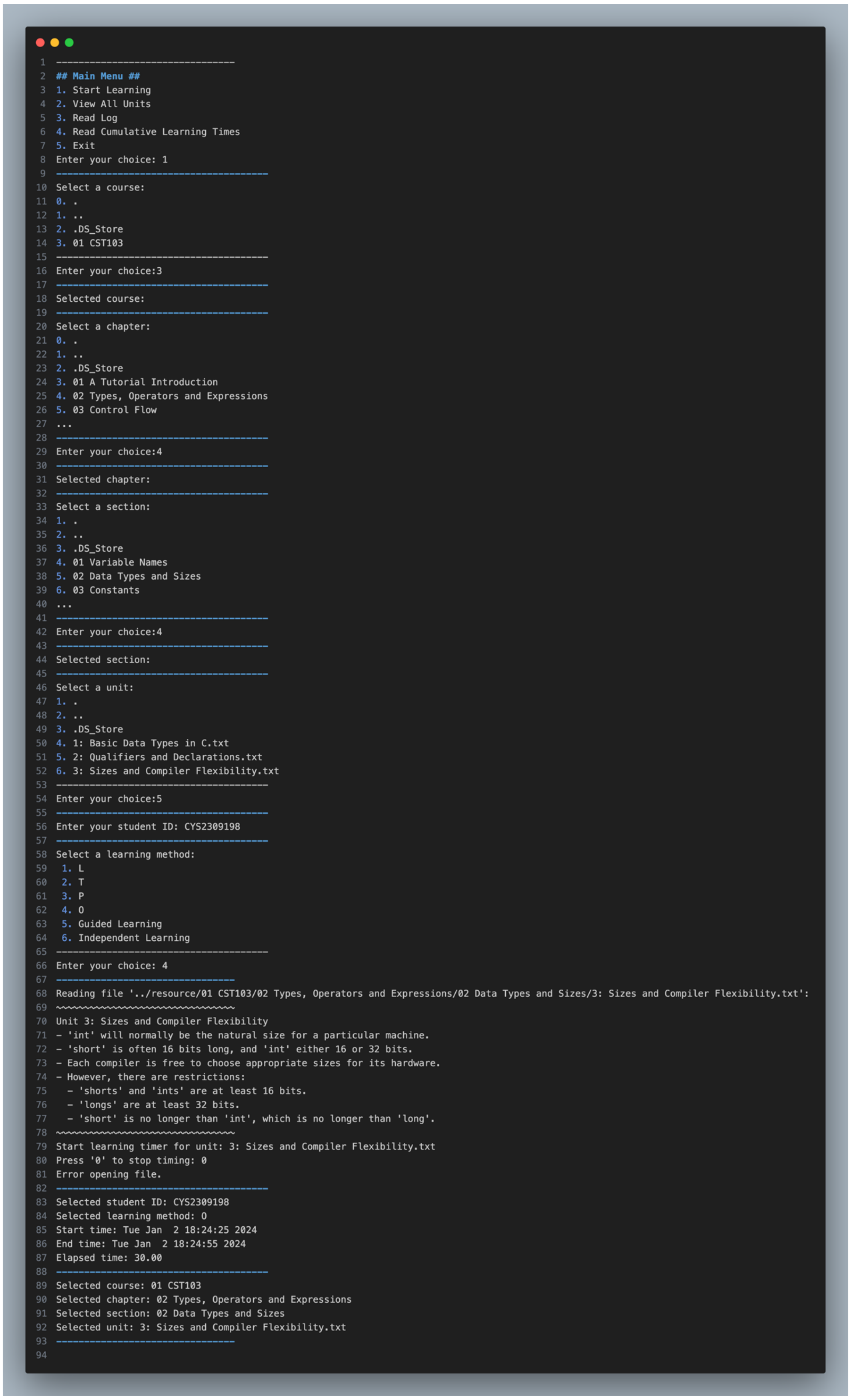
```c
int compare(const void *a, const void *b);
void getStudentID(struct LearningData *selectedData);
void getLearningMethod(struct LearningData *selectedData);
struct LearningData startLearning(const char *folderPath, int level);
// learning functions
void trackLearningTime(time_t *start, time_t *end, double *elapsed);
// timing functions
```

5. Valuable Findings

1. Benefit of file-based storage: scan the nested folder to get the unit, rather than using structure to contain all the names of courses, chapters, sections, units, making it possible to add new item to this system.
2. Cumulative data processing: Managing cumulative data for different categories required careful processing and updating, in this program I use 2 structure to store the data, tempData and selectedData, one for temporary data storage and one for complete data storage.
3. Benefit of modularity: Emphasizing modular design greatly improved the readability and maintainability of the code. The focused responsibility of each module facilitated easier debugging and updates without disturbing the entire system.
4. Data Integrity and File Operations: Ensuring data integrity and proper file operations was critical. proper handling of file access, reading, and writing prevented data corruption and maintained consistency.

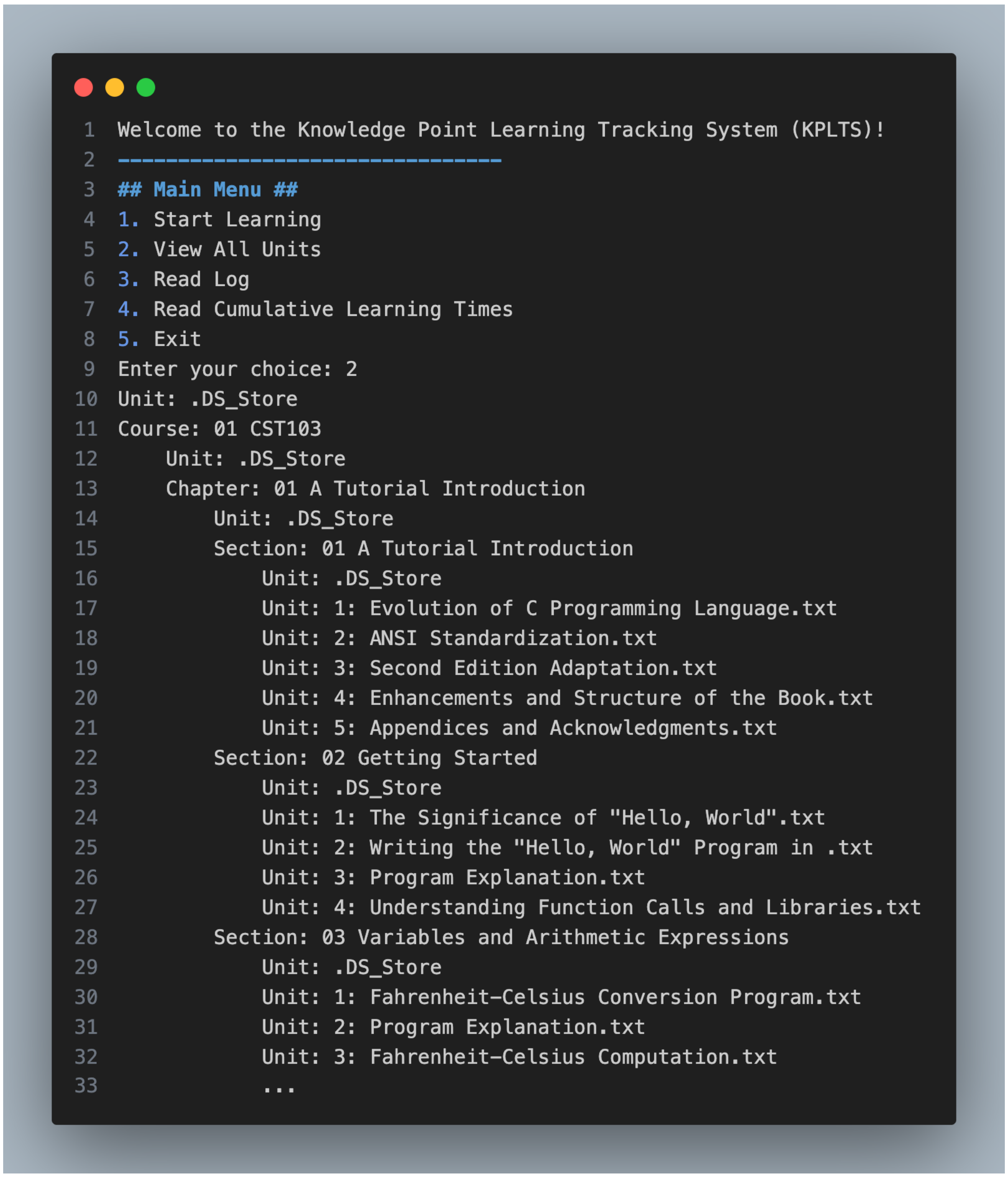
6. Test Cases:

1. Start learning



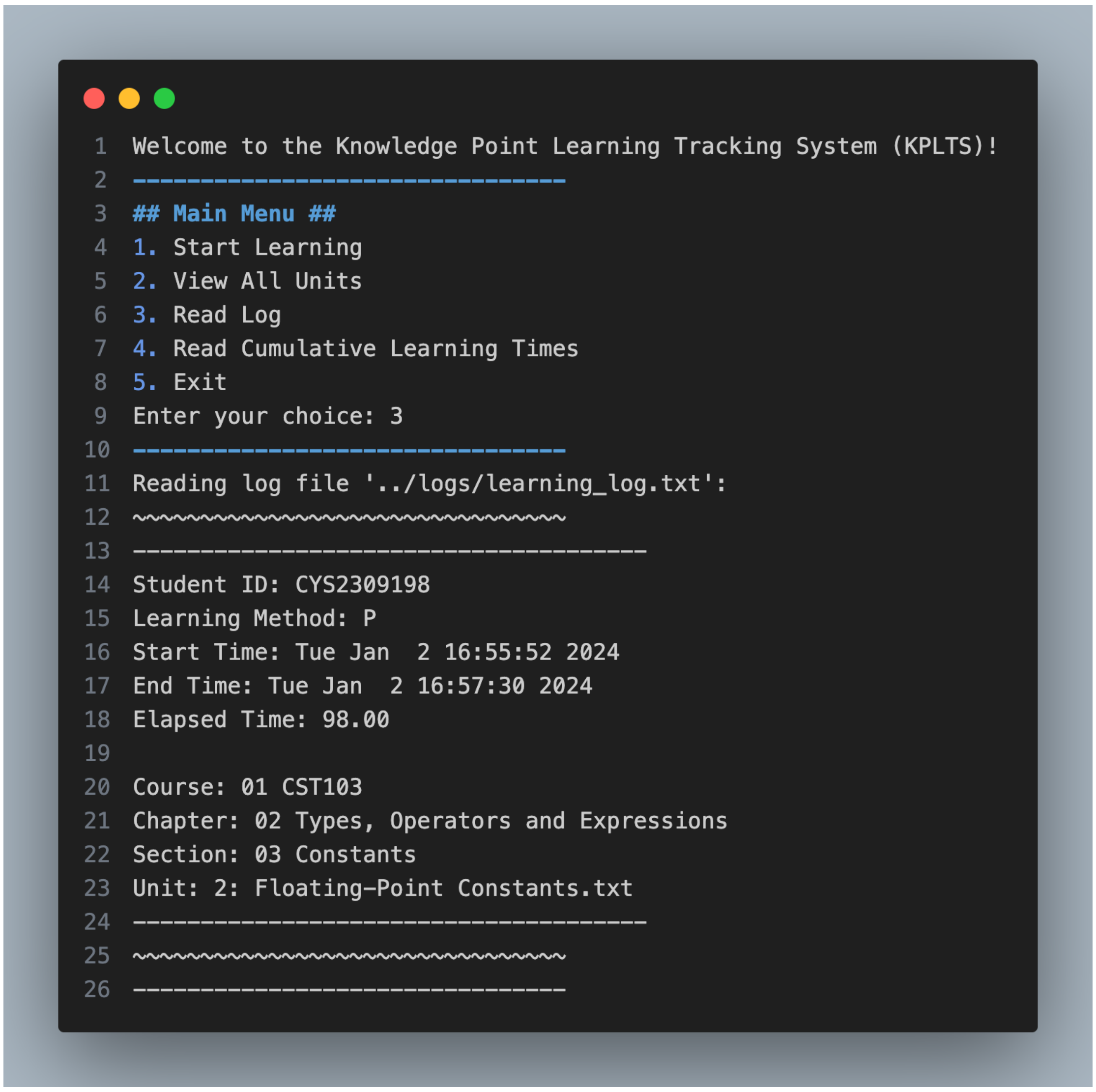
```
--------------------------------
## Main Menu ##
1. Start Learning
2. View All Units
3. Read Log
4. Read Cumulative Learning Times
5. Exit
Enter your choice: 1
--------------------------------------
Select a course:
0. .
1. ..
2. .DS_Store
3. 01 CST103
--------------------------------------
Enter your choice:3
--------------------------------------
Selected course:
--------------------------------------
Select a chapter:
0. .
1. ..
2. .DS_Store
3. 01 A Tutorial Introduction
4. 02 Types, Operators and Expressions
5. 03 Control Flow
...
--------------------------------------
Enter your choice:4
--------------------------------------
Selected chapter:
--------------------------------------
Select a section:
1. .
2. ..
3. .DS_Store
4. 01 Variable Names
5. 02 Data Types and Sizes
6. 03 Constants
...
--------------------------------------
Enter your choice:4
--------------------------------------
Selected section:
--------------------------------------
Select a unit:
1. .
2. ..
3. .DS_Store
4. 1: Basic Data Types in C.txt
5. 2: Qualifiers and Declarations.txt
6. 3: Sizes and Compiler Flexibility.txt
--------------------------------------
Enter your choice:5
--------------------------------------
Enter your student ID: CYS2309198
--------------------------------------
Select a learning method:
 1. L
 2. T
 3. P
 4. O
 5. Guided Learning
 6. Independent Learning
--------------------------------------
Enter your choice: 4
--------------------------------
Reading file '../resource/01 CST103/02 Types, Operators and Expressions/02 Data Types and Sizes/3: Sizes and Compiler Flexibility.txt':
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unit 3: Sizes and Compiler Flexibility
- 'int' will normally be the natural size for a particular machine.
- 'short' is often 16 bits long, and 'int' either 16 or 32 bits.
- Each compiler is free to choose appropriate sizes for its hardware.
- However, there are restrictions:
  - 'shorts' and 'ints' are at least 16 bits.
  - 'longs' are at least 32 bits.
  - 'short' is no longer than 'int', which is no longer than 'long'.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Start learning timer for unit: 3: Sizes and Compiler Flexibility.txt
Press '0' to stop timing: 0
Error opening file.
--------------------------------------
Selected student ID: CYS2309198
Selected learning method: O
Start time: Tue Jan  2 18:24:25 2024
End time: Tue Jan  2 18:24:55 2024
Elapsed time: 30.00
--------------------------------------
Selected course: 01 CST103
Selected chapter: 02 Types, Operators and Expressions
Selected section: 02 Data Types and Sizes
Selected unit: 3: Sizes and Compiler Flexibility.txt
--------------------------------
```

2. View all units



```
Welcome to the Knowledge Point Learning Tracking System (KPLTS)!
--------------------------------
## Main Menu ##
1. Start Learning
2. View All Units
3. Read Log
4. Read Cumulative Learning Times
5. Exit
Enter your choice: 2
Unit: .DS_Store
Course: 01 CST103
    Unit: .DS_Store
    Chapter: 01 A Tutorial Introduction
        Unit: .DS_Store
        Section: 01 A Tutorial Introduction
            Unit: .DS_Store
            Unit: 1: Evolution of C Programming Language.txt
            Unit: 2: ANSI Standardization.txt
            Unit: 3: Second Edition Adaptation.txt
            Unit: 4: Enhancements and Structure of the Book.txt
            Unit: 5: Appendices and Acknowledgments.txt
        Section: 02 Getting Started
            Unit: .DS_Store
            Unit: 1: The Significance of "Hello, World".txt
            Unit: 2: Writing the "Hello, World" Program in .txt
            Unit: 3: Program Explanation.txt
            Unit: 4: Understanding Function Calls and Libraries.txt
        Section: 03 Variables and Arithmetic Expressions
            Unit: .DS_Store
            Unit: 1: Fahrenheit-Celsius Conversion Program.txt
            Unit: 2: Program Explanation.txt
            Unit: 3: Fahrenheit-Celsius Computation.txt
            ...
```

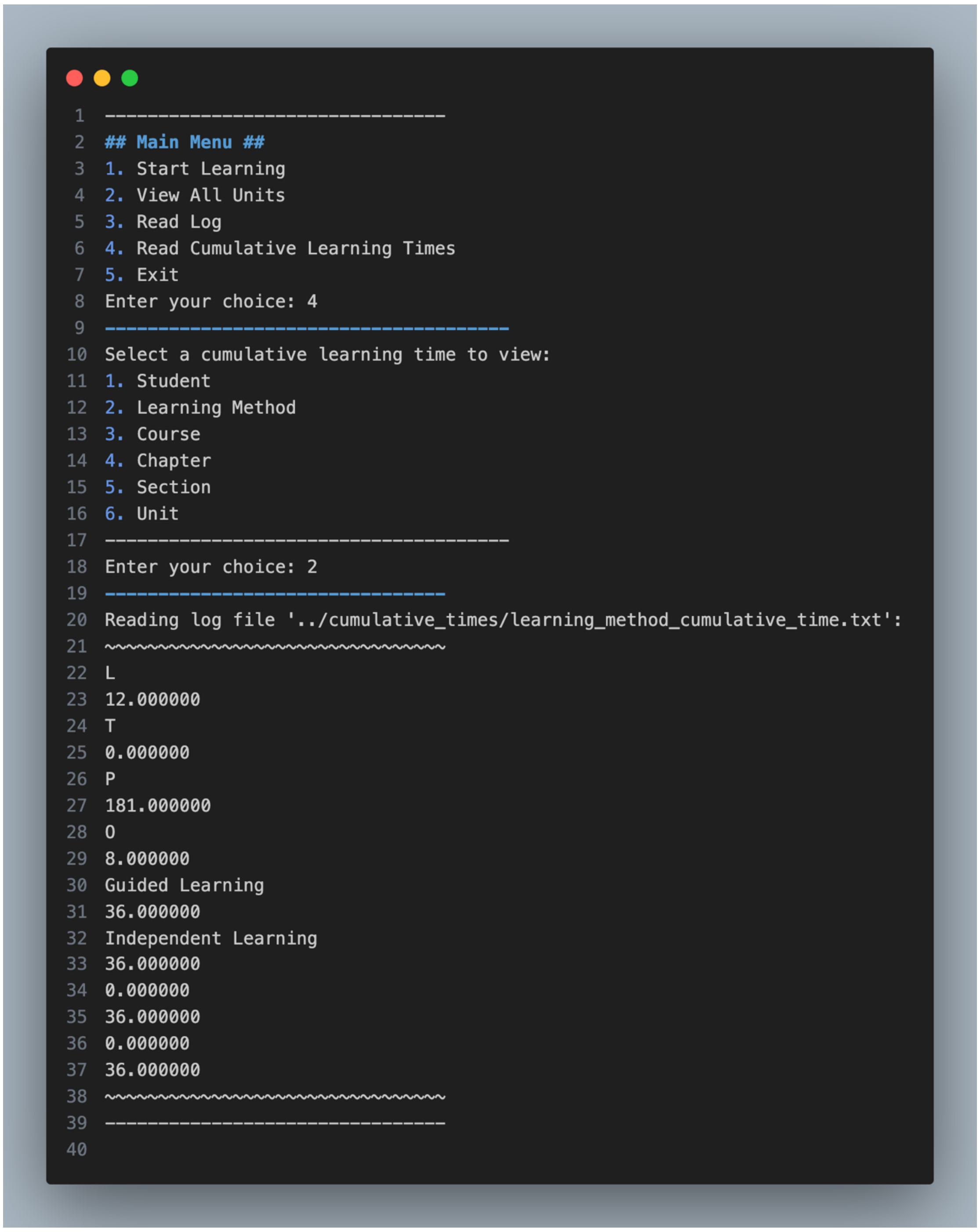
3. Read log



```
Welcome to the Knowledge Point Learning Tracking System (KPLTS)!
--------------------------------
## Main Menu ##
1. Start Learning
2. View All Units
3. Read Log
4. Read Cumulative Learning Times
5. Exit
Enter your choice: 3
--------------------------------
Reading log file '../logs/learning_log.txt':
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--------------------------------------
Student ID: CYS2309198
Learning Method: P
Start Time: Tue Jan  2 16:55:52 2024
End Time: Tue Jan  2 16:57:30 2024
Elapsed Time: 98.00

Course: 01 CST103
Chapter: 02 Types, Operators and Expressions
Section: 03 Constants
Unit: 2: Floating-Point Constants.txt
--------------------------------------
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--------------------------------
```

4. Read cumulative learning times



```
----------------------------------
## Main Menu ##
1. Start Learning
2. View All Units
3. Read Log
4. Read Cumulative Learning Times
5. Exit
Enter your choice: 4
----------------------------------------
Select a cumulative learning time to view:
1. Student
2. Learning Method
3. Course
4. Chapter
5. Section
6. Unit
----------------------------------------
Enter your choice: 2
----------------------------------
Reading log file '../cumulative_times/learning_method_cumulative_time.txt':
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
L
12.000000
T
0.000000
P
181.000000
O
8.000000
Guided Learning
36.000000
Independent Learning
36.000000
0.000000
36.000000
0.000000
36.000000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
----------------------------------
```

6. Conclusion

The development and design of the Knowledge Point Learning Tracking System (KPLTS) was guided by the fundamental goals of creating a robust, user-friendly, and adaptable platform for tracking learning sessions and analyzing cumulative learning data.

The modular architecture of the system is a key aspect that has greatly enhanced its scalability, readability, and maintainability. The division of functionality into separate modules, from handling file operations to tracking learning sessions, has contributed significantly to the manageability of the code, allowing for easier updates and debugging without compromising the integrity of the system.

In addition, the use of file-based storage has proven to be instrumental in ensuring the extensibility of the system. The hierarchical organization of learning materials within directories and files has facilitated easy navigation and addition of new elements, ensuring sustainability and flexibility to accommodate future course expansions.

The inclusion of a time tracking mechanism and detailed data logging has provided valuable insights into learning activities and cumulative study time. However, there are still areas where the system can be improved and expanded to provide a more comprehensive and intuitive learning experience.

7. Future Work

1. Improve reporting: Implement graphing or data visualization for better insight into cumulative learning times.
2. Real-time analytics: Incorporate real-time analytics to dynamically track and analyze learning patterns.
3. Graphical interface: Continuously refine the user interface for improved user experience and intuitive navigation, such as using GUI library to create a more friendly graphical interface.