TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin

.....🕮.....

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC

LẬP TRÌNH PYTHON

Sinh viên: Nguyễn Văn Thứ

Lớp: K58KTP

Giáo viên GIẢNG DẠY: TS. Nguyễn Văn Huy

Link QR Github:

Thái Nguyên – 2025

TRƯỜNG ĐHKTCN **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

KHOA ĐIỆN TỬ ***Độc lập – Tự do – Hạnh phúc***

Thái Nguyên, ngày ... tháng ... năm 2025

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC: LẬP TRÌNH PYTHON

BỘ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên: Nguyễn Văn Thứ

Lớp: *K58KTP* *Ngành: Kỹ thuật máy tính*

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy

Ngày giao đề: 25/05/2025 *Ngày hoàn thành: 10/06/2025*

Tên đề tài: *Chuyển chương trình Trivia Challenge (đọc câu hỏi từ file, Chapter 7) thành ứng dụng GUI.*

Yêu cầu:

Đầu vào – đầu ra:

- *Đầu vào: File định dạng câu hỏi (text) giống mẫu trong sách*
- *Đầu ra: Câu hỏi hiển thị, nhập ô đáp án, hiện điểm.*

Tính năng yêu cầu:

- *Đọc file, bắt lỗi file không tồn tại hoặc format sai.*
- *GUI: Label câu hỏi, Entry đáp án, nút "Nộp", Label điểm.*
- *Tính điểm, chuyển câu hỏi kế tiếp.*
- *Nút "Kết thúc" hiển thị tổng điểm.*

Kiểm tra & kết quả mẫu:

- *Câu "What is 2+2?" → Nhập "4" → +1 điểm, sang câu kế.*
- *Nhập sai → không cộng điểm.*

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Xếp loại: .....Điểm: .....

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN 2
LỜI CAM ĐOAN 3
LỜI NÓI ĐẦU 4
CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI 5
1.1 Giới thiệu đầu bài 5
1.2 Danh mục bài báo cáo 6
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 7
2.1 Xử lý tệp văn bản (File I/O) 7
2.2 Danh sách (List) và xử lý chuỗi (String) 7
2.3 Lập trình hướng đối tượng (OOP) 7
2.4 Giao diện người dùng (GUI) với Tkinter 8
2.5 Kiểm soát luồng chương trình & logic trò chơi 8
2.6 Kỹ thuật chia module 8
CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH 10
3.1 Sơ đồ khối hệ thống 10
3.2 Sơ đồ khối các thuật toán chính 12
3.3 Cấu trúc dữ liệu 15
3.4 Chương trình 16
CHƯƠNG 4. THỰC NGHIỆM VÀ KẾT LUẬN 24
4.1 Thực nghiệm chương trình 24
4.2 Kết luận 27
TÀI LIỆU THAM KHẢO 30

LỜI CẢM ƠN

Em xin chân thành cảm ơn thầy **TS. Nguyễn Văn Huy** khoa Điện tử – Trường đại học Kỹ thuật công nghiệp đã tận tình giảng dạy, cung cấp những kiến thức quý báu trong suốt quá trình học tập.

Em xin bày tỏ lòng biết ơn sâu sắc đến thầy **TS. Nguyễn Văn Huy**, người đã trực tiếp hướng dẫn, hỗ trợ và tạo điều kiện thuận lợi để em hoàn thành đề tài báo cáo này.

Đồng thời, em cũng xin cảm ơn các anh/chị, bạn bè và những cá nhân đã góp ý, chia sẻ tài liệu, cũng như động viên em trong suốt quá trình thực hiện đề tài.

Do thời gian và kiến thức còn hạn chế, báo cáo không tránh khỏi thiếu sót. Em rất mong nhận được sự góp ý của Thầy Cô để em hoàn thiện hơn trong các báo cáo sau.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Em xin cam đoan rằng báo cáo với đề tài **"*Chuyển chương trình Trivia Challenge (đọc câu hỏi từ file, Chapter 7) thành ứng dụng GUI.*"** Là kết quả của quá trình học tập và nghiên cứu của cá nhân em dưới sự hướng dẫn của Thầy **TS. Nguyễn Văn Huy.**

Các số liệu, kết quả trong báo cáo là trung thực, được trích dẫn rõ nguồn gốc (nếu có). Tôi hoàn toàn chịu trách nhiệm trước Nhà trường và pháp luật về tính chính xác, trung thực của nội dung báo cáo này.

Em cam kết không sao chép hoặc vi phạm bản quyền của bất kỳ cá nhân/tổ chức nào.

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

LỜI NÓI ĐẦU

Trong thời đại công nghệ phát triển không ngừng, ngôn ngữ lập trình Python ngày càng được ưa chuộng nhờ tính đơn giản, dễ học nhưng vẫn rất mạnh mẽ và linh hoạt. Qua quá trình học tập môn **Lập trình Python**, em đã được tiếp cận với nhiều kiến thức nền tảng và ứng dụng thực tiễn trong việc xây dựng các chương trình phần mềm.

Để vận dụng hiệu quả những kiến thức đã học, em chọn thực hiện đề tài: **"*Chuyển chương trình Trivia Challenge (đọc câu hỏi từ file, Chapter 7) thành ứng dụng GUI.*"** – một trò chơi trắc nghiệm có giao diện đồ họa đơn giản, kết hợp xử lý đọc/ghi dữ liệu từ tệp. Đề tài này không chỉ giúp em củng cố lại kiến thức về lập trình hướng đối tượng, xử lý tệp và giao diện người dùng với thư viện tkinter, mà còn rèn luyện tư duy lập trình, khả năng xử lý lỗi và tổ chức mã nguồn.

Báo cáo này trình bày chi tiết quá trình phân tích, thiết kế và xây dựng trò chơi, đồng thời phản ánh quá trình học tập và nghiên cứu nghiêm túc của em. Tuy đã cố gắng hoàn thiện tốt nhất trong khả năng, nhưng không thể tránh khỏi thiếu sót. Em rất mong nhận được sự góp ý từ Thầy/Cô để em có thể cải thiện hơn trong những lần sau.

Em xin chân thành cảm ơn!

CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI

1.1 Giới thiệu đầu bài

❖ **Đầu bài: Trivia Challenge có GUI & file I/O**

Chuyển trương trình Trivia Challenge (đọc câu hỏi từ file, chapter 7) thành ứng dụng GUI.

❖ **Đầu vào – đầu ra**

- Đầu vào: file định dạng câu hỏi (text) giống mẫu trong sách.
- Đầu ra: Câu hỏi hiển thị, ô nhập đáp án, hiện điểm.

❖ **Tính năng yêu cầu**

- Đọc file, bắt lỗi file không tồn tại hoặc format sai.
- GUI: Label câu hỏi, Entry đáp án, nút "Nộp", Label điểm.
- Tính điểm, chuyển câu hỏi kế tiếp.
- Nút "Kết thúc" hiển thị tổng điểm.

❖ **Kiểm tra & kết quả mẫu**

- Câu "What is 2+2?" → Nhập "4" → +1 điểm, sang câu kế.
- Nhập sai → không cộng điểm.

❖ **Các bước kiểm tra**

- Viết module đọc file theo cấu trúc open_file(), next_block().
- Class GUI với tkinter; Frame cho câu hỏi, Entry và Button.
- Kết nối logic và giao diện, cập nhật câu hỏi & điểm.

❖ **Thách thức**

- Hiểu và xử lý định dạng file văn bản chứa câu hỏi, đảm bảo không lỗi khi đọc từng dòng.
- Xây dựng giao diện người dùng bằng thư viện tkinter, phải quản lý trạng thái hiển thị câu hỏi và điểm.
- Xử lý luồng logic khi chuyển câu hỏi, nộp đáp án, và kết thúc trò chơi.
- Đảm bảo chương trình không bị dừng khi file bị thiếu, sai định dạng hoặc người dùng nhập sai kiểu dữ liệu.
- Làm quen với cách tổ chức chương trình theo hướng module hóa (tách file xử lý I/O và giao diện GUI riêng).

❖ **Vận dụng**

- Xử lý file với open(), readline(), kiểm tra lỗi file (try-except).

- Làm việc với danh sách (list), chuỗi (str), và câu lệnh điều kiện (if, else).
- Tổ chức chương trình với **hàm** và **lớp** (class, self, __init__()).
- Xây dựng giao diện đồ họa (GUI) với thư viện **Tkinter**: Label, Entry, Button, messagebox.
- Quản lý trạng thái trò chơi: điểm số, câu hiện tại, kết thúc trò chơi.
- Làm việc với hàm zip() và join() để xử lý danh sách động.

1.2 Danh mục bài báo cáo

Nội dung báo cáo được chia làm 4 chương:

Chương 1: Giới thiệu đầu bài

Giới thiệu nội dung cảu đề bài và nêu ra các thách thức và kiến thức vận dụng vào bài để thực hiện bài toán là gì.

Chương 2: Cơ sở lý thuyết

Nêu ra được nhưng nội dung lý thuyết nào cần áp dụng để xây dựng và triển khai bài toán.

Chương 3: Thiết kế và xây dựng chương trình

Nội dung thiết kế và xây dụng chương tình thông qua các công cụ nào và hướng thiết kế xây dụng là gì.

Chương 4: Thực nghiệm và kết luận.

Triển khai vào thực nghiệm và đánh giá chi tiết kết quả đã đạt được sau khi hoàn thành.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Xử lý tệp văn bản (File I/O)

- Sử dụng hàm open() để mở file, readline() để đọc từng dòng dữ liệu.
- Áp dụng khối try-except để xử lý lỗi file không tồn tại hoặc sai định dạng.
- Đọc file theo cấu trúc định sẵn và chia thành từng khối dữ liệu (câu hỏi, đáp án, câu trả lời đúng).

Ví dụ:

file = open("cau_hoi.txt", "r")

dong = file.readline()

2.2 Danh sách (List) và xử lý chuỗi (String)

- Lưu trữ các phương án trả lời dưới dạng danh sách (list). Mỗi câu hỏi có 4 đáp án ⇒ lưu trong 1 danh sách để dễ truy xuất:
- Khi cần hiển thị, dùng for, zip() hoặc chỉ số để in ra từng đáp án.

Ví dụ:

Options = [“4”, “3”, “2”, “1”]

- Xử lý chuỗi bằng .strip(), .join(), f-string để hiển thị dữ liệu một cách đẹp và rõ ràng.
  - .strip() để bỏ ký tự xuống dòng, khoảng trắng thừa khi đọc từ file.
  - f"Chuỗi {biến}" để định dạng văn bản đẹp.
  - “\n".join(danh_sach) để nối danh sách thành chuỗi nhiều dòng.

2.3 Lập trình hướng đối tượng (OOP)

- Sử dụng **class** để tạo ứng dụng có cấu trúc rõ ràng.
- class TriviaGame: là lớp chứa toàn bộ trò chơi.
- __init__(self) là hàm khởi tạo (constructor): tạo cửa sổ, điểm, câu hỏi đầu tiên.
- Các **phương thức (methods)** như:
  - load_next_question(): nạp câu hỏi mới.
  - submit_answer(): xử lý khi người dùng nhấn “Nộp”.
  - end_game(): hiển thị điểm và kết thúc.
  - Lập trình theo hướng đối tượng giúp:
  - Tách biệt dữ liệu và hành vi.

- Dễ bảo trì, mở rộng hoặc tái sử dụng.

2.4 Giao diện người dùng (GUI) với Tkinter

- Tkinter là thư viện GUI cơ bản trong Python.
- Các widget được sử dụng:
  - Label: hiển thị câu hỏi.
  - Entry: nhập đáp án.
  - Button: nộp câu trả lời, kết thúc trò chơi.
  - Messagebox: hiển thị thông báo điểm.

Ví dụ:

self.question_label = tk.Label(self.root, text="Câu hỏi")

self.answer_entry = tk.Entry(self.root)

self.submit_button = tk.Button(self.root, text="Nộp", command=self.submit_answer)

2.5 Kiểm soát luồng chương trình & logic trò chơi

- Mỗi lần người dùng nhấn "Nộp", chương trình:
  - Kiểm tra đáp án.
  - Cập nhật điểm.
  - Hiển thị câu hỏi tiếp theo hoặc kết thúc nếu đã hết.

2.6 Kỹ thuật chia module

Tách riêng phần đọc file (trivia_io.py) và phần giao diện (main.py) giúp chương trình dễ đọc, dễ bảo trì.

- Để làm chương trình gọn gàng, bạn tách thành **nhiều file**:
  - trivia_io.py: chứa hàm đọc file như open_file() và next_block().
  - main.py: chứa GUI và lớp TriviaGame.
  - Dễ quản lý, đọc hiểu.
  - Chạy thử hoặc kiểm tra từng phần riêng biệt.

Tóm tắt chương:

Các kiến thức trên không chỉ giúp hoàn thành bài tập lớn mà còn là nền tảng cho việc xây dựng các ứng dụng thực tế có giao diện và xử lý dữ liệu động. Chương này trình bày các kiến thức lập trình Python được vận dụng trong quá trình xây dựng ứng dụng Trivia Challenge. Cụ thể, bài toán sử dụng kỹ thuật xử lý tệp tin để đọc dữ liệu câu hỏi từ file, xử lý chuỗi và danh sách để hiển thị và kiểm tra đáp án. Lập trình hướng đối tượng được áp dụng để tổ chức mã nguồn có cấu trúc, giúp quản lý trạng thái trò chơi và các chức năng GUI. Giao diện người dùng được xây dựng với thư viện Tkinter, kết hợp các widget như Label, Entry, Button, và messagebox. Cuối cùng, kỹ thuật chia module giúp tăng tính rõ ràng và dễ bảo trì của chương trình.

CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

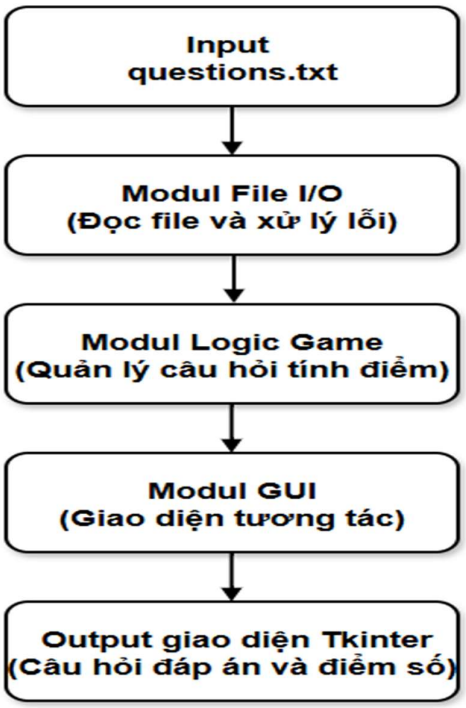
3.1 Sơ đồ khối hệ thống

3.1.1 Mô tả các module chính trong chương trình

❖ ***Mô tả sơ đồ khối***

Sơ đồ khối hệ thống của chương trình Trivia Challenge bao gồm các thành phần chính sau:

- Input: File câu hỏi (questions.txt).
- Processing: Các module xử lý logic và giao diện.
- Output: Giao diện GUI hiển thị câu hỏi, nhập đáp án, và kết quả.



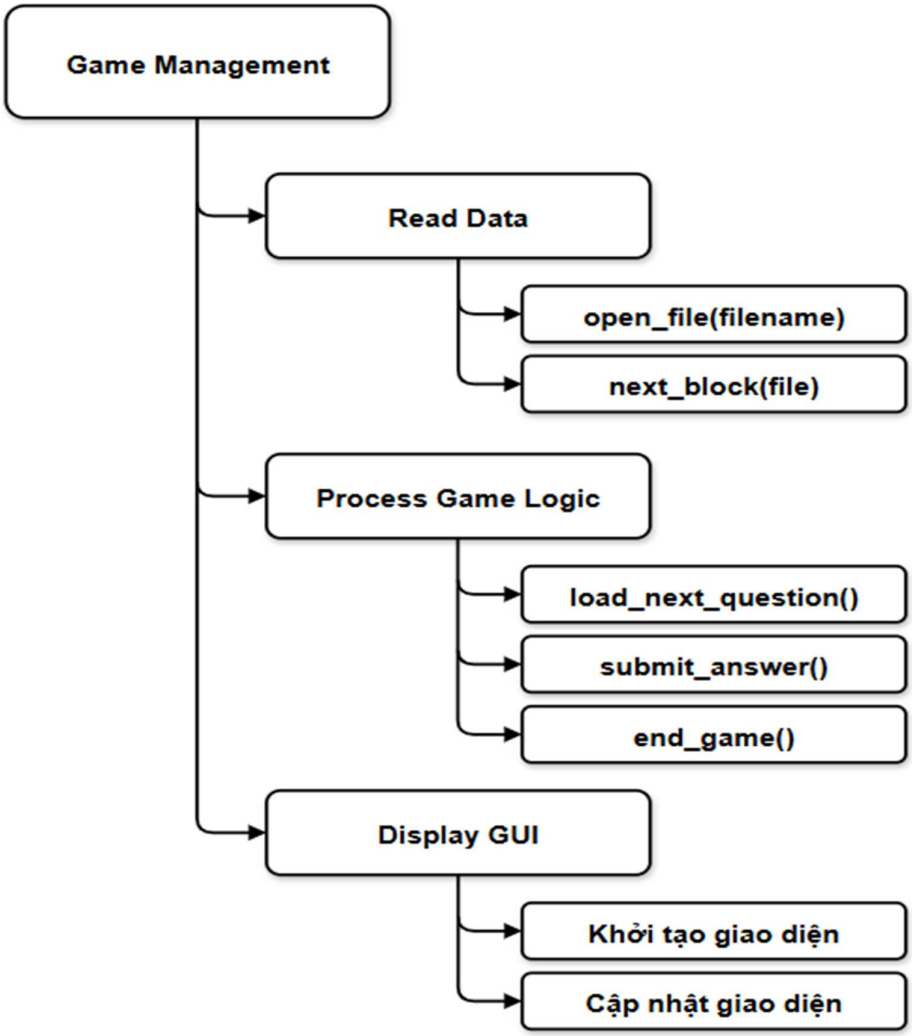
Hình 3.1 Sơ đồ khối hệ thống

3.1.2 Biểu đồ phân cấp chức năng (tức các chức năng chính trong chương trình)

❖ ***Mô tả biểu đồ phân cấp chức năng***

Biểu đồ phân cấp chức năng sẽ hiển thị các chức năng chính của chương trình Trivia Challenge theo thứ tự phân cấp từ cao đến thấp:

- Chức năng cấp cao nhất: Quản lý trò chơi (Game Management).
- Chức năng cấp trung: Đọc dữ liệu, xử lý logic, và giao diện.
- Chức năng cấp thấp: Các hàm cụ thể thực hiện từng nhiệm vụ.



Hình 3.2 Biểu đồ phân cấp chức năng

❖ ***Dựa trên biểu đồ phân cấp, dưới đây là mô tả chi tiết từng chức năng:***

1. **Game Management (Quản lý trò chơi)**:
   - Mô tả: Là chức năng cấp cao nhất, điều phối toàn bộ hoạt động của trò chơi, bao gồm đọc dữ liệu, xử lý logic, và hiển thị giao diện.
   - Mục đích: Đảm bảo luồng trò chơi từ bắt đầu đến kết thúc diễn ra liền mạch.
2. **Read Data (Đọc dữ liệu)**:
   - Mô tả: Chịu trách nhiệm lấy thông tin câu hỏi từ file questions.txt.
   - Chức năng con:

  - open_file(filename): Mở file và xử lý lỗi (FileNotFoundError, định dạng sai).
  - next_block(file): Đọc từng khối câu hỏi (tiêu đề, câu hỏi, đáp án, giải thích) và trả về dictionary.
- Đầu vào: File questions.txt.
- Đầu ra: Dữ liệu câu hỏi dưới dạng cấu trúc.

3. **Process Game Logic (Xử lý logic trò chơi)**:
   - Mô tả: Quản lý luồng chơi, bao gồm kiểm tra đáp án, tính điểm, và chuyển câu hỏi.
   - Chức năng con:
     - load_next_question(): Nạp câu hỏi mới và cập nhật giao diện.
     - submit_answer(): So sánh đáp án người dùng với đáp án đúng, cập nhật điểm, và chuyển câu.
     - end_game(): Hiển thị tổng điểm và kết thúc chương trình.
   - Đầu vào: Đáp án từ người dùng, dữ liệu câu hỏi.
   - Đầu ra: Cập nhật điểm số, thông báo, chuyển trạng thái trò chơi.
4. **Display GUI (Hiển thị giao diện)**:
   - Mô tả: Tạo và quản lý giao diện người dùng bằng tkinter, bao gồm hiển thị câu hỏi, nhập đáp án, và tương tác.
   - Chức năng con:
     - Khởi tạo giao diện: Tạo cửa sổ, label, entry, button với thiết kế đẹp.
     - Cập nhật giao diện: Cập nhật câu hỏi, điểm số, và thông báo khi có sự kiện.
   - Đầu vào: Dữ liệu câu hỏi, hành động người dùng.
   - Đầu ra: Giao diện trực quan với thông tin trò chơi.

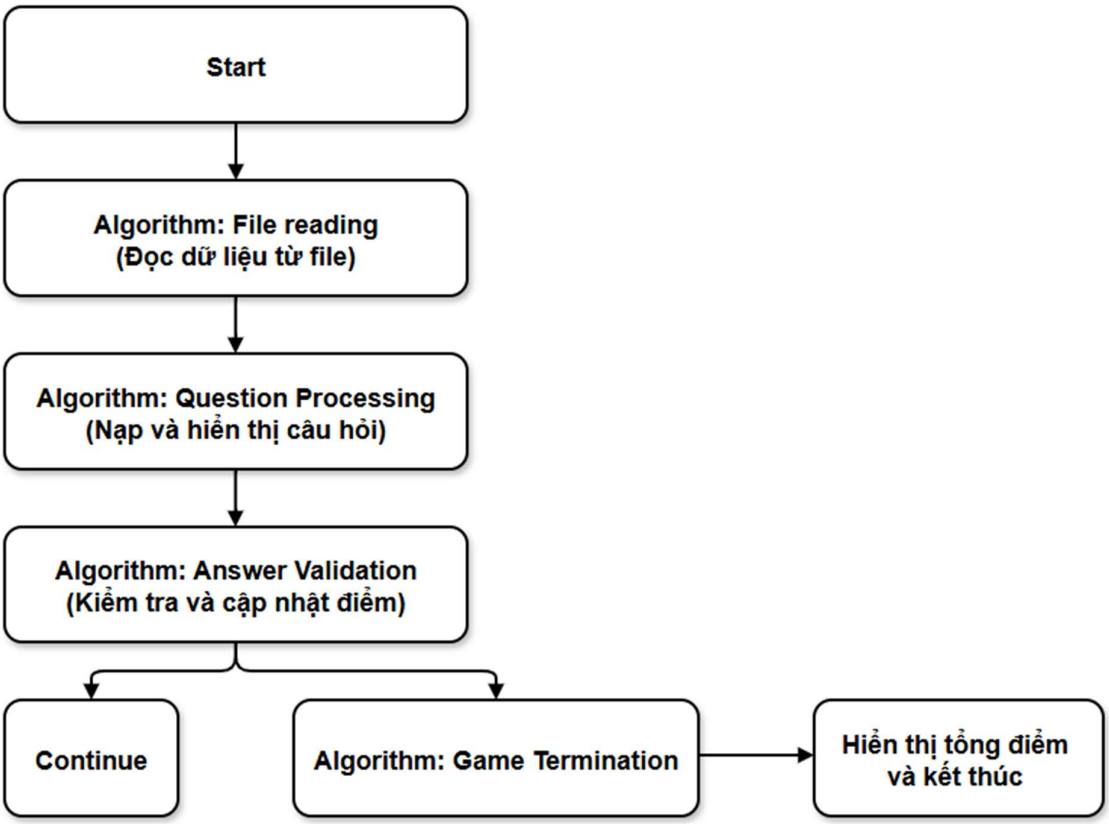
3.2 Sơ đồ khối các thuật toán chính

❖ ***Mô tả sơ đồ khối***

Sơ đồ khối các thuật toán chính sẽ thể hiện các thuật toán cốt lõi trong chương trình, bao gồm cách chúng tương tác để thực hiện chức năng của trò chơi. Các khối chính bao gồm:

- Algorithm: File Reading (Đọc file).
- Algorithm: Question Processing (Xử lý câu hỏi).

- Algorithm: Answer Validation (Kiểm tra đáp án).
- Algorithm: Game Termination (Kết thúc trò chơi.



Hình 3.3 Sơ đồ khối các thuật toán chính

❖ ***Mô tả các khối thuật toán chính, quan hệ đầu vào - đầu ra, và chức năng***

1. Algorithm: File Reading (Đọc file)

- Mô tả: Thuật toán này chịu trách nhiệm đọc dữ liệu câu hỏi từ file questions.txt và xử lý các lỗi liên quan.
- Chức năng:
  - Mở file và kiểm tra sự tồn tại.
  - Đọc từng khối câu hỏi (gồm tiêu đề, câu hỏi, 4 đáp án, đáp án đúng, giải thích) và trả về dưới dạng cấu trúc dữ liệu.
- Đầu vào: Đường dẫn file questions.txt.
- Đầu ra: Đối tượng file (cho open_file) hoặc dictionary chứa thông tin câu hỏi (cho next_block).
- Thuật toán chi tiết:

  - open_file(filename): Sử dụng try-except để mở file, trả về None nếu lỗi.
  - next_block(file): Đọc 8 dòng (tiêu đề, câu hỏi, 4 đáp án, đáp án, giải thích, dòng trống), kiểm tra định dạng, và tạo dictionary.
- Quan hệ: Cung cấp dữ liệu đầu vào cho thuật toán "Question Processing".

2. Algorithm: Question Processing (Xử lý câu hỏi)

- Mô tả: Thuật toán này nạp câu hỏi tiếp theo từ file và hiển thị lên giao diện.
- Chức năng:
  - Lấy câu hỏi từ file thông qua next_block.
  - Định dạng và hiển thị câu hỏi cùng các đáp án trên giao diện GUI.
  - Xóa ô nhập đáp án để chuẩn bị cho câu hỏi mới.
- Đầu vào: Dictionary câu hỏi từ next_block.
- Đầu ra: Cập nhật giao diện với câu hỏi và đáp án mới, hoặc thông báo kết thúc nếu hết câu hỏi.
- Thuật toán chi tiết:
  - Gọi next_block(file) để lấy câu hỏi.
  - Nếu có câu hỏi, tạo chuỗi văn bản với định dạng (câu hỏi + A, B, C, D).
  - Cập nhật question_label và xóa answer_entry.
  - Nếu hết câu hỏi, gọi messagebox.showinfo và đóng ứng dụng.
- **Quan hệ**: Kết quả được truyền sang "Answer Validation" để người dùng nhập đáp án.

3. Algorithm: Answer Validation (Kiểm tra đáp án)

- Mô tả: Thuật toán này kiểm tra đáp án người dùng nhập, cập nhật điểm số, và quyết định tiếp tục hoặc kết thúc.
- Chức năng:
  - So sánh đáp án người dùng với đáp án đúng.
  - Cập nhật điểm nếu đúng, hiển thị thông báo (đúng/sai).
  - Chuyển sang câu hỏi tiếp theo hoặc kết thúc nếu hết câu hỏi.
- Đầu vào: Đáp án từ answer_entry, dictionary câu hỏi hiện tại.
- Đầu ra: Cập nhật điểm số (score_label), thông báo qua messagebox, và gọi lại "Question Processing".
- Thuật toán chi tiết:

- Lấy user_answer từ answer_entry, chuyển thành chữ in hoa.
- So sánh với correct_answer từ dictionary.
- Nếu đúng, tăng score và hiển thị "Đúng"; nếu sai, hiển thị "Sai" với đáp án đúng.
- Cập nhật score_label và gọi load_next_question.

- Quan hệ: Kết quả (tiếp tục hoặc kết thúc) được truyền sang "Question Processing" hoặc "Game Termination".

4. Algorithm: Game Termination (Kết thúc trò chơi)

- Mô tả: Thuật toán này xử lý khi trò chơi kết thúc, hiển thị tổng điểm và đóng ứng dụng.
- Chức năng:
  - Hiển thị tổng điểm của người chơi.
  - Đóng cửa sổ giao diện và kết thúc chương trình.
- Đầu vào: Giá trị score hiện tại, tín hiệu từ "Answer Validation" (khi hết câu hỏi) hoặc nút "Kết thúc".
- Đầu ra: Thông báo tổng điểm qua messagebox, đóng ứng dụng.
- Thuật toán chi tiết:
  - Gọi messagebox.showinfo với thông điệp chứa tổng điểm.
  - Gọi self.root.destroy() để đóng cửa sổ.
- Quan hệ: Kết thúc luồng xử lý, không có đầu ra tiếp theo.

3.3 Cấu trúc dữ liệu

❖ ***Mô tả chi tiết cấu trúc dữ liệu***

1. Nguồn dữ liệu:

- Dữ liệu được lưu trong file questions.txt với định dạng cố định (8 dòng mỗi câu hỏi).
- Khi đọc, dữ liệu được chuyển thành dictionary trong Python (ví dụ: {"question": "What is 2+2?", "options": ["A. 1", "B. 2", "C. 3", "D. 4"], "answer": "D", ...}).

2. Cấu trúc dữ liệu trong chương trình:

- Dictionary: Sử dụng để lưu thông tin từng câu hỏi trong bộ nhớ khi đọc từ file.

- List (tạm thời): Danh sách các câu hỏi có thể được xây dựng nếu đọc toàn bộ file vào bộ nhớ (nhưng trong code hiện tại, chỉ đọc lần lượt bằng next_block).
- Biến đơn: score (số nguyên) để theo dõi điểm số.

3. Lý do không có bảng thực sự:

- Chương trình không sử dụng cơ sở dữ liệu (như SQLite hay MySQL), nên không có bảng lưu trữ cố định.
- Dữ liệu chỉ được xử lý tạm thời trong bộ nhớ RAM khi chương trình chạy.

4. Khả năng mở rộng:

- Nếu muốn, bạn có thể chuyển đổi file questions.txt thành file CSV hoặc cơ sở dữ liệu, nơi bảng "Questions" sẽ được lưu trữ vĩnh viễn với các cột như trên.

3.4 Chương trình

❖ ***Nội dung các hàm trong chương trình chính***

1. Hàm: open_file(filename)

- Thuộc module: Module File I/O (đọc file).
- Chức năng: Mở file câu hỏi (questions.txt) và xử lý các lỗi liên quan (file không tồn tại, lỗi đọc file).
- Tham số đầu vào:
  - filename (String): Đường dẫn hoặc tên file chứa câu hỏi (ví dụ: "questions.txt").
- Giá trị trả về:
  - file (File object): Đối tượng file nếu mở thành công.
  - None: Nếu có lỗi (file không tồn tại hoặc lỗi khác).
- Vai trò trong chương trình: Khởi tạo quá trình đọc dữ liệu bằng cách cung cấp đối tượng file cho các hàm khác (như next_block).

Hàm open_file(filename):

```
def open_file(filename):
    try:
        file = open(filename, 'r', encoding='utf-8')
        return file
    except FileNotFoundError:
        print(f"Lỗi: File '{filename}' không tồn tại!")
        return None
    except Exception as e:
        print(f"Lỗi khi mở file: {e}")
        return None
```

2. Hàm: next_block(file)

- Thuộc module: Module File I/O (đọc file).
- Chức năng: Đọc một khối câu hỏi từ file (gồm tiêu đề, câu hỏi, 4 đáp án, đáp án đúng, giải thích) và trả về dưới dạng dictionary.
- Tham số đầu vào:
  - file (File object): Đối tượng file đã được mở bởi open_file.
- Giá trị trả về:
  - dictionary (Dict): Chứa thông tin câu hỏi (key: "title", "question", "options", "answer", "explanation").
  - None: Nếu hết dữ liệu hoặc định dạng file sai.
- Vai trò trong chương trình: Cung cấp dữ liệu câu hỏi cho module logic trò chơi (qua load_next_question).

Hàm next_block(file):

```
def next_block(file):
    title = file.readline().strip()
    if not title:
        return None
```

```
    question = file.readline().strip()
    options = [file.readline().strip() for _ in range(4)]
    answer_line = file.readline().strip()
    explanation = file.readline().strip()

    if not answer_line.startswith("Đáp án: "):
        print(f"Lỗi định dạng: Dòng đáp án không đúng định dạng: {answer_line}")
        return None

    answer = answer_line.replace("Đáp án: ", "").strip()

    empty_line = file.readline().strip()
    if empty_line and empty_line != "":
        print(f"Lỗi định dạng: Dòng trống bị thiếu sau câu hỏi: {title}")
        return None

    return {
        "title": title,
        "question": question,
        "options": options,
        "answer": answer,
        "explanation": explanation
    }
```

3. Hàm: __init__(self, root) (Phương thức khởi tạo của class TriviaGame)

- Thuộc module: Module GUI và Logic Game.
- Chức năng: Khởi tạo giao diện GUI và các biến cần thiết cho trò chơi, đồng thời nạp câu hỏi đầu tiên.
- Tham số đầu vào: root (tk.Tk): Cửa sổ chính của giao diện tkinter.
- Giá trị trả về: Không có (phương thức khởi tạo).
- Vai trò trong chương trình: Thiết lập môi trường ban đầu, bao gồm giao diện, biến score, và nạp dữ liệu từ file.

Hàm __init__(self, root):

```
def __init__(self, root):
    self.root = root
    self.root.title("Trivia Challenge")
    self.root.geometry("450x400")
    self.root.configure(bg="#f0f0f0")

    self.score = 0
    self.current_question = None
    self.file = open_file("questions.txt")
    if not self.file:
        messagebox.showerror("Lỗi", "Không thể mở file questions.txt!")
        self.root.destroy()
        return

    self.main_frame = tk.Frame(self.root, bg="#f0f0f0")
    self.main_frame.pack(expand=True, padx=20, pady=20)
```

```
self.title_label = tk.Label(self.main_frame, text="Trivia Challenge", font=("Arial", 16, "bold"), fg="#2c3e50", bg="#f0f0f0")
self.title_label.pack(pady=(0, 20))

self.score_label = tk.Label(self.main_frame, text=f"Điểm: {self.score}", font=("Arial", 12), fg="#2c3e50", bg="#ffffff", padx=10, pady=5, relief="groove")
self.score_label.pack(pady=10)

self.question_label = tk.Label(self.main_frame, text="", font=("Arial", 12), fg="#2c3e50", bg="#ffffff", wraplength=350, justify="left", padx=10, pady=10, relief="groove")
self.question_label.pack(pady=10)

self.answer_entry = tk.Entry(self.main_frame, font=("Arial", 12), width=20)
self.answer_entry.pack(pady=10)

self.submit_button = tk.Button(self.main_frame, text="Nộp", font=("Arial", 12), bg="#3498db", fg="white", padx=10, pady=5, relief="raised", command=self.submit_answer)
self.submit_button.pack(pady=10)

self.end_button = tk.Button(self.main_frame, text="Kết thúc", font=("Arial", 12), bg="#e74c3c", fg="white", padx=10, pady=5, relief="raised", command=self.end_game)
self.end_button.pack(pady=10)
```

```
self.load_next_question()
```

4. Hàm: load_next_question(self)

- Thuộc module: Module Logic Game.
- Chức năng: Nạp câu hỏi tiếp theo từ file và cập nhật giao diện.
- Tham số đầu vào: Không (sử dụng self.file đã khởi tạo).
- Giá trị trả về: Không.
- Vai trò trong chương trình: Cung cấp câu hỏi mới cho giao diện và kiểm tra điều kiện kết thúc.

Hàm load_next_question(self):

```
def load_next_question(self):
    self.current_question = next_block(self.file)
    if self.current_question:
        question_text = f"{self.current_question['question']}\n" \
                        f"A. {self.current_question['options'][0]}\n" \
                        f"B. {self.current_question['options'][1]}\n" \
                        f"C. {self.current_question['options'][2]}\n" \
                        f"D. {self.current_question['options'][3]}"
        self.question_label.config(text=question_text)
        self.answer_entry.delete(0, tk.END)
    else:
        messagebox.showinfo("Kết thúc", f"Trò chơi kết thúc! Điểm của bạn: {self.score}")
        self.root.destroy()
```

5. Hàm: submit_answer(self)

- Thuộc module: Module Logic Game.
- Chức năng: Kiểm tra đáp án người dùng, cập nhật điểm, và chuyển sang câu hỏi tiếp theo.

- Tham số đầu vào: Không (sử dụng self.current_question và self.answer_entry).
- Giá trị trả về: Không.
- Vai trò trong chương trình: Xử lý tương tác người dùng khi nhấn "Nộp".

Hàm submit_answer(self):

```
def submit_answer(self):
    if self.current_question:
        user_answer = self.answer_entry.get().strip().upper()
        correct_answer = self.current_question['answer']
        if user_answer == correct_answer:
            self.score += 1
            messagebox.showinfo("Đúng", "Chúc mừng! Đáp án đúng!", parent=self.root)
        else:
            messagebox.showinfo("Sai", f"Đáp án sai! Đáp án đúng là: {correct_answer}", parent=self.root)
        self.score_label.config(text=f"Điểm: {self.score}")
        self.load_next_question()
```

6. Hàm: end_game(self)

- Thuộc module: Module Logic Game.
- Chức năng: Hiển thị tổng điểm và kết thúc chương trình khi người dùng nhấn "Kết thúc".
- Tham số đầu vào: Không (sử dụng self.score).
- Giá trị trả về: Không.
- Vai trò trong chương trình: Kết thúc trò chơi theo yêu cầu người dùng.

Hàm end_game(self):

```
def end_game(self):
```

messagebox.showinfo("Kết thúc", f"Trò chơi kết thúc! Điểm của bạn: {self.score}", parent=self.root)

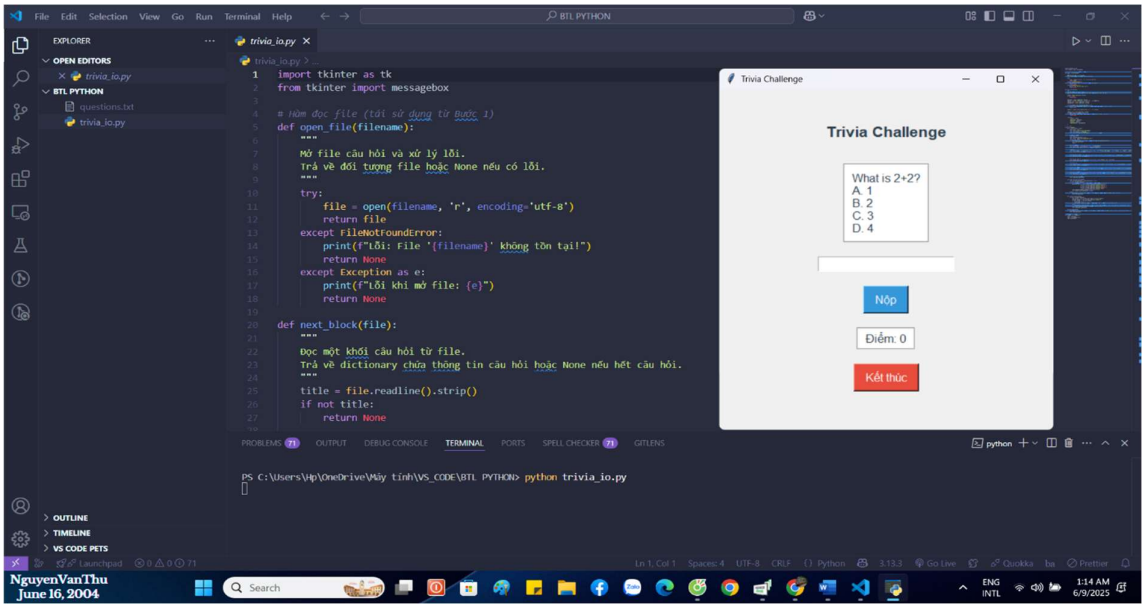
self.root.destroy()

Tóm tắt chương:

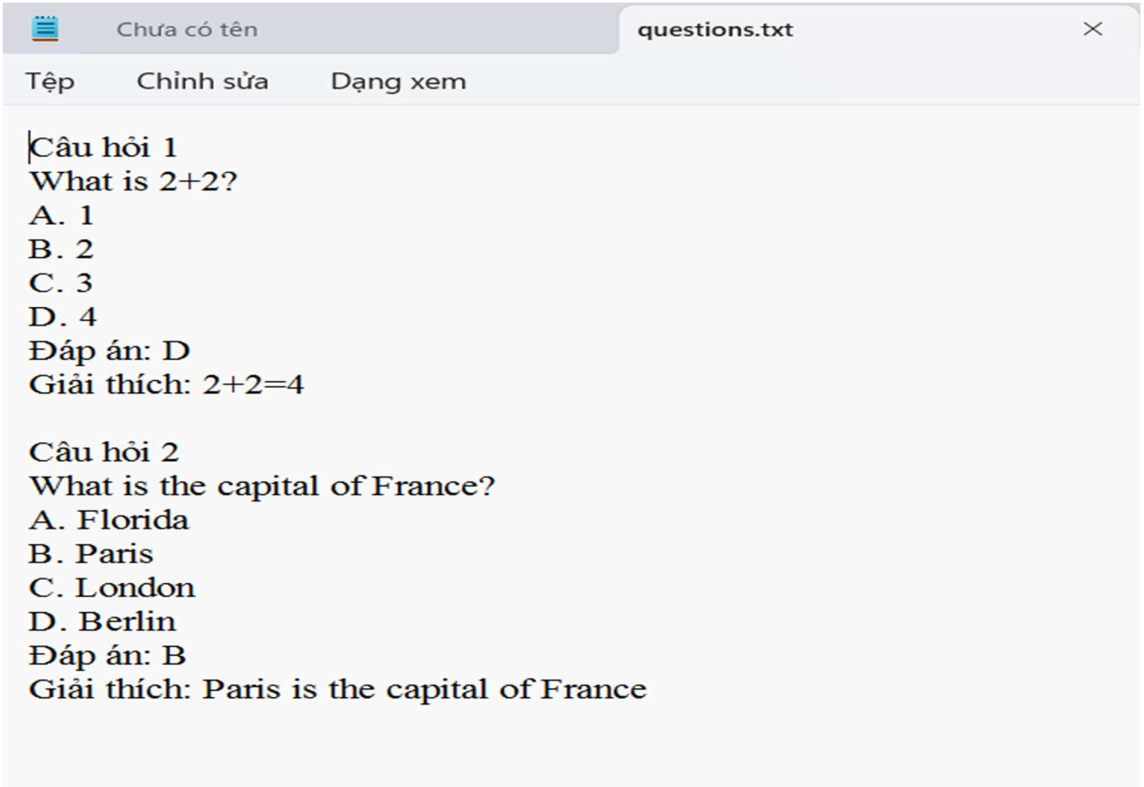
Chương 3 tập trung vào thiết kế và xây dựng trò chơi Trivia Challenge với giao diện GUI bằng tkinter, sử dụng file questions.txt làm nguồn dữ liệu. Hệ thống được tổ chức qua sơ đồ khối với các module File I/O, Logic Game, và GUI, cùng biểu đồ phân cấp chức năng quản lý trò chơi, đọc dữ liệu, xử lý logic, và hiển thị giao diện. Dữ liệu được mô hình hóa thành bảng "Questions" với các trường như QuestionID, QuestionText, và CorrectAnswer, lưu trữ dưới dạng dictionary trong bộ nhớ. Các thuật toán chính gồm File Reading, Question Processing, Answer Validation, và Game Termination, với luồng xử lý tuần tự từ đọc file đến kết thúc trò chơi. Chương trình bao gồm các hàm như open_file, next_block, và các phương thức trong class TriviaGame (__init__, load_next_question, submit_answer, end_game) để thực thi logic và giao diện hiệu quả.

CHƯƠNG 4. THỰC NGHIỆM VÀ KẾT LUẬN

4.1 Thực nghiệm chương trình

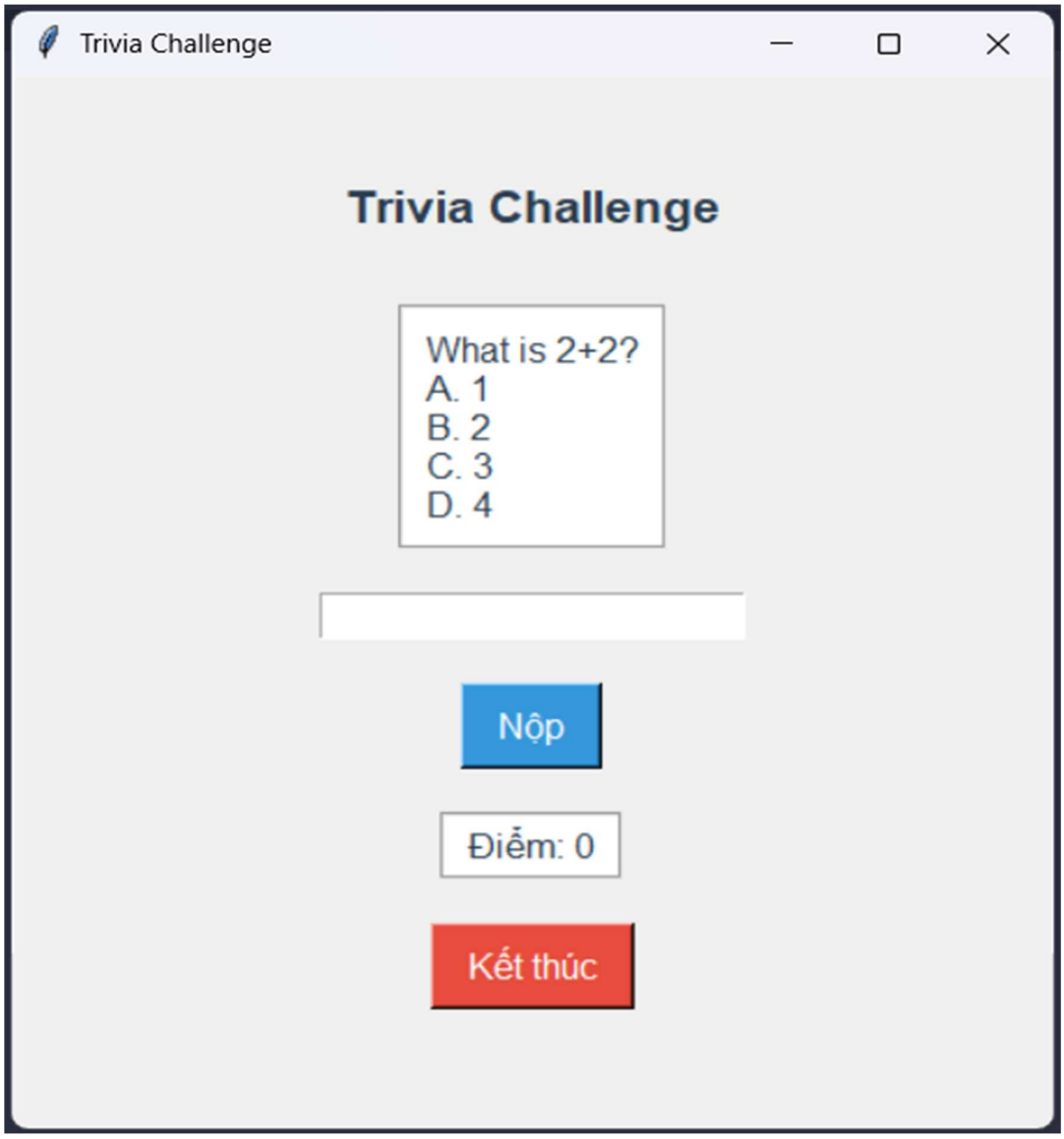


Hình 4.1 Giao diện tổng quan

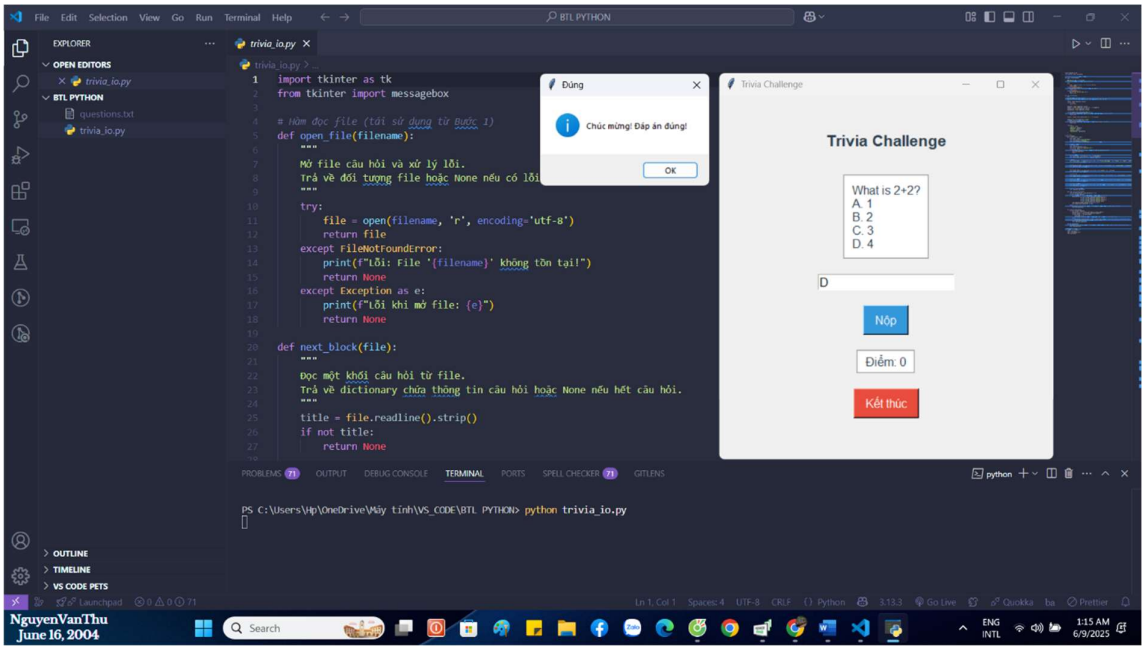


Hình 4.2 Dữ liệu File Questions.txt

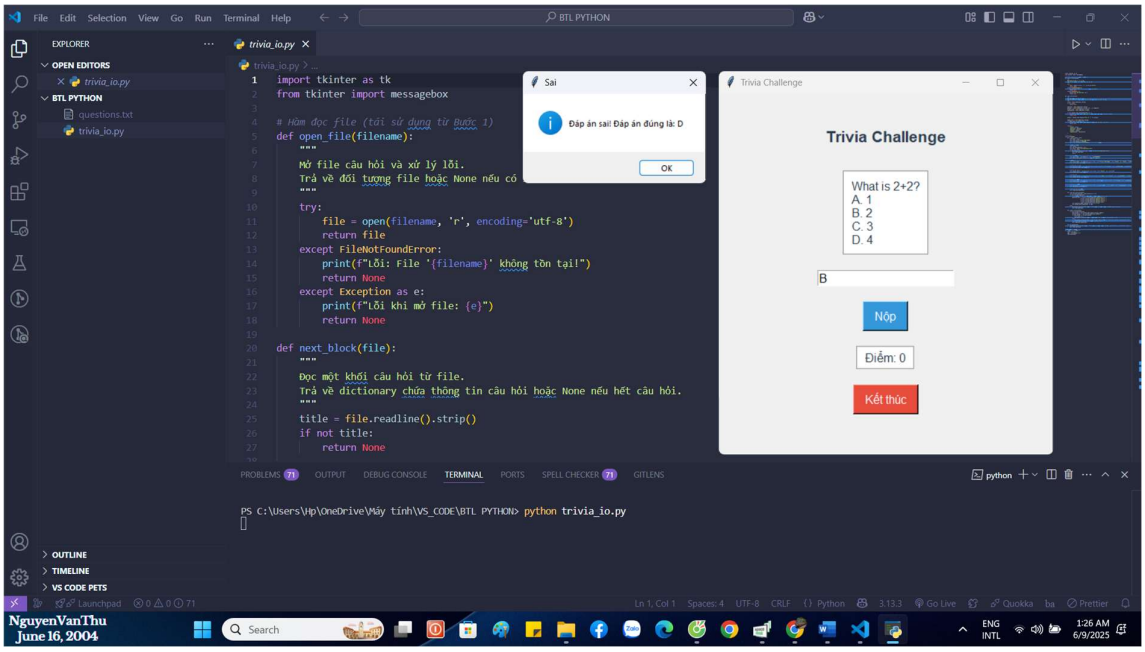
Giao diện tổng quan GUI về Trivia Challenge sau khi xây dựng và được kết quả như hình.



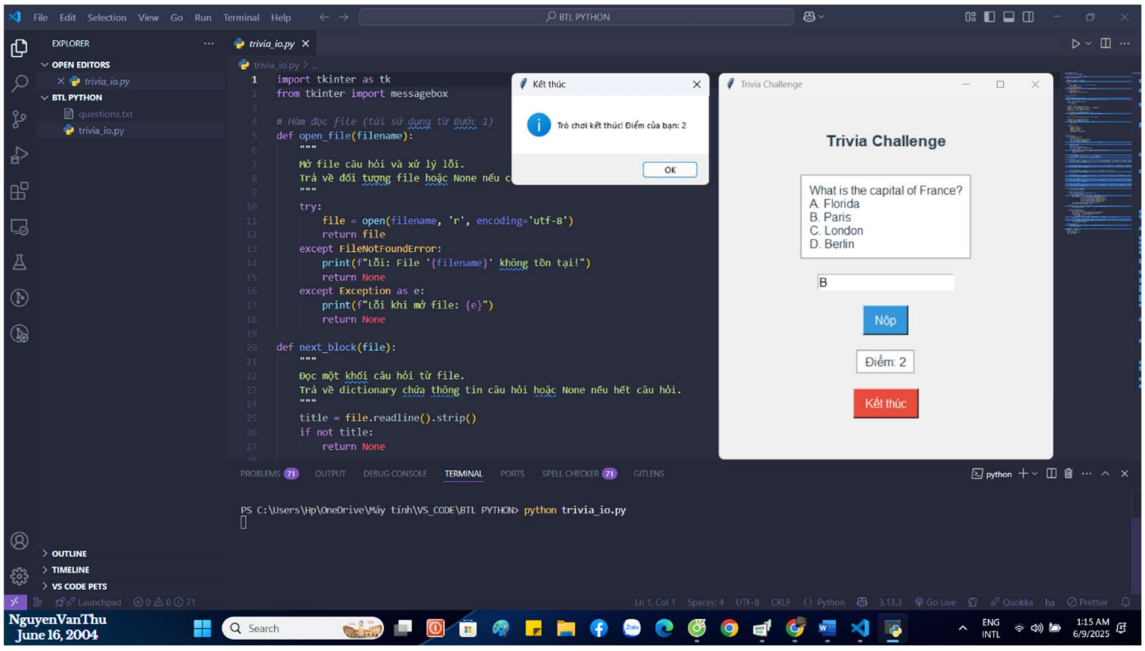
Hình 4.3 Giao diện GUI Trivia Challenge



Hình 4.4 Kết quả test (What is 2+2) và hiển thị thông báo đúng



Hình 4.5 thông báo hiển thị kết quả sai



Hình 4.6 Kết quả hiển thị kết thúc số lượng câu hỏi và tổng điểm

4.2 Kết luận

❖ ***Sản phẩm đã đạt được những gì***

1. Chức năng cơ bản:

- Đọc dữ liệu câu hỏi từ file questions.txt và hiển thị trên giao diện GUI.
- Hiển thị câu hỏi cùng 4 đáp án (A, B, C, D) trong một cột thẳng đứng, với thiết kế đẹp mắt (màu sắc, phông chữ).
- Cho phép người dùng nhập đáp án (A, B, C, hoặc D) và kiểm tra tính đúng/sai.
- Cập nhật điểm số khi trả lời đúng (tăng 1 điểm) và hiển thị thông báo (đúng/sai) qua messagebox.
- Chuyển sang câu hỏi tiếp theo sau mỗi lần nộp đáp án hoặc kết thúc tự động khi hết câu hỏi.
- Kết thúc trò chơi khi nhấn nút "Kết thúc" hoặc hết câu hỏi, hiển thị tổng điểm và đóng giao diện.

2. Thiết kế giao diện:

- Giao diện thân thiện với màu nền xám nhạt, label trắng có viền, button nổi bật (xanh dương cho "Nộp", đỏ cho "Kết thúc"), và tiêu đề "Trivia Challenge".

- Sử dụng phông chữ "Arial" với kích thước phù hợp, đảm bảo dễ đọc.

3. Xử lý lỗi:

- Xử lý lỗi khi file questions.txt không tồn tại hoặc định dạng sai (in thông báo lỗi và đóng chương trình).

❖ ***Kiến thức thu được***

1. Học lập trình Python:

- Rèn luyện kỹ năng viết code với tkinter để tạo giao diện GUI.
- Hiểu cách xử lý file văn bản (open, readline) và quản lý ngoại lệ (try-except).
- Làm quen với cấu trúc dữ liệu (dictionary) và logic lập trình (vòng lặp, điều kiện).

2. Phát triển dự án nhỏ:

- Xây dựng một sản phẩm hoàn chỉnh từ thiết kế đến triển khai, bao gồm phân tích yêu cầu, thiết kế sơ đồ, và lập trình.
- Áp dụng các khái niệm như module, class, và hàm để tổ chức code hiệu quả.

3. Giải trí và giáo dục:

- Tạo một trò chơi đơn giản để tự học hoặc chơi cùng bạn bè, kiểm tra kiến thức qua các câu hỏi.
- Dễ dàng mở rộng bằng cách thêm câu hỏi mới vào file questions.txt.

❖ ***Cải tiến và phát triển***

1. Cải thiện giao diện:

- Thêm hình ảnh hoặc logo (sử dụng tk.PhotoImage) để làm giao diện sinh động hơn.
- Tích hợp hiệu ứng âm thanh (dùng playsound hoặc pygame) khi trả lời đúng/sai.
- Sử dụng grid thay cho pack để sắp xếp widget chính xác hơn (ví dụ: căn giữa câu hỏi và đáp án).

2. Mở rộng chức năng:

- Thêm chế độ chơi nhiều người, lưu điểm số vào file hoặc cơ sở dữ liệu (như SQLite).

- Tích hợp bộ đếm thời gian cho mỗi câu hỏi (sử dụng after trong tkinter).
- Hiển thị giải thích khi trả lời sai để hỗ trợ học tập (hiện tại chỉ hiển thị đáp án đúng).

3. Tối ưu hóa dữ liệu:

- Chuyển từ file questions.txt sang file CSV hoặc cơ sở dữ liệu để quản lý câu hỏi hiệu quả hơn (dễ thêm, xóa, sửa).
- Thêm danh mục câu hỏi (ví dụ: Toán, Lịch sử) để phân loại và chọn ngẫu nhiên.

4. Cải thiện trải nghiệm người dùng:

- Thêm nút "Bỏ qua" để chuyển câu hỏi mà không tính điểm.
- Lưu tiến trình chơi để tiếp tục sau khi đóng chương trình.
- Hỗ trợ đa ngôn ngữ (ví dụ: chuyển đổi giữa tiếng Việt và tiếng Anh).

5. Kiểm thử và bảo trì:

- Thêm kiểm tra đầu vào (ví dụ: chỉ chấp nhận A, B, C, D) để tránh lỗi nhập sai.
- Viết tài liệu sử dụng chi tiết cho người chơi hoặc nhà phát triển khác.

Tóm tắt chương:

Chương 4 đã giúp cho cá nhân em thấy được thực tế hệ thống đang hoạt động tương đối ổn định. Và giúp cho cá nhân em có được những cái nhìn tốt hơn về việc làm của mình và đã tiếp thu được nhưng gì qua bài làm và còn đang thiếu sót ở vị trí nào để tiếp tục có nhưng phương án cải tiến và phát triển tốt về phía sau và giúp ích trong cuộc sống.

TÀI LIỆU THAM KHẢO

[1]. “Automate The Boring Stuff With Python”_San Francisco.

[2]. “Python Cookbook 3rd edition” - David Beazley và Brain K.Jones

[3]. “Think Python" - Allan B. Downey

[4]. Sách “Python cơ bản" - Tác giả: Bùi Việt Hà

[5]. Invent Your Own Computer Game with Python