MÔ TẢ CHI TIẾT VỀ PROJECT

1. Giới thiệu

Dự án "SHERLOCK: A STUDY IN PINK – Phần 2" là một hệ thống game logic có độ phức tạp cao, được xây dựng trên nền tảng lập trình hướng đối tượng và quản lý bộ nhớ động. Dự án mô phỏng quá trình truy đuổi tội phạm qua một mê cung với các nhân vật chính là Sherlock, Watson, tên tội phạm và nhiều đối tượng phụ như robot và vật phẩm hỗ trợ. Tài liệu này trình bày chi tiết các yêu cầu về chức năng của hệ thống, từ việc xây dựng thành phần bản đồ đến các tương tác phức tạp giữa các đối tượng trong game.

2. Phạm vi dự án

Hệ thống được chia thành các module chính sau:

- **Thành phần của bản đồ:** Quản lý các phần tử của bản đồ gồm các loại đường đi (Path), tường thật (Wall) và tường giả (FakeWall) với yêu cầu đặc biệt về phát hiện của Watson dựa trên chỉ số EXP.
- **Bản đồ và vị trí:** Xây dựng bản đồ dưới dạng mảng 2 chiều với các đối tượng được khởi tạo theo vị trí, bao gồm cả lớp Position với các phương thức xử lý định dạng chuỗi và kiểm tra vị trí hợp lệ.
- **Đối tượng di chuyển:** Định nghĩa abstract class MovingObject và các lớp kế thừa cụ thể cho các nhân vật: Sherlock, Watson, Criminal. Mỗi đối tượng có quy tắc di chuyển riêng biệt dựa trên "moving_rule" và các yêu cầu đặc biệt (chẳng hạn điều kiện HP/EXP, khoảng cách Manhattan, …).
- **Mảng đối tượng di chuyển:** Quản lý tập hợp các đối tượng di chuyển với các phương thức thêm mới, kiểm tra đầy và hiển thị thông tin.
- **Cấu hình hệ thống:** Đọc file cấu hình với các thông số về kích thước bản đồ, vị trí, số lượng đối tượng, và các thông số ban đầu cho các nhân vật.
- **Robot và vật phẩm:** Triển khai đa dạng các loại robot (RobotC, RobotS, RobotW, RobotSW) với các quy tắc di chuyển và tính toán khoảng cách riêng, đồng thời tích hợp hệ thống vật phẩm (MagicBook, EnergyDrink, FirstAid, ExcemptionCard, PassingCard) với logic sử dụng vật phẩm thay đổi các thông số của nhân vật.
- **Túi đồ và giao dịch vật phẩm:** Mỗi nhân vật có túi đồ riêng (SherlockBag, WatsonBag) với cơ chế lưu trữ, truy xuất và trao đổi vật phẩm khi gặp nhau.
- **Quy trình game – StudyInPinkProgram:** Điều phối toàn bộ quá trình di chuyển, xử lý va chạm, tạo robot (sau mỗi 3 bước di chuyển của Criminal) và cập nhật trạng thái game cho đến khi đạt điều kiện dừng (HP = 0, bắt được tên tội phạm, …).
- **Kiểm thử:** Cung cấp lớp TestStudyInPink để kiểm tra toàn diện các module và đảm bảo tính chính xác của hệ thống.

3. Yêu cầu chức năng chi tiết

3.1. Thành phần của Bản đồ

- **MapElement:**
  - Thuộc tính: `protected ElementType type` (enum gồm: PATH, WALL, FAKE_WALL).
  - Constructor: Khởi tạo với 1 tham số kiểu ElementType.
  - Virtual destructor và phương thức `getType()`.
- **Các lớp con:**
  - **Path, Wall:** Khởi tạo theo yêu cầu của hệ thống.
  - **FakeWall:** Có thuộc tính `req_exp` được tính bằng công thức: `(r * 257 + c * 139 + 89) % 900 + 1` với r, c là chỉ số hàng, cột.
  - *Lưu ý:* Đối với FakeWall, Sherlock luôn nhận diện được, còn Watson phải có EXP đủ để vượt qua.

3.2. Bản đồ (Class Map)

- Thuộc tính: Số hàng, số cột, và mảng 2 chiều lưu trữ các đối tượng MapElement với tính đa hình.
- Constructor: Tạo bản đồ với các đối tượng Wall, FakeWall dựa trên mảng vị trí nhập vào; các vị trí còn lại là Path.
- Destructor: Thu hồi vùng nhớ cấp phát động.
- Phương thức: `isValid(Position, MovingObject*)` kiểm tra tính hợp lệ của vị trí cho đối tượng di chuyển dựa trên các đặc điểm của đối tượng và thành phần bản đồ.

3.3. Vị trí (Class Position)

- Thuộc tính: `r` và `c` mô tả vị trí theo hàng và cột.
- Constructors:
  - Với tham số (r, c) và với tham số chuỗi theo định dạng "(r,c)".
- Phương thức:
  - Get/Set cho r, c;
  - `str()` trả về định dạng "(r,c)";
  - `isEqual(int, int)` so sánh vị trí;
  - Biến tĩnh `npos` biểu diễn "không có vị trí" (r = −1, c = −1).

3.4. Đối tượng Di chuyển (Abstract Class MovingObject)

- Thuộc tính:
  - `index` – vị trí trong mảng;
  - `pos` – vị trí hiện tại;
  - `map` – con trỏ tới bản đồ;

    - `name` – tên đối tượng.
- Constructor: Khởi tạo với các tham số tương ứng (với name mặc định là rỗng).
- Phương thức:
    - Pure virtual: `getNextPosition()` để tính toán bước đi kế tiếp;
    - `getCurrentPosition()` trả về vị trí hiện tại;
    - Pure virtual: `move()` thực hiện di chuyển;
    - Pure virtual: `str()` trả về chuỗi thông tin định dạng của đối tượng.

3.5. Các Nhân vật Di chuyển

- **Sherlock:**
    - Constructor nhận thêm `moving_rule`, `init_hp`, `init_exp` với các giới hạn: HP [0,500] (nếu vượt 500 sẽ được cài đặt lại) và EXP [0,900].
    - Phương thức `getNextPosition()` sử dụng quy tắc di chuyển theo chuỗi `moving_rule` (lặp lại từ đầu khi hết ký tự).
    - Nếu vị trí tính được không hợp lệ, trả về `npos`.
    - `move()` cập nhật vị trí nếu bước đi hợp lệ;
    - `str()` định dạng chuỗi: `Sherlock[index=<index>;pos=<pos>;moving_rule=<moving_rule>]`.
- **Watson:** Tương tự Sherlock với một số khác biệt trong định dạng hiển thị.
- **Criminal:**
    - Di chuyển đến vị trí có tổng khoảng cách Manhattan từ vị trí của Sherlock và Watson là lớn nhất;
    - Trong trường hợp có nhiều vị trí, ưu tiên theo thứ tự: ‘U’, ‘L’, ‘D’, ‘R’;
    - `str()` định dạng: `Criminal[index=<index>;pos=<pos>]`.

3.6. Mảng Các Đối tượng Di chuyển (ArrayMovingObject)

- Quản lý mảng các đối tượng di chuyển với thuộc tính:
    - `arr_mv_objs` (đảm bảo tính đa hình), `count`, `capacity`.
- Các phương thức:
    - Constructor khởi tạo với `capacity` cho trước;
    - Destructor thu hồi bộ nhớ;
    - `isFull()` kiểm tra mảng đã đầy;
    - `add(MovingObject*)` thêm đối tượng vào mảng nếu có chỗ;
    - `str()` trả về chuỗi định dạng: `ArrayMovingObject[count=<count>;capacity=<capacity>;<MovingObject1>;...]`.

3.7. Cấu hình cho Chương trình (Class Configuration)

- Đọc file cấu hình với các thông số:
  - Số hàng, số cột của bản đồ;
  - Số lượng đối tượng di chuyển tối đa;
  - Danh sách các vị trí của Wall và FakeWall;
  - Các tham số khởi tạo của Sherlock, Watson (moving_rule, vị trí, HP, EXP);
  - Vị trí khởi tạo của Criminal;
  - Số bước di chuyển (NUM_STEPS).
- Constructor xử lý việc đọc file và khởi tạo các thuộc tính;
- `str()` trả về định dạng chuỗi liệt kê đầy đủ thông số theo thứ tự đã quy định.

3.8. Robot và Vật phẩm

- **Robot:**
  - Các loại robot: RobotC, RobotS, RobotW, RobotSW với các đặc thù riêng về thuộc tính (ví dụ: robot có thể đi trên FakeWall, nhưng không đi được trên Wall) và các mối quan hệ (con trỏ đến Criminal, Sherlock, Watson theo loại robot).
  - Phương thức `getNextPosition()` và `move()` được cài đặt với các quy tắc di chuyển chuyên biệt:
    - RobotC: Di chuyển theo vị trí Criminal hiện tại.
    - RobotS, RobotW, RobotSW: Di chuyển theo khoảng cách Manhattan tối ưu với ưu tiên theo thứ tự từ "Up" theo hướng kim đồng hồ (bắt đầu từ hướng lên).
  - `str()` hiển thị thông tin: `Robot[pos=<pos>;type=<robot_type>;dist=<dist>]` (với robot loại C không hiển thị khoảng cách).
- **Vật phẩm (BaseItem và các lớp con):**
  - Các loại vật phẩm: MagicBook, EnergyDrink, FirstAid, ExcemptionCard, PassingCard với các phương thức pure virtual `canUse(...)` và `use(...)`.
  - Mỗi vật phẩm có điều kiện sử dụng riêng (ví dụ: MagicBook chỉ được dùng khi EXP ≤ 350, EnergyDrink chỉ khi HP ≤ 100, …).
  - Quy tắc tạo vật phẩm dựa trên vị trí tạo ra robot với công thức tính "số chủ đạo" từ tích của chỉ số hàng và cột.

3.9. Túi Đồ (BaseBag và các lớp kế thừa)

- **BaseBag:**
  - Cung cấp các phương thức: `insert(BaseItem*)`, `get()`, `get(ItemType)`, và `str()` hiển thị danh sách các vật phẩm theo thứ tự từ đầu đến cuối.
- **SherlockBag & WatsonBag:**

- Mỗi túi có số lượng vật phẩm tối đa khác nhau (13 cho Sherlock, 15 cho Watson);
- Cơ chế trao đổi vật phẩm khi Sherlock và Watson gặp nhau:
  - Sherlock trao PassingCard, Watson trao ExcemptionCard (nếu có nhiều, trao tất cả theo thứ tự từ đầu đến cuối).

*** Chi tiết về các vật phẩm:**

Vật phẩm	Tác dụng	Điều kiện sử dụng
MagicBook	Chứa kiến thức ma thuật cổ đại giúp Sherlock và Watson tăng cường kiến thức, kinh nghiệm và trải nghiệm cho họ một cách nhanh chóng nên dễ dàng hồi phục exp tăng thêm 25% khi sử dụng.	`exp ≤ 350`
EnergyDrink	Nước tăng lực khi được sử dụng sẽ giúp nhân vật hồi phục hp tăng thêm 20% khi sử dụng.	`hp ≤ 100`
FirstAid	Túi đồ cứu thương khi sử dụng sẽ giúp nhân vật hồi phục hp tăng thêm 50% khi sử dụng.	`hp ≤ 100` **hoặc** `exp ≤ 350`
ExcemptionCard	Thẻ miễn trừ ảnh hưởng có tác dụng giúp nhân vật miễn trừ hp, exp khi không vượt qua các thử thách tại một vị trí đi đến.	Chỉ có **Sherlock** sử dụng được thẻ này **và** khi **hp của Sherlock** là **số lẻ**
PassingCard	Khi sử dụng thẻ PassingCard để thực hiện thử thách, nhân vật không cần thực hiện thử thách tại một vị trí đi đến. Thẻ có một thuộc tính là **challenge** (kiểu chuỗi) là tên của một thử thách (ví dụ "RobotS" là thẻ để vượt qua thử thách RobotS). Nếu loại của thẻ là "all", thẻ có thể sử dụng cho bất kỳ loại thử thách nào mà không cần quan tâm thử thách gặp phải. Nếu không, khi sử dụng phương thức `use`, cần kiểm tra loại thẻ có trùng với loại thử thách nhân vật gặp phải hay không. Nếu không trùng, nhân vật sẽ bị trừ **50 EXP** dù tác dụng miễn trừ vẫn được áp dụng.	Chỉ có **Watson** sử dụng được thẻ này **và** khi **hp của Watson** là **số chẵn**

3.10. Quá Trình Game – StudyInPinkProgram

- **Tích hợp các module:**

- Khởi tạo: Đọc file cấu hình, khởi tạo bản đồ, các đối tượng di chuyển (criminal, Sherlock, Watson) và thêm vào mảng ArrayMovingObject.

- **Vòng lặp di chuyển:**
- **Nếu Sherlock gặp**:

  • RobotS: Sherlock cần giải quyết một bài toán để có thể chiến thắng RobotS. Nếu EXP của Sherlock lúc này **lớn hơn 400**, Sherlock sẽ giải quyết được và nhận về vật phẩm mà robot này nắm giữ. Nếu không, EXP của Sherlock sẽ mất đi 10%.

  • RobotW: Sherlock sẽ vượt qua và nhận vật phẩm mà không cần phải chiến đấu.

  • RobotSW: Sherlock chỉ có thể thắng RobotSW khi **EXP** của Sherlock **lớn hơn 300** và **HP** của Sherlock **lớn hơn 335**. Nếu chiến thắng, Sherlock nhận được vật phẩm mà robot này nắm giữ. Nếu không, Sherlock sẽ bị mất 15% HP và 15% EXP.

  • RobotC: Sherlock gặp RobotC tức là đã gặp được vị trí liền kề tên tội phạm. Lúc này, nếu **EXP** của Sherlock lớn hơn **500**, Sherlock sẽ chiến thắng tên robot, bắt được tên tội phạm (không nhận vật phẩm mà robot này nắm giữ). Ngược lại, Sherlock sẽ để tội phạm chạy thoát. Dù vậy vẫn sẽ tiêu diệt được robot và nhận về vật phẩm của robot này nắm giữ.

  • FakeWall: Như mô tả phía trên

- Nếu Watson gặp:
  • RobotS: Watson sẽ không thực hiện hành động gì với robot và cũng không nhận được vật phẩm của robot này nắm giữ.

  • RobotW: Watson cần đối đầu với Robot này và chỉ chiến thắng khi có HP **lớn hơn 350**. Nếu chiến thắng thì Watson sẽ nhận được vật phẩm mà robot nắm giữ, nếu thua thì **HP** của Watson bị giảm **5%**.

  • RobotSW: Watson chỉ có thể thắng RobotSW khi **EXP** của Watson **lớn hơn 600** và **HP** của Watson lớn hơn **165**. Nếu chiến thắng, Watson nhận được vật phẩm mà robot này nắm giữ. Nếu không, Watson sẽ bị mất **15% HP** và **15% EXP**.

  • RobotC: Watson gặp RobotC tức là đã gặp được vị trí liền kề tên tội phạm. Watson chưa thể bắt được tên tội phạm vì bị giữ chân bởi RobotC. Tuy vậy, Watson vẫn sẽ tiêu diệt được robot và nhận về vật phẩm của robot này nắm giữ.

  • FakeWall: Như mô tả phía trên Trước và sau mỗi sự kiện đụng độ, nhân vật sẽ kiểm tra và sử dụng các vật dụng có thể trong túi của mình

- **Điều kiện dừng:**

    - Khi HP của Sherlock hoặc Watson giảm về 0 hoặc khi tên tội phạm bị bắt theo quy định.
- **Phương thức hiển thị:**
    - `printStep`, `printResult` in ra thông tin chi tiết về trạng thái hệ thống tại mỗi bước.

4. Yêu cầu khác

- **Môi trường phát triển:**
    - Phải đảm bảo chương trình biên dịch và chạy trên nền tảng Unix sử dụng g++ với tùy chọn `-std=c++11` và chỉ được sửa đổi 2 file:
        - `study_in_pink2.h`
        - `study_in_pink2.cpp`
- **Quy định về include:**
    - Trong file `study_in_pink2.h`, chỉ có 1 lệnh `#include "main.h"`.
    - Trong file `study_in_pink2.cpp`, chỉ có 1 lệnh `#include "study_in_pink2.h"`.