**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**VIỆN KHOA HỌC KỸ THUẬT BƯU ĐIỆN**

**-----□□-----**



**BÁO CÁO**

**ĐỀ TÀI THI CUỐI KỲ**

***Đề tài :Dự án sản phẩm game***

**Giảng viên: PHAN LÝ HUỲNH**

**Tên nhóm: LITTLE STORK**

**Lớp :** **D23CQCC01- B**

**Sinh viên thực hiện: 1. NGUYỄN NGỌC DUY - B23DCCC049**

**2. NGUYỄN TIẾN TUẤN - B23DCCC173**

**3. NGUYỄN THỊ HÀ - B23DCCC057**

**4. TRẦN TUẤN MINH - B23DCCC113**

**5. HOÀNG THỊ HÒA - B23DCCC069**

**Hà Nội, ngày 01/01/2024**

**Mục lục**

**Chương 1: Giới thiệu bài toán......................................................**

**1.1 Nguồn gốc trò chơi...................................................................**

**1.2 Mô tả trò chơi.........................................................................**

**Chương 2: Cơ sở lý thuyết............................................................**

**2.1 Đặt vấn đề của bài toán.............................................................**

**Chương 3 :Giải thuật....................................................................**

**3.1 Phương án giải quyết ...............................................................**

**Chương 4: Phân tích và thiết kế..................................................**

**4.1 Phân tích bài toán theo tư duy lập trình........................................**

**4.2 Ví dụ minh họa........................................................................**

**4.3 Giao diện................................................................................**

**Chương 5: Ứng dụng....................................................................**

**Chương 6: Kết quả chạy chương trình.......................................**

**Chương 7: Kết luận......................................................................**

**Chương 8: Tài liệu tham khảo.....................................................**

**BẢNG PHÂN CÔNG CÔNG VIỆC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MSV** | **HỌ VÀ TÊN** | **CÔNG VIỆC THỰC HIỆN** | **MỨC ĐỘ HOÀN THÀNH** | **NHẬN XÉT** |
| **B23DCCC049** | Nguyễn Ngọc Duy |  | 100 | - Hoàn thành tốt tham gia họp đầy đủ |
| **B23DCCC173** | Nguyễn Tiến Tuấn |  | 100 | - Hoàn thành tốt tham gia họp đầy đủ |
| **B23DCCC057** | Nguyễn Thị Hà |  | 100 | - Hoàn thành tốt tham gia họp đầy đủ |
| **B23DCCC069** | Hoàng Thị  Hòa |  | 100 | - Hoàn thành tốt tham gia họp đầy đủ |
| **B23DCCC113** | Trần Tuấn Minh |  | 100 | - Hoàn thành tốt tham gia họp đầy đủ |

**CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN**

**1.1 NGUỒN GỐC TRÒ CHƠI**

2

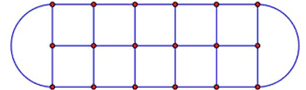
Ô ăn quan, hay còn gọi tắt là ăn quan hoặc ô quan là một trò chơi dân gian của trẻ em Việt Nam mà chủ yếu là các bé gái. Đây là trò chơi có tính chất chiến thuật thường dành cho hai hoặc ba người chơi và có thể sử dụng các vật liệu đa dạng, dễ kiếm để chuẩn bị cho trò chơi.

Hiện chưa rõ nguồn gốc cũng như thời điểm bắt đầu nhưng chắc chắn rằng Ô ăn quan đã có ở Việt Nam từ rất lâu đời, có thể nó được lấy cảm hứng từ những cánh đồng lúa nước ở nơi đây. Những câu truyện lưu truyền về Mạc Hiển Tích (chưa rõ năm sinh, năm mất), đỗ Trạng nguyên năm 1086 nói rằng ông đã có một tác phẩm bàn về các phép tính trong trò chơi Ô ăn quan và đề cập đến số ẩn (số âm) của ô trống xuất hiện trong khi chơi. Ô ăn quan đã từng phổ biến ở khắp ba miền Bắc, Trung, Nam của Việt Nam nhưng những năm gần đây chỉ còn được rất ít trẻ em chơi. Bảo tàng Dân tộc học Việt Nam có trưng bày, giới thiệu và hướng dẫn trò chơi này.

Theo các nhà nghiên cứu, ô ăn quan thuộc họ trò chơi mancala, tiếng Ả Rập là manqala hoặc minqala (khi phát âm, trọng âm rơi vào âm tiết đầu ở Syria và âm tiết thứ hai ở (Ai Cập) có nguồn gốc từ động từ naqala có nghĩa là di chuyển. Bàn chơi mancala đã hiện diện ở Ai Cập từ thời kỳ Đế chế (khoảng 1580 - 1150 TCN). Tuy nhiên còn một khoảng trống giữa lần xuất hiện này với sự tồn tại của mancala ở Ceylon(Srilanka) những năm đầu Công nguyên và ở Ả Rập trước thời Muhammad.(2)

**1.2. MÔ TẢ TRÒ CHƠI**

**Chuẩn bị:**

****

*Bàn chơi*: bàn chơi Ô ăn quan kẻ trên một mặt bằng tương đối phẳng có kích thước linh hoạt miễn là có thể chia ra đủ số ô cần thiết để chứa quân đồng thời không quá lớn để thuận tiện cho việc di chuyển quân, vì thế có thể được tạo ra trên nền đất, vỉa hè, trên miếng gỗ phẳng .... Bàn chơi được kẻ thành một hình chữ nhật rồi chia hình chữ nhật đó thành mười ô vuông, mỗi bên có năm ô đối xứng nhau. Ở hai cạnh ngắn hơn của hình chữ nhật, kẻ hai ô hình bán nguyệt hoặc hình vòng cung hướng ra phía ngoài. Các ô hình vuông gọi là ô dâcòn hai ô hình bán nguyệt hoặc vòng cung gọi là ô quan.



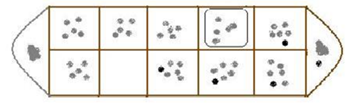
*Quân chơi*: gồm hai loại quan và dân, được làm hoặc thu thập từ nhiều chất liệu có hình thể ổn định, kích thước vừa phải để người chơi có thể cầm, nắm nhiều quân bằng một bàn tay khi chơi. Quan có kích thước lớn hơn dân đáng kể cho dễ phân biệt với nhau. Quân chơi có thể là những viên sỏi, gạch, đá, hạt của một số loại quả... hoặc được sản xuất công nghiệp từ vật liệu cứng mà phổ biến là nhựa. Số lượng quan luôn là 2 còn dân có số lượng tùy theo luật chơi nhưng phổ biến nhất là 50.

Bố trí quân chơi: quan được đặt trong hai ô hình bán nguyệt hoặc cánh cung, mỗi ô một quân, dân được bố trí vào các ô vuông với số quân đều nhau, mỗi ô 5 dân. Trường hợp không muốn hoặc không thể tìm kiếm được quan phù hợp thì có thể thay quan bằng cách đặt số lượng dân quy đổi vào ô quan. Người chơi: thường gồm hai người chơi, mỗi người ở phía ngoài cạnh dài hơn của hình chữ nhật và những ô vuông bên nào thuộc quyền kiểm soát của người chơi bên đó.

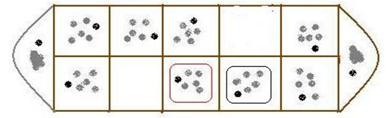
*Luật chơi*: Người thắng cuộc trong trò chơi này là người mà khi cuộc chơi kết thúc có tổng số dân nhiều hơn. Tùy theo luật chơi từng địa phương hoặc thỏa thuận giữa hai người chơi nhưng phổ biến là 1 quan được quy đổi bằng 10 dân hoặc 5 dân.

*Di chuyển quân*: từng người chơi khi đến lượt của mình sẽ di chuyển dân theo phương án để có thể ăn được càng nhiều dân và quan hơn đối phương càng tốt. Người thực hiện lượt đi đầu tiên thường được xác định bằng cách oẳn tù tì hay thỏa thuận.

Khi đến lượt, người chơi sẽ dùng tất cả số quân trong một ô có quân bất kỳ do người đó chọn trong số 5 ô vuông thuộc quyền kiểm soát của mình để lần lượt rải vào các ô, mỗi ô 1 quân, bắt đầu từ ô gần nhất và có thể rải ngược hay xuôi chiều kim đồng hồ tùy ý. Khi rải hết quân cuối cùng, tùy tình huống mà người chơi sẽ phải xử lý tiếp như sau:



Nếu liền sau đó là một ô vuông có chứa quân thì tiếp tục dùng tất cả số quân đó để rải tiếp theo chiều đã chọn.



Nếu liền sau đó là một ô trống (không phân biệt ô quan hay ô dân) rồi đến một ô có chứa quân thì người chơi sẽ được ăn tất cả số quân trong ô đó. Số quân bị ăn sẽ được loại ra khỏi bàn chơi để người chơi tính điểm khi kết thúc.

Nếu liền sau ô có quân đã bị ăn lại là một ô trống rồi đến một ô có quân nữa thì người chơi có quyền ăn tiếp cả quân ở ô này... Do đó trong cuộc chơi có thể có phương án rải quân làm cho người chơi ăn hết toàn bộ số quân trên bàn chơi chỉ trong một lượt đi của mình. Một ô có nhiều dân thường được trẻ em gọi là ô nhà giàu, rất nhiều dân thì gọi là giàu sụ. Người chơi có thể bằng kinh nghiệm hoặc tính toán phương án nhằm nuôi ô nhà giàu rồi mới ăn để được nhiều điểm.

Nếu liền sau đó là ô quan có chứa quân hoặc 2 ô trống trở lên hoặc sau khi vừa ăn thì người chơi bị mất lượt và quyền đi tiếp thuộc về đối phương. Trường hợp đến lượt đi nhưng cả 5 ô vuông thuộc quyền kiểm soát của người chơi đều không có dân thì người đó sẽ phải dùng 5 dân đã ăn được của mình để đặt vào mỗi ô 1 dân để có thể thực hiện việc di chuyển quân.

Nếu người chơi không đủ 5 dân thì phải vay của đối phương và trả lại khi tính điểm.

**Cuộc chơi sẽ kết thúc** khi toàn bộ dân và quan ở hai ô quan đã bị ăn hết. Trường hợp hai ô quan đã bị ăn hết nhưng vẫn còn dân thì quân trong những hình vuông phía bên nào coi như thuộc về người chơi bên ấy; tình huống này được gọi là hết quan, tàn dân, thu quân, kéo về hay hết quan, tàn dân, thu quân, bán ruộng. Ô quan có ít dân (có số dân nhỏ hơn 5 phổ biến được coi là ít) gọi là quan non và để cuộc chơi không bị kết thúc sớm cho tăng phần thú vị, luật chơi có thể quy định không được ăn quan non, nếu rơi vào tình huống đó sẽ bị mất lượt.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

**A. Đặt vấn đề:**

Trong cờ Ô ăn quan, mỗi cách bố trí quân cờ trên bàn cờ là một vị trí. Vị trí ban đầu là sự sắp xếp các quân cờ lúc đầu cuộc chơi. Mỗi bước đi hợp lệ là một phép biến đổi vị trí, nó biến đổi vị trí trên bàn cờ thành một vị trí khác.

Như vậy, ta xác định được các yếu tố:

- Trạng thái bắt đầu (TTBĐ): là trạng thái nguyên vẹn ban đầu của bàn cờ.

- Trạng thái kết thúc (TTKT): Trạng thái khi ván cờ kết thúc khi các trạng thái đã được biến đổi, thay đổi nhưng được di chuyển xung quanh bàn cờ tạo thành 1 không gian bao gồm các trạng thái

Việc cần thực hiện là tìm kiếm cách để quy các trạng thái thành 1 giới hạn nhất định trong từng vị trí ô bàn cờ đi từ đầu đến trạng thái đích. Khi ta cần biết đây là trạng thái nào, nó đã được cố định tại vị trí ô đó.

Như chúng ta đã biết, trò chơi Ô ăn quan cũng là một trò chơi đối kháng. Cụ thể trò chơi này thuộc dạng trò chơi có tổng ban đầu bằng 0, gồm có 2 người chơi. Người chơi phải cần tính toán để có thể ăn được nhiều quân hay quan nhất.

Không thể có trường hợp cả hai bên đều thắng hoặc đều thua. Nếu một bên thắng thì bên kia nhất định thua và ngược lại. Đặc điểm của các trò chơi thể loại này:

- Có hai đấu thủ, mỗi người chỉ được đi một nước khi tới lượt.

- Các đấu thủ đều biết mọi thông tin về tình trạng trận đấu.

- Trận đấu không kéo dài vô tận, phải diễn ra hòa, hoặc một bên thắng và bên kia thua.

Nếu viết chương trình để người chơi với người thì cần xây dựng các tập luật chơi để người chơi không phạm quy và kết thúc khi có người thắng.

**Chương 3: Giải thuật**

**A. Thuật toán**

* 1. **- Phương án giải quyết**

**3.1.1 Ý tưởng:**

- Xây dựng các thành hàm phần: Xét nước đi hợp lệ hay không, xét đến lượt của ai đi, xét thắng thua, hàm di chuyển, xét trạng thái kết thúc game, thay đổi các hiển thị quân trong ô cờ,....

- Ta dùng hàm **\_\_init\_\_** để khởi tạo ra một trạng thái ban đầu của trò chơi, bao gồm bảng, người chơi hiện tại, và cách tính điểm của hai người chơi.

- Ta tiếp tục dùng hàm **play\_ game** để chơi game và dùng để lặp lại các bước cho đến khi trò chơi kết thúc.

- Quan trọng nhất là sử dụng hàm **phanphoi** để thực hiện phân phối đá về phía bên phải hay bên trái dựa trên lựa chọn chiều của người chơi.

- Sau đó ta kiểm tra điều kiện kết thúc và cập nhập trạng thái của game, dùng hàm **raithem** để kiểm tra điều kiện kết thúc trò chơi và cập nhâp điểm cho người chơi.

-Cuối cùng là lặp lại quá trình và chuyển đổi lượt chơi giữa hai người chơi.

**3.1.2 Áp dụng cơ sở lý thuyết**

- Như đã trình bày ở chương 2, ta sẽ xây dựng cây tìm kiếm, với mỗi nút là một trạng thái có thể có của bàn cờ. Nút gốc biểu diễn cho trạng thái bắt đầu của cuộc chơi. Mỗi nút lá biểu diễn cho một trạng thái kết thúc của trò chơi (trạng thái thắng, thua hoặc hòa). Nếu trạng thái x được biểu diễn bởi nút n thì các con của n biểu diễn cho tất cả các trạng thái kết quả của các nước đi có thể xuất phát từ trạng thái x.

- Như chúng ta đã biết, không phải lúc nào cũng đến được các nút lá mang trạng thái kết thúc, vì độ sâu của cây trò chơi là rất lớn. Vì vậy, chúng ta chỉ thực hiện tìm kiếm đến độ sâu cố định nào đó. Độ sâu càng lớn, hàm tìm kiếm càng gần giá trị tối ưu, cũng có nghĩa là “trình độ suy nghĩ” của máy càng cao!. Tuy nhiên sẽ xảy ra tình trạng tốn quá nhiều bộ nhớ cũng như là thời gian tìm kiếm.Vì vậy, chúng ta chỉ thực hiện tìm kiếm đến độ sâu cố định nào đó.

**3.1.3 Phát biểu bài toán một cách hình thức:**

- Không gian trạng thái: Toàn bộ trạng thái có thể có của bàn cờ.

- Trạng thái đầu: Bàn cờ có 10 ô quân và 2 ô quan, mỗi ô quân có 5 quân cờ, ô quan có 1 quan và không có quân (dân) nào rải vào.

- Trạng thái cuối: Có 2 trường hợp là:

+ Khi 2 quan đồng thời bị ăn hết không còn dân nào trong ô quan.

+ Khi các ô thuộc một bên nắm giữ hết sỏi mà trong kho của mình không đủ 5 sỏi để rải đều cho 5 ô của mình.

- Toán tử: Cách bốc ô quân và chiều rải quân trong bàn cờ một cách hợp lý

- Vấn đề tìm kiếm trong KGTT: Tìm kiếm trạng thái sao cho có lợi nhất và ngắn nhất.

**3.2 - Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán**

**3.2.1 Biểu diễn trạng thái:**

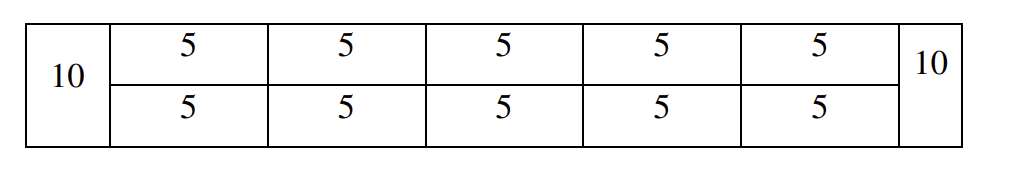
- Mỗi cách bố trí quân cờ trên bàn cờ là một trạng thái. Sử dụng mảng self.board để tạo ra một mảng (list) có 12 phần tử, mỗi phần tử có giá trị mặc định là 5. Mảng này đại diện cho bảng Ô Ăn Quan, trong đó có 12 ô cụ thể là:

* Các ô từ 0 đến 4 là phần của người chơi 1.
* Ô số 5 lỗ để đặt đá.
* Các ô từ 6 đến 10 là phần của người chơi thứ 2.
* Ô số 11 là lỗ để đặt đá.

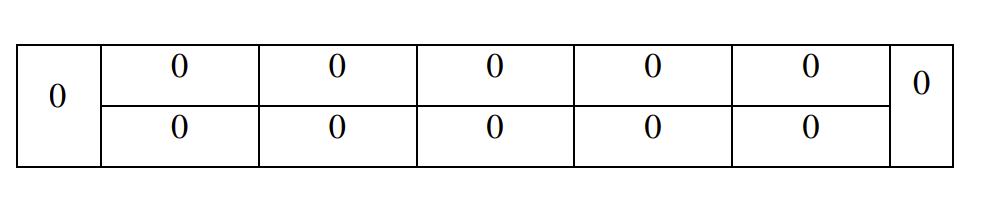
- Mỗi bước đi hợp lệ là 1 toán tử chuyển trạng thái.

**3.2.1 Không gian trạng thái:**

-*Trạng thái đầu*: Gồm 1 bàn cờ. Trong bàn cờ gồm có 2 quan và mỗi ô dân thì có 5 quân.



-*Trạng thái kết thúc*: Khi 2 quan đồng thời bị ăn hết không còn dân nào trong ô quan.



***Nói cách khác***, không còn đối thủ nào có thể thực hiện được nước đi hợp lệ. Các toán tử: chọn ô để rải theo chiều trái hay phải (-1 hay 1).

**3.2.2 Cấu trúc dữ liệu**

* ***Danh sách biến cần dùng:***
* **Self.current\_player** :Biến này đánh dấu người chơi hiện tại.
* **Self .scored\_01**: Biến này để lưu trữ điểm số người chơi thứ 1.
* **Self .scored\_02** :Biến này lưu trữ điểm số người chơi thứ 2.
* **Self.board** : Biến này đại diện cho bảng của trò chơi Ô Ăn Quan, và nó được sử dụng để in ra giá trị của các ô trên bảng.
* **Stones**: Biến này lưu trữ số hạt đá được lấy từ ô của người chơi hiện tại để sau đó phân phối ra các ô tiếp theo.
* **i**: Biến này được sử dụng để duyệt qua các ô trên bàn cờ khi thực hiện phân phối hạt đá.
* **chieu** : Là một biến được sử dụng để xác định hướng phân phối của hạt đá.
* **diem\_cong**: Đây là biến được sử dụng để tính điểm và hiển thị thông tin về số điểm nhận được sau khi ăn quân.
* **new\_stones**: Đây là biến được sử dụng để lưu trữ số quân khi 5 ô của người chơi đó hết sẽ lấy 5 quân ( số quân tích lũy >= 5) hoặc lấy số quân tích lũy( < 5)
* ***Danh sách các hàm:***
* **init:** Đây là hàm khởi tạo
* **print\_board:** dùng để hiển thị trạng thái hiện tại của bảng ô ăn quan.
* **play\_game:** dùng để quản lý quá trình chơi của trò chơi ô ăn quan.
* **def phanphoi(self):** hàm thực hiện quá trình phân phối đá từ một ô đã chọn ra các ô tiếp theo theo chiều trái hoặc phải.
* **def raithem:** hàm thực hiện việc rải đá khi có người chơi hoặc cả hai người chơi đã hết đá trong các lỗ của mình.

**B. Giao Diện**

**Chương 4: Phân tích thuật toán**

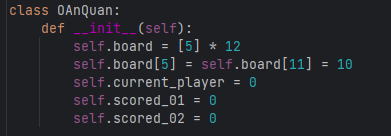
* + 1. **TẠO CLASS CÓ TÊN LÀ Ô ĂN QUAN VÀ HÀM CONTRUCTOR**

Trong hàm contructor, self tham chiếu đến biến tên của toàn bộ **class oanquan**.

- Ban đầu khởi tạo được bàn chơi có 12 ô và mỗi ô có 5 viên đá.

- Sau đó gán giá trị cho 2 ô quan là ô thứ 11 và ô thứ 5 là 10 viên đá.

- Khởi tạo người chơi ban đầu (người chơi thứ 1) = 0 và điểm số mỗi người chơi ban đầu là 0.



**4.1.2. TẠO BÀN CỜ:**

* Dùng **print(“** **”)** để in ra bàn cờ theo định dạng.
* Dùng cấu trúc for để k chạy trên các ô trong khoảng cố định từ (0,5) theo thứ tự tăng dần và (10,5,-1) theo thứ tự giảm dần.
* In giá trị của **self.board[k]** với khoảng trống để tạo ra khoảng cách cho mỗi ô trên bàn cờ.
* In ra một dòng trống để tạo ra khoảng cách giữa các lần in bàn cờ.

**INPUT:**

****

**OUTPUT:**



**4.1.3. ĐIỀU KIỆN CHƠI GAME**

* **while True** là vòng lặp vô hạn, chức năng của nó là lặp không ngừng cho đến khi gặp lệnh **break, continue,....**
* Đặt người chơi thứ 1 là 0 và người chơi thứ 2 là 1 nếu người chơi tiếp theo bằng 2 thì ta sẽ gán cho người chơi đó trở về người chơi thứ 1.
* Câu lệnh **(self.print\_board)** có ý nghĩa là in bàn cờ ra màn hình.
* Trường hợp trò chơi kết thúc hay vòng lặp kết thúc khi và chỉ khi 2 ô quan không còn đá. Khi đó ta sẽ thu đá của mình và so sánh số đá để phân loại thắng thua hòa.
* Với **self.board[0:5]** là đá từ ô 0 đến 4 và **sum(self.board[0:5])** là tổng cộng số đá trong 5 ô.
* if self.current\_player == 2:  
   self.current\_player = 0  
  self.print\_board()  
  if self.board[5] == 0 and self.board[11] == 0:  
    
   self.scored\_01 += sum(self.board[0:5])  
   self.scored\_02 += sum(self.board[6:11])  
    
   if self.scored\_01 < self.scored\_02:  
   print("Player 2: Win!")  
   elif self.scored\_02 < self.scored\_01:  
   print("Player 1: Win!")  
   else:  
   print("Hòa")  
   break

**4.1.4. CHỌN Ô**

* Người chơi thứ 1 được chọn ô bất kì từ 0 đến 4 tương ứng với ô từ 1 đến 5 trên bàn cờ.
* Người chơi thứ 2 được chọn ô bất kì từ 6 đến 10 tương ứng với ô từ 7 đến ô thứ 11 trên bàn cờ ( hay chính là ô từ 1 đến 5 dưới góc nhìn của người thứ 2).
* Trong hình ảnh bên dưới là người chơi thứ 2:

+ Thực hiện chọn ô.

+ Kiểm tra điều kiện: Nếu chọn vào lỗ thứ 5 hoặc ô thứ 11 hoặc ô được chọn không có đá thì sẽ thực hiện chọn lại ô.

+ Nếu là người chơi thứ 1 thì thực hiện tương tự.

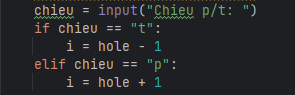
* **Biến toàn cục (global stones,i):** cho biết rằng trong khối này giá trị stones và i sẽ được coi là biến toàn cục nó có thể thay đổi giá trị trong toàn bộ mã.
* Gán stones = số đá trong ô đã chọn và trả về ô chọn giá trị bằng 0.
* if self.current\_player == 1:  
   print("Player 2")  
   hole = int(input("Chọn lỗ (6-11) <=> (1-5): "))  
   if hole == 5 or hole == 11 or self.board[hole] == 0:  
   print(" lựa chọn không hợp lệ. vui lòng chọn lại !")  
   continue  
  global stones, i  
  stones = self.board[hole]  
  self.board[hole] = 0

**4.1.5. CHỌN CHIỀU**

- Nhập từ bàn phím giá trị trái hoặc phải.

+ Nếu chiều là “ t ”(trái) thì giá trị i đc gán bằng tọa độ ô chọn trừ đi 1 ô.

+ Nếu chiều là “ p ”(phải) thì giá trị i đc gán bằng tọa độ ô chọn cộng thêm 1 ô.



**4.1.6. THỰC HIỆN RẢI ĐÁ**

|  |  |  |
| --- | --- | --- |
|  | **CHIỀU BÊN TRÁI** | **CHIỀU BÊN PHẢI** |
| SỐ ĐÁ SAU KHI BỐC Ô | + Sử dụng vòng lặp while để thực hiện vòng lặp rải đá cho đến khi số đá = 0.  + i chạy từ 0 đến 11 vì có 12 ô đá nên nếu trường hợp số đá lớn hơn 12 thì ô nhận thứ 12 sẽ báo lỗi và mỗi lần lặp biến i sẽ trừ đi 1 giá trị nên sẽ âm nên khi i < 0 thì ta sẽ gán i = 11.  + Giá trị i thay đổi, di chuyển sang bên trái sau mỗi lần rải đá và từng ô tương ứng cộng đá.  Input : chọn hole = 0.    Output: | + Sử dụng vòng lặp while để thực hiện vòng lặp rải đá cho đến khi số đá = 0.  + i chạy từ 0 đến 11 vì có 12 ô đá nhưng nếu trường hợp số đá lớn hơn 12 thì ô nhận quân sẽ lỗi và mỗi lần lặp biến i sẽ cộng thêm 1 giá trị nên sẽ vượt quá 12 ô nên khi i > 11 thì ta sẽ gán i = 0.  + Giá trị i thay đổi, di chuyển sang bên phải sau mỗi lần rải đá và từng ô tương ứng cộng đá.  Input: chọn hole = 1.    Output: |

**4.7. KHI CẢ 5 Ô ĐỀU HẾT ĐÁ MÀ TRÒ CHƠI CHƯA KẾT THÚC**

+ Nếu có số đá > 5 thì sẽ rải 5 viên đá vào 5 ô với mỗi ô 1 viên và số điểm sẽ trừ đi 5 sau đó tiếp tục quay lại thực hiện chọn ô để rải đá.

+ Nếu 0 < số đá < 5 thì có bao nhiêu đá sẽ rải vào từng ô theo thứ tự từ ô 0-4 hoặc từ 6-10 và số điểm sẽ trở về 0 sau đó tiếp tục quay lại thực hiện chọn ô để rải đá.

+ Code dưới là người chơi thứ 1 thì ta tính tống đá trong khoảng từ (0,5) và cho các ô trong khoảng đó cộng bằng 1, thực hiện bằng điểm số của người 1.

- Nếu là người chơi thứ 2 thì tính tổng đá trong khoảng (6,11) và cho các ô trong khoảng đó cộng bằng 1, thực hiện bằng điểm số của người 2

if self.current\_player == 0 or self.current\_player == 2:  
 if sum(self.board[0:5]) == 0:  
 if self.scored\_01 == 0 or self.scored\_02 == 0:  
 if self.scored\_01 < self.scored\_02:  
 print("Player 2: Win!")  
 elif self.scored\_02 < self.scored\_01:  
 print("Player 1: Win!")  
 else:  
 print("Hòa")  
 if 0 < self.scored\_01 < 5:  
 new\_stones = self.scored\_01  
 self.scored\_01 = 0  
 for z in range(0, new\_stones):  
 self.board[z] += 1  
 if self.scored\_01 > 5:  
 self.scored\_01 -= 5  
 self.board[0] = self.board[1] = self.board[2] = self.board[3] = self.board[4] = 1

**4.7.2: Điều kiện kết thúc trò chơi**

1. Khi đá ở 2 ô quan đều bằng 0
2. Khi 1 người không còn đá trong 5 ô và tổng số đá tích lũy trong cả bàn chơi bằng 0

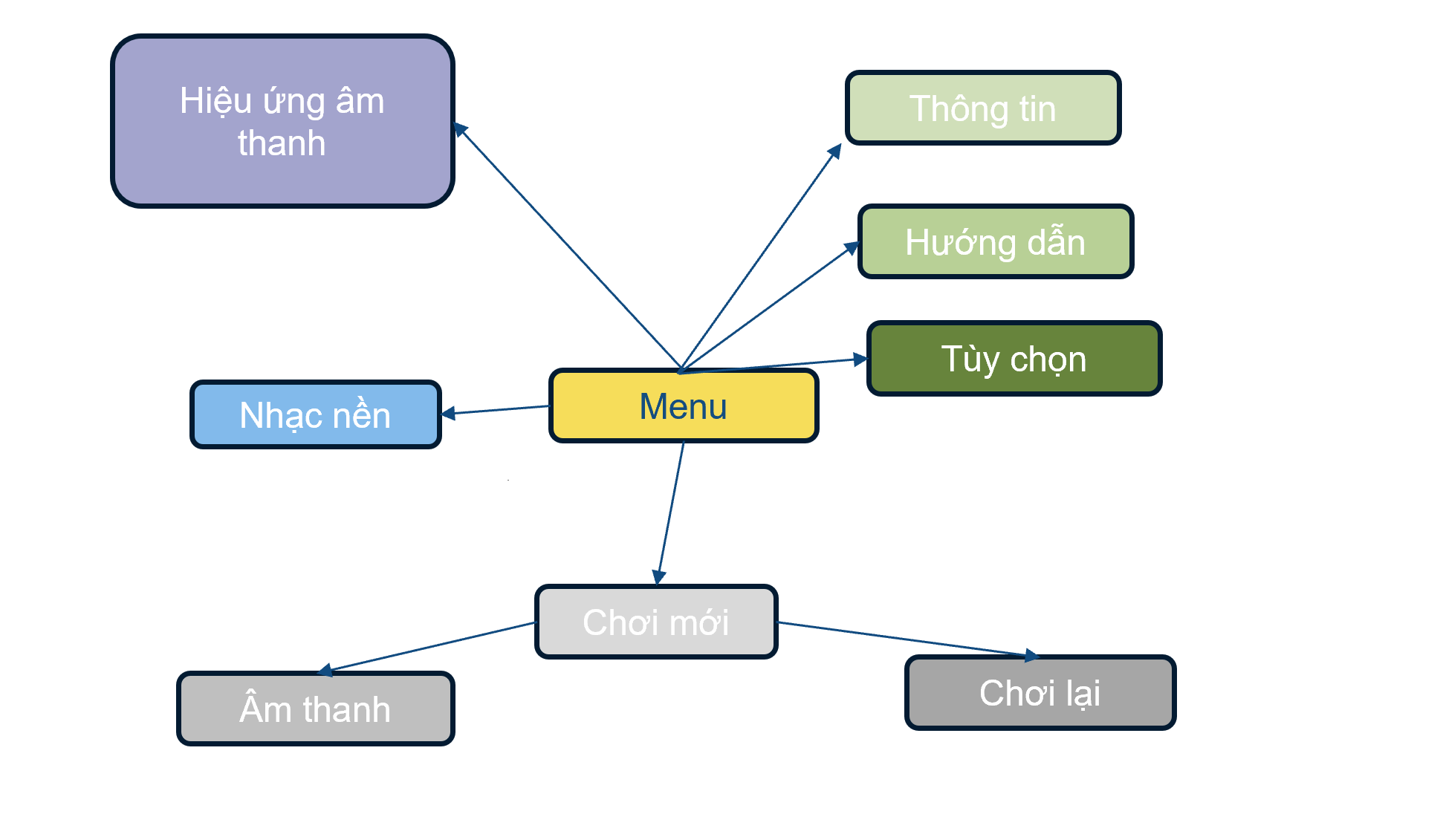
=>> Khi đó ta sẽ thu hết đá thuộc ô của mình và thực hiện so sánh để phân loại thắng thua

* + Nếu số đá người thứ 1 lớn hơn người thứ 2 thì người thứ 1 THẮNG và ngược lại.
  + Nếu số đá người thứ 1 lớn hơn người thứ 2 thì HÒA

**Chương 5: Ứng dụng**

**5.1 Giới thiệu chương trình ứng dụng.**

**5.1.1 Hệ thống mô hình trong game**



Menu chính chứa các đối tượng lựa chọn:

- Chơi mới: Chọn để bắt đầu chơi game. Sau khi chọn sẽ hiện lên một cửa sổ mới để chơi game.

- Thông tin: Sẽ hiện lên thông tin của người viết game.

- Tùy chọn: Hiện lên cấp độ chơi

- Hiệu ứng âm thanh: Âm thanh khi di chuyển và ăn quân - Nhạc nền: Bật tắt nhạc nền.

- Hướng dẫn: Xem hướng dẫn để chơi.

**5.1.2 Giao diện game**

Menu:



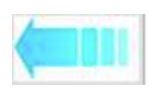
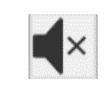
Playgame:

**5.1.3 Các nút chức năng cơ bản trong game**

:Chức năng Undo - Redo: người chơi có thể thu hồi nước cờ bằng cách click chuột vào biểu tượng  và đi tới các nước sau (nếu có) bằng cách click chuột vào 

:Trở về menu

: Chơi lại : Chọn chiều di chuyển qua phải

: Chọn chiều di chuyển qua trái :Bật tắt âm thanh

**- Chức năng hiển thị lượt chơi**: khi tới lượt chơi của quân nào thì màn hình sẽ hiện lên lượt đánh của quân đó.

**5.2: Cài đặt**

**5.2.1 Ngôn ngữ và công cụ sử dụng**

Để xây dựng game Ô ăn quan , nhóm sử dụng: và phù hợp với công cụ lập trình

- Công cụ: Visual Studio, pycharm để thiết kế giao diện

**5.2.2 Các hàm các đoạn code chính**

**-** Dưới đây là những đoạn code chính:

class OAnQuan:

    def \_\_init\_\_(self):

        self.board = [5] \* 12

        self.board[5] = self.board[11] = 10

        self.current\_player = 0

        self.scored\_01 = 0

        self.scored\_02 = 0

* Trong đoạn mã code trên hàm được sử dụng là **\_\_init\_\_ .** Đây là hàm khởi tạo ( contructor ) của lớp “ **Ô Ăn Quan**” . Hàm này được gọi khi một hàm mới của lớp được tạo ra.

def print\_board(self):

        print("     ", end="")

        for k in range(10, 5, -1):

            print(self.board[k], " ", end="")

        print()

        print(self.board[11], "                 ", self.board[5])

        print("     ", end="")

        for k in range(5):

            print(self.board[k], " ", end="")

        print("\n")

* Trong đoạn mã code trên hàm chính được sử dụng là ( **print\_board )** dùng để hiển thị trạng thái hiện tại của bảng ô ăn quan.

 def play\_game(self):

        while True:

            if self.current\_player == 2:

                self.current\_player = 0

            self.print\_board()

            if self.board[5] == 0 and self.board[11] == 0:

                self.scored\_01 += sum(self.board[0:5])

                self.scored\_02 += sum(self.board[6:11])

                if self.scored\_01 < self.scored\_02:

                    print("Player 2: Win!")

                elif self.scored\_02 < self.scored\_01:

                    print("Player 1: Win!")

                else:

                    print("Hòa")

                break

            if self.current\_player == 0:

                print("Player 1")

                hole = int(input("Chọn lỗ (0-4) <=> (1-5): "))

                if hole == 5 or hole == 11 or self.board[hole] == 0:

                    print(" lựa chọn không hợp lệ. vui lòng chọn lại !")

                    continue

            if self.current\_player == 1:

                print("Player 2")

                hole = int(input("Chọn lỗ (6-11) <=> (1-5): "))

                if hole == 5 or hole == 11 or self.board[hole] == 0:

                    print(" lựa chọn không hợp lệ. vui lòng chọn lại !")

                    continue

            global stones, i

            stones = self.board[hole]

            self.board[hole] = 0

* Trong đoạn mã code trên hàm chính được sử dụng là **play\_game** dùng để quản lý quá trình chơi của trò chơi ô ăn quan.

def phanphoi(self):

                global stones, i

                if chieu == "t":

                    while stones > 0:

                        if i < 0:

                            i = 11

                        self.board[i] += 1

                        stones -= 1

                        i = i - 1

                    if i < 0:

                        i = 11

                    while self.board[i] > 0:

                        if i == 5 or i == 11:

                            break

                        stones2 = self.board[i]

                        self.board[i] = 0

                        i = i - 1

                        while stones2 > 0:

                            if i < 0:

                                i = 11

                            self.board[i] += 1

                            stones2 -= 1

                            i = i - 1

                        if stones2 == 0 and i == -1:

                            i = 0

                    diem\_cong = 0

                    while self.board[i] == 0:

                        if self.board[i - 1] > 0:

                            if self.current\_player == 0:

                                self.scored\_01 += self.board[i - 1]

                            if self.current\_player == 1:

                                self.scored\_02 += self.board[i - 1]

                            diem\_cong += self.board[i - 1]

                            self.board[i - 1] = 0

                            i = i - 1

                        elif self.board[i - 1] == 0:

                            break

                        if i == 0:

                            i = 12

                        i = i - 1

                    if self.current\_player == 0:

                        print("Người chơi 1 nhận được: ", diem\_cong)

                        print("Scored 01: ", self.scored\_01)

                    if self.current\_player == 1:

                        print("Người chơi 2 nhận được: ", diem\_cong)

                        print("Scored 02: ", self.scored\_02)

                #Rải quân về bên phải

                elif chieu == "p":

                    while stones > 0:

                        if i > 11:

                            i = 0

                        self.board[i] += 1

                        stones -= 1

                        i = i + 1

                    if i > 11:

                        i = 0

                    while self.board[i] > 0:

                        if i == 5 or i == 11:

                            break

                        stones2 = self.board[i]

                        self.board[i] = 0

                        i = i + 1

                        while stones2 > 0:

                            if i > 11:

                                i = 0

                            self.board[i] += 1

                            stones2 -= 1

                            i = i + 1

                        if stones2 == 0 and i == 12:

                            i = 0

                    #ăn quân

                    diem\_cong = 0

                    while self.board[i] == 0:

                        if self.board[i + 1] > 0:

                            if self.current\_player == 0:

                                self.scored\_01 += self.board[i + 1]

                            if self.current\_player == 1:

                                self.scored\_02 += self.board[i + 1]

                            diem\_cong += self.board[i + 1]

                            self.board[i + 1] = 0

                            i = i + 1

                        elif self.board[i + 1] == 0:

                            break

                        if i == 11:

                            i = -1

                        i = i + 1

                    if self.current\_player == 0:

                        print("Người chơi 1 nhận được: ", diem\_cong)

                        print("Scored 01: ", self.scored\_01)

                    if self.current\_player == 1:

                        print("Người chơi 2 nhận được: ", diem\_cong)

                        print("Scored 02: ", self.scored\_02)

            phanphoi(self)

            self.current\_player += 1

* Trong đoạn mã code trên hàm chính được sử dụng là **def phanphoi(self)** đây là hàm thực hiện quá trình phân phối đá từ một ô đã chọn ra các ô tiếp theo theo chiều trái hoặc phải dựa vào việc lựa chọn “**chiều”** . Hàm này cũng xử lý việc ăn quân và cập nhập điểm cho người chơi
* Trong đoạn mã code dưới đây hàm chính được sử dụng là **def raithem()** . Đây là hàm thực hiện việc rải đá khi có người chơi hoặc cả hai người chơi đã hết đá trong các lỗ của mình

def raithhem():  
 if self.current\_player == 0 or self.current\_player == 2:  
 if sum(self.board[0:5]) == 0:  
 if self.scored\_01 == 0 or self.scored\_02 == 0:  
 if self.scored\_01 < self.scored\_02:  
 print("Player 2: Win!")  
 elif self.scored\_02 < self.scored\_01:  
 print("Player 1: Win!")  
 else:  
 print("Hòa")  
 if 0 < self.scored\_01 < 5:  
 new\_stones = self.scored\_01  
 self.scored\_01 = 0  
 for z in range(0, new\_stones):  
 self.board[z] += 1  
  
 if self.scored\_01 > 5:  
 self.scored\_01 -= 5  
 self.board[0] = self.board[1] = self.board[2] = self.board[3] = self.board[4] = 1  
  
 if self.current\_player == 1:  
 if sum(self.board[6:11]) == 0:  
 if self.scored\_01 == 0 or self.scored\_02 == 0:  
 if self.scored\_01 < self.scored\_02:  
 print("Player 2: Win!")  
 elif self.scored\_02 < self.scored\_01:  
 print("Player 1: Win!")  
 else:  
 print("Hòa")  
  
 if 0 < self.scored\_02 <= 5:  
 new\_stones = self.scored\_02  
 self.scored\_02 = 0  
 for z in range(6, 6 + new\_stones):  
 self.board[z] += 1  
 if self.scored\_02 > 5:  
 self.scored\_02 -= 5  
 self.board[6] = self.board[7] = self.board[8] = self.board[9] = self.board[10] = 1  
  
raithhem()

**5.3 KẾT QUẢ CHẠY CHƯƠNG TRÌNH**

**5.2.1 Kết quả thử nghiệm**

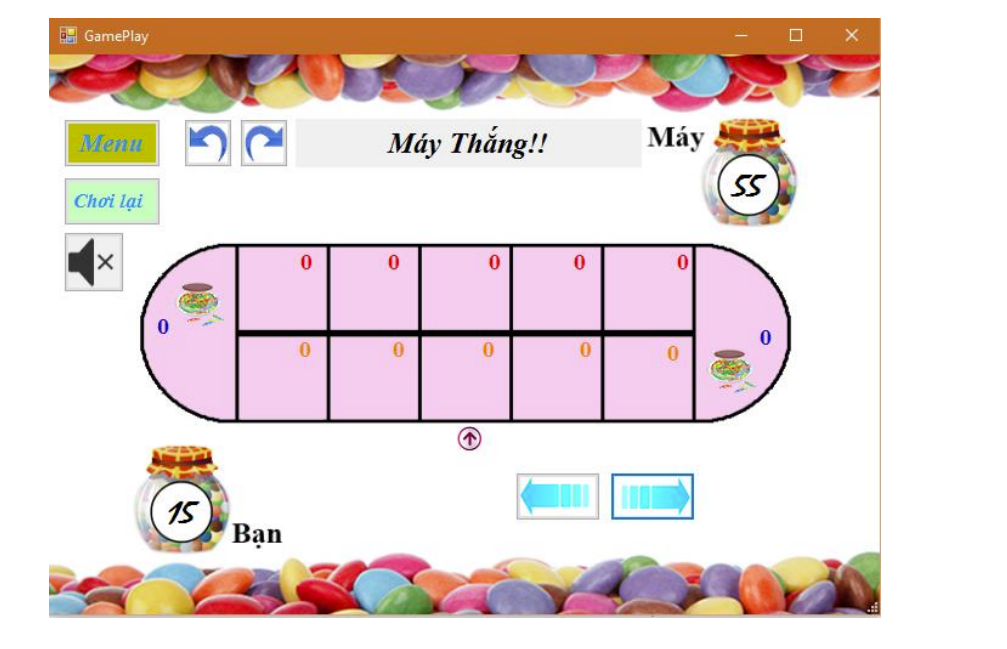
- Các nước đi đã thực hiện đúng luật của trò chơi.

- Các chức năng Undo, Redo hoạt động.

- Các hiệu ứng âm thanh hoạt động tốt.

Kết thúc game:

*-Máy thắng:*

**

-*Bạn thắng:*





**Chương 6: Kết quả chạy chương trình**

- Chương trình khởi tạo 1 bảng Ô Ăn Quan với các điều kiện ban đầu.

- Người chơi lựa chọn ô để thực hiện di chuyển đá.

- Sau mỗi lượt di chuyển, trạng thái mới của bảng được hiển thị trên màn hình.

- Trò chơi tiếp tục cho đến khi một trong hai người chơi không thể thực hiện bước nào nữa.

- Khi trò chơi kết thúc, trò chơi thông báo người chơi chiến thắng hoặc hòa, thua.

**Chương 7 : Kết luận**

**7.1 KẾT QUẢ ĐẠT ĐƯỢC**

1. Game đã chạy đúng luật của trò chơi.
2. Đã hoàn thành các chức năng cơ bản trong game.
3. Có hiệu ứng âm thanh.
4. Giao diện dễ dùng có các chức năng cơ bản.
5. Nhóm khá ưng ý với giao diện nhưng cần hoàn thiện thêm.
6. Tìm hiểu và hiểu được cách máy tính lựa chọn nước đi, tính toán trước nhiều bước giống như suy nghĩ của con người
7. Có các cấp độ cho người chơi lựa chọn.

**7.2 HẠN CHẾ**

1. Cách chọn cấp độ.
2. Chưa đưa thời gian vào trong game.
3. Chưa cài đặt được danh sách lưu điểm người chơi.
4. Chưa có form thông báo kết quả riêng và hiện lên màn hình.

**7.3 HƯỚNG PHÁT TRIỂN**

1. Khắc phục các mặt hạn chế, hoàn thiện các chức năng cơ bản trong game.
2. Tối ưu hóa các vấn đề trong game: Tăng sự thông minh cho máy mà tránh được việc tràn bộ nhớ.
3. Hoàn thiện giao diện hiện tại.
4. Chương trình xử lý việc phân phối đá một cách đầy đủ và chính xác.
5. Điều kiện kết thúc trò chơi được kiểm tra một cách đúng đắn.

**7.4 TỔNG KẾT**

- Chương trình Ô Ăn Quan đã đạt được mục tiêu cơ bản là thực hiện hóa một phiên bản đơn giản của trò chơi truyền thống. Đây có thể là một bước khởi đầu tốt để phát triển và mở rộng chương trình trong tương lai, với sự thêm vào các tính năng mới và cải thiện người dùng.

**Chương 8 : Tài liệu tham khảo**

1. Board Game Geek (BGG).
2. GitHub Repositories.
3. Scholarly Articles.
4. Libraty Resources.
5. Forums và Blogs.