TB28 - KMIN

**Report SnakeGame**

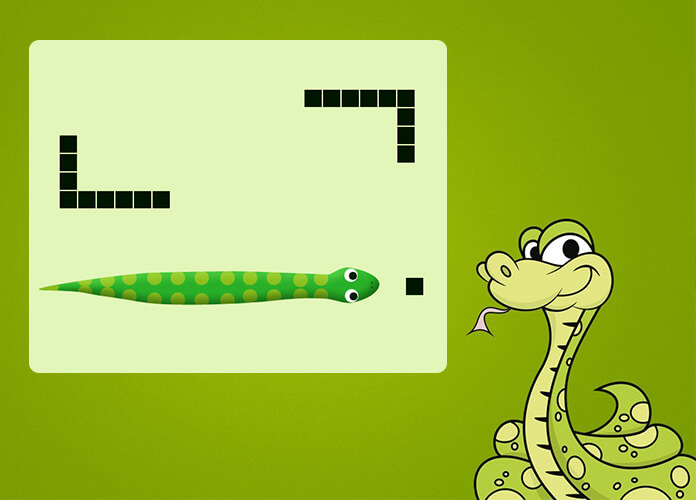
Đồ án cuối khóa

horizontal line

# Placeholder image

# Giới thiệu

**SnakeGame** là 1 game dựa theo logic của game rắn săn mồi. Game này được viết bằng ngôn ngữ TypeScript và chạy bằng terminal.



## Lý do lựa đề tài

Lý do game này được lựa là vì sau khi research, tôi thấy có rất ít người viết game này bằng Javascript và chạy được trên terminal, chứ chưa kể là viết bằng TypeScript. Ngoài ra, vì viết để chạy trên Terminal, tôi nghĩ là vì trước đây mình chưa làm việc nhiều với terminal, nên đây sẽ là kiến thức hữu dụng, và đồng thời giúp tôi rèn luyện tính logic tốt hơn.

Trong quá trình làm có khá nhiều khó khăn tôi gặp phải, tuy nhiên, khi làm xong, tôi có thể tạm thời tổng kết như sau:

* Ứng dụng được các lý thuyết cơ bản của TypeScript được dạy trên lớp.
* Ứng dụng vòng lặp.
* Ứng dụng dạng cấu trúc dữ liệu Array, với mỗi thành phần trong Array là 1 Object.
* Tự tìm hiểu thêm được về process trong Nodejs, và hiểu được sơ sơ thêm về Event Loop.
* Học thêm về Promise
* Hiểu thêm về thư viện csv-writer và csv-parser, 2 thư viện bên ngoài tôi đã cài thêm để làm việc với csv.

## Thiết kế của sản phẩm

Game này được thiết kế với các giao diện cơ bản như sau:

* Giao diện chính
* Giao diện chơi game
* Giao diện sau khi chơi game (gọi là giao diện gameOver)
* Giao diện xem bảng điểm (gọi là giao diện scoreBoard)

### 2.1 Giao diện chính:



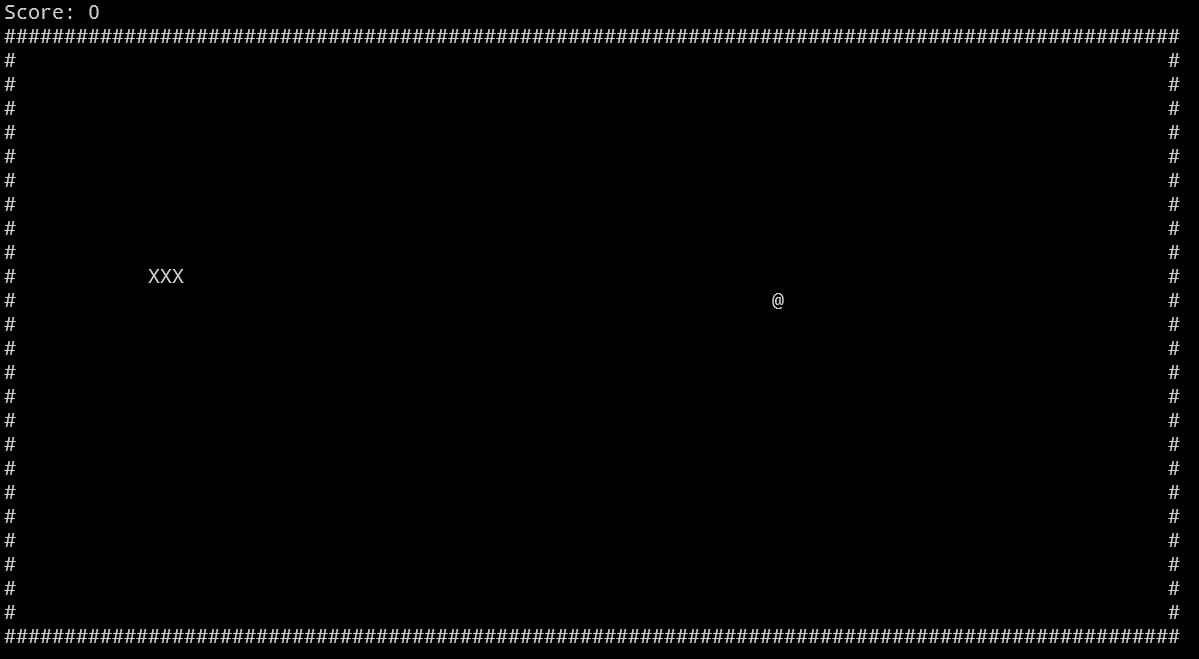
Tại giao diện chính, người chơi sẽ được thấy title của game cùng với menu chính, bao gồm các lựa chọn:

1. Play Game
2. View Scoreboard
3. Exit Game

Nếu lựa a), người chơi sẽ được chuyển qua giao diện chơi game

Nếu lựa b), người chơi sẽ được chuyển qua giao diện scoreBoard

### 2.2 Giao diện chơi game:



Giao diện này sẽ bao gồm các thành phần sau:

* Score - Điểm số của bạn. Khi mới vào điểm này sẽ bằng 0.
* Rắn được hiển thị bằng dấu X, ở đây tôi setup cho con rắn có chiều dài mặc định mới vào là 3, nên màn hình này đang hiển thị 3 dấu X.
* Quả táo được hiển thị bằng dấu @.
* Viền của phần game được chơi được hiển thị bằng dấu #

Nếu người dùng ăn được quả táo, điểm sẽ được cập nhật

### 2.3 Giao diện gameOver:



Giao diện gameOver sẽ cho bạn biết số điểm bạn đạt được. Ngoài ra còn hiển thị 1 menu, bao gồm các lựa chọn:

* Replay
* Save score
* Return main menu
* Exit game

Nếu bạn lựa “Replay”, game sẽ reset các thông số và cho phép bạn chơi lại.

Nếu bạn lựa “Save Score”, lúc này bạn sẽ được yêu cầu nhập tên (không bắt buộc) để lưu điểm số của bạn vào file csv.

Nếu bạn lựa “Return main menu”, lúc này màn hình sẽ chuyển về giao diện chính.

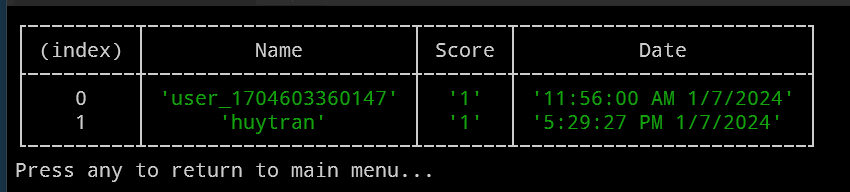
### 2.4 Giao diện scoreBoard:

Tại giao diện này, người dùng sẽ được thấy với 2 trạng thái:

1. Game chưa có điểm của người chơi



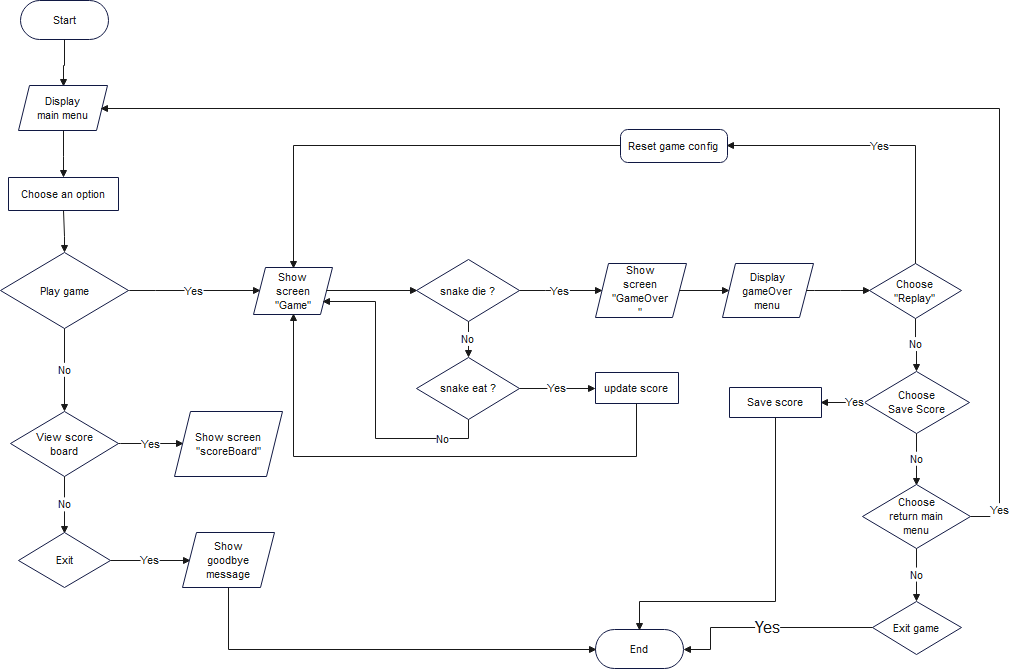
1. Game có điểm của người chơi



Màn hình này hiện tại chỉ dùng để hiển thị điểm. Có thể trong tương lai sẽ được update thêm tính năng cho phép search tên người chơi, hoặc là sort theo các thông số như score, name, thời gian chơi, etc…

### 2.5 Flowchart cơ bản

Sau khi đi qua các design cơ bản, tôi trình bày 1 flowchart tổng quát về hệ thống game “SnakeGame” để giúp các bạn hiểu rõ hơn về hệ thống



## Chi tiết từng tính năng

Game “SnakeGame” này có nhiều tính năng nhỏ nhặt như là update điểm số, blink khi rắn chết hay là loading screen, nhưng tôi sẽ lượt bỏ các tính năng quá cơ bản hay nhỏ nhặt không cần đi sâu vào chi tiết mà chỉ tập trung vào các tính năng sau:

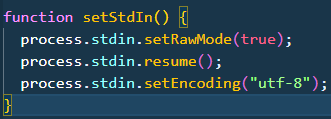
### 3.1 Event listeners:

Đây là cơ chế quan trọng nhất game, nó cho phép terminal nhận data từ bàn phím không cần phải gõ và bấm phím enter để được nhận. Nó cho phép tôi có thể điều khiển lựa chọn menu lên xuống bằng phím mũi tên, hoặc là điều khiển rắn bằng phím mũi tên.

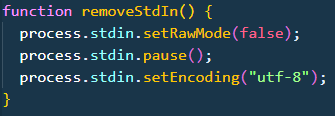
* Chức năng này sử dụng 1 API có sẵn của NodeJs - Process.
* Step- by -Step:

1. Set toàn bộ terminal về raw mode
2. Chuyển dạng encode của terminal về utf-8
3. Setup listener cho sự kiện “data”

* Đây là code mà tôi đã sử dụng để setup bước a và b ở phần trên:



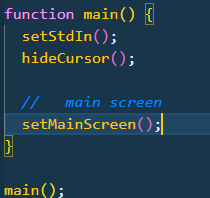
* Sau khi sử dụng process.stdin, chúng ta phải trả terminal lại về trạng thái ban đầu, và hàm sau đây được tôi viết sẵn để trả 2 trạng thái a và b về ban đâu.



Vậy câu hỏi đặt ra là chúng ta sẽ bắt đầu khởi tạo event listener tại đâu. Câu trả lời là ở mỗi màn hình, và tương ứng mỗi chức năng

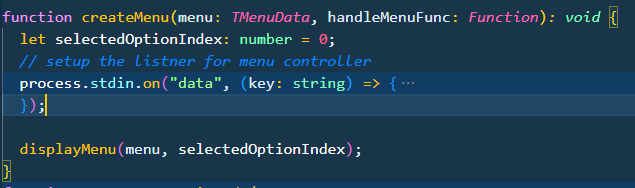
Giờ chúng ta sẽ đi vào 1 ví dụ cụ thể chính là hàm createMenu

Flow chạy sẽ là: Từ hàm main, chúng ta setup stdIn global trước, rồi sau đó mới gọi màn hình chính là “main screen”



Ngay dòng đầu tiên, các bạn có thể thấy hàm setStdIn() được gọi, nhiệm vụ hàm này sẽ là setup toàn bộ process.stdin cho riêng terminal ngay khi app bắt đầu chạy.

Giờ vào hàm createMenu():

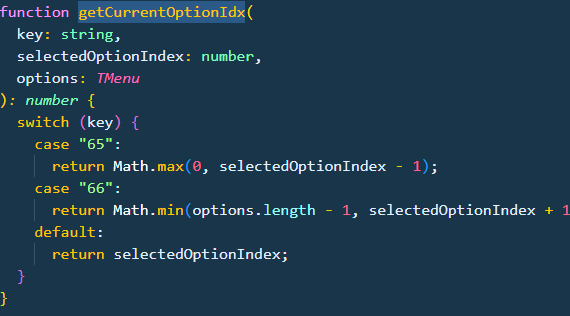


Trong hàm này, dòng “process.stdin.on(‘data’)” chính là event listener, dòng này cho phép khi tôi gõ A, terminal sẽ nhận được và chuyền chữ A về hàm này, và lưu vào biến key, biến được truyền vào hàm callback ngay sau đó dưới dạng đối số.

Đây chính là full của hàm callback thuộc về process.stdin.on(“data”):



Trong hàm này, bạn có thể thấy ở dòng đầu tiên, tôi lấy giá trị của key ra dưới dạng string, và xử lý tùy theo điều kiện. Như trong hình, ở điều kiện 1 chính là nơi mà tôi xử lý việc bấm nút mũi tên lên xuống của menu, cụ thể là hàm **getCurrentOptionIdx**



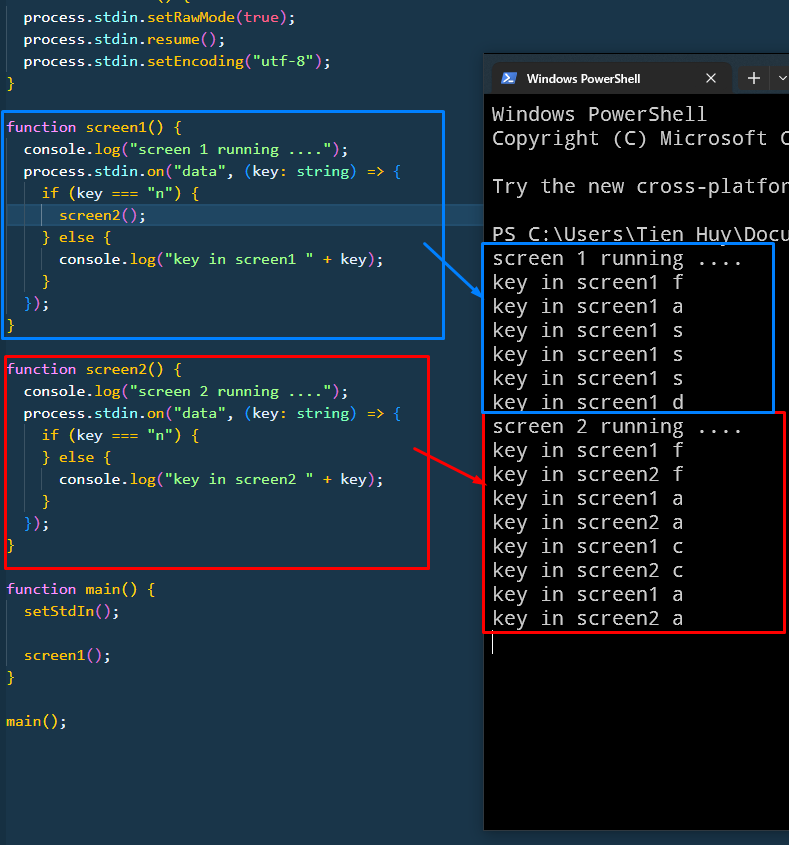
Như bạn thấy trong hình, key lúc này tôi truyền vào sau khi xử lý, nếu giá trị là 65, tức là tôi đang bấm phím lên, và nếu giá trị là 66 tức là tôi đang bấm phím xuống.

Quay lại hàm handler, như bạn thấy ở if số 2 và số 3, khi tôi quit game hay khi tôi bấm enter để tiến hành xử lý 1 lựa chọn nào đó trong menu, việc cần làm là phải gỡ bỏ listener, tại sao như vậy ?



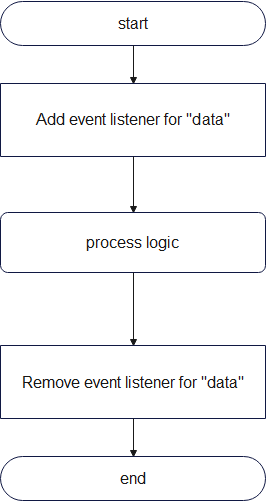
NodeJs không phân biệt được các listener khác nhau, nếu bạn không gỡ ở màn hình a, và tiếp tục tạo 1 dòng code “process.stdin.on (“data”, ()..)” tại màn hình B, lúc này cả 2 listener đều nghe và rất khó debug, hãy tưởng tượng nếu đó là trường hợp bài toán có 10 màn hình khác nhau thì sao ?

Ví dụ:



Trong hình này, tôi define 2 hàm screen1 và screen2, tượng trưng cho 2 màn hình, như bạn thấy,khi từ screen1() gọi screen2(), tôi đã không removeListener, điều này dẫn tới ở screen2, câu lệnh của screen1 vẫn được lắng nghe và cả scree2 cũng vậy. Điều này không nên.

Dưới đây là hình flowchart cơ bản cho việc sử dụng eventListener

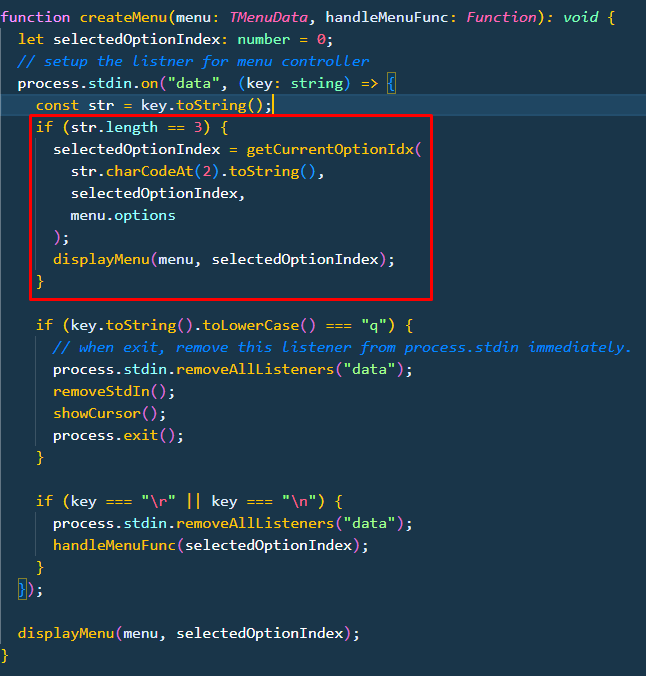


### 3.2 Cơ chế điều khiển menu:

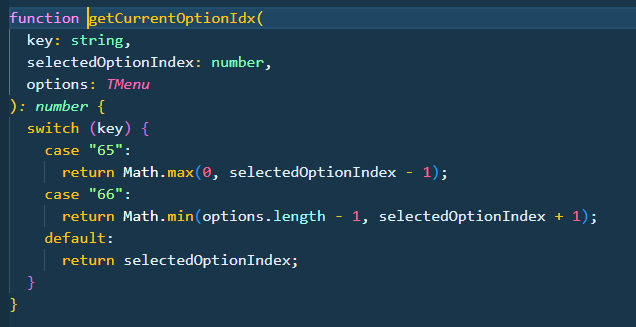
Tôi sẽ giải thích cách mà nút điều khiển kết hợp với các logic code khác cho phép chúng tay thấy hiệu ứng này



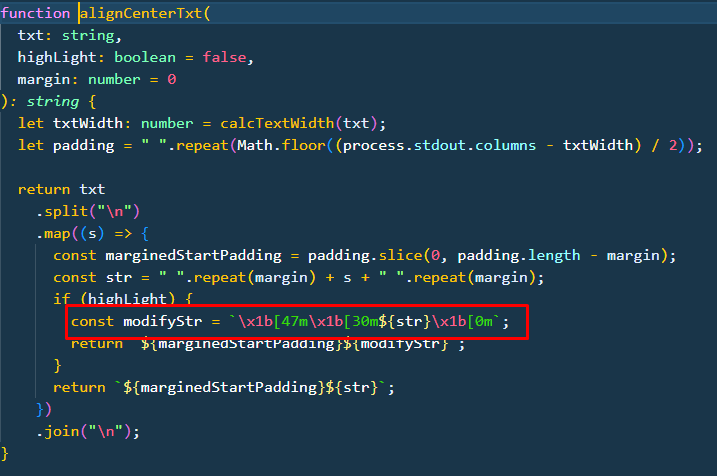
Đầu tiên, hãy xem lại hàm createMenu, tại câu điều kiện If đầu tiên:

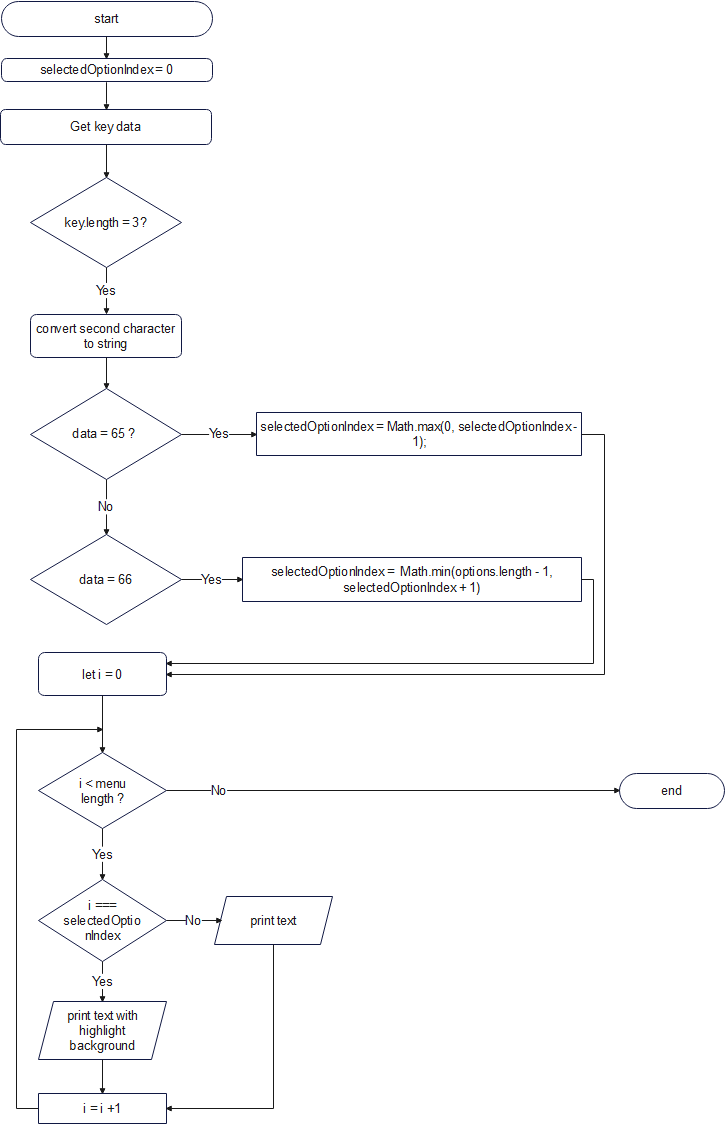


Hàm **getCurrentOptionIndex** sẽ dựa vào 3 yếu tố: key data, index của option hiện tại, và danh sách options của menu.



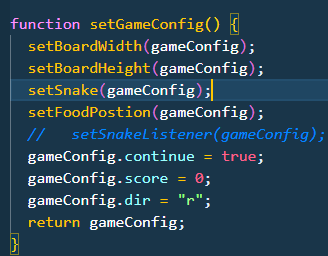
Trong hàm này, nếu bấm mũi tên xuống, key data sẽ là 66, lúc này chúng ta sẽ so sánh và tìm ra số nhỏ nhất giữa 2 số là: Index của option cuối, và Index của option chúng ta muốn lựa tiếp.   
Giả sử: Ở lần 1, index hiện tại là 0, nên selectedOptionIndex = 0, lúc này tôi bấm phím xuống, thì code sẽ thực hiện so sánh Max.min(2, 0+1) = 1, vậy là selectedOptionIndex tiếp theo sẽ là 1. Hàm **getCurrentOptionIndex** trả về 1, và hàm **displayMenu** sẽ dùng data đó, để in ra màn hình, kèm theo background tô màu, còn đối với các option khác, chỉ hiển thị bình thường.



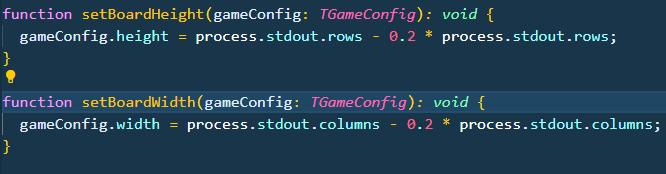
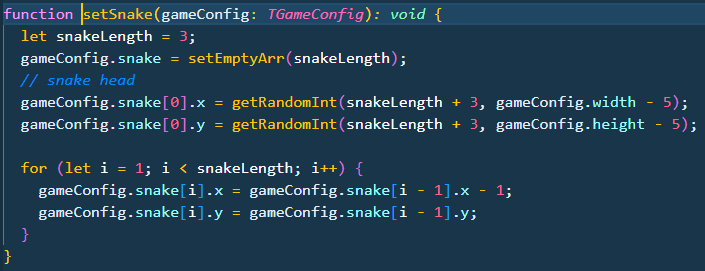
Hàm này tôi tạo ra với mục tiêu là để có thể tái sử dụng và tạo ra nhiều menu khác nhau mà không cần phải làm quá nhiều thao tác. Dưới đây là flowchart cơ bản tôi đã tổng quát hóa.  


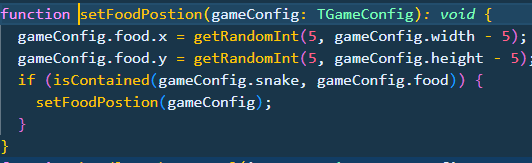
### 3.3 Setup game config:

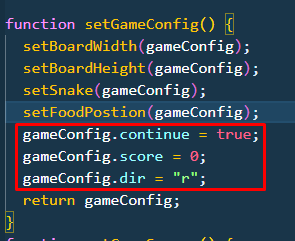
Tại main menu, khi bạn lựa option “Play game”, điều đầu tiên được thực thi chính là gameConfig được setup  

Cơ bản, chúng ta sẽ tính toán ra kích thước của phần game board dựa theo kích thước của terminal, bao gồm chiều cao và chiều rộng.

  
Sau đó, chúng ta phải tính được điểm nào trên màn hình mà chúng ta muốn đặt rắn, và điểm này phải random dựa vào thông số chiều cao và chiều rộng thực sự của game map  
  
Tương tự, dựa vào thông số chiều cao và chiều rộng thực sự của game map, chúng ta tính ra tọa độ bất kỳ của đồ ăn, nhưng, điểm này tuyệt đối không được phép nằm đè lên con rắn.





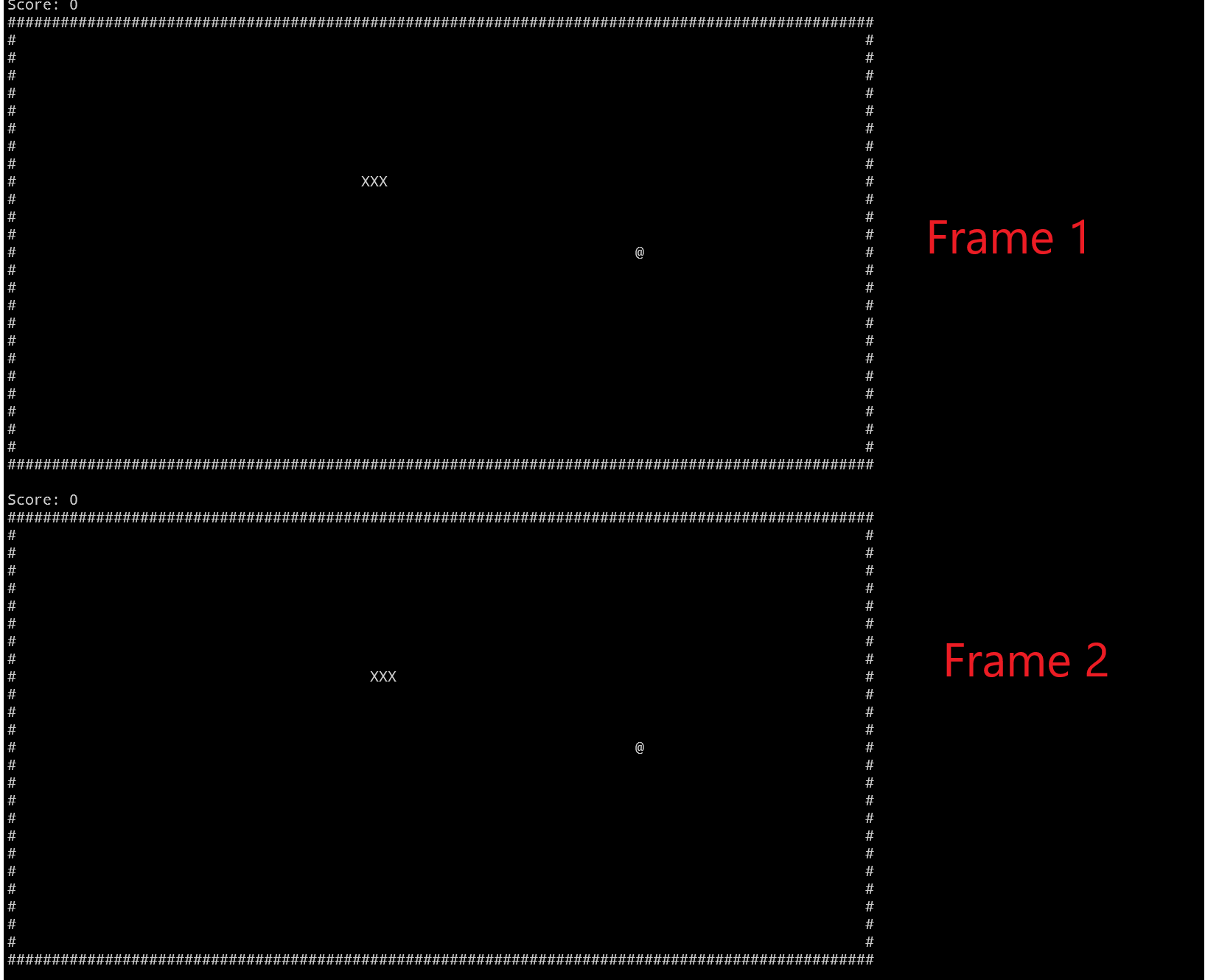
3 dòng dưới đây khá quan trọng, vì nó ảnh hưởng tới việc chúng ta có thể chơi lại game hay không ? Nếu bạn lựa chọn “replay”, chúng ta sẽ gọi lại hàm này, và điều đó có nghĩa là chúng ta phải reset việc continue chơi game, và score = 0.

### 3.4 Cơ chế chơi game:

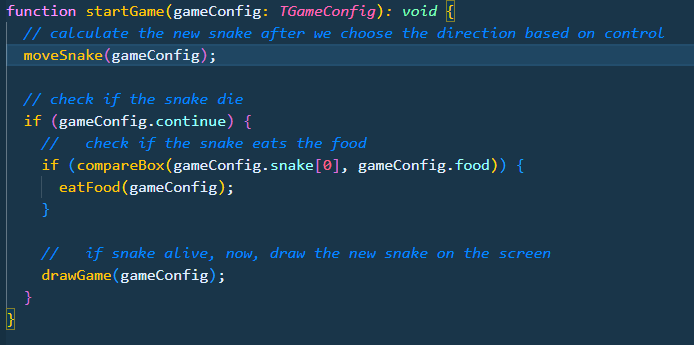
Về cơ bản, game là tập hợp các frame chồng lên nhau và được chạy với tốc độ cao, từ đó cho chúng ta cảm giác nhân vật trong game đang chạy, nhảy. Với logic này, “SnakeGame” được làm ra bằng cách liên tục vẽ lên console map của game, bao gồm: Điểm, bàn chơi, rắn và đồ ăn.



Ngay khi vừa vào, hàm **drawGame** được gọi và tiến hành vẽ game lên màn hình, và khi tiến hành chạy tiếp, bạn thấy tôi sử dụng setInterval với thời gian 500ms, tức là cứ 500ms, màn hình sẽ được refresh.

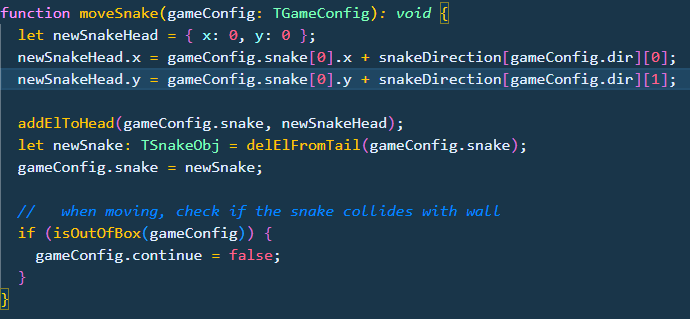


Như trong hình, bạn có thể thấy là con rắn đã di chuyển sang bên phải 1 ô, và đó là 2 frame khác nhau. Giờ thì mình chỉ việc xóa frame 1 đi rồi show frame 2, vậy là chúng ta sẽ thấy con rắn đang di chuyển.

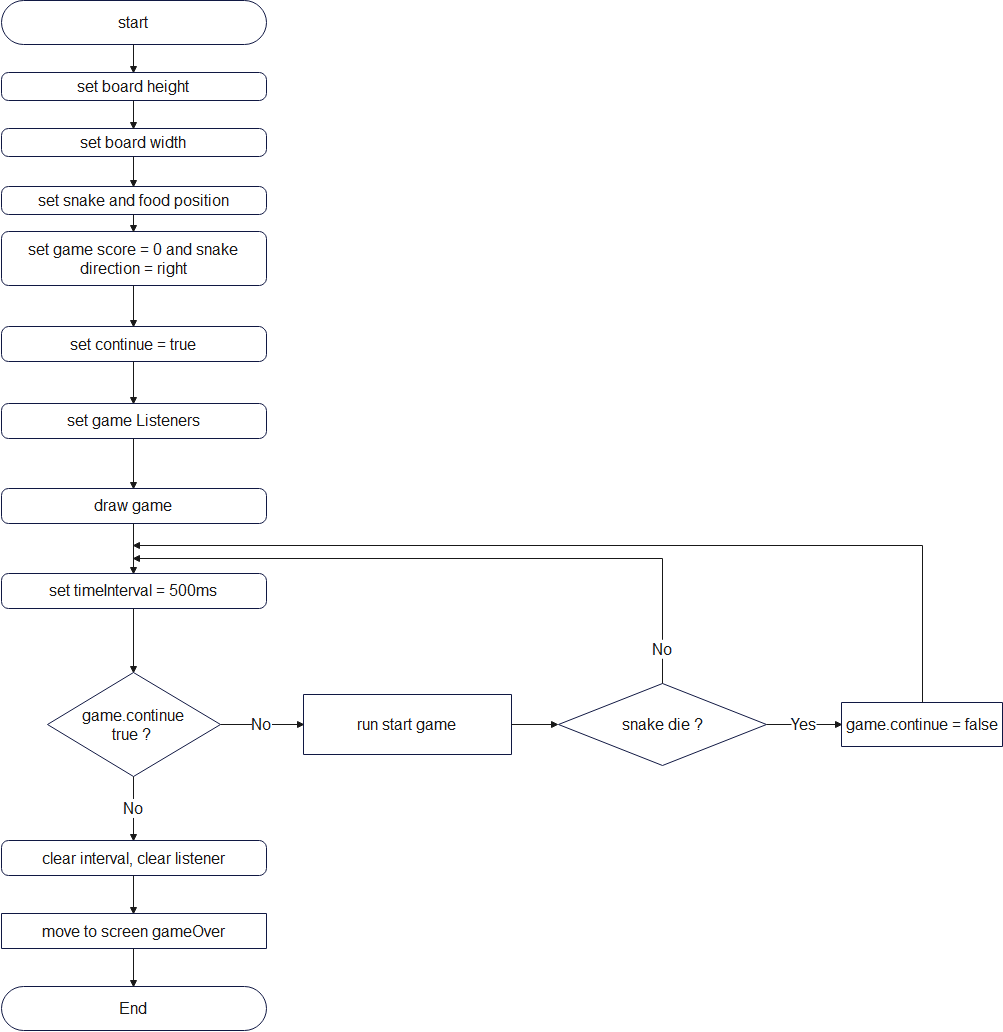


Trong function **startGame**, có 1 hàm gọi là **moveSnake**, hàm này dùng để tính toán xem con rắn đi như thế nào.

Ví dụ: Nếu tôi bấm mũi tên phải, moveSnake sẽ phải tính toán xem sao cho rắn chạy bên phải, bằng cách tính ra điểm mới của head cho con snake, sau đó, remove đi phần tử cuối, và thế là chúng ta có một con rắn mới di chuyển sang bên phải 1 ô.



Flowchart:

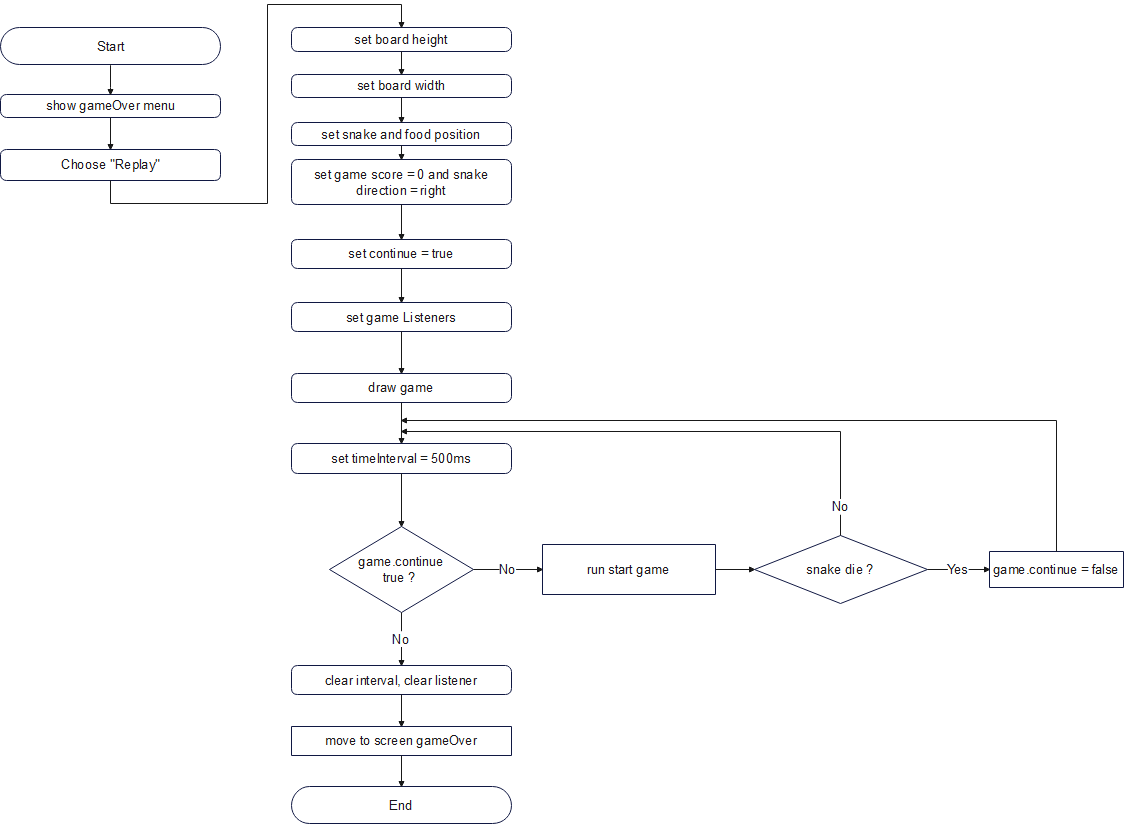


### 3.5 Cơ chế chơi lại game:

Nếu bạn chơi thua, lúc này bạn được chuyển qua màn hình gameOver, lúc này có nhiều lựa chọn.

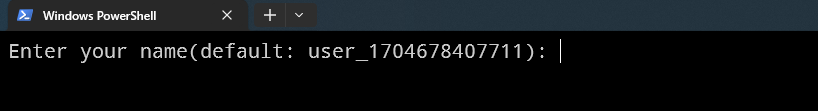
Nếu bạn lựa “Replay”, tức là bạn phải chơi lại, lúc này mọi thông số của game phải được reset, và rồi clear màn hình, bắt đầu in lại game con rắn.

Flowchart:



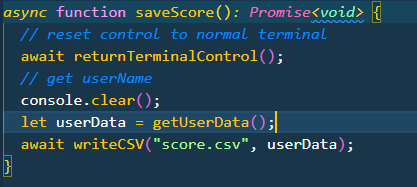
### 3.6 Cơ chế lưu điểm:

Tại màn hình gameOver, bạn có thể chọn option thứ 2 là “saveScore”, lúc này bạn sẽ được phép nhập tên (nhưng không bắt buộc)



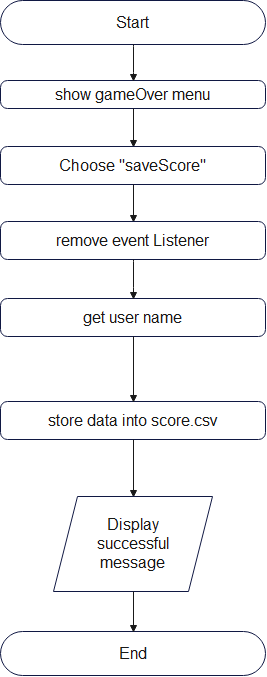
Nếu bạn không nhập tên, hệ thống sẽ tự động lưu điểm của bạn vào file csv với tên là: user\_ + currentDateTime.

Sau đó, data sẽ được lưu vào file “score.csv”, thông qua việc sử dụng thư viện csv\_writer, tuy nhiên, tại đây, chúng ta vẫn còn nhớ là với menu, chúng ta có 1 event listener để điều khiển đúng không ? Vậy chúng ta phải remove event listener đầu tiên, rồi sau đó mới lấy data và lưu vào file csv.





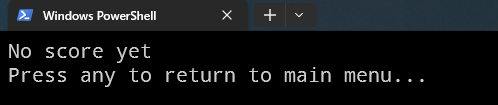
Flowchart:



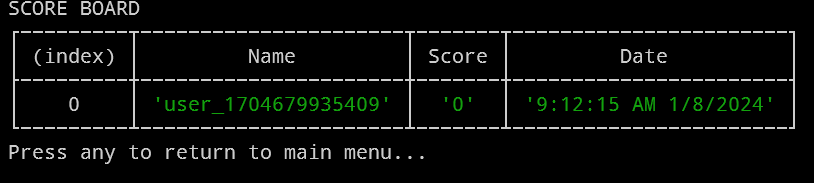
### 3.7 Cơ chế hiển thị scoreBoard:

Ở main menu, chọn “scoreBoard” và bạn sẽ được chuyển sang màn hình scoreBoard để view điểm.

Nếu chưa có điểm nào được lưu, màn hình sau sẽ xuất hiện



Nếu game đã có lưu điểm, bạn sẽ thấy table scoreBoard:



## Tổng kết:

### 4.1 Đánh giá sản phẩm:

* Sản phẩm làm ra có thể tốt hơn, nếu sử dụng kết hợp 1 số thư viện thiên về design cho terminal.
* Việc set vị trí của rắn mới vào đôi lúc hơi sát bờ tường, nên người chơi phải cẩn thận.
* Chưa có cơ chế hiển thị scoreBoard mà cho phép người dùng search hay sort.

### 4.2 Hướng phát triển:

* Thêm cơ chế thay đổi tốc độ của rắn tại menu chính
* Thêm cơ chế cho phép lựa rắn có thể đi xuyên tường.
* Về mặt code-base, bản thân cảm thấy code còn hơi lộn xộn, chưa có sự sắp xếp code tốt lắm