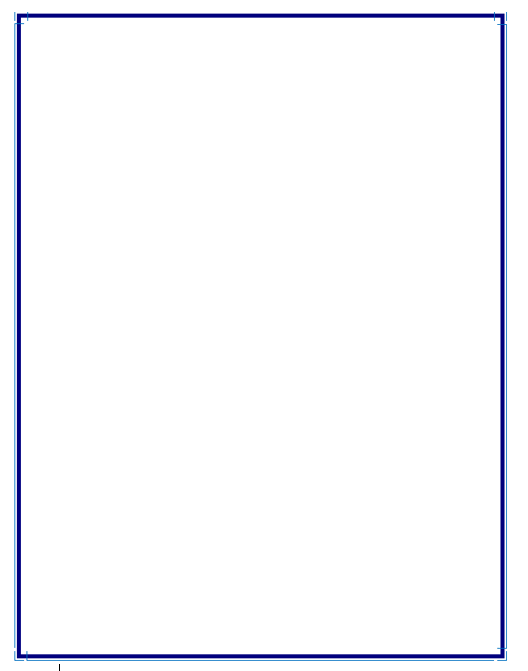
****

**TP. Hồ Chí Minh, tháng 11 năm 2019**

**GVHD: Th.s Trần Công Tú**

**SVTH : Bùi Ngọc Bảo**

**MSSV : 17110100**

**SVTH : Lê Minh Tiến**

**MSSV : 17110236**

**BÁO CÁO ĐỒ ÁN**

**LẬP TRÌNH WEB GAME**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# MỤC LỤC

[LỜI NÓI ĐẦU 1](#_Toc24289730)

[NỘI DUNG 2](#_Toc24289731)

[I. Đặc tả 2](#_Toc24289732)

[II. Phân công công việc 3](#_Toc24289733)

[III. Thiết kế 4](#_Toc24289734)

1.Thuật toán 4

1.1. Game Sudoku 4

1.1.1. Xây dựng giải thuật quay lui bằng phương pháp khử đệ quy 4

1.1.2. Viết hàm Undo và Redo bằng Stack 6

1.2. Game dò mìn (Minesweeper) 7

1.3. Game Snake 9

1.3.1. Khởi tạo rắn và di chuyển 9

1.3.2. Tạo thức ăn cho rắn 12

1.4. Game 2048 13

2. Thiết kế lớp 19

2.1. Danh mục các lớp được sử dụng trong chương trình 19

2.1.1. Game 2048 19

2.1.2. Game Snake 20

2.1.3. Game Minesweeper 20

2.2. Danh mục các phương thức trong một lớp 20

2.2.1. Game 2048 20

2.2.2. Game Snake 21

2.2.3. Game dò mìn 21

[IV. Kết luận 22](#_Toc24289735)

[1. Các khó khăn gặp phải 22](#_Toc24289736)

[2. Ưu điểm 22](#_Toc24289737)

[3. Nhược điểm 22](#_Toc24289738)

[4. Hướng phát triển 22](#_Toc24289739)

[TÀI LIỆU THAM KHẢO 23](#_Toc24289740)

# DANH MỤC CÁC HÌNH

Hình I.1 2

Hình III.1.1.1.1 5

Hình III.1.1.2.1 6

Hình III.1.1.2.2 7

Hình III.1.2.1 7

Hình III.1.2.2 8

Hình III.1.2.3 9

Hình III.1.2.4 9

Hình III.1.3.1.1 10

Hình III.1.3.1.2 10

Hình III.1.3.1.3 11

Hình III.1.3.1.4 11

Hình III.1.3.1.5 11

Hình III.1.3.2.1 12

Hình III.1.3.2.2 12

Hình III.1.3.2.3 12

Hình III.1.3.2.4 13

Hình III.1.4.1 13

Hình III.1.4.2 14

Hình III.1.4.3 15

Hình III.1.4.4 16

Hình III.1.4.5 16

Hình III.1.4.6 17

Hình III.1.4.7 18

Hình III.1.4.8 19

Hình III.1.4.9 19

# DANH MỤC CÁC BẢNG

1. Bảng phân công công việc 3

2. Bảng thiết kế lớp game 2048 19

3. Bảng thiết kế lớp game Snake 20

4. Bảng thiết kế lớp game dò mìn 20

5. Bảng thiết kế phương thức game 2048 20

6. Bảng thiết kế phương thức game Snake 21

7. Bảng thiết kế phương thức game dò mìn 21

# LỜI NÓI ĐẦU

Đầu tiên, chúng em xin cảm ơn quý thầy cô đã đọc bài báo cáo và chấm điểm đồ án cho nhóm. Bài báo cáo này là tổng hợp những nội dung quan trọng sử dụng để làm đồ án. Nội dung dựa trên những môn đã học trong suốt những năm học vừa qua. Như tiêu đề ở trang bìa, nhóm đã lập trình một web game bằng ngôn ngữ JavaScript và chạy trên nền tảng Web. Vì đây là lần đâu viết trên một ngôn ngữ mới nên nhóm sẽ không thể tránh khỏi những sai sót. Kính mong quý thầy cô thông cảm và bỏ qua. Chúng em xin chân thành cảm ơn.

# NỘI DUNG

## I. Đặc tả

Do nhóm có sở thích làm game và muốn tìm hiểu ngôn ngữ mới nên đã xây dựng web game với các trò chơi cổ điển như Sudoku, Snake, 2048, Minesweeper. Mục đích của chúng em là muốn cho người sử dụng những phút giây thư giãn, giải trí sau những giờ làm việc, học tập căng thẳng. Đây là giao diện web game dự kiến mà nhóm em đã tự thiết kế:

Hình I.1. Giao diện web game

## II. Phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| Thứ tự | Tên sinh viên | Mô tả công việc | Phần trăm đóng góp |
| 1 | Lê Minh Tiến | Giao diện web | 55% |
| 2 | Lê Minh Tiến | Sudoku |
| 3 | Lê Minh Tiến | 2048 |
| 4 | Bùi Ngọc Bảo | Snake | 45% |
| 5 | Bùi Ngọc Bảo | Minesweepers |
| 6 | Bùi Ngọc Bảo | Viết báo cáo |

## III. Thiết kế

#### *1. Thuật toán*

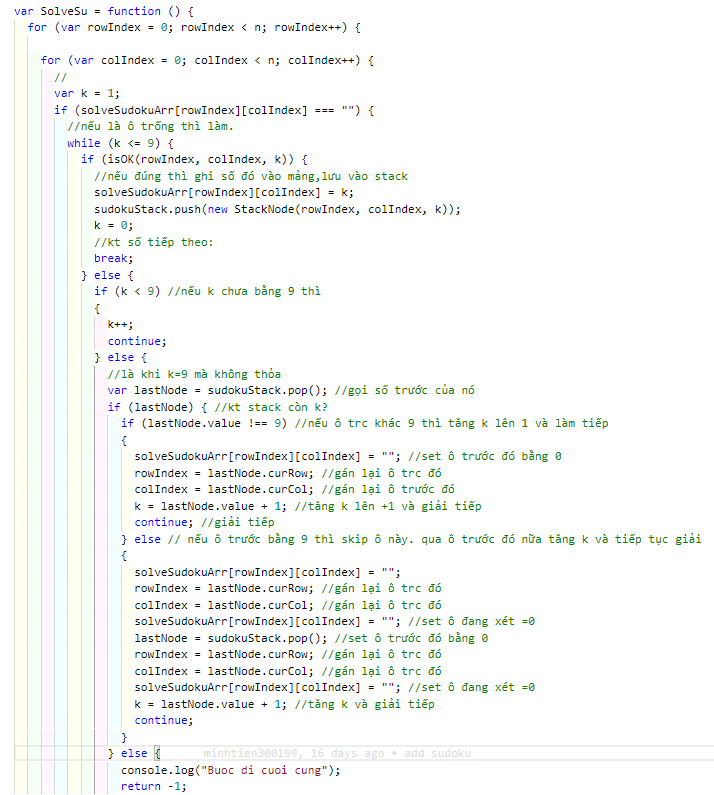
##### 1.1. Game Sudoku

###### **1.1.1. Xây dựng giải thuật quay lui bằng phương pháp khử đệ quy**

Thuật toán quay lui là một thuật toán điển hình để giải các bài toán ứng dụng trong tin học. Bằng việc liệt kê các tình huống, thử các khả năng có thể cho đến khi tìm thấy một lời giải đúng, thuật toán quay lui chia nhỏ bài toán, lời giải của bài toán sẽ là kết quả của việc tìm kiếm theo chiều sâu của tập hợp các bài toán con. Trong suốt quá trình tìm kiếm nếu gặp phải một hướng nào đó mà biết chắc không thể tìm thấy đáp án thì quay lại bước trước đó và tìm hướng khác kế tiếp hướng vừa tìm kiếm đó. Trong trường hợp không còn một hướng nào khác nữa thì thuật toán kết thúc.

Thuật toán giải Sudoku căn bản là sử dụng phương pháp quay lui. Và điển hình của quay lui là sử dụng đệ quy để giải quyết vòng lặp tuy nhiên nó vẫn gặp nhiều hạn chế như tốn bộ nhớ khi xử lý dữ liệu lớn, dẫn tới quá trình xử lý diễn ra một cách chậm chạp. Do đó nhóm đã dùng Stack (ngăn xếp) để tiến hành khử đệ quy.

Áp dụng Stack vào thuật toán giải Sudoku chúng tôi làm như sau: điền ngẫu nhiên từ 1 đến 9, và kiểm tra từng số có phù hợp với ma trận hay không, nếu hợp lệ thì viết số đố vào ô và push vào Stack, tiếp đó là duyệt vị trí tiếp theo của ma trận. Nếu không thỏa thì pop ra, quay ngược về vị trí cũ và điền lại. Dưới đây là code khử đệ quy bằng Stack để giải game Sudoku được viết bằng JavaScript:

Giải thích về các hàm:

Hình III.1.1.1.1 Thuật toán giải Sudoku bằng Stack

+ lastNode: node cuối cùng của Stack.

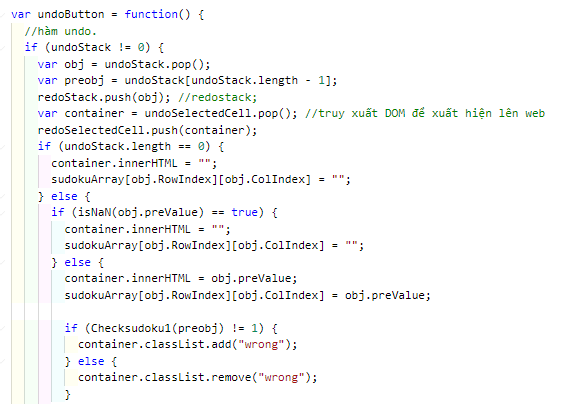
+ isOK(rowIndex, colIndex, k): hàm kiểm tra từng giá trị xem thỏa điều kiện của ma trận hay không.

###### **1.1.2. Viết hàm Undo và Redo bằng Stack**

Trong Sudoku, đôi khi người chơi lựa chọn số không phù hợp với ma trận hiện tại và muốn quay về số trước đó. Vì vậy, chúng tôi viết thêm chức năng Undo và Redo để giải quyết vấn đề trên.

Cả hai chức năng trên đều có những đặc điểm đặc trưng của Stack. Với Undo, khi người chơi nhập bất kì số X nào, thì ta push số đó vào trong Stack thứ nhất, khi người muốn undo thì ta sẽ pop số đó ra, để quay lại số trước và đồng thời số X sẽ được push vào Stack thứ hai với mục đích Redo nếu cần.

Hàm undoButton:

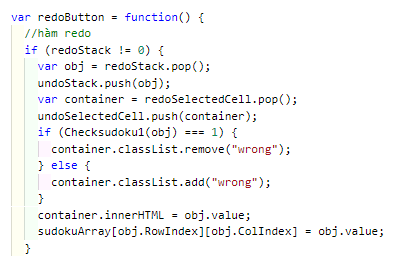


Hình III.1.1.2.1 Hàm undoButton

Trong đó:

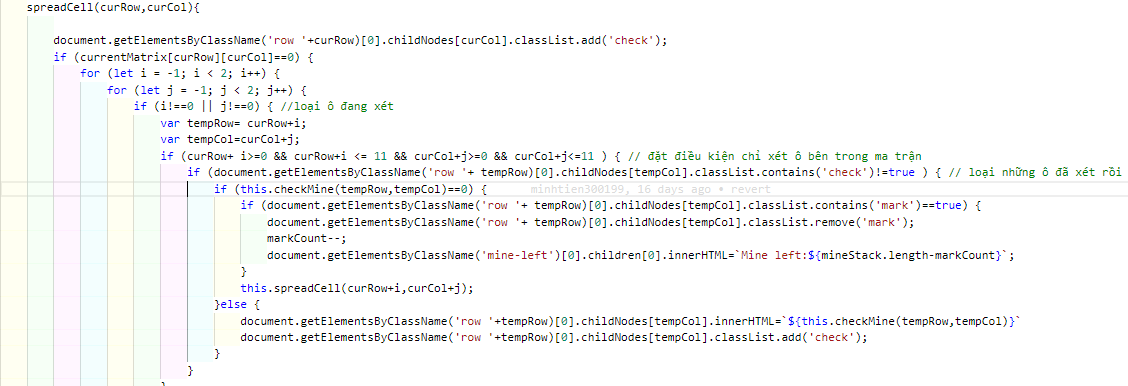
+ Hàm undoStack: lấy giá trị cuối của mảng undoStack. Sau đó cập nhật lại mảng (độ dài trừ đi 1). Rồi bỏ giá trị vào mảng redoStack.

Tiếp theo là hàm redoButton:



Hình III.1.1.2.2 Hàm redoButton

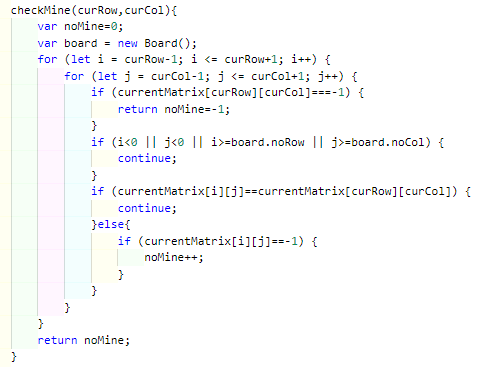
##### 1.2. Game dò mìn (Minesweeper)

Đối với game dò mìn, thì điều khó khăn nhất với chúng tôi là thuật toán loang. Trong trò chơi dò mìn, để giải quyết được trò chơi, ta phải chọn một điểm ban đầu, sau đó loang dần đến những ô trống và dừng lại ở những ô có số để người chơi có thể dò được các bãi mìn. Đây là code về thuật toán loang mà chúng tôi đã viết:

Hình III.1.2.1 Thuật toán loang

Đầu tiên, thuật toán đánh dấu ô đang xét bằng cách add class “check”. Sau đó xét 9 ô xung quanh từ ô đang xét ở giữa. Nếu là ô xét rồi (check) thì bỏ qua, dùng hàm checkMine để kiểm tra có mìn hay không nếu ô đang xét không có mìn mà có cờ thì thuật toán sẽ gỡ cờ và làm đệ quy tiếp. Nếu có số thì điền vào.

Hàm kiểm tra ô đang chọn có phải là mìn hay không (checkMine):



Hình III.1.2.2 Hàm kiểm tra mìn

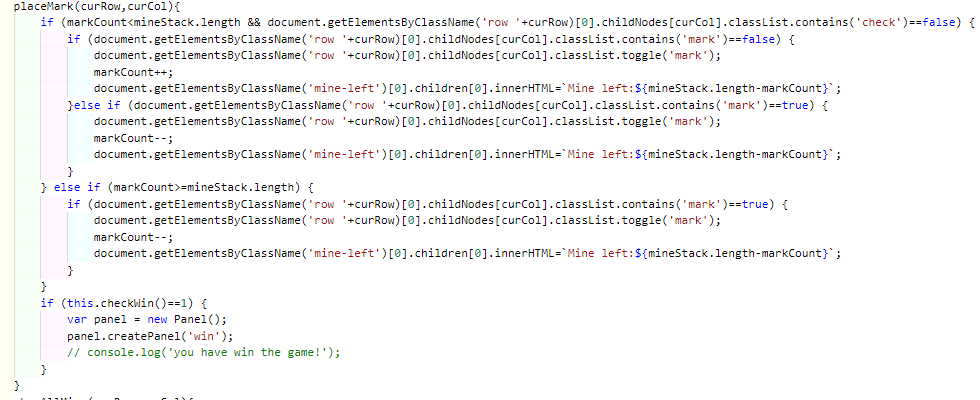
Hàm này kiểm tra ô đang xét có phải là mìn hay không (giá trị -1) sau đó kiểm tra các ô xung quanh, sau đó trả về ô đang xét giá trị tùy trường hợp:

+ Trả về -1 nếu ô đang xét là mìn.

+ Trả về 0 nếu xung quanh và ô đó không có mìn.

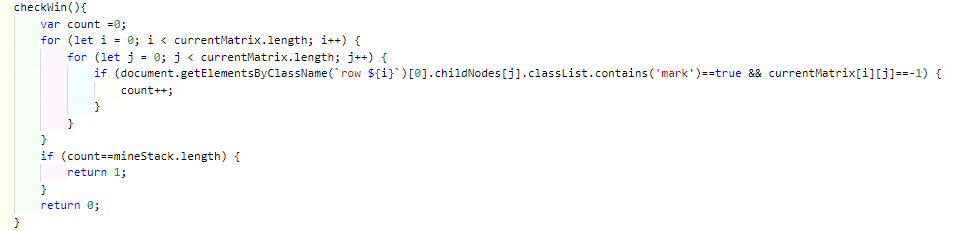
+ Trả về 1 đến 8 tùy số lượng mìn xung quanh.

Ngoài ra, chúng tôi còn viết thêm hàm đặt cờ để người chơi có thể đánh dấu những ô nghi là mìn:



Hình III.1.2.3 Hàm đặt cờ

Hàm này đơn giản chỉ là hàm đặt cờ, người chơi có thể đặt cờ bằng chuột phải, nếu có n trái mìn thì chỉ có đúng n cờ để cắm. Nếu ô nào đã lật thì không cho phép cắm cờ nữa. Cuối cùng, khi người chơi đã cắm cờ đúng vào vị trí tất cả các quả bom thì hàm kiểm tra điều kiện thắng checkwin sẽ trả về giá trị 1 và người chơi sẽ thắng, còn không thì trả về 0 và tiếp tục chơi game. Dưới đây là hàm checkwin:

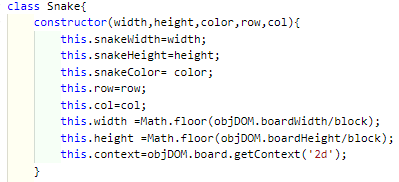


Hình III.1.2.4 Hàm checkwin

##### 1.3. Game Snake

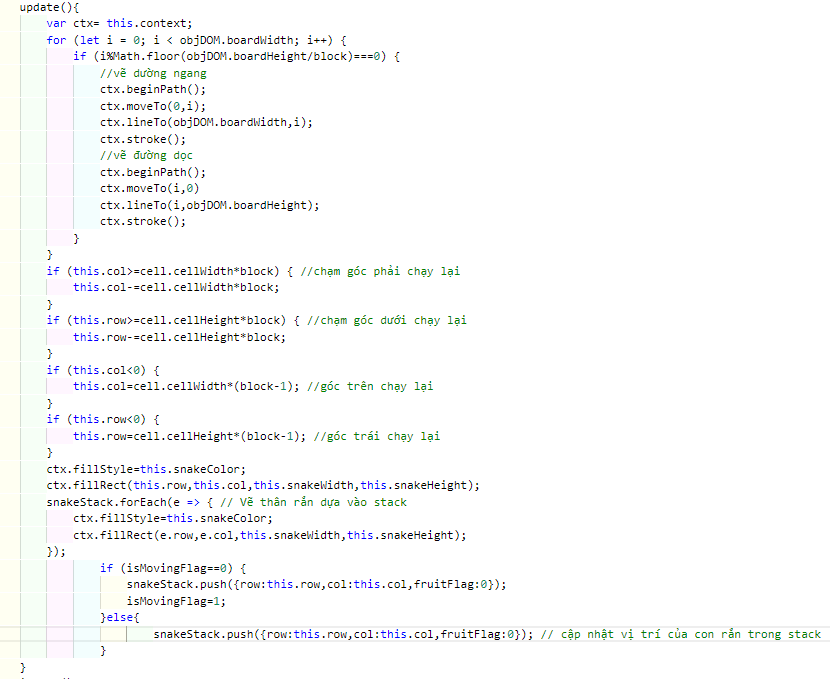
###### **1.3.1. Khởi tạo rắn và di chuyển**

Ở trong JavaScript cũng có class và constructor tương tự C#, nên chúng tôi sử dụng hàm constructor để tạo hướng đối tượng là rắn:



Hình III.1.3.1.1 Hàm khởi tạo rắn

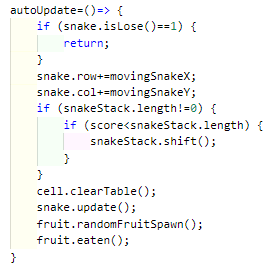
Để làm cho con rắn di chuyển, chúng tôi dùng canvas để vẽ (tương tự hàm graphic trong C#:



Hình III.1.3.1.2 Hàm update- cập nhật di chuyển của rắn

Chúng tôi dùng hàm update để vẽ sự di chuyển của rắn và vẽ lại bàn cờ( vì khi update thì canvas sẽ xóa hết mọi thứ được vẽ trước đó), nên dòng for đầu tiên là để vẽ lại bàn cờ. Bốn dòng If tiếp theo dùng để điều chỉnh hướng cho rắn đi nếu chạm phải biên, ví dụ nếu tới biên trái thì nó sẽ được chuyển sang biên phải. Sau đó là hàm vẽ rắn từ snakeStack ( mảng chứa vị trí của từng thân rắn). Cứ mỗi lần di chuyển thì snakeStack sẽ bỏ vị trí ô tiếp theo vào stack và push vị trí cuối cùng ra ngoài.

Sau đó, chúng tôi dùng hàm autoupdate dùng để xóa tất cả mọi thứ trên canvas và vẽ lại bằng cách gọi hàm update ở trên. Mỗi khi di chuyển thì cập nhật lại đầu rắn và gọi lại hàm update:



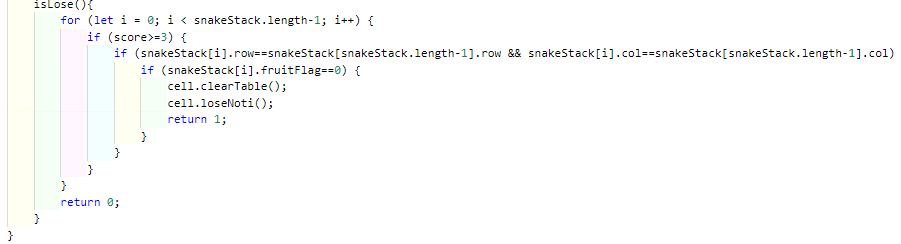
Hình III.1.3.1.3 Hàm autoupdate

Ngoài ra để hàm tự cập nhật thì ta cần phải đặt thêm một timer để tự xóa và tự vẽ:



Hình III.1.3.1.4. Timer

Trong quá trình chơi, nếu người chơi vô tình điều khiển con rắn chạm phải thân mình thì trò chơi sẽ kết thúc. Để kiểm tra, chúng tôi dùng hàm isLose:

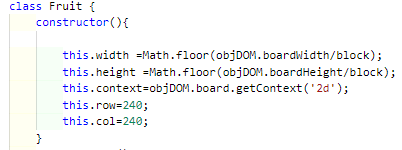


Hình III.1.3.1.5 Hàm isLose

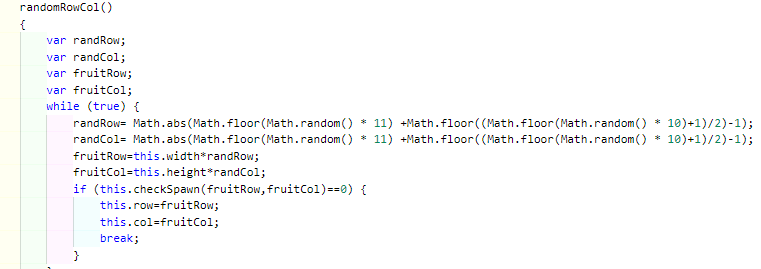
Hàm này sẽ kiểm tra từ đuôi lên đầu của rắn, nếu ở một vị trí nào đó trùng với đầu của rắn thì xóa hết canvas và báo thua còn không trả về 0 và làm tiếp.

###### **1.3.2. Tạo thức ăn cho rắn**

Tương tự như class Snake, ta dùng hàm constructor để khởi tạo một class có tên là Fruit:

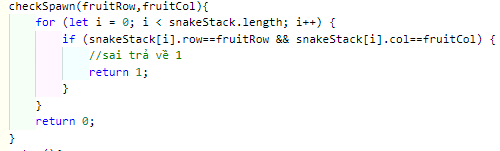


Hình III.1.3.2.1 Class Fruit

Để tạo fruit ở một ô trống ngẫu nhiên, ta dùng hàm randomrowcol:

Hình III.1.3.2.2 Hàm tạo fruit

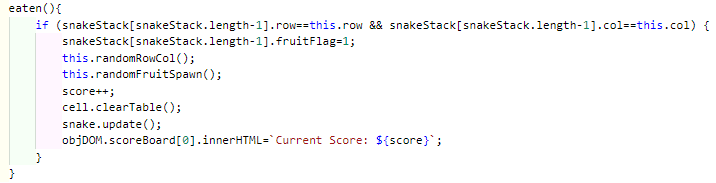
Tuy nhiên để ngăn cho fruit xuất hiện trùng vào rắn thì ta cần kiểm tra bằng hàm CheckSpawn:



Hình III.1.3.2.3 Hàm CheckSpawn

Hàm này sẽ kiểm tra chỗ vừa sinh ra có bị trùng với vị trí hiện tại của rắn hay không. Nếu đúng thì trả về 0, sai thì trả về 1.

Khi rắn ăn một quả thì một quả khác sẽ được sinh ra, để làm điều này, chúng tôi dùng hàm eaten():

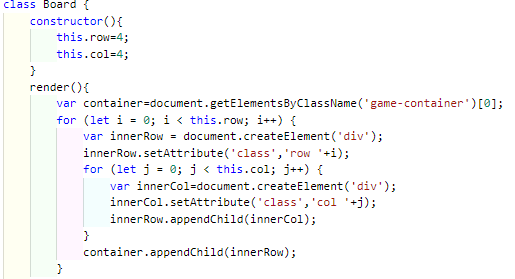


Hình III.1.3.2.4 Hàm eaten

Khi rắn chạm đến vị trí của fruit hiện tại thì cờ fruitflag được bật lên để đánh dấu là đã ăn. Sau đó hàm này sẽ gọi hàm vẽ để cập nhật vị trí fruit mới.

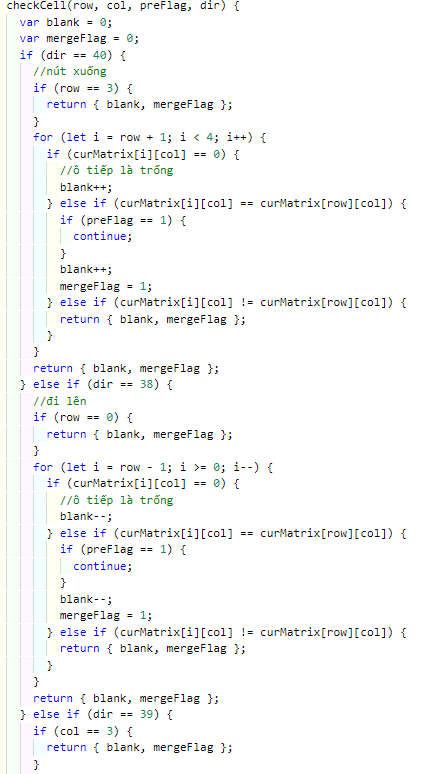
##### 1.4. Game 2048

Để tạo giao diện bảng cho game hoạt động, thay vì phải viết trên html thì chúng tôi chạy hàm render(), đảm nhận nhiệm vụ vẽ bảng:

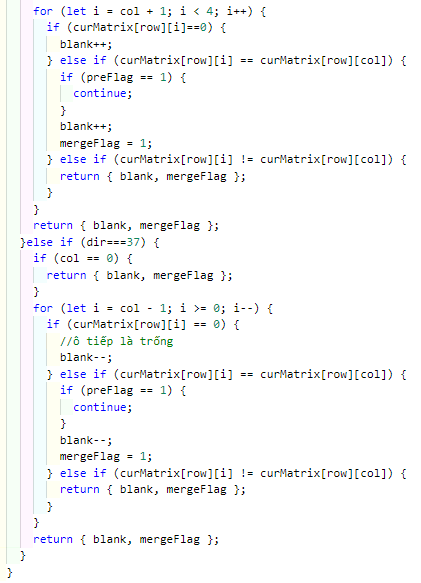


Hình III.1.4.1 Hàm Render

Tiếp theo là hàm checkCell:



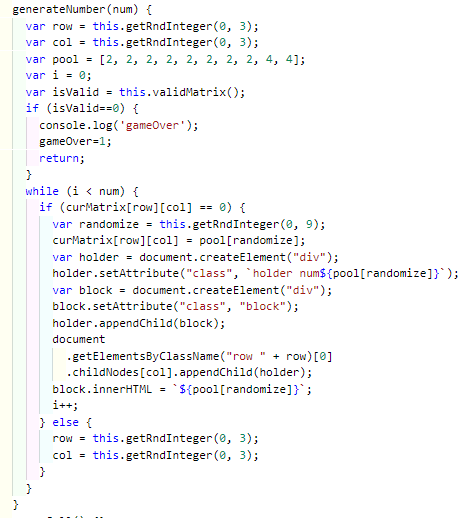
Hình III.1.4.2 Hàm checkcell-phần 1



Hình III.1.4.3 Hàm checkcell-phần 2

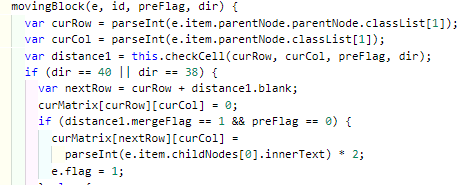
Hàm này để tính khoảng trống của ô đang xét với hướng di chuyển, có bốn hướng di chuyển thì sẽ có bốn trường hợp phải xét. Ví dụ hướng xuống thì ta sẽ xét ô cuối cùng (Row=3) lên tới ô đầu tiên. Chúng tôi chia trường hợp, lấy ô đang hiện tại, xét xem ở dưới có ô nào không, nếu không thì tăng ô blank (blank là biến đếm khoảng cách tính theo ô) và merge= 0, nếu có ô thì xét xem giá trị nó có bằng ô đang xét không, nếu bằng thì đánh dấu cờ merge rồi trả về, còn không bằng nhau thì blank không đếm nữa và trả về blank đã đếm được trước đó cùng với mergeFlag=0.

Tiếp theo là hàm generateNumber dùng để tạo số ngẫu nhiên:

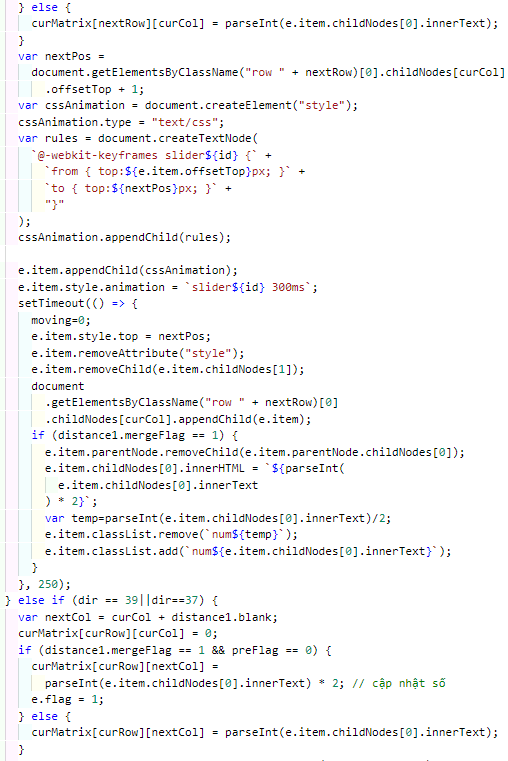


Hình III.1.4.4 Hàm generatenumber

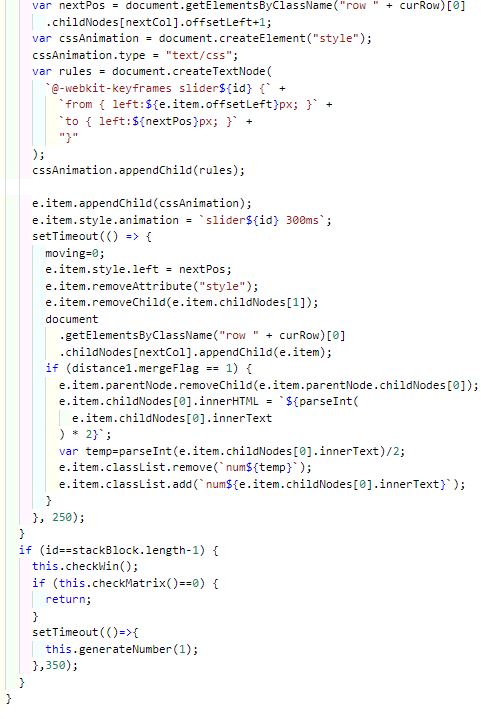
Hàm này dùng để sinh hai số ngẫu nhiên 2 hoặc 4 theo tỉ lệ 80% số 2 và 20% số 4. Nếu tạo số mới trùng vào ô đã có số rồi thì sẽ cho random lại. Ngoài ra hàm còn kiểm tra xem nếu không thêm số được nữa thì sẽ báo thua.

Tiếp theo là hàm movingBlock:

Hình III.1.4.5 Hàm movingBlock-Phần 1

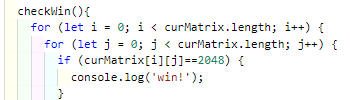


Hình III.1.4.6 Hàm movingBlock-phần 2



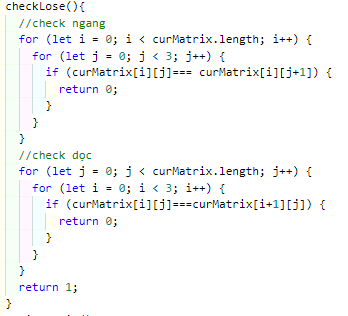
Hình III.1.4.7 Hàm movingBlock-phần 3

Đây là hàm quan trọng nhất của game vì nó phải xử lý trên giao diện lẫn xử lý ma trận trong mảng. Hàm phải vừa phân tích số vừa duy chuyển theo hướng của nút chỉ thị, ngoài ra hàm còn gộp ô khi hai ô có cùng số. Dựa vào hàm checkcell để lấy được ô trống và cờ merge, nếu có cờ merge thì nó sẽ lấy ô đang xét cho bằng 0 và sửa ô cùng giá trị với nó nhân 2. Nếu không có cờ merge thì chỉ dời ô đang xét đến blank, rồi làm giống vậy cho 3 trường hợp các hướng còn lại. Cuối hàm là một hàm kiểm tra điều kiện thắng, hàm checkWin:



Hình III.1.4.8 Hàm checkWin

Người chơi sẽ thắng khi gộp được số 2048 và sẽ thua khi không còn đường đi nào nữa:



Hình III.1.4.9 Hàm checkLose

#### *2. Thiết kế lớp*

##### 2.1. Danh mục các lớp được sử dụng trong chương trình

###### **2.1.1. Game 2048**

Sinh viên phụ trách: Lê Minh Tiến

|  |  |  |
| --- | --- | --- |
| Thứ tự | Tên lớp | Mục đích |
| 1 | Board | Quản lí thao tác trên game |
| 2 | Matrix | Tạo bàn chơi cho game |
| 3 | Panel | Tạo bảng thông báo |

###### **2.1.2. Game Snake**

Sinh viên phụ trách: Bùi Ngọc Bảo

|  |  |  |
| --- | --- | --- |
| Thứ tự | Tên lớp | Mục đích |
| 1 | Fruit | Khởi tạo thức ăn cho rắn |
| 2 | Snake | Khởi tạo rắn |

###### **2.1.3. Game dò mìn**

Sinh viên phụ trách: Bùi Ngọc Bảo

|  |  |  |
| --- | --- | --- |
| Thứ tự | Tên lớp | Mục đích |
| 1 | Matrix | Khởi tạo ma trận và quy định số mìn |
| 2 | Board | Khởi tạo bàn chơi |
| 3 | Panel | Hiện bảng thông báo |

##### 2.2. Danh mục các phương thức trong một lớp

###### **2.2.1. Game 2048**

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên phương thức | Mục đích | Tên file, số thứ tự dòng chứa khai báo |
| 1 | checkWin() | Kiểm tra điều kiện thắng của game | 2048/src/js/index.js (9) |
| 2 | checkLose() | Kiểm tra điều kiện thua | 2048/src/js/index.js (18) |
| 3 | generateNumber(num) | Tạo số ngẫu nhiên giữa 2 và 4 | 2048/src/js/index.js (79) |
| 4 | mergeCell() | Hợp 2 ô nếu cùng số | 2048/src/js/index.js (110) |
| 5 | checkCell() | Tính khoảng trống ô đang xét với hướng di chuyển | 2048/src/js/index.js (170) |
| 6 | movingBlock(e,id,preFlag,dir) | Phân tích số, di chuyển theo nút chỉ thị và gộp 2 ô | 2048/src/js/index.js (253) |
| 7 | createPanel(gameCondition) | Hiện bảng thông báo nếu thắng hoặc thua | 2048/src/js/Panel.js (6) |
| 8 | startGame() | Bắt đầu trò chơi | 2048/src/js/Panel.js (41) |

###### **2.2.2. Game Snake**

|  |  |  |  |
| --- | --- | --- | --- |
| Tt | Tên phương thức | Mục đích | Tên file, số thứ tự dòng khai báo |
| 1 | randomRowCol() | Đặt vị trí ngẫu nhiên cho fruit | snake/js/Fruit.js(11) |
| 2 | checkSpawn(fruitRow,fruitCol) | Kiểm tra vị trí có trùng với rắn không | snake/js/Fruit.js(39) |
| 3 | eaten() | Kiểm tra fruit có bị ăn bởi hay chưa | snake/js/Fruit.js(48) |
| 4 | update() | Vẽ thân rắn và cập nhật vị trí nếu chạm biên | snake/js/Snake.js(12) |
| 5 | isLose() | Kiểm tra nếu đầu rắn đụng thân | snake/js/Snake.js(53) |

###### **2.2.3. Game dò mìn**

|  |  |  |  |
| --- | --- | --- | --- |
| Tt | Tên phương thức | Mục đích | Tên file, số thứ tự dòng khai báo |
| 1 | checkMine(curRow,curCol) | Kiểm tra mìn tại ô đang xét | MineSweeper/js/index.js(5) |
| 2 | spreadCell(curRow,curCol) | Thuật toán loang đến các ô có số thì dừng | MineSweeper/js/index.js(27) |
| 3 | checkWin() | Kiểm tra điều kiện thắng | MineSweeper/js/index.js(58) |
| 4 | placeMark(curRow,curCol) | Đặt cờ | MineSweeper/js/index.js(72) |
| 5 | showAllMine(curRow,curCol) | Hiện vị trí của các mìn còn lại | MineSweeper/js/index.js(96) |
| 6 | generateMatrix() | Khởi tạo ma trận | MineSweeper/js/Matrix.js(10) |
| 7 | placeMine() | Đặt mìn ngẫu nhiên | MineSweeper/js/Matrix.js(18) |
| 8 | createPanel(gameCondition) | Hiện bảng thông báo nếu thắng hoặc thua | MineSweeper/js/Panel.js(10) |
| 9 | levelSelect() | Chọn mức độ khó dễ | MineSweeper/js/Panel.js(42) |

## IV. Kết luận

### 1. Các khó khăn gặp phải

Do khối lượng công việc nhiều và thời gian gấp rút, nên khi viết về game Sudoku, nhóm đã lấy code của một bạn thành viên trong nhóm khi bạn ấy làm đề tài cho môn Cấu trúc dữ liệu và giải thuật. Vì thế cho nên game Sudoku chưa thực sự được viết lại theo hướng đối tượng.

### 2. Ưu điểm

Vì được lập trình trên nền tảng Web (HTML, CSS, JavaScript) vì thế nên có thể dễ dàng chơi ở bất cứ mọi nơi, bất cứ thiết bị điện tử nào (máy tính bàn, Laptop, Điện thoại thông minh…) mà không cần phải cài đặt thêm chương trình nào.

Các game trong đây đều được biết rộng rãi trên toàn thế giới nên không cần phải viết thêm hướng dẫn cách chơi cho từng game. Ngoài ra, với giao diện đơn giản, trực quan sẽ giúp người chơi điều hướng, lựa chọn game mình muốn chơi dễ hơn.

### 3. Nhược điểm

Không thể chơi game nếu không có kết nối mạng với thiết bị điện tử của mình.

Code chưa được tối ưu hóa nên còn ngốn nhiều bộ nhớ của máy.

Hiện tại game chỉ lưu tên người chơi, bảng xếp hạng điểm trên Local Storage của máy tính cá nhân.

### 4. Hướng phát triển

Tối ưu hóa thuật toán để chạy nhanh và mượt hơn

Thêm nhiều game mới như Caro, Chess...

Tạo máy chủ để lưu trữ thông tin người chơi, điểm số để họ có thể tìm thông tin của mình ở bất cứ đâu và so sánh điểm của mình với những người chơi khác.

# TÀI LIỆU THAM KHẢO

1. https://freefrontend.com/index.html

2. https://www.w3schools.com/

3.

https://freefrontend.com/css-cards/?fbclid=IwAR21D5V\_UCMiAwMMGarjr3de3BZHb4Ff7XVybGEe2N8rumBmMzm9NsyeXSg