ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

ĐỒ ÁN MÔN CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

Đề tài: Trò chơi Ô ăn quan

(Đã chỉnh sửa)

Giảng viên hướng dẫn:

Ths. Huỳnh Thị Thanh Thương

Sinh viên thực hiện:

Lê Ngọc Thành	**15520810**
Lê Võ Gia Khang	**15520341**

Mục lục

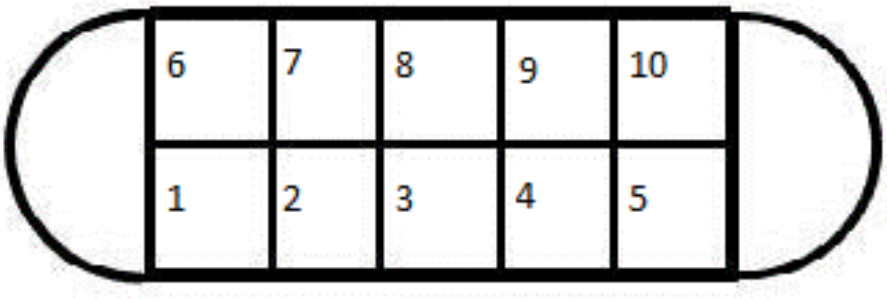
Phần 1. PHÁT BIỂU VẤN ĐỀ BÀI TOÁN

- Tên bài tập: Xây dựng trò chơi ô ăn quan cho 2 người chơi.

- Từ khóa liên quan: Danh sách liên kết đôi, danh sách liên kết vòng.

1.1 Mô tả

- Hình thực tế của trò chơi ô ăn quan.



- Khởi đầu
  - 2 đầu (hai nửa vòng tròn) là nơi chứa quan. Mỗi ô 1 quan.
  - Mỗi ô vương chứa 5 dân.
  - Hướng L (trái): di chuyển từ 1 sang 5 hoặc 6 sang 10.
  - Hướng R (phải): di chuyển từ 5 sang 1 hoặc 10 sang 6.
- Mô hình
  - [1,2,3,4,5]<Quan 1>[10,9,8,7,6] <Quan 2> (Số quân của người 1 || Số quân của người 2)
- Di chuyển và ăn quân
  - Người chơi lựa chọn 1 trong 5 ô vuông (có chứa dân ở phía người chơi) rải quân theo chiều kim hoặc ngược chiều kim đồng hồ tùy ý.
  - Rải mỗi ô 1 dân tính từ ô gần nhất. Nếu rải hết quân cuối cùng:
    - Nếu ô liền sau là ô vuông chứa dân (nếu liền sau là ô chứa quan thì dừng việc rải quân lại) thì chọn tiếp ô đó để rải theo hướng đã lựa chọn lúc ban đầu.
    - Nếu ô liền sau là ô trống (không phân biệt ô quan hay ô dân) rồi đến một ô chứa dân hoặc quan thì người chơi ăn tất cả tại ô có chứa dân hoặc quan. Nếu liền sau ô bị ăn quân lại tiếp tục là

ô trống và sau là một ô có chứa dân hoặc quan thì sẽ tiếp tục bị ăn… tiếp tục quá trình ăn quân đến khi sau ô đang xét không còn trạng thái là ô trống và sau là ô chứa dân hoặc quan.

- Nếu tới lượt người chơi mà tất cả ô vuông (5 ô) phía người chơi không chứa dân thì người phải rải mỗi ô 1 quân – số quân được rải sẽ trừ vào số quân người chơi đã ăn được.
- Trò chơi kết thúc khi 2 ô chứa quan trống hoặc 1 trong 2 người chơi không đủ dân để rải trên 5 ô vuông trống của người chơi đó.

Xây dựng một danh sách liên kết để thực hiện được các yêu cầu của trò chơi.

- Yêu cầu nhập xuất
  - Quy định 1 quan = 10 quân.
  - Mô hình khởi đầu trên máy tính dạng sau:

    [5, 5, 5, 5, 5] <10> [5, 5, 5, 5, 5] <10> (0 || 0)
    - Trong đó, ô trong cặp ngoặc <> là ô chứa quan
    - Mỗi phần tử trong cặp ngoặc [ ] là ô chứa dân và thuộc mỗi người chơi.
    - Mỗi phần tử trong cặp ngoặc ( ) là số quân ăn được mỗi người chơi.
    - Ví dụ: Người chơi 1 lựa chọn ô 1 (trái sang phải) hướng từ trái sang phải kết thúc sẽ như sau:

      [0, 0, 6, 6, 6] <11> [0, 6, 6, 6, 6] <11> (6 || 0)

    - Ngoài ra, danh sách lưu lại các thao tác của 2 người chơi [người chơi thứ i].[ô thứ j trong 5 ô của người đó].[di chuyển sang trái hay hay phải](số quân ăn được. (trái ngược chiều kim đồng hồ - phải thuận chiều kim đồng hồ).
    - Input và output lần lượt:

- 3P
    - [6, 6, 0, 0, 6] <11> [6, 6, 0, 6, 6] <11> (6 || 0)
- 7P
    - [6, 0, 0, 1, 7] <12> [7, 7, 1, 0, 6] <11> (6 || 6)
- 5P
    - [2, 3, 3, 4, 0] <0> [0, 0, 4, 3, 9] <14> (22 || 6)

➢ Hướng dẫn cơ bản

- Xây dựng một danh sách liên kết vòng – liên kết đôi. Quy định vị trí ô quan cụ thể và ô dân thuộc người chơi nào. Khởi tạo như đề bài.
- Xây dựng thủ tục:
    - Rải quân: cho 2 hướng thuận và ngược chiều kim đồng hồ. Lưu ý: vấn đề rải quân liên tục theo yêu cầu người chơi. Có thể xây dựng một thủ tục phụ để kiểm tra trạng thái dừng rải quân.
    - Ăn quân: là thủ tục kèm trong rải quân. Có thể xây dựng một thủ tục kiểm tra trạng thái có ăn quân được hay không.
    - Kiểm tra trạng thái dừng trò chơi (có thể để trong thủ tục ăn quân).
        - Cả 2 ô quan đều không có dân.
        - 1 trong 2 người không còn đủ 5 dân và những ô dân trên bàn cờ của họ trống.
        - Và so sánh số dân của mỗi người để kết luận người chiến thắng. Lưu ý khi kết thúc thì những dân trên

những ô của mỗi người chơi sẽ thu về mỗi người chơi đó để so sánh.

Kiểm tra tới lượt

- Nếu tới lượt người chơi mà 5 ô dân trên bàn trống thì rải mỗi ô 1 dân (trừ vào số dân của người chơi đó).
- Kết hợp thủ tục và kiểm tra lại trò chơi.

➢ Mở rộng

- Xây dựng trò chơi ô ăn quan cho 3 người chơi, 4 người chơi. Bổ sung các tính năng tương tự quay lại…
- Xây dựng một thuật giả đơn giản nhằm để chiến thắng như kiểm tra ăn quân nhiều nhất, hạn chế đối phương ăn quân ít nhất…

1.2 Mô hình hóa

Phần 2. THIẾT KẾ CẤU TRÚC DỮ LIỆU.

Ý tưởng: tạo ra một bàn cờ Ô ăn quan cho 2 người chơi. Bàn cờ có tổng cộng là 10 ô dân và 2 ô quan. Với mỗi ô tương ứng sẽ có số dân và quan trong đó. Như vậy đầu tiên ta sẽ tạo ra 1 ô, sau đó ta tiến hành liên kết các ô đó với nhau tạo thành 1 bàn cờ. Trong quá trình chơi thì điểm số của người chơi và số dân, quan trong mỗi ô sẽ thay đổi. Do đó mỗi ô tương ứng với 1 node có các chỉ số thể hiện số dân và quan trong đó còn bàn cờ sẽ là 1 danh sách liên kết bao gồm các node đã được tạo sẵn trước đó.

Thiết kế: Gồm 2 phần là Node và Danh sách liên kết đôi:

```
struct Node
{
    int dan, quan;
    int laquan;
    Node *pre, *next;
};
```

```
struct List
{
	Node *pHead;
	Node *pTail;
};
```

Node bao gồm các dữ liệu (số dân, số quan, phải là ô quan hay không), con trỏ Next, Pre để liên kết với các Node kế cận nó.

Danh sách (List) bao gồm 2 con trỏ Node là pHead và pTail quản lý phần tử đầu và phần tử cuối.

Mỗi ô dân,quan trong trò chơi là 1 Node tương ứng. Danh sách liên kết đôi thể hiện trò chơi bao gồm các Node đã có sẵn với dữ liệu tương ứng của nó.

Ví dụ:

Khi bắt đầu trò chơi, ta tiến hành định nghĩa các dữ liệu như struct Node và struct List, sau đó ta tiến hành khởi tạo 1 bàn cờ (danh sách) bằng câu lệnh : List l; trong hàm main. Sau đó tiến hành khởi gán các giá trị của danh sách thông qua hàm CreateList(l);

Ở hàm này, ta sẽ gọi lại hàm Init(l) đã có trước đó để gán cho pHead và pTail của l có giá trị là null. Kế tiếp, ta chạy 1 vòng lặp for từ 1 đến 12 để tạo ra 12 node. Ở node thứ nhất và node thứ 7 (ta tạm xem nó là node quan) thì ta sẽ gọi hàm GetNode(0,10,1) để tạo ra node có giá trị dân, quan, và phải là quan hay không lần lượt là 0,10 và 1. Tại những node còn lại ta gọi hàm GetNode(5,0,0) để giá trị dân là 5, quan là 0 và ô đó k phải là ô quan. Sau đó ta tiến hành thêm node đó vào danh sách đã tạo thông qua hàm AddHead(l,p) để thêm node đó vào đầu danh sách. Cuối cùng ta cho pHead->pre = pTail và pTail->next = pHead để hoàn thành 1 danh sách liên kết vòng.

```
void CreateList(List &l)
{
	Init(l);
	for (int i = 1; i <= 12; i++)
	{
		if (i == 1 || i == 7) // vị trí quan là 1,7
```

```
		{
			Node *p = new Node;
			p = GetNode(0, 10, 1);
			AddHead(l, p);
		}
		else	//những vị trí không phải là quan
		{
			Node *p = new Node;
			p = GetNode(5, 0, 0);
			AddHead(l, p);
		}
	}
	l.pHead->pre = l.pTail;
	l.pTail->next = l.pHead;
}
```

Phần 3. THIẾT KẾ GIẢI THUẬT.

3.1 Ý tưởng

Bài toán tổng quát:

Tạo ra 1 bàn cờ với các ô dân. Mỗi ô dân có các chỉ số dân, quan, là quan hay không.

Bàn cờ là 1 danh sách liên kết đôi, vòng được cấu thành từ các node đã tạo sẵn các giá trị và tiến hành đưa các node đó vào danh sách.

Đầu tiên ta tạo các cấu trúc dữ liệu như phần 2 đã nêu. Kế tiếp ta thực hiện các bước sau:

B1. Khởi tạo bàn cờ:

List l;

B2. Khởi tạo các biến điều khiển trò chơi:

int pos1 = 0, pos2 = 0; //pos là vị trí ô mà người chơi sẽ chọn trong quá trình chơi, pos1 là vị trí ô người chơi 1 chọn, pos2 là của người chơi 2 chọn

int clockwise_P1 = 0, clockwise_P2 = 0; //clockwise là hướng mà người chơi chọn sẽ di chuyển quân theo trái hay phải

char way; //way là kí tự mà người chơi nhập vào, từ đó sẽ chuyển thành clockwise để chương trình hiểu và chạy được

int playagain = 1; //playagain là biến để sau khi kết thúc trò chơi thì người chơi sẽ nhập vào để quyết định xem có chơi lại hay không

B3. Tạo ra 1 vòng lặp while chứa biến playagain để thực hiện lại chương trình trong khi người chơi muốn chơi lại:

While(playagain == 1) { … }

B4. Khởi tạo các biến điều khiển khác:

int count1 = 0, count2 = 0; //count là biến để tính điểm người chơi sau khi họ đã ăn được quân và sẽ cộng dồn đến cuối trò chơi để quyết định xem ai là người thắng cuộc

bool pauseP1 = false, pauseP2 = false; //pause là biến để xác định xem người chơi đó có còn đủ điều kiện để tiếp tục chơi nữa hay không

B5. Khởi tạo bàn cờ:

CreateList(l); // tạo ra bàn cờ *Vấn đề 1

Output(l, count1,count2); //in ra bàn cờ

B6. Tạo 1 vòng lặp for:

for (int turn = 1; !isStoped(l) && !pauseP2 && !pauseP1; turn++) {…}

biến turn là để chỉ lượt của người chơi, với turn là số lẻ thì lượt của người chơi 1, ngược lại là người chơi 2.

Điều kiện dừng là khi 2 ô quan trống hoặc điều kiện chơi của 1 trong 2 người chơi không thỏa thông qua các hàm và biến:

isStoped(l) //kiểm tra xem 2 ô quan có trống hay không *Vấn đề 4

pauseP1, pauseP2 trình bày ở B4

Sau đó tăng biến turn để tới lượt người chơi khác.

B7. Lượt người chơi tương ứng:

Cả 2 người chơi đều có các hàm tương tự nhau để thực hiện trò chơi:

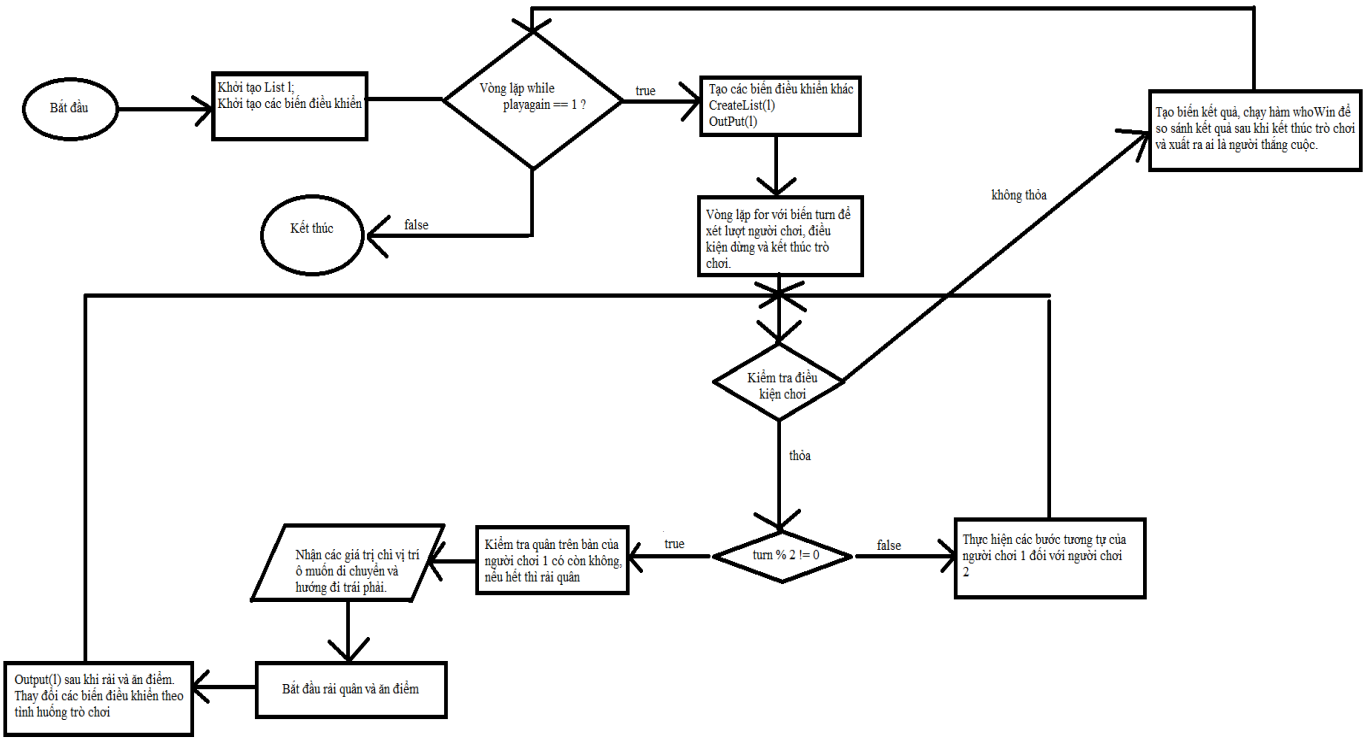
1. CheckToDistribute_P1 hoặc CheckToDistribute_P2 để kiểm tra xem số quân trên bàn của người chơi 1 hoặc người chơi 2 có lớn hơn 0 hay không, nếu số quân trên bàn đã không còn thì thực hiện rải quân: Distribute_P1 hoặc Distribute_P2.
2. Tiếp đến ta tiến hành nhận pos1 hoặc pos2 từ bàn phím do người chơi quyết định và lấy node tại vị trí đó thông qua hàm SetNodeFromPos.
3. Kế tiếp, ta nhận giá trị hướng đi từ bàn phím của người chơi (right hoặc left) từ kí tự đó ta chuyển thành số để xử lý (right là 1, left là 2).
4. Thực hiện rải quân và ăn điểm thông qua hàm Distribute_AndScore (*Vấn đề 2,3)
5. Sau đó in ra lại bàn cờ và chỉnh sửa lại pause nếu cần thiết.

B8. Ghi nhận điểm số và xuất ra người chơi thắng cuộc cùng với điểm số:

1. Tạo ra biến result để ghi nhận ai thắng cuộc thông qua hàm whoWin. Ở hàm này có tham số là điểm số của 2 người chơi, sau khi xét số quân trên bàn của 2 người chơi thì cộng dồn nó vào điểm số ban đầu. Sau đó đem so sánh điểm số đã được cộng với nhau và xuất ra người chơi nào thắng cuộc.

B9. Ghi nhận biến playagain và thực hiện lại vòng lặp nếu có.

Sơ đồ:



3.2 Giải thuật

Vấn đề 1: Khởi tạo bàn cờ

1. Phát biểu bài toán: Khởi tạo 1 bàn cờ gồm 10 ô dân và 2 ô quan. Ở mỗi ô dân có số quân là 5, tại ô quan có số quân là 10. Khởi đầu 10 ô dân đều phải được rải đầy đủ 5 quân và ô quan phải có 10 quân.

   Input: 1 danh sách trống

   Output: 1 danh sách đã có đầy đủ các thông tin trò chơi.
2. Ý tưởng giải quyết: Tạo ra 1 danh sách rỗng. Tạo ra các node tương tự nhau có các chỉ số là dân, quan và là quan. Sau khi tạo node tiến hành thêm các node đó vào danh sách đã có. Tiến hành tạo liên kết vòng.
3. Thuật toán:

   3.1 Tạo danh sách rỗng l: Init(l); ở hàm này ta cho pHead = pTail = Null

3.2 Tạo vòng lặp for chạy từ 1 đến 12: (12 node)

```
for (int i = 1; i <= 12; i++)
    {       do something;       }
```

3.3 Câu lệnh xử lý:

```
    if (i == 1 || i == 7) // vị trí quan là 1,7
            {
                    Node *p = new Node;
                    p = GetNode(0, 10, 1);
                    AddHead(l, p);
            }
            else //những vị trí không phải là quan
            {
                    Node *p = new Node;
                    p = GetNode(5, 0, 0);
                    AddHead(l, p);
            }
```

Ở đây ta tạo 1 node p là node tạm, sau đó p được gán bằng giá trị của hàm GetNode.

Ở hàm GetNode ta tiến hành tạo ra 1 node mới với các giá trị truyền vào:

```
Node* GetNode(int _dan, int _quan, int _laquan)
{
    Node *p = new Node;
    if (p == NULL)
    {
            cout << "Error.The memory is full" << endl;
            return p;
    }

    p->dan = _dan;
    p->quan = _quan;
    p->laquan = _laquan;
    p->next = p->pre = NULL;
    return p;
}
```

Sau đó ta thêm node đã tạo vào đầu danh sách thông qua hàm AddHead:

```
void AddHead(List &l, Node *pnew)
{
    if (l.pHead == NULL)
            l.pHead = l.pTail = pnew;
    else
    {
            l.pHead->pre = pnew;
            pnew->next = l.pHead;
            l.pHead = pnew;
    }
}
```

3.4 Sau khi thực hiện xong việc tạo ra 12 node và gán nó vào danh sách thì ta tiến hành tạo liên kết vòng bằng cách cho pHead->pre = pTail và pTail->next = pHead:

```
            l.pHead->pre = l.pTail;
            l.pTail->next = l.pHead;
```

4. Minh họa:

Đầu tiên ta có danh sách rỗng l: [] <> [] <>

Ở vòng lặp thứ nhất ta tạo được 1 node có giá trị dân, quan, là quan lần lượt là 0,10,1 tức là ô đó là ô quan. Ta gọi hàm AddHead để thêm node đó vào danh sách. Sau khi thêm vào ta được: [10] <> [] <>

Ở vòng lặp thứ 2 ta tạo được 1 node có giá trị dân, quan, là quan lần lượt là 5,0,0 tức là ô đó là ô dân. Ta gọi hàm AddHead để thêm node đó vào danh sách.

Sau khi thêm vào ta được: [5,10] <> [] <>

Thực hiện tương tự như vậy đến node số 6 ta được: [5,5,5,5,5] <10> [] <>

Ở vòng lặp thứ 7 ta cũng tạo ra 1 node quan và thêm node đó vào đầu danh sách:

[10,5,5,5,5] <5> [10] <>

Cứ như vậy cho đến khi hoàn thành xong danh sách:

[5,5,5,5,5] <10> [5,5,5,5,5] <10>

Vấn đề 2: Rải quân

1. Phát biểu bài toán: Khi bắt đầu trò chơi thì số quân có sẵn trên mỗi ô của bàn cờ là 5 đối với ô dân và 10 đối với ô quan. Trong quá trình chơi, quân cờ sẽ liên tục được rải theo hướng của người chơi, kết hợp với việc ăn điểm và rải tiếp sau khi ăn điểm. Trong trường hợp số quân trên bàn không còn quân để di chuyển thì phải lấy số quân ăn được rải đều 5 ô của người chơi mỗi ô 1 quân/

   Input: hướng rải, số quân còn trên bàn cờ hiện hành.

   Output: bàn cờ sau khi thực hiện rải quân.

2. Ý tưởng giải quyết: Đầu tiên chúng ta sẽ xét xem số quân trên bàn còn hay không, nếu không còn quân nào thì ta sẽ kiểm tra số quân ăn được và tiến hành rải mỗi ô 1 quân. Nếu số quân vẫn còn thì ta xét tiếp hướng di chuyển của người chơi là trái hay phải. Ví dụ theo hướng phải thì ta tạo 1 vòng lặp, trong vòng lặp đó ta tiến hành trừ đi số quân tại ô đã chọn đi 1 và tăng số quân ở ô tiếp theo lên 1. Sau đó tạo 1 biến tạm và gán biến đó thành ô kế tiếp để thực hiện hành động tăng quân lên. Vòng lặp sẽ kết thúc khi số quân ở ô được chọn bằng 0. Sau khi thực hiện xong ở ô được chọn, ta tiến hành xét ô kế tiếp của ô cuối cùng nhận được quân ở vòng lặp trên. Trong trường hợp ô kế tiếp đó là trống thì ta sẽ xét ô kế nó. Nếu là 0 thì không ăn được quân, nếu ô đó có quân thì ta tiến hành việc ăn quân và rải lại sau khi ăn quân. Trong trường hợp ô đó không trống thì ta tiến hành việc rải quân như ban đầu và ô đó sẽ được chọn là ô hiện hành.

3. Thuật toán:

   3.1 Tạo 1 Node* temp = new Node;

   3.2 temp = p //p là node được chọn

   3.3 Xét hướng di chuyển if (dir == 1) là hướng phải ngược lại dir = 2 là bên trái

   3.4 Cho vòng lặp while (p->dan != 0)

   ```
   {
        p->dan--;
        temp = temp->next;
   ```

```
        temp->dan++;
    }
```

Ở vòng lặp này ta cho ô được chọn trừ đi số dân là 1, node temp được gán là node kế tiếp và tăng số dân node đó lên, thực hiện cho đến khi số dân trong ô được chọn bằng 0 tức là thực hiện n lần với n là số dân ở ô được chọn.

3.5 temp = temp->next //đặt node temp là node kế tiếp của ô thứ n từ ô được chọn đếm qua phải

3.6 Cho vòng lặp

```
while ((temp->dan != 0) && (temp->laquan == 0))
{
        int t = temp->dan;
        temp->dan = 0;
        while (t != 0)
        {
                t--;
                temp = temp->next;
                temp->dan++;
        }
        temp = temp->next;
}
```

Vòng lặp này ta sẽ xét đến node kế tiếp của node thứ n tính từ ô được chọn đếm qua phải, nói cho dễ hiểu là ta sẽ thực hiện lại công việc ở bước 3.4 đối với node kế tiếp của node vừa kết thúc. Để hiểu rõ hơn xem ở phần 4.

3.7 Cho vòng lặp

```
while (isScored_Next(temp))
{
        count = count + temp->next->dan + temp->next->quan;
        temp->next->dan = 0; temp->next->quan = 0;
        temp = temp->next->next;
}
```

Hàm isScored_Next là kiểm tra xem node kế tiếp có trống hay không. Nếu trống và số dân + quan ở node kế tiếp khác 0 thì trả về true tức là ăn điểm. Ngược lại thì trả về false. Nếu hàm này trả về true thì điểm được gán bằng

điểm cộng với số dân và quan ở node kế tiếp. Sau khi đã ăn được quân ở ô đó thì đặt số quân ở ô đó về 0 và xét tiếp xem có ăn được nữa hay không.

4. Minh họa:

4.1 Ví dụ ô trên bàn có sẵn là [5,5,5,5,5] <10> [5,5,5,5,5] <10> (0 || 0)

Người chơi 1 sẽ chọn node thứ 3 vậy temp sẽ là node ở ô thứ 3 từ trái sang phải. Người chơi 1 chọn hướng di chuyển sang phải. dir = 1.

4.2 while (p->dan != 0) //hiện tại p->dan đang là 5

{ p->dan--; //p->dan = 4;

temp = temp->next; //ô thứ 4;

temp->dan++; //ô thứ 4 có số dân là 6;

}

Lần 2: p->dan = 4 khác 0

{ p->dan--; //p->dan = 3;

temp = temp->next; // ô thứ 5;

temp->dan++; // ô thứ 5 có số dân là 6;

}

Tương tự vậy cho đến khi p->dan = 0

Như vậy ta có bàn cờ sau khi thực hiện 5 lần thao tác trên:

[5,5,0,6,6] <11> [6,6,5,5,5] <10> (0 || 0)

4.3 temp = temp->next; // node temp đang ở node tô đỏ.

4.4 while ((temp->dan != 0) && (temp->laquan == 0) // số quân ở node temp đang là 5, và node temp cũng không phải ô quan nên ta thực hiện câu lệnh trong vòng lặp:

```
Tạo int t = temp->dan; // i = 5;

temp->dan = 0;

while (t != 0)
{
        t--;
        temp = temp->next;
        temp->dan++;
}
```

Ở dòng lặp này ta thực hiện tương tự ở bước 4.2

Sau khi chạy xong vòng này ta được:

[6,6,0,6,6] <11> [6,6,0,6,6] <11> (0 || 0)

Node temp đang ở ô màu đỏ.

4.5 while (isScored_Next(temp)) // ở hàm này ta xét số dân trong ô temp có phải là 0 hay không. Nếu là 0 thì xét ô tiếp theo số quân có khác 0 hay không. Nếu khác 0 thì ta thực hiệm các câu lệnh trong vòng while. Mọi trừ hợp còn lại đều không thực hiện:

```
count = count + temp->next->dan + temp->next->quan; //count được tăng từ 0 lên 6 vì ở ô tiếp theo có 6 quân và ô temp có 0 quân.

temp->next->dan = 0;  //đặt lại số dân ở ô kế tiếp bằng 0
temp->next->quan = 0; // đặt lại số quan ở ô kế tiếp bằng 0
temp = temp->next->next; // gán node temp là node kế tiếp của node kế tiếp
```

xem có thể thực hiện ăn liên tiếp được hay không và thực hiện lại vòng lặp.

Sau khi thực hiện xong bước 4.5 ta có bàn cờ như sau:

[6,6,0,0,6] <11> [6,6,0,6,6] <11> (6 || 0)

Như vậy ta đã giải quyết xong trường hợp rải quân kết hợp với kiểm tra và ăn điểm.

Vấn đề 3: Ăn quân

1. Phát biểu bài toán: Trong quá trình chơi, người chơi sẽ ăn quân khi rơi vào trường hợp sau: sau khi rải lần lượt 1 quân ở những ô kế tiếp theo hướng rải cho đến hết quân, nếu ô kế tiếp ô kết thúc đó là 1 ô trống và ô kế tiếp nữa có số quân khác 0 thì người chơi sẽ được ăn trọn số quân ở ô đó. Trong trường hợp ô kế tiếp của ô vừa ăn là 1 ô trống và ô kế tiếp nữa có số quân khác 0 thì người chơi sẽ được ăn tiếp ô đó.

   Input: ô đang xét

   Output: có ăn được không, số quân ăn được
2. Ý tưởng giải quyết: xét số quân ở node đang xét. Nếu nó = 0 thì kiểm tra xem số quân ở node kế tiếp có lớn hơn 0 hay không, nếu lớn hơn 0 thì trả về true, những trường hợp còn lại trả về false.
3. Thuật toán: Ở đây ta xây dựng 2 hàm là isScored_Next và isScored_Pre để xét theo 2 hướng trái hoặc phải. Lấy ví dụ là hàm isScored_Next để minh họa thuật toán, hàm còn lại chỉ khác là con trỏ sẽ trỏ về node phía trước thay vì node phía sau như hàm này.

   3.1
```
if (p->dan == 0 && p->laquan == 0 &&
    ((p->next->dan + p->next->quan)>0))
        return true;
```

   Nếu số dân ở node đang xét bằng 0 và node đó không phải là node quan và số quân ở node kế tiếp lớn hơn 0 thì trả về true tức là ăn được.

   3.2
```
if (p->dan == 0 && p->laquan == 1 && p->quan == 0 &&
    ((p->next->dan + p->next->quan)>0))
```

return true;

Nếu số dân ở node đang xét bằng 0 và node đó là quan và số quan là 0 và node kế tiếp có số quân lớn hơn 0 thì trả về true tức là ăn được.

3.3 return false;

Mọi trường hợp còn lại đều trả về false tức là không ăn được quân.

4. Minh họa:

Ta lấy ví dụ minh họa được thực hiện ở bước 4.4 trong vấn đề 2:

[6,6,0,6,6] <11> [6,6,0,6,6] <11> (0 || 0)

Ở đây node đang xét được tô màu đỏ, ta có p->dan = 0 thỏa điều kiện, node này không phải node quan nên p->laquan = 0 là thỏa, tổng dân và quan ở node kế tiếp là 6 > 0 nên hàm này sẽ trả về true tức là ăn được.

Xét 1 ví dụ khác:

[6,6,6,6,6] <0> [6,6,0,6,6] <11> (0 || 0)

Node đang xét được tô màu đỏ. Ta có p->dan = 0, nó là 1 ô quan và số quan của nó 0 vì vậy ta vẫn sẽ ăn được giá trị của node kế tiếp nó là 6.

Vấn đề 4: Kiểm tra trạng thái dừng trò chơi

1. Phát biểu bài toán: Trong quá trình chơi, sẽ có những trường hợp mà người chơi sẽ không còn đủ điều kiện để chơi hoặc cả 2 ô quan đều trống thì trò chơi sẽ kết thúc. Vì vậy ta phải thường xuyên kiểm tra các yếu tố đó để trò chơi có thể thực hiện 1 các logic.

   Input: điều kiện hiện tại trên bàn cờ

   Output: Trò chơi có được tiếp tục hay không
2. Ý tưởng giải quyết: Sau mỗi lượt đi kể từ lượt đi đầu thì ta sẽ xét xem số dân tại từng node quan có lớn hơn 0 hay không. Trong trường hợp cả 2 node quan đều không còn quân nào thì ta cho trò chơi kết thúc. Trong trường hợp cả 2 node

quan đề còn thì ta tiến hành xem xét điều kiện chơi của từng người chơi thôi qua 2 biến điều khiển là pause1 và pause2. Khi số quân trên bàn của người chơi không còn và tổng số quân mà người chơi đó ăn được nhỏ hơn 5 thì người chơi đó không còn được phép chơi tiếp.

3. Thuật toán: do có 2 điều kiện nên ta tách thành những hàm khác nhau

   Đầu tiên ta xét hàm isStoped để xét điều kiện 2 ô quan có trống hay không

   3.1 tạo 1 node p tạm: Node *p = new Node;

   3.2 Khởi gán giá trị cho biến i = 1 và temp = 0;

   3.3 Cho vòng lặp for thực hiện lần lượt qua các node (kể cả node dân):

   for (p = l.pHead; i <= 12; i++, p = p->next)

   3.4 Nếu đã thỏa các điều kiện trên thì thực hiện câu lệnh trong vòng lặp, với mỗi Node có p->laquan = 1 tức là ô quan và số dân và quan trong đó = 0 tức là ô đó trống thì ta tăng biến temp lên 1 đơn vị:

```
if (p->laquan == 1 && (p->quan + p->dan) == 0)
        temp++;
```

   3.5 Nếu chạy hết vòng lặp mà biến temp = 2 tức là cả 2 ô quan đều trống thì hàm sẽ trả về true tức là trò chơi kết thúc, ngược lại hàm sẽ trả về false tức là trò chơi còn được tiếp tục trong trường hợp temp nhỏ hơn 2:

```
return temp == 2 ? true : false;
```

   Tiếp đến ta xét đến trường hợp của từng người chơi:

   Ở đây có 2 hàm gần tương tự nhau xét đến cho 2 người chơi là isContinued_P1 và isContinued_P2. Về mặt cấu trúc thì cả 2 hàm này tương đồng nhau nên ta sẽ đề cập cụ thể 1 hàm là isContinued_P1, hàm còn lại tương tự.

   3.6 Trong hàm này có lệnh:

```
if (!isHaveCitizen_InBoard_P1(l) && isGreater_5Citizen(count))
        return true;
else
        return false;
```

Ở đây có 2 hàm mới ta cần xem xét: isHaveCitizen_InBoard_P1 và isGreater_5Citizen.

3.6.1 Đối với hàm isHaveCitizen_InBoard_P1:

```
Node *p = new Node;
int i = 1;
for (p = l.pHead; i <= 5; i++, p = p->next)
{
        if (p->dan>0)
                return true;
}
return false;
```

Ở hàm này ta tạo ra 1 node tạm được gọi là p, gọi 1 vòng lặp for chạy 5 lần từ trái qua phải để xét các node thể hiện ô dân của người chơi 1, qua đó nếu bất kì 1 hoặc nhiều hơn những node thể hiện ô dân của người chơi 1 có số quân lớn hơn 0 thì trả về true, ngược lại trả về false.

Ví dụ:

[0,0,0,0,1] <11> [6,6,0,6,6] <11> (0 || 0)

Trường hợp này hàm sẽ trả về false.

[0,0,0,0,0] <11> [6,6,0,6,6] <11> (0 || 0)

Trường hợp này hàm sẽ trả về true.

3.6.2 Đối với hàm isGreater_5Citizen:

Ở hàm này chỉ xét xem số điểm của người chơi đang có có lớn hơn 5 hay không, nếu lớn hơn hoặc bằng 5 thì trả về true, ngược lại trả về false: return count >= 5 ? true : false;

Như vậy ta xét lại hàm isContinued_P1:

Ở câu lệnh if (!isHaveCitizen_InBoard_P1(l) &&isGreater_5Citizen(count)) có ý nghĩa là nếu người chơi 1 không còn quân trên bàn và tổng số quân

người chơi 1 ăn được nhỏ hơn 5 thì người chơi 1 không còn đủ điều kiện để chơi và trò chơi kết thúc.

4. Minh họa:

Ta xét trường hợp sau:

[0,0,0,0,0] <11> [6,6,0,6,6] <11> (4 || 0)

Ở đây ta thấy người chơi 1 không còn quân trên bàn thì hàm isHaveCitizen_InBoard_P1 sẽ trả về false và tổng số quân của người chơi 1 cũng không còn đủ nên hàm isGreater_5Citizen cũng sẽ trả về false. Do đó hàm isContinued_P1 sẽ trả về false, tức là người chơi 1 không còn đủ điều kiện tham gia trò chơi nữa.

Thực hiện tương tự với người chơi 2.

Ta xét 1 trường hợp khác:

[9,1,7,4,3] <0> [3,4,0,2,10] <0> (8 || 0)

Ở đây cả 2 người chơi đều thỏa mãn yêu cầu được phép chơi tiếp nhưng vì cả 2 ô quan đều trống nên hàm isStoped sẽ trả về false và trò chơi buộc phải dừng lại.

Phần 4. LẬP TRÌNH VÀ CÀI ĐẶT.

4.1 Ngôn ngữ và công cụ sử dụng

- Ngôn ngữ sử dụng : C++

- Lập trình trên môi trường: visual studio 2015

4.2 Các hàm xử lý chính

```
void Init(List &l);
Node *GetNode(int _dan, int _quan, int _laquan);
```

```
void AddHead(List &l, Node *pnew); void CreateList(List &l);
void OutPut(List l, int count1, int count2);

Node *SetNodeFromPos(List l, int pos);
void Distribute_AndScore(List &l, int &count, Node *p, int dir);

bool isScored_P1(Node *p);
bool isScored_P2(Node *p);

bool isStoped(List);
bool isContinued_P1(List, int);
bool isContinued_P2(List, int);

bool isHaveCitizen_InBoard_P1(List);
bool isHaveCitizen_InBoard_P2(List);
bool isGreater_5Citizen(int);

void Distribute_P1(List &l, int &count);
void Distribute_P1(List &l, int &count

void ChangeVirtualPosition(int &pos);
void CheckToDistribute_P1(List l, int &count1, int, bool dung2);
void CheckToDistribute_P2(List l, int, int &count2, bool dung2);

int whoWin(List l, int &count1, int &count2);
```

```
void Init(List &l)
{
	l.pHead = NULL;
}
```

⇨ Cho l.pHead = null.

```
Node* GetNode(int _dan, int _quan, int _laquan)
{
	Node *p = new Node;
	if (p == NULL)
	{
		cout << "Error.The memory is full" << endl;
		return p;
	}

	p->dan = _dan;
	p->quan = _quan;
	p->laquan = _laquan;
	p->next = p->pre = NULL;
	return p;
}
```

⇨ Tạo và truyền giá trị cho Node.

```
void AddHead(List &l, Node *pnew)
{
	if (l.pHead == NULL)
		l.pHead = l.pTail = pnew;
```

```
        else
        {
                l.pHead->pre = pnew;
                pnew->next = l.pHead;
                l.pHead = pnew;
        }
}
```

⇨ Thêm 1 node vào đầu danh sách.

```
void CreateList(List &l)
{
        Init(l);
        for (int i = 1; i <= 12; i++)
        {
                if (i == 1 || i == 7)
                {
                        Node *p = new Node;
                        p = GetNode(0, 10, 1);
                        AddHead(l, p);
                }
                else
                {
                        Node *p = new Node;
                        p = GetNode(5, 0, 0);
                        AddHead(l, p);
                }
        }
        l.pHead->pre = l.pTail;
```

```
	l.pTail->next = l.pHead;
}
```

⇨ Khởi tạo cấu trúc ô ăn quan. Khởi tạo danh sách các node của ô ăn quan gồm 5 nút dân và 2 nút quan.

```
void OutPut(List l, int count1, int count2)
{
	Node *p = new Node;
	int i;
	cout << "-    ";
	for (p = l.pHead, i = 1; i <= 12; i++, p = p->next)
	{
		if (i == 1 || i == 7)
			cout << "[" << p->dan << ", ";
		else if (i == 5 || i == 11)
			cout << p->dan << "]";
		else if (i == 6 || i == 12)
			cout << " <" << p->quan + p->dan << "> ";
		else
			cout << p->dan << ", ";
	}
	cout << "( " << count1 << "||" << count2 << " )" << endl;
}
```

⇨ In ra màn hình List.

```
Node* SetNodeFromPos(List l, int pos)
```

```
{
        Node *p = new Node;
        p = l.pHead;

        for (int i = 1; i <= pos - 1; i++, p = p->next)
        {
                if (i == 5)
                {
                        p = p->next;
                }
        }
        return p;
}
```

⇨ Lấy ra node tại vị trí được chọn. Tại vị trí Node là ô quan thì nhảy qua node kế tiếp.

```
void ChangeVirtualPosition(int &pos)
{
        if (pos == 7)  pos = 9;
        else
                if (pos == 6)  pos = 10;
                else
                        if (pos == 9)  pos = 7;
                        else
                                if (pos == 10)          pos = 6;
}
```

⇨ Vị trí thật là: [1-2-3-4-5] <> [6-7-8-9-10] <> nhưng để trực quan với người dùng nên chỉnh lại vị trí thành [1-2-3-4-5] <> [10-9-8-7-6] <>.

```
bool isScored_P1(Node *p)
{
	if (p->dan == 0 && p->laquan == 0 && ((p->next->dan + p->next->quan)>0))
		return true;
	if (p->dan == 0 && p->laquan == 1 && p->quan == 0 && ((p->next->dan + p->next->quan)>0))
		return true;
	return false;
}
```

⇨ Nếu tại vị trí Node p có dân = 0 và số dân, quan ở node tiếp theo lớn hơn 0 thì Player1 ghi điểm. Ngược lại thì không.

```
bool isScored_P2(Node *p)
{
	if (p->dan == 0 && p->laquan == 0 && ((p->pre->dan + p->pre->quan)>0))
		return true;
	if (p->dan == 0 && p->laquan == 1 && p->quan == 0 && ((p->pre->dan + p->pre->quan)>0))
		return true;
	return false;
}
```

⇨ Nếu tại vị trí Node p có dân = 0 và số dân, quan ở node tiếp theo lớn hơn 0 thì Player1 ghi điểm. Ngược lại thì không.

```
void Distribute_AndScore(List &l, int &count, Node *p, int dir)
{
	Node *temp = new Node;
	temp = p;
	if (dir == 1)
	{
		while (p->dan != 0)
		{
			p->dan--;
			temp = temp->next;
			temp->dan++;
		}
		temp = temp->next;
		while ((temp->dan != 0) && (temp->laquan == 0))
		{
			int t = temp->dan;
			temp->dan = 0;
			while (t != 0)
			{
				t--;
				temp = temp->next;
				temp->dan++;
			}
			temp = temp->next;
		}
		while (isScored_P1(temp))
		{
			count = count + temp->next->dan + temp->next->quan;
```

```
                temp->next->dan = 0; temp->next->quan = 0;
                temp = temp->next->next;
        }
}
else
{
        while (p->dan != 0)
        {
                p->dan--;
                temp = temp->pre;
                temp->dan++;
        }
        temp = temp->pre;
        while ((temp->dan != 0) && temp->laquan == 0)
        {
                int t = temp->dan;
                temp->dan = 0;
                while (t != 0)
                {
                        t--;
                        temp = temp->pre;
                        temp->dan++;
                }
                temp = temp->pre;
        }
        while (isScored_P2(temp))
        {
                count = count + temp->pre->dan + temp->pre->quan;
                temp->pre->dan = 0; temp->pre->quan = 0;
```

```
                temp = temp->pre->pre;
            }
        }
}
```

⇨ Tóm gọn hàm này có nghĩa là tại vị trí đang xét, nếu số dân lớn hơn 0 thì giảm số dân tại ô hiện hành, tăng số dân theo chiều người chơi chọn ở những ô kế tiếp cho đến khi số dân ở node đang xét bằng 0, tức là nếu số dân ở node đang xét bằng 4 thì tiến hành tăng 4 node kế tiếp nó mỗi node 1 đơn vị dân. Sau đó tính điểm.

```
bool isStoped(List l)
{
    Node *p = new Node;
    int i = 1, temp = 0;
    for (p = l.pHead; i <= 12; i++, p = p->next)
    {
        if (p->laquan == 1 && (p->quan + p->dan) == 0)
            temp++;
    }
    return temp == 2 ? true : false;
}
```

⇨ Kiểm tra xem 2 ô quan có trống không, nếu trống cả 2 thì dừng trò chơi.

```
bool isHaveCitizen_InBoard_P1(List l)
{
    Node *p = new Node;
    int i = 1;
    for (p = l.pHead; i <= 5; i++, p = p->next)
```

```
        {
                if (p->dan>0)
                        return true;
        }
        return false;
}
```

⇨ Kiểm tra xem Player1 còn quân trên bàn hay không.

```
bool isHaveCitizen_InBoard_P2(List l)
{
        Node *p = new Node;
        int i = 1;
        for (p = l.pTail->pre; i <= 5; i++, p = p->pre)
        {
                if (p->dan>0)
                        return true;
        }
        return false;
}
```

⇨ Kiểm tra xem Player2 còn quân trên bàn hay không.

```
bool isGreater_5Citizen(int count)
{
        return count >= 5 ? true : false;
}
```

⇨ Kiểm tra xem có còn đủ 5 quân trong số quân ăn được hay không.

```
bool isContinued_P1(List l, int count)
 {
        if (isHaveCitizen_InBoard_P1(l) && isGreater_5Citizen(count))
                return true;
        else
                return false;
}
```

⇨ Nếu P1 còn dân trên bàn và có nhiều hơn hoặc bằng 5 dân trong số dân ăn được thì có thể tiếp tục trò chơi.

```
bool isContinued_P2(List l, int count)
{
        if (isHaveCitizen_InBoard_P2(l) && isGreater_5Citizen(count))
                return true;
        else
                return false;
}
```

⇨ Nếu P2 còn dân trên bàn và có nhiều hơn hoặc bằng 5 dân trong số dân ăn được thì có thể tiếp tục trò chơi.

```
void Distribute_P1(List &l, int &count)
{
        Node *p = new Node;
        int i = 1;
        for (p = l.pHead; i<=5; i++, p = p->next)
        {
```

```
            p->dan++;
        }
        count -= 5;
}
```

⇨ Khi mà P1 hết dân trên bàn thì lấy dân trong số dân ăn được rải đều mỗi ô 1 dân và trừ đi 5 số dân ăn được.

```
void Distribute_P2(List &l, int &count)
{
        Node *p = new Node;
        int i = 1;
        for (p = l.pTail->pre; i<=5; i++, p = p->pre)
        {
            p->dan++;
        }
        count -= 5;
}
```

⇨ Khi mà P2 hết dân trên bàn thì lấy dân trong số dân ăn được rải đều mỗi ô 1 dân và trừ đi 5 số dân ăn được.

```
void CheckToDistribute_P1(List l, int &count1, int count2, bool pause)
{
        if (!pause && !isHaveCitizen_InBoard_P1(l) )
        {
            cout << "\nPLAYER 111 het quan tren ban. Rai Quan: \n" << endl;
            Distribute_P1(l, count1);
```

```
            OutPut(l, count1, count2);
        }
}
```

⇨ Nếu P1 không đủ dân trên bàn thì thông báo ra màn hình và tiến hành rải quân, sau khi rải xong in ra màn hình kết quả đã rải.

```
void CheckToDistribute_P2(List l, int count1, int &count2, bool pause)
{
        if (!pause && !isHaveCitizen_InBoard_P2(l) )
        {
            cout << "\nPLAYER 222 het quan tren ban. Rai Quan: \n" << endl;
            Distribute_P2(l, count2);
            OutPut(l, count1, count2);
        }
}
```

⇨ Nếu P2 không đủ dân trên bàn thì thông báo ra màn hình và tiến hành rải quân, sau khi rải xong in ra màn hình kết quả đã rải.

```
int whoWin(List l, int &count1, int &count2)
{
        Node *p = new Node;
        Node *t = new Node;
        int i = 1, j = 1;
        for (p = l.pHead; i<=5; i++, p = p->next)          count1 += p->dan;
        for (t = l.pTail->pre; j<=5; j++, t = t->pre)      count2 += t->dan;
```

```
        if (count1>count2)  return 1;
        else
                if (count1 == count2) return 0;
                else return 2;
}
```

⇨ Đếm số dân còn lại trên bàn của mỗi người chơi, sau đó cộng với số dân đã ăn được trong quá trình chơi và trả về 1 nếu P1 thắng, 2 nếu P2 thắng và 0 nếu hòa.

Phần 5. KIỂM TRA

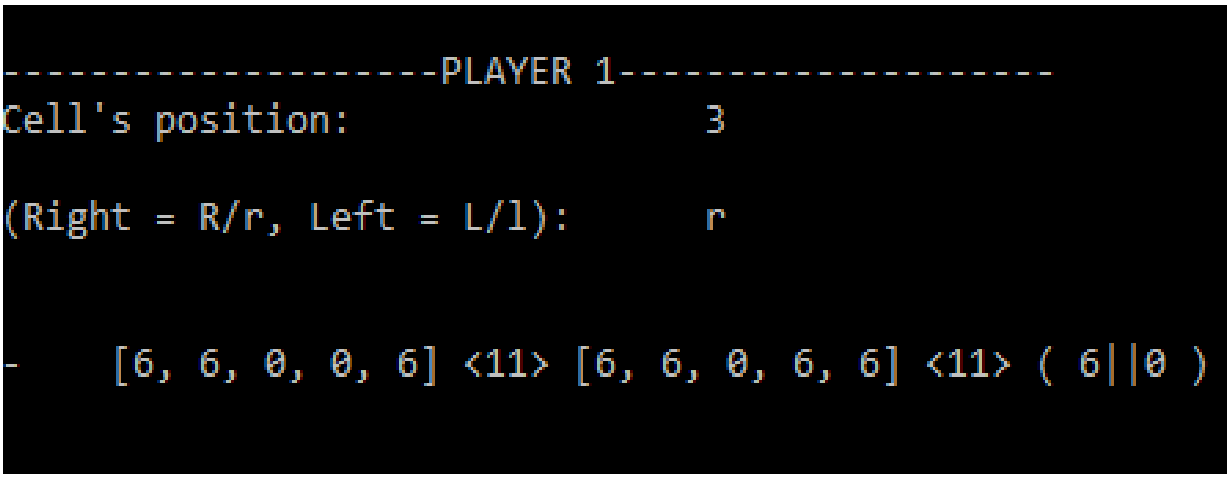
5.1 Cách thức kiểm tra

Nhập lần lượt ô mong muốn di chuyển và hướng di chuyển theo hướng dẫn

Ví dụ: [5, 5, 5, 5, 5] <10> [5, 5, 5, 5, 5] <10> (0||0)

Các ô sẽ được đánh số là 1 – 2 – 3 – 4 – 5 <> 10 – 9 – 8 – 7 – 6 <>

Người chơi bắt đầu là người chơi số 1 (P1) sẽ chỉ được nhập từ 1 tới 5, nếu khác đó thì chương trình sẽ bắt nhập lại. Sau đó nhập vào hướng di chuyển quân, R/r tượng trưng cho hướng bên phải, L/l tượng trưng cho hướng bên trái.



Sau khi P1 đã hoàn thành nước di chuyển của mình sẽ tới P2, P2 cũng thực hiện các thao tác tương tự P1 và P2 chỉ nhập số từ 10-6.

```
--------------------PLAYER 2--------------------
Cell's position:                2

Your cell is INCORRECT or NO CITIZEN ---------- Please choose again :)

Cell's position:                7

(Right = R/r, Left = L/l):      l
```

Trong trường hợp hết quân để di chuyển thì chương trình sẽ thông báo và tự động rải quân.

```
-    [0, 0, 2, 1, 6] <20> [0, 0, 0, 0, 0] <19> ( 21||1 )


--------------------PLAYER 2--------------------

PLAYER 222 het quan tren ban. Rai Quan:

-    [0, 0, 2, 1, 6] <20> [1, 1, 1, 1, 1] <19> ( 21||-4 )
```

Trong trường hợp người chơi không có đủ quân để rải thì cho vay mượn.

Khi 2 ô quan đã không còn quân nào thì trò chơi kết thúc và sẽ hiển thị điểm số. Nếu nhấn số 1 thì chơi lại, nếu nhấn số 0 thì thoát khỏi trò chơi.

```
-    [0, 0, 0, 0, 0] <0> [0, 1, 1, 1, 0] <0> ( 20||47 )

PLAYER 1: 20 POINTS
PLAYER 2: 50 POINTS
PLAYER 2 WIN
Play again? (1-Yes / 0-No)
```

5.2 Kết quả kiểm tra

Sau đây là đoạn test nhỏ để chạy thử chương trình:

3r

7r

1l

10l

1r

8l

2r

6r

1l

6l

2r

10l

3r

9r

2r

8l

3r

9l

1r

10r

5.3 Nhận xét, đánh giá

Phần 6. HƯỚNG PHÁT TRIỂN

6.1 Hạn chế

Do làm bằng console nên chưa có trực quan với người dùng.

Quá trình chơi còn gặp nhiều khó khăn do phải xử lý nhiều tình huống bằng các nút bấm trên bàn phím.

Chưa xây dựng được các thuật toán hỗ trợ người chơi như thuật giả kiểm tra ăn quân nhiều nhất, hạn chế đối phương ăn quân ít nhất…

6.2 Hướng phát triển

Phát triển màn hình dễ sử dụng hơn với các nút tương tác trực tiếp với người dùng.

Xây dựng thêm 1 vài các thuật giải hỗ trợ người dùng.

Xây dựng các chức năng phân tích tình huống, giúp người dùng nâng cao kĩ thuật chơi.

Phát triển chế độ chơi nhiều người.

Phần 7. TÀI LIỆU THAM KHẢO

1. Giáo Trình Cấu Trúc Dữ Liệu Và Giải Thuật – PGS.TS Đỗ Văn Sơn, ThS. Trịnh Quốc Sơn.
2. Internet (wikipedia, gamevui.com,…).

Phần 8. CHỈNH SỬA

Được sự hỗ trợ nhiệt tình từ cô, chúng em đã chỉnh sửa lại nội dung của báo cáo cụ thể như:

Phần 2 THIẾT KẾ CẤU TRÚC DỮ LIỆU:

1. Thêm ý tưởng tổng quát cấu trúc dữ liệu cho trò chơi
2. Thể hiện rõ vì sao nó là danh sách liên kết vòng
3. Cho ví dụ minh họa bài toán

Phần 3 THIẾT KẾ GIẢI THUẬT:

1. Chỉnh sửa lại phần ý tưởng theo định hướng của cô
2. Thêm sơ đồ (vẽ chưa đẹp lắm)
3. Phần giải thuật nêu rõ các vấn đề cần giải quyết
4. Các thuật toán và ví dụ minh họa