1. Game C++ snake game
2. Viết pseudocode cho hàm main

int main(void) {

<Khởi tạo trò chơi>

<Vòng lặp chính> {

<Vẽ màn hình>

<Nhập liệu>

<Thực hiện thuật toán>

}

<Kết thúc trò chơi>

return 0;

}

1. Trong từng hàm con, sinh viên hãy hỏi câu “Để làm gì” và ghi lại câu trả lời cho từng hàm

**1. Hàm Setup để làm gì?**

* Hàm Setup dùng để khởi tạo các biến và thiết lập các giá trị khởi tạo của trò chơi.
* Hàm này khởi tạo các biến trò chơi như trạng thái gameover, vị trí khởi tạo của rắn và trái cây, điểm số, và các biến liên quan đến chiều dài của rắn và các phần của đuôi. Nó đảm bảo rằng trò chơi bắt đầu với các thiết lập đúng.
* Hàm này thiết lập các giá trị ban đầu cho các biến như **gameover**, **dir**, **x**, **y**, **fruitX**, **fruitY**, **score**.

**2. Hàm Draw để làm gì?**

* Hàm Draw dùng để vẽ giao diện trò chơi lên màn hình.
* Hàm này vẽ các thành phần của trò chơi bao gồm các bức tường, rắn (bao gồm cả đuôi của nó), và trái cây. Nó cũng hiển thị điểm số hiện tại của người chơi. Hàm sử dụng system("cls") để xóa màn hình trước khi vẽ lại, tạo hiệu ứng chuyển động cho rắn.

**3. Hàm Input để làm gì**

* Hàm Input dùng để nhận và xử lý các lệnh đầu vào từ bàn phím của người chơi.
* Hàm này kiểm tra xem người chơi có nhấn phím nào không bằng cách sử dụng \_kbhit. Nếu có, nó đọc phím nhấn bằng \_getch và cập nhật hướng di chuyển của rắn hoặc kết thúc trò chơi nếu người chơi nhấn phím 'x'.

**4. Hàm Algorithm để làm gì?**

* Hàm Algorithm dùng để thực hiện các phép toán của trò chơi, cập nhật trạng thái của rắn và kiểm tra các điều kiện thắng/thua.
* Hàm này cập nhật vị trí của rắn, di chuyển các phần của đuôi, kiểm tra xem rắn có ăn trái cây không và cập nhật điểm số cũng như kích thước của rắn. Nó cũng kiểm tra xem rắn có va chạm với chính mình không và điều chỉnh vị trí của rắn nếu nó ra ngoài màn hình.
* Cập nhật vị trí của con rắn dựa trên hướng di chuyển hiện tại.
* Kiểm tra xem con rắn đã chạm vào bức tường hoặc chính nó hay chưa. Nếu có, trò chơi sẽ kết thúc.
* Kiểm tra xem con rắn đã ăn quả ăn hay chưa. Nếu có, điểm số sẽ tăng và quả ăn mới sẽ được tạo ra.

**5. Hàm main để làm gì?**

* Hàm main là hàm chính để khởi chạy trò chơi.
* Hàm này gọi hàm Setup để thiết lập trò chơi và sau đó vào vòng lặp chính của trò chơi. Trong vòng lặp này, nó liên tục gọi các hàm Draw, Input, và Algorithm cho đến khi trò chơi kết thúc.

1. Game Zombie war
2. Viết pseudocode cho hàm main

int main (void){

<Trò chơi bắt đầu>

<Khởi tạo biến `playerAlive` = true>

<Khởi tạo biến `playerSkill` = 9>

<Khởi tạo biến `playerScore` = 1>

<Khởi tạo biến `playerName` = "">

<Khởi tạo biến `zombieCount` = 0>

<Khởi tạo biến `zombiesKilled` = 0>

<Tiêu đề và nhập thông tin>

<In ra màn hình "Welcome to Zombie War.">

<Yêu cầu người chơi nhấn Enter để bắt đầu>

<Yêu cầu người chơi nhập tên người chơi>

<Yêu cầu người chơi nhập số lượng zombie muốn chiến đấu>

<Vòng lặp chính>{

<Tạo một zombie ngẫu nhiên>

<In ra màn hình thông tin về zombie sắp tới>

<Thực hiện trận chiến giữa người chơi và zombie>

IF (<người chơi bị zombie giết>) {

<hiện ra playerAlive = false>

}

ELSE

{

<Người chơi giết zombie, tăng `zombiesKilled` lên 1>

<Cập nhật điểm số dựa trên kết quả trận chiến>

}

<Kết thúc trò chơi>

IF (<người chơi sống xót sau khi giết hết zombie>) {

<In ra màn hình thông báo chiến thắng>

}

ELSE

{

<In ra màn hình thông báo thất bại>

<In ra màn hình số lượng zombie đã giết và điểm số cuối cùng>

}

1. Trong từng hàm con, sinh viên hãy hỏi câu “Để làm gì” và ghi lại câu trả lời cho từng hàm.

**Hàm createZombie để làm gì?**

Hàm createZombie sinh ra một giá trị số nguyên ngẫu nhiên đại diện cho độ khó của zombie. Nếu số ngẫu nhiên nhỏ hơn 10, hàm trả về giá trị 11, tức là một zombie lớn với độ khó cao. Nếu không, hàm trả về một giá trị từ 1 đến 10, đại diện cho một zombie bình thường với độ khó tương ứng.

**Hàm main để làm gì**

Hàm main là hàm chính của chương trình điều khiển toàn bộ quy trình trò chơi. Nó thực hiện các bước sau:

1. Khởi tạo các biến trò chơi và thiết lập các giá trị ban đầu.
2. Hiển thị thông điệp chào mừng và yêu cầu người chơi nhập tên và số lượng zombie mà họ muốn chiến đấu.
3. Thực hiện vòng lặp chính của trò chơi, bao gồm:

* Tạo ra một zombie ngẫu nhiên.
* Thực hiện trận chiến giữa người chơi và zombie.
* Cập nhật điểm số và trạng thái của người chơi.

1. Sau khi kết thúc vòng lặp, nó thông báo kết quả trò chơi và điểm số cuối cùng của người chơi.