**Linked List và Array**

List dùng để lưu những phần tử mà mình muốn nhóm vào 1 chỗ, giống như array.

Thằng list đấy nếu như mình khởi tạo nó 1 struct cùng 1 kiểu dữ liệu thì nó cũng là 1 kiểu dữ liệu

**Vậy khi nào thì list, khi nào thì linked list ấy, khi nào dùng array?**

List có số lượng phần tử ko cố định, khác với array

Còn bài toán số người vào ra liên tục trong 1 phòng làm việc, số người ra vào ko cố định

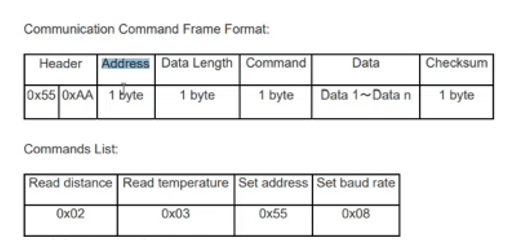
**Bài toán tìm kiếm, list mạnh hơn hay array mạnh hơn?**

List này sẽ phải tìm kiếm từ phần tử đầu, truy xuất dc phần từ trc thì mới truy xuất được phần tử sau còn array thì có thể truy xuất bất cứ phần tử nào trong array

Gán giá trị dùng switch case nhiều hơn

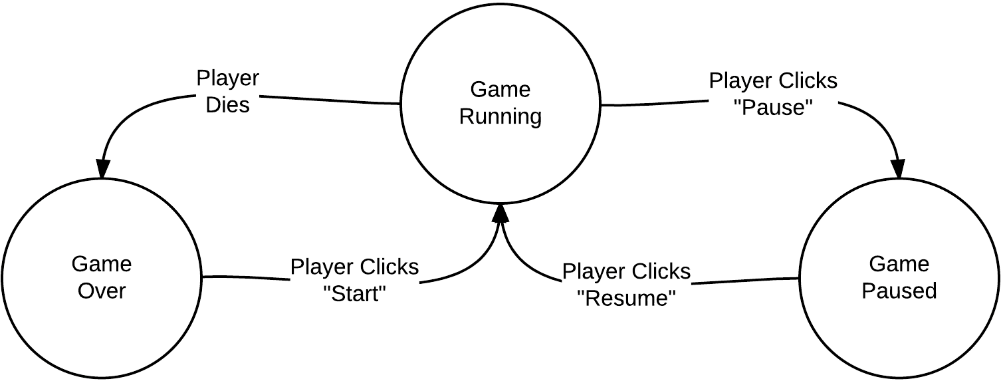
**1 bài toán ntn thì dùng if else**

Nếu muốn vào 1 case thì bắt buộc dùng break, ko có break thì nó chạy luôn tất cả các trường hợp



Lúc cảm biến đẩy dữ liệu lên vi điều khiển, ko phải lúc nào cũng bắn header lên đầu tiên mà có thể random bắn checksum, data chẳng hạn vì nó có điện là cắm lên.

Trong quá trình làm việc, ta sử dụng state machine là đơn giản nhất



Có 1 điều kiện gì xảy ra thì cứ chuyển trạng thái thì để áp dụng switch case

Trên vdk nhận 55 thì gọi hàm nhận header chẳng hạn và trong expresson sẽ nhận là header

Và mình fill vào header đó bằng 2 byte và thoát khỏi hàm này ra, hàm nhận nó nhận hết header rồi thì nó tiếp tục nhận đến địa chỉ

Trong hàm đấy có biến toàn cục để chuyển điều kiện về constant 2 và nhận địa chỉ, biến toàn cục này sẽ được giữ nguyên, thoát khỏi hàm thì vẫn giữ giá trị

Biến static thì nhận địa chỉ thì lúc thoát ra sẽ nhận thêm 1 byte nữa thì trong này sẽ chuyển trạng thái từ constant 2 sang constant 3

Đầu tiên hàm nhận nhận 1 biến ko phải 55 thì bỏ qua, đến khi nào nhận dc 55 thì gọi hàm này. Khi nhân được 55 rồi tiếp tục gọi AA (expression ko còn tìm constant1 nữa mà tìm constant2) và biến expression này giữ nguyên và thoát khỏi hàm, sau khi nhận biến thứ 3 sau thằng AA, gọi tiếp vào hàm này thì lúc này expression ko phải constant 1 nữa mà constant2, constant2 thì sẽ lưu biến đó vào địa chỉ chứ ko phải lưu vào thằng header và xử lý này dc gọi là máy trạng thái và hay dc dùng nhất

**Câu lệnh while**

ứng dụng : do 1 điều gì đó mà 1 thanh ghi bị reset về 0, muốn khi nào reset về 0 thì làm việc tiếp theo và ra khỏi vòng lặp while đấy để đợi và so sánh nó với 1 thì nằm im ở đấy khác 1 (=0 chẳng hạn) thì làm việc tiếp

đợi 1 thanh ghi tới trạng thái nào đó thì mới làm việc được tiếp

ko nên dùng while và do while trong lập trình nhúng vì nó dễ rơi vào vòng lặp vô hạn và gây tốn bộ nhớ. Nếu kiểm soát được số lần lặp thì ko nên dùng mà dùng for

có những bài toán đợi thanh ghi đến 1 trạng thái nhất định mới làm được tiếp thì lúc đấy sẽ ko biết dc nó lặp lại bao nhiêu lần.

**do while thì nó khác với while ntn**

tuỳ vào bài toán mà dùng

goto ko nên dùng vì nó phá vỡ luồng code. Phải kiếm soát chương trình rất gắt

continue sẽ bỏ qua những lệnh sau nó và nó quay trở lại vòng lặp và thực hiện các vòng lặp tiếp theo, nó sẽ khác với break. Break là thoát khỏi vòng lặp luôn

câu hỏi: nếu gọi exit() trong hàm local thì sao ví dụ hàm main gọi 1 hàm A, trong hàm A đó gọi hàm exit() thì dừng cả chương trình luôn

**câu hỏi:** dùng break trong while, do while được không, được, ko vấn đề gì

chỉ thoát khỏi vòng lặp thôi.