# Cấu trúc dữ liệu và giải thuật

## Khái niệm

* **Thông tin** được biểu hiện trên máy tính bằng mã nhị phân (Binary) và mã thập lục phân (HEX)
* **Cấu trúc dữ liệu** là cách thức tổ chức, sắp xếp dữ liệu để biểu diễn thông tin trên máy tính
* **Giải thuật** để mô tả các phương pháp giải quyết vấn đề phù hợp cho việc triển khai trên máy tính
* Cấu trúc dữ liệu và giải thuật là hai thuật ngữ luôn đi kèm với nhau. Cấu trúc dữ liệu là đầu vào cũng là đầu ra của giải thuật

## Kiểu dữ liệu

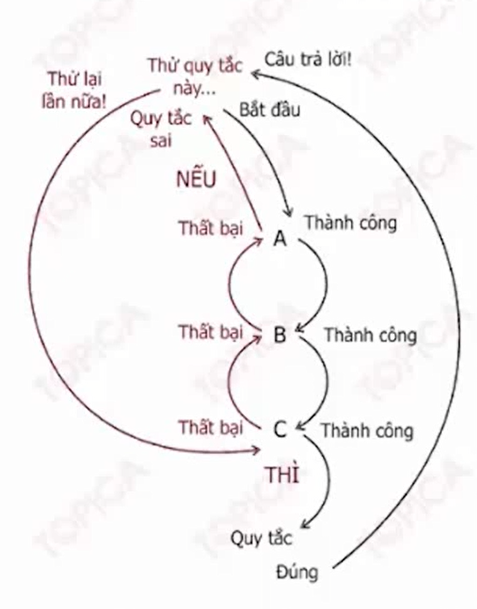
* Kiểu dữ liệu cơ bản: Kiểu số nguyên, Kiểu số thực, Kiểu logic (boolean), Kiểu ký tự (char), Kiểu con trỏ
* Kiểu dữ liệu trừu tượng: Kiểu mảng, Kiểu bản ghi

## Câu lệnh

* Lệnh gán
* Lệnh rẽ nhánh
* Lệnh lặp

# Đệ quy

* Giải thuật theo phương pháp đệ quy là việc gọi lại chính giải thuật đó.
* Đệ quy hoạt động theo cơ chế Stack (Last in First Out)
* Ưu điểm:
  + Tính hiệu quả trong thiết kế, chương trình biểu diễn đơn gian
* Nhược điểm:
  + Tăng kích thước bộ nhớ tăng lên, và tăng thời gian tính toán
* Phương pháp quay lui:



## Đệ quy tuyến tính

* Đệ quy tuyến tính là trường hợp hàm chỉ gọi chính nó một lần

### VD: Thuật toán tính giai thừa của một số

* Thuật toán

GiaiThua(n){

If(n==1 && n==0) return 1

Else return n\*giaiThua(n-1)

}

* Chương trình

#include<stdio.h>

#include<conio.h>

int giaiThua(int n){

if(n==1||n==0){

return 1;

}

else{

return (n\*giaiThua(n-1));

}

}

int main(){

printf("%d", giaiThua(5));

getch();

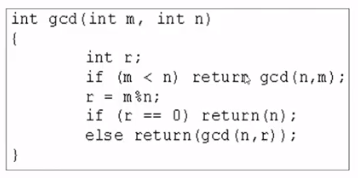
}

## Đệ quy đuôi

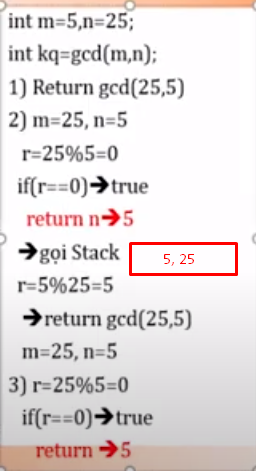
* Là một dạng đặc biệt của đệ quy tuyến tính.

### VD: Đệ quy tìm ước số chung lớn nhất.

* Thuật toán:



* Giải thuật:

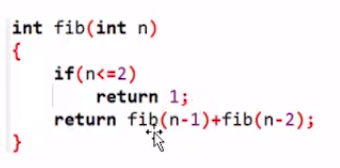


## Đệ quy nhị phân

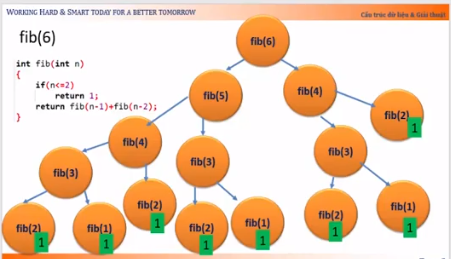
* Là dạng đệ quy gọi 2 lần chính nó.

### VD: Đệ quy tìm số fibonacci

* Thuật toán:

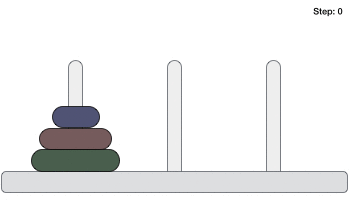


* Giải thuật:



* Fib(6) = 8

## Bài toán tháp Hà Nội



* Áp dụng phương pháp quay lui để giải bài toán

### Phân tích bài toán

* Đầu tiên lấy cột 3 là cọc trung gian chuyển n-1 đĩa từ cột 1 sang cột 2
* Sau đó ta chuyển đĩa lớn nhất từ cột 1 sang cột 3
* Lấy cột 1 làm trung gian chuyển n-1 đĩa từ cột 2 sang cột 3

### Thuật toán

Thap\_ha\_noi(int n, char a, char b, char c)

Với: n là số đĩa

a là cột nguồn

b là cột trung gian

c là cột đích

Với bài toán trên ta phải đệ quy 3 lần:

Thap\_ha\_noi(n-1, a,c,b)

Thap\_ha\_noi(1, a,b,c)

Thap\_ha\_noi(n-1, b,a,c)

### Chương trình

#include<stdio.h>

#include<conio.h>

int thapHN(int n, char a, char b, char c){

if(n==1){

printf("A ---> C");

}

thapHN(n-1, a,c,b);

thapHN(1, a,b,c);

thapHN(n-1,b,a,c);

}

int main(){

char a="A", b="B", c="C";

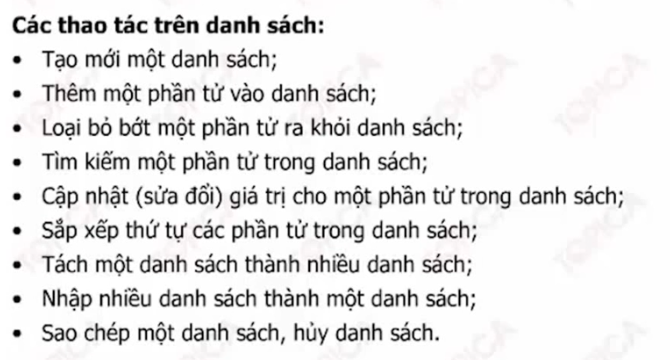
thapHN(3,a,b,c);

getch();

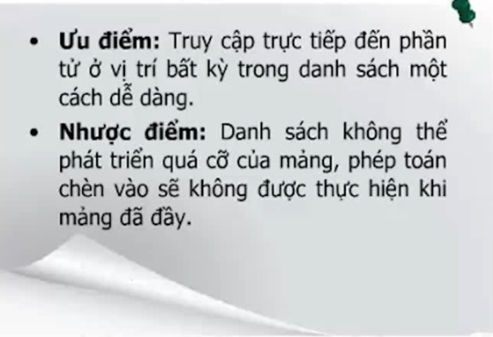
}

# Danh sách

* Là một cấu trúc dữ liệu gồm một hữu hạn các phần tử có kiểu dữ liệu xác định và giữa các phần tử có mối liên hệ với nhau.

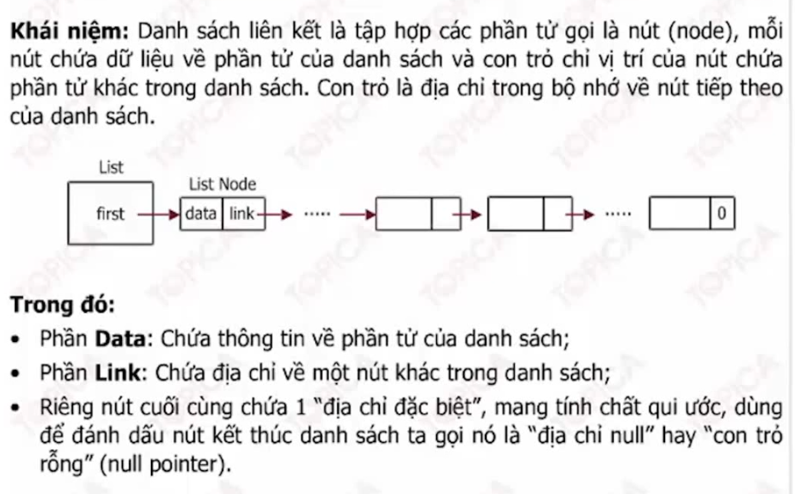


## Danh sách dạng mảng

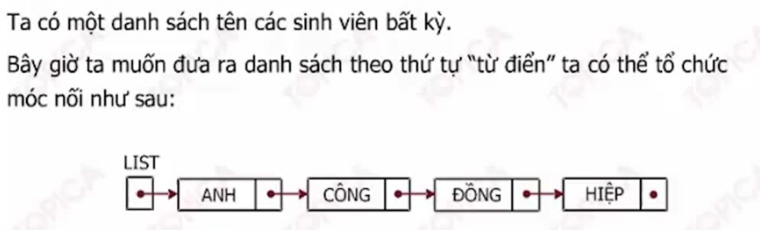


Nhược điểm: Không tối ưu hóa bộ nhớ

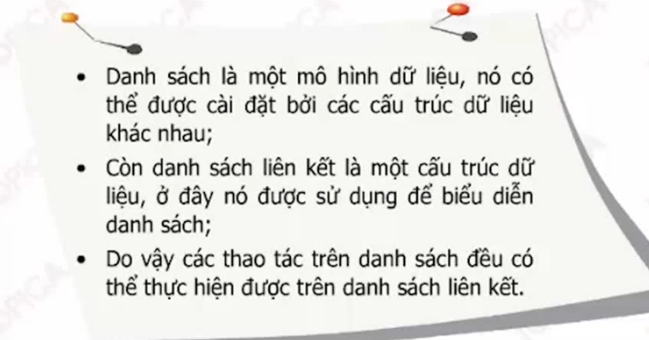
## Danh sách liên kết



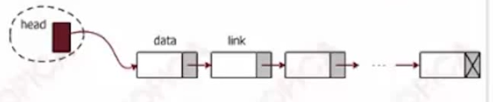
VD:

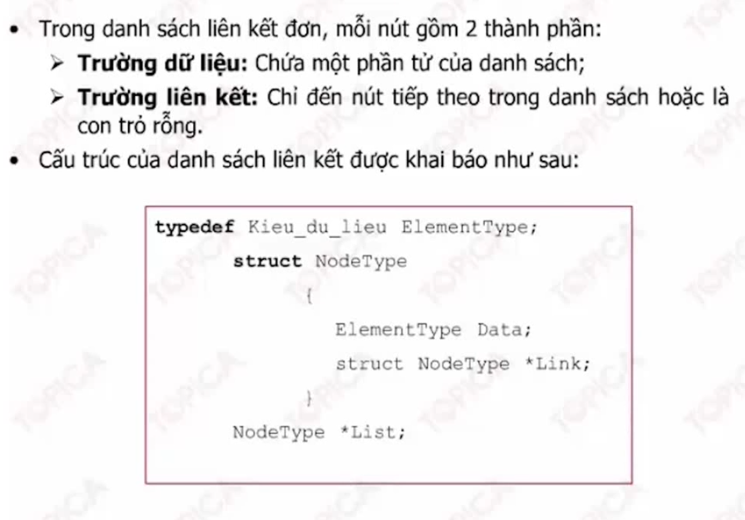


* Sự khác nhau giữa danh sách và danh sách liên kết.

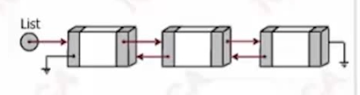


### Danh sách liên kết đơn

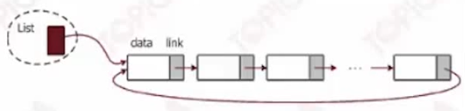




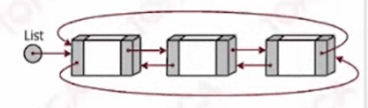
### Danh sách liên kết đôi



### Danh sách liên kết vòng đơn



### Danh sách liên kết vòng đôi



## Ứng dụng của danh sách

# Ngăn xếp

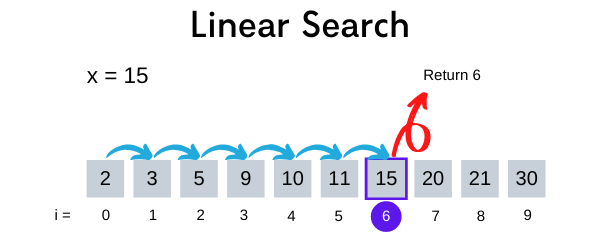
# Hàng đợi

# Cây

# Tìm kiếm

## Tìm kiếm tuyến tính

* Thuật toán tiến hành so sánh x với lần lượt các biến trong mảng hoặc trong list đến khi nào có biến trong mảng trùng với x thì dừng



* Các bước tiến hành giải thuật:

Bước 1: I = 1 //Bắt đầu từ phần tử đầu tiên của dãy

Bước 2: So sánh a[i] với x, có hai khả năng:

+ a[i] = x : Tìm thấy => Dừng

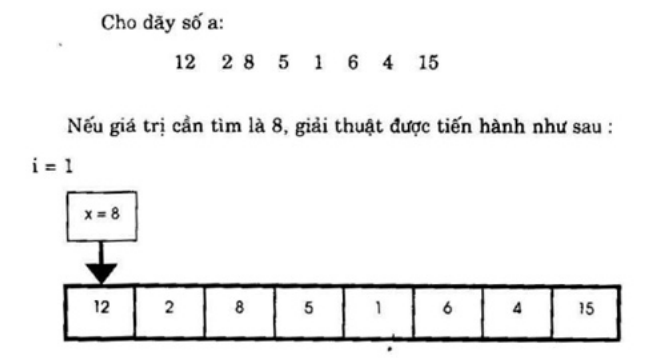
+ a[i] != x: Sang bước 3

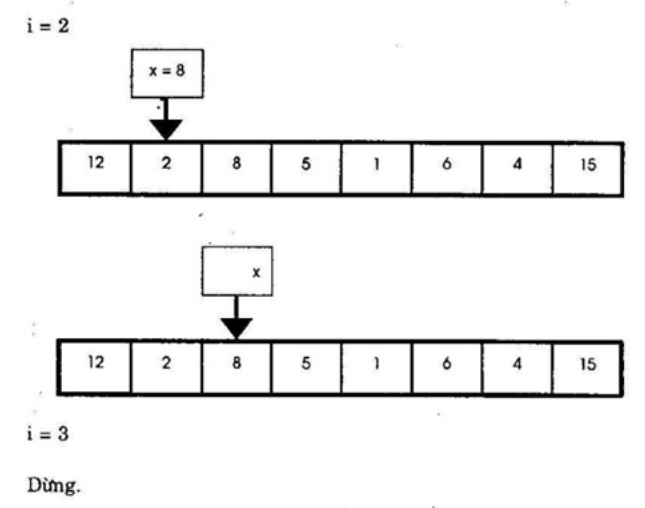
Bước 3: I = i+1 //Xét tiếp các phần tử kế tiếp trong mảng

Nếu i>N: Hết mảng => Không tìm thấy => Dừng

Ngược lại: Lặp bước 2

VD:





## Tìm kiếm nhị phân

## Cây nhị phân tìm kiếm

# Sắp xếp

## Sắp xếp chèn

## Sắp xếp chọn

## Sắp xếp đổi chỗ - Sắp xếp nổi bọt