NỘI DUNG

CẤU TRÚC DỮ LIỆU ĐỘNG

Biến Tĩnh

- Được khai báo tường minh, có tên gọi
- Tồn tại trong phạm vi khai báo
- Được cấp phát trong stack
- Kích thước không đổi => không tận dụng hiệu quả bộ nhớ
- Ví dụ : int x,y;

  char c;

  float f[5];
- Khi biết chắc nhu cầu sử dụng đối tượng trước khi thực sự xử lý : dùng biến tĩnh

Ví Dụ Hạn Chế Của Biến Tinh

- Tổ chức danh sách lớp học
- Dùng mảng tĩnh :

```
typedef        struct
{
  char ten[20];
    int   maso;
}Hocvien;

Hocvien       danhsach[50];
```

- Số lượng học viên <50 => lãng phí
- Số lượng học viện > 50 => thiếu chỗ !

Biến Động

- Không được khai báo tường minh, không có tên gọi
- Xin vùng nhớ khi cần, giải phóng khi sử dụng xong
- Được cấp phát trong heap
- Linh động về kích thước

Kiểu con trỏ

- Kiểu con trỏ dùng lưu địa chỉ của một đối tượng dữ liệu khác.
- Khai báo trong C :

```
int   * p;
```

Biến thuộc kiểu con trỏ p là biến mà giá trị của nó là địa chỉ cuả một vùng nhớ ứng với một biến kiểu T, hoặc là giá trị NULL.

- Bản thân biến con trỏ là không động
- Dùng biến con trỏ để lưu giữ địa chỉ của biến động => truy xuất biến động thông qua biến con trỏ

Các thao tác trên kiểu con trỏ

- Tạo ra một biến động và cho con trỏ ‘p’ chỉ đến nó:
  - void* malloc(size);
  - new // hàm cấp phát bộ nhớ trong C++
- Hủy một biến động do p chỉ đến :
  - Hàm free(p) huỷ vùng nhớ cấp phát bởi hàm malloc do p trỏ tới
  - Hàm delete p huỷ vùng nhớ cấp phát bởi hàm new do p trỏ tới

Sử dụng biến tĩnh, con trỏ và biến động

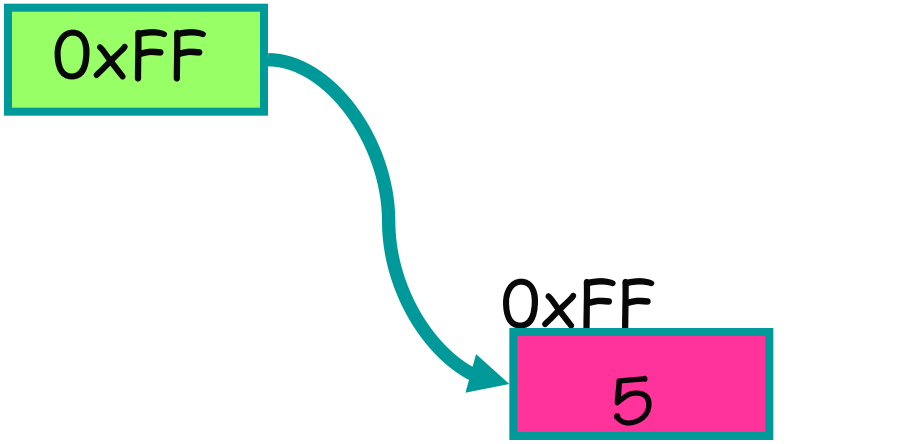
```
int   x;
x = 5 ;
```

Biến không động x


```
int *p;

 p =
 new(int);
*p = 5
```

Biến con trỏ p



Biến động có địa chỉ 0xFF

Kiểu danh sách

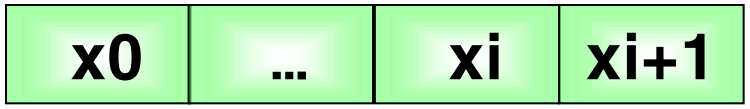
- Danh sách = { các phần tử có cùng kiểu}
- Danh sách là một kiểu dữ liệu tuyến tính :
  - Mỗi phần tử có nhiều nhất 1 phần tử đứng trước
  - Mỗi phần tử có nhiều nhất 1 phần tử đứng sau
- Là kiểu dữ liệu quen thuộc trong thực tế :
  - Danh sách học sinh
  - Danh mục sách trong thư viện
  - Danh bạ điện thoại
  - Danh sách các nhân viên trong công ty
  - …

Các hình thức tổ chức danh sách

- CTDL cho mỗi phần tử ?
- Thể hiện liên kết của các phần tử ?
- Hai hình thức cơ bản :
  - Liên kết ngầm : Mảng
  - Liên kết tường minh : Danh sách liên kết

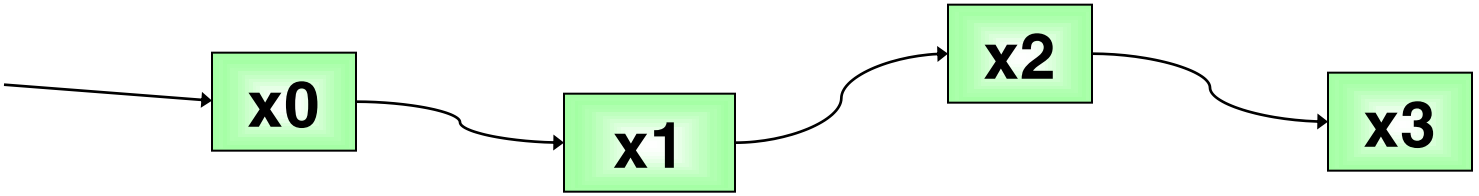
Danh sách liên kết ngầm(mảng)

- Mối liên hệ giữa các phần tử được thể hiện ngầm:
  - xi : phần tử thứ i trong danh sách
  - xi , xi_{+1} là kế cận trong danh sách
- Phải lưu trữ liên tiếp các phần tử trong bộ nhớ
- **Ưu điểm** : Truy xuất trực tiếp, nhanh chóng
- **Nhược điểm**:
  - Sử dụng bộ nhớ kém hiệu quả
  - Kích thước cố định
  - Các thao tác thêm vào , loại bỏ không hiệu quả



Liên kết tuờng minh(Danh sánh liên kết)

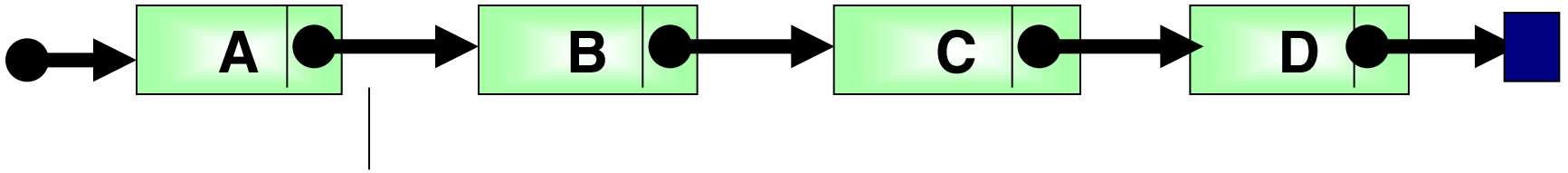
- CTDL cho một phần tử
  - Thông tin bản thân
  - Địa chỉ của phần tử kế trong danh sách



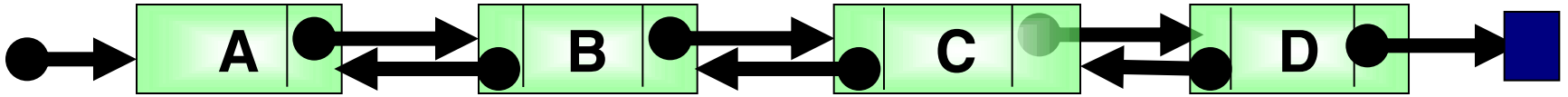
- Mỗi phần tử là một biến động
- **Ưu điểm**
  + Sử dụng hiệu quả bộ nhớ
  + Linh động về số lượng phần tử

Các loại danh sách liên kết

- **Danh sách liên kết đơn**: Mỗi phần tử liên kết với phần tử đứng sau nó trong danh sách



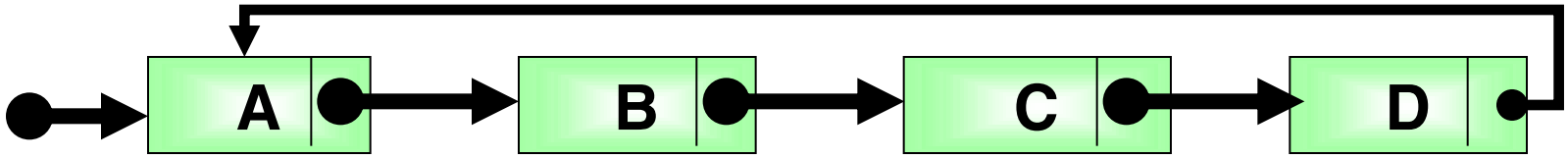
- **Danh sách liên kết kép**: Mỗi phần tử liên kết với phần tử đứng trước và sau nó trong danh sách



- **Danh sách liên Vòng**: Phần tử cuối danh sách liên với phần tử đầu danh sách

Các loại danh sách liên kết (tt)

- **Danh sách liên Vòng:** Phần tử cuối danh sách liên với phần tử đầu danh sách
  - Danh sách liên kết đơn vòng



  - Danh sách liên kết đôi vòng