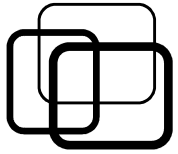
Một số vấn đề nâng cao

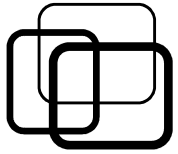
GV. Nguyễn Minh Huy

Nội dung



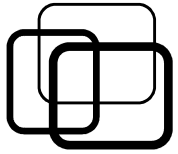
- Mảng động.
- Đệ quy.
- Các thuật toán thông dụng.

Nội dung



- **Mảng động.**
- Đệ quy.
- Các thuật toán thông dụng.

Mảng động



- Vấn đề với mảng tĩnh:
  - Phải khai báo kích thước tối đa.
  - Không phải lúc nào cũng dùng hết.

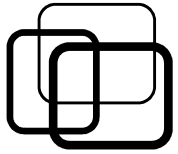
  ➔ Tốn nhiều bộ nhớ.

  ➔ Nhu cầu mảng kích thước tùy biến.
- Khái niệm con trỏ:
  - Biến “lưu động”.

    ➔ Có thể “định cư” ở nhiều vùng nhớ.

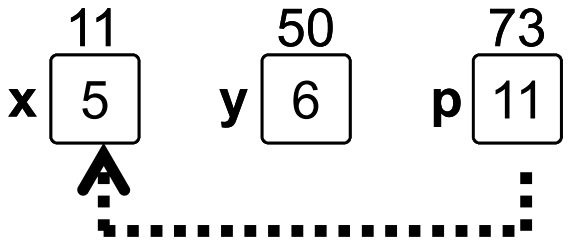
    ➔ “Trỏ” đến các vùng nhớ khác nhau.

Mảng động



- Con trỏ trong C:
  - Khai báo: <Kiểu dữ liệu> ***<Tên con trỏ>**;
  - Lấy địa chỉ biến: <Tên con trỏ> = **&<Tên biến>**;

```
int  x = 5;
int  y = 6;
int  *p;
p = &x;
```



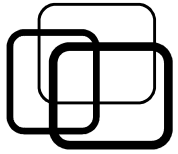
  - Truy xuất dữ liệu ô nhớ:

<Tên biến> = ***<Tên con trỏ>**;

<Tên biến> = **<Tên con trỏ>[**<Chỉ số ô nhớ>**]**;

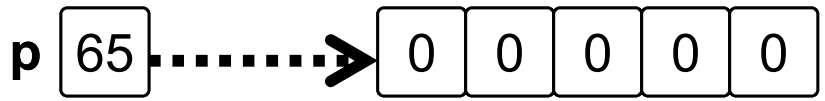
```
int  z = *p;
int  t  = p[2];
```

Mảng động

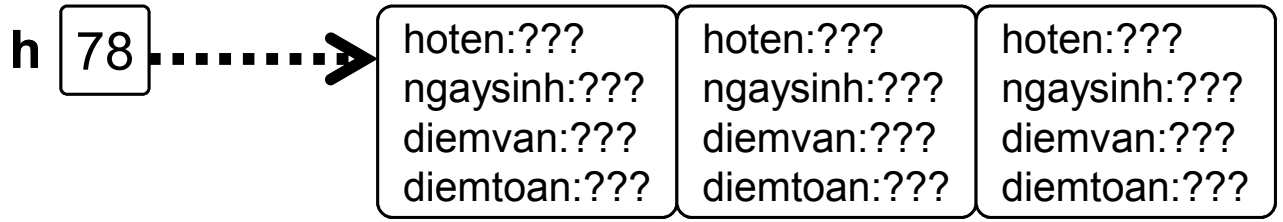


- Con trỏ trong C:
  - Lệnh xin cấp vùng nhớ:
    - Cú pháp: **malloc(**<số byte>**)**;
    - Trả về: địa chỉ vùng nhớ (thành công), NULL (thất bại).
    - Địa chỉ trả về phải cùng kiểu với con trỏ nhận.
  - Lệnh thu hồi vùng nhớ:
    - Cú pháp: **free(**<Tên con trỏ>**)**;

```
int  *p = (int *) malloc( 5 * sizeof( int ) );
```

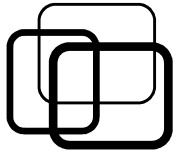



```
HocSinh  *h = (HocSinh *) malloc( 3 * sizeof( HocSinh ) );
```



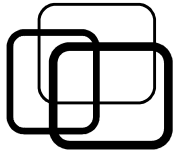

```
free(p);
free(h);
```

Nội dung



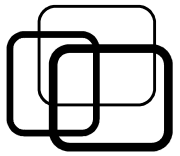
- Mảng động.
- **Đệ quy.**
- Các thuật toán thông dụng.

Đệ quy



- Khái niệm đệ quy:
  - Định nghĩa một khái niệm dựa trên chính nó.
  - Cấu trúc:
    - Phần định nghĩa trường hợp cơ bản (tường minh).
    - Phần định nghĩa đệ quy.
  - Ví dụ: định nghĩa lũy thừa x^n
    - LT(x, 0) = 1.
    - LT(x, n) = x * LT(x, n – 1), n > 0.
  - Ưu điểm:
    - Nêu bản chất vấn đề.
    - Ngắn gọn, dễ hiểu.

Đệ quy

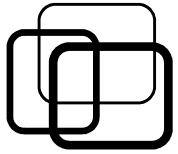


- Hàm đệ quy:
  - Dùng cài đặt khái niệm đệ quy.
  - Cấu trúc:

<Kiểu trả về> **<Tên hàm>(**<Danh sách tham số>**)**
{
if (<Trường hợp cơ bản>)
<Xử lý trường hợp cơ bản>;
else
<Gọi lại hàm đệ quy>;
}

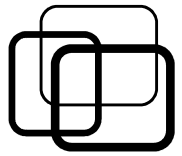
```
long luythua(float  x, int  n)
{
        if ( n == 0 )
            return 1;
        return x * luythua(x, n – 1);
}
```

Nội dung



- Mảng động.
- Đệ quy.
- **Các thuật toán thông dụng.**

Các thuật toán thông dụng



- Bài toán sắp xếp:
  - Nhập vào mảng N số nguyên.
  - Sắp xếp và xuất mảng theo thứ tự tăng dần.

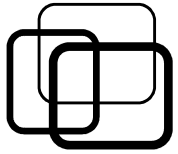
  ➔ Thuật toán Interchange Sort.

  ➔ Thuật toán Selection Sort.

  ➔ Thuật toán Quicksort.

  …

Các thuật toán thông dụng



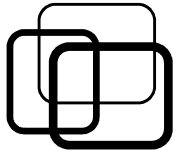
- Bài toán tìm kiếm:
  - Trò chơi đoán số:
    - B1: người chơi nghĩ ra một số nguyên N (1 <= N <= 100).
    - B2: máy đưa ra dự đoán một số nguyên K.
    - B3: người chơi cho biết K lớn hơn, nhỏ hơn hay bằng N.
    - B4: nếu bằng ➔ kết thúc trò chơi.
      ngược lại ➔ quay lại B2.

    Hãy đề ra một thuật toán để số lần đoán của máy là ít nhất.

  ➔ Tìm kiếm tuần tự: O(n).

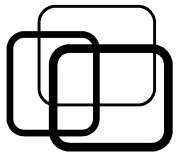
  ➔ Tìm kiếm nhị phân: $O(\log_2 n)$.

Tóm tắt



- Con trỏ:
  - Biến "tham chiếu", "trỏ" đến các ô nhớ.
  - Toán tử &: lấy địa chỉ ô nhớ.
  - Toán tử *: truy xuất ô nhớ.
- Đệ quy:
  - Định nghĩa một khái niệm dựa trên chính nó.
  - Hàm đệ quy: hàm gọi lại chính mình.
- Thuật toán thông dụng:
  - Sắp xếp Interchange Sort.
  - Tìm kiếm nhị phân.

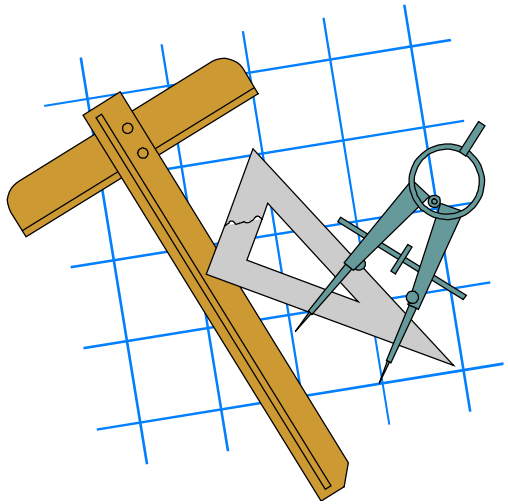
Bài tập



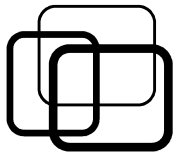
- **Bài tập 10.1:**

  Viết chương trình C cài đặt các thao tác trên mảng động:

  - Nhập mảng N số nguyên từ bàn phím.
  - Chèn một phần tử x vào vị trí i trong mảng.
  - Xóa một phần tử tại vị trí i trong mảng.



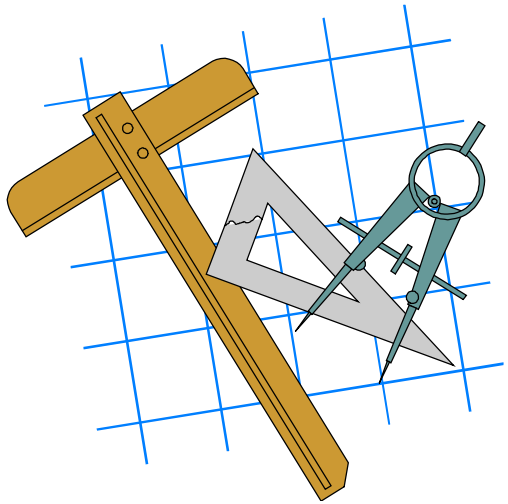
Bài tập



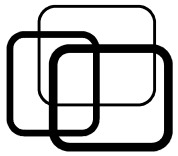
- Bài tập 10.2:

  Viết chương trình C tìm ước chung lớn nhất (sử dụng đệ quy) như sau:

  - Nhập vào hai số nguyên dương a, b.
  - Tìm và xuất ước chung lớn nhất của a, b (thuật toán Euclid).



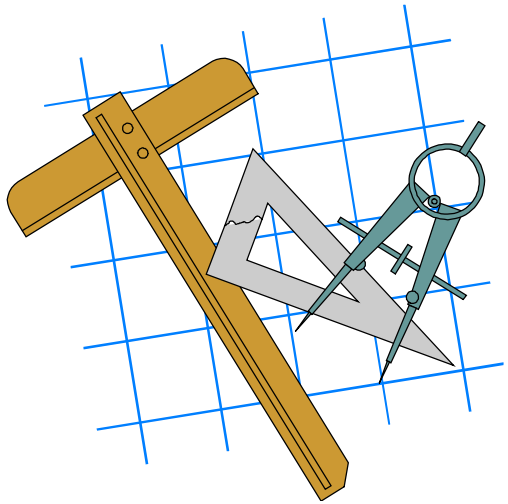
Bài tập



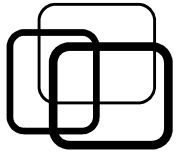
- Bài tập 10.3:

  Viết chương trình C tính dãy Fibonacci (sử dụng đệ quy) như sau:

  - Nhập vào số nguyên N >= 0.
  - Xuất các số Fibonacci từ F_0 đến F_N.



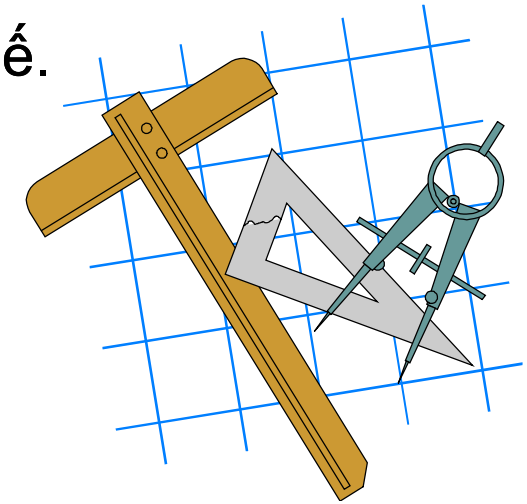
Bài tập



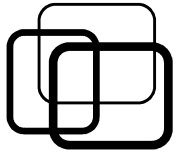
Bài tập 10.4:

Viết chương trình C cài đặt bài toán **"sắp xếp mảng"** trong bài học, dùng thuật toán "Interchange Sort" như sau:

- Nghịch thế:
  - Cặp phần tử đứng không đúng thứ tự.
  - Sắp tăng: cặp (a_i, a_j) nghịch thế $\Leftrightarrow i < j$ và $a_i > a_j$.
- Thuật toán Interchange Sort:
  - Xét tất cả các cặp phần tử mảng.
  - Với mỗi cặp, hoán vị chúng nếu nghịch thế.



Bài tập



Bài tập 10.5:

Viết chương trình C cài đặt **"trò chơi đoán số"** trong bài học, sử dụng thuật toán **"tìm kiếm nhị phân"** như sau:

- B1: đặt [left, right] là đoạn cần tìm.
- B2: **lặp** khoảng cách [left right] > 0

  B2.1: kiểm tra phần tử giữa đoạn.

  B2.2: **nếu** thỏa điều kiện tìm ➔ kết thúc thành công.

  ngược lại

  thu hẹp một nửa [left, right]
  (dựa vào phần tử giữa đoạn).