

**CONFIDENTIAL**

# **C Programming Basic – week 6**

*Searching*

**Lecturer :**

**Do Quoc Huy**

**[huydq@soict.hust.edu.vn](mailto:huydq@soict.hust.edu.vn)**

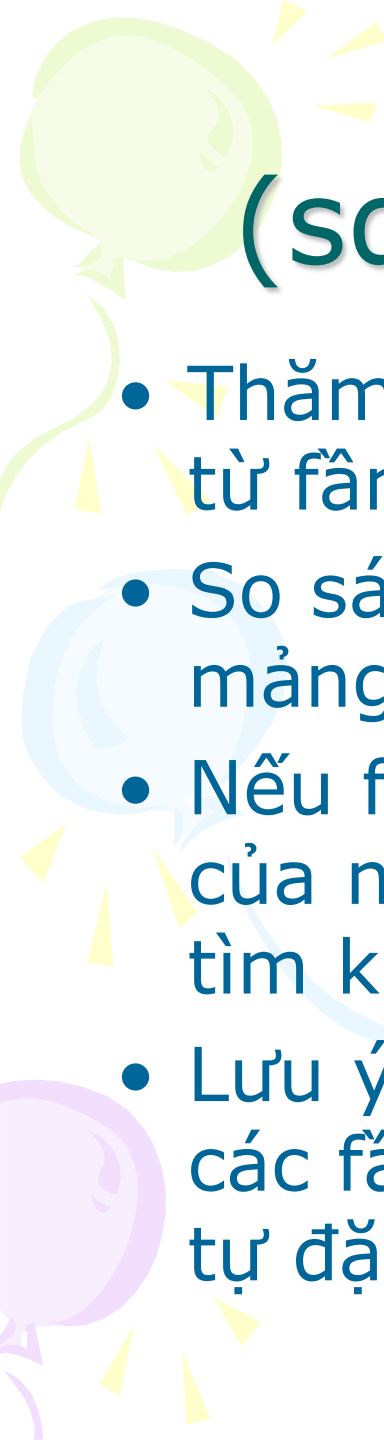
**Dept of Computer Science**

**Hanoi University of Technology**

# Các chủ đề của tuần này

- Các thuật toán tìm kiếm
  - Tìm kiếm tuần tự
  - Sử dụng lính gác
  - Tìm kiếm tự sắp xếp





# Tìm kiếm tuần tự (sequential/linear search)

- Thăm tất cả các phần tử của mảng bắt đầu từ phần tử đầu tiên.
- So sánh **key** với mỗi phần tử của list hoặc mảng.
- Nếu phần tử tìm kiếm được tìm thấy, chỉ số của nó(vị trí trong mảng) được trả về. Nếu tìm kiếm không thành công thì trả về -1.
- Lưu ý rằng tìm kiếm tuần tự không đòi hỏi các phần tử của list phải được đặt theo 1 thứ tự đặc biệt nào.



# Sequential Search

```
int LinearSearch (T M[], int N,  
    T X) {  
    int k = 0;  
    while (M[k] != X && k < N)  
        k++;  
    if (k < N) return (k);  
    return (-1);  
}
```



# Ví dụ

```
#include<stdio.h>
```

```
int sequential_search(char *items, int count, char key)
{
    register int t;

    for(t=0; t < count; ++t)
        if(key == items[t]) return t;
    return -1; /* không tìm thấy */
}
```


```
int main(void){
    char *str = "asdf";

    int index = sequential_search(str, 4, 's');

    printf("%d",index);
}
```



# Lính canh (Sentinel)

- Lưu ý rằng mỗi lần lặp đòi hỏi 2 điều kiện được kiểm tra và 1 câu lệnh được thi hành.
  - Chúng ta có thể tránh kiểm tra cuối mảng trong mỗi bước lặp bằng cách chèn 1 giá trị đích (giá trị mà ta cần tìm) như là 1 phần tử "lính canh" vào cuối của mảng.
  - Ta đặt nó tại vị trí  $n$  và làm theo thuật toán sau:
- 



# Sentinel

- Tìm kiếm tuần tự từ vị trí 0 cho đến khi giá trị đích được tìm thấy. (Giá trị này chắc chắn sẽ được tìm thấy)
- Nếu giá trị được tìm thấy ở vị trí  $n$  thì lính canh đã được tìm thấy, tìm kiếm thất bại.
- Nếu không thì tìm kiếm thành công, trả về chỉ số đầu tiên mà ở đó giá trị đích được tìm thấy.



# Sentinel search

```
int LinearSentinelSearch (T M[],  
    int N, T X) {  
    int k = 0; M[N]=X;  
    while (M[k] != X)  
        k++;  
    return k-1;  
}
```




# Exercise 6-1: tìm kiếm mảng bằng tìm kiếm tuần tự

- Đọc 11 số nguyên từ 1 đầu vào chuẩn và gán 10 số đầu tiên vào mảng.
- Nếu số nguyên thứ 11 có ở trong mảng thì in ra vị trí của nó, nếu không thì in ra 0.



# Exercise 6-2a

- Giả sử rằng bạn viết 1 quyển danh bạ.
  - Khai báo 1 cấu trúc "Address" chứa ít nhất các trường **name**, **telephone number**, **email address**, và viết 1 chương trình có thể thao tác với **100** địa chỉ.
  - Đọc khoảng **10** địa chỉ từ file đầu vào, tìm kiếm 1 name bằng tìm kiếm tuần tự, và ghi dữ liệu phù hợp đầu tiên ra file đầu ra.
  - (1) Triển khai chương trình này sử dụng **mảng cấu trúc**.
- 



# Exercise 6-2b

- (2) Triển khai chương trình này sử dụng danh sách liên kết đơn hoặc đôi.
- Xác nhận cách tìm kiếm thứ (2) đã được tăng tốc bằng cách chuyển dữ liệu phù hợp lên đầu của list. (Tìm kiếm tự tổ chức).

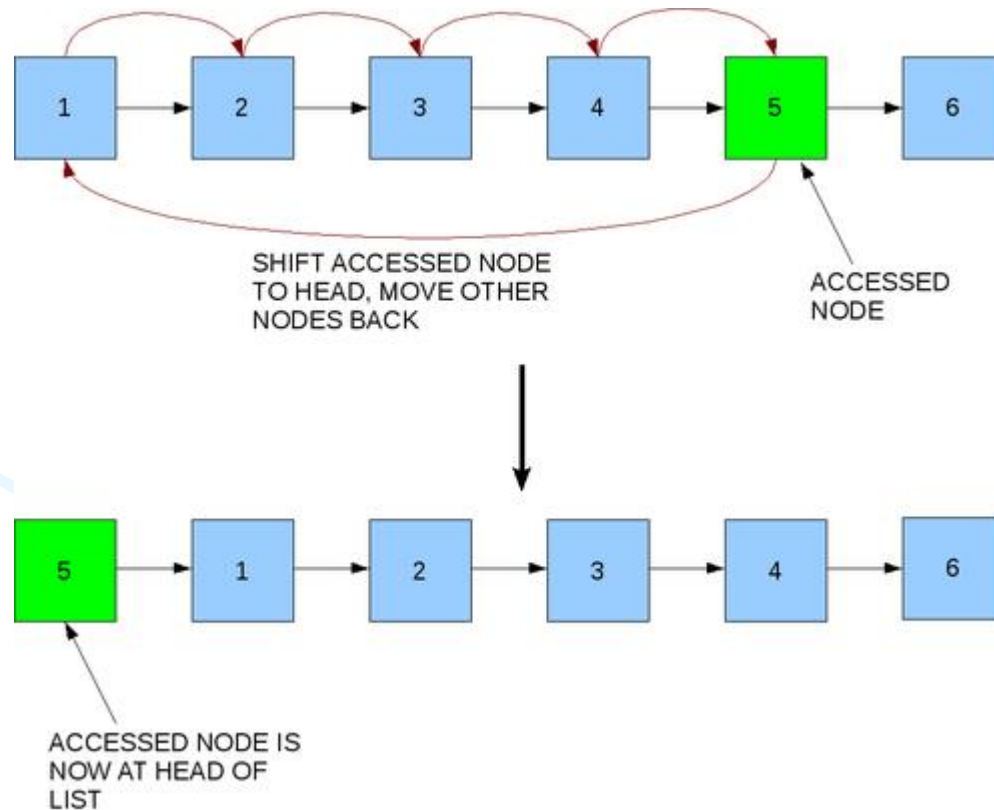


# Tìm kiếm tự sắp xếp (chuyển lên đầu)

- Sự **không** hiệu quả khi duyệt danh sách liên kết
  - Mỗi một nốt trong d/sách sẽ được duyệt và so sánh cho đến khi tìm ra nốt muốn tìm.
- Sự **hiệu quả của danh sách tự sắp xếp**
  - Các nốt được truy cập thường xuyên sẽ được giữ ở đầu danh sách.
  - Tìm một nốt đã được truy cập nhiều lần trước đó sẽ nhanh hơn so với tìm một nốt ít được truy cập. (Ít phải duyệt và so sánh hơn)

# Tìm kiếm tự sắp xếp (chuyển lên đầu)

- Mỗi phần tử được tìm kiếm/yêu cầu được chuyển lên phía trước.

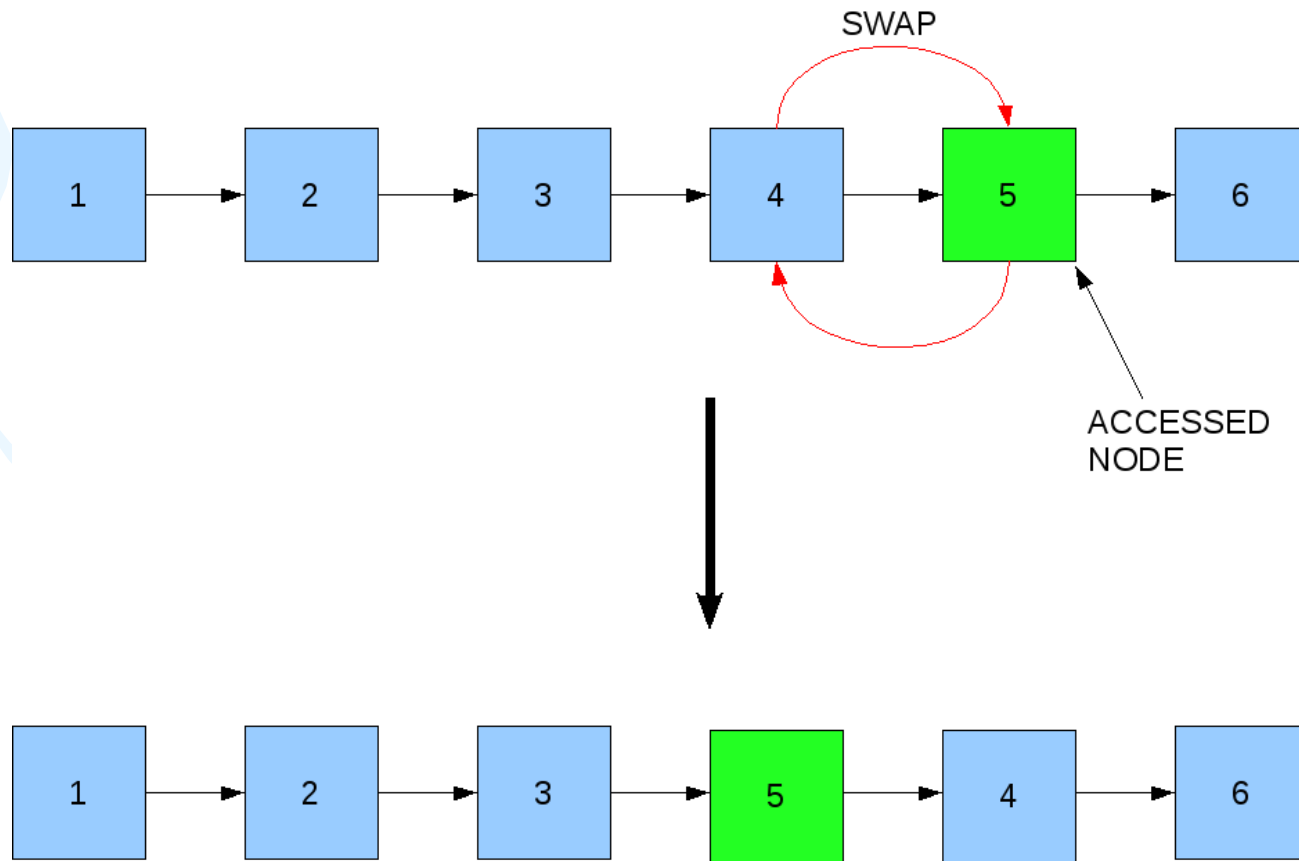


# Tìm kiếm tự sắp xếp(chuyển lên đầu)

```
int search( int key,int r[], int n )
{
    int i,j;
    int tempr;
    for ( i=0; i<n-1 && r[i] != key; i++ )
        if ( key == r[i] )
        { if ( i>0 ) {
                tempr = r[i];
                for (j=0; j<i; j++) r[j+1]=r[j];
                r[0]=tempr;
            };
        }
    return( i );
} else return( -1 );
```

# Tìm kiếm tự sắp xếp (đổi chỗ)

Đổi chỗ phần tử được tìm kiếm với phần tử đứng trước nó





# Tìm kiếm tự sắp xếp (đổi chỗ)

```
int search( int key,int r[], int n )
{
    int i;
    int tempr;
    for ( i=0; i<n-1 && r[i] != key; i++ );
    if ( key == r[i] )
    { if ( i>0 ) {
        /**/ Đổi chỗ với phần tử đứng trước ***/
        tempr = r[i];
        r[i] = r[i-1];
        r[--i] = tempr;
    };
    return( i );
} else return( -1 );
```





## Exercise 6-3: List tự sắp xếp

- Sửa 1 list mà bạn đã tạo ở bài tập trước mà được tự sắp xếp bằng chiến lược “chuyển lên trước”.
- Phát triển hàm tìm kiếm 1 phần tử trong list.



# Exercises 6-4 (Nâng cao)

Tìm kiếm gần đúng

- Tạo một file văn bản trong đó mỗi dòng là một xâu có độ dài  $\leq 30$  kí tự
- Viết chương trình thực hiện công việc sau
- Nhập từ bàn phím một từ cần tìm
- Hiện thị ra màn hình các xâu trong file chứa từ này

Ví dụ: từ nhập vào là computer. Các từ thỏa mãn như: computer, computerize, computerized, computerizes, computerizing, Computers...