## Basic C Programming Bài 6 (Lớp học lần 2)

## Chủ đề của tuần

#### Giải thuật tìm kiếm

- Tuần tự (Sequential)
- Lính canh (Sentinel)
- Tìm kiếm nhị phân



# Tìm kiếm tuần tự (tuyến tính)

- Duyệt tất cả các phần tử trong mảng từ vị trí bắt đầu
- So sánh mỗi phần tử với giá trị cần tìm (key)
- Nếu giống (bằng) nhau thì trả về chỉ số phần tử của mảng
- Nếu duyệt hết mảng mà không tìm thấy, trả về -1
- Không đòi hỏi các phần tử trong mảng sắp xếp có thứ tự

## Tìm kiếm tuần tự

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

## Example

```
#include<stdio.h>
int sequential search(char *items, int count, char key)
   int t;
   for(t=0; t < count; ++t)
     if(key == items[t]) return t;
   return -1; /* no match */
 int main(void) {
    char *str = "asdf";
    int index = sequential search(str, 4, 's');
    printf("%d",index);
```

## Tìm kiếm lính canh

- Mỗi lần lặp, đòi hỏi phải kiểm tra hai điều kiện là K<N và M[K]!=X</li>
- Chúng ta có thể tránh việc kiểm tra duyệt đến cuối mảng chưa bằng việc chèn thêm một phần tử 'lính canh' vào cuối mảng

#### Lính canh

- Tìm kiếm tuần tự từ vị trí 0 cho đến khi tìm thấy giá trị cần tìm (chắc chắn là tìm được)
- Nếu giá trị tìm thấy ở vị trí n => phần tử lính canh => trả về 'failed'
- Ngược lại ở một vị trí <n chúng ta tìm thấy</li>
   => trả về chỉ số này

## Tìm kiếm lính canh

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

- Xem lại chương trình cài đặt Queue bằng mảng và danh sách liên kết ở Tuần 5 với kiểu dữ liệu thành phần là kiểu số nguyên
- Với mỗi cách cài đặt thực hiện công việc sau
  - Nhập dữ liệu vào Queue
  - Nhập một số nguyên để tìm kiếm
  - Cho biết trong Queue có số nguyên này hay không

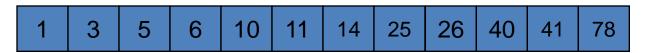
#### Bài 1

 Viết chương trình cho phép nhập vào một mảng các số nguyên từ bàn phím. Sau đó nhập vào một số nguyên X để tìm kiếm. Hãy hiển thị ra chỉ số tất cả các phần tử trong mảng có giá trị X. Nếu không thấy, in ra thông báo không tìm được. Sử dụng giải thuật tìm kiếm tuần tự.

#### Bài 2. Tìm kiếm gần đúng

- Tạo một file văn bản trong đó mỗi dòng là một xâu có độ dài <=30 kí tự</li>
- Viết chương trình thực hiện công việc sau
- Nhập từ bàn phím một từ cần tìm
- Hiến thị ra màn hình các xâu trong file chứa từ này
  - Ví dụ: từ nhập vào là computer.
  - Các từ thỏa mãn như: computer, computers, super computer...

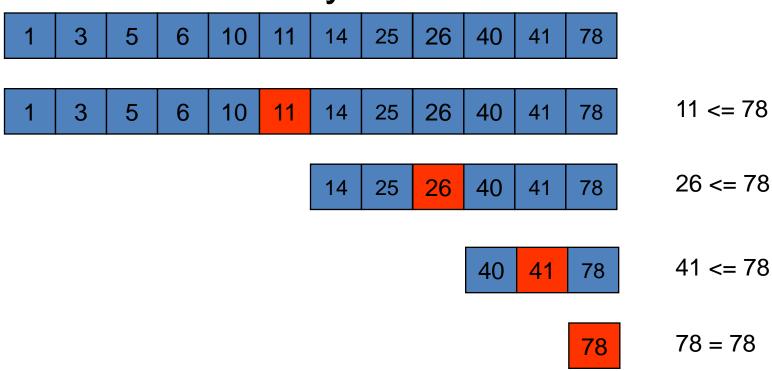
## Binary Search (Tìm kiếm nhị phân)



- Sử dụng chiến lược chia và trị
- Đòi hỏi danh sách có thứ tự (không tăng hoặc không giảm)
- Đầu tiên so sánh giá trị cần tìm (key) với phần tử đứng giữa danh sách
- Nếu bằng nhau => tìm kiếm thành công
- Nếu key nhỏ hơn => tìm kiếm nửa đầu của danh sách
- Ngược lại (key lớn hơn) =>tìm kiếm nửa sau của danh sách

#### Ví dụ

Tìm kiếm cho key=78



4 lần so sánh được sử dụng để đưa ra kết quả Số lần so sánh với giải thuật tìm kiếm tuần tự?

## Binary Search Code

```
int binSearch(int List[], int Target, int Size) {
   int
         Mid.
         Lo = 0,
         Hi = Size - 1;
   while (Lo <= Hi) {
          Mid = (Lo + Hi) / 2;
          if ( List[Mid] == Target )
                    return Mid;
          else if ( Target < List[Mid] )</pre>
                    Hi = Mid - 1;
          else
                    Lo = Mid + 1:
   return -1;
```

#### Test Program

```
#include <stdio.h>
#define NotFound (-1)
typedef int ElementType;
int BinarySearch(ElementType A[], ElementType X, int N ) {
    int Low, Mid, High;
    Low = 0; High = N - 1;
    while( Low <= High ) {
          Mid = (Low + High) / 2;
          if (A[Mid] < X)
                    Low = Mid + 1:
         else if( A[ Mid ] > X )
                    High = Mid - 1;
         else
                     return Mid: /* Found */
   return NotFound; /* NotFound is defined as -1 */
main(){
  static int A[] = \{1, 3, 5, 7, 9, 13, 15\};
  int SizeofA = sizeof(A) / sizeof(A[0]);
  int i:
  for(i = 0; i < 20; i++)
     printf( "BinarySearch of %d returns %d\n", i, BinarySearch( A, i, SizeofA ) );
  return 0;
```

#### Bài 3.

 Cài đặt giải thuật tìm kiếm nhị phân dưới dạng đệ quy

#### Lời giải

```
#define NotFound (-1)
typedef int ElementType;
int BinarySearch(ElementType A[], ElementType X, int Lo, int Hi ) {
   if (Lo > High) return NotFound;
   Mid = (Low + High) / 2;
  if (A[Mid] < X) return BinarySearch(A, X, Mid+1, Hi);
   else if (A[Mid] > X)
                return BinarySearch(A, X, Lo, Mid – 1);
  else
                return Mid; /* Found */
   return NotFound; /* NotFound is defined as -1 */
Usage: BinarySearch(A, X, 0, size -1);
```

## Thứ tự từ điển

- Khi chúng ta so sánh hai xâu, việc so sánh dựa trên thứ tự trong bảng mã ASCII (hay trong từ điển)
- Chúng ta có:
  - 'a' < 'd', 'B' < 'M'
  - "acerbook" < "addition"</p>
  - "Chu Trong Hien" > "Bui Minh Hai"
- Sử dụng hàm strcmp

## Bài tập 4

- Thông tin về một điện thoại gồm có:
  - Tên điện thoại là xâu tối đa 30 kí tự (Nokia 5530, Iphone 4...)
  - Giá điện thoại kiểu long như 5.000.000,
    10.000.000...
  - Đánh giá trên thang điểm 10 như Nokia 5530
     được 8 điểm, Iphone4 được 9 điểm..., kiểu số thực
- Viết chương trình có giao diện menu thực hiện các công việc sau

- Chọn 1. Cho phép nhập từ bàn phím thông tin điện thoại, cho đến khi tên điện thoại là \$\$\$ thì dừng lại. Lưu toàn bộ thông tin điện thoại ra file nhị phân.
  - (không sử dụng mảng để lưu trữ ở bước này)
- Chọn 2. Đọc lại file nhị phân và lưu vào một mảng
- Chọn 3. Sắp xếp mảng theo thứ tự tăng dần của tên điện thoại
- Chọn 4. Nhập vào một tên cần tìm, in ra giá và điểm đánh giá điện thoại đó. Thực hiện tìm kiếm tuần tự và nhị phân
- Chọn 5. Kết thúc

## Bài tập về nhà

- Bài tập 4
- Bài 30 (file bài tập)