C Programming Basic – week 6

Why Search ?

- Daily activity – yellow pages, universities, hairdressers
- Computers can search for us
- World wide web – different searching mechanisms, yahoo.com, ask.co.uk, google.com
- Spreadsheet –list of names – searching mechanism to find a name
- Databases – use to search for a record - *select* * *from* ...
- Large records –1000s takes time - many comparison slow system – user won't wait long time

Content

1. Sequential search
2. Self-organizing searching
3. Binary search

1. Sequential search

- (Linear search)
- Visit all the elements of array from the beginning
- Compare the key with each element of a list
- If the search item is found, its index is returned. If search is unsuccessful, -1 is returned
- List elements are not necessary to be in any particular order

Algorithm

```
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;
}
```

Example

```
#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; /* no match */
  }

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

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

     printf("%d",index);

  }
```

Sentinel

- In sequential search, each iteration requires
  - Two conditions to be checked
  - One statement to be executed
- We can avoid checking for the end of the array on every iteration by inserting the target as an extra ‘sentinel’ element at the end of the array.

Algorithm

- Search sequentially from position 0 until the target is found (it will definitely be found).
- If the target is found in position *n* then the sentinel has been found – search has 'failed',
- else search was successful, return first index where target was found.

Algorithm (cont)

```
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

- Mobile phone address book.
- Declare a structure "Address" that can hold at least name, telephone number, and e-mail address, the program can handle 100 addresses
- Read 10 addresses from an input file, search a name by sequential search, and write the first matched address to an output file.
  (1) Use an array of structure.
  (2) Use singly-linked list or doubly-linked list

Exercise 6.2

- Read 11 integers from the standard input and assign first ten integers to an array.
- If the 11th integer is in the array, output the index of the element. If not, output -1.

List verification

- Compare lists to verify that they are identical or identify the discrepancies.
- example
  - international revenue service (e.g., employee vs. employer)
- complexities
  - random order: O(m.n)
  - ordered list: O(tsort(n)+tsort(m)+m+n)

Exercise 6.3

- Given two lists whose elements are in the same type. Find :

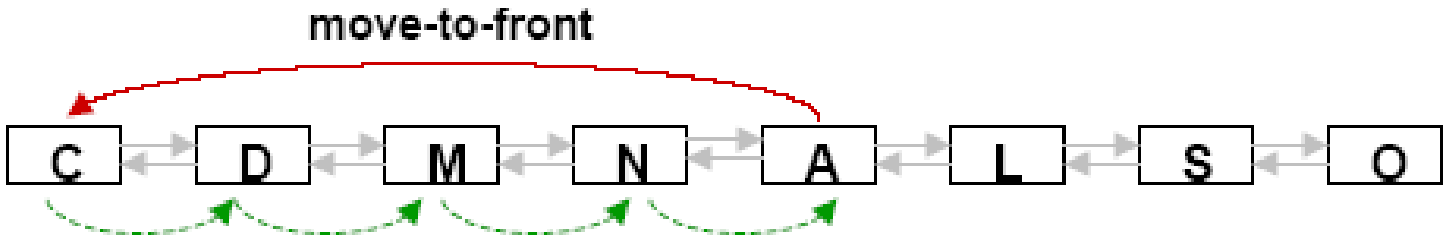
(a) all records found in list1 but not in list2

(b) all records found in list2 but not in list1

(c) all records that are in list1 and list2 with the same key but have different values for different fields.

2. Self-organizing search

- Move-to-front
- Transpose



Move-to-front

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

Transpose

```
int search(int key, int r[], int n ) {
   int i;
   int temp;
   for (i = 0; i < n && r[i] != key; i++);
   if (key == r[i]) {
    if (i > 0) {
          /*** Transpose with predecessor ***/
          temp = r[i];
          r[i] = r[i-1];
          r[--i] = temp;
      };
      return i;
   }else return -1;
}
```

Exercise 6.4

- Modify search function in the list of Exercise 6.1 as self-organizing search
  (1) using move-to-front strategy
  (2) using transpose strategy

Homework 1

- Implement move-to-front and transpose strategy for List library.
- Apply to Nokia DB problem :
  - Search and update mobile by model
  - Use menu from previous exercises.

Homework 2

- A dictionary is stored at /usr/share/dict/words
- Write a program that takes a character sequence and output all the words in the dictionary beginning with the sequence

[Example]

```
% look computer
computer
computerize
computerized
computerizes
computerizing
computers
```