C/C++ PROGRAMMING

Search an Array

(Sequential Search)

Dan McElroy

Search Methods

There are several ways to search a block of data. The two most popular are:

- Sequential – Start at the beginning of the data and look for a match. If none is found, move to the next piece of data and look again.
- Binary Search – The data must be sorted first. Start in the middle of the data and see if the search value is too high or too low. If your value is too low, divide the data lower half of the data in half again and try again. Etc.

Samples of Search Methods

Binary Search: Suppose you are looking for the name Jones in the phone book.

If you open the phone book n the middle at the M's, you have gone too far. Divide the lower half of the book and you may end up at the F's. Oops, you are now too low. Divide what is left and you may end up at the K's. Now you are too high, divide what is left and you end up at the J's.

Sequential Search: Now you are at the J's, go down the list one name at a time until you find the name Jones that you are looking for.

Sequential Search

This discussion covers only the sequential search. A disk text file is read into an array. The user is asked for a two-character State Abbreviation (e.g. CA) and the program searches the array and displays a match if found, or a message saying that no match was found.

Know the Data

Know what the data looks like before writing the program. The text file to be read is named "States.txt" and contains a two-character abbreviation for the state, commonwealth or federal district, a tab character and the name of the state. The name of the state may contain one or more characters.

There are 51 lines, each less than 100 characters.

```
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DE	Delaware
DC	District of Columbia
FL	Florida
GA	Georgia
HI	Hawaii
ID	Idaho
IL	Illinois
IN	Indiana
IA	Iowa
KS	Kansas
KY	Kentucky
LA	Louisiana
ME	Maine
MD	Maryland
MA	Massachusetts
MI	Michigan
MN	Minnesota
MS	Mississippi
MO	Missouri
MT	Montana
NE	Nebraska
NV	Nevada
NH	New Hampshire
```

Make It Simple

To make this first discussion simple, just read each line into an element of an array and display the entire line if a match on the abbreviation is found.

```
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DE	Delaware
DC	District of Columbia
FL	Florida
GA	Georgia
HI	Hawaii
ID	Idaho
IL	Illinois
IN	Indiana
IA	Iowa
KS	Kansas
KY	Kentucky
LA	Louisiana
ME	Maine
MD	Maryland
MA	Massachusetts
MI	Michigan
MN	Minnesota
MS	Mississippi
MO	Missouri
MT	Montana
NE	Nebraska
NV	Nevada
NH	New Hampshire
```

Create an Array in Memory

From looking at the data, it looks like there are 51 lines and the longest line is 26 characters for the District of Columbia.

The minimum array size would be [51][27]. Make sure there is room for the NULL byte at the end of each line.

To be save, set the array size to hold up to 60 lines, 30 char max each line.

```
AL       Alabama
AK       Alaska
AZ       Arizona
AR       Arkansas
CA       California
CO       Colorado
CT       Connecticut
DE       Delaware
DC       District of Columbia
FL       Florida
GA       Georgia
HI       Hawaii
ID       Idaho
IL       Illinois
IN       Indiana
IA       Iowa
KS       Kansas
KY       Kentucky
LA       Louisiana
ME       Maine
MD       Maryland
MA       Massachusetts
MI       Michigan
MN       Minnesota
MS       Mississippi
MO       Missouri
MT       Montana
NE       Nebraska
NV       Nevada
NH       New Hampshire
```

HIPO Chart

INPUT	PROCESSING	OUTPUT
States file User Input: 2-char abbrev	1) Open the States file 2) Read the file into an array 3) Ask user for abbreviation 4) Search array for abbreviation	Name of State --or— "Not Found"

The program will be written in four parts not including the information for the header files and declaration of variables. A separate function will be used to search the array for a match for the state abbreviation.

Header Files & Variables

```
// SequentialSearch.cpp
// CIS-054 C/C++ Programming
// Dan McElroy

#include <iostream> // for cin and cout
#include <fstream>  // for file access
#include <cctype>   // for toupper()
using namespace std;

// prototypes for functions declared later in the project
char* SearchStates(char States[60][30], int length, char ch1, char ch2);

int main(int argc, char* argv[])
{
   ifstream stateFile;           // connection to the disk
   char listOfStates[60][30];    // room for 60 lines, 30 char each
   int  linesInFile;             // lines in the text file
   char abbrev1, abbrev2;        // two char state abbreviation
   char *foundMsg;               // line with State or "Not Found"
   char tryAgain;
```

Part 1 – Open the File

```
// ----- PART 1 open the file, check for errors
stateFile.open("/Users/Dan/Desktop/States.txt");
if (stateFile.fail())
{
    cout << "Unable to open States.txt" << endl;
    return 1;
}
```

Part 2 – Read the File

```
// ----- PART 2 read the file into the listOfStates array
linesInFile = 0;  // set array index to 0
stateFile.getline(listOfStates[linesInFile], 30); // 1st record
while (linesInFile<60 &&
       !stateFile.eof() &&
       listOfStates[linesInFile]!=0)
{
  linesInFile++;    // successful read
  stateFile.getline(listOfStates[linesInFile], 30); // next
}
stateFile.close();  // all done with the file, close it
```

Part 3 – Ask for Abbreviation, Search

```
// ----- PART 3 ask for an abbreviation and search the array
do
{
   cout << "Enter a two character state abbreviation: ";
   cin  >> abbrev1 >> abbrev2;
   abbrev1 = toupper(abbrev1);     // convert to upper case
   abbrev2 = toupper(abbrev2);

   // ----- PART 4 search the array
   foundMsg = SearchStates(listOfStates, linesInFile,
                           abbrev1, abbrev2);
   cout << foundMsg << endl;

   cout << endl << "Do you want to look for again? ";
   cin  >> tryAgain;
} while (tryAgain=='y'  ||  tryAgain=='Y');
```

Part 4 – Search Array Function

```
// Search List of States Array for a two-character abbreviation
//   returns:  line containing abbreviation, or "Not Found"
char* SearchStates(char States[60][30], int length,
                    char ch1, char ch2)
{
  int i = 0;      // start at the first element in the array
  while (i < length)
  {
      // see if the abbreviation is first two char on line
      if (ch1==States[i][0] && ch2==States[i][1])
         return States[i];     // match was found, return it
      i++;  // move to the next elemen, keep looking
  }
  // reached the end and did not find a match
  return "Not Found";
}
```