APS105: Lecture 20

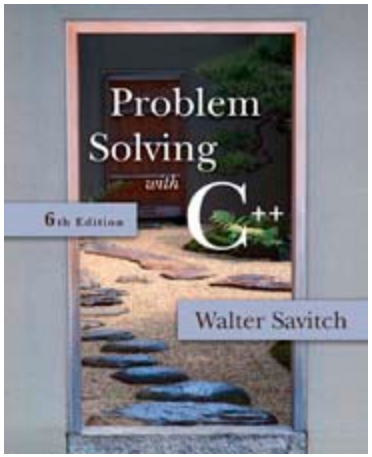
Wael Aboelsaadat

wael@cs.toronto.edu

http://ccnet3.utoronto.ca/20079/aps105h1f/

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley

Download the code shown in lecture from course website:
Handouts ➔ Lectures Source Code - Wael



Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Searching Arrays

- A sequential search is one way to search an array for a given value
  - Look at each element from first to last to see if the target value is equal to any of the array elements
  - The index of the target value can be returned to indicate where the value was found in the array
  - A value of -1 can be returned if the value was not found

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The search Function

- The search function of Display 7.10…
  - Uses a while loop to compare array elements to the target value
  - Sets a variable of type bool to true if the target value is found, ending the loop
  - Checks the boolean variable when the loop ends to see if the target value was found
  - Returns the index of the target value if found, otherwise returns -1

Display 7.10 (1) **Display 7.10 (2)**

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Program Example: Sorting an Array

- Sorting a list of values is very common task
  - Create an alphabetical listing
  - Create a list of values in ascending order
  - Create a list of values in descending order
- Many sorting algorithms exist
  - Some are very efficient
  - Some are easier to understand

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Bubble sort

```
void bubbleSort(int arrNumbers[], int length)
{
  int i, j, temp;

  bool bContinue = true;

  while( bContinue == true )
  {
    bContinue=false;
    for(j = 0; j < (length-1); j++)
    {
      if(arrNumbers[j] > arrNumbers[j+1]) /* compare neighboring elements */
      {
        temp = arrNumbers[j];    /* swap array[j] and array[j+1] */
        arrNumbers[j] = arrNumbers[j+1];
        arrNumbers[j+1] = temp;
        bContinue = true;
      }
    } /*end for j*/

  }

}
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Display 7.10 (1/2)

Back Next

Searching an Array (*part 1 of 2*)

```
//Searches a partially filled array of nonnegative integers.
#include <iostream>
const int DECLARED_SIZE = 20;

void fill_array(int a[], int size, int& number_used);
//Precondition: size is the declared size of the array a.
//Postcondition: number_used is the number of values stored in a.
//a[0] through a[number_used-1] have been filled with
//nonnegative integers read from the keyboard.

int search(const int a[], int number_used, int target);
//Precondition: number_used is <= the declared size of a.
//Also, a[0] through a[number_used -1] have values.
//Returns the first index such that a[index] == target,
//provided there is such an index; otherwise, returns -1.

int main( )
{
    using namespace std;
    int arr[DECLARED_SIZE], list_size, target;

    fill_array(arr, DECLARED_SIZE, list_size);

    char ans;
    int result;
    do
    {
        cout << "Enter a number to search for: ";
        cin >> target;

        result = search(arr, list_size, target);
        if (result == -1)
            cout << target << " is not on the list.\n";
        else
            cout << target << " is stored in array position "
                 << result << endl
                 << "(Remember: The first position is 0.)\n";

        cout << "Search again?(y/n followed by Return): ";
        cin >> ans;
    }while ((ans != 'n') && (ans != 'N'));

    cout << "End of program.\n";
    return 0;
}
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Display 7.10 (2/2)

Back Next

Searching an Array (*part 2 of 2*)

```
//Uses iostream:
void fill_array(int a[], int size, int& number_used)
<The rest of the definition of fill_array is given in Display 10.9.>

int search(const int a[], int number_used, int target)
{

    int index = 0;
    bool found = false;
    while ((!found) && (index < number_used))
        if (target == a[index])
            found = true;
        else
            index++;

    if (found)
        return index;
    else
        return -1;
}
```

Sample Dialogue

```
Enter up to 20 nonnegative whole numbers.
Mark the end of the list with a negative number.
10 20 30 40 50 60 70 80 -1
Enter a number to search for: 10
10 is stored in array position 0
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 40
40 is stored in array position 3
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 42
42 is not on the list.
Search again?(y/n followed by Return): n
End of program.
```

Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley