CMPSC 122 Lab 16 Report

Code

// Programmer: Yizhou Wang

// Section: 2

// Lab: 16

// Date: Apr. 16, 2014

// Description: Implementation of bubble sort

#include <iostream>

using namespace std;

void swap(int & a, int & b)

//PRE: both a and b are initialized

//POST: a and b exchange their value

{

//DATA DICTIONARY

int temp; //a temporary variable storing the data being swapped

temp = a;

a = b;

b = temp;

}

void traversal(int A[], int n)

//PRE: n > 0 and A[0..n-1] is initialized

//POST: the elements of A[] are printed to the console in the order of indices

{

for (int i = 0; i < n; i ++)

{

cout << A[i] << " ";

}

cout << endl;

}

void bubble\_sort (int A[], int n)

//PRE: n > 0 and A[0..n-1] is initialized

//POST: A[0..n-1] is sorted in ascending order

{

//At the "bottom"th iteration, A[bottom+1..n-1] is the subarray that has been sorted, A[0..bottom]

// is the subarray to be sorted. Two consecutive elements form one comparison,

// we need (n-1)-1+1 = n-1 comparisons to move n-1 larger elements to bottom in ascending order

for (int bottom = n-1; bottom>=1; bottom--)

{

//for the ith iteration, we need (bottom-1)-i+1 = bottom-i comparisons to move

// the largest element in A[0..bottom-1] to the position A[bottom]

for (int i=0; i<=bottom-1; i++)

{

if (A[i+1] < A[i]) //if A[i+1] < A[i], swap A[i+1] and A[i]

{

swap(A[i+1], A[i]);

}

}

}

}

int main()

{

//DATA DICTIONARY

int array1[8];

int array2[10];

int array3[15];

//initialization

array1[0] = 5;

array1[1] = 3;

array1[2] = 7;

array1[3] = 9;

array1[4] = 25;

array1[5] = 42;

array1[6] = 99;

array1[7] = 35;

array2[0] = 10;

array2[1] = 5;

array2[2] = 50;

array2[3] = 35;

array3[0] = 100;

array3[1] = 500;

array3[2] = 300;

array3[3] = 400;

array3[4] = 200;

//Test1

cout << "\*\*\*Test1: The original array1 is:" << endl;

traversal(array1, 8);

bubble\_sort(array1, 8);

cout << "After bubble sort, the array1 turns into:" << endl;

traversal(array1, 8);

cout << endl;

//Test2

cout << "\*\*\*Test2: The original array2 is:" << endl;

traversal(array2, 4);

bubble\_sort(array2, 4);

cout << "After bubble sort, the array2 turns into:" << endl;

traversal(array2, 4);

cout << endl;

//Test3

cout << "\*\*\*Test3: The original array3 is:" << endl;

traversal(array3, 5);

bubble\_sort(array3, 5);

cout << "After bubble sort, the array3 turns into:" << endl;

traversal(array3, 5);

cout << endl;

return 0;

}

Sample Runs

\*\*\*Test1: The original array1 is:

5 3 7 9 25 42 99 35

After bubble sort, the array1 turns into:

3 5 7 9 25 35 42 99

\*\*\*Test2: The original array2 is:

10 5 50 35

After bubble sort, the array2 turns into:

5 10 35 50

\*\*\*Test3: The original array3 is:

100 500 300 400 200

After bubble sort, the array3 turns into:

100 200 300 400 500

Program ended with exit code: 0

Discussion

As the pseudocode was given, it was not hard to implement bubble sort. However, I had a hard time figuring out that the array indices are different between pseudocode and C++ code, as the former uses 1-based array, while the latter uses 0-based array.

On the other hand, it is challenging to describe the dynamic process of sorting in words, especially without the help of graphs.