CMPSC 122 Lab 17 Report (revision on April 30, 2014)

Code

// Programmer: Yizhou Wang

// Section: 2

// Lab: 17

// Date: Apr. 21, 2014

// Description: Implementation of insertion sort

#include <iostream>

using namespace std;

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

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

//POST: A[0..n-1] contains all elements from A[0..n-1] at the time of invocation and

// is sorted in ascending order and

{

//DATA DICTIONARY

int key; //the value of A[j] which is to be inserted

int i; //the index of iteration representing the

// position of the element whose value is being

// inspected

//Before the iteration indexed j, A[0..j-1] contains values from A[0..j-1] at time of invocation,

// now sorted in ascending order

for (int j=1; j<=n-1; j++) //take each element of A[0..n-1] and insert it

// into A[0..j-1] where it goes

{

//initialization

key = A[j];

i = j-1; //shift A[0..j-1] s.t. there is an appropriate

// slot in A[0..j] for key

//For the iteration indexed i, we compare A[i] with key. If A[i] is greater than key, we

// shift A[i] to A[i+1]. We keep doing this with i-1 iteratively until we find the right

// position to insert key (the position of A[i+1] such that A[i] =< key and A[i+1] > key or

// the position of A[0])

while (i>-1 && A[i]>key)

{

A[i+1] = A[i]; //shift A[i] to A[i+1]

i = i-1;

}

A[i+1] = key; //insert key at A[i+1]

}

}

void traversal(int A[], int n)

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

//POST: all the elements of A[0..n-1] are printed to the console in a sequential order with a space

// in between

{

for (int i = 0; i < n; i ++) //traverse the array A[0..n-1] and print

// elements spaced by a whitespace

{

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

}

cout << endl;

}

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] = 5;

array2[1] = 10;

array2[2] = 35;

array2[3] = 50;

array3[0] = 500;

array3[1] = 400;

array3[2] = 300;

array3[3] = 200;

array3[4] = 100;

//Test1

cout << "\*\*\*Test1: The original array1 (mixed order) is:" << endl;

traversal(array1, 8);

insertion\_sort(array1, 8);

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

traversal(array1, 8);

cout << endl;

//Test2

cout << "\*\*\*Test2: The original array2 (ascending order) is:" << endl;

traversal(array2, 4);

insertion\_sort(array2, 4);

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

traversal(array2, 4);

cout << endl;

//Test3

cout << "\*\*\*Test3: The original array3 (descending order) is:" << endl;

traversal(array3, 5);

insertion\_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 (mixed order) 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 (ascending order) is:

5 10 35 50

After bubble sort, the array2 turns into:

5 10 35 50

\*\*\*Test3: The original array3 (descending order) is:

500 400 300 200 100

After bubble sort, the array3 turns into:

100 200 300 400 500

Discussion

It is still an headache to write comments for outer and inner loops. Sometimes the loop invariant can be written to facilitate maintenance. Lessons learned: never declare variables within loop body.