CS211 Programming and Operating Systems

Lab 2: functions

08 March 2021

The goal of this week's lab is to get you started writing functions in C. We will practice using the `if` and `for` expressions by implementing the Selection Sort and a **Bubble Sort** algorithms.

If possible, use `Code::Blocks` (for Windows or Linux) or `xcode` (for Mac). But it will suffice to use an online compiler, such as

- https://www.onlinegdb.com/online_c_compiler
- http://www.tutorialspoint.com/compile_c_online.php
- http://cpp.sh/
- https://www.codechef.com/ide

Selection Sort

The idea is: rearrange a list of n integers $\{a_0, a_1, a_2, \ldots, a_{n-1}\}$ so that $a_0 \leq a_1 \leq a_2 \leq \cdots \leq a_{n-1}$. (*Remember*: C indexes its arrays from 0, so the list a of n elements is $a[0]$, $a[1]$, ..., $a[n-1]$).

The **Selection Sort** algorithm is one of the simplest (and slowest) algorithms for doing this.

- Find the index of the smallest entry in $\{a_0, a_1, a_2, \ldots, a_{n-1}\}$, and swap a_0 with the entry with that index. When this is done, the (new) a_0 is the smallest element in the list. (Note: this uses the `Swap()` function from Week 4).
- Next find the index of the smallest entry in $\{a_1, a_2, \ldots, a_{n-1}\}$, and swap a_1 with the entry with that index. When this is done, the (new) a_1 is the second smallest element in the list.
- Repeat the process until we have sorted the entire list.

Selection Sort

```
FOR $i = 0, 1, \ldots, n-2$
    $MinIndex = i$
    FOR $j = i+1, \ldots, n$
      IF $a_j < a_{MinIndex}$
        $MinIndex = j$
      END IF
    $Swap(a_i, a_{MinIndex})$
    END FOR
END FOR
```

Selection Sort

This is implemented in the `Selection.c` program, which you can download from `http://www.maths.nuigalway.ie/~niall/CS211/lab2`.

Q1 (a) Compile and run the `SelectionSort.c`. Make sure you understand each line of code, and how the programme works.

(b) Adjust the code so that, instead of running on a list of 8 elements between 0 and 30, it prompts the user for

- The total number of elements in the list. You may assume that this is at most 30.
- The maximum value that any element can have.

(c) Modify the code so that a count is kept of the total number of "swaps" that are made. This should be reported when the program finishes running.

Bubble Sort

The **Bubble Sort** algorithm works as follows:

- Compare a_0 with a_1 and swap them if they are not in order. Now compare a_1 and a_2, and swap them if they are not in order. Repeat until we have compared a_{n-2} with a_{n-1}, and swapped them in necessary.
- When this is done, we should have that the largest element is in a_{n-1} (it will have "bubbled" to the top of the list).
- We continue by applying the process to the set $\{a_0, a_1, a_2, \dots, a_{n-2}\}$. At the end of that the second largest element will be in a_{n-2}.
- Now apply the process to the set $\{a_0, a_1, a_2, \dots, a_{n-3}\}$, etc.

Bubble Sort

```
FOR $i = 0, 1, \dots, n-2$
    FOR $j = 0, 1, \dots, (n-i-1)$
        IF $a_{j+1} < a_j$ THEN
            $Swap(a_j, a_{j+1})$
        END IF
    END FOR
END FOR
```

Q2 Write a C program that implements this and check if it is more efficient than the linear sort method: i.e., it requires less swapping of values.

Functions

Q3 Write a programme that contains functions with the headers

```
        int SelectionSort(int *a, int n);   and
        int BubbleSort(int *a, int n);
```

These take as their arguments an integer array, `a`, of length `n`, and sorts their entries in ascending order using, respectively, the Selection Sort and Bubble Sort algorithms. Both return the number of `Swap`s that was preformed.

Assignment

Submit a programme that contains the `SelectionSort` and `BubbleSort` functions, along with a `main` function that calls them for *copies* the same list of random integers. The output from both sortings should be displayed, along with the number of swaps that each preformed, and a statement as to which required fewer swaps.

Submit your code to the `Lab 2` section on Blackboard, **no later than 12:00 (noon), Monday 15 March**.

The code **must** include

(i) comments at the start that include your name, ID number, and email address;

(ii) a short description of what the program does, written in your own words;

(iii) the name and email address of any person you collaborated with on the assignment, and a statement of what each of you contributed.