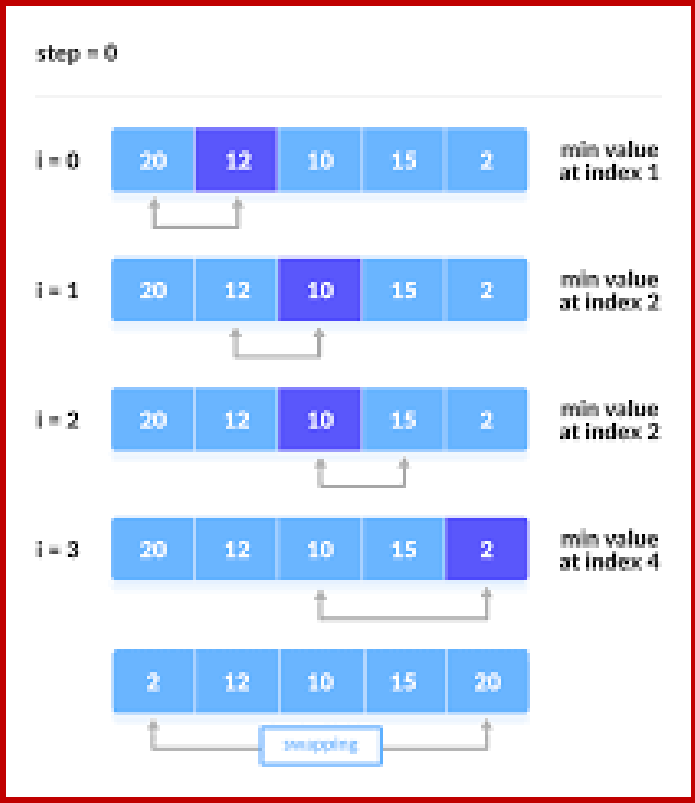
Homework #5



ECE 330 – Function Template for SelectionSort() Report

Spring 2021

Author: Clarizza Morales

Date: March 12th, 2021

Instructor: Amir Raeisi

Table of Contents

List of Figures

SelectionSort() With Pass-By-Reference: Base Code

```
// Fig. 8.13: fig08_13.cpp
// Selection sort with pass-by-reference. This program puts values into an
// array, sorts them into ascending order and prints the resulting array.
#include <iostream>
#include <iomanip>
using namespace std;

void selectionSort( int * const, const int ); // prototype
void swap( int * const, int * const ); // prototype

int main()
{
   const int arraySize = 10;
   int a[ arraySize ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };

   cout << "Data items in original order\n";

   for ( int i = 0; i < arraySize; ++i )
      cout << setw( 4 ) << a[ i ];

   selectionSort( a, arraySize ); // sort the array

   cout << "\nData items in ascending order\n";

   for ( int j = 0; j < arraySize; ++j )
      cout << setw( 4 ) << a[ j ];

   cout << endl;
} // end main

// function to sort an array
void selectionSort( int * const array, const int size )
{
   int smallest; // index of smallest element

   // loop over size - 1 elements
   for ( int i = 0; i < size - 1; ++i )
   {
      smallest = i; // first index of remaining array

      // loop to find index of smallest element
      for ( int index = i + 1; index < size; ++index )
```

Figure 1. Selection Sort with pass by reference from fig. 8.13

```
// function to sort an array
void selectionSort( int * const array, const int size )
{
   int smallest; // index of smallest element

   // loop over size - 1 elements
   for ( int i = 0; i < size - 1; ++i )
   {
      smallest = i; // first index of remaining array

      // loop to find index of smallest element
      for ( int index = i + 1; index < size; ++index )

         if ( array[ index ] < array[ smallest ] )
            smallest = index;

      swap( &array[ i ], &array[ smallest ] );
   } // end if
} // end function selectionSort

// swap values at memory locations to which
// element1Ptr and element2Ptr point
void swap( int * const element1Ptr, int * const element2Ptr )
{
   int hold = *element1Ptr;
   *element1Ptr = *element2Ptr;
   *element2Ptr = hold;
} // end function swap

/**************************************************************************
 * (C) Copyright 1992-2012 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

Figure 2. Selection Sort with pass by reference from fig. 8.13

C++ Function Template for Selection Sort from Base Code

```
// Clarizza Morales
// ECE 330 - Software Design
// Homework #5 : Templates Ch.8
// Due on: 03/12/2021

#include <iostream>
#include <stdio.h>
#include <string>


template <class X>
//non-returning function
void selectionSort(X array[], int array_length)
{
     int lower_element;

   // iterate over the elements
   for (int i = 0; i < array_length - 1; i++)
   {
      lower_element = i; // get first address of array

      // iterate to find the address of the first smallest element in the given array
      for (int address = i+1; address < array_length; address++)

         if ( array[address] < array[lower_element])
                lower_element = address;
                X cont = array[i];
                array[i]=array[lower_element];
                array[lower_element]= cont;
   }
}
```

Figure 3. Function Template for Selection Sort from fig 8.13

```
int main()
{
    int int_array[20]; //int array
    int i;  //iteration variable i
    int userIn; //user input to get the # of elements or length of array
    float float_array[20]; // float array


    std::cout<< "Please enter how many elements you want to sort: \n";

    std::cin>>userIn;
    std::cout<< "\nPlease, enter array elements to sort:\n";

    int count = 1; // counter to assign the # of element
    for(i=0; i<userIn; i++){
       std::cout << "\nEnter array element # "<< std::to_string(count) << " :\n";
       std::cin >> float_array[i];
       count ++;
    }

    selectionSort(float_array,userIn); // call function to sort array
    std::cout << std::endl;

    std::cout << "Final Sorted Array:\n";

    for(i=0; i<userIn; i++){ // print the sorted array
       std::cout << float_array[i] << "\t";
    }

    return 0;
}
```

Figure 4. Function Template for Selection Sort from fig 8.13

C++ Function Template for SelectionSort() – Output

```
(base) clarizza@MacBook-Pro Module 8 % ./a.out
Please enter how many elements you want to sort:
4

Please, enter array elements to sort:

Enter element 1 :

2

Enter element 2 :
7

Enter element 3 :
3

Enter element 4 :
9

Sorted Array:
2       3       7       9


(base) clarizza@MacBook-Pro Module 8 % ./a.out
Please enter how many elements you want to sort:
3

Please, enter array elements to sort:

Enter array element # 1 :
1

Enter array element # 2 :
7

Enter array element # 3 :
4

Final Sorted Array:
1       4       7
```

Figure 5. Function Template SelectionSort() - Output

```
%(base) clarizza@MacBook-Pro Module 8 % ./a.out
Please enter how many elements you want to sort:
3

Please, enter array elements to sort:

Enter array element # 1 :
1.5

Enter array element # 2 :
7.4

Enter array element # 3 :
0.6

Final Sorted Array:
0.6     1.5     7.4
~
```

Figure 6. Function Template SelectionSort() - Output