# Lab 13
# SORTING
# Dr. John P. Abraham

One of the most important features of an array is that it can be sorted in numerical or alphabetical order. Sorting is a process by each all values in an array is reordered in ascending or descending order. Human beings can look through a list to find the smallest value, pick it out and place it in a new list and mark it off the old one. Repeat this process until only one value is left, which will be the largest one to be placed in the new list. Having two lists (particularly if the array is large) will be very costly (memory) in a computer and should be avoided. Computer based sort can be done various ways, we will cover only two in this class: **selection sort** and **bubble sort**. For selection sort keep two variables, *first* and *last*, to hold the place value of the first item and the last item in the list. Find the smallest value from a list ranging from *first to last*, and switch it with the first value in the list. Increment the value of the *first*, and repeat the process. It should become apparent to you that smallest values should be found n-1 times (n is the number of elements in a list). One common mistake students make is to have a single loop to find a single smallest number.

In the program below (13-1), pay close attention to the sort routine, you will see that we implement the sort routine with a nested loop, the first loop found within the sort function, and the second loop found in the findSmallest function (which is called from within the first loop). The variable *first* changes, but the variable *last* does not change (last changed only in the getScores function. I would ask you to refer to our prior lab exercises (lab 6) where we found smallest and largest values.

Program 13-1

```
/**************************************************
Selection Sort
Teaching objective  -Array processing. Array sorting
By Dr. John Abraham
Created for 1370 students
**************************************************/

#include <iostream>
using namespace std;

int getScores (int scores[]);
int findSmallest(int scores[],int, int);
void sort(int scores[], int);
void display(int scores[], int);
//--------------main-------------------------------------
int main ()
{
      int scores[90];
      int  last; // last for number of scores.
      last = getScores(scores); //go get the scores and how many
      cout <<"\nHere are the unsorted scores:\n";
      display(scores,last);
      sort(scores, last);
      cout <<"\nHere are the sorted scores:\n";
      display(scores,last);
      return(0);

}

//--------------function getscores----------------------

int getScores(int scores[])
{
    int last=1;
      cout << "ENTER A SCORE AND PRESS ENTER. YOU QUIT ANY TIME BY
ENTERING A NEG NUMBER!\n";
      cout << "Enter score# " << last << " ";
      cin >> scores[last];
      while (scores[last] >= 0)
      {
            last++;
            cout << "Enter score# " << last <<" ";
            cin >> scores[last];
      }
return last-1; //n-1 actual scores read (ignore negative score).
}

//-------------------function find smallest-------------------
int findSmallest(int scores[],int begin, int n)
      {
            int i, smallest=begin;
            for (i=begin; i<=n; i++)
              if (scores[i] < scores[smallest]) smallest=i;
        return smallest;
      }
//--------------------function Sort--------------------------
void sort(int scores[], int last)
{
      int sindex;
      int  tmp, first;
      for (first=1; first<last; first++) //do it number minus one
      {
      sindex = findSmallest(scores, first, last);
            tmp           = scores[first];                  //swap
            scores[first]     = scores[sindex];
            scores[sindex]= tmp;
      }
}
//-------------function display scores------------------
void display(int scores[], int n)

{
      int i;
      for (i=1; i<=n; i++)
            cout << scores[i] <<endl;
}
```

Program Run 13-1

```
ENTER A SCORE AND PRESS ENTER. YOU QUIT ANY TIME BY ENTERING A
NEG NUMBER!
77 88 99 44 23 89 47 86 99 64 88 -1
Here are the unsorted scores:
77
88
99
44
23
89
47
86
99
64
88

Here are the sorted scores:
23
44
47
64
77
86
88
88
89
99
99
```

If the selection sort algorithm was difficult to follow, I suggest you make some cards with numbers and place them inside some squares numbered 1 to n. Then follow the movements of the sort routine and shift the cards around. You will be able to learn the sort routine quicker that way.

While selection sort is an easy sort routine, the algorithm is not designed to detect an already sorted list. I already discussed that we have to find the smallest numbers n-1 times. Now think of a situation that a large list was entered in the sorted order. However, the programmer has no way of knowing it and must run the sort routine. If we used selection sort, we would not be able to discover that the list was sorted to begin with. There is another sort routine called the bubble sort that will allow us discover if a list is sorted, and exit the sort routine. The bubble sort is the most popular sorting routine. Bubble sort algorithm is based on comparing one item in the list with the next one and swapping them if needed. This way the largest number settles to the bottom while the smaller ones bubbles up. It is very important to avoid comparing the very last item in the list with a non-existent item. If no swapping is required while comparing all items in the list, the list is sorted and the sort operation may be stopped.

Just like the select short, the bubble sort also uses two loops. The external loop checks to see if the list is sorted. In order to determine if a list is sorted, the list should be gone over once comparing each value with the next to check if the value located in an index is lesser than the value located in the next index. The inner loop does just that; it compares two numbers (one value with the next one), if the first number is greater than the second number they are swapped. If you want the sort to be in descending order, just change the comparison to swap if the first value is smaller than the second. Again if you are not clear how the bubble sort works, make some note-cards, and write numbers on them; follow the sort algorithm and learn how it works.

Please note that each time it goes through the loop, the largest number goes to the bottom and we only have the compare the rest of the numbers. That the purpose of decrementing the n in the sort routine.

Program13-2

```
/*************************************************
Bubble Sort
Teaching objective  -Array processing. Array sorting
By Dr. John Abraham
Created for 1370 students
*************************************************/

#include <iostream>
using namespace std;

int getScores (int scores[]);
void sort(int scores[], int);
void display(int scores[], int);
//--------------main------------------------------------
int main ()
{
      int scores[90];
      int  last; // last for number of scores.
      last = getScores(scores); //go get the scores and how many
      cout <<"\nHere are the unsorted scores:\n";
      display(scores,last);
      sort(scores, last);
      cout <<"\nHere are the sorted scores:\n";
      display(scores,last);
      return(0);

}

//--------------function getscores----------------------

int getScores(int scores[])
{
    int last=1;
      cout << "ENTER A SCORE AND PRESS ENTER. YOU QUIT ANY TIME BY ENTERING
A NEG NUMBER!\n";
```

```
      cout << "Enter score# " << last << " ";
      cin >> scores[last];
      while (scores[last] >= 0)
      {
            last++;
            cout << "Enter score# " << last <<" ";
            cin >> scores[last];
      }
return last-1; //n-1 actual scores read (ignore negative score).
}


//--------------------function Sort--------------------------

void sort(int scores[], int n)
{
      int   i;
      bool cont;
      do
      {
            cont = false;

            for (i = 1; i <n; i++) // max i is n-1
            {
                  if (scores[i] > scores [i+1]) // compare with next
                  {
                  swap(scores[i],scores[i+1]); //using built-in function
                  cont = true;
                  }
            }
            n = n-1;

      }while (cont);

}
//-------------function display scores------------------
void display(int scores[], int n)

{
      int i;
      for (i=1; i<=n; i++)
            cout << scores[i] <<endl;
}
```

For some reason your compiler does not have the swap function in the <iostream>, use the following routine:

```
void swap (int & a, int & b)
{
      int tmp;
      tmp = a;
      a = b;
      b = tmp;
}
```

**Assignment:**
**Write a program with parallel arrays to hold student names and student IDs. Sort the parallel array based on names. Make sure to swap the student IDs along with the student names.**