SORTING\_VISUALIZATION

Visualize the sorting process of bubble sort. The integers are stored in an array. The sorting procedure sorts the integers in **an ascending order**. After each swap operation, show the result of the integers in the array.

The number of integers is 20. The integers are randomly generated inside the interval [mMinValue, mMaxValue]. Initially mMinValue = 1 and mMaxValue = 10. mMaxValue >= mMinValue.

**Key usage**

‘ ‘ : swap two elements for one time (if any) /\*press the spacebar\*/

‘r’, ‘R’ : reset the process. generate new elements

‘i’,’I’ : show system information. Invoke askForInput( )

‘<’ : decrease mMaxValue by 1.

‘>’ : increase mMaxValue by 1. The maximum value of mMaxValue is 20.

The pseudo code of bubble sort for sorting the elements of an array a:

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

for ( int j = 0; j < n – i-1; ++j ) {

if ( a[ j ] > a[ j+1 ]) **swap**( a[ j ], a[ j+1 ] ) // perform one swap operation

}

}

Thus the indices i and j must be tracked.

**How do we visualize the process of sorting?**

We must not finish the two loops at one function call. We need to process only one swap operation at a time and then show the current result. The entire process is repeated until all the elements are sorted. Thus, we must have to update indices i and j in an incremental manner. Once a swap operation is performed, we need to update i and j and then return to the main process of the entire system. We need to know when j should be reset to 0 again, and also when i should be increased by 1. Finally, we must need to determine when the entire process of sorting all the elements is done.

/\*

Generate randomly the elements between mMinValue and mMaxValue.

The total number of elements is mNumElements.

\*/

void SORTING\_VISUALIZATION::reset( )

/\*

Show the system title.

Show system information

Show key usage

\*/

void SORTING\_VISUALIZATION::askForInput( )

{

cout << "//////////////////////////////" << endl;

cout << "SORTING\_VISUALIZATION" << endl;

……

cout << "//////////////////////////////" << endl;

cout << "Press SPACEBAR to visualize the process..." << endl;

cout << "//////////////////////////////" << endl;

……

}

/\*

Two elements whose indices are (index) and (index+1).

Swap these two elements if their order is incorrect.

\*/

void SORTING\_VISUALIZATION::checkAndSwapTwoElements( int index )

/\*

Perform one swap step for two elements.

The first element is indicated by mCurrentIndex, i.e., j?

\*/

void SORTING\_VISUALIZATION::processOneSwapStep( )

/\*

Return the current element that will be processed, i.e., return j?

\*/

int getCurrentElementIndex( ) const

/\*

Return the index of the last non-handled element, i.e., return n-j?

\*/

int SORTING\_VISUALIZATION::getMaxNonHandledElementIndex( ) const

/\*

Return the number of elements.

\*/

int SORTING\_VISUALIZATION::getNumOfElements( ) const

/\*

Return the element whose index is elementIndex.

\*/

int SORTING\_VISUALIZATION::getElement( int elementIndex ) const

Add extra functions if you want.