



DEPARTMENT OF CYBER SECURITY

SUBJECT:

Data Structure

INSTRUCTOR:

Ma'am Mehmoona Jabeen

SECTION:

BSCYS-F-23-A

Project Title:

Sorting Algorithm Visualizer

Team Members:

Alyan Gulzar (231304)

Saad Bin Arif (231310)

Areej Ishfaq (231364)

Laiba Imran (231332)

Muhammad Mueez (231276)

Project Report: Sorting Algorithm Visualizer

Introduction

The "Sorting Algorithm Visualizer" is an interactive application designed to demonstrate how different sorting algorithms operate in real-time. Built using the wxWidgets library in C++, this project aims to provide an educational tool for students and enthusiasts to learn and understand sorting algorithms visually.

Explanation of Sorting Algorithms

Here's a detailed explanation of each sorting algorithm implemented in your project, which I'll integrate into the report:

1. Bubble Sort

Bubble Sort is a simple comparison-based algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted. The algorithm's name comes from the way larger elements "bubble up" to the end of the list.

Characteristics:

- **Time Complexity:** $O(n^2)$ in the worst and average cases, $O(n)$ in the best case (when the list is already sorted).
- **Space Complexity:** $O(1)$, as it works in-place.
- **Use Case:** Simple and easy to implement, but inefficient for large datasets.

Visualization in the Application:

- Bars representing adjacent elements are highlighted.
- Swapped elements change color, making the sorting process clear.

2. Insertion Sort

Insertion Sort builds the sorted portion of the list one element at a time by repeatedly picking the next element and inserting it into its correct position within the sorted part.

Characteristics:

- **Time Complexity:** $O(n^2)$ in the worst and average cases, $O(n)$ in the best case.
- **Space Complexity:** $O(1)$, as it works in-place.
- **Use Case:** Efficient for small datasets or lists that are almost sorted.

Visualization in the Application:

- The current element being placed is highlighted.
- The movement of elements as they are shifted to make space is clearly shown.

3. Selection Sort

Selection Sort repeatedly selects the smallest (or largest, depending on sorting order) element from the unsorted portion and moves it to the sorted portion. It does not perform unnecessary swaps, making it distinct from Bubble Sort.

Characteristics:

- **Time Complexity:** $O(n^2)$ in all cases.
- **Space Complexity:** $O(1)$, as it works in-place.
- **Use Case:** Useful when the number of swaps needs to be minimized.

Visualization in the Application:

- Highlights the smallest element being selected.
- The swap between the selected element and its correct position is animated.

4. Merge Sort

Merge Sort is a divide-and-conquer algorithm that splits the list into halves, recursively sorts each half, and then merges the sorted halves back together.

Characteristics:

- **Time Complexity:** $O(n \log n)$ in all cases.
- **Space Complexity:** $O(n)$, due to the auxiliary arrays used during merging.
- **Use Case:** Preferred for larger datasets due to its efficiency and stability.

Visualization in the Application:

- The division of the list into smaller parts is shown.
- The merging process, where two sorted subarrays are combined, is animated.

5. Quick Sort

Quick Sort is another divide-and-conquer algorithm that partitions the list around a pivot. Elements smaller than the pivot are moved to its left, and elements larger are moved to its right. The process is recursively applied to the subarrays.

Characteristics:

- **Time Complexity:** $O(n^2)$ in the worst case (when the pivot is poorly chosen), $O(n \log n)$ on average.
- **Space Complexity:** $O(\log n)$ due to recursive calls.
- **Use Case:** One of the most efficient sorting algorithms for large datasets.

Visualization in the Application:

- Highlights the pivot element and the partitioning process.
 - The reorganization of elements around the pivot is animated.
-

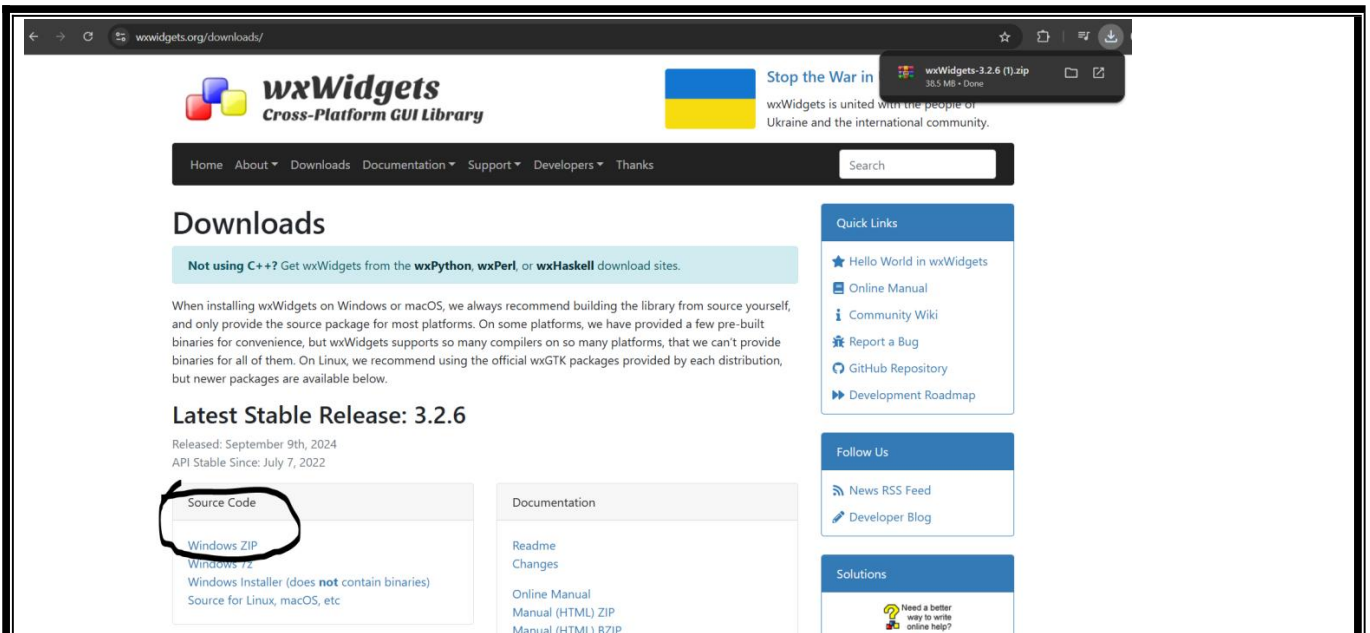
3. Tools and Frameworks

- **Language:** C++
- **Framework:** wxWidgets (GUI)
- **Development Environment:** Visual Studio 2022

4. Downloading the library

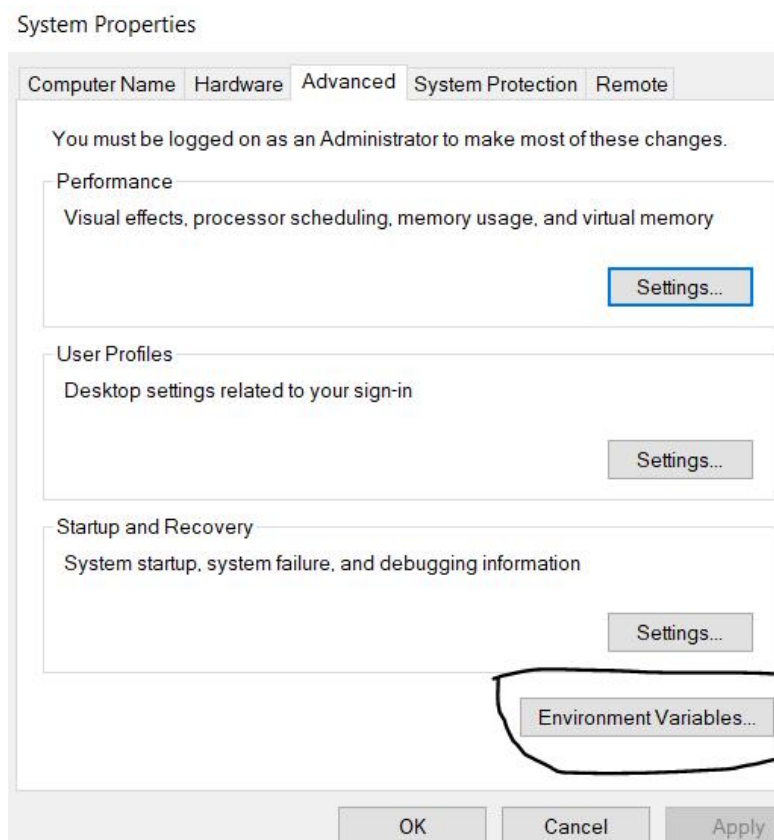
The First and main thing for this project is installing wx widget library and implementing it into visual studio code 2022

1. Visit [wxWidgets website](#) and download the Windows Zip from source code section.



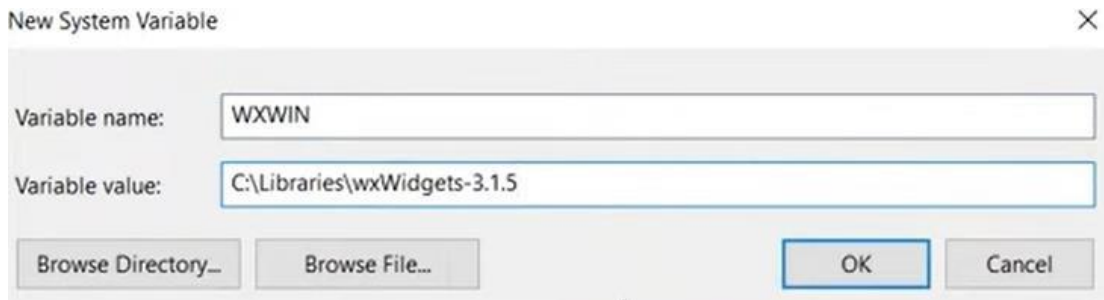
Then extract the wxWidgets.zip

3. Next we have to add the **wxWidgets** to **Environment Path Variables**
Go to **Edit the system Environment Variables**



Then click on **Environment Variables**4. Click on New

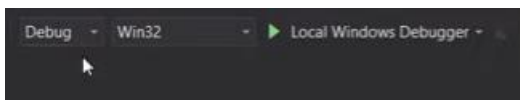
5. Add the path to the wxWidgets library and give variable name WXWIN



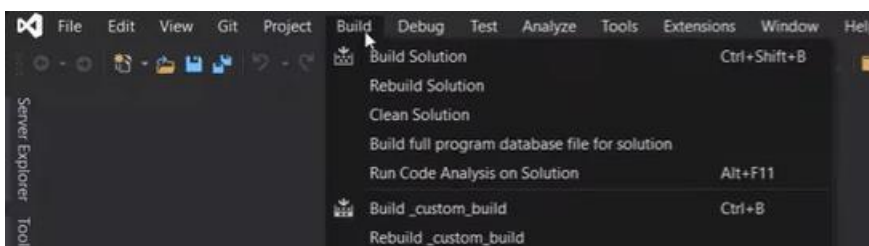
6. Then go to **wxWidgets-3.2.6 \build\msw** and click on **wx_vc17.sln** it will start visual studio 2022.

wx_vc11.sln	9/3/2024 4:43 PM	Visual Studio Solut...	38 KB
wx_vc12.sln	9/3/2024 4:43 PM	Visual Studio Solut...	38 KB
wx_vc14.sln	9/3/2024 4:43 PM	Visual Studio Solut...	38 KB
wx_vc15.sln	9/3/2024 4:43 PM	Visual Studio Solut...	38 KB
wx_vc16.sln	9/3/2024 4:43 PM	Visual Studio Solut...	39 KB
wx_vc17.sln	9/3/2024 4:43 PM	Visual Studio Solut...	39 KB

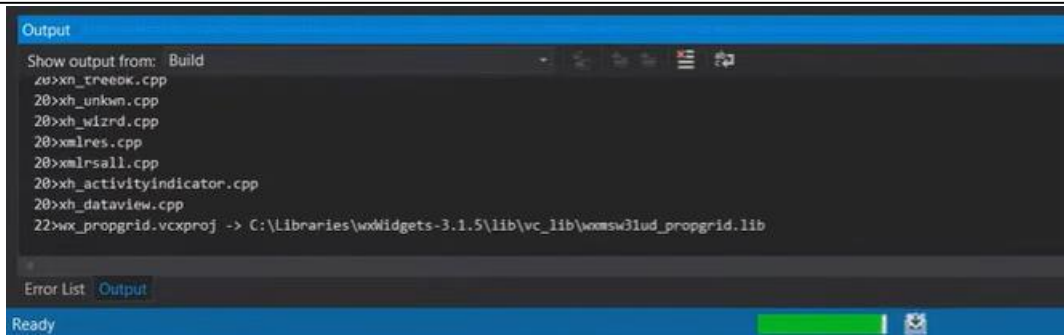
7. First choose Debug and Win32.



Then Click on Build Solution

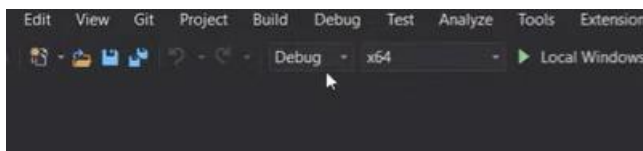


It will start the Build and it will make a **vc_lib** folder after completion

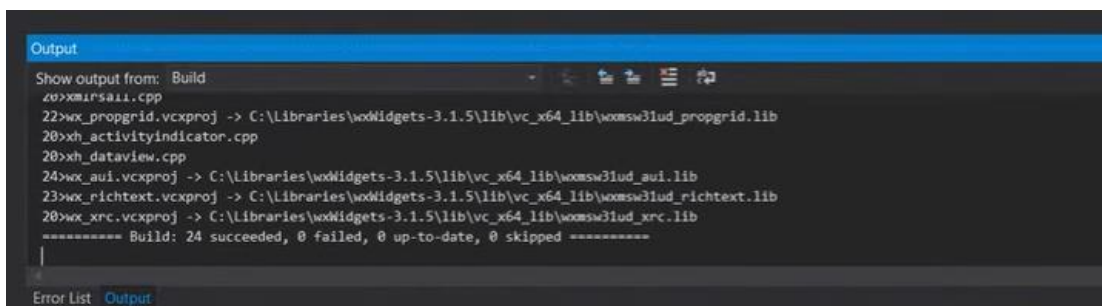


```
Output
Show output from: Build
z0>xn_treeok.cpp
20>xh_unkwn.cpp
20>xh_wizrd.cpp
20>xmlres.cpp
20>xmlrsall.cpp
20>xh_activityindicator.cpp
20>xh_dataview.cpp
22>wx_propgrid.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_lib\wxmsw31ud_propgrid.lib
Error List Output
Ready
```

Then change it from debug to release start the build solution again. Now we will also do the same for the 64 bit. Change it to 64 and debug

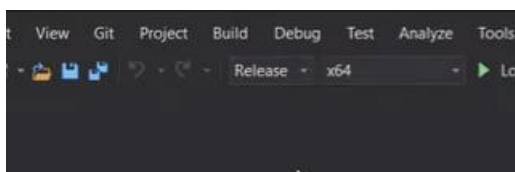


Then build Solution

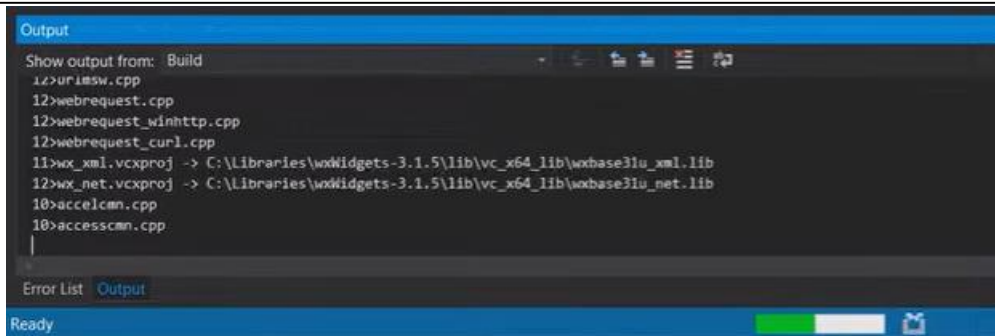


```
Output
Show output from: Build
z0>xmlrsall.cpp
22>wx_propgrid.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxmsw31ud_propgrid.lib
20>xh_activityindicator.cpp
20>xh_dataview.cpp
24>wx_aui.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxmsw31ud_aui.lib
23>wx_richtext.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxmsw31ud_richtext.lib
20>wx_xrc.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxmsw31ud_xrc.lib
***** Build: 24 succeeded, 0 failed, 0 up-to-date, 0 skipped *****
Error List Output
```

Lastly change debug to release

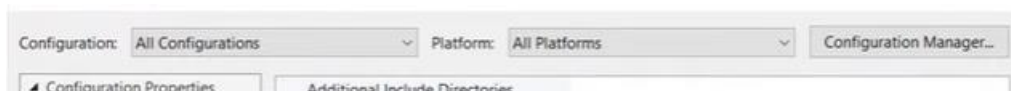


And Build the solution



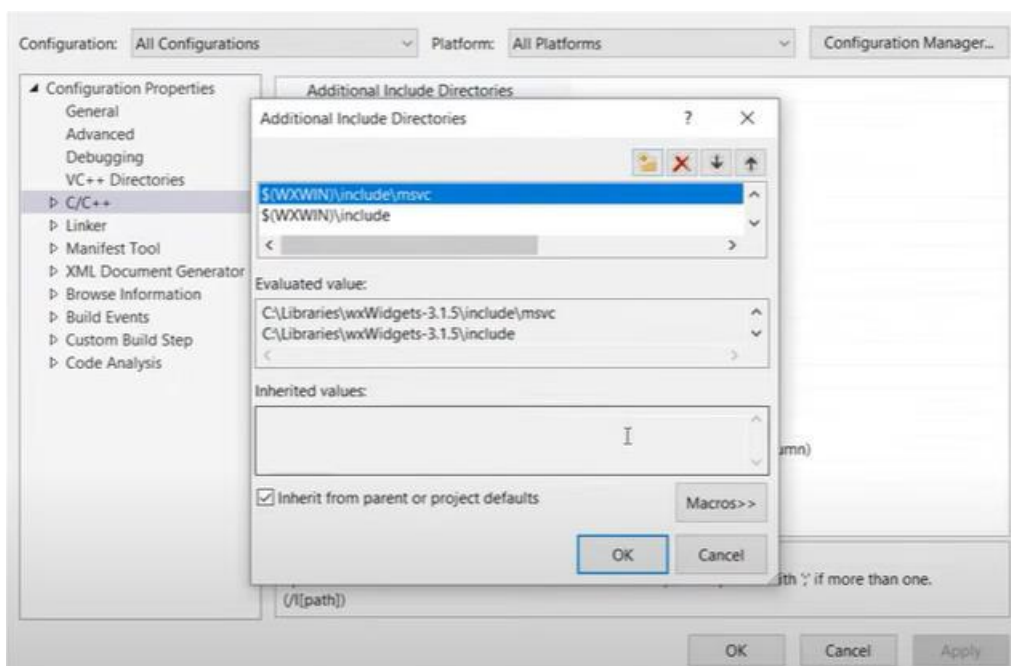
```
Output
Show output from: Build
12>urimsw.cpp
12>webrequest.cpp
12>webrequest_winhttp.cpp
12>webrequest_curl.cpp
11>wx_xml.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxbase31u_xml.lib
12>wx_net.vcxproj -> C:\Libraries\wxWidgets-3.1.5\lib\vc_x64_lib\wxbase31u_net.lib
10>accelcmn.cpp
10>accesscmn.cpp
Error List Output
Ready
```

8. Now we have library for 64 and 32 bit version installed but we need to configure it into the visual code.



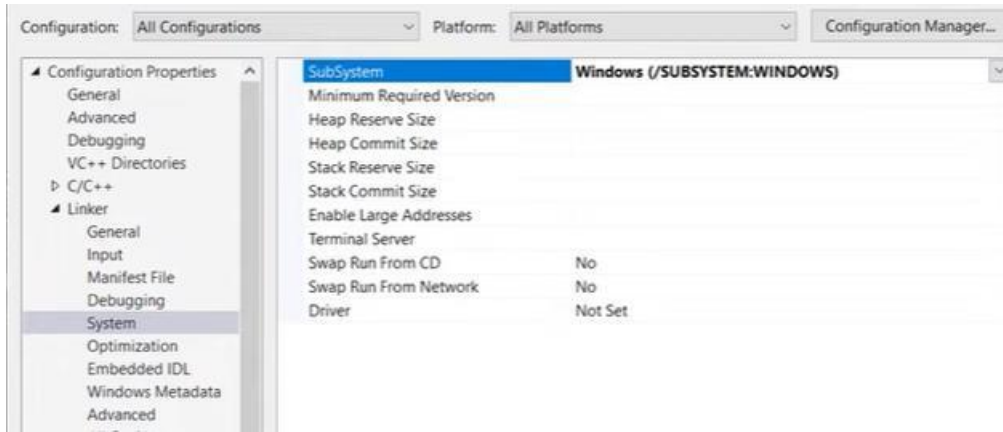
Make a project and then right click on the project click on **properties**

Then make sure the configurations and platform is selected to All Then click on C/C++ Additional Include Directories

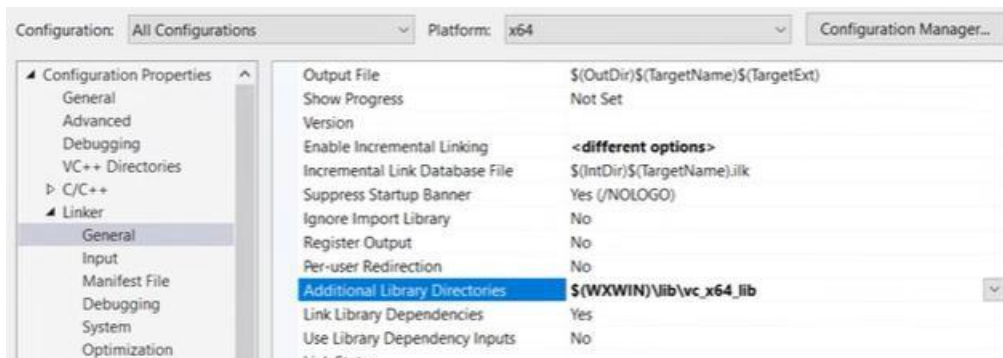


Add the path to include and then include msvc

Then click on linker then system, here we have to change from console to Windows add libraries for both win32 and win 64. Give the path to **vc_x64_lib** library in

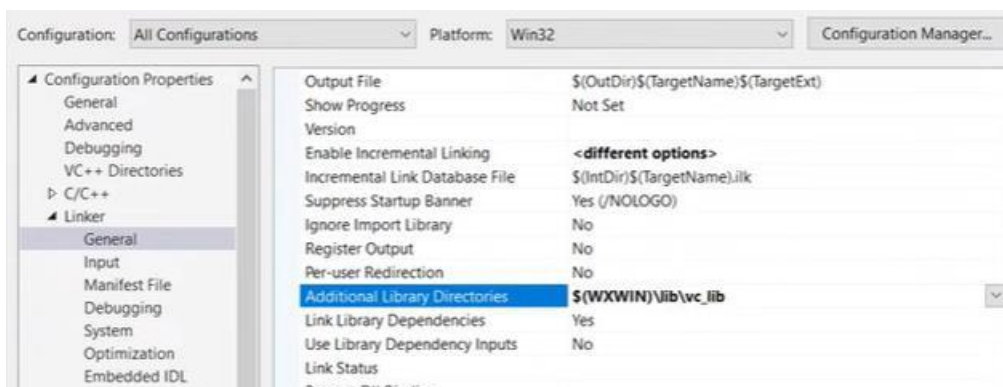


Now Click on **General** in **Linker** and change the platform to 64 as we have to



Additional Library Directories,

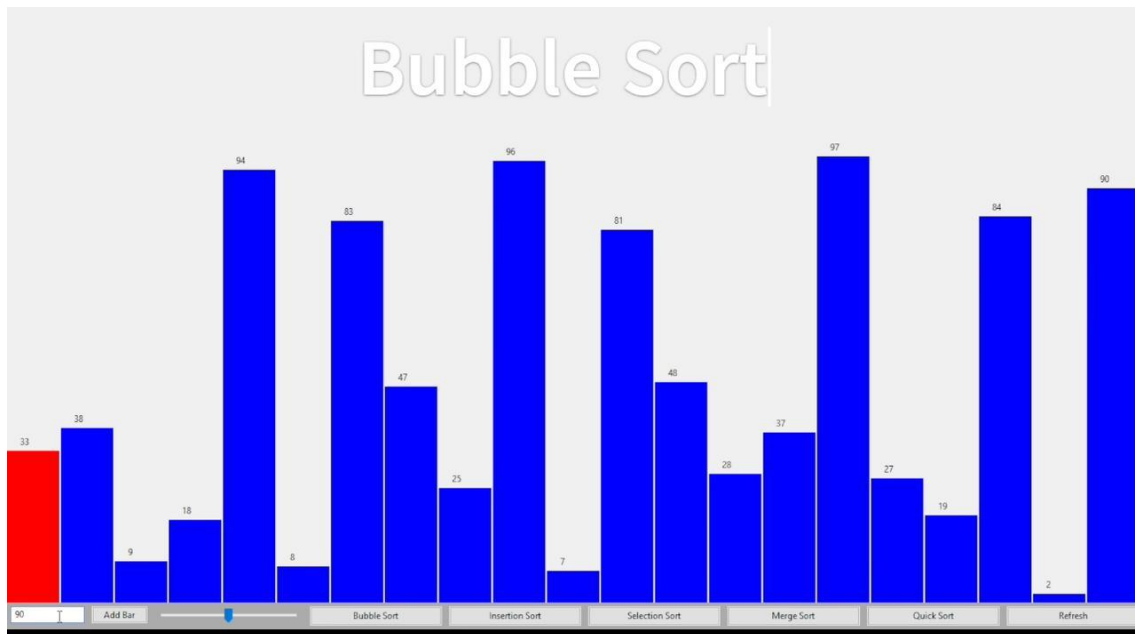
We will do this same but for **Win32 bit** platform and we will add path to **vc_lib**



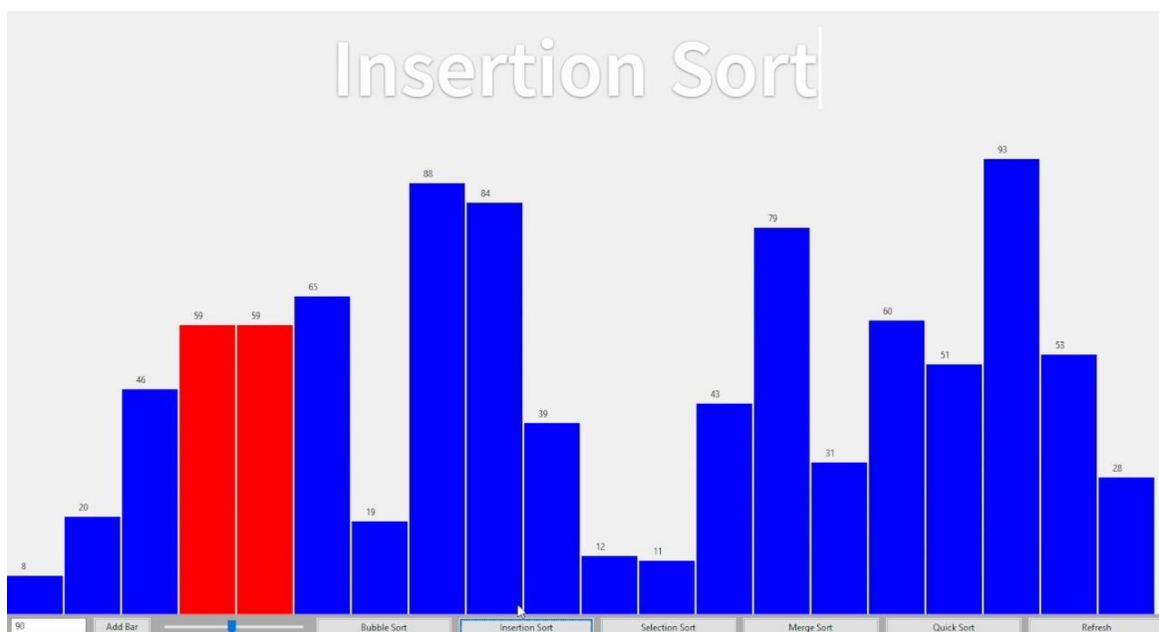
At last the wxWidgets library is successfully Installed!

Demonstration:

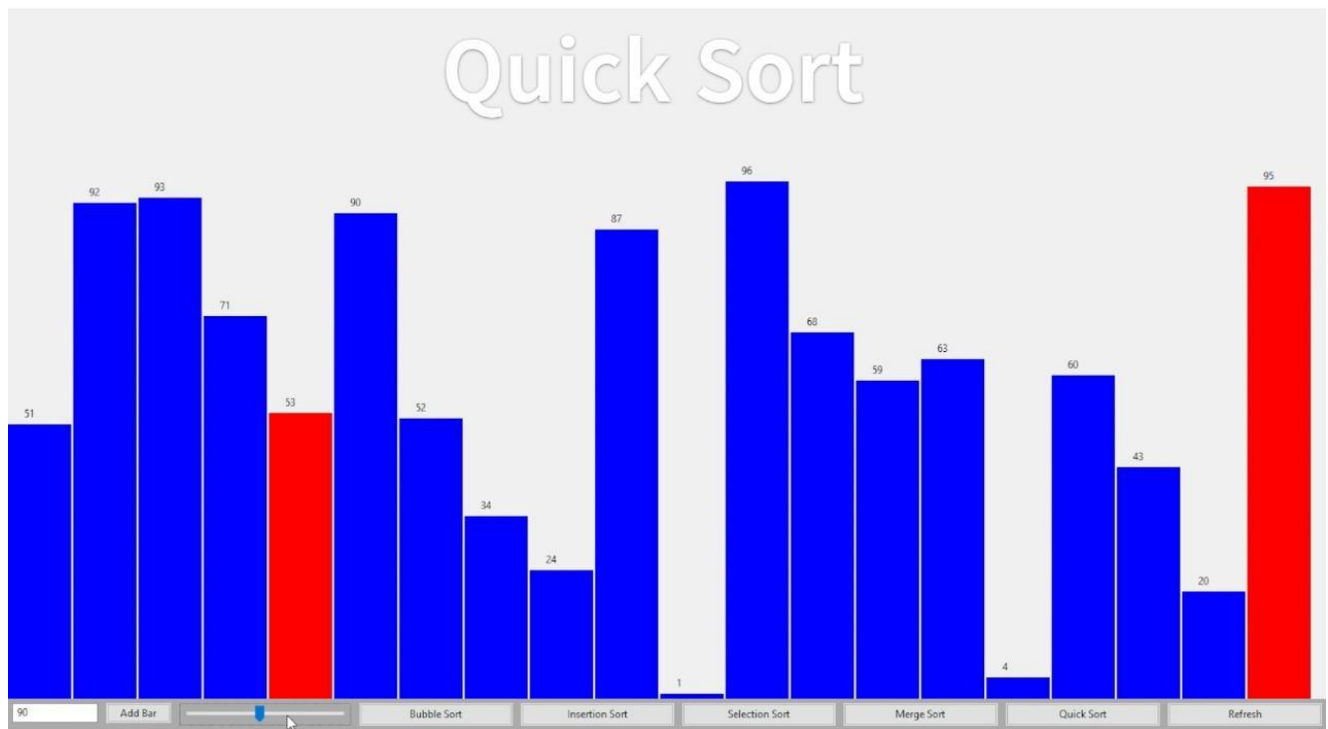
➤ Bubble Sort



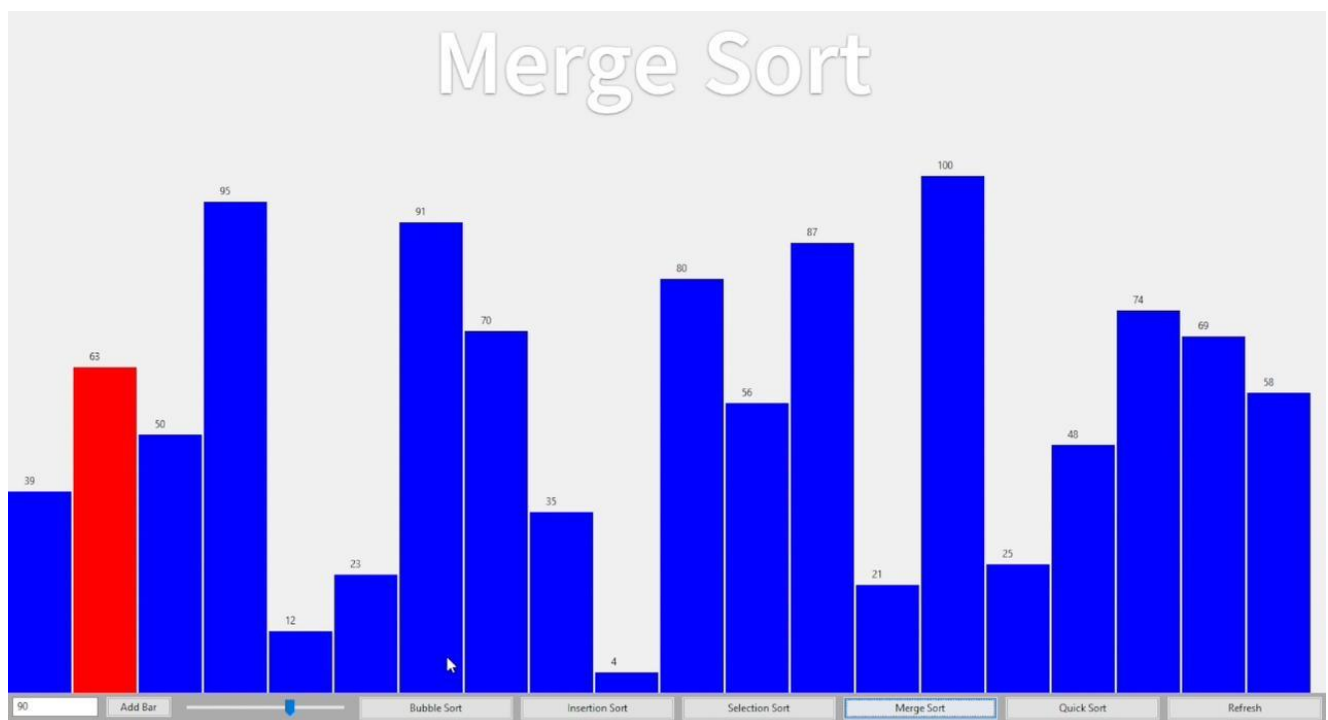
➤ Insertion Sort



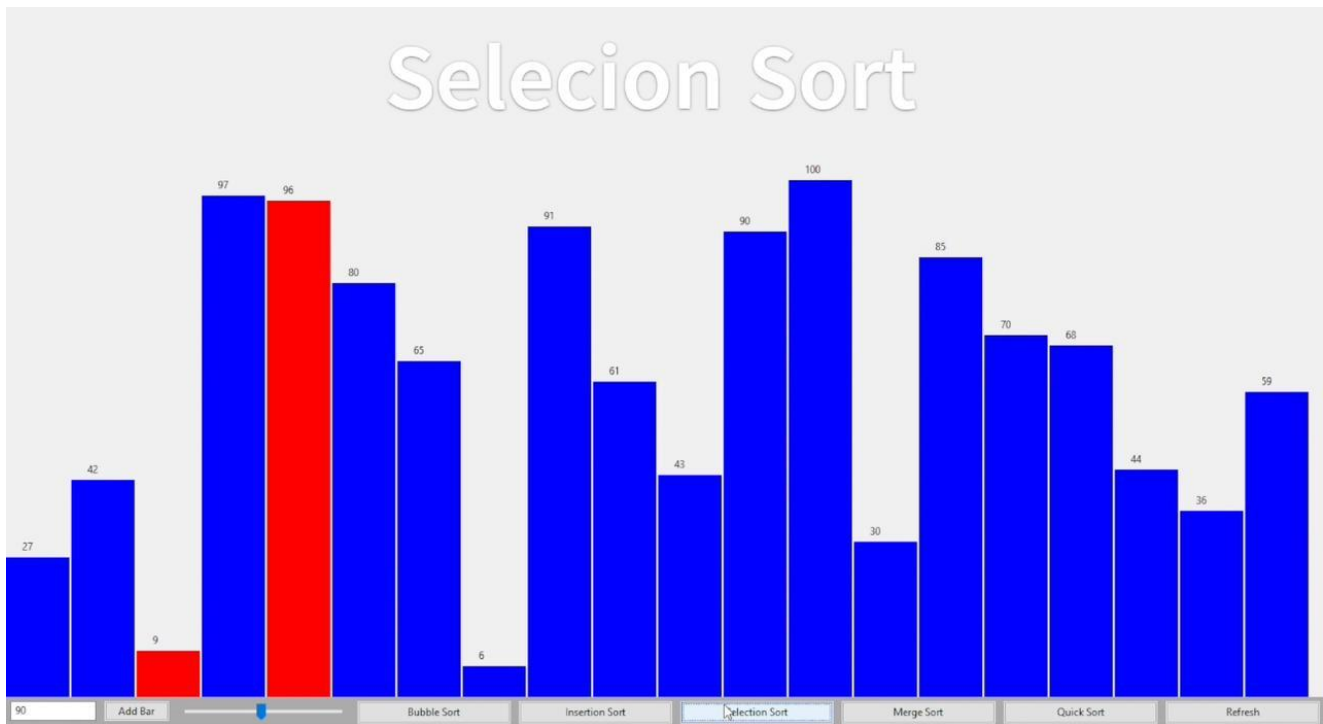
➤ Quick Sort



➤ Merge Sort



➤ Selection Sort



- The array values are displayed as **vertical bars** of varying heights.
- Each bar's height is proportional to the value of the corresponding element in the array.
- During sorting:
 - **Red bars** indicate that sorting is ongoing.
 - **Blue bars** indicate idle or unmodified bars.

Project Objectives

The primary objectives of the Sorting Algorithm Visualizer are:

1. **Educational Value:** To help users understand the mechanics of various sorting algorithms through a visual medium.
2. **User Interaction:** To provide an interactive interface where users can add data, control the speed of sorting, and choose the algorithm to visualize.
3. **Customization:** To allow users to refresh the dataset, adjust visualization speed, and add new elements dynamically.
4. **Performance Awareness:** To illustrate the differences in performance and behavior between sorting algorithms.

Features

Interactive GUI: The application provides a graphical user interface with:

1. Buttons for selecting sorting algorithms.
2. A slider for adjusting sorting speed.
3. An input field to add custom data.
4. A refresh button to generate a new dataset.

Dynamic Visualization:

1. Bars represent data values, with heights proportional to their magnitude.
2. Real-time updates as sorting progresses, with bars changing colors to indicate activity.

Multiple Sorting Algorithms:

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Quick Sort

Customizable Speed:

A slider allows users to control the delay between visualization updates, offering flexibility in observing the algorithms.

Data Management:

1. Users can dynamically add elements to the dataset.
2. The dataset is randomized and refreshed with unique values for every session.

Code Overview

1. Core Components

(a) SortingPanel Class

The SortingPanel class handles the visualization of the sorting process and user interaction. Key responsibilities include:

- Drawing the dataset as bars on the panel.
- Handling events such as painting, refreshing, and updating the panel.
- Providing methods to update visualization dynamically.

Key Methods:

- OnPaint(wxPaintEvent&): Handles the drawing of bars.
- UpdateVisualization(): Refreshes the GUI to display the current state of the dataset.
- RefreshArray(): Resets the dataset with randomized unique values.
- AddBar(int value): Allows users to add new elements dynamically.
- SetSpeed(int newSpeed): Updates the delay for visualization.

(b) Sorting Algorithms

The application implements five sorting algorithms. Each algorithm is provided with real-time visualization using the UpdateVisualization() method:

Bubble Sort:

- Compares adjacent elements and swaps them if necessary.
- Visualization highlights comparisons and swaps.

Insertion Sort:

- Builds a sorted section of the array one element at a time.
- Visualization shows elements being moved into position.

Selection Sort:

- Repeatedly selects the smallest element and places it at the correct position.
- Highlights the selection process.

Merge Sort:

- Recursively divides the dataset into smaller sections and merges them in sorted order.
- Visualization displays merging of subarrays.

Quick Sort:

- Partitions the dataset around a pivot and recursively sorts each partition.
- Visualization highlights pivot selection and partitioning.

(c) SortingVisualizerApp Class

The main application class initializes the GUI and binds user interactions to their respective actions.

Key Elements:

- Buttons for each sorting algorithm.
- A slider for adjusting speed.
- Input field and button for adding new data.
- A refresh button to randomize the dataset.

2. Data Visualization

The bars representing the dataset are dynamically scaled based on the panel size. The height of each bar corresponds to its value, and the width adjusts to fit the panel. During sorting, the bars change colors to indicate active comparisons and swaps.

Technical Details

Development Environment:

- Language: C++
- Framework: wxWidgets
- Platform: Linux (tested on Kali Linux)

Dependencies:

- wxWidgets library for GUI development.
- Standard C++ libraries for multithreading and randomization.

Multithreading:

- Sorting algorithms run in separate threads to ensure a responsive GUI.

Customization:

- Adjustable dataset size and sorting speed.
- Dynamic addition of elements.

User Guide

1. Launching the Application

To run the application:

1. Ensure wxWidgets is installed.
2. Compile the code using a compatible C++ compiler.
3. Execute the compiled binary.

2. Using the Visualizer

- Select a sorting algorithm using the corresponding button.
- Adjust the sorting speed using the slider.
- Add new elements by entering a value and clicking "Add Bar."
- Refresh the dataset using the "Refresh" button.

Conclusion

The Sorting Algorithm Visualizer successfully demonstrates the inner workings of various sorting algorithms in an intuitive and interactive manner. Its modular design and user-friendly interface make it an excellent tool for learning and teaching. By visualizing the step-by-step process of sorting, users gain a deeper understanding of algorithmic behavior and performance.
