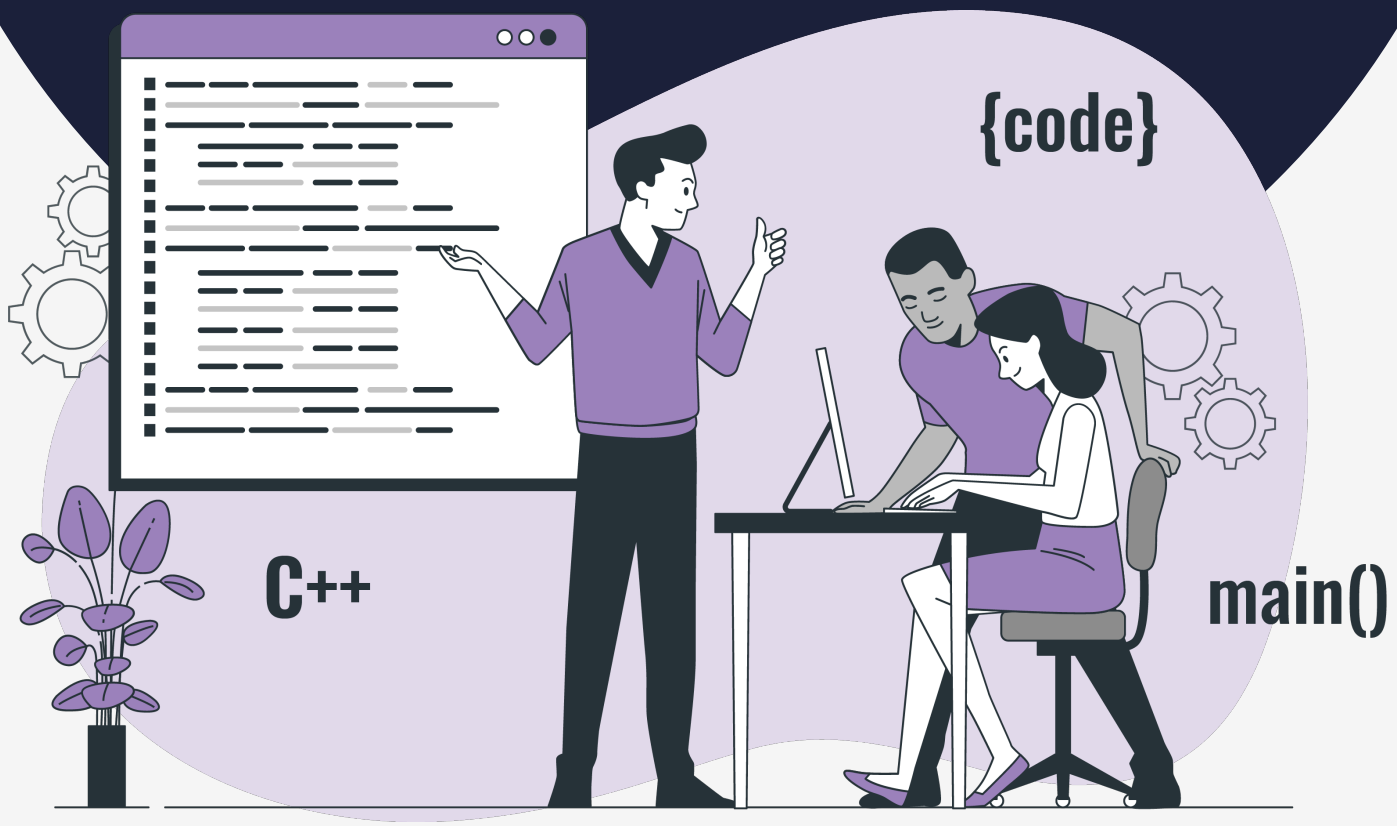
Lesson:

Problems based on sorting



Pre-Requisites

- Basics of c++
- Arrays
- Bubble sort
- Selection sort
- Insertion sort.

List of Concepts Involved

- Problem based on bubble sort
- Problem based on selection sort

Problem 1: Given an integer array arr, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

Input:
Enter the size of vector
5
Enter the elements of vector
0 5 0 3 42

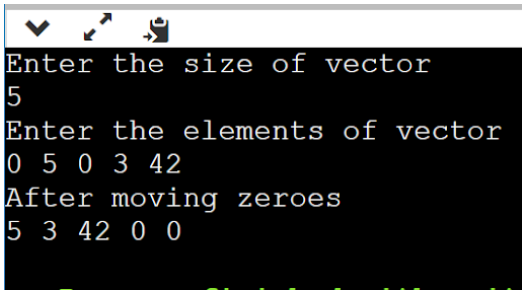
Output:
After moving zeroes
5 3 42 0 0

Approach:
The main idea of bubble sort is swapping the smallest (or largest) element to the end of the array, in this problem, **we should swap '0' to the end.**

- In the main function we have taken the input of the array and made a call to the “moveZeroes” function where we have passed our array as the parameter.
- In the function “moveZeroes” we have traversed the array from back to front where any position of i denotes the last value till where we have any non zero number, and beyond which we will have only zeros left.
- Another pointer j is initialized to 0 which will traverse the array from front to back(till i) because we know after i there will only be zeroes.
- And just in bubble sort we make sure to fit the maximum number at the very end of the array after every iteration (ascending order), in this case also we made sure that after every single iteration the last position must be grabbed by a zero.
- Now if an array does not contain any zero, for that case we have explicitly taken a flag variable which is initialized to 0 in the beginning of every iteration and once we found any value at the jth index to be 0 we will raise the flag indicating that we still need to move this 0 to the end, hence we cannot terminate the process.
- And we keep on swapping that particular arr[j] to the end whose value is found to be 0. Hence we swap every arr[j] with arr[j+1].
- After every iteration the value of i keeps on decreasing showing the last positions grabbed by 0.

- Once in any iteration if there are no 0 between j and i then automatically the value of flag won't increase, it will stay 0 and this will be a sign showing that there are no more 0 in between and we can break from the loop.
- This way we have moved all the zeros to the end while maintaining the relative order between the non-zero elements.

Code: https://pastebin.com/pxSn1Afv



Problem 2: Give an array of names of the fruits; you are supposed to sort it in lexicographical order using the selection sort

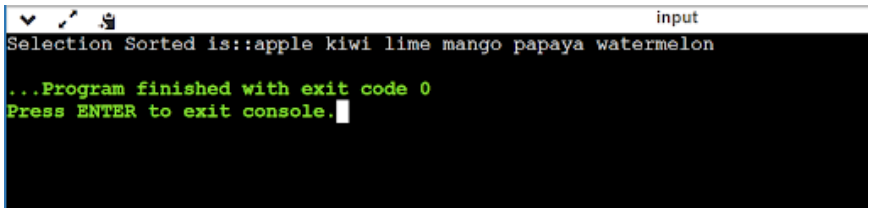
Input : ["papaya","lime","watermelon","apple","mango","kiwi"]
Output: ["apple","kiwi","lime","mango","papaya","watermelon"]

Approach:

- In the main function we have created a 2D character array, where we have taken some strings as input.
- We have initiated the size of the array, and made a call to the function "selSort" in which we have passed the fruit array and its size.
- In the "selSort" function we have created a temporary character array in which we will temporarily store the lexicographically smallest string.
- Here we have used two inbuilt functions:
  - strcpy() : this function is used to create a copy of a given string.
  - strcmp(): this function is used to compare two strings.
    - If both the strings are equal this function will return 0.
    - If the first string is lexicographically greater than the first string then this function will return a value greater than 0.
    - If the second string is lexicographically greater than the first string then this function will return a value lesser than 0.
- Here we have applied a nested loop, firstly we have assumed our ith string to be the most minimum (lexicographical order wise) and we will explore other strings from i+1 to n to find if there is any other string smaller than this current one.
- For that purpose we have used strcmp() function and we have compared the strings at ith and jth indices where i is the parameter of first for loop and j is the parameter of the second/nested for loop.
- If the value returned by the strcmp() function is greater than 0 that means the ith string that we have assumed to be smallest is not the smallest and we have stored the jth string as the new smaller string and j will be its index whose value is stored in the min variable.

- If we have found any string at index other than i which is smaller than the i^{th} string then we need to swap the ith and j^{th} strings in order to make the first index string to be the smallest.
- For string swapping we have basically used the strcpy()function whose task is to copy the strings.
- Traverse the remaining values (except the value at 0th index), to find the next smallest, then swap this value with the value at index 1.
- Scan the remaining values (Except the first two values) to find the next smallest, then swap this value with the value at index 2.
- Continue until the array is sorted.

Code: https://pastebin.com/V3TeQ1BS



Upcoming Class Teasers:

- Merge sort