## Project

A playlist is a list of video or audio files that can be played back on a media player either sequentially or in a shuffled order. In its most general form, an audio playlist is simply a list of songs, but sometimes a loop.

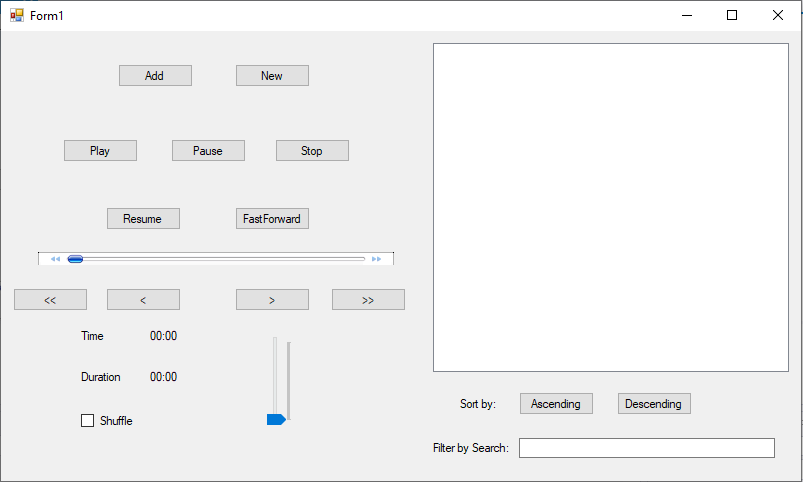
Some Internet streaming services, such as Spotify, Amazon Music, Project Playlist, MagicPlaylist, 8tracks, Plurn and Webjay, allow users to categorize, edit, and listen to playlists online. Other sites focus on playlist creation aided by personalized song recommendations, ratings, and reviews. On certain sites, users create and share annotated playlists, giving visitors the option to read contextual information or reviewer comments about each song while listening

My project is to create a music player that has the means to function like a player should. It could be done using any Programming language, on addition must have three things (Search Algorithm, Sort Algorithm and an Abstract Data Type).

The following are the choices I made:

* For Language: **C#** Language.
* For ADT: **Queue** with Array Implementation.
* For Search Algorithm: **Linear** Search.
* For Sort Algorithm: **Bubble** Sort.

## Initial Box



**19**

**20**

**12**

**10**

**18**

**17**

**15**

**14**

**13**

**11**

**16**

**9**

**8**

**7**

**6**

**5**

**4**

**3**

**2**

**1**

Figure (Initial, 2020)

The Initial Box is the interface when you run the code which contains couple of function which are labeled through number. The following label are below:

1. Next Song
2. Last Song
3. Time of Music
4. Duration of Music
5. Shuffle
6. Volume Bar
7. List Box
8. Sort by Ascending
9. Sort by Descending
10. Search Music
11. Add
12. New
13. Play
14. Pause
15. Stop
16. Resume
17. Fast Forward
18. Progress Bar
19. First Song
20. Previous Song

## Functions

### Add Button (1)

The add button gives the option to browse the music shown in the figure 60.

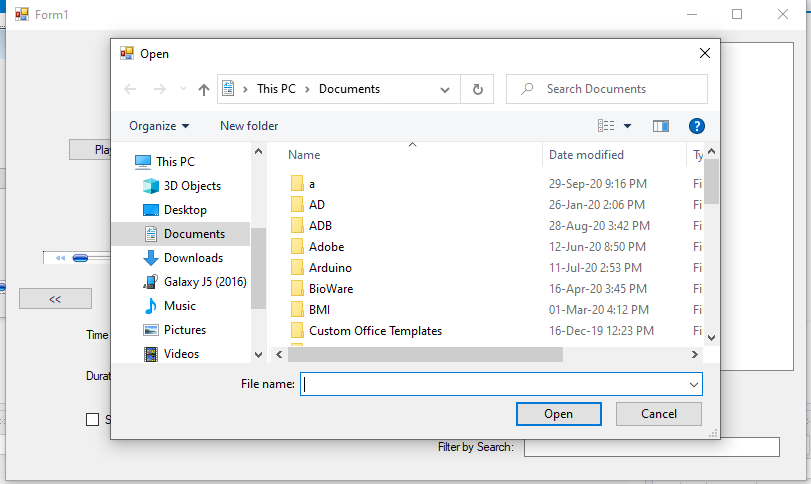


Figure (Browse, 2020)

Once the songs are selected the empty list box in Figure 59 as “**17**” will be updated to the songs selected as figure 61. In this case we have the first selection of number that were selected, after where the letters were selected and as you can see both of them are showed.

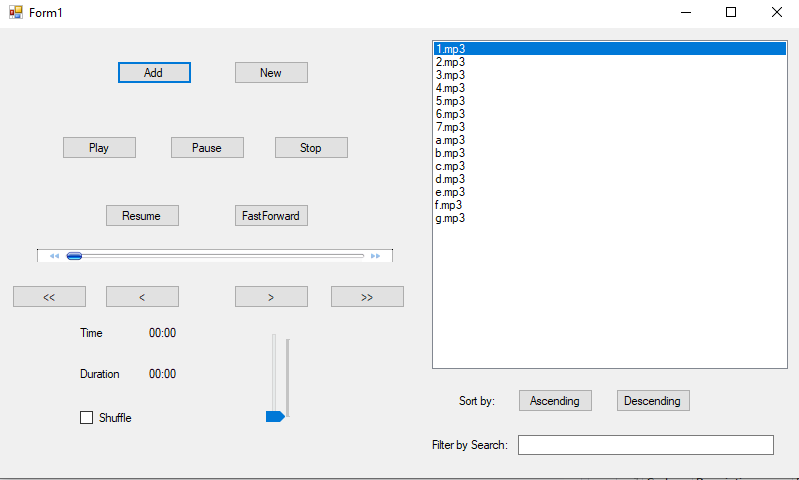


Figure (Add, 2020

### New Button (2)

The “New” works same as the “Add” button. It will also have the step to browse the music as figure 60. However, the only minor difference is that the “Queue” will get emptied and only the selected list will be shown. Unlike the add where the previous list will also be shown.

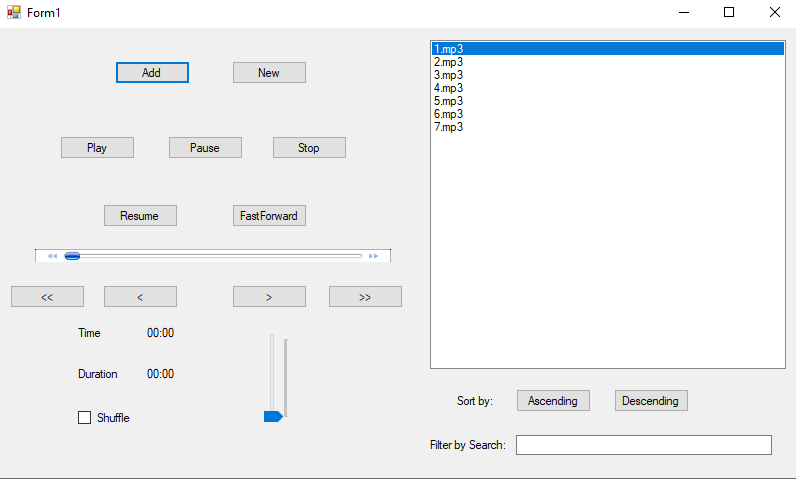


Figure (New, 2020)

### Ascending Button (18)

The “Ascending” Button will sort the list of music from Low to High. As shown in the figure 63 the List are all in order of ascending.

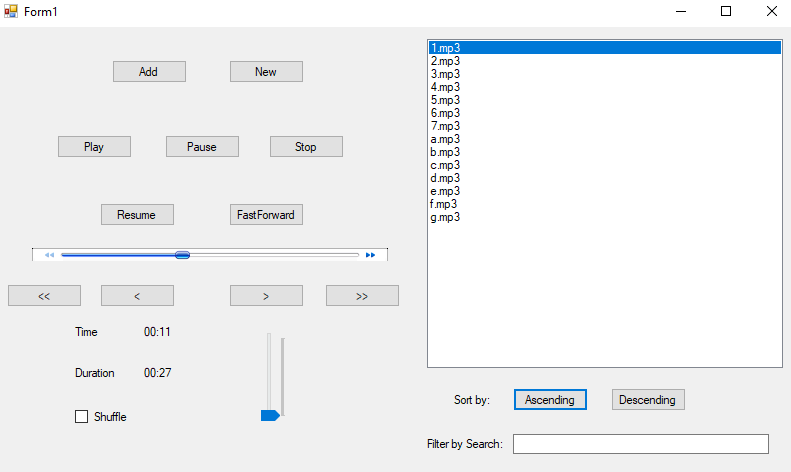


Figure (Ascending, 2020)

### Descending Button (19)

The “Descending” Button will sort the list of music from High to Low. As shown in the figure 64 the List are all in order of descending.

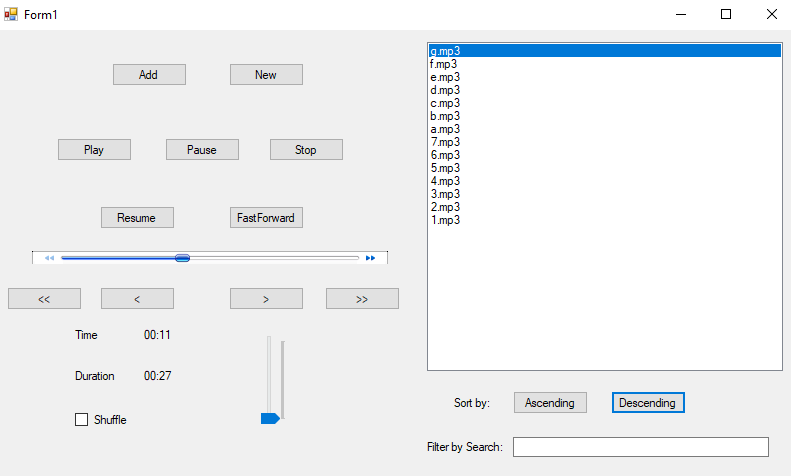


Figure (Descending, 2020)

### Search Box (20)

The “Search” Box as the function of searching the specific types label. So for example as shown in figure 65 the song “d” is entered as the specific searched song is shown in the list box.

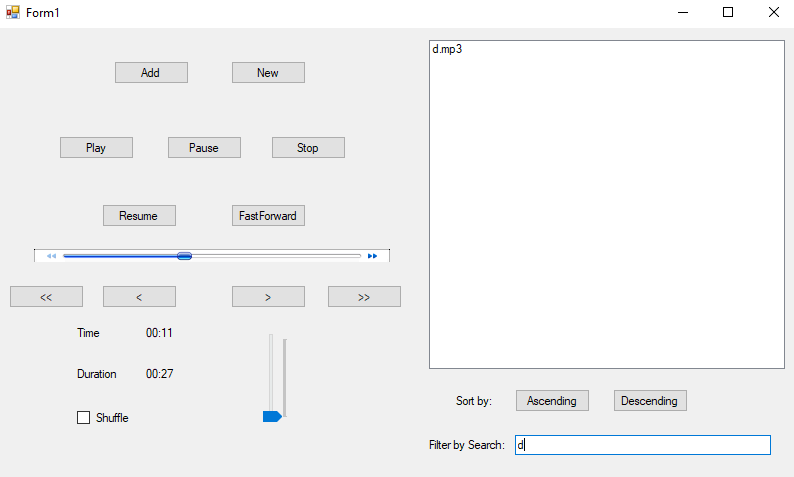


Figure (Search, 2020)

### Others

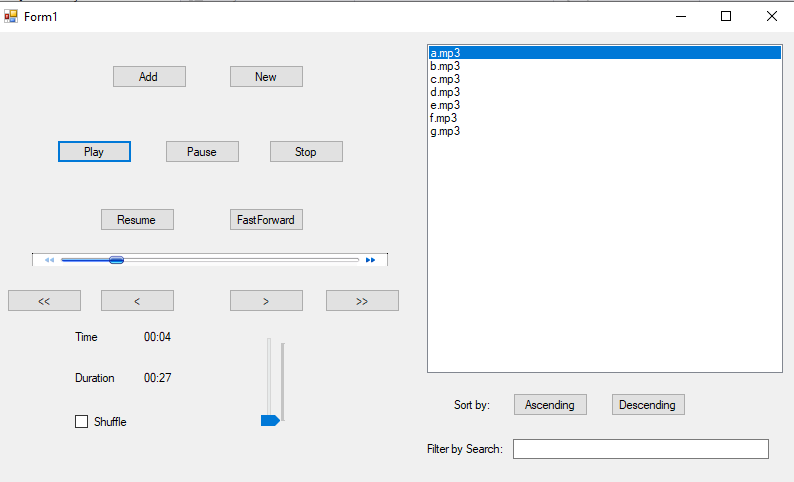


Figure (Other, 2020)

The rest of the functions are self-explanatory. And on the figure above you can see that when the song is being selected the time and duration is shown as well.

## Code

The Project is done in Visual Studio in the name as “MusicPlayer” and all the function are coded to their own locations. For example, rather than making a class for a function I double clicked the button and coded in it so it’s much cleaner.

### Declare

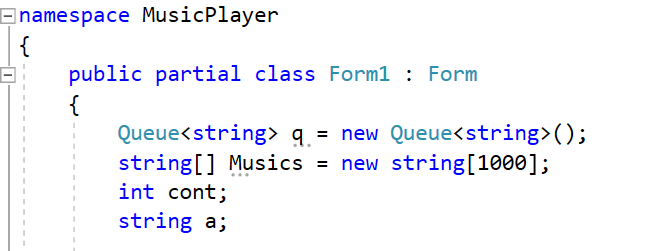


Figure (Declare, 2020)

The above figure shows the declaration of the variable. I have created a “String Queue” which is needed to the requirement. And an array as “Musics” to use that as the array to show it in the list box and to play is since the windows media player cant play a Queue ADT but and Array. The “int cont” is for the index value for the loops. And the “string a” is for the search variable when the user enters in the text box.

### Play

The Figure below shows that I have used the built-in function of the windows media player. Where one of the ways to give is to use the key word “Player” and connect the array which is the “Musics” by the term “URL”. And since I have to show the time and duration, I have started the timer so that the minutes and the seconds can be retrieved.

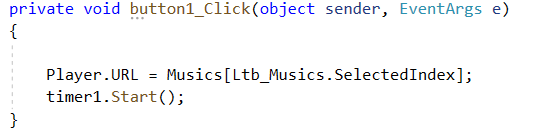


Figure (Play, 2020)

### Pause

The Figure below shows that I have used the built-in function of the windows media player. Where I used the “Ctlcontrol” library that contains the “pause()” feature for pausing the player when the “pause” button is pressed.

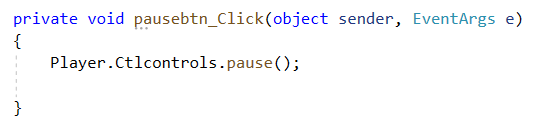


Figure (Pause, 2020)

### Resume

The Figure below shows that I have used the built-in function of the windows media player. Where I used the “Ctlcontrol” library that contains the “play()” feature for resuming the player when the “resume” button is pressed.

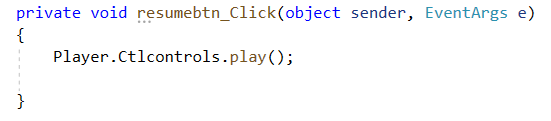


Figure (Resume, 2020)

### Stop

The Figure below shows that I have used the built-in function of the windows media player. Where I used the “Ctlcontrol” library that contains the “stop()” feature for stopping the player when the “stop” button is pressed.

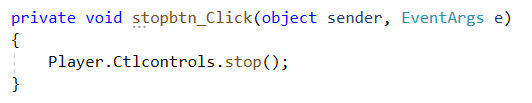


Figure (Stop, 2020)

### Fast Forward

The Figure below shows that I have used the built-in function of the windows media player. Where I used the “Ctlcontrol” library that contains the “fastForward()” feature for fast forwarding the player when the “fastforward” button is pressed.

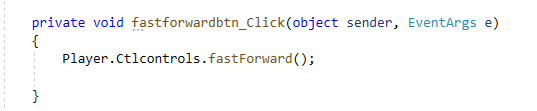


Figure (Fast Forward, 2020)

### Volume Bar

The Figure below shows that I have used the built-in function of the windows media player. Where I used the “settings” library that contains the “.volume” feature for enabling the volume and using the “volumeScroll” so that its movable from mute to max in the interface.

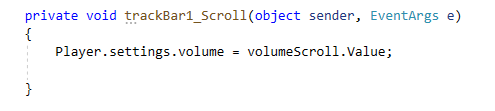


Figure (Volume Bar, 2020)

### Add

The figure below shows the function for the “add” button. The term “OpenFileDialog()” is used to be able to browse the songs and once those are selected it goes to the “if” loop. Where the value of each song is taken one by one in the variable “r” in the loop and converted to string to be able to “enqueue” it to the “Queue”. While doing that the next step uses “cont” to give the index value for all the elements so that I can use to sort the array out.

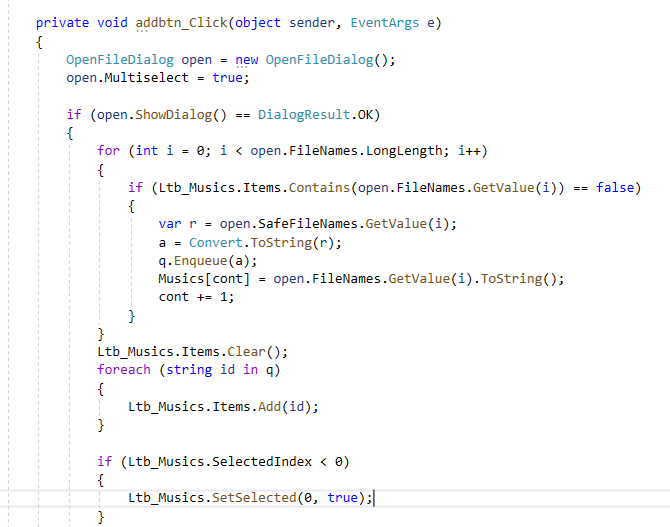


Figure (Add Part 1, 2020)

After that it clears the current list as shown in the figure below. And displays the whole “Queue” element in the list box by using “foreach” loop. The final part is made so that after element are added it will automatically highlight the first song so that we don’t need to select and just can click the “play” button.

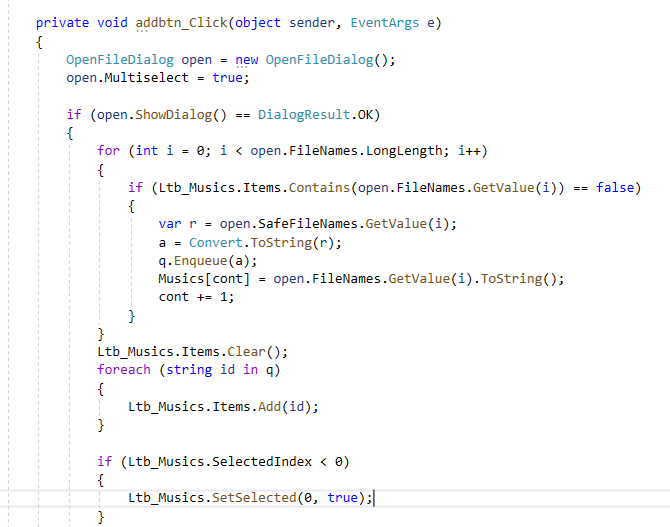


Figure (Add Part 2, 2020)

### New

The “New” button is the exact cope of the “add” button with addition code of “q.Clear()” as shown in the figure below. Which is to empty out the “Queue” so that its can start newly.

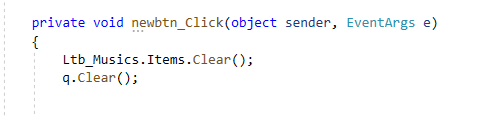


Figure (New, 2020)

### Previous

The figure below shows the function for when the “Previous” button is clicked. The first condition is that if the index value is not zero then to subtract an index value of one and play that music. It made that way since there is nothing before zero. Hence the condition and if it less than zero then it goes the last song of the list.

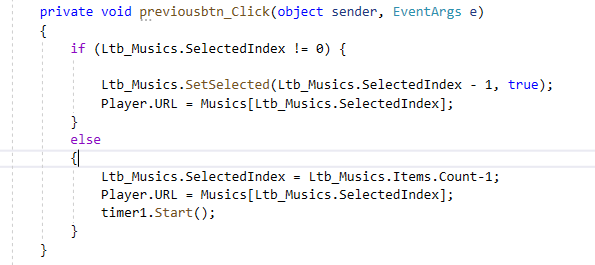


Figure (Previous, 2020)

### Next

The figure below shows the function for when the “Next” button is clicked. The first condition is that if the index value is not max then to add an index value of one and play that music. It made that way since there is nothing after max. Hence the condition and if it greater than max then it goes the first song of the list.

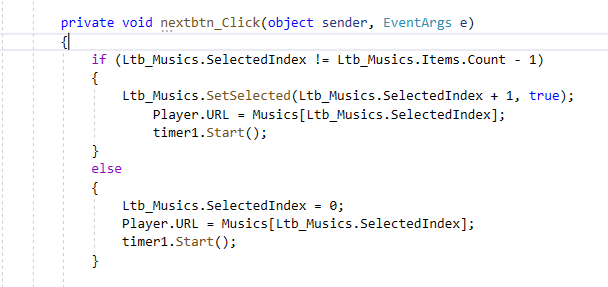


Figure (Next, 2020)

### First

The figure below shows the function for the “First song” button. Where I manually insert the selected music and the first song which is “0”. So that when the button is pressed it goes to the first song to be played.

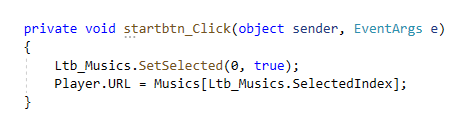


Figure (Start, 2020)

### Last

The figure below shows the function for the “Last song” button. Where I manually insert the selected music and the last song which is “.Count -1”. So that when the button is pressed it goes to the last song to be played.

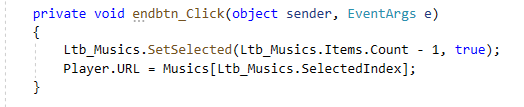


Figure (Last, 2020)

### Shuffle

The figure shows the place where I have implemented the “shuffle” function by using the “Random” object to change the next index value in “random.Next”. And after the selected random index is played by using “.SelectedIndex”.

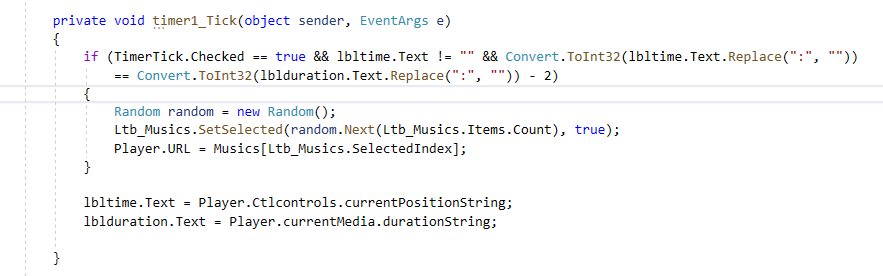


Figure (Shuffle, 2020)

### Time Duration

The time and the duration has to be made by time a timer. Since the time has to run to be retrieved after all. While the time is running I used the inbult function of “currentPosition” which takes the current time and assigned it to the label in the interface. And the same done for the duration by using “.currentMedia.duration”.

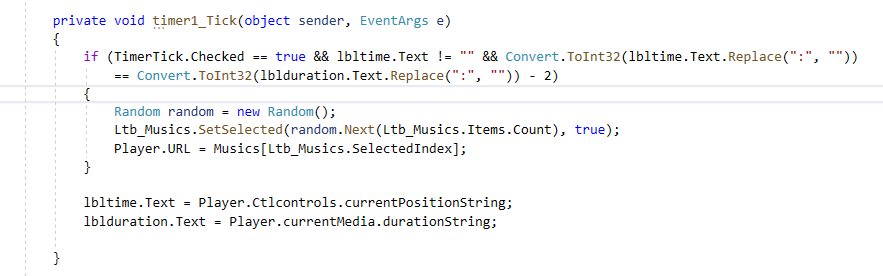


Figure (Time Duration, 2020)

### Auto Play Next Music

The figure below shows the function of playing the next song once the current song is finished played. Once the song is played the state is “stop” for the player, that’s why I used the “e.newState” term that has numbers for so many states and one of them for stop is the number 8. So I am saying that if the player stops by itself meaning that the current song is finished then the condition becomes true. And if it is true then change the index value to the next one or if it is in the last song change the index into the first one so its becomes in a loop and doesn’t give an error of null. After the selection is done then play is as usual with timer which what the lines are the end. However since these must be don’t automatically the keyword “BeginInvoke” is used for the purpose.

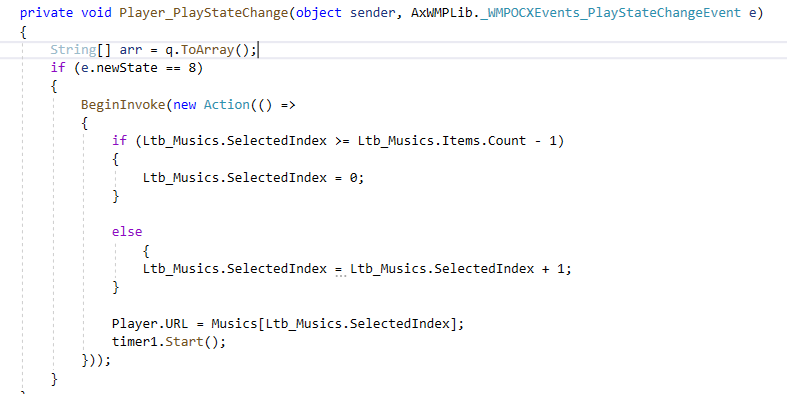


Figure (AutoPlay Next Music, 2020)

### Ascending Sort

The ascending sort uses the bubble sort algorithm. The bubble sort uses the method of comparing pairs and swapping them one to the larger side and the other to the smaller side. Therefore, by doing that I have to use a variable to swap which “temp” and a condition which is “swap”. The loop must go on until there isn’t anything to swap that why the condition is true in the “while” loop. After inside consist of another loop for going on and on till the arrays length which is in the “for” loop. Finally, the numbers in the pair are compared inside the “if” condition. On addition it also have the same feature as shown in figure 75.

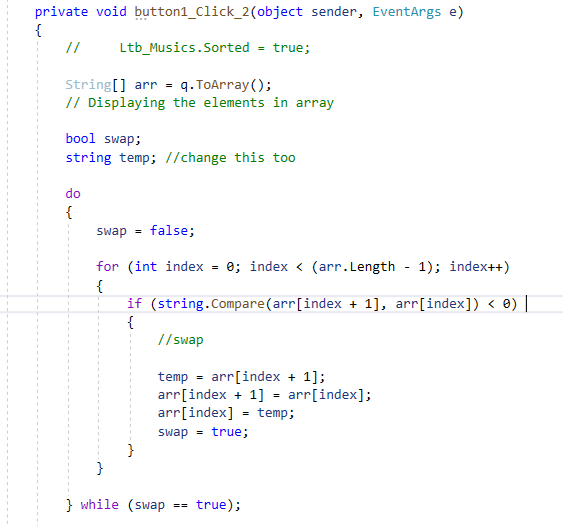


Figure (Ascending, 2020)

### Descending Sort

The descending sort works exactly the same way as the ascending but the only difference is the “swap” value. Where there is “arr[index]” it is replaced with “arr[index]+1” and vise versa.

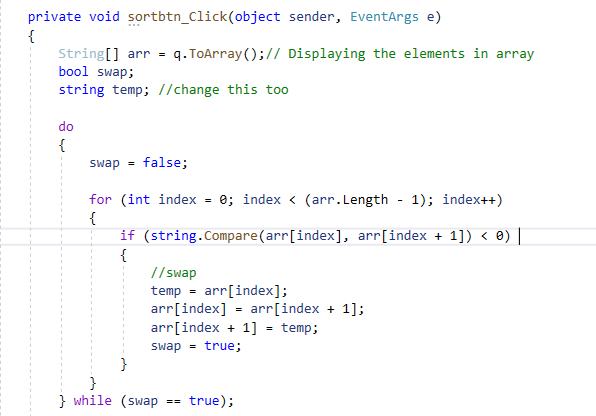


Figure (Descending, 2020)

### Search

The figure below shows the search function where the text is taken from the textbox as “searchtxt.Text” and assigned it to a variable “search”. After a query is made in form of a condition where the text that is matched in the array is to be retrieved and be assigned to the “items”. Finally once it is done the list is being cleared so that only the elements that we searched for is to be displayed.

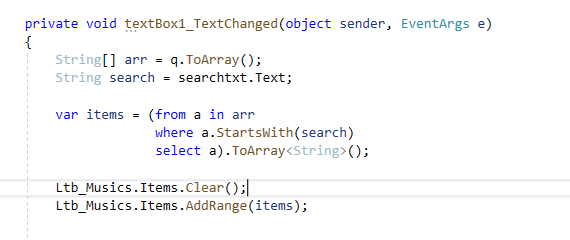


Figure (Search, 2020)