

AT3 Project



April 1, 2019

Bryce huston

30003673

Table of Contents

[Introduction 0](#_Toc5025164)

[Analysis 0](#_Toc5025165)

[Test Data 1](#_Toc5025166)

[Program Code: 1](#_Toc5025167)

[User Guide 7](#_Toc5025168)

# Introduction

The program requires the user to input into 3 text boxes with “Game”, “Platform” and “Genre”. When pressing the Add Button the users input will display in the Output box to the left. When more items are added the Output box sorts the input using a bubble sort algorithm and displays the inputs in alphabetical order. If you would like to clear the text boxes press the Clear Button. When selecting a line in the Output box and press the Delete Button it will delete the selected index from the Output box. By pressing the Reset Button it will clear all items in the Output box. If you would like to change a part of one of the items in the Output Box, selected the line you would like to change, type the appropriate text you would like to see and press the Update Button for it to update the section you have changed. To search for an item in the Output box, enter the designed text in the Name Text box and press the Search button. When finished press the Save Button to save all entry to a .dat file located in the programs bin/debug folder. When you reopen the application press the Open Button to load all previous entry’s that where saved from earlier.

# Analysis

|  |  |
| --- | --- |
| Input: | * Click events on buttons * User entered string data to textbox fields (tbName, tbPlatform, tbGenre) * List Box index selection for various features * Binary File Data |
| Process: | * Load saved binary file data into “numItems” and array variables on click and program startup * Save current “numItems” and array data to binary file * Add string data to array with numItems controller * Delete string data from array with numItems controller * Clear data from text box control * Reset locally saved array data with numItems controller * Update selected list box entry with new data from text box controls * Search from array data that matches data added to “Name” textbox controller |
| Outputs: | * Display sorted added numItems in list box * Display error message saying all fields are not entered * Remove deleted entries from list box * Show empty textbox controls on “Clear” click * Show empty list box on “Reset” click * Show updated information in list box on “Update” click * Display error message saying all fields not entered * Select list box index matched to string “Name” text box controller input * Display error message saying search not found * Create/overwrite binary file * Display data loaded from binary file within list box * Display message saying saved file does not exist |

# Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Description | Expected outcome | Evidence |
| Case 1. | Add new game | Data added to Array, then displayed in List Box |  |
| Case 2 | Edit data | Click on data in List Box and edit data and press Update Button. Data will be updated. |  |
| Case 3. | Delete data | Selected Data will be Deleted from the List Box |  |
| Case 4. | Search data | Enter desired data in the “Names” text box and press the Search Button. Data will be displayed in text box |  |
| Case 5. | Save data | Data will be saved as a .dat file within the program. When relaunching the program press the Open button to display saved data |  |

# Program Code:

//Adds input from text boxs to the array and then displays it on the list box

private void btnAdd\_Click(object sender, EventArgs e)

{

//if text box is empty display error message

if (String.IsNullOrEmpty(tbName.Text))

{

MessageBox.Show("Please enter game name", "Error", MessageBoxButtons.OK);

}

//if it exceeds 20 arrays

else if (numItems > 20)

{

MessageBox.Show("You have exceeded the amount of entrys available", "Error", MessageBoxButtons.OKCancel);

}

//else display input

else

{

names[numItems] = tbName.Text;

platforms[numItems] = tbPlatform.Text;

genres[numItems] = tbGenre.Text;

numItems++;

Bubblesort(names, platforms, genres);

DisplayGames();

Clear();

}

}

//Method for displaying the 3 arrays in the list box (lstOutput)

private void DisplayGames()

{

lstOutput.Items.Clear();

for (int i = 0; i < numItems; i++)

{

lstOutput.Items.Add(names[i] + " " + platforms[i] + " " + genres[i]);

}

}

//Method for clearing all text boxs

private void Clear()

{

tbName.Clear();

tbPlatform.Clear();

tbGenre.Clear();

}

//Method for deleting items in the list box

private void DeleteAt(string[] names, int index)

{

for (int i = index; i < numItems - 1; i++)

{

names[i] = names[i + 1];

platforms[i] = platforms[i + 1];

genres[i] = genres[i + 1];

}

numItems--;

}

//Delete button that calls upon the DeleteAt method

private void btnDelete\_Click(object sender, EventArgs e)

{

int index = lstOutput.SelectedIndex;

if (index == -1)

{

MessageBox.Show("Select an item", "Error", MessageBoxButtons.OK);

return;

}

else

{

DeleteAt(names, index);

DisplayGames();

Clear();

}

}

//Clears all text boxs

private void btnClear\_Click(object sender, EventArgs e)

{

Clear();

}

//Save button which saves data onto a .dat file which we can open later

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

using (BinaryWriter writer =

new BinaryWriter(File.Open("mygames.dat", FileMode.Create)))

{

writer.Write(numItems);

for (int i = 0; i < numItems; i++)

{

writer.Write(names[i]);

writer.Write(platforms[i]);

writer.Write(genres[i]);

}

}

}

catch (IOException x)

{

MessageBox.Show("Exception: " + x.Message);

}

}

//Open button which opens data that was saved as a .dat file

private void btnOpen\_Click(object sender, EventArgs e)

{

if (File.Exists("mygames.dat"))

{

try

{

using (BinaryReader reader = new

BinaryReader(File.Open("mygames.dat", FileMode.Open)))

{

numItems = reader.ReadInt32();

for (int i = 0; i < numItems; i++)

{

names[i] = reader.ReadString();

platforms[i] = reader.ReadString();

genres[i] = reader.ReadString();

DisplayGames();

}

}

}

catch (IOException x)

{

MessageBox.Show("Exception: " + x.Message);

}

}

else

{

MessageBox.Show("Saved file does not exist", "Error", MessageBoxButtons.OK);

return;

}

}

//Sorting algorithm which sorts the input in the list box into alphabetical order

private void Bubblesort(string[] names, string[]platforms, string[]genres)

{

//Bubble sort Algorithm

for(int outer = numItems - 1; outer > 0; outer--)

{

for(int inner = 0; inner < outer; inner++)

{ /// . mod there (use a string instead)

if (names[inner].CompareTo(names[inner + 1]) > 0)

{

string tmpnames = names[inner + 1];

names[inner + 1] = names[inner];

names[inner] = tmpnames;

string tmpplatforms = platforms[inner + 1];

platforms[inner + 1] = platforms[inner];

platforms[inner] = tmpplatforms;

string tmpgenres = genres[inner + 1];

genres[inner + 1] = genres[inner];

genres[inner] = tmpgenres;

}

}

}

}

//Search method that is called with the Search button is clicked

private int BinarySearch(string[] names, string search)

{

//Binary Search Algorithm

int lowerBound = 0;

int upperBound = numItems - 1;

int i;

while (true)

{

i = (lowerBound + upperBound) / 2;

if (names[i] == search)

{

return i; //found it

}

else if (lowerBound > upperBound)

{

return -1; //Can't find it

}

else

{

if (names[i].CompareTo(search) < 0)

{

lowerBound = i + 1;

}

else

{

upperBound = i - 1;

}

}

}

}

//Search’s items in list box also calling on the Binary Search method

private void btnSearch\_Click\_1(object sender, EventArgs e)

{

string search = tbName.Text;

int index = BinarySearch(names, search);

if (index != -1)

{

lstOutput.SelectedIndex = index;

tbName.Text = names[index];

tbPlatform.Text = platforms[index];

tbGenre.Text = genres[index];

}

else

{

MessageBox.Show("Item not found");

}

}

//Clears everything in the list box, but does not delete from the array

private void btnReset\_Click(object sender, EventArgs e)

{

lstOutput.Items.Clear();

}

//Changes selected input in list box to desired entry

private void btnUpdate\_Click(object sender, EventArgs e)

{

int index = lstOutput.SelectedIndex;

names[index] = tbName.Text;

platforms[index] = tbPlatform.Text;

genres[index] = tbGenre.Text;

DisplayGames();

}

private void lstOutput\_SelectedIndexChanged(object sender, EventArgs e)

{

int index = lstOutput.SelectedIndex;

if(index == -1)

{

MessageBox.Show("Please select item");

return;

}

// Select item from Output box

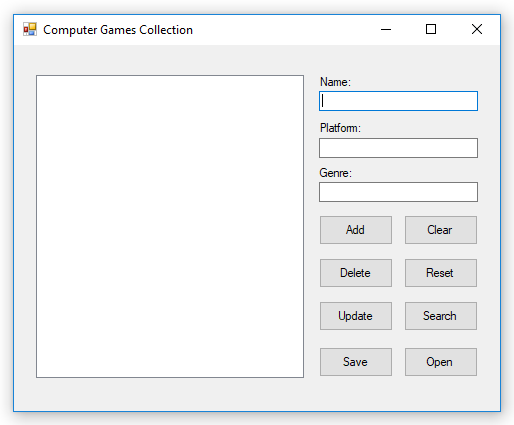
tbName.Text = names[index];

tbPlatform.Text = platforms[index];

tbGenre.Text = genres[index];

# User Guide

Add:



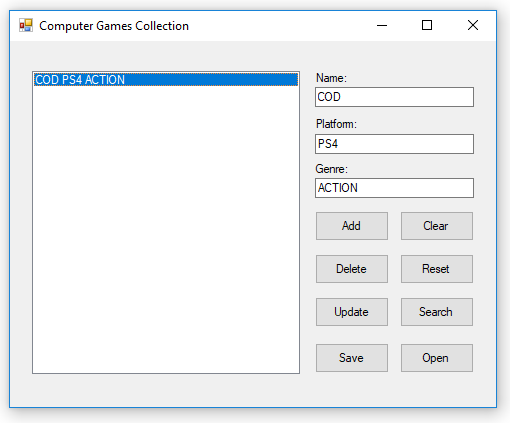
Press the Add Button

Enter a game platform

Enter a game genre

Enter a game name

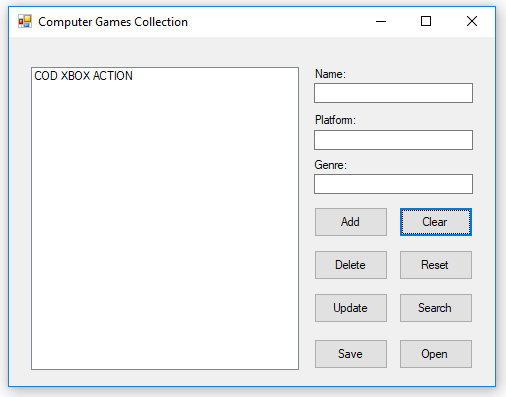
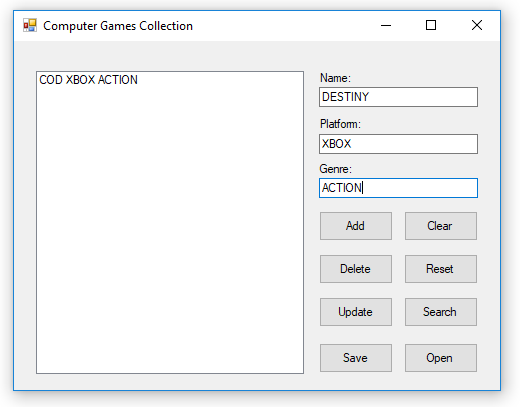
Delete:



Then Press the Delete Button

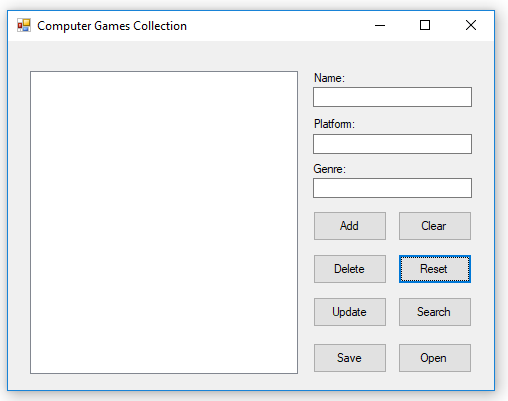
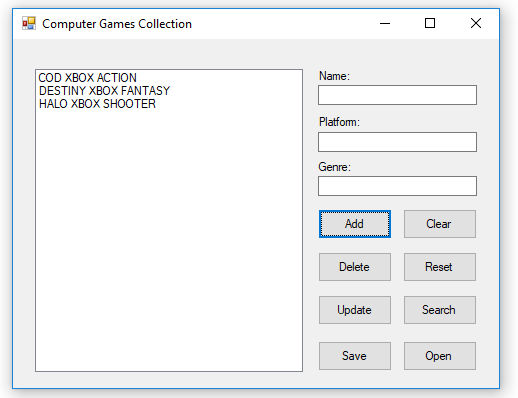
Selected Item from list box

Clear:



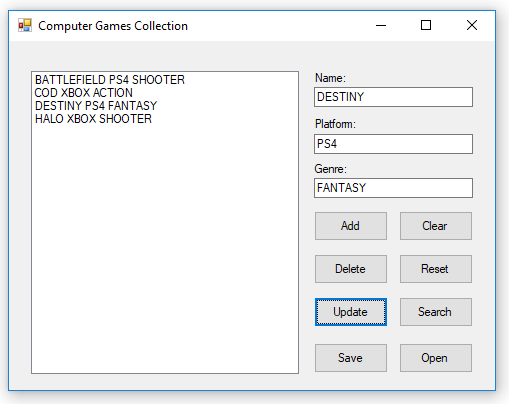
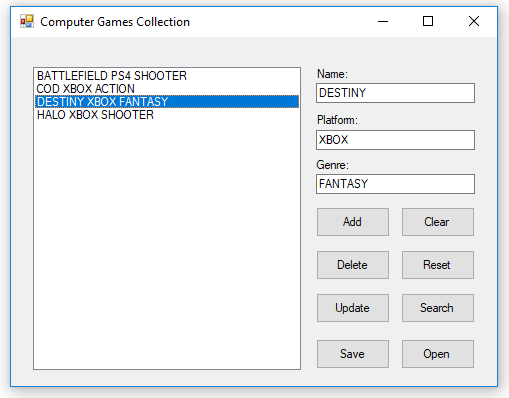
Press the Clear Button

Reset:



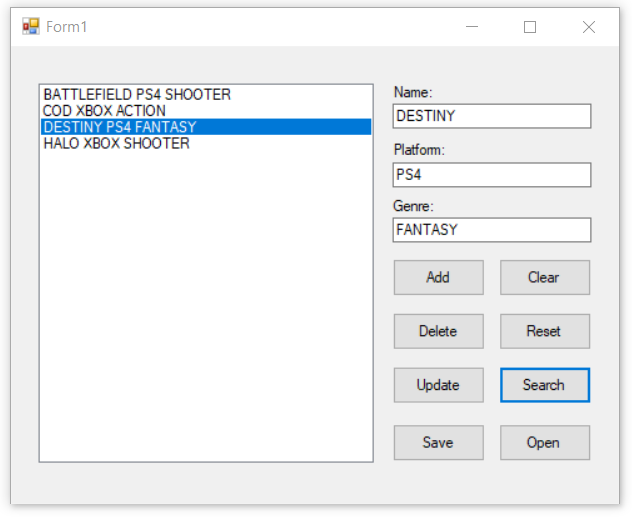
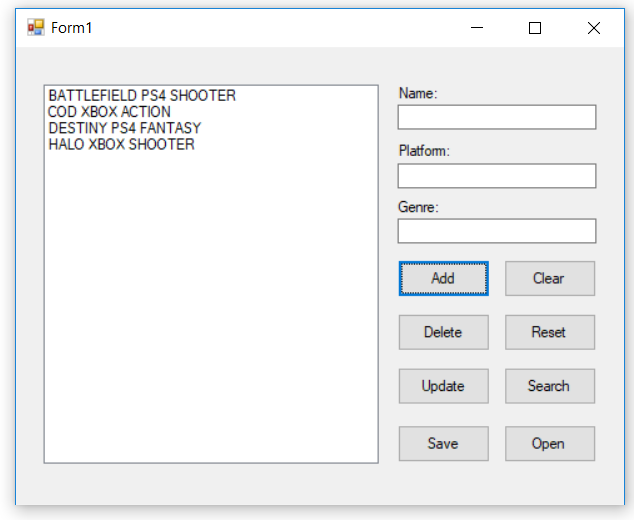
Press the Reset Button

Update :



Press the Update Button

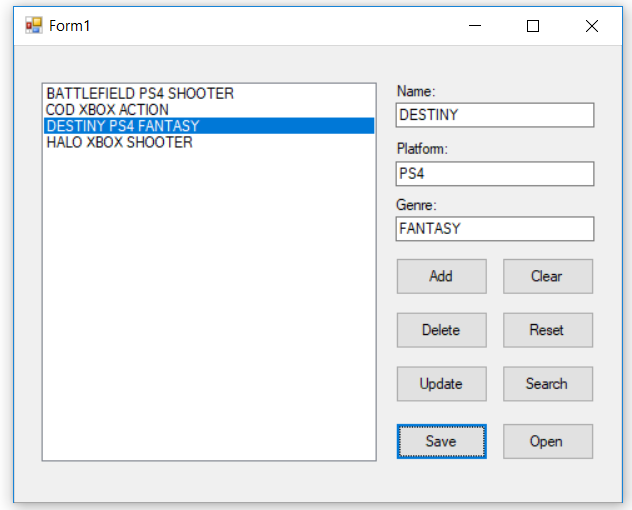
Search:



Press the Search Button

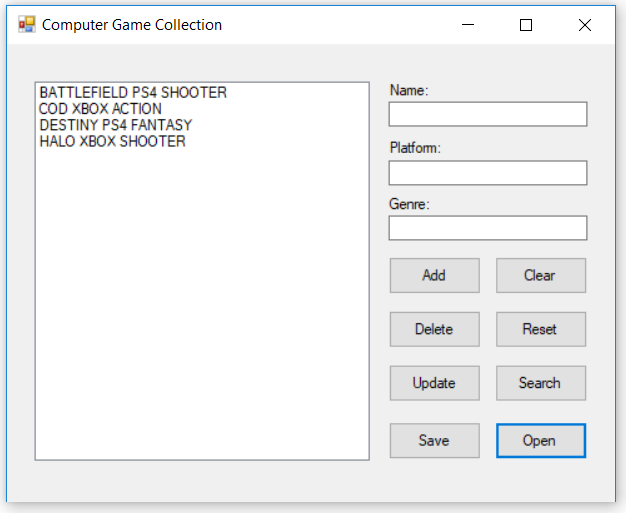
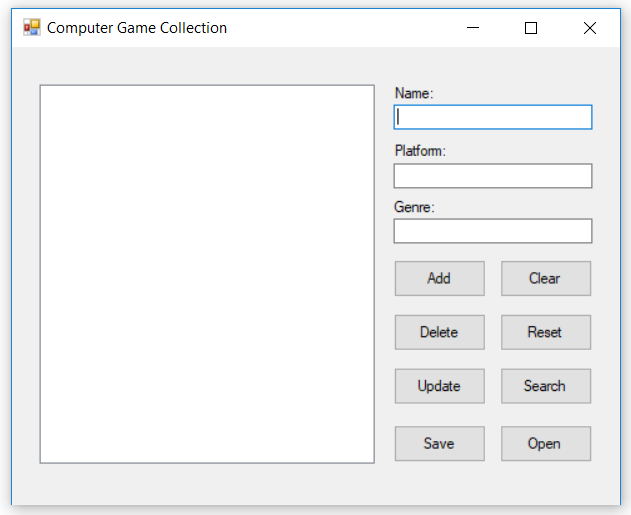
Enter name you would like to search

Save:



To save the file press the Save Button

Open:



To open the saved file press the Open Button