|  |  |
| --- | --- |
| VERSION | CHANGE DESCRIPTION |
| 1.0 | Initial code created by Machyl 30017609 |
| 1.1 | Updated Form:   * repositioned buttons, text box, and list box * added label at top for title of application * added background image and updated buttons, title label, to match background image of neutrinos   Code updated by: Bruce Fisher P197681 |
| 1.2 | * Organised code methods into logical block order   Code updated by: Bruce Fisher |
| 1.3 | * Added comments to code where needed * Added regions to code for readability * Added Tool Tips for buttons hover over   Code updated by: Bruce Fisher P197681 |
| 1.4 | * Check code logic and add comments to code * Change order of Methods in code to better reflect order on the form * Add regions to code for readability * Add tooltips to buttons * Change errors messages to Status Bar * Add messages for user to status bar to reflect button operations completed   Code updated by: Bruce Fisher P197681 |
| 1.5 | * Limited TextBox input to 2 digits only * Added clear TextBox when data entered is invalid * Binary search added message for when not found * Added clearing textbox contents once processing finished to prevent mistakes * Delete Button added confirm to delete Message * Fixed spelling for my method (which displays flashed errors in toolStripLabel) from DisplayTooLableMsg to DisplayToLabelMsg   Code updated by: Bruce Fisher P197681 |
| 1.6 | * Lists CR: Client Requirements & PR: Program Requirements for CODE and FORM DESIGNER   Code updated by: Bruce Fisher P197681 |
| 2.0 | Final version produced after Test Document completed  Errors found and fixed:   * Test Case 3 – Value not cleared from Text Box when error occurs “List already full” added to line 80 TextBoxMain.Clear() before displaying error message. * Test Case 5 – Value not cleared from Text Box when user selects NO from confirm to delete added to line 146 TextBoxMain.Clear() before displaying “Did NOT Delete…” message. * Test Case 6 – User after selecting NO to confirm delete can click on delete again without selected a value again from the list, added to line 147 SelectedIndex = -1 to clear the selected item.   Code updated by: Bruce Fisher P197681 |
| 2.1 | * Test Case 23 – Binary search algorithms cannot deal with finding a search item with a list of duplicate entrees very well. * Changed Binary Search Algorithm to deal with single and multiple searches found with the following code.   Line 196 to 198 added  int firstFound = 0;  int lastFound = 0;  bool foundDuplicates = false;  Lines now 203 to 241 amended code  if (NeutrinoInteractions[mid].HasValue && NeutrinoInteractions[mid].Value == inputInt)  {  firstFound = mid;  if (mid != 0 && (NeutrinoInteractions[mid - 1].HasValue && NeutrinoInteractions[mid - 1].Value == inputInt))  {  foundDuplicates = true;  while (firstFound != 0 && (NeutrinoInteractions[firstFound - 1].HasValue && NeutrinoInteractions[firstFound - 1].Value == inputInt))  {  firstFound--;  }  }  lastFound = mid;  if (mid != NeutrinoInteractions.Length - 1 && (NeutrinoInteractions[mid + 1].HasValue && NeutrinoInteractions[mid + 1].Value == inputInt))  {  foundDuplicates = true;  while (lastFound != NeutrinoInteractions.Length - 1 && (NeutrinoInteractions[lastFound + 1].HasValue && NeutrinoInteractions[lastFound + 1].Value == inputInt))  {  lastFound++;  }  }  if (foundDuplicates)  {  SelectedIndex = firstFound; // set item found in list to first item found  ListBoxMain.SetSelected(firstFound, true);  DisplayToLabelMsg("Multiple Values Found : " + NeutrinoInteractions[mid] + " at indexes " + (firstFound + 1) + " to " + (lastFound + 1));  SelectedIndex = -1; // Set nothing selected from list box  TextBoxMain.Clear(); // clear search item  return;  }  else  {  SelectedIndex = mid; // set item found in list  ListBoxMain.SetSelected(mid, true);  DisplayToLabelMsg("Value Found : " + NeutrinoInteractions[mid] + " at index " + (mid + 1));  SelectedIndex = -1; // Set nothing selected from list box  TextBoxMain.Clear(); // clear search item  return;  }  }  New Binary Search Developed and Coded by: © Bruce Fisher P197681 |
| 2.2 | * Test Case 25 – User input to Text Box within range 10 to 99 only added to lines 59, 103   if (Int32.TryParse(inputText, out inputInt) && inputInt >= 10)  and line 185  if (!Int32.TryParse(searchText, out inputInt) || inputInt < 10)  and lines 85, 114, 188  DisplayToLabelMsg("Error - Please enter an integer 10 to 99");  Code updated by: Bruce Fisher P197681 |
| 2.3 | * Test Case 27 – Test that Text Box is refocussed after input from Text Box has been processed bug fix added to lines 49, 73, 82, 87, 113, 118, 142, 147, 154, 168, 188, 197, 240, 250, 268   TextBoxMain.Focus();  Code Update by: Bruce Fisher P197681 |
| 2.4 | * Cleaned up code further – removed redundant line 376   TextBoxMain.Text = "";   * Cleaned up code further put new binary search into a method, removed from lines 282 to 353 the old code to place in method in utilities region. * Added call for new binary search to line 202   DuplicateBinarySearch(inputInt);   * Added new binary search method to line 282 to 353 within utilities region.   #region New Binary Search  private void DuplicateBinarySearch(int inputInt)  {  int min = 0;  int max = NeutrinoInteractions.Length - 1;  int mid;  int firstFound;  int lastFound;  bool foundDuplicates = false;  while (min <= max)  {  mid = (min + max) / 2;  if (NeutrinoInteractions[mid].HasValue && NeutrinoInteractions[mid].Value == inputInt)  {  firstFound = mid;  if (mid != 0 && (NeutrinoInteractions[mid - 1].HasValue && NeutrinoInteractions[mid - 1].Value == inputInt))  {  foundDuplicates = true;  while (firstFound != 0 && (NeutrinoInteractions[firstFound - 1].HasValue && NeutrinoInteractions[firstFound - 1].Value == inputInt))  {  firstFound--;  }  }  lastFound = mid;  if (mid != NeutrinoInteractions.Length - 1 && (NeutrinoInteractions[mid + 1].HasValue && NeutrinoInteractions[mid + 1].Value == inputInt))  {  foundDuplicates = true;  while (lastFound != NeutrinoInteractions.Length - 1 && (NeutrinoInteractions[lastFound + 1].HasValue && NeutrinoInteractions[lastFound + 1].Value == inputInt))  {  lastFound++;  }  }  if (foundDuplicates)  {  SelectedIndex = firstFound; // set item found in list to first item found  ListBoxMain.SetSelected(firstFound, true);  DisplayToLabelMsg("Multiple Values Found : " + NeutrinoInteractions[mid] + " at indexes " + (firstFound + 1) + " to " + (lastFound + 1));  SelectedIndex = -1; // Set nothing selected from list box  TextBoxMain.Clear(); // clear search item  TextBoxMain.Focus(); // place cursor back in TextBoxMain  return;  }  else  {  SelectedIndex = mid; // set item found in list  ListBoxMain.SetSelected(mid, true);  DisplayToLabelMsg("Value Found : " + NeutrinoInteractions[mid] + " at index " + (mid + 1));  SelectedIndex = -1; // Set nothing selected from list box  TextBoxMain.Clear(); // clear search item  TextBoxMain.Focus(); // place cursor back in TextBoxMain  return;  }  }  else  {  if (NeutrinoInteractions[mid].HasValue && NeutrinoInteractions[mid].Value > inputInt)  {  max = mid - 1;  }  else if (!NeutrinoInteractions[mid].HasValue || NeutrinoInteractions[mid].Value < inputInt) //assuming nulls where sorted to start of array  {  min = mid + 1;  }  }  DisplayToLabelMsg("Value NOT Found : " + inputInt);  SelectedIndex = -1; // Set nothing selected from list box  TextBoxMain.Clear(); // clear search item  TextBoxMain.Focus(); // place cursor back in TextBoxMain  }  }  #endregion  Code Update by: Bruce Fisher P197681 |