|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CS 1400 Lab #22**  **Partially Filled Arrays and the Split Method**  **Version 1.0**  **Objectives:**  The objective of this assignment is to help you better understand how to work with partially filled arrays and how to use the string Split method.  **The Split Method**  You will have to use the ***Split*** method to complete this lab and you will also need to know how to use it to complete this week's programming project. You can find a description of the Split method below:   |  | | --- | | **Using the Split Method**  Oftentimes you will read a line of text from the console that contains multiple pieces of data. For example, you might have an application that asks for a first name and a bowling score to be entered on the same line, like this:  **Sam 230 score**  We have been using the ReadLine( ) method to read in one line of data. This works fine when there is just one piece of data on the line. But what do you do in this case, when a line contains two or more pieces of data?  **The Split Method**  It turns out that the string class has a method named ***Split*** that will parse a string into individual pieces. These pieces are stored in an array of strings. The code to do this looks like the code below. This example reads one line of data from the Console and parses it into a string (the first name) and an integer (the bowling score) and a string description.  **enum** INDX { **FROW** **=** 0, **SROW**, **TROW** }  **// Prompt the user**  Console**.Write**(**"Enter your first name, score and description on one line\n**  **separated by a space: "**);  **// Read one line of data from the file and save it in inputStr**  **string** **inputStr** **=** Console**.ReadLine**();  **// The Split method creates an array of three strings**  **string**[] **scoreInfo** **=** **inputStr.Split**();  **// Parse each element of the array into the correct data type**  **string** **name** **=** **scoreInfo**[(**int**)INDX**.FROW**];  **int** **score** **=** **int.Parse**(**scoreInfo**[(**int**)INDX**.SROW**]);  **string** **description** **=** **scoreInfo**[(**int**)INDX**.TROW**];  Console**.WriteLine**(**"You entered\n**  **name:\t{0}score:\t{1:D}\tdescription:\t{2}"**  , **name**, **score**, **description**);  **. . .**  The general algorithm for reading and splitting a string containing multiple pieces of data separated by a space then looks like this:   * Read a line of data from the file and store it in a string. * Call the string Split method to break the string into an array of strings - each element of the array will contain one piece of data from the original string. * Parse each element of the array into its correct data type and save it. |   **Partially Filled Arrays**  Sometimes you will write a program where an array does not get completely filled with data. For example, I might write a grading program that contains an array of student scores big enough to hold 100 values. As long as a class has 100 students in it, your program will work just fine. Of course, most classes will have fewer than 100 students, and your program should work for them as well (partially filled array). To do this, you will have to keep track of how much data is actually stored in the array, and then use this number in any processing that you do, on the array. When you pass a partially filled array to a method, you will have to pass the number of values stored in the array as well or have a sentinel to signal the end of valid data. The Length property won't work -- in this example, it will always be 100. The **for** and **foreach** loop won't work either, unless you terminate the loop with a sentinel, otherwise it will process all 100 elements in the array.  **Programming Exercise**  Create an array of 25 int’s. Then ask the user to enter five int values separated by a “,” (comma) i.e. 20, 15, 19, 5, 1 <cr>. For this assignment, write code, that uses two methods:   1. Void GetData( ){…} is the first method that: 2. creates the array of int’s and fills it with int’s entered by the user. 3. The string entered by the user is split into five strings, using the string split method. 4. Then parsed to int’s and stored in the array of int’s. 5. void ProductData(int[] oArray, out int[] pArray, out int product){…} is the second method that is called from the GetData( ) method and it then: 6. Adds 2 to every element of the array passed to ProductData( ) method and stores it in a separate array. 7. Produces the product of all the values in the second array. 8. Returns both the product and the second array. 9. Make sure that your methods will handle a partially filled and a full array, by passing the number of values stored in the array as a parameter. When filling the array, allow the user to stop inputting data at any point by typing a zero.   **File(s) to Submit:**  Place your complete project folder in a zip file and name the zip file Lab\_22\_your-initials\_V1.0.zip. For example, I would name my file Lab\_22\_DAF\_V1.0.zip. Submit this assignment as Lab #22 on Canvas.   |  |  |  | | --- | --- | --- | |  | **Grading Checklist** |  | | # | Program | C(correct)  X(incorrect) | | 1 | Meets & works to specifications | 6 points | | 2 | Error Free, elegant & efficient | 4 points | | 3 | Pseudo-Code | -3 points | | 4 | Style Guidelines | -2 points | | 6 | Source Files(s) & Formatting | -2 points | | 7 | Project Prolog | -1 points | | 8 | Function Prologs | -1 points | | 9 | Zip Filename | -1 points | | 10 | Lab & Project Names | -1 points | | 11 | Zip File is invalid or will not unzip | Lab = 0 pts | |  | Total Points | 10 | 0-9 | |