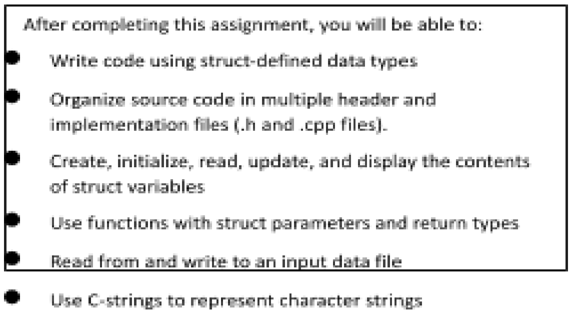
CS162 Assignment 1: Structs Review Airplane Information Database

For this assignment, write a C++ program that keeps track of some single engine aircraft. Load the airplane information into memory from a text file, so they can be listed and searched. When the program starts, your program will ask the user to enter the name of an input file, which your program will open and from which the airplane information will be loaded into memory using an array of structs. Each struct in the array will contain the information for one type of airplane. The struct definition is listed below in a textbox. The input text file uses a semicolon to separate the values. Once the data is loaded from the file, close it and only manipulate the data in memory (don't write to the file until program termination). When the program terminates, open the same file and write the data back to the original file, so that it will be updated with any new information.

Requirements:

You must use an array of structs to store the information. You must use c-strings (null-terminated char arrays) to store words. You may not declare string type variables. You may not use the <string> or <vector> libraries, or any other STL containers (<array>, etc). You must use functions and have multiple code files and header files. You must follow the coding style listed in the C++ style guidelines. See the requirements section below for more details.

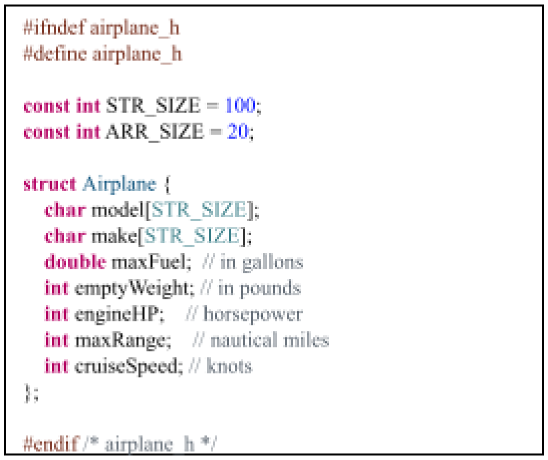


After completing this assignment, you will be able to:

- Write code using struct-defined data types
- Organize source code in multiple header and implementation files (.h and .cpp files).
- Create, initialize, read, update, and display the contents of struct variables
- Use functions with struct parameters and return types
- Read from and write to an input data file
- Use C-strings to represent character strings

Tasks:

- Complete the zylab from chapter 15, 15. CS 161B: Structs Part II zyLabs 15.9. The assignment will not receive a grade until you have completed the zylab with 100%, completed 70% of the challenge activity, and completed 80% of the participation activity.
- Open the Algorithmic Design Document, make a copy, and follow the steps to create your algorithm. You must complete the algorithm design document to receive a grade.
- Load the data from the database file in sorted order by name. You may not use a user-defined sorting function or sort function from a library - this means that each airplane must be inserted into the array in sorted order.
- Load the data when the program is started and write the data back to the file at program termination. Only load from the file at the start of the program. Only write to the file at program termination.
- After the file is loaded, the program will enter a user-controlled loop. Inside of the loop, the program will ask the user what they want to do.
- The rest of the tasks are split into 3 different versions based on the type of grade you want. There are 3 versions for this assignment, and each version has the minimum requirements for a particular grade – A, B or C. The B version builds on the requirements of the C version, and the A version builds on the requirements of the A version. Here are the requirements for each version:



```
#ifndef airplane_h
#define airplane_h

const int STR_SIZE = 100;
const int ARR_SIZE = 20;

struct Airplane {
   char model[STR_SIZE];
   char make[STR_SIZE];
   double maxFuel;  // in gallons
   int emptyWeight; // in pounds
   int engineHP;    // horsepower
   int maxRange;    // nautical miles
   int cruiseSpeed; // knots
};

#endif /* airplane_h */
```

Version C:

- List all the airplanes, one per line, using line numbers.
- Add a new airplane to the database in sorted order.
- Quit.

Version B:

- List all the airplanes, one per line, using line numbers.
- Add a new airplane to the database in sorted order.
- Remove an airplane by index (new for B version).
- Quit.

Version A:

- List all the airplanes, one per line, using line numbers.
- Add a new airplane to the database in sorted order.
- Remove an airplane by index.
- Search for an airplane by name (new for A version).
- List all airplanes by make (new for A version).
- Quit.

NOTE: There are different error checking requirements when adding a new airplane or removing an airplane for each one of the grade versions above:

- C version – check to make sure all number values are positive.
- B version – check to make sure all number values are positive and check for proper index to remove (not out of bounds). Also:
  - o Make sure the maximum fuel is less than 150 gallons.
  - o Make sure the empty weight is less than 3000 pounds.
  - o Make sure the engine horsepower is less than 400.
  - o Make sure the range is less than 2000.
- A version – All the error checks for version B and check for input failure (don't allow the program to crash or go into an infinite loop when letters are typed for input to numbers).

See the Programming Requirements section for more details.

The file format lists all the information for one airplane on one line, and includes:

1. model (name)	2. make (manufacturer)
3. fuel capacity	4. empty weight
5. engine horsepower	6. range (nautical miles)
7. cruise speed (knots)	

Make sure that there is a newline (return) at the end of every line, including the last line. Vim and vi will automatically make sure that your text files have a newline at the end of the last line, but you must add the newline to the end if you use TextEdit or Notepad. The function that reads data from the file must be able to handle the newline at the end of the file. If it's not handled correctly, then an empty struct will be added to the array. See the programming strategies section for more information. Along with this document there will be an example text file that contains the following airplanes in this format:

```
360;Lancair;43.00;1090;180;990;208
Skyhawk 172;Cessna;53.00;1663;180;515;123
K35 Bonanza;Beechcraft;70.00;1832;250;534;168
RangeMaster H;Navion;40.00;1945;330;1381;160
Tomahawk;Piper;30.00;1128;112;383;107
M20R Ovation;Mooney;89.00;2205;280;969;189
C23 Sundowner;Beechcraft;57.00;1494;180;564;115
RV-12;Vans Aircraft;20.00;750;100;451;119
TB-21 GT Trinidad;Socata;88.00;1911;250;1025;168
RV-9;Vans Aircraft;36.00;1057;160;616;163
152;Cessna;26.00;1081;110;414;106
Tiger;Grumman;51.00;1360;180;529;139
Super Cub;Piper;36.00;845;125;449;96
```

Sample Run / Example Output

The examples below have all the options, which is the A version of the assignment. For the C version, you will only present the options to Add, List, and Quit:

```
What would you like to do? (A)dd a plane, (L)ist all planes, or (Q)uit?
```

For the B version, you will have the options to Add, List, Remove and Quit:

```
What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane, or (Q)uit?
```

(User Input in Bold):

```
Welcome to the airplane collection program. What is the name of the airplane collection file? notAFile.txt
notAFile.txt was not found. Try again or type 'quit' to exit the program.
What is the name of the airplane collection file? planes.txt
Successfully loaded 13 airplanes.

     Model              Make      Fuel Capacity   Empty Weight   Horsepower   Range     Cruise speed
-----------------------------------------------------------------------------------------------------
1 .  152                Cessna           26.00         1081         110         414         106
2 .  360                Lancair          43.00         1090         180         990         208
3 .  C23 Sundowner      Beechcraft       57.00         1494         180         564         115
4 .  K35 Bonanza        Beechcraft       70.00         1832         250         534         168
5 .  M20R Ovation.      Mooney           89.00         2205         280         969         189
6 .  RV-12              Vans Aircraft    20.00          750         100         451         119
7 .  RV-9               Vans Aircraft    36.00         1057         160         616         163
8 .  RangeMaster H      Navion           40.00         1945         330        1381         160
9 .  Skyhawk 172        Cessna           53.00         1663         180         515         123
10.  Super Cub          Piper            36.00          845         125         449          96
11.  TB-21 GT Trinidad  Socata           88.00         1911         250        1025         168
12.  Tiger              Grumman          51.00         1360         180         529         139
13.  Tomahawk           Piper            30.00         1128         112         383         107

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? A
What is the model (name) of the airplane? Malibu Mirage
What is the make (manufacturer) of the airplane? Piper
What is the fuel capacity in gallons? One hundred and twenty
Please enter a decimal number for fuel capacity between 1.00 and 150.00.
What is the fuel capacity in gallons? 120.00
What is the empty weight? 20500
The weight must be a whole number between 1 and 3000 pounds.
What is the empty weight? 2435
What is the horsepower of the engine? 550
The horsepower must be a whole number between 1 and 400.
What is the horsepower of the engine? 350
What is the range? 1342
What is the cruise speed? 212
Successfully added Malibu Mirage to the database.

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? S
For what airplane would you like to search? Boeing 777
The Boeing 777 was not found in the database.

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? S
For what airplane would you like to search? Tiger
Information on the Tiger is as follows:
Make: Grumman, Fuel Capacity: 51.00, Empty weight: 1360, Horsepower: 180, Range: 529, Cruise speed: 139

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? L

     Model              Make      Fuel Capacity   Empty Weight   Horsepower   Range     Cruise speed
-----------------------------------------------------------------------------------------------------
1 .  152                Cessna           26.00         1081         110         414         106
2 .  360                Lancair          43.00         1090         180         990         208
3 .  C23 Sundowner      Beechcraft       57.00         1494         180         564         115
4 .  K35 Bonanza        Beechcraft       70.00         1832         250         534         168
5 .  M20R Ovation       Mooney           89.00         2205         280         969         189
6 .  Malibu Mirage      Piper           120.00         2435         350        1342         212
7 .  RV-12              Vans Aircraft    20.00          750         100         451         119
8 .  RV-9               Vans Aircraft    36.00         1057         160         616         163
9 .  RangeMaster H      Navion           40.00         1945         330        1381         160
10.  Skyhawk 172        Cessna           53.00         1663         180         515         123
11.  Super Cub          Piper            36.00          845         125         449          96
12.  TB-21 GT Trinidad  Socata           88.00         1911         250        1025         168
13.  Tiger              Grumman          51.00         1360         180         529         139
14.  Tomahawk           Piper            30.00         1128         112         383         107
```

```
What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? M
Please type the make of the airplanes you would like to list: Jabiru
There are no airplanes made by Jabiru in the database.

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? M
Please type the make of the airplanes you would like to list: Piper
The airplanes in the list made by Piper are:
     Model              Make      Fuel Capacity   Empty Weight   Horsepower   Range     Cruise speed
-----------------------------------------------------------------------------------------------------
1 .  Malibu Mirage      Piper          120.00         2435         350        1342         212
2 .  Super Cub          Piper           36.00          845         125         449          96
3 .  Tomahawk           Piper           30.00         1128         112         383         107


What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? R
Which index would you like to remove (1 - 14)? 0
Invalid Index. Please type an index between 1 and 14: 15
Invalid Index. Please type an index between 1 and 14: four
Invalid Index. Please type an index between 1 and 14: 9
Index 9 has been removed.

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? L

     Model              Make      Fuel Capacity   Empty Weight   Horsepower   Range     Cruise speed
-----------------------------------------------------------------------------------------------------
1 .  152                Cessna          26.00         1081         110         414         106
2 .  360                Lancair         43.00         1090         180         990         208
3 .  C23 Sundowner      Beechcraft      57.00         1494         180         564         115
4 .  K35 Bonanza        Beechcraft      70.00         1832         250         534         168
5 .  M20R Ovation       Mooney          89.00         2205         280         969         189
6 .  Malibu Mirage      Piper          120.00         2435         350        1342         212
7 .  RV-12              Vans Aircraft   20.00          750         100         451         119
8 .  RV-9               Vans Aircraft   36.00         1057         160         616         163
9 .  Skyhawk 172        Cessna          53.00         1663         180         515         123
10.  Super Cub          Piper           36.00          845         125         449          96
11.  TB-21 GT Trinidad  Socata          88.00         1911         250        1025         168
12.  Tiger              Grumman         51.00         1360         180         529         139
13.  Tomahawk           Piper           30.00         1128         112         383         107

What would you like to do? (A)dd a plane, (L)ist all planes, (R)emove a plane by index, (S)earch for a plane,
List all planes by (M)ake, or (Q)uit? Q

Database file updated. Terminating Program.
```

Academic Integrity

You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can't claim as your own. Please check out the Student Code of Conduct at PCC.

If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment. I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does and why you wrote it in a particular way, then you should not expect to receive credit. Do not share your code with any other students and do not post it on a public website.

Programming Requirements

- Don't use the string library or any of the STL containers such as vectors. Use the <cstring> library and c-strings (char arrays) to store strings. You may use a reasonable length for your c-strings, such as 128 or 256, declared as a global constant (remember not to use literals in your code for things like array size): `const int STR_SIZE = 256; const int ARR_SIZE = 30;`
- Create your array of structs in the main function. Do not use any global variables (constant globals are OK).
- You must use functions for this assignment. There are at least four that you must create for the C version, including: `loadPlanes(), writePlanes(), addAPlane(), and listPlanes().` For the B

version, you must add `remove()`, and for the A version, `listByMake()` and `search()`. You may add other functions, such as `welcome()` if you wish, and you may change the name of these functions, such as `removeByIndex()` instead of just `remove()`. These functions must be in a file other than main.cpp, and each function must have a prototype in a header file (do not place the prototypes in a code file).

- Make sure you check for index out of bounds before adding new data to the array. When loading data, compare the index to the ARR_SIZE (or whatever name you use for the size of an array) constant. If they are equal, there is no more room, so stop loading data. Suppose you use a variable called `count` to keep track of the number of airplanes in the array. Count will start out at zero (the first location in the array), and then add 1 every time you add a new airplane: `count++;` You can check if the array is out of room with an if statement:
  `if(count >= ARR_SIZE) // don't load the new data.`
  There are two functions where you need to do this check: `loadPlanes()` and `addAPlane()`. Alternatively, you can have `loadPlanes()` and `addAPlane()` call a third function, perhaps `insert()`, to do this check there. The `insert()` function is a good idea so you won't have to duplicate your code.
- Check to make sure that the input file is open before reading from it, using `ifstream.is_open()`. Ask the user to try a different file name if the file does not open or have them type "quit" to quit the program at that point (if they don't have a collection file).
- Make sure that you check for negative numbers and bad data when reading number data from the user.
- For version C, make sure the number values are positive (greater than zero).
- For version B, check the values for version C, but also make sure the values are less than the values spelled out in Tasks - B version. For removal in version B, make sure that the index is not out of bounds.
- For version A, do all the error checking for version B but also make sure your program doesn't crash or go into an infinite loop for input failure.
- Separate your project into multiple files. Notice that the textbox above containing the struct information has file guards: `#ifndef`, `#define` and `#endif`. These are the preprocessor directives that should be placed around the contents of a header file. Then use `#include "plane.h"` to include the header file in your code files. You should never include code (.cpp) files in other files. Only header files should be included in other code or header files. Also notice that to include a header file, you use the double quotes around the file name.
- You must place the struct definition in a header file and not in a code file, and it must have the format (number and type of the member variables) listed in the textbox above. You may change the name of the struct and member variables, such as, instead of `struct Airplane`, you could use `struct planeInfo`, and perhaps just `weight` instead of `emptyWeight`.
- You must have a main.cpp file and a plane.h file, but you may have other files as well. To help me streamline my grading, please use these standard names:
  - `tools.h` and `tools.cpp`, or
  - `functions.h` and `functions.cpp`.

  Please don't use any other names for your code and header files, and don't change the capitalization.

Programming Strategies

- You may assume that the input file is formatted correctly, so no error checking is necessary when reading from the file. There are a few different ways to check for non-number input. The first method is to check for input failure, and the other is to use atoi(), found in the `<cstdlib>` library. Suppose you have a variable called `planeWeight` set up to take input from cin. After storing the input, use a while statement to check for input failure and negative values:

```
cout << "What is the empty weight? ";
```

```
cin >> planeWeight;
while(cin.fail() || (planeWeight <= 0 || planeWeight > 3000)) {
    cin.clear();                    // clear the fail state.
    cin.ignore(LARGE_NUMBER, '\n');  // remove chars from the input buffer.
    cout << " The weight must be a whole number between 1 and 3000 pounds. ";
    cout << "What is the empty weight? ";
    cin >> planeWeight;
}
```

The other way to check for non-number input is to use atoi(), which means "ascii to int." If the input string can't be converted to a number, then atoi will return zero. So, check for input failure and negative values by checking the result for less than or equal to zero:

```
char weightStr[STR_SIZE];
cout << "What is the empty weight? ";
cin.getline(weightStr, STR_SIZE);
planeWeight = atoi(weightStr);
while(planeWeight <= 0 || planeWeight > 3000) {
    cout << "The weight must be between 1 and 3000 pounds. ";
    …
}
```

By the way, don't copy-and-paste any of this code from the document into your code file. If you do, you may have non-ASCII characters in your code file, which will cause errors.

- You can use the example database file, or you can create your own input file using your favorite text editor on the Linux system: vim, emacs, nano, etc. If you use a Windows or Mac program like Notepad++ or TextEdit, make sure that the last character in the file is a newline (return character), and make sure you save the input file in text file format with Unix line termination, rather than rich text, Word, or some other format. Use the <fstream> library to load the file. You must use C++-style input and output objects and manipulators; do not use any C-style functions for input and/or output, such as fprintf, fscanf, etc.
- When doing list output, you may use setw() and left/right from the <iomanip> library to align the data into columns, but it is not required. Use fixed, showpoint, and setprecision(2) for floating point number output.
- Do not assume that the input file contains a certain number of airplanes. Read from the file using an EOF-controlled loop (loop to the end-of-file marker. See below for more details).
- Do not use regular cin or ifstream extraction (>>) to read data into any c-strings, because this is a potential security issue called buffer overflow. Extraction also stops at whitespace, so you won't be able to take multiple-word input. Use `cin.getline()` or `inFile.getline()` instead. These functions can take 2 or 3 arguments: the c-string, the number of characters to read, and the delimiter (optional). If you leave off the delimiter, the default is the newline ('\n'). The other version of `ifstream.getline()` allows you to set the delimiter. For example, the semicolon: `planesFile.getline(name, STR_SIZE, ';');` There is another method called get() to read data from an input stream. The difference between `get()` and `getline()` is that `getline()` removes the delimiter from the stream, whereas `get()` does not. So you will most likely want to use `getline()`. For more information, check out the cplusplus.com page.
- You should ask the user for the file name in main so it can be opened as an input file first, and then later as an output file. You can either create your ifstream object in main, or wait to create it in the `loadPlanes()` function. Don't forget to close the file at the appropriate time. If you create the ifstream object in main, then you must pass it as a reference parameter to the `loadPlanes()` function. If you create the file in the `loadPlanes()` function, then you will have to pass the filename as a c-string. Remember to close the file when you are done reading from it. Use the same name for an ofstream for `writePlanes()`.
- The struct array must be passed to all of the functions (`loadPlanes(), searchPlanes(), addAPlane(), and listPlanes()`), because the array must be declared in main. So, the prototype for the `loadPlanes()` function would be either:

`int loadPlanes(planeInfo planes[], ifstream & planeFile);` or:
`int loadPlanes(planeInfo planes[], const char filename[]);`
The first version passes the file to the function, and the second one passes the name as a c-string. Notice that loadPlanes() has an int return type. This is so it can pass back the number of airplanes that were loaded into the array of planes (13 airplanes, in the example above). Announce the number of airplanes that were loaded when the function returns to main, and store the count in a local variable, so it can be passed to the other functions. You can use the return value to signal whether the search was successful or unsuccessful. The loadPlanes() function may also be unsuccessful, in which case it could return an error code. Remember not to use literals, so set up a constant: `const int ERROR = -1;`

- When testing your code for `loadPlanes(),` it may appear that your program is loading the last line of the file twice or there is an empty struct. What's really happening is that you are trying to read a line from the end of the file, after all the data has been loaded, but before the end-of-file marker has been reached. A common fix for this is to always read a new line as the last statement in a loop, and check for eof in the loop condition. This strategy may also require you to read data once before entering the loop. Another common way to check for end of file is to use a getline statement as the loop condition: `while (inFile.getline(myCString, STRSIZE)) // false if eof()`. The other way to check for eof() is to use a peek() statement near the end of the loop: `inFile.peek()`. If peek() encounters the end of file marker, then inFile.eof() will return true.

Before you get started:

- **Check out the video store example in zyBooks, CS161B:Section 15.6**
- **Check out the insert sorted into an array example in zyBooks, CS 161B:Section 15.7**
- Complete zyBooks section **15. CS 161B: Structs Part II zyLabs 15.9.**
- Open the Algorithmic Design Document, make a copy, and follow the steps to create your algorithm.
- **Be sure to include your screenshot of the zyBooks completion in the algorithm document. The screenshot must be present for your assignment to be graded.**
- Create your source code files, build and test your program.
- Code may be written using any development environment but must use Standard C++.
- PSU-bound students are strongly encouraged to do all development on the PCC Linux server.
- Please open and compare your work with the grading rubric before submitting.
- Remember to follow all C++ style guidelines.
- Download the algorithm document as a PDF file, compress it with the other code files and your txt file, and submit to the D2L dropbox.
- You may express your algorithm as pseudocode or a flowchart. Please note that your pseudocode must follow the syntax requirements shown in the document - **you may not use C++ syntax as a substitute for pseudocode**.

Additional Support

- Post a question for the instructor in the Ask Questions! area of the Course Lobby.
- Set up a Zoom session with the instructor or the tutor.
- Go to the Discord server and post a question there.
- Make sure you complete all zyBook activities.