Lab 10 - Struct & Vector

Let's create a ***very*** simple social media system. We will have users (each with their own ID number), and messages from one user to another. These messages can also be public (directed to everyone).

1. Display a Struct

1. Create a C++ program named **`lab10.cpp`** which will model social media posts.
   - NOTE: CourSys will expect the file to be named `lab10.cpp` to ensure it is submitted to the correct place.
2. Define a new `struct` globally (near the top of the file; outside of all functions):
   - name the struct `post_t`. It is referred to in this document as the "*post struct*"
   - have it contain:
     - an `int` named `authorID`: The ID number of the person posting the message.
     - an `int` named `targetID`: The ID number of the person receiving the message.
     - a `string` named `message`: The text of the social media post.
   - Hint: Your struct to store a single social media post might look a little like:

```
struct post_t {
  int authorID;
  // Some fields omitted...
};
```

3. The `targetID` will tell the program whom the post is shared with (one person at a time).
   - The ID can be set to 0 to indicate that the post is public post (anyone can see it).
   - Create a named global constant to store the ID used for public posts (i.e., create a global *constant* of value 0, which is the value used for `targetID` for public posts).
   - Remember to make constants all upper-case like POST_ID_PUBLIC.
4. Create a well named function which accepts a single *post struct* and displays it to the screen. You may use pass-by-value, by-reference, or by-constant-reference (your choice).
   - Have your function print out the post's `authorID`, `targetID`, and `message`.
   - Add the following code to `main()` to test your function (change the function name)!

```
// Initial code to figure out structs:
post_t demo = {101, 9999, "Just a test"};
daFunctionYouJustMade(demo);
```

   - When you run your program, it should output something similar to this:

```
101      9999 Just a test
```

5. **Understanding**
   - If your function accepts a struct named `post`, how can you access the `authorID` field of the struct?
   - What is the prototype for a function named `send` which accepts a *post struct* by reference?

2. Vector of Posts

1. Create the `readPostsFromFile()` function using the code below:

```
// Returns a new string with all whitespace (space, tab, \r) removed from
// beginning and end of the string.
string trim(string str)
{
    while (!str.empty() && isspace(str.at(0))) {
        str.erase(0, 1);
    }
    while (!str.empty() && isspace(str.at(str.size() - 1))) {
        str.erase(str.size() - 1, 1);
    }
    return str;
}

vector<post_t> readPostsFromFile(const string &fileName)
{
    vector<post_t> posts;

    ifstream file(fileName);
    if (file.fail()) {
        cout << "Unable to open data file.\n" ;
        exit(EXIT_FAILURE);
    }

    // Read file data
    while (true) {
        post_t post;
        file >> post.authorID;

        // Check if there is in fact data:
        if (file.fail() && file.eof()) {
            break;
        }

        // Read rest of struct
        file >> post.targetID;
        getline(file, post.message);

        // Trim any tailing spaces ('\r' or '\n') and store
        post.message = trim(post.message);
        posts.push_back(post);

        // Check for errors
        if (file.fail()) {
            cout << "Unable to read course info from file.\n";
            exit(EXIT_FAILURE);
        }
    }

    file.close();
    return posts;
}
```

   This includes a trim function to remove any extra ‘\r’ whitespace at the end.

2. Download the `postfile.txt` file from the course website and copy it into the same folder as your `.cpp` code for this lab.

3. From `main()`, call the `readPostsFromFile()` function and read the contents of `postfile.txt` into a vector of posts.
4. Create a well named function to count the number of "*public posts*" in a vector of posts.
   - Use **pass-by-const-reference** to pass the vector of posts.
   - Remember a "public" post is one with a `targetID` of 0; use your named constant.
   - From `main()`, call your function with the vector of posts from the file. Have `main()` print the count to the screen.

5. Create a well named function to find the `authorID` of the most prolific author: i.e., the author who creates the most posts (private or public).
   - Use **pass-by-constant-reference** to pass the vector of posts.
   - Use a **for-each loop** to iterate through the posts in the vector.
   - **Return** the ID of the most prolific poster (this function should *print* nothing).
   - From `main()`, call your function. Have `main()` use `cout` to print the ID your function returns.

   - Hints:
     - First create another function which is passed one user ID and the vector of posts. Have this function count and **return** the number of posts this user ID has posted.
     - Back in your first function, to find the most prolific author, count the number of posts for each author, selecting the maximum:
       1. for each post in the vector, get that post's author ID and count the number of posts that author has made.
       2. Track the maximum number you have found, and which author made that post.
       3. Return the ID of the most prolific author.
     - Note that function which has a parameter passed by *const-reference* may only pass it *by-value*, or *by-const-reference*.

       For example, if you have the following three functions:
       ```
       void fooByValue(vector<int> data);
       void fooByConstRef(const vector<int> &data);
       void fooByRef(vector<int> &data);
       ```

       Then if you write the following function:
       ```
       void myBar(const vector<int> &data)
       {
           fooByValue(data);
           fooByConstRef(data);
           // fooByRef(data);   // generates compile-time error
       }
       ```

       In other words, you cannot pass your *const-ref* parameter to a function which may try to change it. It can't change it; it's *const*! So the compiler won't allow it.
6. Sample output:
   ```
          101      9999 Just a test
   # Public posts:        4
   Most common author ID: 42
   ```

7. **Understanding**
   - How can we iterate through a vector of structs named `posts` using a for-each loop.
   - How can we iterate through a vector of structs named `posts` using a for loop.

3. Display All Posts

1. Create a well named function which displays all the posts in a vector of posts.
   - Use *pass-by-constant-reference* to pass in the vector.
   - Display a row of headings, then a row or "-"s and then the posts.
2. Sample output:

```
       101      9999 Just a test
# Public posts:        4
Most common author ID: 42
  AuthorID  TargetID Message
  --------  -------- -------
         1         0  Hello world!
         2         3  Hello you.
         3        52  I love you!
         1      2352  Yes!
        42         5  Space is big.
         5         2  On my way.
        42         0  Don't panic.
       352         1  Make it so.
         0         0  Is anybody out there?
        42        42  I'd far rather be happy than right any day
       343         0  Permission to give the covenant back their bomb?
```

3. Hint: you already wrote a function to display a single post: loop through all the *structs* and call that function! Change the function as needed to meet your current needs.
4. Test your function by calling it from `main()`.

4. Change All Posts

1. Create a function named `makeAllPostsHappy()` which accepts a vector of posts.
   - Pass the vector of posts using *pass-by-reference*.
   - For each post in the vector, append `" :)"` to the end of the message.
   You are adding a smiley to the message to force everyone to be happy. `:)`

2. Test your function by calling `makeAllPostsHappy()` before displaying all the *structs*:

```
       101      9999 Just a test
# Public posts:        4
Most common author ID: 42
  AuthorID  TargetID Message
  --------  -------- -------
         1         0  Hello world! :)
         2         3  Hello you.  :)
         3        52  I love you!  :)
         1      2352  Yes! :)
        42         5  Space is big. :)
         5         2  On my way. :)
        42         0  Don't panic. :)
       352         1  Make it so. :)
         0         0  Is anybody out there? :)
        42        42  I'd far rather be happy than right any day :)
       343         0  Permission to give the covenant back their bomb? :)
```


3. **Understanding**
    1. Why must `makeAllPostsHappy()` use pass-by-reference instead of by value or by constant-reference?

5. Lab credit

Submit the following to CourSys to get credit for the lab:

- Complete above steps in the files. OK to skip any “optional” sections.
- In your code, type your answers to Understanding questions.
- Submit your C++ code.

