This is a graded discussion: 5 points possible due Sep 4

82 106

Assignment 2 Discussion

Post your contribution to the assignment 2 discussion here. This could involve asking a question, answering another student's question, giving an example of something that you struggled with and then overcame (or didn't!), giving an example of something you found particularly cool, or any other constructive way you can think of to participate.

Search entries or author | Unread | Subscribed

Reply

Eric Barnard (https://santarosajc.instructure.com/courses/24402/users/68744)

(https://santarosajc.instructure.com/courses/24402/users/68744)

Aug 23, 2017

Dave, a quick question on the use of for vs while loops, when reading in c-strings.

In the syllabus, you state to only use for loops for counter controlled loops only. Perhaps this is either ambiguous or I'm not reading it right. Does 'counter controlled' refer only to when you are terminating the loop based on the value of a counter? What if we're using a counter in the loop (as is clearly done in some functions in this assignment), but it doesn't necessarily determine when the loop terminates? Or when you are using the counter as an iterator?

Seems to me its more succinct to use the for loop than a while loop, and separately using a counter variable.

Edited by Eric Barnard (https://santarosajc.instructure.com/courses/24402/users/68744) on Aug 23 at 7:21pm

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Aug 24, 2017

Counter (or count) controlled loops are defined in most intro textbooks and also in lesson 4.3. In the lesson I say it like this:

a counter controlled loop is used when we know before entering the loop how many times it will be executed.

My personal preference -- and the style convention -- is to only use for loops in this case -- when all three components of the for loop header refer to the counter.

I wouldn't penalize you if you used the for loop for a different type of loop as long as it is reasonable.

Reply

(https://santarosajc.instructure.com/courses/24402/users/74745)

James O'Hara

(https://santarosajc.instructure.com/courses/24402/users/74745)

Aug 27, 2017

t.a. Jim's Hopefully Helpful Hints for Assignment 2

1. In 2.2, do not design your program to store the user's input in an array (or vector, or list, or tree...no containers for the user input). There should of course be an array for the frequency table. Dave has repeatedly rewritten the instructions to try to clarify this (without giving away too much), so hopefully it won't be much of a problem. But, if you find yourself unable to design under this constraint, please discuss on Assignment 2 Discussion.

2. In 2.2, you should not need to loop through your frequency table looking for a match for a user input character. Because chars are numeric values in memory (as is all data), the frequency table can be designed so that the index for each struct can be deduced from the letter for which it is keeping score (so that, when it's time to match user input, you can know the index of the matching struct just by knowing the character's value--if the input character is 'h' (or any letter), you can know exactly where it goes, immediately (after handling any upper/lowercase issue). Hint: Run this code:

```
int a = 'a', b = 'b', z = 'z', A = 'A', B = 'B', Z = 'Z';
cout << "a " << a << " b " << b << " ... z " << z << endl;
cout << "A " << A << " B " << B << " ... Z " << Z << endl;
```

3. Students often lose points in 2.1 by overlooking or forgetting bullets 2 and 3 in "Please note the following:", and the bullets about not using a second array for reverse() and isPalindrome().

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Aug 27 at 6:21pm

(https://santarosajc.instructure.com/courses/24402/users/114750)

Terrence Dubois

(https://santarosajc.instructure.com/courses/24402/users/114750)

Aug 30, 2017

I am a bit confused on your first point. If we are not allowed to use variables of type string, and C-strings are stored in arrays (which you say should not be used either), how would you store any input from the user? I think I am just misunderstanding something here.

(https://santarosajc.instructure.com/courses/24402/users/60154)

David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Aug 30, 2017

You'll want to store the user's input in a single character variable. This will be just for an instant, so you can use that input to determine which letter needs to be incremented, then you'll store the next character in the same single character variable.

(I don't say that you should not use arrays in the assignment.)

Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934)

(https://santarosajc.instructure.com/courses/24402/users/64934)

Sep 5, 2017

So we stored the user's input, one character at a time, in the char variable in each element of the array right? For example, if the user enters 'ab cd', we store them individually in the 5 char variables found in the first 5 elements of the array?

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 5, 2017

Probably not. There is only one array, the array of struct, which is sort of a scoreboard, with counts for each letter. The sorted array, without the zeroes, for "do be do be do." would be (I think, only eyeballed this):

```
letter    timesUsed
   d           3
   o           3
   b           2
   e           2
```

The array should be filled with 26 letters and 0 times used for each letter, before processing any user input. (There are other acceptable ways to do this, but I recommend this). After that, no letters are added to the array--all the letters are already in the array before you look at any user input. Then as you examine each letter in the input stream, increment "timesUsed" for that letter.

In assignment 1.2, you took user input from the console input stream (cin), and inserted it into int variables (cin >> numScores; cin >> scores[i];). In this assignment, you want to take user input from cin, and insert it into what? Remember that you want examine each letter in the input stream. So, you want to see the characters one-by-one. So where would you target your insert from cin? (cin >> __?__)

Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934)

(https://santarosajc.instructure.com/courses/24402/users/64934)

Sep 5, 2017

Into an array? Dave mentioned above that we should store the input in a single character variable, but how can a string literal with multiple characters be stored in a single character variable? Wouldn't that require an array? Or should we create another struct? I'm confused.

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 5, 2017

Don't store the user input into an array--sorry if what I wrote seemed to say that, not what I meant. Sorry, missed Dave's saying that about a char variable, otherwise I would have said that too. There's no string literal, as far as the program is concerned. There is a stream of characters (cin), a struct consisting of a letter and a number, an array of those structs (to keep the counts of the various letters in the input), and a single char variable. You use

the insertion operator on the input stream, and direct the stream to the char variable. Once it inserts one character from the input stream into the char variable, it will stop, until it (the insertion operator) is called again.

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 6, 2017

Here is a sketch. "cin" is a C++ object, so should probably be contained by the "a2_2.exe" box. But it's an object for the input stream, so I'm showing it as a label for the stream of characters from the keyboard to the program.

a2_2.png (https://santarosajc.instructure.com/files/938613/download?download_frd=1&verifier=RAppjUtfhzAynC9ay4lVhKldXYvcsRrzNAjEpGQ1)

Reply

Jesse Browne (https://santarosajc.instructure.com/courses/24402/users/114708)

(https://santarosajc.instructure.com/courses/24402/users/114708)

Aug 29, 2017

I found that, when doing assignment 2.2, there were multiple ways to solve it. I've done similar problems before where I count the times each character appears in a string using an array of integers (indexed alphabetically in this case). However, the need to sort them means you can no longer tell which count belongs to which character based on its index alone, hence the need for the structure.

So, the first thing I tried was first initializing an array of the structure, one for each letter, and changing their counts as I read the input. Afterwards, I could sort them by their counts. The only thing I didn't like about this is the need to loop initially in order to initialize the array with all the letters in the alphabet.

My solution right now is to first use an array of integers, populate it with counts based on the input, then create an array of the structure. I can know the size the array of structures will need to be by counting the number of unique letters in the input. I can then insert the correct counts and letters into it in descending order. I realized after doing this that with my first solution I could actually skip the first loop and only set the letter of the structure when I encountered it in the input, but I still think this is a less ideal solution. My reasoning is that I would still have to sort an array of 26 structures making the time complexity of the sort always $26 * 26$ or 26^2 (using a quadratic sorting method). However, by first using the array of integers and counting the unique occurrences (thus being able to only create an array of structures that is exactly the size needed), I can reduce the time complexity of the sort in cases where less than the full alphabet is found in the input to $N * 26$ where N is the number of the unique letters found.

Eric Bernard (https://santarosajc.instructure.com/courses/24402/users/68744)

(https://santarosajc.instructure.com/courses/24402/users/68744)

Aug 29, 2017

Seems to me your 2 techniques performance largely depends on the length of the user input.

Using your 'less than ideal solution', no matter what, when you sort its always sorting 26 objects regardless of the length of user input. And with your current solution using an array of ints to tally prior to creating your array of

structs, you may indeed save some rather small amounts of time if all 26 letters are not entered, and they are entered a relatively small number of times.

On the flipside however, with your current solution, the longer the user input is, the more you are traversing your array of ints. But with the 'less than ideal solution', the only time you are ever looping through the array is when sorting (with the correct technique that James was hinting at above). And again, its a static number of times.

My current solution is your 'less than ideal' solution, but I'm trying to figure out a way to reduce the sort algorithm to only hit the structs with letters that were entered, instead of just blindly looping through each struct including those with a count of zero.

*Edited by **Eric Barnard** (https://santarosajc.instructure.com/courses/24402/users/68744) on Aug 29 at 8:06pm*

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Aug 30, 2017

Eric, regarding your last paragraph, I don't think that is possible (at least not in any way that would be remotely practical).

Jesse Browne (https://santarosajc.instructure.com/courses/24402/users/114708)

(https://santarosajc.instructure.com/courses/24402/users/114708)

Aug 30, 2017

In some ways, yes, but only in that the time it takes to fill the initial array of ints is always dependent on the length of the input, because I have to read each character. Inserting into the array of ints is constant time because you can get the index of the letter in the alphabet easily. However, the performance of the sorting algorithm is not dependent on the length of the input because I am always iterating through the counts of 26 characters and I do that iteration for every 'unique' letter encountered (<= 26 times). Obtaining the number of unique letters in the input doesn't require any additional iteration (I do it while reading). Since I am sorting while creating the structs, I don't have to iterate through that array of structs until the last part when I print their values. So I actually don't have to iterate through an array any more than I would with my less than ideal solution.

I also thought about how one might make the sorting algorithm $N * N$ but it seems that you would have to lose the constant time insert for the counts, making the time complexity much more dependent on the length of the input. It's definitely an interesting problem to explore. I wonder if a different type of data structure could be used in order to keep the counts sorted as you read them and if it would be faster to do it that way, considering you would have a slower insert while counting occurrences.

Reply

Rory Rogers (https://santarosajc.instructure.com/courses/24402/users/113253)

(https://santarosajc.instructure.com/courses/24402/users/113253)

Aug 30, 2017

How do we format our function comments I missed that part somewhere

(https://santarosajc.instructure.com/courses/24402/users/74745) James O'Hara

(https://santarosajc.instructure.com/courses/24402/users/74745)

Aug 30, 2017

The program outline should be

```
/*
initial file comment (about 50 words for early assignments)
*/
function prototypes
int main(){
    code for main()
}
function definitions, with function comments above, and at
least six blank lines between.
```

The requirements for the function comments are in Style Convention 1C. Here is one way of doing them.

```
/*
bool isLetter(int count);
This function returns true if the int parameter has the same
numeric value as a char with a letter value. Otherwise it
returns false. Unfortunately, this function violates Style
Convention 7a, because it uses numeric literals instead of
global constants.  On the plus side, it conforms to Style
Convention 7p.  Why this programmer didn't just use isalpha()
is unknown.

parameter(s): count. The value tested.
*/
bool isLetter(int count){
    return (count > 64 && count < 91) ||
        (count > 96 && count < 123);
}
```

Does that help?

(https://santarosajc.instructure.com/courses/24402/users/113253) Rory Rogers

(https://santarosajc.instructure.com/courses/24402/users/113253)

Aug 31, 2017

Yes thank you!

(https://santarosajc.instructure.com/courses/24402/users/60154)

David Harden

(https://santarosajc.instructure.com/courses/24402/users/60154)

Aug 31, 2017

I'm going to have to deduct some points from Jim's example, because the style convention required you to mention the parameters by name, which Jim did not do.

(Class, don't worry, I'm just poking fun at Jim, I wont actually take any points away from him.)

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Aug 31, 2017

I'd have to pass those costs on to the consumer. If I had any points to deduct from.

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Aug 31 at 5:20pm

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Aug 31, 2017

Oh, my, Jim pointed out to me that his example actually DOES mention the parameter by name. I guess it's a good thing he will be grading your work instead of me!

Reply

Melissa Goh (https://santarosajc.instructure.com/courses/24402/users/63762)

(https://santarosajc.instructure.com/courses/24402/users/63762)

Sep 1, 2017

Hi,

In 2.1, index would start from count 0, correct?

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 2, 2017

Do you mean the array of char for a c-string? Yes, any array's first element is index 0. (In Assignment 1.1 we could refer to the first element in a as p2[0] for instruction 15. That meant, take the address of a, and then go 0

integers of memory further. So you can think of array indexes that way--go to the address of the beginning of the array, and then go index elements further, to get the desired element. If you want the first element, you go 0 elements from the beginning). Does that help?

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 2 at 7:39am

Melissa Goh (https://santarosajc.instructure.com/courses/24402/users/63762)

(https://santarosajc.instructure.com/courses/24402/users/63762)

Sep 2, 2017

Hi, James, that helps! Thanks!

Reply

Harry Doyle (https://santarosajc.instructure.com/courses/24402/users/118245)

(https://santarosajc.instructure.com/courses/24402/users/118245)

Sep 1, 2017

What I found interesting for the palindrome and the reverse function in 2.1 was walking through an array from the left and the right at the same time and then meeting in the middle (for reverse, you end up swapping left and right, and doing a comparison for palindrome) and creatively doing that with as few statements as possible. I leveraged the length operator / 2 to figure out how far I needed to walk in from the left (and thus the right as well), and that works well with odd numbers as it would just drop off the .5 and i don't need to compare or reverse the middle character. Fun exercise to get your brain thinking.....

Edited by Harry Doyle (https://santarosajc.instructure.com/courses/24402/users/118245) on Sep 2 at 7:42am

Andrew Bieneman (https://santarosajc.instructure.com/courses/24402/users/99711)

(https://santarosajc.instructure.com/courses/24402/users/99711)

Sep 4, 2017

That's slightly more efficient than the way I did it. I checked every step of the way if I was "done".

i.e.

```
while(indexOfRightCharacter > indexOfLeftCharacter) {
    doPalindromeStuff();
  indexOfRightCharacter--;
  indexOfLeftCharacter++;
}
```

with your way it's just one arithmetic operation, then make a for loop

Edited by Andrew Bieneman (https://santarosajc.instructure.com/courses/24402/users/99711) on Sep 4 at 8:44pm

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 5, 2017

The other way also has to do a compare to test for done/not done. Actually, I think right > left is clearer. To me, anyway.

Reply

David Novosartov (https://santarosajc.instructure.com/courses/24402/users/76726)

(https://santarosajc.instructure.com/courses/24402/users/76726)

Sep 2, 2017

Had trouble implementing the swap() function with the array elements but managed to find a way, very fun assignment. It was also challenging to find a way to program 2_1 without using string functions other than 'strlen()'.

Greg Fischer (https://santarosajc.instructure.com/courses/24402/users/15956)

(https://santarosajc.instructure.com/courses/24402/users/15956)

Sep 2, 2017

2.1 If "flower" gets passed in it should be reversed ***in place*** to "rewolf".

Is there an example in the reading or book of how to do this, and pass to a function from main?

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 2, 2017

Hi Greg,

I think you are asking about how to pass a c-string as an argument, is that right?

The answer is, just exactly the same way you pass any other variable. Regarding the parameter, if you are treating it as a pointer, then the rules are exactly the same. If you are treating it as an array, the rules are slightly nuanced, and explained in lesson 9.3, "Arrays as Arguments and Parameters".

Let me know if i misunderstood your question.

Greg Fischer (https://santarosajc.instructure.com/courses/24402/users/15956)

(https://santarosajc.instructure.com/courses/24402/users/15956)

Sep 3, 2017

Yes, thanks. That section cleared some things up.

Reply

(https://santarosajc.instructure.com/courses/24402/users/63762)
Melissa Goh
(https://santarosajc.instructure.com/courses/24402/users/63762)
Sep 2, 2017

Hi! I need some confirmation on a2.2: the frequency table is displayed at the end of the program (to be summarized after a period appears and user hits enter)?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 2, 2017

Yes

Reply

(https://santarosajc.instructure.com/courses/24402/users/108173)
Olivia Piazza
(https://santarosajc.instructure.com/courses/24402/users/108173)
Sep 2, 2017

Hi, quick question about 2.1.

When the instructions say "*If "flower" gets passed in it should be reversed **in place** to "rewolf". For efficiency, this must be done "in place", i.e., without creating a second array*.", does that mean that I should not make a temp character and switch the array around using the temp character? Maybe another way I could phrase my question is this, should I be saving the reversed array back into the original array OR is the function just supposed to print the reverse of the array?

I am leaning towards the first scenario just because comparing the reverse array with the original for the palindrome function seems the way to go, but some clarification on this would be much appreciated :)

Cheers,

Olivia

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 2, 2017

A temp character is ok. (Also, there is a std::swap() function , very similar to the swap() written for assignment 1.1, except doesn't require pointer arguments. E.g., swap(char1, char2);) May have to #include <utility> or <algorithm>.

The function is *not* supposed to just print the array in reverse. When main() passes the array to reverse(), the array should be reversed when main() sees it again.

```
int main(){
...
reverse(myArray);
// If myArray was {'a', 'b', 'c', '\0'} before the call,
// it should now be {'c', 'b', 'a', '\0'}.
```

Notice you can't use reverse() in isPalindrome(). Again, check the array in place, without creating a second array.

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 2 at 7:41pm

Reply

Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934)

(https://santarosajc.instructure.com/courses/24402/users/64934)

Sep 3, 2017

In assignment 2.1 we are not supposed to create a second array if I'm being correct. But when I'm doing the palindrome problem I want to use an array to read in a palindrome entered by the user, should I first erase the first array that I created earlier?

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 3, 2017

You can create many arrays in 2.1. For example, you could have three arrays for each assigned function, and test the function three times, once with each array. And you could have six sets of three arrays, one set for each assigned function.

Let's say you test isAPalindrome() three times, with the strings "radar", "peanut butter", and "able was I ere I saw Elba". You can then have three c-string arrays, one for each test. But you can't create a new array inside isAPalindrome(). For example, it's illegal, within isAPalindrome(), to make a copy of the c-string argument, and then compare the last letter of the copy to the first letter of the argument, and so on.

Does that clarify?

(In fact, it's a good thing to have many tests of each function--try to think of normal tests, and also unusual tests, such as testing with an empty string, which is a palindrome, by the way).

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 3 at 9:50am

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 3, 2017

Just to add: The assignment says that ***the isPalindrome() function*** may not create a second array, so the restriction Hugh is asking about is not related to the situation he is describing (reading in the palindrome), because that is not done in the isPalindrome() function.

EDIT: If you read my post before, I probably totally confused you, because I was mixing up the reverse() function and the isPalindrome() function. I think I have it cleared up now.

Edited by David Harden (https://santarosajc.instructure.com/courses/24402/users/60154) on Sep 3 at 6:35pm

Hugh Lin (https://santarosajc.instructure.com/courses/24402/users/64934)

(https://santarosajc.instructure.com/courses/24402/users/64934)

Sep 3, 2017

Yes it does clarify. Thanks guys!

Reply

Lance Tran (https://santarosajc.instructure.com/courses/24402/users/112937)

(https://santarosajc.instructure.com/courses/24402/users/112937)

Sep 3, 2017

For assignment 2, part 2. My code seems to work but I would often get this debug warning. It had told me it's an error due to Run-Time Check Failure#2 - Stock around the variable 'arrayl' (my variable I used to have an array of 26 characters) was corrupted. I tried debugging but it wouldn't. I had to fix this situation by carefully not overloading on characters when I'm on the user input console screen, rather at least 1 - 8 characters, then it would run. It's odd but it works.

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 3, 2017

Sorry, but I can't figure out what you mean by "not overloading on characters" or what it means to be "on the user input console screen" or what you are referring to when you say "at least 1 - 8 characters." It sounds like maybe your program doesn't work if the user enters too many or too few characters? If that's the case, I hope you are able to get it fixed. Let me know if I can help.

Reply

(https://santarosajc.instructure.com/courses/24402/users/102629)
Carolyn O'Rourke
(https://santarosajc.instructure.com/courses/24402/users/102629)
Sep 3, 2017

Should our functions be in the same file as main for a2_1? If not what naming convention should we employ?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 3, 2017

Yes, just one file.

(https://santarosajc.instructure.com/courses/24402/users/102629)
Carolyn O'Rourke
(https://santarosajc.instructure.com/courses/24402/users/102629)
Sep 3, 2017

Thank-you I have a problem with reading too much into directions.

Reply

(https://santarosajc.instructure.com/courses/24402/users/4456)
Ryan Martins
(https://santarosajc.instructure.com/courses/24402/users/4456)
Sep 3, 2017

Ryan here I'm been looping on this replacement loop for too long now, it replaces all the targets but its not returning the number of times that it replaced it. hallppppp... I've rewritten it a million ways at this point and I'm sure its something super silly I'm over looking.

[code deleted]

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 3 at 5:01pm

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 3, 2017

Maybe the value of length is such that the line that returns the number of replacements never executes?

Reply

(https://santarosajc.instructure.com/courses/24402/users/114750)
Terrence Dubois
(https://santarosajc.instructure.com/courses/24402/users/114750)
Sep 3, 2017

Okay, so I've finished all three parts of the assignment. Assignments 2.1 and 2.3 were relatively simple for me, but I am still not completely happy with my solution for 2.2. I know this is late notice, but if anyone has more information on this I might go in and improve my code. The main issue I have been having is in getting the user's input, but other parts also feels a little tacky. I made an array of 26 structs that starts out filled with zeros. It takes each letter in a user given C-string and increments the letterCount of the character's numeric value minus 97 (because a - 97 = 0, b - 97 = 1, and so on). It then sorts the array of structs by the letterCount variable, and finally displays any score in the array that is larger than zero. It probably is not the ideal solution, but it was the only solution I could actually manage to get working.

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 4, 2017

Unfortunately, this won't get credit because it doesn't meet a requirement of the assignment that is stated like this:

No limit may be placed on the length of the input.

And again like this:

Don't forget that limiting the length of the input is prohibited. If you understand the above paragraph you'll understand why it is not necessary to limit the length of the input.

It also ignores this hint:

THEN cin.get(ch) reads the first character from the input stream, then it is counted, then the second time through the loop the second character is read from the input stream and counted.

Also, I suspect that it may not meet this requirement (although I'd have to see the code to be sure):

Your program should allow the user to enter multiple lines of input by pressing the enter key at the end of each line.

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 4, 2017

The good news is that the array of structs sounds like it was designed correctly. If the user input is read into a c-string, that's the problem Dave is talking about. But the part about updating the array based on the input character sounds right, although the detail isn't described exactly. If you can get away from the c-string, I think you may have a good 2.2

Terrence Dubois (https://santarosajc.instructure.com/courses/24402/users/114750)

(https://santarosajc.instructure.com/courses/24402/users/114750)

Sep 4, 2017

Thanks for the reply, Dave and James! I'm currently rethinking and rewriting my code to get something better that meets the requirements.

*Edited by **Terrence Dubois** (https://santarosajc.instructure.com/courses/24402/users/114750) on Sep 4 at 11:59am*

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 4, 2017

In retrospect, I was exaggerating the negative consequences a little. You could still receive some credit for the assignment even if the user's input is limited. Looks like I was a little grumpy at the time :)

Terrence Dubois (https://santarosajc.instructure.com/courses/24402/users/114750)

(https://santarosajc.instructure.com/courses/24402/users/114750)

Sep 4, 2017

No worries; I actually think I got it this time. I'm now using cin.get to go through all user-entered characters and using that single character variable you were talking about (not a C-string or array). You can enter as many lines as you want now.

Reply

Kevin Drake (https://santarosajc.instructure.com/courses/24402/users/4932)

(https://santarosajc.instructure.com/courses/24402/users/4932)

Sep 4, 2017

For assignment 2.1 you stated we could use the toupper function in the cctype library. You then have a void function called toupper in step 5. Is this for us to replace the one already included or can we just use the cctype library function?

*Edited by **Kevin Drake** (https://santarosajc.instructure.com/courses/24402/users/4932) on Sep 4 at 1:26pm*

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 4, 2017

The two functions have the same name but do different things. So, yes, you may use the cctype library toupper() function in your toupper() function.

Reply

(https://santarosajc.instructure.com/courses/24402/users/81230)
David Holstedt
(https://santarosajc.instructure.com/courses/24402/users/81230)
Sep 4, 2017

For 2.1 can we make our own function to print c-strings?

(https://santarosajc.instructure.com/courses/24402/users/60154)
David Harden
(https://santarosajc.instructure.com/courses/24402/users/60154)
Sep 4, 2017

Hmmm, not sure why you would want to do that when you can just use the built-in insertion operator (<<) to print them.

(https://santarosajc.instructure.com/courses/24402/users/81230)
David Holstedt
(https://santarosajc.instructure.com/courses/24402/users/81230)
Sep 4, 2017

Oops, I didn't realize that worked.

(https://santarosajc.instructure.com/courses/24402/users/80446)
Evan Maier
(https://santarosajc.instructure.com/courses/24402/users/80446)
Sep 11, 2017

I did the exact same thing...guess I should have tried << first.

Reply

(https://santarosajc.instructure.com/courses/24402/users/89449)
Lucas Bebervanzo
(https://santarosajc.instructure.com/courses/24402/users/89449)
Sep 4, 2017

Just a quick question about how arrays are accessed in C/C++.

If I had an array, and I tried to access the data at index i, does the computer traverse the entire array up to i? Or does it just go "straight there"? I'm asking because I realized that my justification for how I did part of 2.1 was based on a potentially incorrect assumption that accessing the data at index i requires traversing the array from 0 to i.

Sean Callahan (https://santarosajc.instructure.com/courses/24402/users/52648)

(https://santarosajc.instructure.com/courses/24402/users/52648)

Sep 4, 2017

The computer will go "straight there" like you mentioned. Since each item in the array has the same size, It can easily find the data at that index by moving the pointer at beginning of the array by the size of each item.

Something like this is done internally:

```
start_pointer + (sizeof(item_type) * index)
```

Hopefully that makes sense.

Calling strlen() is prohibited because it finds the length of the string by traversing the entire string until the null terminator ('\0'). The assignment calls for traversing the array once. Therefore, finding the length of the string first and then using something like a 'for' loop would be unnecessarily traversing twice.

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 6, 2017

Thanks Sean, good explanation.

Reply

Kevin Drake (https://santarosajc.instructure.com/courses/24402/users/4932)

(https://santarosajc.instructure.com/courses/24402/users/4932)

Sep 4, 2017

In problem 2.2 I am stuck on how to get multiple line on user input and how to read each character individually.

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 4, 2017

If you recall from 1.2 (and 2.3 this week), the user enters the number of scores for the High Scores game/contest. Through the magic of C++, the insertion operator takes the console input (cin) and inserts it correctly into an int variable. In 2.2, you want C++ to insert the console input into what? (so that you can read each character individually).

Although the user may enter multiple lines without a '.' character, those lines consist of individual characters that can be examined one-by-one in a loop. They don't need to be stored in an array.

If the user enters "do be do be do", you loop through the characters until you see a '.' Since you don't see a '.' in that example, your loop should process those 14 characters and then just sit there waiting for more input. No c-string formatting, no buffer array, no worrying about a string terminator. Just look for '.'.

Reply

(https://santarosajc.instructure.com/courses/24402/users/102629)
Carolyn O'Rourke
(https://santarosajc.instructure.com/courses/24402/users/102629)
Sep 4, 2017

I have implemented 2-2 with a static array of structures is that ok or should I be using a dynamic array since some letters may never be used?

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 5, 2017

Static is fine. Be sure to size it with a constant, not a variable.

Reply

(https://santarosajc.instructure.com/courses/24402/users/111918)
Neal Siegrist
(https://santarosajc.instructure.com/courses/24402/users/111918)
Sep 4, 2017

This program was probably the most challenging I have done between this class and CS10. It was very challenging to find a way to initialize the array in 2.2 effectively but I found a way to do it. Reading in the characters was a bit challenging in the beginning and allowing the user to hit 'enter' as many times as they liked. This was a very rewarding assignment in the end and it felt good to finish it after all the hard work it took.

(https://santarosajc.instructure.com/courses/24402/users/4456)
Ryan Martino
(https://santarosajc.instructure.com/courses/24402/users/4456)
Sep 6, 2017

I've been so stumped on this, written my code so many times that I don't know if I'm hotter or colder at this point. seems my array is set up right but reading them one at a time from the input working with that char and looping to

the next and so forth..

Reply

(https://santarosajc.instructure.com/courses/24402/users/100398)
Tristan Girard
(https://santarosajc.instructure.com/courses/24402/users/100398)
Sep 4, 2017

I felt really good about 2.1 and had a lot of fun relearning some of what I learned in CS 10. The manipulations of c-string null values was hard for me because I have never used c-string pointers before and they make sharing a variable between functions much easier.

I had a tough time remembering 2.2 but finally got some help by looking at past chapters. Unfortunately, I had some trouble creating the correct logic for 2.3, but finally used past tips from stack overflow to help.

Reply

(https://santarosajc.instructure.com/courses/24402/users/57042)
Tim Banch
(https://santarosajc.instructure.com/courses/24402/users/57042)
Sep 4, 2017

In the assignment description, you said to avoid using for loops with strlen() as the limit, but I couldn't find a way to do the reverse() and isPalindrome() functions without knowing the length of the C-string. As far as I can see, calling strlen() is the only way to look at the first and last elements. Am I missing something?

(https://santarosajc.instructure.com/courses/24402/users/99711)
Andrew Bieneman
(https://santarosajc.instructure.com/courses/24402/users/99711)
Sep 4, 2017

You're allowed to use strlen() for those two (I'm pretty sure). You're just not supposed to traverse the array unnecessarily (because that's what strlen does when you call it). If you really want to avoid it you could set up a variable for the string length, then run a while loop, incrementing the index of your char[] and the string length until you hit a null char. Not necessary though if I'm understanding the instructions right.

(https://santarosajc.instructure.com/courses/24402/users/81540)

Ryan Roncancio

(https://santarosajc.instructure.com/courses/24402/users/81540)

Sep 4, 2017

The way I did it is I created an int called "last" and stored strlen() - 1 in it. This way I only had to call strlen() one time but I still knew the last char of my c-string, with the first being str[0].

James O'Hara

(https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 5, 2017

Ryan's got it. Thanks, Ryan. OK, Tim? Andrew's right that sometimes you do need to find the string length, so calling strlen there is the way to go. But don't call it repeatedly, just the one time and store the result, as Ryan says, then use that to control your loop. Not sure about "incrementing ... the string length until you hit a null char" being used in reverse() and isPalindrome().

Reply

Fred Straub

(https://santarosajc.instructure.com/courses/24402/users/73008)

(https://santarosajc.instructure.com/courses/24402/users/73008)

Sep 4, 2017

I don't have anything to offer the class for this assignment. I procrastinated and thought I would work on this this weekend. Either I remained stumped, or I was rolled over by this weather, I am hopeful that this coming week will be better.

Oh yeah, no more procrastinating.

Aarole Mao

(https://santarosajc.instructure.com/courses/24402/users/84454)

(https://santarosajc.instructure.com/courses/24402/users/84454)

Sep 4, 2017

I'm not proud of it, but I am in the same boat -- I'm having more trouble with c-strings than I'd anticipated, and the amount of time I gave myself for these programs is not going to be enough.

But yes, going to try for better time management for future assignments!

Reply

Cinthya Rosales (https://santarosajc.instructure.com/courses/24402/users/110095)

(https://santarosajc.instructure.com/courses/24402/users/110095)

Sep 4, 2017

I found that assignment 2_1 was hard to comment on as some of the functions had fewer words than the comments ultimately did. It really made me take a step back and think thoroughly about what each function and its parameters were doing.

Reply

Ryan Roncancio (https://santarosajc.instructure.com/courses/24402/users/81540)

(https://santarosajc.instructure.com/courses/24402/users/81540)

Sep 4, 2017

I was having trouble in assignment 2.2 when I was initializing the objects of my struct. I was using a for loop where I added 65 to my counter variable, **int** i, and tried to store that number in the **char** variable of my struct (65 being the ASCII value for 'A'). I was getting an error about a narrowing conversion from **int** to **char**. I found that the easiest solution for my problem was to simply make my counter variable a **char** instead of an **int**. This way no conversion was required.

Reply

Mario Franco (https://santarosajc.instructure.com/courses/24402/users/19287)

(https://santarosajc.instructure.com/courses/24402/users/19287)

Sep 4, 2017

Well... It was interesting. This one made me work quite a bit! The last part was quite challenging since my original program was not very good. It took me several trials in order to make it work.

Reply

Craig Nicholas (https://santarosajc.instructure.com/courses/24402/users/118709)

(https://santarosajc.instructure.com/courses/24402/users/118709)

Sep 4, 2017

I must be the only one who had a problem with 2.3. The user input and output of the c-strings just wasn't working, after hours of wrangling. I must be doing something dumb.

In other news, I wasted a lot of time trying to do isPalindrome() with a single pass through (and without copying, without recursion, without having the length). It involved a lot of stupid gymnastics with iteratively constructed polynomials and symbolic representation of algebraic numbers. It took me too long to realize a one-pass function (with 100% accuracy) would need to be able to store an unbounded amount of information in order to "remember" the portion of the input its seen so far. I don't know if isPalindrome() is intended to work with any size of input but not having a bound on the length entails using some form of data that's flexible with the amount of information it can hold.

The ultimate futility of the task hit me when I looked up upper bounds on the magnitude of roots of polynomials with integer coefficients that are themselves bounded (because I want an algebraic number t so that p(t) = 0 implies p(x) has all zero coefficients). The coefficients are bounded by the maximum size of ASCII character codes, which is 127. So I can work out an upper bound... and it's 128. This is a power of 2, and the reason it's a power of 2 suddenly made clear why all this was dumb. Plugging in a sufficiently large number into polynomial is a cheap trick for constructing a number that's just the concatenation of the coefficients (in the right number base), which themselves represent ASCII characters. All this was just a heavily disguised version of copying the input into a new array. Hahaha you wouldn't believe how many pages of my notebook are filled with scribbles trying to make this work.

So I scrapped all that and did this other scheme with two indices stepping their way through the input and looking for candidates for a palindrome's center. I managed to get that work without first finding the length of the input. But even here there are echoes of the previous problem. This will fail on inputs that are longer than MAX_INT. A "simpler" form of my complicated approach would fail at something like log(INT_MAX). Does that difference really matter? Though there is a major qualitative distinction in terms of using data structures that carry significant information about the input.

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 5, 2017

I'm not aware of a solution to the palindrome problem that doesn't get the length of the c-string parameter. When Dave said not to over-use strlen(), he didn't mean not to use it at all, just as little as possible. It's good that you're thinking of corner cases, but it's okay to assume that the input length is less than INT_MAX (there are other integer types which have a greater top end value than int, but don't go there). Send me your 2.3 code using Canvas messaging, or send to Dave at **dharden@santarosa.edu, (mailto:dharden@santarosa.edu,)** if you're still stuck. Be careful of overthinking. Give your brain a chance to loosen its grip on the problem for a while; it may come up with a new approach while you're taking a walk and enjoying the cooler weather.

2.3 is basically the same as 1.2, where you're dealing with a set of names and scores, except each name is bounded to its score, and vice-versa, in a struct. So just plug in the struct syntax for the name where you had the array syntax for the name, in 1.2 user input of names.

*Edited by **James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)** on Sep 5 at 5:27pm*

Craig Nicholas (https://santarosajc.instructure.com/courses/24402/users/118709)

(https://santarosajc.instructure.com/courses/24402/users/118709)

Sep 5, 2017

Until I'm told it's fatally flawed I am fairly confident I have a solution for the palindrome problem that doesn't know the length of the input. In other words, finding the NULL character immediately resolves into determining the answer.

I did attempt 2.3 by minimally altering 1.2 (except for the sorting function). What appeared to be the laziest way to switch it over didn't work.

But on the basis of your reply under Andrew Langwell's comment I tried changing a cin.ignore('\n') to cin.ignore() ... and that did it. I'm sure I tried that at some point but it must have been offset by other things I was doing wrong at that time.

Thanks! Can I just resubmit this? I'll try resubmitting and see what happens...

David Harden (https://santarosajc.instructure.com/courses/24402/users/60154)

(https://santarosajc.instructure.com/courses/24402/users/60154)

Sep 6, 2017

Yes, you can resubmit up until the final deadline.

Keep in mind that "simple code" is 20% of your grade on the assignment. So if your solution to isPalindrome() is more complex than it needs to be, that could hurt you. It's hard for me to imagine a solution that doesn't start by determining the length of the string that isn't a lot more complex than necessary.

Craig Nicholas (https://santarosajc.instructure.com/courses/24402/users/118709)

(https://santarosajc.instructure.com/courses/24402/users/118709)

Sep 6, 2017

Ah I'll have to remember the simple code thing, that was not exactly a guiding principle of what I turned in this week.

*Edited by **Craig Nicholas** (https://santarosajc.instructure.com/courses/24402/users/118709) on Sep 6 at 11:09pm*

Reply

Alexis Ordonez (https://santarosajc.instructure.com/courses/24402/users/3920)

(https://santarosajc.instructure.com/courses/24402/users/3920)

Sep 5, 2017

Assignment 2.1 was very useful and very fun to figure out how to manipulate c strings in new ways. Honestly seems much more straight forward than what I remember from CS10 and strings in general. Assignemtn 2.3 was basically a copy/paste of last weeks assignment, just swapping how the arrays are called since it is now a struct. However, assignment 2.3 was nice just to see how a set of arrays are similar to a struct and how much more useful a struct is when you have multiple arrays. 2.2 was a struggle at first just before of the "allow the user to input multiple lines" and how to ignore the enter to stop the input stream. Then I figured using a while loop that only exits once the period is entered fixed this issue. After that, it was just a tolower and sort algorithm away.

(https://santarosajc.instructure.com/courses/24402/users/78465)
Sutton Laird
(https://santarosajc.instructure.com/courses/24402/users/78465)
Sep 5, 2017

Using structs for the high score program in 2.3 rather than the arrays from 1.2 definitely seems like a more efficient way to handle the data. I also got stuck on the issue of the enter in the input stream, and spent quite a while pondering that dilemma. I chose to use toupper instead of tolower though, but obviously it doesn't make a difference.

Reply

(https://santarosajc.instructure.com/courses/24402/users/56027)
Andrew Langwell
(https://santarosajc.instructure.com/courses/24402/users/56027)
Sep 5, 2017

I was able to get through assignments 2.1 and 2.2 after being stuck on some minor details. I'm currently stuck on assignment 2.3. I'm trying to use a loop with cin.get(ch) to read in the characters but for some reason the program doesn't wait for an input at the cin.get(ch) line of code. It goes on to the next cout statement to enter a score.

(https://santarosajc.instructure.com/courses/24402/users/56027)
Andrew Langwell
(https://santarosajc.instructure.com/courses/24402/users/56027)
Sep 5, 2017

I figured out my mistake. I was making the code more complicated than it needed to be.

(https://santarosajc.instructure.com/courses/24402/users/118709)
Craig Nicholas
(https://santarosajc.instructure.com/courses/24402/users/118709)
Sep 5, 2017

I was having your same problem on 2.3, though I already submitted. Are you able to briefly describe what was needed?

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 5, 2017

I'll bet it's a problem where both the >> operator and getline() and/or get() are used in the same program. cin >> myVariable; will leave the '\n' character in the input stream. Then getline() will see that '\n' and interpret it as end of input. I still have problems with this and should put together some good notes. Basically, though, if you put a cin.ignore() or cin.clear() between your >> and your getline*, that should clear it up. If it's the problem I think it is.

You can look up >>, get(), and getline() on cplusplus.com. What I usually do is experiment with cin.ignore() until things are working. (You can look up ignore() on cplusplus.com, too).

*that's between in a sequential sense. Remember that just clearing between those two statements in a given function doesn't mean that a getline() from some other part of the program won't follow an >> in the given function.

Edited by James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745) on Sep 5 at 5:37pm

Andrew Langwell (https://santarosajc.instructure.com/courses/24402/users/56027)

(https://santarosajc.instructure.com/courses/24402/users/56027)

Sep 6, 2017

Yes I think that was the problem too. Thanks for your help and providing me with the cin.ignore() fix. I found a solution that was much more simple without using get().

Reply

Kenneth Catanzano (https://santarosajc.instructure.com/courses/24402/users/61745)

(https://santarosajc.instructure.com/courses/24402/users/61745)

Sep 5, 2017

I struggled pretty hard with the reading from the text on this assignment, I feel like I'm just not grasping it the way I would in class.

Drew Rodrigues (https://santarosajc.instructure.com/courses/24402/users/59022)

(https://santarosajc.instructure.com/courses/24402/users/59022)

Sep 5, 2017

A good method of understanding is asking yourself questions while reading then making sure you answer them with clear understanding. For example, I use Evernote while I'm reading to list questions. Once I'm done with the chapter, if there are questions left unanswered, I make sure to answer them either through search or through messaging the instructor. Also, if you want, it may be a great idea to start a Slack group for the class, so we can help each other if something isn't understood well. Hang in there man!

Reply

Sam Ho (https://santarosajc.instructure.com/courses/24402/users/57399)

(https://santarosajc.instructure.com/courses/24402/users/57399)

Sep 5, 2017

```
Letter:    Number of Occurrences
     o     3
     b     2  <---- this one different from Dave's output anyone have the same answer??
     d     2
     e     1
```

James O'Hara (https://santarosajc.instructure.com/courses/24402/users/74745)

(https://santarosajc.instructure.com/courses/24402/users/74745)

Sep 6, 2017

It's also a good answer for the sample input. There are two b's in the input, and the output is sorted by number of occurrences. Maybe your sort algorithm gives b before d, while Dave's gives d before b. Both are correct.

```
Enter a sequence of characters (end with '.'): do be Do bo. xyz

Letter:    Number of Occurrences
     o     3
     d     2
     b     2
     e     1
```

Reply

Nathan Cortez (https://santarosajc.instructure.com/courses/24402/users/63243)

(https://santarosajc.instructure.com/courses/24402/users/63243)

Sep 5, 2017

So my reply apparently didn't send yesterday...

Anyways! Something that I struggled with was actually setting up the input for 2.2. It was pretty challenging as I was confused on how to even begin and it took me some testing to finally find a solution. I understood the concept of reading the characters one by one, but to implement it was the more difficult part.

On assignment 2.3 I had problems trying to limit the amount of characters that you can type into for the names. The furthest I got from actually succeeding was using a for loop to count how many times I can input a word and if it reached the maximum possible words, it terminates the c-string.

Reply

(https://santarosajc.instructure.com/courses/24402/users/111342)

Matthew Young

(https://santarosajc.instructure.com/courses/24402/users/111342)

Sep 5, 2017

For such a core fundamental of programming, pointers were always confusing to me. This assignment got me more comfortable using pointers but I'll need to keep practicing to really solidify the topic in my head. Structs are very useful for managing various types of data and I can imagine the possibilities of how useful they will be in future assignments.

Reply

(https://santarosajc.instructure.com/courses/24402/users/79900)

Peter Wright

(https://santarosajc.instructure.com/courses/24402/users/79900)

Sep 6, 2017

I was able to slowly finish each assignment. 2.1 took the most time. I had some difficulty understanding c-string functions and how they worked.

Edited by Peter Wright (https://santarosajc.instructure.com/courses/24402/users/79900) on Sep 6 at 10:27pm

Reply

(https://santarosajc.instructure.com/courses/24402/users/66384)

Julia Otten

(https://santarosajc.instructure.com/courses/24402/users/66384)

Sep 6, 2017

I found this assignment to be quite a bit more challenging than the previous one and 2.2 took me a while to work on. I will definitely start earlier in the week going forward to give myself more time to work on the assignments.

Reply

(https://santarosajc.instructure.com/courses/24402/users/89029)

Dany Rodriguez

(https://santarosajc.instructure.com/courses/24402/users/89029)

Sep 6, 2017

It was hard to understand this concept without the help of a lesson and examples like the first assignment or in class help. I'm going to have to turn the assignment in late because I don't want to turn in an unfinished program.

Reply

(https://santarosajc.instructure.com/courses/24402/users/50114)
Anthony Cindi
(https://santarosajc.instructure.com/courses/24402/users/50114)
Sep 6, 2017

What is the difference making copies of an character array and an array of structures?

(https://santarosajc.instructure.com/courses/24402/users/74745)
James O'Hara
(https://santarosajc.instructure.com/courses/24402/users/74745)
Sep 7, 2017

The character array is of type char, and is composed of elements of that type.

An array of structs would be an array of the type of the struct--LetterFrequency or HighScore (and of course the structs would composed of its fields, which would have their own types. In 2.2, the struct has a char and an int. In 2.3, the struct has an int and an array of char).

The way to make copies is basically the same (but shouldn't be necessary in 2.2 or 2.3). Allocate an array of the same type and size as the original, and then copy the data from the original to the copy. (The array of struct copy can be allocated with one statement, just as the original can. The parts of the struct don't have to be allocated first). If the size is known at compile time, the copy can be a stack array. If the size is not known at compile time, the copy must be dynamically allocated (use "new").

If the array of char is a c-string, the strcpy() utility can be used to copy the c-string data. The other data would have to be copied element-by-element in a loop.

Does that answer the question?

Reply

(https://santarosajc.instructure.com/courses/24402/users/84805)
Ulises Ramirez Hernandez
(https://santarosajc.instructure.com/courses/24402/users/84805)
Sep 6, 2017