Converting a C-String Array to a C++ String Array & Vice Versa
Laboratory 10B

CS-102

Spring 2022

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- Using the same data that was in Lab10A, we will convert its type.
  - "Joe Looney, 555-0097",
  - "Geri Palmer, 555-8787",
  - "Li Chen, 555-1212",
  - "Holly Gaddis, 555-8878",
  - "Sam Wiggins, 555-0998",
  - "Alejandra Cruz, 555-1223“,
  - "Bob Kain, 555-8712",
  - "Tim Haynes, 555-7676",
  - "Warren Gaddis, 555-9037",
  - "Jean James, 555-4939",
  - "Ron Palmer, 555-2783"

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- Whether we use C-Strings to do our String work in C++, or use C++ Strings to do this work, is often just a matter of taste.
- However, your choice of which to use may also be determined by what you want to do with arrays of string.
- There is a wide assortment of functions and methods that exist in the C-String libraries as well as in the C++ String libraries. Some functions in a particular library may be more appropriate to solving a particular problem than the corresponding functions in the other library.
- In this lab, we are going to convert a C-String Array of Strings into a C++ String Array of strings, as well as doing the reverse.
- This will enable us to use the wide assortment of "Member Functions" (a.k.a. Methods) that are available in the C++ String library on an array that may have been given to us as a C-String array. (See pages 598 & 599 in Gaddis, Starting Out With C++, Edition 9).

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- In this lab you will be asked to take the data given below as a C-String Array of strings and convert it into a C++ String Array of strings, and then print it out to the screen.

```
char charArray[SIZE][25] = { "Joe Looney, 555-0097",
                             "Geri Palmer, 555-8787",
                             "Li Chen, 555-1212",
                             "Holly Gaddis, 555-8878",
                             "Sam Wiggins, 555-0998",
                             “Alejandra Cruz, 555-1223",
                             "Bob Kain, 555-8712",
                             "Tim Haynes, 555-7676",
                             "Warren Gaddis, 555-9037",
                             "Jean James, 555-4939",
                             "Ron Palmer, 555-2783"};
```

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- The first thing that you will notice about the C-String Array of Strings is that it is a 2-dimensional array, where the first dimension has to do with the row that each string occupies, while the second dimension has to do with where each letter falls, horizontally, as we parse through the string in a particular row.
- We are going to convert this two dimensional array of type, **char [][]** , into a one dimensional array of C++ strings of type, **string[]**.
- If you were to look at a C-String stored in memory and compare that with a C++ String stored in memory, at first glance the two strings would appear identical to your eye.
- They both contain consecutive bytes each holding a string character, and they are terminated in a “\0” or Null character (an ASCII 00).
- The main difference is that the C++ String does not care how long the string is as long as the string is terminated in the Null, whereas the C-String does care.

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- The following is an algorithm that will tell you how to proceed to transform the C-String Array of Strings into a C++ String Array of strings.

1. Start by declaring your data as an Array of C-String as shown two pages back.
2. Next declare an empty Array of C++ String:
   - string stringArray[SIZE];
   - Note that the SIZE refers to the number of strings in the array, which would be the same as for the C-String Array.
3. Now make a loop that will convert every row of the C-String Array of char into a String for that row. [See row 3 of Table 10-7].
4. Next, display the resulting C++ Array of String.

Laboratory 10B – Part 1 : Converting C-String Arrays to C++ String Arrays

- Your program should display the original C-String Array of strings as they are given to you.
- Then your program should convert the array into a C++ String Array of strings.
- Finally, it should display the C++ String Array.
- The two arrays should look identical.
- Give your program the name: ***YourName*-Lab10B-1.cpp**
- Once you have your program working:
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 10B – Part2: Converting C++String Arrays to C-String Arrays

- Now we will do the reverse procedure.
- In Part 2 you will be asked to take the data given below as a C++String Array of strings and convert it into a C-String Array of strings, and then print it out to the screen.

```
string stringArray[SIZE]     =    { "Joe Looney, 555-0097",
                                    "Geri Palmer, 555-8787",
                                    "Li Chen, 555-1212",
                                    "Holly Gaddis, 555-8878",
                                    "Sam Wiggins, 555-0998",
                                    “Alejandra Cruz, 555-1223",
                                    "Bob Kain, 555-8712",
                                    "Tim Haynes, 555-7676",
                                    "Warren Gaddis, 555-9037",
                                    "Jean James, 555-4939",
                                    "Ron Palmer, 555-2783"};
```

Laboratory 10B – Part2: Converting C++String Arrays to C-String Arrays

- The first thing that you will notice about the C++String Array of Strings is that it is a 1-dimensional array, where the single dimension has to do with the row that each string occupies
- We are going to convert this one dimensional array of type, **string[]**, into a two dimensional array of C-strings of type, **char [][]** .
- The following is an algorithm that will tell you how to proceed to transform the C++String Array of Strings into a C-String Array of strings.

1. Start by declaring your data as an Array of C++String as shown on the previous page.
2. We need to start by finding the largest string of data in the array.
   - We can use the **sizeof()** function to determine the size of each string.
   - Loop through all the strings in the array of strings, finding **maxsize**, the largest string in the array.

Laboratory 10B – Part2: Converting C++String Arrays to C-String Arrays

3. Now you may declare your C-String array: char charArray[][]
   a. Question 1: What should the first array element be set to?
   b. Question 2: What should the second array element be set to?
4. Now load up the array of Char from the array of string.
   a. Note: Both Arrays of Char and Arrays of string have the property that if only one subscript is used, the entire string (or line) is what is being referenced.
   b. If two subscripts are used then the individual characters in each string (or line) are individually referenced.
   c. You will want to access each individual character in the entire array of Char and make it equal to each individual character in the entire array of String.
   d. Remember that the width of the Array has to include the '\0' character, so it has to be one larger that the number of letters you have allowed for the width.

Laboratory 10B – Part2: Converting C++String Arrays to C-String Arrays

6. Now print out the contents of the new two dimensional array of char.

7. Also print out the Minimum width, **maxsize + 1**, that your two dimensional array of char must be set to.

8. Call your program: *YourName*-Lab10B-2.cpp

9. When you have this running:
   - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
   - If you are doing this Lab asynchronously, submit your program to Canvas.

Table 10-8

Member Function Example	Description
`mystring.begin();`	Returns an iterator pointing to the first character in the `string`. (For more information on iterators, see Chapter 16.)
`mystring.c_str();`	Converts the contents of `mystring` to a C-string, and returns a pointer to the C-string.
`mystring.capacity();`	Returns the size of the storage allocated for the `string`.
`mystring.clear();`	Clears the `string` by deleting all the characters stored in it.
`mystring.compare(str);`	Performs a comparison like the `strcmp` function (see Chapter 4), with the same return values. `str` can be a `string` object or a character array.
`mystring.compare(x, n, str);`	Compares `mystring` and `str`, starting at position `x`, and continuing for `n` characters. The return value is like `strcmp`. `str` can be a `string` object or character array.
`mystring.copy(str, x, n);`	Copies the character array `str` to `mystring`, beginning at position `x`, for `n` characters. If `mystring` is too small, the function will copy as many characters as possible.
`mystring.empty();`	Returns true if `mystring` is empty.
`mystring.end();`	Returns an iterator pointing to the last character of the string in `mystring`. (For more information on iterators, see Chapter 16.)
`mystring.erase(x, n);`	Erases `n` characters from `mystring`, beginning at position `x`.
`mystring.find(str, x);`	Returns the first position at or beyond position `x` where the string `str` is found in `mystring`. `str` may be either a `string` object or a character array.
`mystring.find('z', x);`	Returns the first position at or beyond position `x` where `'z'` is found in `mystring`.
`mystring.front();`	Returns the first character in the `string`. (This member function was introduced in C++ 11.)
`mystring.insert(x, n, 'z');`	Inserts `'z'` `n` times into `mystring` at position `x`.
`mystring.insert(x, str);`	Inserts a copy of `str` into `mystring`, beginning at position `x`. `str` may be either a `string` object or a character array.
`mystring.length();`	Returns the length of the string in `mystring`.
`mystring.replace(x, n, str);`	Replaces the `n` characters in `mystring` beginning at position `x` with the characters in string object `str`.
`mystring.resize(n, 'z');`	Changes the size of the allocation in `mystring` to `n`. If `n` is less than the current size of the string, the string is truncated to `n` characters. If `n` is greater, the string is expanded and `'z'` is appended at the end enough times to fill the new spaces.
`mystring.size();`	Returns the length of the string in `mystring`.
`mystring.substr(x, n);`	Returns a copy of a substring. The substring is `n` characters long and begins at position `x` of `mystring`.
`mystring.swap(str);`	Swaps the contents of `mystring` with `str`.

Appendix A: Methods belonging to the String Class Of particular interest:

mystring.insert(x, str);

- Inserts a copy of **str** into **mystring**, beginning at position x.
- **str** may be either a string object or a character array.

mystring.clear()

- Clears the string by deleting all the characters stored in it.

Appendix B: Character Testing Functions

- Requires `cctype` or `string` header file

FUNCTION	MEANING
`isalpha(arg)`	`true` if arg. is a letter, `false` otherwise
`isalnum(arg)`	`true` if arg. is a letter or digit, `false` otherwise
`isdigit(arg)`	`true` if arg. is a digit 0-9, `false` otherwise
`islower(arg)`	`true` if arg. is lowercase letter, `false` otherwise
`isprint(arg)`	`true` if arg. is a printable character, `false` otherwise
`ispunct(arg)`	`true` if arg. is a punctuation character, `false` otherwise
`isupper(arg)`	`true` if arg. is an uppercase letter, `false` otherwise
`isspace(arg)`	`true` if arg. is a whitespace character, `false` otherwise

Appendix C:
Useful String Functions

Table 10-7 Examples of `string` Object Definitions

Definition	Description
`string address;`	Defines an empty `string` object named `address`.
`string name("William Smith");`	Defines a `string` object named `name`, initialized with "William Smith."
`string person1(person2);`	Defines a `string` object named `person1`, which is a copy of `person2`. `person2` may be either a `string` object or character array.
`string str1(str2, 5);`	Defines a `string` object named `str1`, which is initialized to the first five characters in the character array `str2`.
`string lineFull('z', 10);`	Defines a `string` object named `lineFull` initialized with 10 `'z'` characters.
`string firstName(fullName, 0, 7);`	Defines a `string` object named `firstName`, initialized with a substring of the `string fullName`. The substring is seven characters long, beginning at position 0.