

CSE 332 Studio Session on C++ Pointers, References, and Arrays

These studio exercises are intended to introduce basic features of C++ pointers, references, and arrays, and to give you experience using those concepts and techniques within the Visual C++ environment.

In this studio you will again work in self-selected small groups of 2 or 3 people. As before, students who are more familiar with the material are encouraged to help those for whom it is less familiar. Asking questions of your professor and teaching assistants (as well as of each other) during the studio sessions is highly encouraged as well. Please record your answers as you work through the following exercises. After you have finished please post your answers to the required exercises, and to any of the enrichment exercises you completed, to the course message board as a reply to my posting titled "Pointers, References, and Arrays Studio". The enrichment exercises are optional but are a good way to dig into the material a little deeper, especially if you breeze through the required ones.

PART I: REQUIRED EXERCISES

1. Find/choose your team members in the studio area, log in to one of the Windows machines, open up Visual Studio, and create a new Visual C++ Win32 Console Application project for this studio. Change the signature of the main function in the source file that Visual C++ generated to match the one that was used in the lecture slides on C++ program structure, which are available on the course web page. Write down the names of the team members who are present (if a team member arrives late please catch them up on the work, and add their name) as the answer to this exercise.
2. In your main function, declare a pointer to const character (of type `const char *`) as the loop variable of a `for` loop, and using only pointer operators (i.e., without using the square bracket array indexing operator `[]`) do the following: (1) initialize the pointer to point to the first character of the program name that was passed into the main program (**hint**: the program name is always in `argv[0]`); (2) at each new position test whether the pointer still points to a non-zero character (anything other than `'\0'`) and if so print out the character at that position and then an end of line; and (3) after each step move the pointer to the next character position in the program name. As the answer to this exercise give what your program printed.
3. What happens if you try to repeat exercise 3 using a reference instead of a pointer? Please give three ways in which references differ from pointers in C++, and say which one of those three explains what happens when you try to use a reference that way in C++, as the answer to this exercise.
4. Again without using array operators, modify your solution to exercise 3 so that instead of only iterating through the characters in the program name, it iterates through all of the characters of all of the strings that were passed as arguments to the program. As the answer to this exercise, please identify all of the pointer operators (not array operators) that you used, and describe briefly what each one does.

5. Declare a (3-dimensional) 2 by 3 by 5 array of unsigned integers and initialize all of its elements to zero when you declare it. Iterate through the cells in the array, and set the value of the cell to be the product of its coordinates (the cell at position 0,0,0 would have value 0, the cell at position 1,2,4 would have value 8, etc.) and print out the value that you stored in each cell as the answer to this exercise.

6. Write a function that takes two pointers to unsigned integer by value and returns a pointer positioned half way between them (**hint:** although adding two pointers is not defined in C++, subtracting a pointer from another pointer and adding that integer result to a pointer is defined in C++). Declare a one-dimensional array of unsigned integers in which each cell's value is initialized to be the same as its position index. Pass in the addresses of different pairs of positions in the array to the function, and print out what addresses you passed in and what address was returned by the function each time. As the answer to this exercise, explain how you can tell easily (from the passed and returned pointer addresses): (1) when the addresses passed in are the same, vs. (2) when the addresses are for adjacent cells in the array, vs. (3) when there is at least one cell between the cells whose addresses were passed in to the function.

PART II: ENRICHMENT EXERCISES (optional, do the ones that interest you).

7. Modify your solution to exercise 5 so that in addition to printing out the contents of the cell, it also prints out (1) the memory address of the cell, and (2) the coordinates of the cell in the array (e.g., 0,0,0 vs. 1,2,4 etc.). Please describe how the array is laid out in memory relative to the position indexes of the cells (for example as a formula involving coordinates i , j , and k), as the answer to this exercise.

8. Write a function with a void return type, which takes two pointers to unsigned integer by reference and positions both pointers at the position in the range between them (including the positions to which they point) in which the smallest value is stored (**hint:** to do this you can use one of the pointers to “remember” the position with the smallest value seen so far, and move the other pointer towards it until it finds a smaller value).

Declare a one dimensional array of unsigned integers and as you do this exercise modify its initial cell values to hold different combinations of unsigned integers. Declare two pointers to unsigned integers, and as you do this exercise, try initializing them with the addresses of different pairs of cells in the array and passing them into the function.

Each time you do this, please print out the addresses of the pointers before the function call and what values were in the positions in between them (including the positions whose addresses they stored), and the value at the position where both of them point after the function call, as the answer to this exercise.