CSC 150

Pre Lab #9: Arrays

**Purpose**:

1. Initializing and accessing array contents
2. Inputting, processing, and outputting array contents
3. Demonstration of accessing past array bounds
4. Passing arrays as arguments to functions
5. Complete the section exercises in chapter 10 of the course’s textbook.

Section 10.2 exercises (pages 277-278)

Section 10.3 exercises (pages 281-282)

Section 10.4 exercises (page 284)

Section 10.6 (pages 291-292) (function declaration == prototype)

For the following problems, first attempt to determine their output by analysis of the code. You can then copy the code and paste it into a project to check your answer.

1. What does the following program print?

#include <iostream>

using namespace std;

int main()

{

int i;

int my\_array[10]; // change this line in problem 3

for (i = 0; i< 10; i++)

my\_array[i] = i;

for (i = 0; i < 10; i++)

cout << my\_array[i] << " ";

return 0;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. What happens if you change int my\_array[10]; to int my\_array;



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



What does the following program print?

#include <iostream>

using namespace std;

int main()

{

int i;

// change the next line in #5, #6

int my\_array[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

for (i = 0; i < 10; i++)

cout << my\_array[i] << " "; // change this line in #7

return 0;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. Change the declaration to

int my\_array[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

What happens?



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. Change the declaration to

int my\_array[10] = { 0, 1, 2, 3, 4 };

What happens?



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. Change the cout << my\_array[i]; to cout << my\_array;   
   What happens?



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



What does the following program print?

#include <iostream>

using namespace std;

int main()

{

int i;

int k = 55;

int my\_array[10];

for (i = 0; i < 10; i++)

my\_array[i] = i;

for (i = 0; i <= 12; i++)

cout << my\_array[i] << " ";

cout << endl;

cout << " k = " << k << endl;

return 0;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. What does the following program print? Where do the last two values come from?

#include <iostream>

using namespace std;

int main()

{

int i;

int my\_array[10];

int your\_array[10];

for (i = 0; i < 10; i++)

your\_array[i] = i \* 20;

for (i = 0; i < 10; i++)

my\_array[i] = i;

for (i = 0; i <= 13; i++)

cout << your\_array[i] << " ";

return 0;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



What happens when you compile the following program? How do you fix the problem so that both arrays contain the same content?



#include <iostream>

using namespace std;

int main()

{

int i;

int my\_array[10];

int your\_array[10];

for (i = 0; i < 10; i++)

my\_array[i] = i;

your\_array = my\_array ;



for (i = 0; i < 10; i++)

cout << my\_array[i] << " ";

cout << endl;

for (i = 0; i <= 10; i++)

cout << your\_array[i] << " ";

return 0;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. What does the following print?

#include <iostream>

using namespace std;

void add\_one(int temp\_array[5]);

int main()

{

int i;

int my\_array[5] = { 1, 2, 3, 4, 5 };

add\_one(my\_array);

for (i = 0; i < 5; i++)

cout << my\_array[i] << " ";

return 0;

}

void add\_one(int temp\_array[ ])

{

int j;

for (j = 0; j< 5; j++)

temp\_array[j]++;

}

answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Is the array passed by reference or by value?

answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



In the function header and the function prototype, place an & in front of temp\_array[5] What happens?



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



1. The following program passes elements into the function. What does the following print?

#include <iostream>

using namespace std;

void add\_one(int temp\_element);

int main()

{

int i;

int my\_array[5] = { 1, 2, 3, 4, 5 };

add\_one(my\_array[1]);

for (i = 0; i < 5; i++)

cout << my\_array[i] << " ";

return 0;

}

void add\_one(int temp\_element)

{

int j;

for (j = 0; j < 5; j++)

temp\_element++;

}



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Is this pass by reference or pass by value?

answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



What happens if you place an & in front of temp\_element in the prototype and the function header?



answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Is this pass by reference or pass by value?

answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



What does the program print if you leave off the subscript in the function call?

answer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Write a function which fills an array, returning the number of elements entered, and a function that returns the **index** of the largest element in an array. The parameters for the filling function are the array and the number of elements in the array, for the largest index function are the array and the number of valid elements. **Do Not Modify** the **main( )** function except for filling in the parameters to the function calls.

#include <iostream>

using namespace std;

int fill\_array(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

int find\_largest( );

int main()

{

int j;

int index;

int my\_array[10];

int num\_elements = 0;

num\_elements = fill\_array(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

index = find\_largest( );

cout << "The largest is: " << my\_array[index] << endl;

return 0;

}

//fills array till full or user enters ctrl-z, whichever comes

//first

int fill\_array(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{

}

//see next page for find\_largest( )

int find\_largest( )

{

} // end of function find\_largest