

Assignment 4.3	
Pointers	
Course Code: CPE007	Program: Computer Engineering
Course Title: Programming Logic & Design	Date Performed: 9/23/2025
Section: CPE11S1	Date Submitted: 9/23/2025
Name(s) Ralph Angelov F. Braganza	Instructor: Engr. Jimlord M. Quejado
Output	
Answer the following questions briefly: (OUTPUT SECTION)	
1. What is a pointer in C++?	<ul style="list-style-type: none"> - it is a variable that stores the memory address of another variable instead of storing a direct value. <p>Example, int x = 21; and the value of that is 21 this is a normal variable while a pointer doesn't store the value itself, what it essentially does is store the address of where the value is in memory (these addresses can vary on different computers and the reason to this is how memory is managed by the operating system).</p>
	<p>You use "*" to declare a pointer and "&" to get an address.</p> <pre>int x = 21; (normal variable) int *ptr = &x; (pointer stores the address of x)</pre>
2. How does a pointer differ from a regular variable?	<ul style="list-style-type: none"> - The difference between the two is that a regular variable stores a value directly meaning if x = 21 then the output of that will be 10. A pointer on the other hand stores the memory address of that variable and not the actual value like for example int *ptr = &x will print the address of x which could be anything but in my case it is 0x6ffe44.
3. What operator is used to get the address of a variable?	<ul style="list-style-type: none"> - The operator used to get the address of a variable is the ampersand (&) operator. <pre>int x = 21; std::cout << &x; (prints the memory address of x and if you put x as is it will print 21)</pre>
4. What operator is used to access the value stored at a pointer's address?	<ul style="list-style-type: none"> - The operator we have to use to be able to access the value stored at a pointer's address is the asterisk (*) operator, also called the dereference operator. <p>What I've provided before from question 1</p> <pre>int x = 21; (normal variable) int *ptr = &x; (pointer stores the address of x) std::cout << *ptr; (prints 21, the value stored at that address)</pre>
5. Why are pointers important in C++? Give two uses.	<ul style="list-style-type: none"> - One of the important uses of pointers in C++ is when we are working with arrays, a pointer can be used to navigate or manipulate elements in an array efficiently. Another use that makes pointers important is hardware interaction, it allows direct access to the memory address, which is necessary for systems programming and device drivers. Like for example a keyboard, it doesn't send characters out of nowhere it is connected to the computer's hardware, and the operating system that stores the pressed keys in a specific memory location (as I mentioned before that some computer's have different memory addresses because of the operating system). This is where pointers come into play, that key press can be directed to that memory location so it can read what key was pressed.

REFERENCE: GeeksforGeeks. (2025, July 25). C++ Pointers. GeeksforGeeks. <https://www.geeksforgeeks.org/cpp/cpp-pointers/>

Supplementary Activity

Identify the Output

For each code snippet, predict the output without compiling:

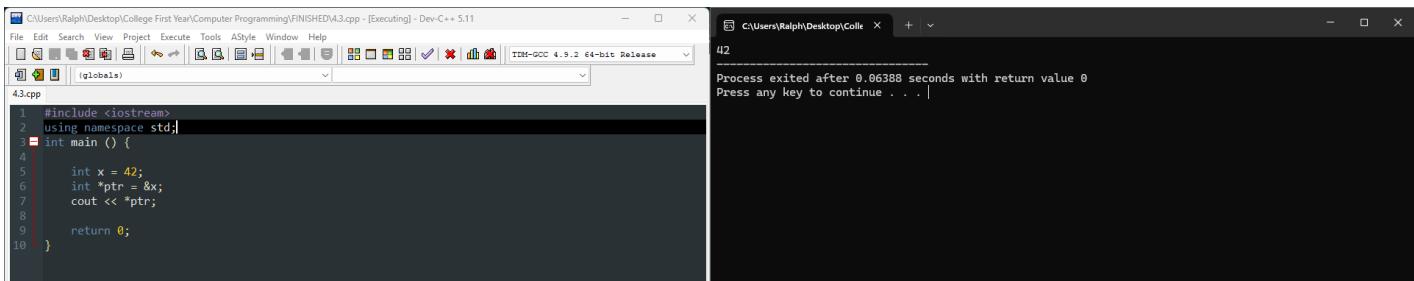
1.

```
int x = 42;
int *ptr = &x;
cout << *ptr;
```

PREDICTION:

It will print the value stored at that address which is 42.

RESULT:



```
43.cpp
1 #include <iostream>
2 using namespace std;
3 int main () {
4
5     int x = 42;
6     int *ptr = &x;
7     cout << *ptr;
8
9     return 0;
10 }
```

```
42
Process exited after 0.06388 seconds with return value 0
Press any key to continue . . .
```

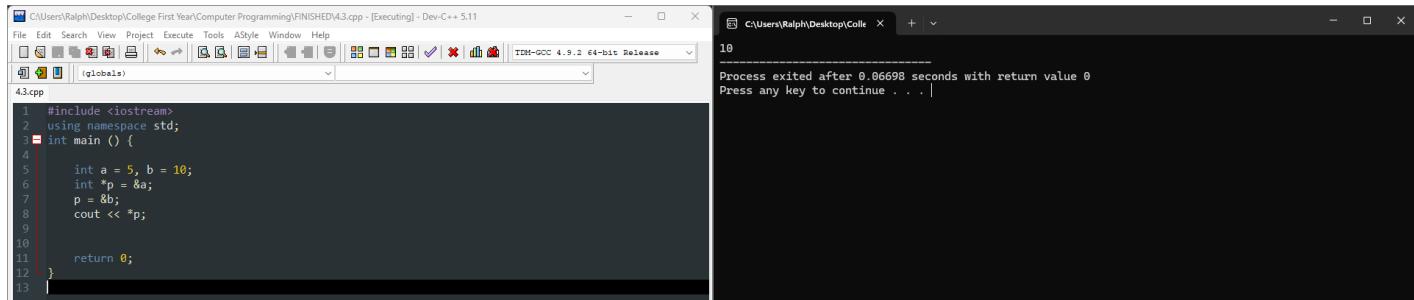
2.

```
int a = 5, b = 10;
int *p = &a;
p = &b;
cout << *p;
```

PREDICTION:

It will print the value stored at that address “a” which is 5.

RESULT:



```
43.cpp
1 #include <iostream>
2 using namespace std;
3 int main () {
4
5     int a = 5, b = 10;
6     int *p = &a;
7     p = &b;
8     cout << *p;
9
10    return 0;
11 }
12 }
```

```
10
Process exited after 0.06698 seconds with return value 0
Press any key to continue . . .
```

3.

```
int arr[3] = {10, 20, 30};  
int *p = arr;  
cout << *p;
```

PREDICTION:

It will print the value stored at the first element which is 10.

RESULT:

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays the following C++ code:

```
#include <iostream>  
using namespace std;  
int main () {  
    int arr[3] = {10, 20, 30};  
    int *p = arr;  
    cout << *p;  
    return 0;  
}
```

On the right, the terminal window shows the output of the program:

```
10  
Process exited after 0.06698 seconds with return value 0  
Press any key to continue . . .
```

4.

```
int arr[4] = {2, 4, 6, 8};  
int *p = arr;  
p++;  
cout << *p;
```

PREDICTION:

It will print the stored value of the second element which is 4.

RESULT:

The screenshot shows the Dev-C++ IDE interface. On the left, the code editor displays the following C++ code:

```
#include <iostream>  
using namespace std;  
int main () {  
    int arr[4] = {2, 4, 6, 8};  
    int *p = arr;  
    p++;  
    cout << *p;  
    return 0;  
}
```

On the right, the terminal window shows the output of the program:

```
4  
Process exited after 0.06149 seconds with return value 0  
Press any key to continue . . .
```

5.

```
int arr[3] = {5, 15, 25};  
int *p = arr;  
cout << *(p + 2);
```

PREDICTION:

It will print the stored value of the second element which is 15.

RESULT:

The screenshot shows two windows from the Dev-C++ IDE. The left window displays the source code for file 4.3.cpp, which contains the following C++ code:

```
1 #include <iostream>  
2 using namespace std;  
3 int main () {  
4     int arr[3] = {5, 15, 25};  
5     int *p = arr;  
6     cout << *(p + 2);  
7     return 0;  
8 }
```

The right window shows the terminal output of the program execution. It displays the number 25, indicating the value at memory location p + 2, and ends with the message "Press any key to continue . . .".

Error Spotting

Identify and fix the error (if any) in the codes below.

1.

```
int arr[3] = {1, 2, 3};  
int *p = &arr; // &arr means it wants the address of the whole array, but p is declared as a single int* so this will throw an  
error
```

FIXED VERSION:

```
int arr[3] = {1, 2, 3};  
int *p = &arr[0]; // now it will print the first element of the array (that is if you are printing *p).
```

2.

```
int arr[5];  
int *p; // p is a pointer so its expecting a memory address not an integer value.  
p = arr[2]; // arr[2] gives the value stored in the 3rd element of the array which is an int
```

This will give an error because it is trying to assign an int to a pointer.

FIXED VERSION:

```
int arr[5];  
int *p;  
p = &arr[2] // I just added the & operator to get the address of the 3rd element of the array.
```

3.

```
int arr[4] = {10, 20, 30, 40};  
cout << *arr[2]; // arr[2] is already an integer and using the "*" operator is asking it to dereference it even though it isn't an  
address and this is why it will cause an error.
```

FIXED VERSION:

```
int arr[4] = {10, 20, 30, 40};  
cout << *(arr + 2); // so what I learned from number 5 with the snippets (because I got it wrong) we will write it like this
```

This will use a pointer arithmetic where arr is the array and + 2 this moves 2 positions forward so this points to the 3rd element. The “*” essentially tells it to go the address of that pointer and get its value which is 30.

Conclusion

This assignment has given me the opportunity to further expand my knowledge about pointers and the importance of them and why we need them. I didn't know that pointers were used to direct keyboard or inputs from our hardware to the software by directing the addresses of the keypresses to identify what key was pressed or inputted. I've also learned the basics of how to use pointers with arrays, but I still don't know how to use pointers to get the address of the specific value in an array element. I get the gist of how the procedure works for the supplementary but not completely, but I did learn or somewhat understand how number 5 works and implemented that to fix the error on number 3 of the error-spotting supplementary activity. In my opinion I didn't really do that well; I was really confused on some parts, like the int and int*, which confused me for quite a bit, but I managed to pull through. I guess the things I really have to improve when it comes to this type of thing are just more practice like these activities and trying to understand it using different ways of using pointers. I'm getting confused on some parts, but as long as I search and try to understand to the best of my abilities, I'm able to do the task as long as I have time, and that is another thing I want to improve, which is my efficiency, because I want to be able to answer well in quizzes and onsite tasks. Other than that, this was a helpful activity to get the basics of what pointers are used for and how I can apply them to coding.