Activity No. 4.3	
Pointers	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Computer Programming Language and Design	**Date Performed: September 22, 2025**
Section: CpE11S1	**Date Submitted: September 23, 2025**
Name(s): Lopez, Andrei Dion C.	**Instructor: Engr. Jimlord M. Quejado**

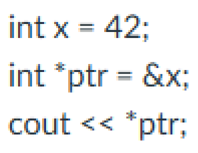
6. Output

- What is a pointer in C++?
  - It is a variable that stores the address of another variable and uses that address as a reference point to tell the compiler which value to output using the addresses of the variables.
- How does a pointer differ from a regular variable?
  - A regular variable stores the value that is initialized for it, while a pointer stores the location of a variable.
- What operator is used to get the address of a variable?
  - The operator used to get the address of a variable is "**&**", also known as the ampersand.
- What operator is used to access the value stored at a pointer's address?
  - The operator used the access the value stored at a pointer's address is "*****", known as the asterisk symbol.
- Why are pointers important in C++? Give two uses.
  - It allows for a more efficient way of handling memory as it allows for better memory management, because it uses the address of a variable, it allows it to no longer need to copy large amounts of data just to show the same value. It is also a great way to manage complex data structures as it is, as stated earlier, efficient at manipulating memory which allows for better performance.

7. Supplementary Activity

Code Prediction:

- Code 1:



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

  - The output of this code is going to be 42. The reason for this is because of the second line of code where it initializes the pointer "***ptr**" to the value that is stored at the address of "**&x**". Or in other terms it tells the compiler to go to the address of x and assign the address to ptr, then output the dereferenced value of ptr which is 42.

- Code 2:

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

  - The output here will be 10, The reason for this is because of the third line overwriting the value of p with the address of the value b. and then it would output the dereferenced value of p which is 10.

- Code 3:

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

  - In this code the output will be 10. This is because the default value of an array when the index number is not specified will always be the first number. So when initializing p it sets its value to the first element of the array which is 10.

- Code 4:

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

  - The code here will output 4. I think the reason for this is because since pointers are the addresses of a value, if we increment the address by 1, it changes the location address to the next value that is located at the next address which is 4.

- Code 5:

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

  - The output of this code will be 25, This is because at the third line, we add 2 to the address of the values inside the array which would move the value from the default value of 5, to the value 2 addresses higher than 5 which would be 25.

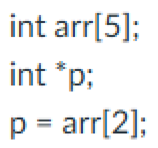
Error Spotting:

- Code 1:

```
int arr[3] = {1, 2, 3};
int *p = &arr;
```

  - The error in this code is located at the second line as the reference operator “**&**” can not get the addresses of all the values inside the array and will ask for a specific index. To fix this we can add a bracket “**[ ]**”, then add the index of any of the 3 elements stored inside the array.

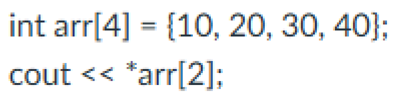
- Code 2:



```
int arr[5];
int *p;
p = arr[2];
```

  - The error in this code is at the third line. This is because we can not set a normal value as the value of a pointer. The way that we can fix this is by changing the operation “**p = arr[2]**” to the operation “**p = &arr[2]**”, This will set the value of the pointer **p** to the address of the third index of the array.

- Code 3:



```
int arr[4] = {10, 20, 30, 40};
cout << *arr[2];
```

  - The error is located at the second line “***arr[2]**”. This is because the variable “**arr**” is a regular integer and was not initialized as a pointer value. The fix for this is by initializing another integer as a pointer like “**int *ptr = &arr[2]**” which will give us the the address of the element 30 and store it into our pointer value, then changing the output statement “**cout << *arr[2]**” to ‘**cout << *ptr**” to output the value located at the address that is stored at the variable ptr.

8. Conclusion

In this assignment I have understood more of pointers and when to use them, I have also understood how to predict the output of a program better, a great example of this is code number 4, where the value of p changed because of the increment of the address value by +1 which would change the address from index 0 to the address of index 1. I have also gained more knowledge in analyzing codes for errors, like in the second example where there was an error in the operation that set the value of a pointer to the integer value of an element inside an array.