

Hands-on Activity 5.2	
Structures	
Course Code: CPE007	Program: Computer Engineering
Course Title: Programming Logic and Design	Date Performed: September 30, 2025
Section: CPE11S1	Date Submitted: October 4, 2025
Name(s): Mendoza, Nathaniel B.	Instructor: Engr. Jimlord M. Quejado
<b>6. Output</b>	
<b>1. CODE ANALYSIS</b>	
<p>The program started with 2 header files, which is the <code>#include &lt;iostream&gt;</code> in which it allows input and output operations for it to be able to utilize <code>cout</code>, and <code>#include string</code> to hold names, digits and other special characters. Then it implied a structure with its structure name as "Card" which includes a data type of string with a variable name of face and suit. Then inside the <code>int main</code> function where the program starts running which it include "Card a;" as its structure variable then another structure where it used a pointer called "Card*aptr;" so that it will store its memory address to another variable. Then it assigned values such as Ace for the face and Spades for the suit. After that it uses an Address Operator so that the stored memory address (aPtr) will equal to "&amp;a". Next it first printed using a dot operator(.) a.face (Where it is equal to "Ace") then implied "of", and then a.suit (Where it is equal to "Spades") then ended its line therefore the first output should be "Ace of Spades". The second print is where it uses the arrow operator (-&gt;) which is basically the same as the first one only replacing the "." dot operator into an arrow operator where the result is also "Ace of Spades" because the arrow operator is also used as the dot operator in which both are used to access a member of a class. Lastly, for the third one it used dereference operator in combine with dot operator, so the printing only differs as it implied a new set of function [Usage of * (Dereference) and . (Dot) Operators] therefore it would also print as "Ace of Spades", Reason being *aPtr became both becoming a.face/suit. Therefore all the outputs are still "Ace of Spades" which successfully encoded with return to 0.</p>	
<b>2. CODE ANALYSIS</b>	
<p>The program started with 2 header files, in which it utilised for the code performance and functionality. These are, <code>#include &lt;iostream&gt;</code> as its first header in which this is the one responsible for allowing inputs and output operations such as the utilization of <code>cout</code> that were used within the code for specification in printing. Then next he implied <code>#include string</code> as the next header file so that it can hold names, digits and other any special characters that would be used in the code so that it would allow texts. After that, it implemented using namespace <code>std</code>;, to decrease the code text and its repetition of using <code>std::</code>. Before the main program of <code>int main</code>, the code utilized a structure in which it would define any data types such as strings and integers. So, it implemented a structure called "Books" which the one inside it are strings and integers that are implemented as Title, Author, Subjects and its book identification or Book ID. Next the usage of <code>int main()</code> is the starting point of the program where everything inside <code>{}</code> will run when executed. The first thing that is inside it was it declared two book variables assigned as 1 and 2 and then assigned those two books its specifications by its title, author, subjects, and its ID. Finally it printed those specifications with the use of <code>cout</code> and end line for proper spacing with return to 0.</p>	
<b>3. CODE ANALYSIS</b>	
<p>The program started with 2 header files, in which it is the one that is going to be utilized within code outputs and functionality. The first header is <code>#include &lt;iostream&gt;</code> because it is the one responsible for allowing inputs and output operations specifically for <code>cout</code> that were used for printing. Then next it implies <code>#include string</code> as its next header file so that it can hold names, digits and other any special characters that would be used in the code so that it would allow texts. After that, it implemented using namespace <code>std</code>;, to decrease the code text and its repentance of using <code>std::</code>. Before the main function of the code, it first implied a function declaration as "void" for printing books. Then after that inside <code>int main</code>,</p>	

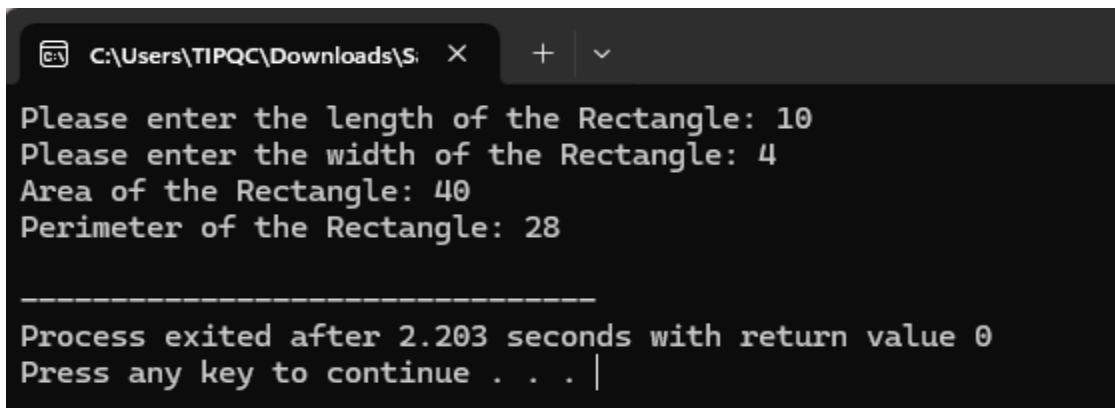
it also declared two Books as variables assigned as 1 and 2 then assigned both books specifications that contains the Title of the Book, its Author, Subjects and also the Book ID or the Books Identification. Next, it implemented its function to print those book details by passing its structure into a function printing both book1 and book2 with return to 0. Lastly, it programmed another void function to print books with the usage of cout and coding its respective names such as (book.title, book.author, book.subject, and book\_book.id). In comparison to the code analysis number 2, functions were used to break larger parts of its program and to avoid repeatability such as the usage of cout book.1 and cout book.2 for the number 2 code and which can be lessened with the use of function.

## 7. Supplementary Activity

### Code

```
[*] Recanglenew.cpp
1  #include <iostream>
2  using namespace std;
3
4  struct Rectangle{
5      double length;
6      double width;
7  };
8
9  void calculateRectangle(const Rectangle rectangle, double& areaofRectangle, double& perimeterofRectangle){
10     areaofRectangle = rectangle.length * rectangle.width;
11     perimeterofRectangle = 2 * (rectangle.length + rectangle.width);
12 }
13
14 int main(){
15     Rectangle input;
16
17     cout<<"Please enter the length of the Rectangle: ";
18     cin>>input.length;
19
20     cout<<"Please enter the width of the Rectangle: ";
21     cin>>input.width;
22
23     double areaOutput;
24     double perimeterOutput;
25
26     calculateRectangle(input, areaOutput, perimeterOutput);
27
28     cout<<"Area of the Rectangle: "<< areaOutput << "\n";
29     cout<<"Perimeter of the Rectangle: " << perimeterOutput << "\n";
30
31 }
32 }
```

### Code Output



```
C:\Users\TIPQC\Downloads\S...  X  +  v
Please enter the length of the Rectangle: 10
Please enter the width of the Rectangle: 4
Area of the Rectangle: 40
Perimeter of the Rectangle: 28

-----
Process exited after 2.203 seconds with return value 0
Press any key to continue . . . |
```

## Code Analysis

I started the program with a header file of `#include <iostream>` for the code outputs and functionality, as this header allows me to do input and out operations which I implemented on my code which is the (cin and cout). After that, I used using namespace std;, so I can avoid the repetition of std:: usage for my input and output operators. Next, I made a structure with the name of "Rectangle" in which the one inside it is length and width as a Rectangle consists of length and width which is going to be utilized in getting its area and perimeter. Then, I made a function called void with a name of calculateRectangle for its calculation, then the one inside are: const or constant Rectangle as rectangle so that it will stay as is when its used for calculation, then double with ampersand (&) with the name of "areaofRectangle" and "perimeterofRectangle") the reason why I used & so that it will give the reference of the original double so meaning in any modifications that happens or when I input a specific number, it will directly be affected and not stay as is. After that, I just implemented the area and the perimeter of the rectangle formulas which correspond to the name I implemented. Next, inside the main function int main where everything inside it will be executed, I set my Rectangle as my "input" so that I can put any number as input which corresponds to the rectangle. Then, I printed that asks for the length and width input of the rectangle that requires a user input. Additionally, I set double areaOutput and perimeterOutput so it will also read decimal places if ever the user implied it. Finally, in order to calculate and reveal its answer I just printed it and used areaOutput and perimeterOutput in which those 2 correspond to the formula.

## Code

```
[*] Multiple.cpp
1  #include <iostream>
2  using namespace std;
3
4  // Function to check if num is a multiple of x
5  bool multiple(int num, int x) {
6      return (num % x == 0);
7  }
8
9  int main() {
10     int num, x;
11
12     cout << "Enter a number: ";
13     cin >> num;
14
15     cout << "Enter the integer to check multiple of: ";
16     cin >> x;
17
18     if (multiple(num, x)) {
19         int quotient = num / x;
20         cout << num << " is a multiple of " << x << endl;
21         cout << "Multiple form: " << num << " = " << x << " * " << quotient << endl;
22     } else {
23         cout << num << " is NOT a multiple of " << x << endl;
24     }
25
26     return 0;
27 }
```

## Code Output

```
C:\Users\Nathaniel\Documents\Multiple.exe
Enter a number: 100
Enter the integer to check multiple of: 2
100 is a multiple of 2
Multiple form: 100 = 2 * 50

-----
Process exited after 2.985 seconds with return value 0
Press any key to continue . . .

C:\Users\Nathaniel\Documents\Multiple.exe
Enter a number: 2
Enter the integer to check multiple of: 4
2 is NOT a multiple of 4

-----
Process exited after 2.055 seconds with return value 0
Press any key to continue . . .
```

## Code Analysis

I started the program with the one header file of `#include <iostream>`, which includes that the program can use input and output operations such as `cin` and `cout`. After that, the line using `namespace std;` is written to avoid repeatedly typing `std::` before standard input and output commands. Next, a function named `multiple` is created. This function takes two integers as parameters, `num` and `x`, and checks if `num` is a multiple of `x` by using the modulus operator `%`. If the remainder is zero, the function returns `true`, meaning that the number is a multiple of `x`. Then inside the `main` function, two integer variables, `num` and `x`, are declared to store user inputs. The program first asks the user to enter a number and then asks for another integer that will be used as the divisor to check if the first number is a multiple of it. These values are read using `cin`. After taking the inputs, the program calls the `multiple` function to perform the check. If the function returns `true`, the program calculates the quotient by dividing `num` by `x`, then prints a message confirming that the number is a multiple. It also displays the multiple form, showing the number expressed as the divisor times the quotient (for example as shown at the code outputs,  $100 = 2 \times 50$ ). If the number is not a multiple, the program prints a message saying that it is not a multiple of the given integer. Finally, the `main` function ends with `return 0;`, which the program ran successfully.

## 8. Conclusion

From this activity, I was able to practice applying C++ programming concepts with our new topic such as structures, functions, and passing arguments by reference. In the rectangle program, I learned how to create a structure to represent a real world object known as (Rectangle) and its relevance to length and width and on how I can get its Area and Perimeter. This activity also made me implement a separate function for me to be able to calculate Rectangles, so that I don't have to keep repeating printing in computing the area and perimeter using the structure data. Additionally, through the use of references (&), the calculated results were directly reflected in the variables declared in `main`. This helped me see the advantage of programming, where separating computation into functions makes the code more organized and reusable. In the multiple-checker program, I applied similar concepts like the rectangle activity but much in a simpler form as I only need to get its multiple and tell if it's not a multiple. Where I implemented a function and its formula in getting it with reliance through functions. Overall, from both the two supplementary activity programs, I learned the importance of breaking down problems into smaller functions, and the usefulness of structures to represent data. I also gained more

experience in applying operators (like % for modulus and / for division), and how references (&) can be used to directly manipulate variables outside of a function. Therefore, I conclude that this activity improved my ability to write structured and maintainable code while also strengthening my logical thinking in solving computational problems.