Activity No. 5.2	
Structures	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Computer Programming Logic and Design	**Date Performed: September 30, 2025**
Section: CPE11S1	**Date Submitted: October 4, 2025**
Name(s): Lopez, Andrei Dion C.	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

Code 1:

- Screenshot of Code(Readable):

```
#include <iostream>
#include <string>
using namespace std;

struct Card {
   string face;
   string suit;
};

int main() {
   Card a;      // structure variable
   Card* aPtr;  // structure pointer

   // Assign values
   a.face = "Ace";
   a.suit = "Spades";

   // Pointer points to structure 'a'
   aPtr = &a;

   // Accessing members:
   //  Using the dot operator (.)
   cout << a.face << " of " << a.suit << endl;

   //  Using the arrow operator (->)
   cout << aPtr->face << " of " << aPtr->suit << endl;

   //  Using dereference (*) and dot
   cout << (*aPtr).face << " of " << (*aPtr).suit << endl;

   return 0;
}
```

- In this code it initializes 2 preprocessor commands, those being "**<iostream>**" and "**<string>**", what this does is it allows the use of the standard C++ library and the C++ string library. It then initialized a structure called "**Card**" . This structure's main goal is to initiate 2 members for face and suit, afterwards it initializes the main code where it declares the structure variable and pointer "**a**", which is then assigned the values "**Ace**" and "**Spades**" which are respectively assigned to the values "**face**" and "**suit**", afterwards we assign the value of the structure pointer to the address of "**a**" which is used to access the members of the structure "**Card**". The code then moves onto the different ways to access the members and output the assigned values for the members. Using dot operator "**.**", arrow operator "**->**", and dereference and dot operators " *** and .**" which will all print out the text "**Ace of Spades**".

Code 2:

- Screenshot of Code(Readable):

```
#include <iostream>
#include <string>
using namespace std;

// Define the structure
struct Books {
   string title;
   string author;
   string subject;
   int book_id;
};

int main() {
   // Declare two Book variables
   Books Book1;
   Books Book2;

   // Book 1 specification
   Book1.title = "C Programming";
   Book1.author = "Nuha Ali";
   Book1.subject = "C Programming Tutorial";
   Book1.book_id = 6495407;

   // Book 2 specification
   Book2.title = "Telecom Billing";
   Book2.author = "Zara Ali";
   Book2.subject = "Telecom Billing Tutorial";
   Book2.book_id = 6495700;

   // Print Book 1 info
   cout << "Book 1 title   : " << Book1.title << endl;
   cout << "Book 1 author  : " << Book1.author << endl;
   cout << "Book 1 subject : " << Book1.subject << endl;
   cout << "Book 1 book_id : " << Book1.book_id << endl;

   cout << endl; // for spacing

   // Print Book 2 info
   cout << "Book 2 title   : " << Book2.title << endl;
   cout << "Book 2 author  : " << Book2.author << endl;
   cout << "Book 2 subject : " << Book2.subject << endl;
   cout << "Book 2 book_id : " << Book2.book_id << endl;

   return 0;
}
```

- This code uses a structure to output the information of 2 different books. The way it does this is by initializing a structure called "**Books**" and puts 4 members inside that structure, Those members being the strings "**title**", "**author**", "**subject**". and the integer "**book_id**". after initializing the structure it moves on to initializing the main function where it first declares 2 book variables called "**Book1**" and "**Book2**", then it starts assigning values for each book like the title, the author, its subject, and its id. afterwards it outputs those values using the dot operator.

Code 3:

- Screenshot of Code(Readable):

```
#include <iostream>
#include <string>
using namespace std;

// Define a structure
struct Books {
   string title;
   string author;
   string subject;
   int book_id;
};

// Function declaration (structure passed by value)
void printBook(Books book);

int main() {
   // Declare two Book variables
   Books Book1;
   Books Book2;

   // Book 1 specification
   Book1.title = "C Programming";
   Book1.author = "Nuha Ali";
   Book1.subject = "C Programming Tutorial";
   Book1.book_id = 6495407;

   // Book 2 specification
   Book2.title = "Telecom Billing";
   Book2.author = "Zara Ali";
   Book2.subject = "Telecom Billing Tutorial";
   Book2.book_id = 6495700;

   // Print details by passing the structure to a function
   printBook(Book1);
   cout << endl; // just for spacing
   printBook(Book2);

   return 0;
}

// Function definition
void printBook(Books book) {
   cout << "Book title   : " << book.title << endl;
   cout << "Book author  : " << book.author << endl;
   cout << "Book subject : " << book.subject << endl;
   cout << "Book book_id : " << book.book_id << endl;
}
```

- This program, similarly to the program earlier, prints out the information of 2 different books. But what makes this one different is how it uses a function to print out the information stored inside the structure called "**Books**" where, after initializing the structure declares the void function "**printBook**" before the main function, and defines the function afterwards by inserting the predefined object "**cout**" and the string values that are located inside the quotation of the cout object.

7. Supplementary Activity

Code 1:

- Screenshot of Code(Readable):

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

struct Rect_Dim {
    float length;
    float width;
};

void Calc(const Rect_Dim& rect, float& area, float& perimeter) {
    area = rect.length * rect.width;
    perimeter = 2 * (rect.length + rect.width);
}

int main() {

        Rect_Dim Rect;

              cout << "Input Length: ";
              cin >> Rect.length;
              cout << "Input Width: ";
              cin >> Rect.width;

        float Calcarea;
        float Calcperimeter;

        Calc(Rect, Calcarea, Calcperimeter);

        cout << "Perimeter of the Rectangle is: " << setprecision(3) << Calcperimeter<< endl;
        cout << "Area of the Rectangle is: " << setprecision(3) << Calcarea << endl;

    return 0;
}
```

- Output of Code(label and compile ALL possible outputs):

```
Input Length: 5
Input Width: 8
Perimeter of the Rectangle is: 26
Area of the Rectangle is: 40

--------------------------------
Process exited after 1.994 seconds with return value 0
Press any key to continue . . .
```

- In this code I used 2 preprocessor directives “**iostream**” and “**iomanip**” . What iomanip does is allow me to use the “**setprecision**” function to allow a specific amount of numbers to be displayed. Then I initiated a structure that would store the length and width of a rectangle and a void function to calculate both the area and the perimeter of the rectangle without needing to return a value, and then I moved onto the main function where I declared the variable for the structure and added an input function and then initialized 2 variables for the calculated perimeter and calculated area and then called the function to calculate the 2 values given by the user.

Code 2:

- Screenshot of Code(Readable):

```cpp
#include <iostream>
using namespace std;

bool multiple(int n, int x) {
    return (n % x == 0);
}

int main() {
    int n, x;

    cout << "Enter a number: ";
    cin >> n;
    cout << "Enter divisor x: ";
    cin >> x;

    cout << endl;

    if (multiple(n, x)) {
        cout << n << " is a multiple of " << x << endl;
    } else {
        cout << n << " is not a multiple of " << x << endl;
    }


    return 0;
}
```

- Output of Code(label and compile ALL possible outputs):

```
Enter a number: 20
Enter divisor x: 5

20 is a multiple of 5

--------------------------------
Process exited after 3.383 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number: 5
Enter divisor x: 20

5 is not a multiple of 20

--------------------------------
Process exited after 1.667 seconds with return value 0
Press any key to continue . . .
```

- In this program I used a bool function to output true if the function n % x is equal to the value of 0, other than that the function will return false. afterwards I started on the main function where I initialized 2 variables that will store the numbers the user inputted. Then using an if statement to check whether the output of the bool function is true or false, where if it was true it would output that the integer "**n**" is a multiple of the integer "**x**". Otherwise, if it was false, it would output that the integer "**n**" is not a multiple of the integer "**x**".

8. Conclusion

In this lesson and activity I have learnt the usage of structure and functions in the world of C++, how there are many ways to access the members of a structure, and the multiple uses of functions to optimize coding in C++. These 2 lessons were used in our activity to create a program that stores the length and width of a rectangle and computes its area and perimeter, and a program that checks if a number is a multiple of another number using functions. The way this is achieved is by using the modulo operator "**%**" and checking if the output of the operation is equal to zero. I felt that in this activity I have done better than compared to my past activities as I have understood how these lessons work together to create a code that is more optimized and easier to use.