Activity No. 5.2	
Structures	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: September 30-October 3, 2025**
Section: CPE11S1	**Date Submitted: October 3, 2025**
Name(s): Jaime Luis M. Demain	**Instructor: Engr. Jimlord M. Quejado**
6. Output	

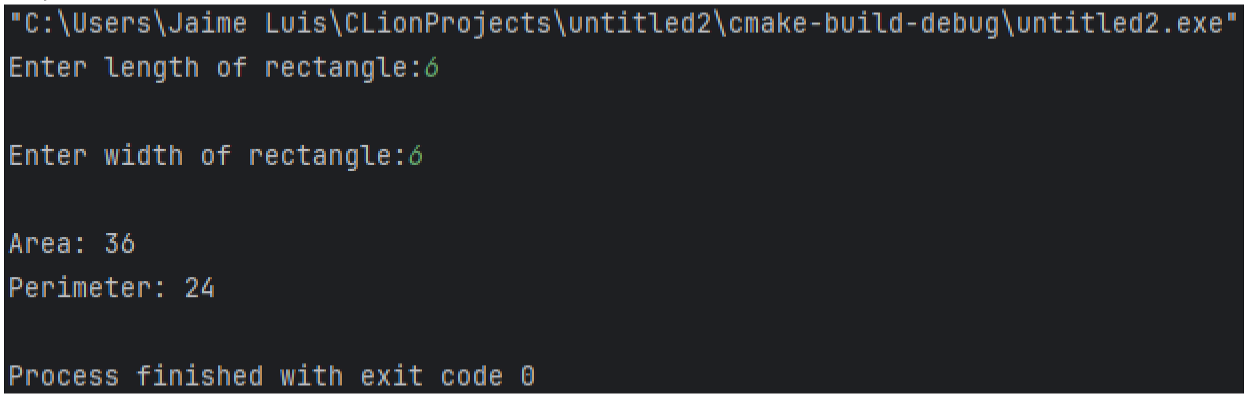
1. **Analysis:**
   In this item, we will print out groups of words or strings using structures in three different modes (dot, arrow, and dereference + dot). The first part to do this we will initialize a string header first so that we can a string data type. And then, a struc function will be initialized together with the structure tag. After this, we will declare structure members within its brace so that we will have variables to assign values later on. We will declare structure members under string data type. After declaring members of structure within struc function, we will now initialize a main function. After main function initialization, we will declare a structure variable (a) to be used in assigning values and pointer (->) to access values. And then, values will be assigned through accessing a structure member via a structure variable in the dot operator. Then, a pointer will point towards a memory address of structure variable "a" so that a means of printing out a value using the pointer will be initialized. After this, the values will now be printed out in first mode (dot operator). It will just consist of variable and structure members with dots sitting between them. And then, we will print out the same values using second mode (arrow operator). It will basically just do the same function (printing values) although in a different form (pointer accessing values through memory address). And the third mode (dereference + dot) is just the same as in the second mode. The only difference between second and third mode is how the language is used to execute a function. After printing out the values using three modes, the output will be the same as all of modes perform similar functions.
2. **Analysis:**
   In this item, we will print out a book's info and its details (values) using structures and dot operators. The first thing in this code is to initialize a string header in order to utilize the string function because this function is core part to our objective that contains words (group of letters). Then after initializing a string header, the struct function, its tag and members will be initialized. Only book id variable will be initialized an int function instead of string function because it contains an actual number values instead of just sequence of number and it is not a letter. After initializing and defining a struct function, the main function will now be initialized to be followed by its variables, assigned values, and actual printing out. After the main function's initialization, structure variables will be declared to be used in assigning values. The next step is to assign values using the structure variable (in the main function) and structure members (in the struct function) with dots sitting between them. After assigning values, it will proceed to the next step which is printing out the desired info. It will consist of cout << "(Users intended message" << (Structure Variables).(Structure Member) << endl with cout << endl sitting between Book 1 and 2. After all of this process, the output will be printed out.
3. **Analysis:**
   In this part, the output will be almost similar but it will use a different approach. That is a void function so that the code will be shorter and more optimized. How has the process been conducted? Well before initializing a void function, a string header file and a struct function has been initialized first. And then, a void function will be initialized alongside its name variable and parameter (its first variable to be used in declaring storage variables and second variable in printing out info that is attached to the first variable) to allow details of separate books (separate structure variables) to be printed out in one go by calling out the structure variables instead of manually printing it out and to make the main function more concise. After the initialization of the void function, a main function will be initialized . Within a brace on the main function, structure variables will be declared followed by assigning values (the details) on each book. And then after values have been assigned on two separate books, the details will be printed out according to the name variable of void function and the number of books (to be referred to what number of books will be printed out) within a parenthesis. This is where void function helps by

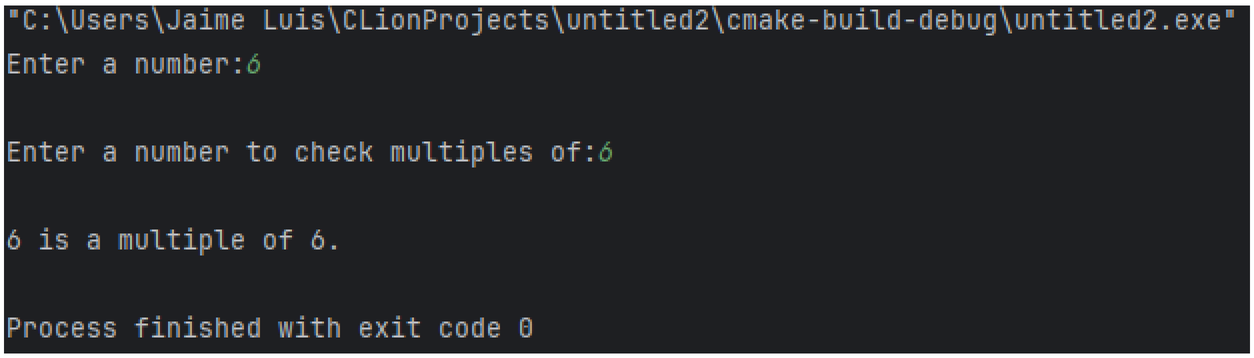
avoiding multiple manual cout operations to print out the values. Endl will sit between Book 1 and 2 for spacing. Outputs will be printed out afterwards. The code isn't done yet afterwards as we need to print out the book specifications details (title, author, subject, and book id). We will also use void function on this part like the first one but we will utilize the second parameter (book) of a void function to print out the specification details. The code within a second void function is also similar to the previous item, it just uses a book variable instead of the Book(number) variable in the first part (before dot) of the variable for code to be optimized in printing out the specification details of a book. Overall, the first void function in this item is that it can refer to what block will be printed out first before the actual details/specifications of a block will be printed out (in second void function).

7. Supplementary Activity

1.
Code:

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

struct Size{
    double length;
    double width;
};

int compute(const Size rectangle){
    double area = rectangle.width * rectangle.length;
    double perimeter = 2 * (rectangle.width + rectangle.length);

    cout << "Area: " << area << endl;
    cout << "Perimeter: " << perimeter << endl;
}

int main (){
    Size myRectangle;

    cout << "Enter length of rectangle: ";
    cin >> myRectangle.length;
    cout << "Enter width of rectangle: ";
    cin >> myRectangle.width;

    compute (myRectangle);

    return 0;
}
```

Output:

```
"C:\Users\Jaime Luis\CLionProjects\untitled2\cmake-build-debug\untitled2.exe"
Enter length of rectangle:6

Enter width of rectangle:6

Area: 36
Perimeter: 24

Process finished with exit code 0
```
Analysis:
What I did in this part is to code a structure to store a rectangle's length and width, use a void function to initialize and process out area and perimeter variables according to the inputted length and width and print it out, and initialize an inputting mechanism for length and width. The first part is to initialize a struct function first to initialize and store "length" and "width" variables as structure members that can be used later in the coding. And then, the int function will be initialized and then the "area" and "perimeter" variables will be declared and processed out their respective formulas through mathematical operators. After processing out their formulas, the "area" and "perimeter" variables will be printed out. After this part, the main function will be initialized. Next is to declare a structure variable in order to have variables for coding the inputting mechanisms for length and width. After this, the inputting mechanisms will now be processed out. In the last part, the int function and its contents will be called out for actual processing and printing.
2.
Code:
```
#include <iostream>
using namespace std;

bool multiple(int num, int mult) {
    return (num % mult == 0);
}

int main() {
    int num, mult;

    cout << "Enter a number: ";
    cin >> num;

    cout << "Enter a number to check multiples of: ";
    cin >> mult;

    if (multiple(num, mult)) {
        cout << num << " is a multiple of " << mult << "." << endl;
    }
    else {
        cout << num << " is NOT a multiple of " << mult << "." << endl;
    }

    return 0;
}
```
Output:



```
"C:\Users\Jaime Luis\CLionProjects\untitled2\cmake-build-debug\untitled2.exe"
Enter a number:6

Enter a number to check multiples of:6

6 is a multiple of 6.

Process finished with exit code 0
```

Analysis:
In this part, I will code a function that checks whether an inputted number is multiple of the particular another inputted number. In the first part, the boolean function will be used in order to take a variable “num” and “mult” as inputs and define a condition (0 remaining integer) when the value can be considered true. After this, the main function will be initialized so that we can code an inputting mechanism for number and a multiple for checking. Then, an if-else statement will be used from the boolean function’s variable to check if a number is a multiple of an another number or not.

8. Conclusion

What I learned in this activity is that we analyzed and coded structures and different functions so that we can understand its process and allow us to code more efficiently and optimized. Structures are used to store a group of variables with different data types. This will allow us to encapsulate data and make our coding more organized and readable. Functions meanwhile is a reusable block of code that executes block statements and it helps us to make our coding more efficient and optimized by cutting down repetitions in our code. In the output section of this activity, we just analyzed an example of structures and function coding so that we can understand how it structures and how it functions. This will help us to gain more knowledge about structures and function. Supplementary activities meanwhile allow us to code something that utilizes structures and functions to see how it helps to make coding more readable, organized, and optimized. From what I notice, it indeed reduces repetitions in coding and improves the readability of code. When we will code in the future, we already have an idea how an optimized version of code can be made. This will help when we will program something and perform its best. Overall, structures and functions are really big help to those who are novel about coding particularly in C++ language.