

Activity No. 2.2	
Hands-on Activity 2.2: Control Structures (part 1)	
<b>Course Code:</b> CPE007	<b>Program:</b> Computer Engineering
<b>Course Title:</b> Programming Logic and Design	<b>Date Performed:</b> 8/11/2025
<b>Section:</b> CPE11S1	<b>Date Submitted:</b> 8/12/2025
<b>Name(s)</b> Ralph Angelov F. Braganza	<b>Instructor:</b> Engr. Jimlord M. Quejado
<b>Output</b>	
<p><b>Exercise 1:</b> Counter- Controlled Repetition. A class of ten students took a quiz. The grades (integers in the range of 0 to 100) for this quiz are available to you. Determine the class average on the quiz. Put your answer in the output section of the activity template. <b>Ensure that the screen shot of the code and the output are readable.</b></p> <p>Using the following <b>pseudocode</b> the program can be as follows:</p> <p><i>Set total to zero  Set grade counter to one  While grade counter is less than or equal to ten  Input the next grade  Add the grade into the total  Add one to the grade counter  Set the class average to the total divided by ten  Print the class average</i></p> <p><b>CODE:</b></p> <pre>#include &lt;iostream&gt;  int main() {     int total = 0;     int grade_counter = 1;     int grade;     double average; // this will be able to display decimals      while (grade_counter &lt;= 10) { // this will create a loop         std::cout &lt;&lt; "Enter Grade " &lt;&lt; grade_counter &lt;&lt; ": ";         std::cin &gt;&gt; grade;          total = total + grade;         grade_counter = grade_counter + 1;     }      average = static_cast&lt;double&gt;(total) / 10;     std::cout &lt;&lt; "\nClass average is: " &lt;&lt; average &lt;&lt; std::endl;      return 0; }</pre>	

## RESULT:

main.cpp				Output	
<pre>1 #include &lt;iostream&gt; 2 3 int main() { 4 5     int total = 0; 6     int grade_counter = 1; 7     int grade; 8     double average; // this will be able to display decimals 9 10    while (grade_counter &lt;= 10) { // this will create a loop 11        std::cout &lt;&lt; "Enter Grade " &lt;&lt; grade_counter &lt;&lt; ": "; 12        std::cin &gt;&gt; grade; 13 14        total = total + grade; 15        grade_counter = grade_counter + 1; 16    } 17 18    average = static_cast&lt;double&gt;(total) / 10; 19    std::cout &lt;&lt; "\nClass average is: " &lt;&lt; average &lt;&lt; std::endl; 20 21    return 0; 22 }</pre>	Enter Grade 1: 91 Enter Grade 2: 92 Enter Grade 3: 93 Enter Grade 4: 94 Enter Grade 5: 95 Enter Grade 6: 96 Enter Grade 7: 97 Enter Grade 8: 98 Enter Grade 9: 99 Enter Grade 10: 100  Class average is: 95.5  ==== Code Execution Successful ===				

## Supplementary Activity

1. Using conditional statements (if-else statements), write a program that asks a user for a number and prints out if it is an even or an odd number.

## CODE:

```
#include <iostream>
```

```
int main() {
    int value;

    std::cout << "Enter Value: ";
    std::cin >> value;

    if (value % 2 == 0) {
        std::cout << value << " is an even value." << std::endl;
    } else {
        std::cout << value << " is an odd value." << std::endl;
    }

    return 0;
}
```

## RESULT: EVEN VALUE

The screenshot shows a code editor window with the file name 'main.cpp'. The code is a simple C++ program that checks if a user-entered value is even or odd. When the user enters '100', the output window shows '100 is an even value.' and a message indicating successful execution.

```
1 #include <iostream>
2
3 int main() {
4     int value;
5
6     std::cout << "Enter Value: ";
7     std::cin >> value;
8
9     if (value % 2 == 0) {
10         std::cout << value << " is an even value." << std::endl;
11     } else {
12         std::cout << value << " is an odd value." << std::endl;
13     }
14
15     return 0;
16 }
```

Output:

```
Enter Value: 100
100 is an even value.

*** Code Execution Successful ***
```

## ODD VALUE

The screenshot shows a code editor window with the file name 'main.cpp'. The code is identical to the even value program but checks for odd values. When the user enters '99', the output window shows '99 is an odd value.' and a message indicating successful execution.

```
1 #include <iostream>
2
3 int main() {
4     int value;
5
6     std::cout << "Enter Value: ";
7     std::cin >> value;
8
9     if (value % 2 == 0) {
10         std::cout << value << " is an even value." << std::endl;
11     } else {
12         std::cout << value << " is an odd value." << std::endl;
13     }
14
15     return 0;
16 }
```

Output:

```
Enter Value: 99
99 is an odd value.

*** Code Execution Successful ***
```

2. Using conditional statements, write a program that computes for 10 percent fare discount of a senior citizen and 8 percent fare discount of a student. There will be no discount if not a senior citizen and not a student. The user will be asked to enter age. The minimum fare is 9 pesos.

## CODE:

```
#include <iostream>

int main() {
    int age;
    double fare = 9.0;
    double farefinal;

    std::cout << "Enter Your Age: ";
    std::cin >> age;

    if (age >= 60) { // 60 years old is considered being a senior citizen (Philippines).
        farefinal = fare - (fare * 0.10);
        std::cout << "Senior Discount Fare: " << farefinal << " Pesos" << std::endl;
    } else if (age <= 25) { // anyone could be still studying at a later age but Im going with this
        farefinal = fare - (fare * 0.08);
        std::cout << "Student Discount Fare: " << farefinal << " Pesos" << std::endl;
    }
}
```

```

} else {
    farefinal = fare; // no discount
    std::cout << "Regular Fare: " << farefinal << " Pesos" << std::endl;
}

return 0;
}

```

## RESULT:

### SENIOR DISCOUNT FARE

The screenshot shows a code editor interface with a dark theme. On the left, the code file `main.cpp` is displayed. The code implements a fare discount logic based on age. It includes comments explaining the logic for senior citizens (age ≥ 60) and students (age ≤ 25). The `Run` button is highlighted in blue. To the right, the `Output` panel shows the program's output. A user enters the age `69`, and the program calculates and prints the senior discount fare as `8.1 Pesos`. A message at the bottom indicates a successful code execution.

```

main.cpp
[Run] Output [Clear]

1 #include <iostream>
2
3 int main() {
4     int age;
5     double fare = 9.0;
6     double farefinal;
7
8     std::cout << "Enter Your Age: ";
9     std::cin >> age;
10
11    if (age >= 60) { // 60 years old is considered being a senior citizen (Philippines).
12        farefinal = fare - (fare * 0.10);
13        std::cout << "Senior Discount Fare: " << farefinal << " Pesos" << std::endl;
14    } else if (age <= 25) { // anyone could be still studying at a later age but Im going
15        with this
16        farefinal = fare - (fare * 0.08);
17        std::cout << "Student Discount Fare: " << farefinal << " Pesos" << std::endl;
18    } else {
19        farefinal = fare; // no discount
20        std::cout << "Regular Fare: " << farefinal << " Pesos" << std::endl;
21    }
22
23 }
24

```

Enter Your Age: 69  
Senior Discount Fare: 8.1 Pesos  
--- Code Execution Successful ---

### STUDENT DISCOUNT FARE

This screenshot is similar to the previous one, showing the same code editor interface. The code is identical to the first one but with a different input age. A user enters the age `19`, and the program calculates and prints the student discount fare as `8.28 Pesos`. A message at the bottom indicates a successful code execution.

```

main.cpp
[Run] Output [Clear]

1 #include <iostream>
2
3 int main() {
4     int age;
5     double fare = 9.0;
6     double farefinal;
7
8     std::cout << "Enter Your Age: ";
9     std::cin >> age;
10
11    if (age >= 60) { // 60 years old is considered being a senior citizen (Philippines).
12        farefinal = fare - (fare * 0.10);
13        std::cout << "Senior Discount Fare: " << farefinal << " Pesos" << std::endl;
14    } else if (age <= 25) { // anyone could be still studying at a later age but Im going
15        with this
16        farefinal = fare - (fare * 0.08);
17        std::cout << "Student Discount Fare: " << farefinal << " Pesos" << std::endl;
18    } else {
19        farefinal = fare; // no discount
20        std::cout << "Regular Fare: " << farefinal << " Pesos" << std::endl;
21    }
22
23 }
24

```

Enter Your Age: 19  
Student Discount Fare: 8.28 Pesos  
--- Code Execution Successful ---

## REGULAR FARE

The screenshot shows a code editor window with the following details:

- Title Bar:** Shows "main.cpp".
- Toolbar:** Includes icons for Open, Save, Share, Run, and Clear.
- Code Area:** Contains the following C++ code:

```
1 #include <iostream>
2
3 int main() {
4     int age;
5     double fare = 9.0;
6     double farefinal;
7
8     std::cout << "Enter Your Age: ";
9     std::cin >> age;
10
11    if (age >= 60) { // 60 years old is considered being a senior citizen (Philippines).
12        farefinal = fare - (fare * 0.10);
13        std::cout << "Senior Discount Fare: " << farefinal << " Pesos" << std::endl;
14    } else if (age <= 25) { // anyone could be still studying at a later age but Im going
15        with this
16        farefinal = fare - (fare * 0.08);
17        std::cout << "Student Discount Fare: " << farefinal << " Pesos" << std::endl;
18    } else {
19        farefinal = fare; // no discount
20        std::cout << "Regular Fare: " << farefinal << " Pesos" << std::endl;
21    }
22
23    return 0;
24 }
```
- Output Area:** Displays the program's output:

```
Enter Your Age: 30
Regular Fare: 9 Pesos
*** Code Execution Successful ***
```

3. **Case Study:** Sentinel Controlled Repetition. Given the following pseudocode, create a program that will implement a sentinel-controlled repetition. For example, you can use (-1) as the sentinel value. You can use Problem 1 as your reference.

### Pseudocode:

Initialize total to zero

Initialize counter to zero

Input the first grade

While the user has not as yet entered the sentinel

    Add this grade into the running total

    Add one to the grade counter

    Input the next grade (possibly the sentinel)

If the counter is not equal to zero

    Set the average to the total divided by the counter

    Print the average

else

    Print "No grades were entered"

**CODE:**

```
#include <iostream>
```

```
int main() {
    int grade;
    int total = 0;
    int counter = 0;
    double average;

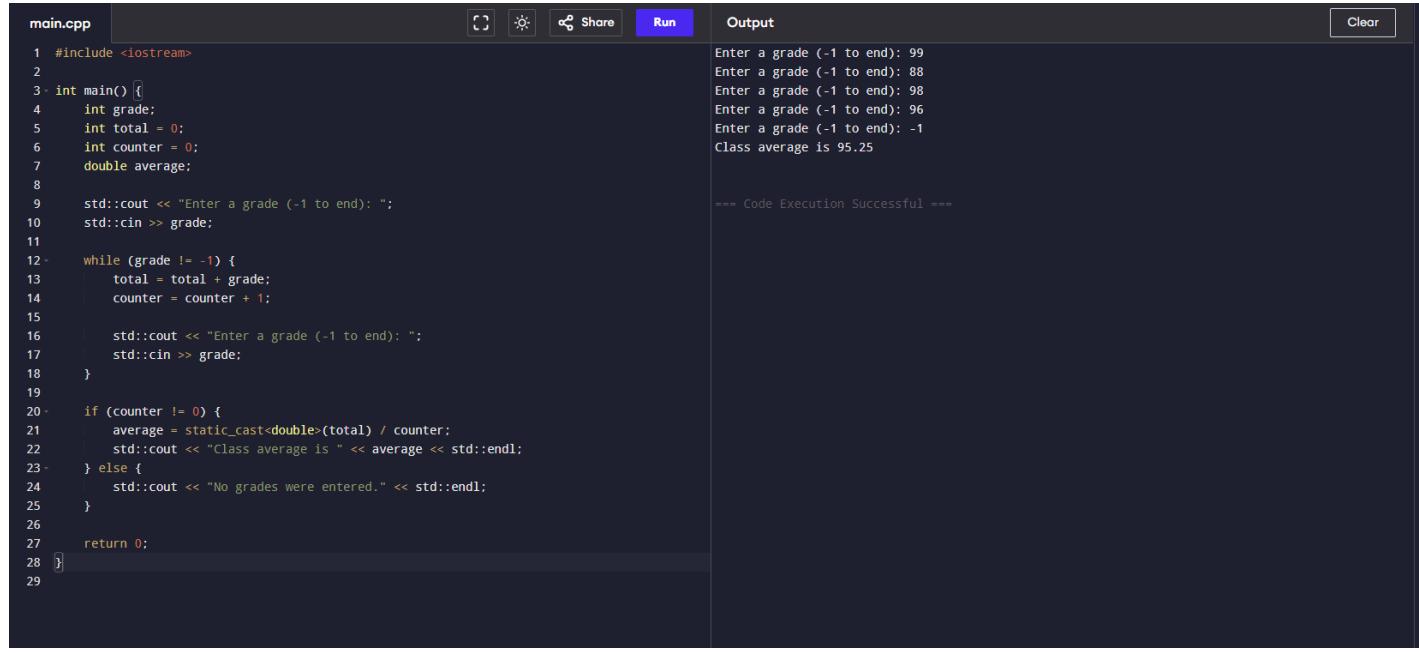
    std::cout << "Enter a grade (-1 to end): ";
    std::cin >> grade;

    while (grade != -1) {
        total = total + grade;
        counter = counter + 1;

        std::cout << "Enter a grade (-1 to end): ";
        std::cin >> grade;
    }

    if (counter != 0) {
        average = static_cast<double>(total) / counter;
        std::cout << "Class average is " << average << std::endl;
    } else {
        std::cout << "No grades were entered." << std::endl;
    }
}

return 0;
}
```

**RESULT:****CLASS AVERAGE**

The screenshot shows a code editor interface with a dark theme. On the left is the code editor pane displaying the `main.cpp` file. On the right is the output pane showing the execution results.

**Code Editor (main.cpp):**

```
1 #include <iostream>
2
3 int main() {
4     int grade;
5     int total = 0;
6     int counter = 0;
7     double average;
8
9     std::cout << "Enter a grade (-1 to end): ";
10    std::cin >> grade;
11
12    while (grade != -1) {
13        total = total + grade;
14        counter = counter + 1;
15
16        std::cout << "Enter a grade (-1 to end): ";
17        std::cin >> grade;
18    }
19
20    if (counter != 0) {
21        average = static_cast<double>(total) / counter;
22        std::cout << "Class average is " << average << std::endl;
23    } else {
24        std::cout << "No grades were entered." << std::endl;
25    }
26
27    return 0;
28 }
```

**Output:**

```
Enter a grade (-1 to end): 99
Enter a grade (-1 to end): 88
Enter a grade (-1 to end): 98
Enter a grade (-1 to end): 96
Enter a grade (-1 to end): -1
Class average is 95.25

*** Code Execution Successful ***
```

NO GRADES WERE ENTERED.

main.cpp

The screenshot shows a code editor interface with the following components:

- Title Bar:** The title "main.cpp" is visible at the top left.
- Toolbar:** A toolbar at the top right contains icons for Save, Run, Share, and Clear.
- Code Area:** The main area displays the C++ code. The code includes declarations for `grade`, `total`, and `counter` as integers, and `average` as a double. It uses `std::cout` and `std::cin` for input and output. The code prompts the user to enter grades until -1 is entered, calculates the total and count, and then prints the average or a message if no grades were entered.
- Output Area:** To the right of the code, the output window shows the interaction with the program:
  - It asks for a grade: "Enter a grade (-1 to end): -1".
  - It responds with "No grades were entered."
  - At the bottom, it displays the message "== Code Execution Successful ==".

```
1 #include <iostream>
2
3 int main() {
4     int grade;
5     int total = 0;
6     int counter = 0;
7     double average;
8
9     std::cout << "Enter a grade (-1 to end): ";
10    std::cin >> grade;
11
12    while (grade != -1) {
13        total = total + grade;
14        counter = counter + 1;
15
16        std::cout << "Enter a grade (-1 to end): ";
17        std::cin >> grade;
18    }
19
20    if (counter != 0) {
21        average = static_cast<double>(total) / counter;
22        std::cout << "Class average is " << average << std::endl;
23    } else {
24        std::cout << "No grades were entered." << std::endl;
25    }
26
27    return 0;
28 }
29
```

## Conclusion

The tasks were very difficult for me, I struggled A LOT with the operators and how they functioned, but I managed to pull through and complete the task that was given to me. I had a lot of trouble assigning things to one another, and I kept forgetting basic things like putting a semicolon at the end of the line, adding "std:::" and many more. One of the most difficult tasks was the case study; figuring out and understanding how the "!" function works took me a long time, and the senior/student fair activity was also another task that made me confused because of the loops. Other than that, after hours of coding and trying to understand, I could say I have obtained newfound knowledge with this hands-on activity.

## **Assessment Rubric**

Rubric for SO 7 (7)							
Criteria	Ratings						Pts
Ⓐ SO 7 PI 1 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent   Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good   Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory   Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory   Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor   Relies on classroom instruction only	1 pts Very Poor   No initiative or interest in acquiring new knowledge	6 pts
Ⓑ SO 7 PI 2 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent   Completes an assigned task independently and practices continuous improvement	5 pts Good   Completes an assigned task without supervision or guidance	4 pts Satisfactory   Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory   Requires detailed or step-by-step instructions to complete a task	2 pts Poor   Shows little interest to complete a task independently	1 pts Very Poor   No interest to complete a task independently	6 pts
Ⓒ SO 7 PI 3 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent   Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good   Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory   Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory   Apply the gathered information to formulate the problem	2 pts Poor   Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor   Gather information from a variety of sources	6 pts
Ⓓ SO 7 PI 4 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent   Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good   Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory   Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory   Shows some creative ways to solve the problem	2 pts Poor   Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor   Ideas are copied or restated from the sources consulted	6 pts