National University of Modern Language

(Karachi Campus)

Student: **Ali Abid**

Roll: **BSCS-KC-006**

Department: **Computer-Science batch-1**

Course: **Programming Fundamentals (LAB TASKS)**

Instructor**: Miss Sidra Yousuf**

**For GitHub Repo :** [**Click here**](https://github.com/abidkk/NUML_PF_LAB_TASKS)

**OR**

**Visit : https://github.com/abidkk/NUML\_PF\_LAB\_TASKS**

Week 1 Introduction

Note: Week 1 is theory based. No example of this week.

**Week 2**

# Topic: character-set\_white-space

**Example-1**

|  |
| --- |
| // Example 1: Using Character Set  #include <iostream>  int main() {  char letter = 'A';  std::cout << "Character: " << letter << std::endl;  return 0;  } |

**Example-2**

|  |
| --- |
| #include <iostream>  int main() {  std::cout << "This is a line with a tab\tand a newline\n" << std::endl;  return 0;  } |

**Example-3**

|  |
| --- |
| #include <iostream>  int main() {  std::cout << "Hello" << ' ' << "World" << '\n'; // space and newline  std::cout << "Hello\tWorld\n"; // horizontal tab and newline  std::cout << "Line1\rLine2\n"; // carriage return and newline  std::cout << "Hello\fWorld\n"; // form feed and newline  std::cout << "Hello\vWorld\n"; // vertical tab and newline  return 0;  } |

**Topic: 02\_naming-rules**

**Example-1**

|  |
| --- |
| // Valid Variable Name  #include <iostream>  int main() {  int age = 25;  double salary = 50000.50;  std::cout << "Age: " << age << ", Salary: " << salary << std::endl;  return 0;  } |

**Example-2**

|  |
| --- |
| // Example 2: Invalid Variable Names  #include <iostream>  int main() {  // int 1stValue = 10; // Invalid variable name  int firstValue = 10; // Valid variable name  std::cout << "First Value: " << firstValue << std::endl;  return 0;  } |

**Example-3**

|  |
| --- |
| #include <iostream>  int main() {  // Valid identifiers  int \_value = 10;  int value2 = 20;  int Value = 30;  int my\_variable = 40;  int anotherVariable = 50;  std::cout << \_value << " " << value2 << " " << Value << " " << my\_variable << " " << anotherVariable << std::endl;  // Invalid identifiers (uncomment to see errors)  // int 2value = 60; // starts with a digit  // int my-variable = 70; // contains a hyphen  // int class = 80; // 'class' is a keyword  return 0;  } |

**Topic : 03\_data-types\_sizes**

**Example-1**

|  |
| --- |
| // Different Datatypes  #include <iostream>  int main() {  int integerVar = 100;  float floatVar = 10.5;  char charVar = 'A';  bool boolVar = true;  std::cout << "Integer: " << integerVar << "\nFloat: " << floatVar << "\nCharacter: " << charVar << "\nBoolean: " << boolVar << std::endl;  return 0;  } |

**Example-2**

|  |
| --- |
| // Data types sizes  #include <iostream>  int main() {  std::cout << "Size of int: " << sizeof(int) << " bytes" << std::endl;  std::cout << "Size of float: " << sizeof(float) << " bytes" << std::endl;  std::cout << "Size of double: " << sizeof(double) << " bytes" << std::endl;  std::cout << "Size of char: " << sizeof(char) << " bytes" << std::endl;  return 0;  } |

**Week 3**

**Topic :**

**01\_first-program**

**Example-1**

|  |
| --- |
| #include <iostream>  using namespace std;  int main(){  cout << "Hello, world!" << endl;  cout << "Hello, Abid" << endl;  return 0;  } |

**Example-2**

|  |
| --- |
| #include <iostream>  using namespace std;  int main(){  cout << "Hello, world!" << endl;  cout << "Hello, Abid" << endl;  return 0;  } |

Topic :

**02\_preprocessor-directive**

**Example-1**

|  |
| --- |
| // Example 1: Using #define  #include <iostream>  #define PI 3.14159  int main() {  std::cout << "The value of PI is: " << PI << std::endl;  return 0;  } |

**Example-2**

|  |
| --- |
| // Example 2: Using #include  #include <iostream>  #include <cmath>  int main() {  std::cout << "Square root of 16 is: " << sqrt(16) << std::endl;  return 0;  } |

Topic :

**03\_function \_body\_and\_statement\_terminator**

**Example-1**

|  |
| --- |
| // Example 1: Function Body  #include <iostream>  void greet() {  std::cout << "Hello, welcome to C++ programming!" << std::endl;  }  int main() {  greet();  return 0;  } |

**Example-2**

|  |
| --- |
| // Example 2: Statement Terminator  #include <iostream>  int main() {  int a = 10; // Statement terminator is the semicolon  int b = 20;  int sum = a + b;  std::cout << "Sum: " << sum << std::endl;  return 0;  } |

**Week 4**

Topic :

**01\_Variable\_definition-and-declaration**

**Example-1**

|  |
| --- |
| // Variable declarations  #include <iostream>  int main() {  // Declaration of variables  extern int a; // 'a' is declared, not defined  extern float b;    return 0;  } |

**Example-2**

|  |
| --- |
| #include <iostream>  // Declaration of variables  extern int x;  extern double y;  int main() {  // Definition of variables  int x = 20;  double y = 5.67;  std::cout << "x: " << x << ", y: " << y << std::endl;  return 0;  } |

Topic :

**02\_Escape-sequences**

**Example-1**

|  |
| --- |
| #include <iostream>  int main() {  std::cout << "Hello, World!\n"; // newline  std::cout << "Hello,\tWorld!\n"; // horizontal tab  std::cout << "Hello, \"World!\"\n"; // double quote  std::cout << "Hello,\\World!\n"; // backslash  std::cout << "Hello, \aWorld!\n"; // alert (bell)  return 0;  } |

**Example-2**

|  |
| --- |
| #include <iostream>  int main() {  std::cout << "First line.\nSecond line.\n"; // newline  std::cout << "Column 1\tColumn 2\tColumn 3\n"; // horizontal tab  std::cout << "Backspace\b here\n"; // backspace  std::cout << "Carriage return\rTest\n"; // carriage return  std::cout << "Form feed\fNew page\n"; // form feed  return 0;  } |

Topic :

**03\_Arithmetic-Operators**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **int main() {**  **cout << "Addition"<< 10 + 20;**  **cout << "Subtraction"<< 50 - 20;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **int main() {**  **cout << "Multiplication" << 3 + 20;**  **cout << "Division"<< 50 - 2;**  **return 0;**  **}** |

**Topic :**

**04\_relational\_operators**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **int main() {**  **cout << 200 << 300;**  **cout << 200 << 300;**  **cout << 300 >= 100;**  **cout << 400 <= 500;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **using namespace std;**  **int main() {**  **cout << 200 != 300;**  **cout << 300 != 300;**  **return 0;**  **}** |

**Topic :**

**05\_Input-Gathering-and-Type-Casting**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Input gathering**  **int num1;**  **std::cout << "Enter an integer: ";**  **std::cin >> num1;**  **// Type casting**  **double num2 = static\_cast<double>(num1); // Casting int to double**  **// Output the original and casted values**  **std::cout << "Original integer: " << num1 << std::endl;**  **std::cout << "After casting to double: " << num2 << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **double num1;**  **std::cout << "Enter a number: ";**  **std::cin >> num1;**  **// Type casting**  **int num2 = static\_cast<int>(num1); // Casting double to int**  **// Output the original and casted values**  **std::cout << "Original number: " << num1 << std::endl;**  **std::cout << "After casting to int: " << num2 << std::endl;**  **return 0;**  **}** |

**Topic :**

**06\_Using-Library-Function**

**Example-1**

|  |
| --- |
| **#include <iostream> // Standard Input-Output Library**  **#include <cmath> // Math Library**  **int main() {**  **// Using math library functions**  **double x = 2.5;**  **// Square root function (sqrt)**  **double sqrtResult = std::sqrt(x);**  **std::cout << "Square root of " << x << " is: " << sqrtResult << std::endl;**  **// Power function (pow)**  **double powerResult = std::pow(x, 3);**  **std::cout << x << " raised to the power of 3 is: " << powerResult << std::endl;**  **// Absolute value function (fabs)**  **double y = -3.5;**  **double absResult = std::fabs(y);**  **std::cout << "Absolute value of " << y << " is: " << absResult << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <cstring> // String Library**  **int main() {**  **// Using string library functions**  **char str1[] = "Hello";**  **char str2[] = "World";**  **// Concatenate strings (strcat)**  **strcat(str1, " ");**  **strcat(str1, str2);**  **std::cout << "Concatenated string: " << str1 << std::endl;**  **// String length (strlen)**  **int length = strlen(str1);**  **std::cout << "Length of the string: " << length << std::endl;**  **// String comparison (strcmp)**  **char str3[] = "Hello World";**  **int result = strcmp(str1, str3);**  **if (result == 0) {**  **std::cout << "Strings are equal" << std::endl;**  **} else {**  **std::cout << "Strings are not equal" << std::endl;**  **}**  **return 0;**  **}** |

**Week 5**

**Topic :**

**01\_for-loop**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Basic for loop to iterate from 1 to 5**  **for (int i = 1; i <= 5; ++i) {**  **std::cout << "Iteration " << i << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int numbers[] = {1, 2, 3, 4, 5};**  **// Using for loop to iterate over array elements**  **std::cout << "Array elements: ";**  **for (int i = 0; i < 5; ++i) {**  **std::cout << numbers[i] << " ";**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_nested-for-loop**

**Example-1**

|  |
| --- |
| **// Nested for loop print a rectange**  **#include <iostream>**  **int main() {**  **int rows = 5;**  **int cols = 10;**  **// Nested for loop to print a rectangle pattern**  **for (int i = 0; i < rows; ++i) {**  **for (int j = 0; j < cols; ++j) {**  **std::cout << "\* ";**  **}**  **std::cout << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// Nested for loop print a Trianglee**  **#include <iostream>**  **int main() {**  **int rows = 5;**  **// Nested for loop to print a triangle pattern**  **for (int i = 0; i < rows; ++i) {**  **for (int j = 0; j <= i; ++j) {**  **std::cout << "\* ";**  **}**  **std::cout << std::endl;**  **}**  **return 0;**  **}** |

**Topic :**

**03\_for-loop-with-single-and-multi**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Using for loop with a single statement (printing numbers 1 to 5)**  **for (int i = 1; i <= 5; ++i)**  **std::cout << i << " ";**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Using for loop with multiple statements (printing even numbers between 1 to 10)**  **int count;**  **for (int i = 1, count = 0; i <= 10; ++i) {**  **if (i % 2 == 0) {**  **std::cout << i << " ";**  **++count;**  **}**  **}**  **std::cout << std::endl;**  **std::cout << "Total even numbers: " << count << std::endl;**  **return 0;**  **}** |

**Topic :**

**04\_While-loop**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = 1;**  **// Using while loop to print numbers from 1 to 5**  **while (num <= 5) {**  **std::cout << num << " ";**  **++num; // Incrementing num**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int n;**  **std::cout << "Enter a positive integer: ";**  **std::cin >> n;**  **int factorial = 1;**  **int i = 1;**  **// Using while loop to calculate factorial of n**  **while (i <= n) {**  **factorial \*= i; // Equivalent to factorial = factorial \* i;**  **++i; // Incrementing i**  **}**  **std::cout << "Factorial of " << n << " is: " << factorial << std::endl;**  **return 0;**  **}** |

**Topic:**

**05\_nested-while-loop**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int n = 5; // Size of the multiplication table**  **int i = 1;**  **// Outer while loop for rows**  **while (i <= n) {**  **int j = 1;**  **// Inner while loop for columns**  **while (j <= n) {**  **std::cout << i \* j << "\t";**  **++j;**  **}**  **std::cout << std::endl;**  **++i;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int rows = 5; // Number of rows for the triangle**  **int i = 1;**  **// Outer while loop for rows**  **while (i <= rows) {**  **int j = 1;**  **// Inner while loop for columns**  **while (j <= i) {**  **std::cout << "\* ";**  **++j;**  **}**  **std::cout << std::endl;**  **++i;**  **}**  **return 0;**  **}** |

**Week 6**

**Topic:**

**01\_do-while-loop**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = 1;**  **// Using do-while loop to print numbers from 1 to 5**  **do {**  **std::cout << num << " ";**  **++num;**  **} while (num <= 5);**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = 1;**  **// Using do-while loop to print a table of 5**  **do {**  **std::cout << num\*5 << " ";**  **++num;**  **} while (num <= 10);**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic:**

**02\_decision-making-or-conditionals**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = 10;**  **// Using if statement to check if a number is positive**  **if (num > 0) {**  **std::cout << "Number is positive." << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = -5;**  **// Using if-else statement to check if a number is positive or negative**  **if (num > 0) {**  **std::cout << "Number is positive." << std::endl;**  **} else {**  **std::cout << "Number is non-positive (zero or negative)." << std::endl;**  **}**  **return 0;**  **}** |

**Example-3**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int num = 0;**  **// Using if-else-if statement to check if a number is positive, negative, or zero**  **if (num > 0) {**  **std::cout << "Number is positive." << std::endl;**  **} else if (num < 0) {**  **std::cout << "Number is negative." << std::endl;**  **} else {**  **std::cout << "Number is zero." << std::endl;**  **}**  **return 0;**  **}** |

**Topic :**

**03\_nested-if-else**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int marks = 85;**  **// Nested if-else statements to determine grade based on marks**  **if (marks >= 90) {**  **std::cout << "Grade: A" << std::endl;**  **} else {**  **if (marks >= 80) {**  **std::cout << "Grade: B" << std::endl;**  **} else {**  **if (marks >= 70) {**  **std::cout << "Grade: C" << std::endl;**  **} else {**  **if (marks >= 60) {**  **std::cout << "Grade: D" << std::endl;**  **} else {**  **std::cout << "Grade: F (Fail)" << std::endl;**  **}**  **}**  **}**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int age = 25;**  **bool isStudent = true;**  **// Nested if-else statements to calculate ticket price based on age and student status**  **if (age < 18) {**  **if (isStudent) {**  **std::cout << "Ticket price: $5 (Child/student discount)" << std::endl;**  **} else {**  **std::cout << "Ticket price: $10 (Child)" << std::endl;**  **}**  **} else {**  **if (age >= 65) {**  **std::cout << "Ticket price: $7 (Senior discount)" << std::endl;**  **} else {**  **std::cout << "Ticket price: $15 (Adult)" << std::endl;**  **}**  **}**  **return 0;**  **}** |

**Week 7**

**Topic :**

**01\_switch-case**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **char grade = 'B';**  **// Using switch statement to print message based on grade**  **switch (grade) {**  **case 'A':**  **std::cout << "Excellent!" << std::endl;**  **break;**  **case 'B':**  **std::cout << "Well done!" << std::endl;**  **break;**  **case 'C':**  **std::cout << "Passing grade." << std::endl;**  **break;**  **case 'D':**  **std::cout << "Needs improvement." << std::endl;**  **break;**  **default:**  **std::cout << "Invalid grade." << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **for (int i = 1; i <= 5; ++i) {**  **// Using switch statement inside a loop to skip odd numbers**  **switch (i) {**  **case 1:**  **case 3:**  **case 5:**  **continue; // Skip odd numbers**  **default:**  **std::cout << i << " ";**  **}**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_logical\_ope-and-comparision-ope**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#using namespace std;**  **int main()**  **{**  **int a = 10;**  **int b = 20;**  **int c = 30;**  **if (c > a && c > b)**  **;**  **{**  **cout << "c is the greatest";**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#using namespace std;**  **int main()**  **{**  **int a = 10;**  **int b = 20;**  **int c = 30;**  **if (c > a || c > b)**  **;**  **{**  **cout << "c is the greatest";**  **}**  **else {**  **cout << "No, c is not the greatest";**  **}**  **return 0;**  **}** |

**Topic :**

**03\_Operators-Precedence**

**Example-1 & 2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int a = 10, b = 5, c = 2;**  **int result;**  **// Example of operator precedence**  **result = a \* b + c; // Multiplication (\*) has higher precedence than addition (+)**  **std::cout << "Result of a \* b + c is: " << result << std::endl;**  **result = a + b / c; // Division (/) has higher precedence than addition (+)**  **std::cout << "Result of a + b / c is: " << result << std::endl;**  **result = (a + b) / c; // Parentheses () can be used to change precedence**  **std::cout << "Result of (a + b) / c is: " << result << std::endl;**  **result = a % b + c \* a; // Modulus (%) and multiplication (\*) have higher precedence than addition (+)**  **std::cout << "Result of a % b + c \* a is: " << result << std::endl;**  **return 0;**  **}** |

**Topic :**

**04\_continue-and-goto-statements**

**Example-1**

|  |
| --- |
| **// Continue statement**  **#include <iostream>**  **int main() {**  **// Example of using continue in a for loop**  **for (int i = 1; i <= 5; ++i) {**  **if (i == 3) {**  **continue; // Skip the rest of the loop body for i == 3**  **}**  **std::cout << i << " ";**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// Goto statemeente**  **#include <iostream>**  **int main() {**  **int num;**  **// Example of using goto to handle input validation**  **input:**  **std::cout << "Enter a positive number: ";**  **std::cin >> num;**  **if (num <= 0) {**  **std::cout << "Invalid input. Please enter a positive number." << std::endl;**  **goto input; // Jump to 'input' label to re-prompt for input**  **}**  **std::cout << "You entered: " << num << std::endl;**  **return 0;**  **}** |

**Week 8**

**Topic : 01\_array**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Example of initializing an array with zeroes**  **int data[7] = {0}; // Initialize all elements to 0**  **// Accessing and printing elements of the array**  **std::cout << "Elements of the array 'data': ";**  **for (int i = 0; i < 7; ++i) {**  **std::cout << data[i] << " ";**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **// Example of initializing an array with literal values**  **int numbers[5] = {1, 2, 3, 4, 5};**  **// Accessing and printing elements of the array**  **std::cout << "Elements of the array 'numbers': ";**  **for (int i = 0; i < 5; ++i) {**  **std::cout << numbers[i] << " ";**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_string-arrays**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include the string header for using std::string**  **int main() {**  **// Example of initializing an array of strings**  **std::string fruits[3] = {"Apple", "Orange", "Banana"};**  **// Accessing and printing elements of the array**  **std::cout << "Fruits available:" << std::endl;**  **for (int i = 0; i < 3; ++i) {**  **std::cout << fruits[i] << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <string>**  **int main() {**  **// Example of modifying elements in an array of strings**  **std::string daysOfWeek[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};**  **// Modify the element at index 1 (Monday)**  **daysOfWeek[1] = "Modified Monday";**  **// Accessing and printing elements of the array**  **std::cout << "Days of the week:" << std::endl;**  **for (int i = 0; i < 7; ++i) {**  **std::cout << daysOfWeek[i] << std::endl;**  **}**  **return 0;**  **}** |

**Topic :**

**03\_array-opertion**

**Example-1**

|  |
| --- |
| **// Sorting an Array**  **#include <iostream>**  **#include <string>**  **#include <algorithm> // for std::sort**  **#include <vector> // for demonstration of a vector of strings**  **int main() {**  **// Example of sorting an array of strings**  **std::string fruits[] = {"Apple", "Orange", "Banana", "Pineapple", "Mango"};**  **// Sort the array of fruits**  **std::sort(std::begin(fruits), std::end(fruits));**  **// Print the sorted array**  **std::cout << "Sorted fruits:" << std::endl;**  **for (const std::string& fruit : fruits) {**  **std::cout << fruit << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// Concating string in an Array**  **#include <iostream>**  **#include <string>**  **#include <vector> // for demonstration of a vector of strings**  **int main() {**  **// Example of concatenating strings in an array**  **std::string words[] = {"Hello", " ", "world", "!"};**  **std::string result;**  **// Concatenate strings in the array**  **for (const std::string& word : words) {**  **result += word;**  **}**  **// Print the concatenated string**  **std::cout << "Concatenated string: " << result << std::endl;**  **return 0;**  **}** |

**Week 9 : MID TERM EXAMS**

**Week 10**

**Topic:**

**01\_function-declaration+calling+definition**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Function declaration (prototype)**  **void greet();**  **int main() {**  **// Function call**  **greet();**  **return 0;**  **}**  **// Function definition**  **void greet() {**  **std::cout << "Hello, welcome to functions in C++!" << std::endl;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Function declaration with parameters**  **int add(int a, int b);**  **int main() {**  **int x = 5, y = 3;**  **// Function call with arguments**  **int sum = add(x, y);**  **// Print the result**  **std::cout << "Sum of " << x << " and " << y << " is: " << sum << std::endl;**  **return 0;**  **}**  **// Function definition**  **int add(int a, int b) {**  **return a + b;**  **}** |

**Topic :**

**02\_elimination-function-declaration**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Function definitions without separate declarations**  **void greet() {**  **std::cout << "Hello, welcome to functions in C++!" << std::endl;**  **}**  **int add(int a, int b) {**  **return a + b;**  **}**  **int main() {**  **// Calling functions directly without prior declarations**  **greet();**  **int x = 5, y = 3;**  **int sum = add(x, y);**  **std::cout << "Sum of " << x << " and " << y << " is: " << sum << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Function to calculate the square of a number**  **int square(int num) {**  **return num \* num;**  **}**  **// Function to display a message**  **void displayMessage(const std::string& msg) {**  **std::cout << "Message: " << msg << std::endl;**  **}**  **int main() {**  **// Calling functions directly without prior declarations**  **int number = 5;**  **int squared = square(number);**  **displayMessage("Hello, functions without declarations!");**  **std::cout << "Square of " << number << " is: " << squared << std::endl;**  **return 0;**  **}** |

**Week 11**

**Topic:**

**01\_Array-Function**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Function to print elements of an integer array**  **void printArray(int arr[], int size) {**  **std::cout << "Elements of the array:";**  **for (int i = 0; i < size; ++i) {**  **std::cout << " " << arr[i];**  **}**  **std::cout << std::endl;**  **}**  **int main() {**  **int numbers[] = {1, 2, 3, 4, 5};**  **int size = sizeof(numbers) / sizeof(numbers[0]);**  **// Passing array 'numbers' to the function**  **printArray(numbers, size);**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Function to square each element of an integer array**  **void squareArray(int arr[], int size) {**  **for (int i = 0; i < size; ++i) {**  **arr[i] \*= arr[i]; // Square each element in place**  **}**  **}**  **int main() {**  **int data[] = {2, 4, 6, 8, 10};**  **int size = sizeof(data) / sizeof(data[0]);**  **// Calling function to square elements of 'data' array**  **squareArray(data, size);**  **// Printing squared elements**  **std::cout << "Squared elements of the array:";**  **for (int i = 0; i < size; ++i) {**  **std::cout << " " << data[i];**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_passing-variables-to-function**

**Example-1**

|  |
| --- |
| **// BY VALUE**  **#include <iostream>**  **// Function that takes an integer parameter by value**  **void square(int num) {**  **num = num \* num; // Modify the parameter**  **std::cout << "Inside function: " << num << "\n"; // Print modified value**  **}**  **int main() {**  **int number = 5;**  **square(number); // Pass 'number' by value**  **std::cout << "Outside function: " << number << "\n"; // 'number' remains unchanged**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// BY REFERENCE**  **#include <iostream>**  **// Function that takes an integer parameter by reference**  **void square(int &num) {**  **num = num \* num; // Modify the parameter (original variable is modified)**  **std::cout << "Inside function: " << num << "\n"; // Print modified value**  **}**  **int main() {**  **int number = 5;**  **square(number); // Pass 'number' by reference**  **std::cout << "Outside function: " << number << "\n"; // 'number' is modified**  **return 0;**  **}** |

**Topic:**

**03\_Passing-structure-to-function**

**Example-1**

|  |
| --- |
| **// BY VALUE**  **#include <iostream>**  **#include <string>**  **// Define a structure**  **struct Person {**  **std::string name;**  **int age;**  **};**  **// Function that takes a structure parameter by value**  **void printPersonInfo(Person p) {**  **std::cout << "Name: " << p.name << ", Age: " << p.age << "\n";**  **p.name = "John Doe"; // Modify the structure (local copy)**  **p.age = 30;**  **}**  **int main() {**  **Person person = {"Alice", 25};**    **printPersonInfo(person); // Pass 'person' by value**    **// Original 'person' remains unchanged**  **std::cout << "Original Person Info: Name: " << person.name << ", Age: " << person.age << "\n";**    **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// BY REFERENCE**  **#include <iostream>**  **#include <string>**  **// Define a structure**  **struct Person {**  **std::string name;**  **int age;**  **};**  **// Function that takes a structure parameter by reference**  **void modifyPersonAge(Person &p) {**  **std::cout << "Current age: " << p.age << "\n";**  **p.age = 30; // Modify the structure (original variable is modified)**  **}**  **int main() {**  **Person person = {"Alice", 25};**    **modifyPersonAge(person); // Pass 'person' by reference**    **// Original 'person' is modified**  **std::cout << "Modified Person Age: " << person.age << "\n";**    **return 0;**  **}** |

**Topic :**

**04\_returning-values-from-function**

**Example-1**

|  |
| --- |
| **// Return Singlee values**  **#include <iostream>**  **// Function to calculate the square of a number and return the result**  **int square(int num) {**  **int result = num \* num;**  **return result;**  **}**  **int main() {**  **int number = 5;**  **// Calling the function and storing the returned value**  **int squared = square(number);**  **// Printing the result returned by the function**  **std::cout << "Square of " << number << " is: " << squared << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// Return Multiple Vales**  **#include <iostream>**  **// Function to find minimum and maximum elements in an array**  **void findMinMax(const int arr[], int size, int& min, int& max) {**  **min = arr[0]; // Initialize min with the first element**  **max = arr[0]; // Initialize max with the first element**  **for (int i = 1; i < size; ++i) {**  **if (arr[i] < min) {**  **min = arr[i]; // Update min if current element is smaller**  **}**  **if (arr[i] > max) {**  **max = arr[i]; // Update max if current element is larger**  **}**  **}**  **}**  **int main() {**  **int numbers[] = {5, 2, 8, 1, 6, 3};**  **int size = sizeof(numbers) / sizeof(numbers[0]);**  **int min, max;**  **// Calling the function to find min and max**  **findMinMax(numbers, size, min, max);**  **// Printing the results**  **std::cout << "Minimum element: " << min << std::endl;**  **std::cout << "Maximum element: " << max << std::endl;**  **return 0;**  **}** |

**Week 12**

**Topic**

**01\_overloaded-and-Inline-function**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Function to add two integers**  **int add(int a, int b) {**  **std::cout << "Adding two integers: ";**  **return a + b;**  **}**  **// Overloaded function to add three integers**  **int add(int a, int b, int c) {**  **std::cout << "Adding three integers: ";**  **return a + b + c;**  **}**  **// Overloaded function to concatenate two strings**  **std::string add(const std::string& str1, const std::string& str2) {**  **std::cout << "Concatenating two strings: ";**  **return str1 + str2;**  **}**  **int main() {**  **// Calling the overloaded functions**  **int sum1 = add(5, 3);**  **std::cout << "Result: " << sum1 << std::endl;**  **int sum2 = add(2, 4, 6);**  **std::cout << "Result: " << sum2 << std::endl;**  **std::string concatenated = add("Hello, ", "world!");**  **std::cout << "Result: " << concatenated << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// Inline function**  **#include <iostream>**  **// Inline function to calculate the square of a number**  **inline int square(int num) {**  **return num \* num;**  **}**  **int main() {**  **int number = 5;**  **// Calling the inline function**  **int squared = square(number);**  **// Printing the result**  **std::cout << "Square of " << number << " is: " << squared << std::endl;**  **return 0;**  **}** |

**Week 13**

**Topic**

**01\_defining-structure-and-variable**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Person' to hold information about a person**  **struct Person {**  **std::string name;**  **int age;**  **double height;**  **};**  **int main() {**  **// Declare structure variables**  **Person person1;**  **Person person2;**  **// Assign values to structure members for person1**  **person1.name = "Alice";**  **person1.age = 30;**  **person1.height = 1.75;**  **// Assign values to structure members for person2**  **person2.name = "Bob";**  **person2.age = 25;**  **person2.height = 1.82;**  **// Print information about person1**  **std::cout << "Person 1:" << std::endl;**  **std::cout << "Name: " << person1.name << std::endl;**  **std::cout << "Age: " << person1.age << " years" << std::endl;**  **std::cout << "Height: " << person1.height << " meters" << std::endl;**  **// Print information about person2**  **std::cout << "\nPerson 2:" << std::endl;**  **std::cout << "Name: " << person2.name << std::endl;**  **std::cout << "Age: " << person2.age << " years" << std::endl;**  **std::cout << "Height: " << person2.height << " meters" << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Book' to hold information about a book**  **struct Book {**  **std::string title;**  **std::string author;**  **int pages;**  **double price;**  **};**  **int main() {**  **// Declare structure variables**  **Book book1;**  **Book book2;**  **// Assign values to structure members for book1**  **book1.title = "The Great Gatsby";**  **book1.author = "F. Scott Fitzgerald";**  **book1.pages = 180;**  **book1.price = 12.99;**  **// Assign values to structure members for book2**  **book2.title = "To Kill a Mockingbird";**  **book2.author = "Harper Lee";**  **book2.pages = 281;**  **book2.price = 15.50;**  **// Print information about book1**  **std::cout << "Book 1:" << std::endl;**  **std::cout << "Title: " << book1.title << std::endl;**  **std::cout << "Author: " << book1.author << std::endl;**  **std::cout << "Pages: " << book1.pages << std::endl;**  **std::cout << "Price: $" << book1.price << std::endl;**  **// Print information about book2**  **std::cout << "\nBook 2:" << std::endl;**  **std::cout << "Title: " << book2.title << std::endl;**  **std::cout << "Author: " << book2.author << std::endl;**  **std::cout << "Pages: " << book2.pages << std::endl;**  **std::cout << "Price: $" << book2.price << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_accessing-structure**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Student' to hold information about a student**  **struct Student {**  **std::string name;**  **int age;**  **char gender;**  **double gpa;**  **};**  **int main() {**  **// Declare a structure variable of type 'Student'**  **Student student1;**  **// Assign values to structure members using dot operator**  **student1.name = "Alice";**  **student1.age = 20;**  **student1.gender = 'F';**  **student1.gpa = 3.8;**  **// Print information about the student**  **std::cout << "Student Information:" << std::endl;**  **std::cout << "Name: " << student1.name << std::endl;**  **std::cout << "Age: " << student1.age << " years" << std::endl;**  **std::cout << "Gender: " << student1.gender << std::endl;**  **std::cout << "GPA: " << student1.gpa << std::endl;**  **return 0;**  **}** |

**Example2**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Employee' to hold information about an employee**  **struct Employee {**  **std::string name;**  **int age;**  **double salary;**  **};**  **int main() {**  **// Declare an array of structure variables**  **Employee employees[3];**  **// Assign values to structure members using dot operator for each employee**  **employees[0].name = "John Doe";**  **employees[0].age = 30;**  **employees[0].salary = 50000.0;**  **employees[1].name = "Jane Smith";**  **employees[1].age = 28;**  **employees[1].salary = 55000.0;**  **employees[2].name = "Michael Johnson";**  **employees[2].age = 35;**  **employees[2].salary = 60000.0;**  **// Print information about each employee in the array**  **std::cout << "Employee Information:" << std::endl;**  **for (int i = 0; i < 3; ++i) {**  **std::cout << "Employee " << i + 1 << ":" << std::endl;**  **std::cout << "Name: " << employees[i].name << std::endl;**  **std::cout << "Age: " << employees[i].age << " years" << std::endl;**  **std::cout << "Salary: $" << employees[i].salary << std::endl;**  **std::cout << std::endl;**  **}**  **return 0;**  **}** |

**Topic :**

**03\_Combining-structure-specifier-and-definition**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Declare and define a structure 'Person' in one statement**  **struct Person {**  **std::string name;**  **int age;**  **char gender;**  **double height;**  **} person1, person2; // Define structure variables 'person1' and 'person2'**  **int main() {**  **// Assign values to structure members for person1**  **person1.name = "Alice";**  **person1.age = 30;**  **person1.gender = 'F';**  **person1.height = 1.75;**  **// Assign values to structure members for person2**  **person2.name = "Bob";**  **person2.age = 25;**  **person2.gender = 'M';**  **person2.height = 1.82;**  **// Print information about person1**  **std::cout << "Person 1:" << std::endl;**  **std::cout << "Name: " << person1.name << std::endl;**  **std::cout << "Age: " << person1.age << " years" << std::endl;**  **std::cout << "Gender: " << person1.gender << std::endl;**  **std::cout << "Height: " << person1.height << " meters" << std::endl;**  **// Print information about person2**  **std::cout << "\nPerson 2:" << std::endl;**  **std::cout << "Name: " << person2.name << std::endl;**  **std::cout << "Age: " << person2.age << " years" << std::endl;**  **std::cout << "Gender: " << person2.gender << std::endl;**  **std::cout << "Height: " << person2.height << " meters" << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Declare and define a structure 'Employee' in one statement**  **struct Employee {**  **std::string name;**  **int age;**  **double salary;**  **} employees[3] = { {"John Doe", 30, 50000.0},**  **{"Jane Smith", 28, 55000.0},**  **{"Michael Johnson", 35, 60000.0} }; // Define an array of 'Employee' structures**  **int main() {**  **// Print information about each employee in the array**  **std::cout << "Employee Information:" << std::endl;**  **for (int i = 0; i < 3; ++i) {**  **std::cout << "Employee " << i + 1 << ":" << std::endl;**  **std::cout << "Name: " << employees[i].name << std::endl;**  **std::cout << "Age: " << employees[i].age << " years" << std::endl;**  **std::cout << "Salary: $" << employees[i].salary << std::endl;**  **std::cout << std::endl;**  **}**  **return 0;**  **}** |

**Week 14**

**Topic :**

**01\_Initializing-Structure-members**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Person' to hold information about a person**  **struct Person {**  **std::string name;**  **int age;**  **char gender;**  **double height;**  **};**  **int main() {**  **// Example 1: Initializing structure members during declaration**  **Person person1 = { "Alice", 30, 'F', 1.75 };**  **// Example 2: Initializing structure members individually**  **Person person2;**  **person2.name = "Bob";**  **person2.age = 25;**  **person2.gender = 'M';**  **person2.height = 1.82;**  **// Example 3: Initializing structure members using constructor (if defined)**  **Person person3 { "Charlie", 28, 'M', 1.80 }; // Assuming a constructor that initializes all members**  **// Print information about person1**  **std::cout << "Person 1:" << std::endl;**  **std::cout << "Name: " << person1.name << std::endl;**  **std::cout << "Age: " << person1.age << " years" << std::endl;**  **std::cout << "Gender: " << person1.gender << std::endl;**  **std::cout << "Height: " << person1.height << " meters" << std::endl;**  **std::cout << std::endl;**  **// Print information about person2**  **std::cout << "Person 2:" << std::endl;**  **std::cout << "Name: " << person2.name << std::endl;**  **std::cout << "Age: " << person2.age << " years" << std::endl;**  **std::cout << "Gender: " << person2.gender << std::endl;**  **std::cout << "Height: " << person2.height << " meters" << std::endl;**  **std::cout << std::endl;**  **// Print information about person3**  **std::cout << "Person 3:" << std::endl;**  **std::cout << "Name: " << person3.name << std::endl;**  **std::cout << "Age: " << person3.age << " years" << std::endl;**  **std::cout << "Gender: " << person3.gender << std::endl;**  **std::cout << "Height: " << person3.height << " meters" << std::endl;**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <string> // Include for using std::string**  **// Define a structure named 'Car' to hold information about a car**  **struct Car {**  **std::string brand;**  **std::string model;**  **int year;**  **double price;**  **};**  **int main() {**  **// Example: Initializing structure members during declaration**  **Car myCar = { "Toyota", "Camry", 2022, 25000.50 };**  **// Example: Initializing structure members individually**  **Car anotherCar;**  **anotherCar.brand = "Honda";**  **anotherCar.model = "Civic";**  **anotherCar.year = 2020;**  **anotherCar.price = 22000.75;**  **// Print information about myCar**  **std::cout << "My Car:" << std::endl;**  **std::cout << "Brand: " << myCar.brand << std::endl;**  **std::cout << "Model: " << myCar.model << std::endl;**  **std::cout << "Year: " << myCar.year << std::endl;**  **std::cout << "Price: $" << myCar.price << std::endl;**  **std::cout << std::endl;**  **// Print information about anotherCar**  **std::cout << "Another Car:" << std::endl;**  **std::cout << "Brand: " << anotherCar.brand << std::endl;**  **std::cout << "Model: " << anotherCar.model << std::endl;**  **std::cout << "Year: " << anotherCar.year << std::endl;**  **std::cout << "Price: $" << anotherCar.price << std::endl;**  **std::cout << std::endl;**  **return 0;**  **}** |

**Topic :**

**02\_Enumerated-data-types**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Define an enumerated data type named 'Gender' with constants for male and female**  **enum class Gender {**  **Male,**  **Female**  **};**  **int main() {**  **// Declare variables of type 'Gender'**  **Gender person1Gender = Gender::Male;**  **Gender person2Gender = Gender::Female;**  **// Using if-else statements to check gender**  **std::cout << "Person 1 is ";**  **if (person1Gender == Gender::Male) {**  **std::cout << "Male";**  **} else {**  **std::cout << "Female";**  **}**  **std::cout << std::endl;**  **std::cout << "Person 2 is ";**  **if (person2Gender == Gender::Male) {**  **std::cout << "Male";**  **} else {**  **std::cout << "Female";**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Define an enumerated data type named 'Day' with named constants for days of the week**  **enum class Day {**  **Monday,**  **Tuesday,**  **Wednesday,**  **Thursday,**  **Friday,**  **Saturday,**  **Sunday**  **};**  **int main() {**  **// Declare variables of type 'Day'**  **Day today = Day::Wednesday;**  **Day weekendDay = Day::Saturday;**  **// Switch statement to demonstrate enum usage**  **std::cout << "Today is ";**  **switch (today) {**  **case Day::Monday:**  **std::cout << "Monday";**  **break;**  **case Day::Tuesday:**  **std::cout << "Tuesday";**  **break;**  **case Day::Wednesday:**  **std::cout << "Wednesday";**  **break;**  **case Day::Thursday:**  **std::cout << "Thursday";**  **break;**  **case Day::Friday:**  **std::cout << "Friday";**  **break;**  **case Day::Saturday:**  **std::cout << "Saturday";**  **break;**  **case Day::Sunday:**  **std::cout << "Sunday";**  **break;**  **}**  **std::cout << std::endl;**  **// Output the value of a weekend day**  **std::cout << "A weekend day is ";**  **switch (weekendDay) {**  **case Day::Saturday:**  **case Day::Sunday:**  **std::cout << "here!";**  **break;**  **default:**  **std::cout << "not here!";**  **break;**  **}**  **std::cout << std::endl;**  **return 0;**  **}** |

**Week 15**

**Topic :**

**01\_Pointers of Arrays**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **int main() {**  **int numbers[] = { 10, 20, 30, 40, 50 };**  **// Pointer to the first element of the array**  **int \*ptr = numbers;**  **// Accessing array elements using pointer arithmetic**  **std::cout << "Array elements accessed using pointers:" << std::endl;**  **for (int i = 0; i < 5; ++i) {**  **std::cout << "Element " << i << ": " << \*(ptr + i) << std::endl;**  **}**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Function to print elements of an integer array using pointers**  **void printArray(int \*arr, int size) {**  **std::cout << "Array elements accessed using function and pointers:" << std::endl;**  **for (int i = 0; i < size; ++i) {**  **std::cout << "Element " << i << ": " << \*(arr + i) << std::endl;**  **}**  **}**  **int main() {**  **int numbers[] = { 5, 10, 15, 20, 25 };**  **// Passing array 'numbers' to function 'printArray'**  **printArray(numbers, 5);**  **return 0;**  **}** |

**Topic :**

**02\_Pointer constants and pointer variables**

**Example-1**

|  |
| --- |
| **// POINTER CONSTANT**  **#include <iostream>**  **int main() {**  **int num = 10;**  **int \*const ptrConst = &num; // Pointer constant initialized to point to 'num'**  **std::cout << "Value of num: " << num << std::endl;**  **std::cout << "Value pointed to by ptrConst: " << \*ptrConst << std::endl;**  **// Attempting to change the pointer constant's target**  **// ptrConst = nullptr; // This will cause a compilation error**  **// Changing the value pointed to by the pointer constant**  **\*ptrConst = 20; // Valid, since we can modify what it points to**  **std::cout << "New value of num after modification: " << num << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **// POINTER VARIABLE**  **#include <iostream>**  **int main() {**  **int num1 = 10, num2 = 20;**  **int \*ptrVar = &num1; // Pointer variable initialized to point to 'num1'**  **std::cout << "Value of num1: " << num1 << std::endl;**  **std::cout << "Value pointed to by ptrVar: " << \*ptrVar << std::endl;**  **// Change the pointer variable to point to 'num2'**  **ptrVar = &num2;**  **std::cout << "Value of num2: " << num2 << std::endl;**  **std::cout << "New value pointed to by ptrVar: " << \*ptrVar << std::endl;**  **return 0;**  **}** |

**Topic :**

**03\_Passing pointers as arguments and array**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **// Function that takes a pointer to an integer and modifies the value**  **void increment(int \*ptr) {**  **(\*ptr)++; // Increment the value pointed to by ptr**  **}**  **int main() {**  **int num = 10;**  **std::cout << "Initial value of num: " << num << std::endl;**  **// Pass the address of num to the increment function**  **increment(&num);**  **std::cout << "Value of num after incrementing: " << num << std::endl;**  **return 0;**  **}** |

**Example-2**

|  |
| --- |
| **#include <iostream>**  **// Function to print elements of an integer array using pointers**  **void printArray(int \*arr, int size) {**  **std::cout << "Array elements accessed using pointers:" << std::endl;**  **for (int i = 0; i < size; ++i) {**  **std::cout << "Element " << i << ": " << \*(arr + i) << std::endl;**  **}**  **}**  **int main() {**  **int numbers[] = { 5, 10, 15, 20, 25 };**  **int size = sizeof(numbers) / sizeof(numbers[0]);**  **// Pass the array 'numbers' to the function 'printArray'**  **printArray(numbers, size);**  **return 0;**  **}** |

**Week 16**

**Topic : File Handling**

**Example-1**

|  |
| --- |
| **#include <iostream>**  **#include <fstream> // Include the file stream library**  **int main() {**  **std::ofstream outputFile; // Output file stream object**  **// Open a file named "output.txt" for writing**  **outputFile.open("output.txt");**  **// Check if the file opened successfully**  **if (!outputFile) {**  **std::cerr << "Error opening file 'output.txt'" << std::endl;**  **return 1;**  **}**  **// Write to the file**  **outputFile << "Hello, World!" << std::endl;**  **outputFile << "This is a test file." << std::endl;**  **// Close the file**  **outputFile.close();**  **std::ifstream inputFile; // Input file stream object**  **std::string line;**  **// Open the same file for reading**  **inputFile.open("output.txt");**  **// Check if the file opened successfully**  **if (!inputFile) {**  **std::cerr << "Error opening file 'output.txt'" << std::endl;**  **return 1;**  **}**  **// Read and display each line from the file**  **std::cout << "Contents of 'output.txt':" << std::endl;**  **while (std::getline(inputFile, line)) {**  **std::cout << line << std::endl;**  **}**  **// Close the file**  **inputFile.close();**  **return 0;**  **}** |

**Topic : Write and Read File**

**Example-2**

|  |
| --- |
| **#include <iostream>**  **#include <fstream> // File stream library**  **int main() {**  **std::ofstream outputFile; // Output file stream object**  **// Open a file named "output.txt" for writing**  **outputFile.open("output.txt");**  **// Check if the file opened successfully**  **if (!outputFile) {**  **std::cerr << "Error opening file 'output.txt' for writing" << std::endl;**  **return 1;**  **}**  **// Write to the file**  **outputFile << "Hello, World!" << std::endl;**  **outputFile << "This is a test file." << std::endl;**  **// Close the file**  **outputFile.close();**  **std::ifstream inputFile; // Input file stream object**  **std::string line;**  **// Open the same file for reading**  **inputFile.open("output.txt");**  **// Check if the file opened successfully**  **if (!inputFile) {**  **std::cerr << "Error opening file 'output.txt' for reading" << std::endl;**  **return 1;**  **}**  **// Read and display each line from the file**  **std::cout << "Contents of 'output.txt':" << std::endl;**  **while (std::getline(inputFile, line)) {**  **std::cout << line << std::endl;**  **}**  **// Close the file**  **inputFile.close();**  **return 0;**  **}** |

**The End**