Assignment 1(practical)

23.Write a C++ program to demonstrate the use of const\_cast by modifying a constant variable

#include <iostream>

using namespace std;

int main() {

const int x = 10;

int\* y = const\_cast<int\*>(&x);

\*y = 20;

cout << "Modified value: " << \*y << endl;

return 0;

}

24.Write a C++ program to demonstrate the use of reinterpret\_cast to convert an int to a char

#include <iostream>

using namespace std;

int main() {

int x = 65;

char c = reinterpret\_cast<char&>(x);

cout << "Char: " << c << endl;

return 0;

}

25.write a C++ program to demonstrate the use of C-style casting by converting an int to a double.

#include <iostream>

using namespace std;

int main() {

int x = 5;

double d = (double)x;

cout << "Double: " << d << endl;

return 0;

}

26.Write a C++ program to print the numbers from 1 to 10 using a for loop.

for (int i = 1; i <= 10; i++) cout << i << " ";

27.Write a C++ program to calculate the sum of the first 50 natural numbers using a while loop.

int i = 1, sum = 0;

while (i <= 50) sum += i++;

cout << "Sum: " << sum;

28.Write a C++ program to print the multiplication table of a given number using a do-while loop

int n, i = 1;

cin >> n;

do cout << n << " x " << i << " = " << n\*i << endl; while (++i <= 10);

29.Write a C++ program to find the factorial of a given number using a for loop.

int n, f = 1;

cin >> n;

for (int i = 1; i <= n; i++) f \*= i;

cout << "Factorial: " << f;

30.Write a C++ program to reverse a given number using a while loop

int n, rev = 0;

cin >> n;

while (n) rev = rev \* 10 + n % 10, n /= 10;

cout << "Reversed: " << rev;

31.Write a C++ program to check if a given number is a prime number using a for loop.

int n, i;

cin >> n;

for (i = 2; i < n; i++) if (n % i == 0) break;

cout << (i == n ? "Prime" : "Not Prime");

32.Write a C++ program to print the Fibonacci series up to a given number of terms using a for loop

int n, a = 0, b = 1;

cin >> n;

for (int i = 0; i < n; i++) cout << a << " ", tie(a, b) = make\_pair(b, a + b);

33.Write a C++ program to count the number of digits in a given number using a while loop.

int n, c = 0;

cin >> n;

while (n) n /= 10, c++;

cout << "Digits: " << c;

34.Write a C++ program to print the sum of all even numbers between 1 and 100 using a for loop.

int sum = 0;

for (int i = 2; i <= 100; i += 2) sum += i;

cout << "Sum: " << sum;

35.Write a C++ program to find the largest number in an array using a for loop

#include <iostream>

using namespace std;

int main() {

int arr[] = {5, 8, 12, 3, 15}, max = arr[0];

for(int i = 1; i < 5; i++) if(arr[i] > max) max = arr[i];

cout << "Largest: " << max << endl;

return 0;

}

36.Write a C++ program to print the elements of an array in reverse order using a for loop.

#include <iostream>

using namespace std;

int main() {

int arr[] = {10, 20, 30, 40, 50};

for(int i = 4; i >= 0; i--) cout << arr[i] << " ";

return 0;

}

37.Write a C++ program to calculate the average of numbers in an array using a for loop.

#include <iostream>

using namespace std;

int main() {

int arr[] = {4, 8, 6, 10, 12}, sum = 0;

for(int i = 0; i < 5; i++) sum += arr[i];

cout << "Average: " << sum / 5.0 << endl;

return 0;

}

38. Write a C++ program to find the GCD (Greatest Common Divisor) of two numbers using a while loop.

#include <iostream>

using namespace std;

int main() {

int a = 48, b = 18;

while(a != b) {

if(a > b) a -= b;

else b -= a;

}

cout << "GCD: " << a << endl;

return 0;

}

39. Write a C++ program to print all the prime numbers between 1 and 100 using a for loop.

#include <iostream>

using namespace std;

int main() {

for(int i = 2; i <= 100; i++) {

bool prime = true;

for(int j = 2; j \* j <= i; j++) {

if(i % j == 0) {

prime = false;

break;

}

}

if(prime) cout << i << " ";

}

return 0;

}

40. Write a C++ program to find the sum of the digits of a given number using a while loop

#include <iostream>

using namespace std;

int main() {

int num = 1234, sum = 0;

while(num > 0) {

sum += num % 10;

num /= 10;

}

cout << "Sum of digits: " << sum << endl;

return 0;

}

41. Write a C++ program to check if a given number is a palindrome using a while loop

#include <iostream>

using namespace std;

int main() {

int num = 121, rev = 0, temp = num;

while(temp > 0) {

rev = rev \* 10 + temp % 10;

temp /= 10;

}

if(num == rev) cout << "Palindrome" << endl;

else cout << "Not a palindrome" << endl;

return 0;

}

42. Write a C++ program to find the smallest number in an array using a for loop

#include <iostream>

using namespace std;

int main() {

int arr[] = {12, 3, 7, 2, 9}, min = arr[0];

for(int i = 1; i < 5; i++) if(arr[i] < min) min = arr[i];

cout << "Smallest: " << min << endl;

return 0;

}

43. Write a C++ program to print the first N even numbers using a for loop. write all these codes in series wise

#include <iostream>

using namespace std;

int main() {

int N = 10;

for(int i = 1; i <= N; i++) cout << 2 \* i << " ";

return 0;

}

44. Write a C++ program to print a pattern using nested for loops.

#include <iostream>

using namespace std;

int main() {

int rows = 5;

for(int i = 1; i <= rows; i++) {

for(int j = 1; j <= i; j++) cout << "\* ";

cout << endl;

}

return 0;

}

45. Write a C++ program to print the numbers from 1 to 100, but replace multiples of 3 with "Fizz" and multiples of 5 with "Buzz" using a for loop.

#include <iostream>

using namespace std;

int main() {

for(int i = 1; i <= 100; i++) {

if(i % 3 == 0 && i % 5 == 0) cout << "FizzBuzz\n";

else if(i % 3 == 0) cout << "Fizz\n";

else if(i % 5 == 0) cout << "Buzz\n";

else cout << i << endl;

}

return 0;

}

46. Write a C++ program to calculate the sum of digits of a given number using a while loop.

#include <iostream>

using namespace std;

int main() {

int num = 5678, sum = 0;

while(num > 0) {

sum += num % 10;

num /= 10;

}

cout << "Sum of digits: " << sum << endl;

return 0;

}

47. Write a C++ program to reverse a given string using a for loop.

#include <iostream>

#include <string>

using namespace std;

int main() {

string str = "Hello", rev = "";

for(int i = str.length() - 1; i >= 0; i--) rev += str[i];

cout << "Reversed string: " << rev << endl;

return 0;

}

48. Write a C++ program to check if a given number is an Armstrong number using a while loop.

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int num = 153, sum = 0, temp = num, digits = 0;

while(temp > 0) {

temp /= 10;

digits++;

}

temp = num;

while(temp > 0) {

int d = temp % 10;

sum += pow(d, digits);

temp /= 10;

}

if(sum == num) cout << "Armstrong number" << endl;

else cout << "Not an Armstrong number" << endl;

return 0;

}

49. Write a C++ program to find the sum of all elements in an array using a for loop.

#include <iostream>

using namespace std;

int main() {

int arr[] = {3, 5, 7, 9, 11}, sum = 0;

for(int i = 0; i < 5; i++) sum += arr[i];

cout << "Sum of elements: " << sum << endl;

return 0;

}

50. Write a C++ program to check if a given number is a palindrome using a while loop. write all these codes in series wise

#include <iostream>

using namespace std;

int main() {

int num = 12321, rev = 0, temp = num;

while(temp > 0) {

rev = rev \* 10 + temp % 10;

temp /= 10;

}

if(num == rev) cout << "Palindrome" << endl;

else cout << "Not a palindrome" << endl;

return 0;

}

51. Write a C++ program to find the sum of all even and odd numbers between 1 and 100 using separate for loops

#include <iostream>

using namespace std;

int main() {

int evenSum = 0, oddSum = 0;

for(int i = 2; i <= 100; i += 2) evenSum += i;

for(int i = 1; i < 100; i += 2) oddSum += i;

cout << "Even Sum: " << evenSum << "\nOdd Sum: " << oddSum << endl;

return 0;

}

52. Write a C++ program to count the number of vowels and consonants in a given string using a for loop.

#include <iostream>

#include <cctype>

using namespace std;

int main() {

string str = "Hello World!";

int vowels = 0, consonants = 0;

for(char c : str) {

if(isalpha(c)) {

char ch = tolower(c);

if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') vowels++;

else consonants++;

}

}

cout << "Vowels: " << vowels << ", Consonants: " << consonants << endl;

return 0;

}

53. Write a C++ program to convert a binary number to a decimal number using a while loop.

#include <iostream>

using namespace std;

int main() {

int binary = 1101, decimal = 0, base = 1;

while(binary > 0) {

int rem = binary % 10;

decimal += rem \* base;

binary /= 10;

base \*= 2;

}

cout << "Decimal: " << decimal << endl;

return 0;

}

54. Write a C++ program to print all prime numbers between two given numbers using a for loop.

#include <iostream>

using namespace std;

int main() {

int start = 10, end = 50;

for(int i = start; i <= end; i++) {

bool isPrime = i > 1;

for(int j = 2; j \* j <= i; j++) {

if(i % j == 0) {

isPrime = false;

break;

}

}

if(isPrime) cout << i << " ";

}

return 0;

}

55. Write a C++ program to find the transpose of a given matrix using nested for loops.

#include <iostream>

using namespace std;

int main() {

int mat[2][3] = {{1, 2, 3}, {4, 5, 6}};

int trans[3][2];

for(int i = 0; i < 2; i++)

for(int j = 0; j < 3; j++)

trans[j][i] = mat[i][j];

cout << "Transpose:\n";

for(int i = 0; i < 3; i++) {

for(int j = 0; j < 2; j++) cout << trans[i][j] << " ";

cout << endl;

}

return 0;

}

56. Write a C++ program to calculate the sum of the harmonic series up to n terms using a for loop.

#include <iostream>

using namespace std;

int main() {

int n = 5;

double sum = 0.0;

for(int i = 1; i <= n; i++) sum += 1.0 / i;

cout << "Harmonic Sum: " << sum << endl;

return 0;

}

57. Write a C++ program to find duplicate elements in an array using nested for loops.

#include <iostream>

using namespace std;

int main() {

int arr[] = {4, 3, 2, 4, 1, 3}, n = 6;

cout << "Duplicates: ";

for(int i = 0; i < n; i++) {

for(int j = i + 1; j < n; j++) {

if(arr[i] == arr[j]) {

cout << arr[i] << " ";

break;

}

}

}

return 0;

}

58. Write a C++ program to print a pyramid pattern using nested for loops.

#include <iostream>

using namespace std;

int main() {

int rows = 5;

for(int i = 1; i <= rows; i++) {

for(int j = i; j < rows; j++) cout << " ";

for(int k = 1; k <= (2 \* i - 1); k++) cout << "\*";

cout << endl;

}

return 0;

}

59. Write a C++ program to find the LCM (Least Common Multiple) of two numbers using a while loop.

#include <iostream>

using namespace std;

int main() {

int a = 12, b = 18, lcm;

lcm = (a > b) ? a : b;

while(true) {

if(lcm % a == 0 && lcm % b == 0) break;

lcm++;

}

cout << "LCM: " << lcm << endl;

return 0;

}

60. Write a C++ program to generate the Fibonacci series up to n terms using a for loop

#include <iostream>

using namespace std;

int main() {

int n = 10, a = 0, b = 1;

cout << a << " " << b << " ";

for(int i = 2; i < n; i++) {

int c = a + b;

cout << c << " ";

a = b;

b = c;

}

return 0;

}

61. Write a C++ program to add two matrices using nested for loops

#include <iostream>

using namespace std;

int main() {

int a[2][2] = {{1, 2}, {3, 4}}, b[2][2] = {{5, 6}, {7, 8}}, sum[2][2];

for(int i = 0; i < 2; i++)

for(int j = 0; j < 2; j++)

sum[i][j] = a[i][j] + b[i][j];

cout << "Matrix Sum:\n";

for(int i = 0; i < 2; i++) {

for(int j = 0; j < 2; j++) cout << sum[i][j] << " ";

cout << endl

62. Write a C++ program to create a number guessing game where the user has to guess a number between 1 and 100 using a while loop.

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main() {

srand(time(0)); // Seed for random number

int secret = rand() % 100 + 1; // Random number between 1 and 100

int guess;

cout << "Guess a number between 1 and 100:\n";

while (true) {

cin >> guess;

if (guess == secret) {

cout << "Congratulations! You guessed it right.\n";

break;

} else if (guess < secret) {

cout << "Too low. Try again:\n";

} else {

cout << "Too high. Try again:\n";

}

}

return 0;

}

// 63. Count frequency of each character in a string using for loop

#include <iostream>

#include <map>

using namespace std;

int main() {

string str = "hello world";

map<char, int> freq;

for (char c : str) {

if (c != ' ') freq[c]++;

}

cout << "Character Frequencies:\n";

for (auto pair : freq) {

cout << pair.first << ": " << pair.second << endl;

}

return 0;

}

63. Write a C++ program to count the frequency of each character in a string using a for loop.

#include <iostream>

#include <map>

using namespace std;

int main() {

string str = "hello world";

map<char, int> freq;

for (char c : str) {

if (c != ' ') freq[c]++;

}

cout << "Character Frequencies:\n";

for (auto pair : freq) {

cout << pair.first << ": " << pair.second << endl;

}

return 0;

}

64. Write a C++ program to convert a decimal number to a binary number using a while loop

#include <iostream>

#include <stack>

using namespace std

int main() {

int decimal = 13;

stack<int> binary;

while (decimal > 0) {

binary.push(decimal % 2);

decimal /= 2;

}

cout << "Binary: ";

while (!binary.empty()) {

cout << binary.top();

binary.pop();

}

cout << endl;

return 0;

}

Assignment 2(practical)

### **26. Hello World Program**

#include <iostream>

int main() {

std::cout << "Hello, World!" << std::endl;

return 0;

}

### **27. Sum of Two Integers from Command Line**

#include <iostream>

#include <cstdlib>

int main(int argc, char\* argv[]) {

if (argc != 3) {

std::cout << "Usage: <program> num1 num2" << std::endl;

return 1;

}

int a = std::atoi(argv[1]);

int b = std::atoi(argv[2]);

std::cout << "Sum: " << (a + b) << std::endl;

return 0;

}

### **28. Function Prototype for Factorial**

int factorial(int n);

### **29. Max of Three Numbers (with Function Prototype)**

#include <iostream>

int maxOf hree(int a, int b, int c);

int main() {

std::cout << "Max: " << maxOfThree(5, 12, 7) << std::endl;

return 0;

}

int maxOfThree(int a, int b, int c) {

return (a > b) ? (a > c ? a : c) : (b > c ? b : c);

}

### **30. Area of a Circle**

#include <iostream>

#define PI 3.14159

double areaOfCircle(double radius) {

return PI \* radius \* radius;

}

### **31. Product of Two Integers**

int product(int a, int b) {

return a \* b;

}

### **32. Print Array Elements**

#include <iostream>

void printArray(int arr[], int size) {

for (int i = 0; i < size; ++i)

std::cout << arr[i] << " ";

std::cout << std::endl;

}

### **33. Swap Two Integers (Call by Reference)**

void swap(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

### **34. Increment by 10 (Call by Reference)**

void incrementByTen(int &num) {

num += 10;

}

### **35. Modify Array Elements (Call by Reference)**

void modifyArray(int arr[], int size) {

for (int i = 0; i < size; ++i)

arr[i] += 1;

}

### **36. Inline Function to Calculate Square**

inline int square(int x) {

return x \* x;

}

### **37. Inline Function to Calculate Cube**

inline int cube(int x) {

return x \* x \* x;

}

### **38. Inline Function to Sum Two Numbers**

#include <iostream>

inline int sum(int a, int b) {

return a + b;

}

int main() {

std::cout << "Sum: " << sum(5, 6) << std::endl;

return 0;

}

### **39. Macro vs Inline for Square**

#include <iostream>

#define SQUARE(x) ((x)\*(x))

inline int squareInline(int x) {

return x \* x;

}

int main() {

int a = 5;

std::cout << "Macro Square: " << SQUARE(a) << std::endl;

std::cout << "Inline Square: " << squareInline(a) << std::endl;

return 0;

}

### **40. Macro vs Inline for Max**

#include <iostream>

#define MAX(a, b) ((a) > (b) ? (a) : (b))

inline int maxInline(int a, int b) {

return (a > b) ? a : b;

}

int main() {

std::cout << "Macro Max: " << MAX(3, 7) << std::endl;

std::cout << "Inline Max: " << maxInline(3, 7) << std::endl;

return 0;

}

### **41. Overloaded Area Functions**

#include <iostream>

double area(double r) {

return 3.14159 \* r \* r;

}

double area(double l, double w) {

return l \* w;

}

double area(double b, double h, bool triangle) {

return 0.5 \* b \* h;

}

### **42. Overloaded Max Functions**

int max(int a, int b) {

return (a > b) ? a : b;

}

int max(int a, int b, int c) {

return std::max(a, std::max(b, c));

}

### **43. Overloaded Print Functions**

#include <iostream>

void print(int x) {

std::cout << "Int: " << x << std::endl;

}

void print(float x) {

std::cout << "Float: " << x << std::endl;

}

void print(std::string s) {

std::cout << "String: " << s << std::endl;

}

### **44. Compound Interest with Default Arguments**

double compoundInterest(double principal, double rate = 5.0, int time = 2) {

return principal \* pow((1 + rate / 100), time);

}

### **45. Greeting Function with Default Argument**

#include <iostream>

void greet(std::string name = "Guest") {

std::cout << "Hello, " << name << "!" << std::endl;

}

### **46. Power with Default Exponent**

double power(double base, int exponent = 2) {

return pow(base, exponent);

}

### **47. Recursive Function Example**t

#include <iostream>

int factorial(int n) {

if (n <= 1) return 1;

return n \* factorial(n - 1);

}

### **48. Array of Function Pointers**

#include <iostream>

int add(int a, int b) { return a + b; }

int sub(int a, int b) { return a - b; }

int main() {

int (\*operations[2])(int, int) = {add, sub};

std::cout << "Add: " << operations[0](5, 3) << std::endl;

std::cout << "Sub: " << operations[1](5, 3) << std::endl;

return 0;

}

### **49. Function Template Example**

#include <iostream>

template <typename T>

T max(T a, T b) {

return (a > b) ? a : b;

}

### **50. Function Pointer and Callback Example**

#include <iostream>

void display(int result) {

std::cout << "Result: " << result << std::endl;

}

void compute(int a, int b, void(\*callback)(int)) {

int res = a + b;

callback(res);

}

int main() {

compute(4, 5, display);

return 0;

}

Assignment3(practcal)

**54. Class Person with Private Attributes**

class Person {

private:

string name;

int age;

public:

void setDetails(string n, int a) {

name = n;

age = a;

}

void getDetails() {

cout << "Name: " << name << ", Age: " << age << endl;

}

};

**55. Student Inheriting Person**

class Student : public Person {

private:

int studentID;

public:

void setStudentID(int id) { studentID = id; }

void getStudentID() { cout << "Student ID: " << studentID << endl; }

};

**56. Car Class**

class Car {

public:

string make, model;

int year;

void set(string m, string mo, int y) {

make = m; model = mo; year = y;

}

void display() {

cout << year << " " << make << " " << model << endl;

}

};

**57. Array of Car Objects**

int main() {

Car cars[2];

cars[0].set("Toyota", "Corolla", 2020);

cars[1].set("Ford", "Focus", 2021);

for (int i = 0; i < 2; i++) cars[i].display();

return 0;

}

**58. BankAccount Class**

class BankAccount {

private:

string accountNumber;

double balance;

public:

void deposit(double amount) { balance += amount; }

void withdraw(double amount) { if (balance >= amount) balance -= amount; }

void showBalance() { cout << "Balance: " << balance << endl; }

};

**59. Rectangle Class**

cpp

CopyEdit

class Rectangle {

private:

double length, width;

public:

void set(double l, double w) { length = l; width = w; }

double area() { return length \* width; }

double perimeter() { return 2 \* (length + width); }

};

**60. Employee Class**

class Employee {

private:

string name, position;

double salary;

public:

void set(string n, string p, double s) {

name = n; position = p; salary = s;

}

void display() {

cout << "Name: " << name << ", Position: " << position << ", Salary: " << salary << endl;

}

};

**61. Counter Class with Static Member**

class Counter {

static int count;

public:

Counter() { count++; }

static void showCount() { cout << "Objects created: " << count << endl; }

};

int Counter::count = 0;

**62. Math Class with Static Methods**

class Math {

public:

static int add(int a, int b) { return a + b; }

static int sub(int a, int b) { return a - b; }

};

**63. Student Class with Static Count**

class Student {

static int total;

public:

Student() { total++; }

static void showTotal() { cout << "Total Students: " << total << endl; }

};

int Student::total = 0;

**64. Book with Parameterized Constructor**

class Book {

string title;

int pages;

public:

Book(string t, int p) : title(t), pages(p) {}

void display() {

cout << "Title: " << title << ", Pages: " << pages << endl;

}

};

**65. Point Constructors**

class Point {

int x, y;

public:

Point() : x(0), y(0) {}

Point(int a, int b) : x(a), y(b) {}

Point(const Point &p) : x(p.x), y(p.y) {}

};

**66. Matrix with Parameterized Constructor**

class Matrix {

int mat[2][2];

public:

Matrix(int a[2][2]) {

for (int i = 0; i < 2; i++)

for (int j = 0; j < 2; j++)

mat[i][j] = a[i][j];

}

};

**67. FileHandler with Destructor**

#include <fstream>

class FileHandler {

fstream file;

public:

FileHandler(string name) {

file.open(name, ios::out);

}

~FileHandler() {

file.close();

cout << "File closed\n";

}

};

**68. DynamicArray with Destructor**

class DynamicArray {

int\* arr;

public:

DynamicArray(int size) {

arr = new int[size];

}

~DynamicArray() {

delete[] arr;

cout << "Memory deallocated\n";

}

};

**69. Logger with Destructor**

class Logger {

public:

~Logger() {

cout << "Logger destroyed\n";

}

};

**70. Operator Overloading "+" for Complex**

class Complex {

int real, imag;

public:

Complex(int r, int i) : real(r), imag(i) {}

Complex operator+(Complex c) {

return Complex(real + c.real, imag + c.imag);

}

};

**71. Overload ">>" and "<<" for Fraction**

class Fraction {

int num, den;

public:

friend istream& operator>>(istream& in, Fraction& f) {

in >> f.num >> f.den;

return in;

}

friend ostream& operator<<(ostream& out, Fraction f) {

out << f.num << "/" << f.den;

return out;

}

};

**72. Overload "==" for Date**

class Date {

int d, m, y;

public:

Date(int day, int mon, int yr) : d(day), m(mon), y(yr) {}

bool operator==(Date date) {

return (d == date.d && m == date.m && y == date.y);

}

};

**73. Overload "[]" for Vector**

class Vector {

int arr[10];

public:

int& operator[](int index) {

return arr[index];

}

};

**74. Box with Friend Function**

class Box {

int length, width, height;

public:

Box(int l, int w, int h) : length(l), width(w), height(h) {}

friend int volume(Box b);

};

int volume(Box b) {

return b.length \* b.width \* b.height;

}

**75. Circle with Friend Function**

class Circle {

double radius;

public:

Circle(double r) : radius(r) {}

friend double area(Circle c);

};

double area(Circle c) {

return 3.14 \* c.radius \* c.radius;

}

**76. Distance with Friend Function**

class Distance {

int feet, inches;

public:

Distance(int f, int i) : feet(f), inches(i) {}

friend Distance add(Distance a, Distance b);

} ;

Distance add(Distance a, Distance b) {

int in = a.inches + b.inches;

int ft = a.feet + b.feet + (in / 12);

in %= 12;

return Distance(ft, in);

}

**77. Shape with Derived Classes**

class Shape {

public:

virtual void draw() { cout << "Shape\n"; }

};

class Circle : public Shape {

public:

void draw() override { cout << "Circle\n"; }

};

class Rectangle : public Shape {

public:

void draw() override { cout << "Rectangle\n"; }

};

class Triangle : public Shape {

public:

void draw() override { cout << "Triangle\n"; }

};

**78. Animal with Derived Classes**

class Animal {

public:

virtual void sound() { cout << "Animal sound\n"; }

};

class Dog : public Animal {

public:

void sound() override { cout << "Bark\n"; }

};

class Cat : public Animal {

public:

void sound() override { cout << "Meow\n"; }

};

class Bird : public Animal {

public:

void sound() override { cout << "Chirp\n"; }

};

**79. Vehicle with Car and Bike**

class Vehicle {

public:

virtual void move() { cout << "Vehicle moving\n"; }

};

class Car : public Vehicle {

public:

void move() override { cout << "Car driving\n"; }

};

class Bike : public Vehicle {

public:

void move() override { cout << "Bike riding\n"; }

};

**80. Single Inheritance**

class Person {

public:

void intro() { cout << "I am a person\n"; }

};

class Employee : public Person {

public:

void job() { cout << "I am an employee\n"; }

};

**81. Multiple Inheritance**

#include <iostream>

using namespace std;

class Parent {

public:

void show() { cout << "Parent class\n"; }

};

class Child1 : public Parent {

public:

void display1() { cout << "Child1 class\n"; }

};

class Child2 : public Parent {

public:

void display2() { cout << "Child2 class\n"; }

};

**82. Hierarchical Inheritance**

class Base {

public:

void show() { cout << "Base class\n"; }

};

class Derived1 : public Base {

public:

void d1() { cout << "Derived1\n"; }

};

class Derived2 : public Base {

public:

void d2() { cout << "Derived2\n"; }

};

class Derived3 : public Base {

public:

void d3() { cout << "Derived3\n"; }

};

**83. Multilevel Inheritance**

class Base {

public:

void base() { cout << "Base\n"; }

};

class Intermediate : public Base {

public:

void intermediate() { cout << "Intermediate\n"; }

};

class Derived : public Intermediate {

public:

void derived() { cout << "Derived\n"; }

};

**84. Hybrid Inheritance**

class Base {

public:

void base() { cout << "Base\n"; }

};

class Derived1 : virtual public Base {};

class Derived2 : virtual public Base {};

class Derived3 : public Derived1, public Derived2 {

public:

void hybrid() { cout << "Hybrid Derived3\n"; }

};

**85. Access Specifiers**

class Library {

private:

int books;

protected:

int shelves;

public:

void setData(int b, int s) {

books = b;

shelves = s;

}

void display() {

cout << "Books: " << books << ", Shelves: " << shelves << endl;

}

};

**86. Encapsulation with Account**

class Account {

private:

double balance;

public:

void setBalance(double b) { balance = b; }

double getBalance() { return balance; }

};

**87. Function Overriding**

class Shape {

public:

virtual void draw() { cout << "Drawing Shape\n"; }

};

class Circle : public Shape {

public:

void draw() override { cout << "Drawing Circle\n"; }

};

**88. Employee and Manager**

class Employee {

public:

virtual void role() { cout << "Employee\n"; }

};

class Manager : public Employee {

public:

void role() override { cout << "Manager\n"; }

};

**89. Virtual Base Class**

class Entity {

public:

void id() { cout << "Entity ID\n"; }

};

class Person : virtual public Entity {};

class Organization : virtual public Entity {};

**90. Diamond Problem Solution**

class Animal {

public:

void sound() { cout << "Animal sound\n"; }

};

class Mammal : virtual public Animal {};

class Bird : virtual public Animal {};

class Bat : public Mammal, public Bird {};

**91. Polynomial Class**

class Polynomial {

public:

int coeff[10];

Polynomial() { fill(coeff, coeff+10, 0);

Polynomial add(Polynomial p) {

Polynomial res;

for (int i = 0; i < 10; i++)

res.coeff[i] = coeff[i] + p.coeff[i];

return res;

}

};

**92. Sparse Matrix**

class SparseMatrix {

int mat[3][3];

public:

void set(int a[3][3]) {

for(int i=0;i<3;i++)

for(int j=0;j<3;j++)

mat[i][j] = a[i][j];

}

SparseMatrix add(SparseMatrix b) {

SparseMatrix res;

for(int i=0;i<3;i++)

for(int j=0;j<3;j++)

res.mat[i][j] = mat[i][j] + b.mat[i][j];

return res;

}

};

**93. Time Operations**

class Time {

public:

int h, m;

void set(int hours, int mins) {

h = hours; m = mins;

}

void add(Time t) {

m += t.m;

h += t.h + m / 60;

m %= 60;

}

bool isEqual(Time t) {

return h == t.h && m == t.m;

}

};

**94. BigNumber**

#include <string>

using namespace std;

class BigNumber {

public:

string add(string a, string b) {

string res = "";

int carry = 0;

int i = a.length() - 1, j = b.length() - 1;

while (i >= 0 || j >= 0 || carry) {

int sum = carry;

if (i >= 0) sum += a[i--] - '0';

if (j >= 0) sum += b[j--] - '0';

carry = sum / 10;

res.insert(res.begin(), (sum % 10) + '0');

}

return res;

}

};

**95. File Compressor (Simulation)**

class FileCompressor {

public:

void compress(string file) { cout << "Compressing " << file << endl; }

void decompress(string file) { cout << "Decompressing " << file << endl; }

};

**96. Network Simulation**

class Network {

public:

void sendPacket(string data) {

cout << "Sending: " << data << endl;

}

void receivePacket(string data) {

cout << "Received: " << data << endl;

}

};

**97. Cache**

#include <unordered\_map>

using namespace std;

class Cache {

unordered\_map<string, string> data;

public:

void store(string key, string value) {

data[key] = value;

}

string retrieve(string key) {

return data.count(key) ? data[key] : "Not Found";

}

};

**98. Game Simulation**

class Game {

int score = 0;

public:

void action(string act) {

if (act == "jump") score += 10;

else if (act == "duck") score += 5;

cout << "Action: " << act << ", Score: " << score << endl;

}

};

Assignment 4(practical)

**51. Shape with virtual area() function**

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

virtual float area() = 0;

};

class Circle : public Shape {

float radius;

public:

Circle(float r) : radius(r) {}

float area() { return 3.14f \* radius \* radius; }

};

class Rectangle : public Shape {

float length, width;

public:

Rectangle(float l, float w) : length(l), width(w) {}

float area() { return length \* width; }

};

class Triangle : public Shape {

float base, height;

public:

Triangle(float b, float h) : base(b), height(h) {}

float area() { return 0.5f \* base \* height; }

};

**52. Animal speak() example**

class Animal {

public:

virtual void speak() = 0;

};

class Dog : public Animal {

public:

void speak() { cout << "Dog barks\n"; }

};

class Cat : public Animal {

public:

void speak() { cout << "Cat meows\n"; }

};

class Bird : public Animal {

public:

void speak() { cout << "Bird chirps\n"; }

};

**53. Function overriding - Employee**

class Employee {

public:

virtual void showRole() {

cout << "Employee\n";

}

};

class Manager : public Employee {

public:

void showRole() override {

cout << "Manager\n";

}

};

class Worker : public Employee {

public:

void showRole() override {

cout << "Worker\n";

}

};

**54. Pointer arithmetic**

int arr[] = {10, 20, 30, 40};

int\* ptr = arr;

for (int i = 0; i < 4; i++)

cout << \*(ptr + i) << " ";

**55. Dynamic memory allocation**

int\* arr = new int[5];

for (int i = 0; i < 5; i++) {

\*(arr + i) = i \* 10;

cout << \*(arr + i) << " ";

}

delete[] arr;

**56. Swap using pointer**

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

**57. Dynamic object of Student**

class Student {

public:

string name;

void display() {

cout << "Student: " << name << endl;

}

};

Student\* s = new Student();

s->name = "Alice";

s->display();

delete s;

**58. Pointer to array of objects**

t

class Book {

public:

string title;

void show() { cout << "Title: " << title << endl; }

};

Book\* books = new Book[2];

books[0].title = "C++";

books[1].title = "Java";

for (int i = 0; i < 2; i++) books[i].show();

delete[] books;

**59. Pointer to object in member function**

class Demo {

public:

void show(Demo\* d) {

cout << "Accessing via pointer\n";

}

};

**60. this pointer to return object**

class Box {

int length;

public:

Box(int l) : length(l) {}

Box& getBox() {

return \*this;

}

};

**61. Chaining member function calls with this**

class Person {

string name;

int age;

public:

Person& setName(string n) {

name = n;

return \*this;

}

Person& setAge(int a) {

age = a;

return \*this;

}

void display() {

cout << name << " " << age << endl;

}

};

**62. Compare two Counter objects using this**

class Counter {

int count;

public:

Counter(int c) : count(c) {}

void compare(const Counter& other) {

if (this->count > other.count)

cout << "This is greater\n";

else

cout << "Other is greater or equal\n";

}

};

**63. Abstract Vehicle class with pure virtual function**

class Vehicle {

public:

virtual void start() = 0;

};

class Car : public Vehicle {

public:

void start() { cout << "Car starts\n"; }

};

class Bike : public Vehicle {

public:

void start() { cout << "Bike starts\n"; }

};

**64. Runtime polymorphism with Shape**

class Shape {

public:

virtual void draw() { cout << "Drawing shape\n"; }

};

class Circle : public Shape {

public:

void draw() override { cout << "Drawing circle\n"; }

};

class Square : public Shape {

public:

void draw() override { cout << "Drawing square\n"; }

};

**65. Abstract Account class**

class Account {

public:

virtual float calculateInterest() = 0;

};

class SavingsAccount : public Account {

public:

float calculateInterest() { return 1000 \* 0.04f; }

};

class CurrentAccount : public Account {

public:

float calculateInterest() { return 1000 \* 0.02f; }

};

**66. Media base class with polymorphism**

class Media {

public:

virtual void display() = 0;

};

class Book : public Media {

public:

void display() { cout << "Book media\n"; }

};

class DVD : public Media {

public:

void display() { cout << "DVD media\n"; }

};

**67. Appliance class hierarchy**

class Appliance {

public:

virtual void function() = 0;

};

class WashingMachine : public Appliance {

public:

void function() { cout << "Washing clothes\n"; }

};

class Refrigerator : public Appliance {

public:

void function() { cout << "Cooling food\n"; }

};

class Microwave : public Appliance {

public:

void function() { cout << "Heating food\n"; }

};

**68. Area with Shape base class**

class Shape {

public:

virtual float area() = 0;

};

class Circle : public Shape {

float r;

public:

Circle(float r) : r(r) {}

float area() { return 3.14f \* r \* r; }

};

class Rectangle : public Shape {

float l, b;

public:

Rectangle(float l, float b) : l(l), b(b) {}

float area() { return l \* b; }

};

**69. Abstract Employee class**

class Employ

ee {

public:

virtual float calculateSalary() = 0;

virtual void displayDetails() = 0;

};

class Manager : public Employee {

public:

float calculateSalary() { return 50000; }

void displayDetails() { cout << "Manager\n"; }

};

class Engineer : public Employee {

public:

float calculateSalary() { return 30000; }

void displayDetails() { cout << "Engineer\n"; }

};

**70. Abstract Payment class**

class Payment {

public:

virtual void processPayment() = 0;

};

class CreditCardPayment : public Payment {

public:

void processPayment() { cout << "Credit card processed\n"; }

};

class DebitCardPayment : public Payment {

public:

void processPayment() { cout << "Debit card processed\n"; }

};

**71. Device abstract class**

class Device {

public:

virtual void turnOn() = 0;

};

class Laptop : public Device {

public:

void turnOn() { cout << "Laptop turned on\n"; }

};

class Smartphone : public Device {

public:

void turnOn() { cout << "Smartphone turned on\n"; }

};

**72. Division by zero exception**

try {

int a = 10, b = 0;

if (b == 0) throw runtime\_error("Division by zero");

cout << a / b;

} catch (exception& e) {

cout << e.what();

}

**73. Multiple catch blocks**

try {

throw 3.14;

} catch (int e) {

cout << "Integer exception\n";

} catch (double e) {

cout << "Double exception\n";

} catch (...) {

cout << "Unknown exception\n";

}

**74. Custom InvalidAgeException**

class InvalidAgeException : public exception {

public:

const char\* what() const noexcept override {

return "Invalid age!";

}

};

void checkAge(int age) {

if (age < 0 || age > 120)

throw InvalidAgeException();

}

**75. File I/O error handling**

#include <fstream>

ifstream file("nonexistent.txt");

if (!file) {

cout << "File error\n";

}

**76. Finally-like behavior**

try {

throw runtime\_error("Error");

} catch (...) {

cout << "Caught error\n";

}

// Finally block simulation

cout << "Releasing resources\n";

**77. Class Shape with derived classes Circle, Rectangle, and Triangle:**

class Shape {

public:

virtual float area() = 0;

};

class Circle : public Shape {

float radius;

public:

Circle(float r) : radius(r) {}

float area() override { return 3.14f \* radius \* radius; }

};

class Rectangle : public Shape {

float length, width;

public:

Rectangle(float l, float w) : length(l), width(w) {}

float area() override { return length \* width; }

};

class Triangle : public Shape {

float base, height;

public:

Triangle(float b, float h) : base(b), height(h) {}

float area() override { return 0.5f \* base \* height; }

};

**78. Class Animal with derived classes Dog, Cat, Bird:**

class Animal {

public:

virtual void speak() = 0;

};

class Dog : public Animal {

public:

void speak() override { cout << "Woof!" << endl; }

};

class Cat : public Animal {

public:

void speak() override { cout << "Meow!" << endl; }

};

class Bird : public Animal {

public:

void speak() override { cout << "Tweet!" << endl; }

};

Assignment 5(practical)

### **50. Basic input and output using cin and cout**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

cout << "You entered: " << number << endl;

return 0;

}

### **51. Read and display multiple lines of text**

#include <iostream>

#include <string>

using namespace std;

int main() {

string line;

cout << "Enter multiple lines (type 'exit' to stop):\n";

while (true) {

getline(cin, line);

if (line == "exit") break;

cout << "You entered: " << line << endl;

}

return 0;

}

### **52. Read integers and display their sum**

#include <iostream>

using namespace std;

int main() {

int sum = 0, num;

cout << "Enter integers to sum (0 to end): ";

while (true) {

cin >> num;

if (num == 0) break;

sum += num;

}

cout << "Sum: " << sum << endl;

return 0;

}

### **53. Input and output various data types**

#include <iostream>

using namespace std;

int main() {

int i;

float f;

char c;

string s;

cout << "Enter an integer: ";

cin >> i;

cout << "Enter a float: ";

cin >> f;

cout << "Enter a character: ";

cin >> c;

cout << "Enter a string: ";

cin >> s;

cout << "Integer: " << i << "\nFloat: " << f << "\nChar: " << c << "\nString: " << s << endl;

return 0;

}

### **54. Format output using manipulators**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

float pi = 3.1415926535;

cout << fixed << setprecision(2);

cout << "Formatted Pi (2 decimal): " << pi << endl;

cout << setw(10) << "Right" << endl;

cout << left << setw(10) << "Left" << "Aligned" << endl;

return 0;

}

### **55. Input name, age, salary and display formatted**

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

int main() {

string name;

int age;

float salary;

cout << "Enter name: ";

cin.ignore(); // Clear newline

getline(cin, name);

cout << "Enter age: ";

cin >> age;

cout << "Enter salary: ";

cin >> salary;

cout << fixed << setprecision(2);

cout << "\nName: " << setw(20) << name << "\nAge: " << age << "\nSalary: $" << salary << endl;

return 0;

}

### **56. Demonstrate ifstream and ofstream**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream outFile("example.txt");

outFile << "Hello File!\n";

outFile.close();

ifstream inFile("example.txt");

string line;

getline(inFile, line);

cout << "Read from file: " << line << endl;

inFile.close();

return 0;

}

### **57. Read integers from file and display**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream inFile("numbers.txt");

int num;

while (inFile >> num) {

cout << num << " ";

}

inFile.close();

return 0;

}

### **58. Write list of strings to a file**

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int main() {

ofstream outFile("strings.txt");

vector<string> lines = {"Apple", "Banana", "Cherry"};

for (const auto& s : lines) {

outFile << s << endl;

}

outFile.close();

return 0;

}

### **59. Unformatted I/O using get and put**

#include <iostream>

using namespace std;

int main() {

char ch;

cout << "Enter characters (Ctrl+D to stop): ";

while (cin.get(ch)) {

cout.put(ch);

}

return 0;

}

### **60. Read and write characters using get and put**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream outFile("chars.txt");

outFile.put('A');

outFile.put('\n');

outFile.put('B');

outFile.close();

ifstream inFile("chars.txt");

char ch;

while (inFile.get(ch)) {

cout.put(ch);

}

inFile.close();

return 0;

}

### **61. Formatted input/output to display a table**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

cout << left << setw(15) << "Name" << setw(10) << "Age" << setw(10) << "Score" << endl;

cout << "------------------------------------" << endl;

cout << left << setw(15) << "Alice" << setw(10) << 20 << setw(10) << 95.5 << endl;

cout << left << setw(15) << "Bob" << setw(10) << 22 << setw(10) << 89.0 << endl;

cout << left << setw(15) << "Charlie" << setw(10) << 19 << setw(10) << 72.25 << endl;

return 0;

}

### **62. Use getline to read a full line**

#include <iostream>

#include <string>

using namespace std;

int main() {

string text;

cout << "Enter a full line of text: ";

getline(cin, text);

cout << "You entered: " << text << endl;

return 0;

}

### **63. Format floating-point numbers with different precisions**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double value = 3.14159265;

cout << fixed;

cout << "Precision 2: " << setprecision(2) << value << endl;

cout << "Precision 4: " << setprecision(4) << value << endl;

cout << "Precision 6: " << setprecision(6) << value << endl;

return 0;

}

### **64. Use setw to align text output in columns**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

cout << setw(10) << "Item" << setw(10) << "Qty" << setw(10) << "Price" << endl;

cout << setw(10) << "Pen" << setw(10) << 10 << setw(10) << 1.25 << endl;

cout << setw(10) << "Notebook" << setw(10) << 5 << setw(10) << 2.50 << endl;

return 0;

}

### **65. Format currency and percentage values**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double price = 49.99;

double discount = 0.15;

cout << fixed << setprecision(2);

cout << "Price: $" << price << endl;

cout << "Discount: " << (discount \* 100) << "%" << endl;

return 0;

}

### **66. Read data from a text file**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream file("data.txt");

string line;

while (getline(file, line)) {

cout << line << endl;

}

file.close();

return 0;

}

### **67. Write user input to a text file**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ofstream file("output.txt");

string input;

cout << "Enter text to write to file (type 'end' to stop):\n";

while (true) {

getline(cin, input);

if (input == "end") break;

file << input << endl;

}

file.close();

return 0;

}

### **68. Copy contents of one file to another**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream source("source.txt");

ofstream destination("destination.txt");

string line;

while (getline(source, line)) {

destination << line << endl;

}

source.close();

destination.close();

return 0;

}

### **69. Append new data to an existing file**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ofstream file("append.txt", ios::app);

string input;

cout << "Enter text to append (type 'end' to stop):\n";

while (true) {

getline(cin, input);

if (input == "end") break;

file << input << endl;

}

file.close();

return 0;

}

### **70. Read binary data from a file**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file("data.bin", ios::binary);

int number;

file.read(reinterpret\_cast<char\*>(&number), sizeof(number));

cout << "Read number: " << number << endl;

file.close();

return 0;

}

### **71. Write binary data to a file**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream file("data.bin", ios::binary);

int number = 12345;

file.write(reinterpret\_cast<char\*>(&number), sizeof(number));

file.close();

return 0;

}

### **72. Use fstream for both input and output**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

fstream file("data.txt", ios::in | ios::out | ios::trunc);

file << "Hello, file stream!" << endl;

file.seekg(0);

string line;

getline(file, line);

cout << "Read from file: " << line << endl;

file.close();

return 0;

}

### **73. Read/write complex data structures using binary file streams**

#include <iostream>

#include <fstream>

using namespace std;

struct Person {

char name[50];

int age;

float salary;

};

int main() {

Person p1 = {"John Doe", 30, 55000.50};

ofstream outFile("person.dat", ios::binary);

outFile.write(reinterpret\_cast<char\*>(&p1), sizeof(p1));

outFile.close();

Person p2;

ifstream inFile("person.dat", ios::binary);

inFile.read(reinterpret\_cast<char\*>(&p2), sizeof(p2));

inFile.close();

cout << "Name: " << p2.name << "\nAge: " << p2.age << "\nSalary: " << p2.salary << endl;

return 0;

}

### **74. Rename and delete files**

#include <iostream>

#include <cstdio>

using namespace std;

int main() {

if (rename("oldname.txt", "newname.txt") == 0)

cout << "File renamed successfully.\n";

else

perror("Rename failed");

if (remove("newname.txt") == 0)

cout << "File deleted successfully.\n";

else

perror("Delete failed");

return 0;

}

### **75. Create, open, and close files**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

fstream file;

file.open("myfile.txt", ios::out); // Create and open

if (file.is\_open()) {

file << "File created and opened.\n";

file.close(); // Close

cout << "File closed successfully." << endl;

} else {

cout << "Failed to create/open file." << endl;

}

return 0;

}

### **76. Use seekg and tellg to manipulate get pointer**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file("example.txt");

if (file.is\_open()) {

file.seekg(0, ios::end);

streampos endPos = file.tellg();

cout << "File size: " << endPos << " bytes" << endl;

file.seekg(0);

streampos startPos = file.tellg();

cout << "Start position: " << startPos << endl;

file.close();

}

return 0;

}

### **77. Use seekp and tellp to manipulate put pointer**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream file("output.txt");

if (file.is\_open()) {

streampos pos = file.tellp();

cout << "Initial put pointer: " << pos << endl;

file.seekp(10);

file << "Hello";

pos = file.tellp();

cout << "Put pointer after writing: " << pos << endl;

file.close();

}

return 0;

}

### **78. Open file in read, write, and append modes**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

// Write mode (overwrites file)

ofstream writeFile("filemode.txt");

writeFile << "This is write mode.\n";

writeFile.close();

// Append mode

ofstream appendFile("filemode.txt", ios::app);

appendFile << "This is appended line.\n";

appendFile.close();

// Read mode

ifstream readFile("filemode.txt");

string line;

while (getline(readFile, line)) {

cout << line << endl;

}

readFile.close();

return 0;

}

### **79. Read and write in binary mode**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

int number = 123456;

ofstream outFile("binaryfile.bin", ios::binary);

outFile.write(reinterpret\_cast<char\*>(&number), sizeof(number));

outFile.close();

int readNum;

ifstream inFile("binaryfile.bin", ios::binary);

inFile.read(reinterpret\_cast<char\*>(&readNum), sizeof(readNum));

inFile.close();

cout << "Read number: " << readNum << endl;

return 0;

}

### **80. Difference between text and binary file modes**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

int num = 12345;

ofstream textFile("text.txt");

textFile << num;

textFile.close();

ofstream binFile("binary.bin", ios::binary);

binFile.write(reinterpret\_cast<char\*>(&num), sizeof(num));

binFile.close();

cout << "Written in text and binary mode. Check file sizes." << endl;

return 0;

}

### **81. Open a file in truncation mode**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

// Write initial content

ofstream file("truncate.txt");

file << "Old data...\n";

file.close();

// Truncate and write new content

ofstream truncFile("truncate.txt", ios::trunc);

truncFile << "New data.\n";

truncFile.close();

cout << "File truncated and rewritten." << endl;

return 0;

}

### **82. Read/write binary data using read and write**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

float value = 3.14;

ofstream outFile("float.bin", ios::binary);

outFile.write(reinterpret\_cast<char\*>(&value), sizeof(value));

outFile.close();

float readVal;

ifstream inFile("float.bin", ios::binary);

inFile.read(reinterpret\_cast<char\*>(&readVal), sizeof(readVal));

inFile.close();

cout << "Read float: " << readVal << endl;

return 0;

}

### **83. Random access read/write in binary file**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

fstream file("random.bin", ios::in | ios::out | ios::binary | ios::trunc);

int nums[] = {10, 20, 30, 40, 50};

for (int i = 0; i < 5; ++i)

file.write(reinterpret\_cast<char\*>(&nums[i]), sizeof(int));

int newValue = 999;

file.seekp(sizeof(int) \* 2, ios::beg); // Go to 3rd int

file.write(reinterpret\_cast<char\*>(&newValue), sizeof(int));

file.seekg(0);

int read;

for (int i = 0; i < 5; ++i) {

file.read(reinterpret\_cast<char\*>(&read), sizeof(int));

cout << read << " ";

}

file.close();

return 0;

}

### **84. Read/write a structure using random access**

#include <iostream>

#include <fstream>

using namespace std;

struct Record {

int id;

char name[20];

};

int main() {

fstream file("struct.bin", ios::in | ios::out | ios::binary | ios::trunc);

Record r1 = {1, "Alice"};

Record r2 = {2, "Bob"};

file.write(reinterpret\_cast<char\*>(&r1), sizeof(Record));

file.write(reinterpret\_cast<char\*>(&r2), sizeof(Record));

Record r3;

file.seekg(sizeof(Record) \* 1, ios::beg); // Second record

file.read(reinterpret\_cast<char\*>(&r3), sizeof(Record));

cout << "ID: " << r3.id << ", Name: " << r3.name << endl;

file.close();

return 0;

}

### **85. Update specific record in a binary file**

#include <iostream>

#include <fstream>

using namespace std;

struct Employee {

int id;

char name[20];

};

int main() {

fstream file("employees.bin", ios::in | ios::out | ios::binary | ios::trunc);

Employee e1 = {1, "John"};

Employee e2 = {2, "Jane"};

file.write(reinterpret\_cast<char\*>(&e1), sizeof(Employee));

file.write(reinterpret\_cast<char\*>(&e2), sizeof(Employee));

// Update second employee

Employee updated = {2, "Janet"};

file.seekp(sizeof(Employee), ios::beg); // Move to 2nd record

file.write(reinterpret\_cast<char\*>(&updated), sizeof(Employee));

// Read back to verify

file.seekg(0);

Employee temp;

while (file.read(reinterpret\_cast<char\*>(&temp), sizeof(Employee))) {

cout << "ID: " << temp.id << ", Name: " << temp.name << endl;

}

file.close();

return 0;

}

### **86. Read and display contents of a binary file in reverse order**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file("data.bin", ios::binary);

if (!file) {

cerr << "Cannot open file.\n";

return 1;

}

file.seekg(0, ios::end);

streampos size = file.tellg();

int num;

for (int i = 1; i <= size / sizeof(int); ++i) {

file.seekg(-i \* sizeof(int), ios::end);

file.read(reinterpret\_cast<char\*>(&num), sizeof(int));

cout << num << " ";

}

file.close();

return 0;

}

### **87. Read user input, process it, and write results to a file**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

getline(cin, name);

cout << "Enter your age: ";

cin >> age;

ofstream file("output.txt");

if (!file) {

cerr << "Cannot open file.\n";

return 1;

}

file << "Name: " << name << "\nAge: " << age << endl;

file.close();

cout << "Data written to output.txt\n";

return 0;

}

### **88. Read a configuration file and use its settings**

#include <iostream>

#include <fstream>

#include <string>

#include <map>

using namespace std;

int main() {

ifstream config("config.txt");

if (!config) {

cerr << "Cannot open config file.\n";

return 1;

}

map<string, string> settings;

string line;

while (getline(config, line)) {

size\_t pos = line.find('=');

if (pos != string::npos) {

string key = line.substr(0, pos);

string value = line.substr(pos + 1);

settings[key] = value;

}

}

config.close();

cout << "Configuration Settings:\n";

for (const auto& pair : settings) {

cout << pair.first << " = " << pair.second << endl;

}

return 0;

}

### **89. Log error messages to a file using file streams**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream log("error.log", ios::app);

if (!log) {

cerr << "Cannot open log file.\n";

return 1;

}

log << "Error: Sample error message.\n";

log.close();

cout << "Error logged to error.log\n";

return 0;

}

### **90. Simple text editor using file streams**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

string filename;

cout << "Enter filename to edit: ";

getline(cin, filename);

ofstream file(filename, ios::app);

if (!file) {

cerr << "Cannot open file.\n";

return 1;

}

cout << "Enter text (type 'EXIT' to quit):\n";

string line;

while (getline(cin, line)) {

if (line == "EXIT") break;

file << line << endl;

}

file.close();

cout << "Text saved to " << filename << endl;

return 0;

}

### **91. Read and process a CSV file using file streams**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

using namespace std;

int main() {

ifstream file("data.csv");

if (!file) {

cerr << "Cannot open CSV file.\n";

return 1;

}

string line;

while (getline(file, line)) {

stringstream ss(line);

string item;

while (getline(ss, item, ',')) {

cout << item << " ";

}

cout << endl;

}

file.close();

return 0;

}

### **92. Search for a specific word in a text file and count its occurrences**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream file("text.txt");

if (!file) {

cerr << "Cannot open text file.\n";

return 1;

}

string word, target;

int count = 0;

cout << "Enter word to search: ";

cin >> target;

while (file >> word) {

if (word == target) ++count;

}

file.close();

cout << "The word '" << target << "' occurred " << count << " times.\n";

return 0;

}

### **93. Demonstrate exception handling with file operations**

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file;

file.exceptions(ifstream::failbit | ifstream::badbit);

try {

file.open("nonexistent.txt");

cout << "File opened successfully.\n";

file.close();

} catch (const ifstream::failure& e) {

cerr << "Exception opening file: " << e.what() << endl;

}

return 0;

}

### **94. Compress and decompress text files using simple encoding**

#include <iostream>

#include <fstream>

using namespace std;

void compress(const string& input, const string& output) {

ifstream in(input);

ofstream out(output);

char ch;

while (in.get(ch)) {

out.put(ch + 1); // Simple Caesar cipher

}

in.close();

out.close();

}

void decompress(const string& input, const string& output) {

ifstream in(input);

ofstream out(output);

char ch;

while (in.get(ch)) {

out.put(ch - 1); // Reverse Caesar cipher

}

in.close();

out.close();

}

int main() {

compress("original.txt", "compressed.txt");

decompress("compressed.txt", "decompressed.txt");

cout << "Compression and decompression completed.\n";

return 0;

}

### **95. Merge contents of multiple text files into a single file**

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int main() {

vector<string> files = {"file1.txt", "file2.txt", "file3.txt"};

ofstream out("merged.txt");

if (!out) {

cerr << "Cannot open output file.\n";

return 1;

}

for (const auto& fname : files) {

ifstream in(fname);

if (!in) {

cerr << "Cannot open " << fname << endl;

continue;

}

out << in.rdbuf();

in.close();

}

out.close();

cout << "Files merged into merged.txt\n";

return 0;

}

### **96. Read and process large data files using memory-mapped files**

#include <iostream>

#include <fstream>

#include <sys/mman.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/stat.h>

using namespace std;

int main() {

const char\* filename = "largefile.txt";

int fd = open(filename, O\_RDONLY);

if (fd == -1) {

cerr << "Cannot open file.\n";

return 1;

}

struct stat sb;

if (fstat(fd, &sb) == -1) {

cerr << "Cannot get file size.\n";

close(fd);

return 1;

}

char\* data = static\_cast<char\*>(mmap(NULL, sb.st\_size, PROT\_READ, MAP\_PRIVATE, fd, 0));

if (data == MAP\_FAILED) {

cerr << "Memory mapping failed.\n";

close(fd);

return 1;

}

// Process data

for (off\_t i = 0; i < sb.st\_size; ++i) {

// Example: print each character

cout << data[i];

}

munmap(data, sb.st\_size);

close(fd);

return 0;

}

### **97. Perform basic encryption and decryption of text files**

#include <iostream>

#include <fstream>

using namespace std;

void encrypt(const string& input, const string& output, int key) {

ifstream in(input);

ofstream out(output);

char ch;

while (in.get(ch)) {

out.put(ch + key);

}

in.close();

out.close();

}

void decrypt(const string& input, const string& output, int key) {

ifstream in(input);

ofstream out(output);

char ch;

while (in.get(ch)) {

out.put(ch - key);

}

in.close();

out.close();

}