**DAY-02**

**Question-6** : .In a high-performance computing application, you need to frequently calculate the square of numbers. To optimize these calculations, you decide to use an inline function.

**Sol** : #include <iostream>

inline int square(int num) {

return num \* num;

}

int main() {

int num = 8;

std::cout << "Square of " << num << " is: " << square(num) << std::endl;

return 0;

}

**O/P** : Square of 2 is: 4

--------------------------------

Process exited after 0.09099 seconds with return value 0

Press any key to continue . . .

**Question 07** : Background:

You are implementing a configuration function for a game engine that initializes game settings like screen resolution, difficulty level, and whether to enable fullscreen mode.

Question:Design a configureGame function that includes default arguments, allowing users to specify only the settings they want to change from the defaults.

**Sol:** #include <iostream>

#include <string>

struct GameSettings {

int screenWidth;

int screenHeight;

int difficultyLevel;

bool fullscreen;

};

void configureGame(int screenWidth = 1920, int screenHeight = 1080, int difficultyLevel = 2, bool fullscreen = true) {

std::cout << "Configuring game settings:" << std::endl;

std::cout << "Screen Resolution: " << screenWidth << "x" << screenHeight << std::endl;

std::cout << "Difficulty Level: " << difficultyLevel << std::endl;

std::cout << "Fullscreen Mode: " << (fullscreen ? "Enabled" : "Disabled") << std::endl;

}

int main() {

configureGame();

configureGame(2560, 1440);

configureGame(1920, 1080, 3);

configureGame(1920, 1080, 2, false);

return 0;

}

**O/P** : Configuring game settings:

Screen Resolution: 1920x1080

Difficulty Level: 2

Fullscreen Mode: Enabled

Configuring game settings:

Screen Resolution: 2560x1440

Difficulty Level: 2

Fullscreen Mode: Enabled

Configuring game settings:

Screen Resolution: 1920x1080

Difficulty Level: 3

Fullscreen Mode: Enabled

Configuring game settings:

Screen Resolution: 1920x1080

Difficulty Level: 2

Fullscreen Mode: Disabled

--------------------------------

Process exited after 0.09847 seconds with return value 0

Press any key to continue . . .

**Question-08** : 8. Problem Statement: You are given a dataset representing the daily average temperatures of a city over a month, stored in an array. Write a C++ program that includes functions to perform the following tasks:

Calculate the average temperature of the month.

Find the maximum and minimum temperatures recorded in the month.

Determine the number of days the temperature was above the monthly average.

**Sol :** #include <iostream>

double calculateAverageTemperature(const double temperatures[], int size) {

double total = 0;

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

total += temperatures[i];

}

return total / size;

}

double findMaxTemperature(const double temperatures[], int size) {

double maxTemp = temperatures[0];

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

if (temperatures[i] > maxTemp) {

maxTemp = temperatures[i];

}

}

return maxTemp;

}

double findMinTemperature(const double temperatures[], int size) {

double minTemp = temperatures[0];

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

if (temperatures[i] < minTemp) {

minTemp = temperatures[i];

}

}

return minTemp;

}

int countDaysAboveAverage(const double temperatures[], int size) {

double average = calculateAverageTemperature(temperatures, size);

int count = 0;

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

if (temperatures[i] > average) {

count++;

}

}

return count;

}

int main() {

double temperatures[] = {22.5, 25.3, 20.2, 18.6, 21.7, 24.5, 26.1, 27.8, 19.2, 23.4, 25.6, 24.1, 22.9, 19.8, 23.0, 24.7, 26.5, 27.3, 21.4, 20.3, 22.2, 23.9, 25.0, 26.2, 27.4, 28.0, 19.5, 18.4, 21.6, 23.7};

int size = sizeof(temperatures) / sizeof(temperatures[0]);

double average = calculateAverageTemperature(temperatures, size);

double maxTemp = findMaxTemperature(temperatures, size);

double minTemp = findMinTemperature(temperatures, size);

int daysAboveAverage = countDaysAboveAverage(temperatures, size);

std::cout << "Average Temperature of the month: " << average << std::endl;

std::cout << "Maximum Temperature recorded: " << maxTemp << std::endl;

std::cout << "Minimum Temperature recorded: " << minTemp << std::endl;

std::cout << "Number of days the temperature was above the monthly average: " << daysAboveAverage << std::endl;

return 0;

}

**O/P** : Average Temperature of the month: 23.36

Maximum Temperature recorded: 28

Minimum Temperature recorded: 18.4

Number of days the temperature was above the monthly average: 16

--------------------------------

Process exited after 0.1274 seconds with return value 0

Press any key to continue . . .