**Práctica 2do parcial**

**Ejercicio 1:**

#include <iostream>

#include <algorithm>

using namespace std;

const int NUM\_ESTUDIANTES = 5;

const int NUM\_ASIGNATURAS = 3;

void calcularPromedio(float calificaciones[][NUM\_ASIGNATURAS]) {

for (int i = 0; i < NUM\_ESTUDIANTES; ++i) {

float suma = 0;

for (int j = 0; j < NUM\_ASIGNATURAS; ++j) {

suma += calificaciones[i][j];

}

float promedio = suma / NUM\_ASIGNATURAS;

cout << "El promedio del estudiante " << i + 1 << " es: " << promedio << endl;

}

}

void ordenarCalificaciones(float calificaciones[][NUM\_ASIGNATURAS]) {

for (int i = 0; i < NUM\_ESTUDIANTES; ++i) {

sort(calificaciones[i], calificaciones[i] + NUM\_ASIGNATURAS, greater<float>());

}

}

float buscarCalificacionMaxima(float calificaciones[][NUM\_ASIGNATURAS], int asignatura) {

float maxima = calificaciones[0][asignatura];

for (int i = 1; i < NUM\_ESTUDIANTES; ++i) {

if (calificaciones[i][asignatura] > maxima) {

maxima = calificaciones[i][asignatura];

}

}

return maxima;

}

int main() {

float calificaciones[NUM\_ESTUDIANTES][NUM\_ASIGNATURAS] = {

{8, 7, 9},

{6, 8, 7},

{9, 9, 8},

{7, 6, 7},

{8, 9, 9}

};

calcularPromedio(calificaciones);

ordenarCalificaciones(calificaciones);

cout << "Calificaciones ordenadas de mayor a menor:\n";

for (int i = 0; i < NUM\_ESTUDIANTES; ++i) {

cout << "Estudiante " << i + 1 << ": ";

for (int j = 0; j < NUM\_ASIGNATURAS; ++j) {

cout << calificaciones[i][j] << " ";

}

cout << endl;

}

int asignatura;

cout << "Ingresa el número de la asignatura para encontrar la calificación más alta: ";

cin >> asignatura;

if (asignatura >= 0 && asignatura < NUM\_ASIGNATURAS) {

float maxima = buscarCalificacionMaxima(calificaciones, asignatura);

cout << "La calificación más alta en la asignatura " << asignatura + 1 << " es: " << maxima << endl;

} else {

cout << "Número de asignatura inválido." << endl;

}

return 0;

}

**Ejercicio 2**

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

const int MAX\_PRODUCTOS = 100;

const int COLUMNAS = 3;

struct Producto {

int codigo;

string nombre;

int stock;

};

void agregarProducto(Producto inventario[], int& numProductos) {

if (numProductos < MAX\_PRODUCTOS) {

cout << "Ingrese el código del producto: ";

cin >> inventario[numProductos].codigo;

cout << "Ingrese el nombre del producto: ";

cin >> inventario[numProductos].nombre;

cout << "Ingrese la cantidad en stock: ";

cin >> inventario[numProductos].stock;

numProductos++;

cout << "Producto agregado correctamente." << endl;

} else {

cout << "No se pueden agregar más productos, inventario lleno." << endl;

}

}

void ordenarPorNombre(Producto inventario[], int numProductos) {

sort(inventario, inventario + numProductos, [](const Producto& a, const Producto& b) {

return a.nombre < b.nombre;

});

cout << "Inventario ordenado por nombre del producto." << endl;

}

void buscarPorCodigo(Producto inventario[], int numProductos, int codigo) {

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

if (inventario[i].codigo == codigo) {

cout << "Producto encontrado: " << inventario[i].nombre << endl;

return;

}

}

cout << "Producto no encontrado." << endl;

}

void venderProducto(Producto inventario[], int numProductos, int codigo, int cantidad) {

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

if (inventario[i].codigo == codigo) {

if (inventario[i].stock >= cantidad) {

inventario[i].stock -= cantidad;

cout << "Venta realizada correctamente." << endl;

} else {

cout << "No hay suficiente stock para realizar la venta." << endl;

}

return;

}

}

cout << "Producto no encontrado." << endl;

}

void recibirStock(Producto inventario[], int numProductos, int codigo, int cantidad) {

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

if (inventario[i].codigo == codigo) {

inventario[i].stock += cantidad;

cout << "Stock actualizado correctamente." << endl;

return;

}

}

cout << "Producto no encontrado." << endl;

}

int main() {

Producto inventario[MAX\_PRODUCTOS];

int numProductos = 0;

agregarProducto(inventario, numProductos);

agregarProducto(inventario, numProductos);

ordenarPorNombre(inventario, numProductos);

buscarPorCodigo(inventario, numProductos, 1);

venderProducto(inventario, numProductos, 1, 5);

recibirStock(inventario, numProductos, 1, 10);

return 0;

}

**Ejercicio 3**

**#include <iostream>**

Using namespace std;

Const int ROWS = 10;

Const int COLS = 10;

Void initializeBoard(int board[ROWS][COLS]) {

For (int i = 0; i < ROWS; ++i) {

For (int j = 0; j < COLS; ++j) {

Board[i][j] = 0;

}

}

}

Bool placePiece(int board[ROWS][COLS], int row, int col, int piece) {

If (row >= 0 && row < ROWS && col >= 0 && col < COLS && (piece == 1 || piece == 2)) {

Board[row][col] = piece;

Return true;

}

Return false;

}

Bool checkForWin(int board[ROWS][COLS], int piece) {

For (int i = 0; i < ROWS; ++i) {

For (int j = 0; j <= COLS – 4; ++j) {

If (board[i][j] == piece && board[i][j + 1] == piece &&

Board[i][j + 2] == piece && board[i][j + 3] == piece) {

Return true;

}

}

}

For (int i = 0; i <= ROWS – 4; ++i) {

For (int j = 0; j < COLS; ++j) {

If (board[i][j] == piece && board[i + 1][j] == piece &&

Board[i + 2][j] == piece && board[i + 3][j] == piece) {

Return true;

}

}

}

For (int i = 0; i <= ROWS – 4; ++i) {

For (int j = 0; j <= COLS – 4; ++j) {

If (board[i][j] == piece && board[i + 1][j + 1] == piece &&

Board[i + 2][j + 2] == piece && board[i + 3][j + 3] == piece) {

Return true;

}

}

}

For (int i = 3; i < ROWS; ++i) {

For (int j = 0; j <= COLS – 4; ++j) {

If (board[i][j] == piece && board[i – 1][j + 1] == piece &&

Board[i – 2][j + 2] == piece && board[i – 3][j + 3] == piece) {

Return true;

}

}

}

Return false;

}

Void printBoard(int board[ROWS][COLS]) {

For (int i = 0; i < ROWS; ++i) {

For (int j = 0; j < COLS; ++j) {

Cout << board[i][j] << “ “;

}

Cout << endl;

}

}

Int main() {

Int board[ROWS][COLS];

initializeBoard(board);

placePiece(board, 0, 0, 1);

placePiece(board, 1, 1, 1);

placePiece(board, 2, 2, 1);

placePiece(board, 3, 3, 1);

printBoard(board);

if (checkForWin(board, 1)) {

cout << “Player 1 wins!” << endl;

} else {

Cout << “No winner yet.” << endl;

}

Return 0;

}

**Ejercicio 4**

#include <iostream>

#include <algorithm>

using namespace std;

const int DAYS = 30;

const int MEASUREMENTS\_PER\_DAY = 4;

void calculateDailyAverage(const double temperatures[DAYS][MEASUREMENTS\_PER\_DAY], double dailyAverage[DAYS]) {

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

double sum = 0;

for (int j = 0; j < MEASUREMENTS\_PER\_DAY; ++j) {

sum += temperatures[i][j];

}

dailyAverage[i] = sum / MEASUREMENTS\_PER\_DAY;

}

}

void sortDailyTemperatures(double temperatures[DAYS][MEASUREMENTS\_PER\_DAY]) {

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

sort(temperatures[i], temperatures[i] + MEASUREMENTS\_PER\_DAY);

}

}

void findExtremeTemperatures(const double temperatures[DAYS][MEASUREMENTS\_PER\_DAY], int &dayWithHighestTemp, int &dayWithLowestTemp) {

double highestTemp = temperatures[0][0];

double lowestTemp = temperatures[0][0];

dayWithHighestTemp = 0;

dayWithLowestTemp = 0;

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

for (int j = 0; j < MEASUREMENTS\_PER\_DAY; ++j) {

if (temperatures[i][j] > highestTemp) {

highestTemp = temperatures[i][j];

dayWithHighestTemp = i;

}

if (temperatures[i][j] < lowestTemp) {

lowestTemp = temperatures[i][j];

dayWithLowestTemp = i;

}

}

}

}

void printDailyAverages(const double dailyAverage[DAYS]) {

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

cout << "Dia " << i + 1 << ": " << dailyAverage[i] << " grados" << endl;

}

}

int main() {

double temperatures[DAYS][MEASUREMENTS\_PER\_DAY] = {

{15.5, 20.1, 22.3, 18.7}, {16.2, 21.0, 23.4, 19.1}, {14.8, 19.7, 21.9, 17.3}, {17.0, 22.2, 24.5, 20.0},

{15.6, 20.2, 22.8, 18.5}, {16.0, 20.8, 23.0, 19.0}, {15.3, 19.9, 22.2, 18.2}, {16.8, 21.5, 23.7, 19.4},

{14.9, 19.6, 21.8, 17.2}, {17.1, 22.3, 24.6, 20.1}, {15.4, 20.0, 22.4, 18.6}, {16.1, 20.9, 23.3, 19.2},

{15.2, 19.8, 22.1, 18.1}, {16.7, 21.4, 23.6, 19.3}, {15.0, 19.5, 21.7, 17.1}, {17.2, 22.4, 24.7, 20.2},

{15.1, 19.9, 22.3, 18.0}, {16.6, 21.3, 23.5, 19.2}, {15.7, 20.3, 22.9, 18.4}, {16.3, 21.1, 23.8, 19.0},

{14.7, 19.6, 21.8, 17.2}, {17.0, 22.2, 24.5, 20.0}, {15.6, 20.2, 22.8, 18.5}, {16.0, 20.8, 23.0, 19.0},

{15.3, 19.9, 22.2, 18.2}, {16.8, 21.5, 23.7, 19.4}, {14.9, 19.6, 21.8, 17.2}, {17.1, 22.3, 24.6, 20.1},

{15.4, 20.0, 22.4, 18.6}, {16.1, 20.9, 23.3, 19.2}

};

double dailyAverage[DAYS];

calculateDailyAverage(temperatures, dailyAverage);

int dayWithHighestTemp, dayWithLowestTemp;

findExtremeTemperatures(temperatures, dayWithHighestTemp, dayWithLowestTemp);

cout << "Temperaturas promedio diarias:" << endl;

printDailyAverages(dailyAverage);

cout << endl;

cout << "El dia con la temperatura mas alta es el dia " << dayWithHighestTemp + 1 << endl;

cout << "El dia con la temperatura mas baja es el dia " << dayWithLowestTemp + 1 << endl;

return 0;

}

**Ejercicio 5**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void ingresarDistancias(vector<vector<int>>& distancias) {

}

vector<int> calcularDistanciaTotal(vector<vector<int>>& distancias) {

vector<int> distanciaTotal(distancias.size(), 0);

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

for (int j = 0; j < distancias[i].size(); ++j) {

distanciaTotal[i] += distancias[i][j];

}

}

return distanciaTotal;

}

bool compararDistanciaTotal(pair<int, int>& a, pair<int, int>& b) {

return a.second < b.second;

}

vector<pair<int, int>> ordenarCiudades(vector<int>& distanciaTotal) {

vector<pair<int, int>> ciudades;

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

ciudades.push\_back({i, distanciaTotal[i]});

}

sort(ciudades.begin(), ciudades.end(), compararDistanciaTotal);

return ciudades;

}

int buscarDistanciaMinima(vector<vector<int>>& distancias, int ciudadA, int ciudadB) {

return distancias[ciudadA][ciudadB];

}

int main() {

vector<vector<int>> distancias;

ingresarDistancias(distancias);

vector<int> distanciaTotal = calcularDistanciaTotal(distancias);

vector<pair<int, int>> ciudadesOrdenadas = ordenarCiudades(distanciaTotal);

cout << "Matriz de distancias:" << endl;

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

for (int j = 0; j < distancias[i].size(); ++j) {

cout << distancias[i][j] << " ";

}

cout << endl;

}

cout << "Ciudades ordenadas por distancia total:" << endl;

for (auto& ciudad : ciudadesOrdenadas) {

cout << "Ciudad " << ciudad.first << ": " << ciudad.second << endl;

}

int ciudadA = 0;

int ciudadB = 2;

int distanciaMinima = buscarDistanciaMinima(distancias, ciudadA, ciudadB);

cout << "Distancia mínima entre Ciudad " << ciudadA << " y Ciudad " << ciudadB << ": " << distanciaMinima << endl;

return 0;

}

**Ejercicio 6**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <algorithm>

using namespace std;

const int NUM\_ESTUDIANTES = 5;

const int NUM\_MATERIAS = 3;

void inicializarNotas(vector<vector<int>>& notas) {

srand(time(nullptr));

for (int i = 0; i < NUM\_ESTUDIANTES; ++i) {

for (int j = 0; j < NUM\_MATERIAS; ++j) {

notas[i][j] = rand() % 101;

}

}

}

vector<double> calcularPromedios(const vector<vector<int>>& notas) {

vector<double> promedios;

for (const auto& estudiante : notas) {

double sum = 0;

for (int nota : estudiante) {

sum += nota;

}

promedios.push\_back(sum / NUM\_MATERIAS);

}

return promedios;

}

void ordenarNotas(vector<vector<int>>& notas) {

for (auto& estudiante : notas) {

sort(estudiante.begin(), estudiante.end(), greater<int>());

}

}

pair<int, int> buscarNotasExtremas(const vector<vector<int>>& notas, int materia) {

int notaMaxima = INT\_MIN;

int notaMinima = INT\_MAX;

for (const auto& estudiante : notas) {

notaMaxima = max(notaMaxima, estudiante[materia]);

notaMinima = min(notaMinima, estudiante[materia]);

}

return {notaMaxima, notaMinima};

}

void imprimirNotas(const vector<vector<int>>& notas, const vector<double>& promedios, int materia) {

cout << "Notas de los estudiantes:" << endl;

for (int i = 0; i < NUM\_ESTUDIANTES; ++i) {

cout << "Estudiante " << i + 1 << ": ";

for (int j = 0; j < NUM\_MATERIAS; ++j) {

cout << notas[i][j] << " ";

}

cout << "Promedio: " << promedios[i] << endl;

}

auto extremas = buscarNotasExtremas(notas, materia);

cout << "Nota más alta en la materia " << materia << ": " << extremas.first << endl;

cout << "Nota más baja en la materia " << materia << ": " << extremas.second << endl;

}

int main() {

vector<vector<int>> notas(NUM\_ESTUDIANTES, vector<int>(NUM\_MATERIAS));

inicializarNotas(notas);

vector<double> promedios = calcularPromedios(notas);

ordenarNotas(notas);

imprimirNotas(notas, promedios, 0);

return 0;

}