**1. Đếm số âm, số dương và số 0**

#include <stdio.h>

int main() {

int n, value, positive = 0, negative = 0, zero = 0;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &value);

if (value > 0) positive++;

else if (value < 0) negative++;

else zero++;

}

printf("Number of positive elements: %d\n", positive);

printf("Number of negative elements: %d\n", negative);

printf("Number of zeros: %d\n", zero);

return 0;

}

**2. Đếm số chẵn và lẻ**

#include <stdio.h>

int main() {

int n, value, odd = 0, even = 0;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &value);

if (value % 2 == 0) even++;

else odd++;

}

printf("Number of even elements: %d\n", even);

printf("Number of odd elements: %d\n", odd);

return 0;

}

**3. Tính tổng và trung bình của mảng**

#include <stdio.h>

int main() {

int n;

float sum = 0;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

sum += arr[i];

}

printf("Sum of elements: %.2f\n", sum);

printf("Average value: %.2f\n", sum / n);

return 0;

}

**4. Tìm giá trị nhỏ nhất và lớn nhất**

#include <stdio.h>

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

int min = arr[0], max = arr[0];

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

if (arr[i] < min) min = arr[i];

if (arr[i] > max) max = arr[i];

}

printf("Minimum value: %d\n", min);

printf("Maximum value: %d\n", max);

return 0;

}

**5. Hiển thị mảng theo thứ tự ngược lại**

#include <stdio.h>

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Array in reverse order:\n");

for (int i = n - 1; i >= 0; i--) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**6. Chuyển đổi số nguyên sang nhị phân**

#include <stdio.h>

void decimalToBinary(int n) {

if (n > 1)

decimalToBinary(n / 2);

printf("%d", n % 2);

}

int main() {

int n;

printf("Enter an integer N (N >= 0): ");

scanf("%d", &n);

if (n < 0) {

printf("N must be greater than or equal to 0!\n");

return 1;

}

printf("Binary representation: ");

decimalToBinary(n);

printf("\n");

return 0;

}

**7. Chuyển đổi số nguyên có dấu sang nhị phân (2's complement)**

#include <stdio.h>

void printBinary(short n) {

for (int i = 15; i >= 0; i--) {

printf("%d", (n >> i) & 1);

}

printf("b\n");

}

int main() {

short n;

printf("Enter a 16-bit integer: ");

scanf("%hd", &n);

printf("Binary (2's complement): ");

printBinary(n);

return 0;

}

**8. Chuyển đổi số nguyên sang hệ thập lục phân (Hexadecimal)**

#include <stdio.h>

int main() {

int n;

printf("Enter an integer N (N >= 0): ");

scanf("%d", &n);

if (n < 0) {

printf("N must be greater than or equal to 0!\n");

return 1;

}

printf("Hexadecimal representation: %Xh\n", n);

return 0;

}

**9. Chuyển đổi nhị phân 16-bit sang thập phân**

#include <stdio.h>

#include <string.h>

int binaryToDecimal(char \*binary) {

int decimal = 0;

for (int i = 0; i < 16; i++) {

decimal = decimal \* 2 + (binary[i] - '0');

}

return decimal;

}

int main() {

char binary[17];

printf("Enter a 16-bit binary number: ");

scanf("%16s", binary);

printf("Decimal value: %d\n", binaryToDecimal(binary));

return 0;

}

**10. Chuyển đổi giữa các hệ cơ số (2, 8, 10, 16)**

#include <stdio.h>

#include <stdlib.h>

int main() {

char number[17];

int baseA, baseB;

printf("Enter an integer number: ");

scanf("%s", number);

printf("Enter the base A of your number (2, 8, 10): ");

scanf("%d", &baseA);

if (baseA != 2 && baseA != 8 && baseA != 10) {

printf("Base A must be 2, 8, or 10!\n");

return 1;

}

printf("Enter the base B for the result (2, 8, 10, 16): ");

scanf("%d", &baseB);

if (baseB != 2 && baseB != 8 && baseB != 10 && baseB != 16) {

printf("Base B must be 2, 8, 10, or 16!\n");

return 1;

}

int decimal = strtol(number, NULL, baseA);

printf("Conversion result: ");

if (baseB == 2) {

for (int i = 15; i >= 0; i--) {

printf("%d", (decimal >> i) & 1);

}

printf("b\n");

} else if (baseB == 8) {

printf("%o\n", decimal);

} else if (baseB == 10) {

printf("%d\n", decimal);

} else if (baseB == 16) {

printf("%Xh\n", decimal);

}

return 0;

}

**11. Hiển thị dãy Fibonacci với N phần tử**

#include <stdio.h>

void fibonacci(int n) {

int a = 0, b = 1, next;

printf("Fibonacci sequence: %d, %d", a, b);

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

next = a + b;

printf(", %d", next);

a = b;

b = next;

}

printf("\n");

}

int main() {

int n;

printf("Enter a positive integer N: ");

scanf("%d", &n);

if (n <= 0) {

printf("N must be greater than 0!\n");

return 1;

}

fibonacci(n);

return 0;

}

**12. Tìm phần tử thứ N của dãy Fibonacci**

#include <stdio.h>

int fibonacciNth(int n) {

if (n <= 1)

return n;

return fibonacciNth(n - 1) + fibonacciNth(n - 2);

}

int main() {

int n;

printf("Enter a positive integer N: ");

scanf("%d", &n);

if (n <= 0) {

printf("N must be greater than 0!\n");

return 1;

}

printf("The %dth Fibonacci number is: %d\n", n, fibonacciNth(n - 1));

return 0;

}

**13. Sắp xếp mảng bằng Bubble Sort (Tăng dần)**

#include <stdio.h>

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

bubbleSort(arr, n);

printf("Sorted array: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**14. Sắp xếp mảng bằng Selection Sort (Giảm dần)**

#include <stdio.h>

void selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

int maxIdx = i;

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

if (arr[j] > arr[maxIdx]) {

maxIdx = j;

}

}

int temp = arr[i];

arr[i] = arr[maxIdx];

arr[maxIdx] = temp;

}

}

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

selectionSort(arr, n);

printf("Sorted array in descending order: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**15. Sắp xếp mảng bằng Insertion Sort (Tăng dần)**

#include <stdio.h>

void insertionSort(int arr[], int n) {

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

}

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

insertionSort(arr, n);

printf("Sorted array in ascending order: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**16. Kiểm tra mảng có được sắp xếp theo thứ tự tăng/giảm không**

#include <stdio.h>

#include <stdbool.h>

bool isAscending(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

if (arr[i] > arr[i + 1]) return false;

}

return true;

}

bool isDescending(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

if (arr[i] < arr[i + 1]) return false;

}

return true;

}

int main() {

int n;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

if (isAscending(arr, n)) {

printf("The array is sorted in ascending order.\n");

} else if (isDescending(arr, n)) {

printf("The array is sorted in descending order.\n");

} else {

printf("The array is not sorted.\n");

}

return 0;

}

**17. Thêm phần tử mới vào mảng theo chỉ số**

#include <stdio.h>

void insertElement(int arr[], int \*n, int index, int newValue) {

for (int i = \*n; i > index; i--) {

arr[i] = arr[i - 1];

}

arr[index] = newValue;

(\*n)++;

}

int main() {

int n, index, newValue;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[100]; // Tạo mảng lớn hơn để chứa thêm phần tử

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the value to insert: ");

scanf("%d", &newValue);

printf("Enter the index to insert the value: ");

scanf("%d", &index);

if (index < 0 || index > n) {

printf("Invalid index! Must be between 0 and %d.\n", n);

return 1;

}

insertElement(arr, &n, index, newValue);

printf("Array after insertion: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**18. Xóa phần tử khỏi mảng theo chỉ số**

#include <stdio.h>

void removeElement(int arr[], int \*n, int index) {

for (int i = index; i < \*n - 1; i++) {

arr[i] = arr[i + 1];

}

(\*n)--;

}

int main() {

int n, index;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[100]; // Tạo mảng lớn hơn để chứa các phần tử

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the index to remove: ");

scanf("%d", &index);

if (index < 0 || index >= n) {

printf("Invalid index! Must be between 0 and %d.\n", n - 1);

return 1;

}

removeElement(arr, &n, index);

printf("Array after removal: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**19. Tìm phần tử đầu tiên trùng khớp trong mảng**

#include <stdio.h>

int findFirstMatch(int arr[], int n, int searchValue) {

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

if (arr[i] == searchValue) return i;

}

return -1;

}

int main() {

int n, searchValue;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the value to search: ");

scanf("%d", &searchValue);

int index = findFirstMatch(arr, n, searchValue);

if (index != -1) {

printf("The first occurrence of %d is at index [%d].\n", searchValue, index);

} else {

printf("The value %d is not found in the array.\n", searchValue);

}

return 0;

}

**20. Tìm phần tử cuối cùng trùng khớp trong mảng**

#include <stdio.h>

int findLastMatch(int arr[], int n, int searchValue) {

for (int i = n - 1; i >= 0; i--) {

if (arr[i] == searchValue) return i;

}

return -1;

}

int main() {

int n, searchValue;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the value to search: ");

scanf("%d", &searchValue);

int index = findLastMatch(arr, n, searchValue);

if (index != -1) {

printf("The last occurrence of %d is at index [%d].\n", searchValue, index);

} else {

printf("The value %d is not found in the array.\n", searchValue);

}

return 0;

}

**21. Tìm tất cả các phần tử trùng khớp trong mảng**

#include <stdio.h>

void findAllMatches(int arr[], int n, int searchValue) {

int found = 0;

printf("All elements matching %d are at indexes: ", searchValue);

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

if (arr[i] == searchValue) {

printf("[%d] ", i);

found = 1;

}

}

if (!found) {

printf("No elements match %d in the array.", searchValue);

}

printf("\n");

}

int main() {

int n, searchValue;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the value to search: ");

scanf("%d", &searchValue);

findAllMatches(arr, n, searchValue);

return 0;

}

**22. Thay thế tất cả các phần tử trùng khớp**

#include <stdio.h>

void replaceAll(int arr[], int n, int oldValue, int newValue) {

int replaced = 0;

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

if (arr[i] == oldValue) {

arr[i] = newValue;

replaced = 1;

}

}

if (replaced) {

printf("Array after replacement: ");

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

printf("%d ", arr[i]);

}

printf("\n");

} else {

printf("No elements with value %d found in the array.\n", oldValue);

}

}

int main() {

int n, oldValue, newValue;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the old value to replace: ");

scanf("%d", &oldValue);

printf("Enter the new value: ");

scanf("%d", &newValue);

replaceAll(arr, n, oldValue, newValue);

return 0;

}

**23. Xóa tất cả các phần tử trùng khớp**

#include <stdio.h>

int removeAll(int arr[], int n, int value) {

int j = 0;

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

if (arr[i] != value) {

arr[j++] = arr[i];

}

}

return j; // Trả về kích thước mới của mảng

}

int main() {

int n, value;

printf("How many elements in the array? ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements must be greater than 0!\n");

return 1;

}

int arr[n];

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

printf("Enter value for element %d: ", i + 1);

scanf("%d", &arr[i]);

}

printf("Enter the value to remove: ");

scanf("%d", &value);

n = removeAll(arr, n, value);

printf("Array after removal: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**24. Gộp hai mảng**

#include <stdio.h>

void mergeArrays(int a[], int n, int b[], int m, int result[]) {

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

result[i] = a[i];

}

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

result[n + i] = b[i];

}

}

int main() {

int n, m;

printf("How many elements in array A? ");

scanf("%d", &n);

int a[n];

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

printf("Enter value for element %d of A: ", i + 1);

scanf("%d", &a[i]);

}

printf("How many elements in array B? ");

scanf("%d", &m);

int b[m];

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

printf("Enter value for element %d of B: ", i + 1);

scanf("%d", &b[i]);

}

int result[n + m];

mergeArrays(a, n, b, m, result);

printf("Merged array: ");

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

printf("%d ", result[i]);

}

printf("\n");

return 0;

}

**25. Gộp 2 mảng đã sắp xếp theo thứ tự tăng dần**

#include <stdio.h>

void mergeSortedArrays(int a[], int n, int b[], int m, int result[]) {

int i = 0, j = 0, k = 0;

while (i < n && j < m) {

if (a[i] <= b[j]) {

result[k++] = a[i++];

} else {

result[k++] = b[j++];

}

}

while (i < n) result[k++] = a[i++];

while (j < m) result[k++] = b[j++];

}

int main() {

int n, m;

printf("How many elements in array A? ");

scanf("%d", &n);

int a[n];

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

printf("Enter value for element %d of A: ", i + 1);

scanf("%d", &a[i]);

}

printf("How many elements in array B? ");

scanf("%d", &m);

int b[m];

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

printf("Enter value for element %d of B: ", i + 1);

scanf("%d", &b[i]);

}

int result[n + m];

mergeSortedArrays(a, n, b, m, result);

printf("Merged sorted array: ");

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

printf("%d ", result[i]);

}

printf("\n");

return 0;

}

**26. Gộp 2 mảng đã sắp xếp theo thứ tự giảm dần**

#include <stdio.h>

void mergeSortedDescending(int a[], int n, int b[], int m, int result[]) {

int i = n - 1, j = m - 1, k = 0;

while (i >= 0 && j >= 0) {

if (a[i] >= b[j]) {

result[k++] = a[i--];

} else {

result[k++] = b[j--];

}

}

while (i >= 0) result[k++] = a[i--];

while (j >= 0) result[k++] = b[j--];

}

int main() {

int n, m;

printf("How many elements in array A? ");

scanf("%d", &n);

int a[n];

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

printf("Enter value for element %d of A: ", i + 1);

scanf("%d", &a[i]);

}

printf("How many elements in array B? ");

scanf("%d", &m);

int b[m];

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

printf("Enter value for element %d of B: ", i + 1);

scanf("%d", &b[i]);

}

int result[n + m];

mergeSortedDescending(a, n, b, m, result);

printf("Merged sorted array (descending): ");

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

printf("%d ", result[i]);

}

printf("\n");

return 0;

}

**28. Tìm các phần tử yên ngựa trong ma trận**

#include <stdio.h>

int isSaddlePoint(int matrix[10][10], int row, int col, int m, int n) {

int minInRow = matrix[row][0];

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

if (matrix[row][j] < minInRow) minInRow = matrix[row][j];

}

int maxInCol = matrix[0][col];

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

if (matrix[i][col] > maxInCol) maxInCol = matrix[i][col];

}

return minInRow == matrix[row][col] && maxInCol == matrix[row][col];

}

int main() {

int m, n;

printf("Enter number of rows: ");

scanf("%d", &m);

printf("Enter number of columns: ");

scanf("%d", &n);

int matrix[10][10];

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

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

printf("Enter value for matrix[%d][%d]: ", i + 1, j + 1);

scanf("%d", &matrix[i][j]);

}

}

printf("Saddle points found at: ");

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

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

if (isSaddlePoint(matrix, i, j, m, n)) {

printf("[%d][%d] ", i + 1, j + 1);

}

}

}

printf("\n");

return 0;

}

**29. Đếm số 0, tìm giá trị lớn nhất và nhỏ nhất trong ma trận**

#include <stdio.h>

void analyzeMatrix(int matrix[10][10], int m, int n) {

int zeroCount = 0, max = matrix[0][0], min = matrix[0][0];

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

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

if (matrix[i][j] == 0) zeroCount++;

if (matrix[i][j] > max) max = matrix[i][j];

if (matrix[i][j] < min) min = matrix[i][j];

}

}

printf("Number of zeros: %d\n", zeroCount);

printf("Maximum value: %d\n", max);

printf("Minimum value: %d\n", min);

}

int main() {

int m, n;

printf("Enter number of rows: ");

scanf("%d", &m);

printf("Enter number of columns: ");

scanf("%d", &n);

int matrix[10][10];

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

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

printf("Enter value for matrix[%d][%d]: ", i + 1, j + 1);

scanf("%d", &matrix[i][j]);

}

}

analyzeMatrix(matrix, m, n);

return 0;

}

**30. Cộng hai ma trận cùng kích thước**

#include <stdio.h>

void addMatrices(int A[10][10], int B[10][10], int C[10][10], int m, int n) {

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

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

C[i][j] = A[i][j] + B[i][j];

}

}

}

int main() {

int m, n;

printf("Enter the number of rows: ");

scanf("%d", &m);

printf("Enter the number of columns: ");

scanf("%d", &n);

int A[10][10], B[10][10], C[10][10];

printf("Enter elements for matrix A:\n");

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

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

scanf("%d", &A[i][j]);

printf("Enter elements for matrix B:\n");

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

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

scanf("%d", &B[i][j]);

addMatrices(A, B, C, m, n);

printf("Result matrix C (A + B):\n");

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

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

printf("%d ", C[i][j]);

}

printf("\n");

}

return 0;

}

**31. Nhân hai ma trận**

#include <stdio.h>

void multiplyMatrices(int A[10][10], int B[10][10], int C[10][10], int m, int n, int p) {

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

for (int j = 0; j < p; j++) {

C[i][j] = 0;

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

C[i][j] += A[i][k] \* B[k][j];

}

}

}

}

int main() {

int m, n, p;

printf("Enter the number of rows of matrix A: ");

scanf("%d", &m);

printf("Enter the number of columns of matrix A / rows of matrix B: ");

scanf("%d", &n);

printf("Enter the number of columns of matrix B: ");

scanf("%d", &p);

int A[10][10], B[10][10], C[10][10];

printf("Enter elements for matrix A:\n");

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

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

scanf("%d", &A[i][j]);

printf("Enter elements for matrix B:\n");

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

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

scanf("%d", &B[i][j]);

multiplyMatrices(A, B, C, m, n, p);

printf("Result matrix C (A \* B):\n");

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

for (int j = 0; j < p; j++) {

printf("%d ", C[i][j]);

}

printf("\n");

}

return 0;

}

**32. Ma trận xoắn ốc**

#include <stdio.h>

void generateSpiralMatrix(int n) {

int matrix[10][10], value = 1;

int top = 0, bottom = n - 1, left = 0, right = n - 1;

while (value <= n \* n) {

for (int i = left; i <= right; i++) matrix[top][i] = value++;

top++;

for (int i = top; i <= bottom; i++) matrix[i][right] = value++;

right--;

for (int i = right; i >= left; i--) matrix[bottom][i] = value++;

bottom--;

for (int i = bottom; i >= top; i--) matrix[i][left] = value++;

left++;

}

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

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

printf("%d ", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int n;

printf("Enter the size of the matrix: ");

scanf("%d", &n);

if (n <= 0) {

printf("The size of the matrix must be greater than 0!\n");

return 1;

}

generateSpiralMatrix(n);

return 0;

}

**34. Ma trận ma thuật (Magic Square)**

#include <stdio.h>

void generateMagicSquare(int n) {

int magic[10][10];

int i = 0, j = n / 2;

for (int num = 1; num <= n \* n; num++) {

magic[i][j] = num;

int newI = (i - 1 + n) % n;

int newJ = (j + 1) % n;

if (magic[newI][newJ]) {

i = (i + 1) % n;

} else {

i = newI;

j = newJ;

}

}

printf("Magic Square of size %d:\n", n);

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

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

printf("%d ", magic[i][j]);

}

printf("\n");

}

}

int main() {

int n;

printf("Enter the size of the magic square (odd number): ");

scanf("%d", &n);

if (n % 2 == 0 || n <= 0) {

printf("Size must be a positive odd number!\n");

return 1;

}

generateMagicSquare(n);

return 0;

}

**Bài 35 – Đếm số ao trong khu đất ngập lụt** được triển khai bằng ngôn ngữ C. Bài này yêu cầu tìm các ao trong một ma trận, với mỗi ô có giá trị W (ngập lụt) hoặc . (đất khô). Ao được định nghĩa là các ô W kết nối với nhau qua các cạnh. Dưới đây là mã hoàn chỉnh:

**35. Đếm số ao trong khu đất ngập lụt**

#include <stdio.h>

#define MAX 100

// Các bước di chuyển trong 4 hướng: trên, dưới, trái, phải

int dx[] = {-1, 1, 0, 0};

int dy[] = {0, 0, -1, 1};

// Hàm DFS để đánh dấu tất cả các ô kết nối với ô (x, y)

void dfs(char land[MAX][MAX], int x, int y, int m, int n) {

land[x][y] = '.'; // Đánh dấu ô này là đã thăm (bỏ W)

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

int newX = x + dx[i];

int newY = y + dy[i];

// Kiểm tra nếu ô mới nằm trong giới hạn và là 'W'

if (newX >= 0 && newX < m && newY >= 0 && newY < n && land[newX][newY] == 'W') {

dfs(land, newX, newY, m, n);

}

}

}

int countPonds(char land[MAX][MAX], int m, int n) {

int ponds = 0;

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

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

if (land[i][j] == 'W') {

ponds++;

dfs(land, i, j, m, n); // Đánh dấu toàn bộ ao này

}

}

}

return ponds;

}

int main() {

int m, n;

printf("Enter the number of rows: ");

scanf("%d", &m);

printf("Enter the number of columns: ");

scanf("%d", &n);

char land[MAX][MAX];

printf("Enter the land matrix (%d x %d):\n", m, n);

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

scanf("%s", land[i]);

}

int result = countPonds(land, m, n);

printf("There are %d ponds on the land.\n", result);

return 0;

}

**Giải thích mã**

1. **Ma trận land[][]**: Đây là ma trận chứa các ô, với mỗi ô có giá trị 'W' (nước) hoặc '.' (đất khô).
2. **DFS**: Dùng để duyệt tất cả các ô liên thông với ô bắt đầu có giá trị 'W'. Khi tìm được ô 'W', nó sẽ được đánh dấu là '.' để tránh tính lại.
3. **Hàm countPonds**: Duyệt qua toàn bộ ma trận và đếm số lượng ao (các vùng nước liên thông).
4. **Nhập dữ liệu**: Người dùng nhập kích thước và nội dung của ma trận.

**Ví dụ chạy chương trình**

mathematica

Sao chép mã

Enter the number of rows: 4

Enter the number of columns: 6

Enter the land matrix (4 x 6):

W.W.WW

W.WW..

...W..

WW....

There are 3 ponds on the land.