**Вариант 2**

Задать случайно значения элементов вещественной матрицы размера *n×m.*

1) Упорядочить элементы в каждой строке по убыванию методом выбора.

2) Затем упорядочить элементы в столбцах матрицы, используя метод вставки.

Вывести на экран массив перед сортировкой и после каждого упорядочивания.

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#define MAX\_SIZE 100**

**// Функция для вывода матрицы**

***void* print\_matrix(*float* *matrix*[MAX\_SIZE][MAX\_SIZE], *int* *n*, *int* *m*) {**

**printf("Matrix:\n");**

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

**for (*int* j = 0; j < *m*; j++) {**

**printf("%.2f ", *matrix*[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\n");**

**}**

**// Функция для сортировки строки методом выбора**

***void* selection\_sort\_row(*float* *matrix*[MAX\_SIZE][MAX\_SIZE], *int* *row*, *int* *m*) {**

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

***int* min\_index = i;**

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

**if (*matrix*[*row*][j] < *matrix*[*row*][min\_index]) {**

**min\_index = j;**

**}**

**}**

**// Обмен элементов**

***float* temp = *matrix*[*row*][i];**

***matrix*[*row*][i] = *matrix*[*row*][min\_index];**

***matrix*[*row*][min\_index] = temp;**

**}**

**}**

**// Функция для сортировки всех строк**

***void* sort\_rows(*float* *matrix*[MAX\_SIZE][MAX\_SIZE], *int* *n*, *int* *m*) {**

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

**selection\_sort\_row(*matrix*, i, *m*);**

**printf("Matrix after row sorting %d:\n", i + 1);**

**print\_matrix(*matrix*, *n*, *m*);**

**}**

**}**

**// Функция для сортировки столбца методом вставки**

***void* insertion\_sort\_column(*float* *matrix*[MAX\_SIZE][MAX\_SIZE], *int* *col*, *int* *n*) {**

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

***float* key = *matrix*[i][*col*];**

***int* j = i - 1;**

**while (j >= 0 && *matrix*[j][*col*] > key) {**

***matrix*[j + 1][*col*] = *matrix*[j][*col*];**

**j--;**

**}**

***matrix*[j + 1][*col*] = key;**

**}**

**}**

**// Функция для сортировки всех столбцов**

***void* sort\_columns(*float* *matrix*[MAX\_SIZE][MAX\_SIZE], *int* *n*, *int* *m*) {**

**for (*int* j = 0; j < *m*; j++) {**

**insertion\_sort\_column(*matrix*, j, *n*);**

**printf("Matrix after column sorting %d:\n", j + 1);**

**print\_matrix(*matrix*, *n*, *m*);**

**}**

**}**

***int* main() {**

***int* n, m;**

***float* matrix[MAX\_SIZE][MAX\_SIZE];**

**printf("Enter the number of rows (n) and columns (m): ");**

**scanf("%d %d", &n, &m);**

**// Инициализация генератора случайных чисел**

**srand(time(0));**

**// Генерация случайных значений для матрицы**

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

**for (*int* j = 0; j < m; j++) {**

**matrix[i][j] = (*float*)(rand() % 100) / 10.0; // Случайные числа от 0 до 9.9**

**}**

**}**

**printf("Original matrix:\n");**

**print\_matrix(matrix, n, m);**

**// 1. Сортировка строк**

**sort\_rows(matrix, n, m);**

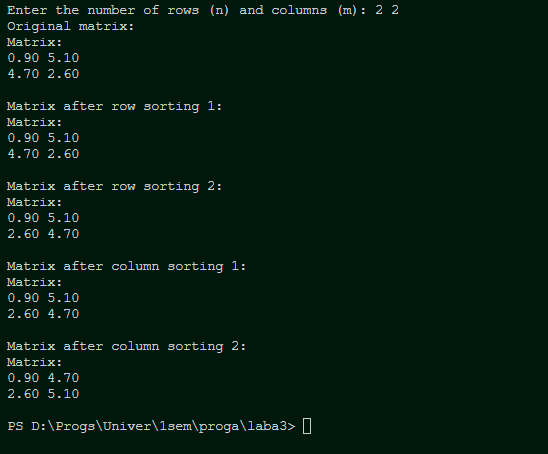
**// 2. Сортировка столбцов**

**sort\_columns(matrix, n, m);**

**return 0;**

**}**

Результат



Разбор кода

Заголовочные файлы

#include <stdio.h>

Подключает стандартную библиотеку ввода-вывода. Это позволяет использовать функции printf для вывода данных на экран и scanf для чтения данных с клавиатуры.

#include <stdlib.h>

Подключает библиотеку, содержащую функции для работы с памятью и генерации случайных чисел, такие как rand() и srand().

#include <time.h>

Подключает библиотеку для работы с временными значениями, используемую для инициализации генератора случайных чисел с помощью srand(time(0));.

#define MAX\_SIZE 100

Определяет константу MAX\_SIZE, равную 100, которая задает максимальные размеры матрицы (максимум 100 строк и 100 столбцов).

Функция для вывода матрицы

void print\_matrix(float matrix[MAX\_SIZE][MAX\_SIZE], int n, int m) {

Объявляет функцию print\_matrix, которая будет принимать двумерный массив matrix и его размеры n (количество строк) и m (количество столбцов).

printf("**Matrix**:\n");

Выводит текст "Матрица:", подготавливая пользователя к тому, что будет показано содержимое матрицы.

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

Начинает цикл, который будет проходить по каждой строке матрицы от 0 до n-1.

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

Вложенный цикл, который проходит по каждому столбцу текущей строки от 0 до m-1.

printf("%.2f ", matrix[i][j]);

Выводит значение элемента матрицы с форматированием: число с двумя знаками после запятой, добавляя пробел после каждого числа.

printf("\n");

После вывода всех элементов строки добавляет перевод строки, чтобы следующая строка матрицы начиналась с новой строки.

printf("\n");

}

Добавляет пустую строку для лучшей визуальной ясности и завершает функцию print\_matrix.

Функция сортировки строки методом выбора

void selection\_sort\_row(float matrix[MAX\_SIZE][MAX\_SIZE], int row, int m) {

Определяет функцию selection\_sort\_row, которая сортирует одну строку матрицы с индексом row методом выбора. Параметр m — количество элементов в строке.

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

Начинает цикл, который проходит по всем элементам строки, кроме последнего (поскольку последний элемент уже будет на своем месте после всех сравнений).

int min\_index = i;

Инициализирует переменную min\_index, устанавливая ее равной текущему индексу i. Это предполагает, что текущий элемент является минимальным.

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

Вложенный цикл, который ищет минимальный элемент в оставшейся части строки, начиная с элемента, следующего за i.

if (matrix[row][j] < matrix[row][min\_index]) {

min\_index = j;

}

Если текущий элемент меньше, чем элемент с индексом min\_index, обновляет min\_index до текущего индекса j.

// Обмен элементов

float temp = matrix[row][i];

Сохраняет значение текущего элемента в временной переменной temp.

matrix[row][i] = matrix[row][min\_index];

Меняет текущий элемент с минимальным элементом (который был найден ранее) в строке.

matrix[row][min\_index] = temp;

Устанавливает значение из temp в позицию минимального элемента, завершив обмен элементов.

}

Закрывает функцию selection\_sort\_row.

Функция для сортировки всех строк

void sort\_rows(float matrix[MAX\_SIZE][MAX\_SIZE], int n, int m) {

Объявляет функцию sort\_rows, которая сортирует все строки матрицы. Параметры: матрица, n — количество строк и m — количество столбцов.

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

Начинает цикл по всем строкам от 0 до n-1.

selection\_sort\_row(matrix, i, m);

Вызывает функцию selection\_sort\_row для сортировки строки с индексом i.

printf("Matrix after row sorting%d:\n", i + 1);

Выводит сообщение о том, что текущая строка была отсортирована.

print\_matrix(matrix, n, m);

Вызывает функцию print\_matrix, чтобы продемонстрировать текущее состояние матрицы после сортировки строки.

}

Закрывает функцию sort\_rows.

Функция для сортировки столбца методом вставки

void insertion\_sort\_column(float matrix[MAX\_SIZE][MAX\_SIZE], int col, int n) {

Определяет функцию insertion\_sort\_column для сортировки конкретного столбца с индексом col. Параметр n — количество элементов в столбце.

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

Начинает цикл с 1, поскольку первый элемент уже считается отсортированным.

float key = matrix[i][col];

Сохраняет значение текущего элемента в переменной key.

int j = i - 1;

Устанавливает j как индекс предыдущего элемента.

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

Запускает цикл, который продолжается, пока текущий элемент больше key. Это позволяет сдвигать элементы вниз, чтобы освободить место.

matrix[j + 1][col] = matrix[j][col];

j--;

Сдвигает элементы вниз на одну позицию.

}

matrix[j + 1][col] = key;

Вставляет key на освобожденную позицию.

}

Закрывает функцию insertion\_sort\_column.

Функция для сортировки всех столбцов

void sort\_columns(float matrix[MAX\_SIZE][MAX\_SIZE], int n, int m) {

Объявляет функцию sort\_columns, которая сортирует каждый столбец матрицы. Принимает параметры: matrix, n, и m.

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

Начинает цикл по всем столбцам от 0 до m - 1.

insertion\_sort\_column(matrix, j, n);

Вызывает insertion\_sort\_column, чтобы отсортировать столбец с индексом j.

printf("Matrix after column sorting %d:\n", j + 1);

Выводит сообщение о том, что текущий столбец был отсортирован.

print\_matrix(matrix, n, m);

Вызывает функцию print\_matrix, чтобы показать текущее состояние матрицы после сортировки столбца.

}

Закрывает функцию sort\_columns.

Основная функция

int main() {

Начинает основную часть программы с функции main, которая является точкой входа.

int n, m;

Объявляет переменные n и m для хранения количества строк и столбцов матрицы.

float matrix[MAX\_SIZE][MAX\_SIZE];

Объявляет двумерный массив matrix, который будет хранить элементы матрицы.

printf("Введите количество строк (n) и столбцов (m): ");

Печатает инструкцию для пользователя, чтобы он ввел количество строк и столбцов матрицы.

Считывает введенные значения и сохраняет их в переменные n и m.

srand(time(0));

Инициализирует генератор случайных чисел, чтобы каждый раз при запуске программы получались разные случайные числа.

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

Начинает цикл по строкам от 0 до n-1.

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

Вложенный цикл по столбцам от 0 до m-1.

matrix[i][j] = (float)(rand() % 100) / 10.0; // Случайные числа от 0 до 9.9

Генерирует случайное число от 0 до 99, делит его на 10, чтобы получить дробное число от 0 до 9.9 и присваивает его текущему элементу матрицы.

}

}

Закрывает циклы для заполнения матрицы.

printf("Original matrix:\n");

Печатает сообщение о том, что будет показана исходная матрица.

print\_matrix(matrix, n, m);

Вызывает функцию print\_matrix, чтобы показать исходное состояние матрицы.

sort\_rows(matrix, n, m);

Вызывает функцию sort\_rows, чтобы отсортировать все строки матрицы с использованием метода выбора.

sort\_columns(matrix, n, m);

Вызывает функцию sort\_columns, чтобы отсортировать все столбцы матрицы, используя метод вставки.

return 0;

}

Завершает выполнение функции main, возвращая 0, что обычно означает успешное выполнение программы.