|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Двумерный массив»** | |
|  | |
| Выполнил студент группы ИКБО-72-23 | Шатохин Б.А. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

1. Цель работы

Получение навыков по определению двумерного массива для структуры данных задачи. Получение навыков по разработке алгоритмов операций на двумерном массиве в соответствии с задачей.

1. Постановка задачи

Разработать программу по обработке данных, представленных в задаче матрицей и реализованной в программе двумерным (многомерным) динамическим массивом.

1. Размеры массива должны определяться пользователем.
2. Двумерный массив определить как двойной указатель и выполнить его создание операцией new.
3. Разработать функции по реализации алгоритмов заполнения матрицы: с клавиатуры, датчиком случайных чисел. Разработать функции по реализации алгоритма вывода массива на экран построчно.
4. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
5. Разработать программу, демонстрирующую работу всех функций.

Разработать программу по задаче варианта с реализацией данных задачи с применением контейнера vector библиотеки STL.

1. Реализовать структуру хранения данных на основе шаблона <vector>, размеры определить при вводе с клавиатуры.
2. Разработать функции: заполнение структуры хранения исходных данных, вывода структуры хранения.
3. Выполнить декомпозицию задачи варианта, разработать алгоритм решения. Реализовать функцию, выполняющую задачу и отладить ее.
4. Разработать программу, демонстрирующую работу всех функций

Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования. При тестировании геометрической задачи наглядно изобразить решение на рисунке для проверки.

Индивидуальное задание 1:

Дана квадратная матрица, состоящая из натуральных чисел. Зеркально отразить (транспонировать) ее элементы относительно побочной диагонали.

Индивидуальное задание 2:

Дано множество из n (нечетное) точек на плоскости. Найти горизонтальную медиану этого множества точек на плоскости в предположении, что никакие две точки не лежат на одной прямой.

Медианой множества называется прямая, которая делит множество на два подмножества одинаковой мощности. Медиана строится через точку этого множества.

1. Решение

Двумерный массив — это массив, в котором для определения местоположения элемента в массиве нужно указать значения двух индексов. Двумерный массив ассоциируется с таблицей. Если количество столбцов и строк массива (таблицы) одинаковое, то массив еще называют матрицей. Остальную теорию по массивам смотри в отчёте по первой работе.

Для заполнения массива используем функцию matrixInput. Функция принимает три аргумента: сам массив, размер массива и режим в котором он будет работать.

int matrixInput(int\*\* matrix, int matrix\_size, int mode) {

switch (mode) {

case 0:

for (int i = 0; i < matrix\_size; i++) {

for (int j = 0; j < matrix\_size; j++) {

std::cin >> matrix[i][j];

}

}

break;

case 1:

for (int i = 0; i < matrix\_size; ++i) {

for (int j = 0; j < matrix\_size; ++j) {

matrix[i][j] = 1 + std::rand() % 99;

}

}

break;

default:

return -1;

}

return 0;

}

Для вывода данных из двумерного массива используется функция matrixOutput. На входе принимает матрицу и размер матрицы.

Для транспонирования матрицы была разработана функция tranposeMaxtrix. На вход принимает три аргумента: матрицу, транспонированную матрицу (мы создали её заранее, но в ней хранятся «мусорные» значения) и размер матрицы.

int tranposeMatrix(int\*\* original\_matrix, int\*\* transposed\_matrix, int matrix\_size) {

for (int i = 0; i < matrix\_size; ++i) {

for (int j = 0; j < matrix\_size; j++) {

transposed\_matrix[i][j] = original\_matrix[j][i];

}

}

return 1;

}

void matrixOutput(int\*\* matrix, int matrix\_size) {

for (int i = 0; i < matrix\_size; ++i) {

for (int j = 0; j < matrix\_size; ++j) {

std::cout << matrix[i][j];

}

}

}

Для выполнения второго задания была использована структура point. Внутри этой структуры находятся два поля x и y.

struct point {

int x = 0;

int y = 0;

};

Так же для создания наших точек была создана функция addDots, которая принимает вектор объектов структуры point, а также количество точек.

std::vector<point> addDots(std::vector<point> plurality, int n) {

point dot;

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

std::cout << "Введите x,y\n";

std::cin >> dot.x;

std::cin >> dot.y;

plurality.push\_back(dot);

}

return plurality;

}

Для того чтобы показать результат второго задания была разработана функция vectorOutput, так же принимающая на вход вектор и количество точек.

Для нахождения медианы была разработана функция countMedian, так же принимающая на вход вектор и количество точек.

float countMedian(std::vector<point> plurality, int n) {

float median;

int temp\_x, temp\_y, median\_index = n/2;

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

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

if (plurality[j].x > plurality[j + 1].x) {

temp\_x = plurality[j + 1].x;

temp\_y = plurality[j + 1].y;

plurality[j + 1].x = plurality[j].x;

plurality[j + 1].y = plurality[j].y;

plurality[j].x = temp\_x;

plurality[j].y = temp\_y;

}

}

}

median = plurality[median\_index].x;

return median;

}

void vectorOutput(std::vector<point> plurality, int n) {

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

std::cout << "x: " << plurality[i].x << "\t" << "y: " << plurality[i].y;

}

}

1. Тестирование

При запуске программы нас просят ввести размер матрицы в консоль. Для удобства демонстрации работы программы выберу размер равный двум (квадратная матрица с двумя столбцами и двумя строками). После ввода размера матрицы пользователю на выбор предлагается на выбор шесть пунктов меню.

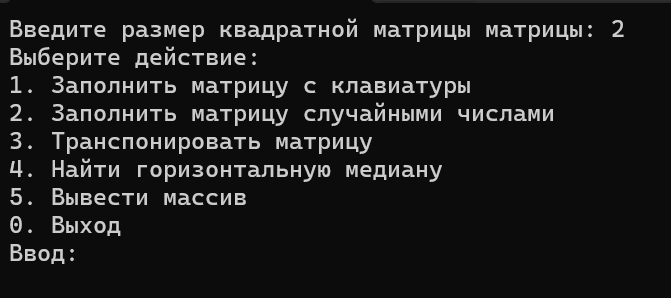


Рисунок 1. Демонстрация меню программы

Выберем первый пункт, заполнив матрицу числами 1 2 3 4. Сразу после этого выведем получившуюся матрицу в консоль.

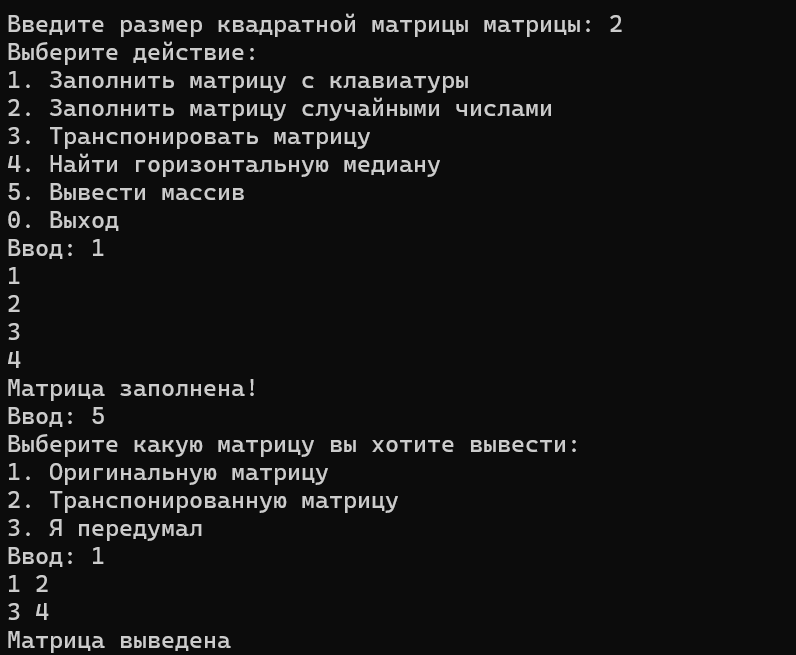


Рисунок 2. Демонстрация работы ввода и вывода

Повторим аналогичные действия, но для матрицы размером 3 х 3, однако для заполнения используем функцию случайного заполнения массива, которая была реализована заранее. Как видим, матрица была заполнена числами от 0 до 100 и выведена согласно её алгебраическому представлению (в виде некой таблицы).

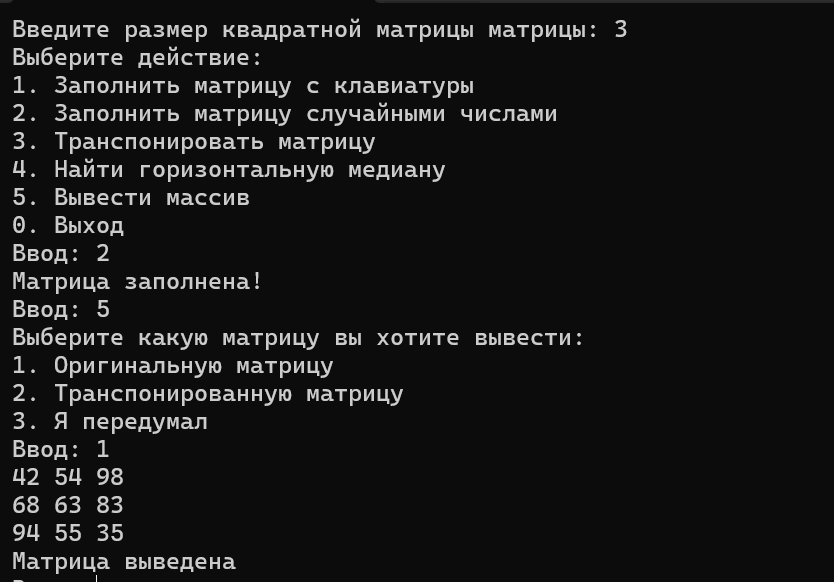


Рисунок 3. Демонстрация работы автоматического заполнения

Теперь, согласно индивидуальному варианту, транспонирую матрицу. Для уменьшения количества запуска программы я буду работать с уже созданной случайным образом матрицей. После транспонирования, выведем новую матрицу и сравним. Как видим, столбцы матрицы стали строками, а строки столбцами, следовательно транспонирование прошло успешно.

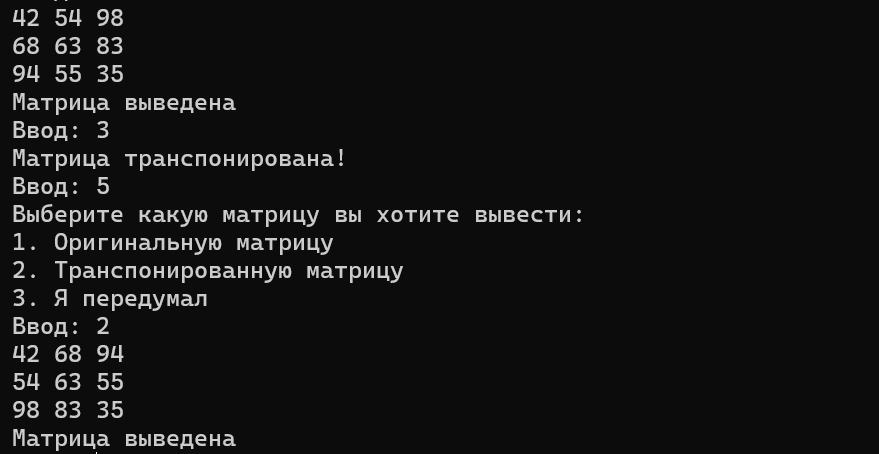


Рисунок 4. Демонстрация транспонирования матрицы

Не забудем про исключение, которое было установлено вторым заданием. По его условию, количество точек должно быть больше 2 и не должно быть чётное количество. Введём количество точек равное 4. Как видим, произошла обработка исключения.

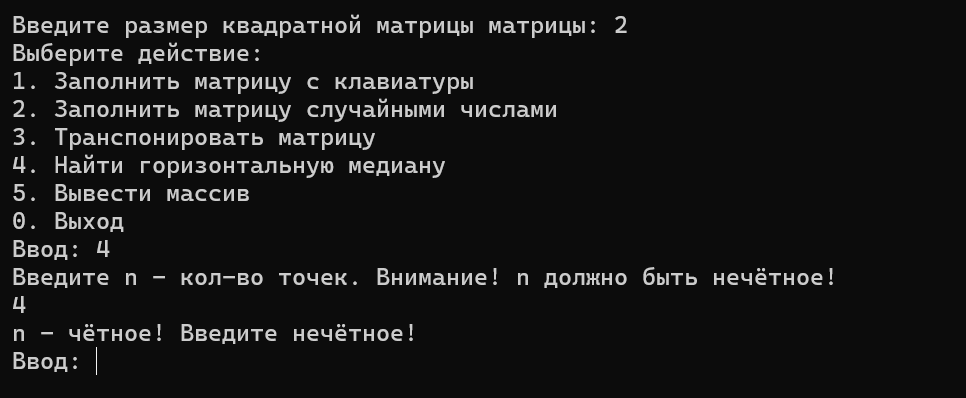


Рисунок 5. Демонстрация обработки исключения для второго задания

Теперь введём количество точек равное трём. Введём абсциссу и ординату каждой точки и выведем их медиану.

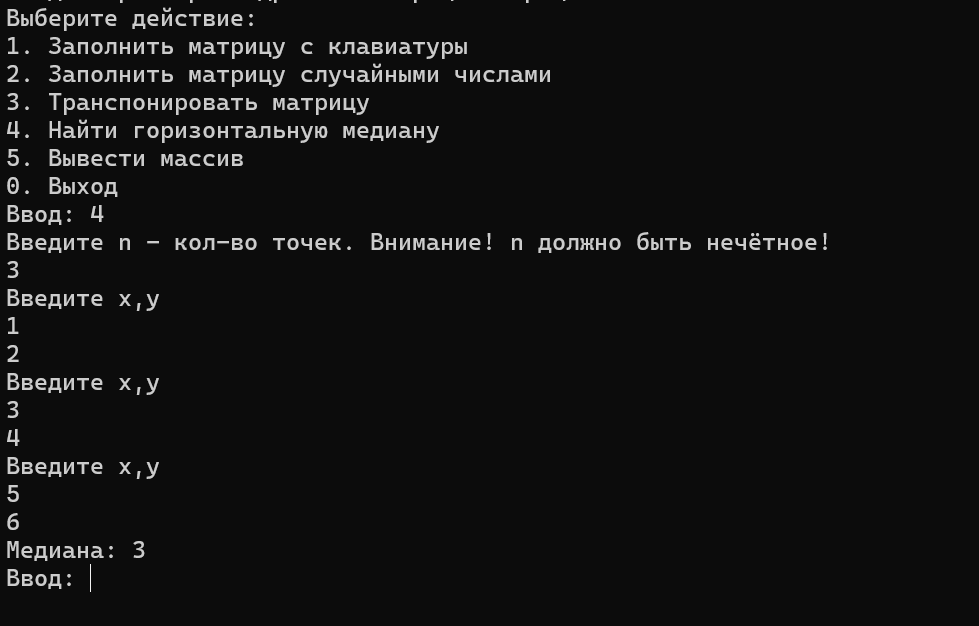


Рисунок 6. Демонстрация выполнения второго задания

1. Вывод

Были изучены двумерные массивы, получены навыки создания алгоритмов по работе с двумерными массивами.

1. Исходный код

int main() {

setlocale(LC\_ALL, "Ru");

int matrix\_size, menu2 , n, median;

char menu;

std::vector <point> plurality;

point dot;

std::cout << "Введите размер квадратной матрицы матрицы: ";

std::cin >> matrix\_size;

int\*\* matrix = new int\* [matrix\_size];

for (int i = 0; i < matrix\_size; ++i) {

matrix[i] = new int[matrix\_size];

}

int\*\* transposed\_matrix = new int\* [matrix\_size];

for (int i = 0; i < matrix\_size; ++i) {

transposed\_matrix[i] = new int[matrix\_size];

}

std::cout << "Выберите действие: \n"

<< "1. Заполнить матрицу с клавиатуры\n"

<< "2. Заполнить матрицу случайными числами\n"

<< "3. Транспонировать матрицу\n"

<< "4. Найти горизонтальную медиану\n"

<< "5. Вывести массив\n"

<< "0. Выход\n";

while (true) {

std::cout << "Ввод: ";

std::cin >> menu;

switch (menu) {

case '0':

std::cout << "Программа звершена!\n";

return 1;

case '1':

matrixInput(matrix, matrix\_size, 0);

std::cout << "Матрица заполнена!\n";

break;

case '2':

matrixInput(matrix, matrix\_size, 1);

std::cout << "Матрица заполнена!\n";

break;

case '3':

tranposeMatrix(matrix, transposed\_matrix, matrix\_size);

std::cout << "Матрица транспонирована!\n";

break;

case '4':

std::cout << "Введите n - кол-во точек. Внимание! n должно быть нечётное!\n";

std::cin >> n;

if (n % 2 == 0) {

std::cout << "n - чётное! Введите нечётное!\n";

break;

}

else {

plurality = addDots(plurality, n);

median = countMedian(plurality, n);

std::cout << "Медиана: " << median << "\n";

break;

}

case '5':

std::cout << "Выберите какую матрицу вы хотите вывести:\n"

<< "1. Оригинальную матрицу\n"

<< "2. Транспонированную матрицу\n"

<< "3. Я передумал\n";

std::cout << "Ввод: ";

std::cin >> menu2;

switch (menu2)

{

case 1:

matrixOutput(matrix, matrix\_size);

std::cout << "Матрица выведена\n";

break;

case 2:

matrixOutput(transposed\_matrix, matrix\_size);

std::cout << "Матрица выведена\n";

break;

case 3:

std::cout << "Ну ладно:(\n";

break;

default:

std::cout << "Такого варианта нет!\n";

break;

}

break;

default:

std::cout << "Неверно, попробуй ещё раз!\n";

std::cout << "Выберите действие: \n"

<< "1. Заполнить матрицу с клавиатуры\n"

<< "2. Заполнить матрицу случайными числами\n"

<< "3. Транспонировать матрицу\n"

<< "4. Найти горизонтальную медиану\n"

<< "5. Вывести массив\n"

<< "0. Выход\n";

}

}

}

case '5':

std::cout << "Выберите какую матрицу вы хотите вывести:\n"

<< "1. Оригинальную матрицу\n"

<< "2. Транспонированную матрицу\n"

<< "3. Я передумал\n";

std::cout << "Ввод: ";

std::cin >> menu2;

switch (menu2)

{

case 1:

matrixOutput(matrix, matrix\_size);

std::cout << "Матрица выведена\n";

break;

case 2:

matrixOutput(transposed\_matrix, matrix\_size);

std::cout << "Матрица выведена\n";

break;

case 3:

std::cout << "Ну ладно:(\n";

break;

default:

std::cout << "Такого варианта нет!\n";

break;

}

break;

default:

std::cout << "Неверно, попробуй ещё раз!\n";

std::cout << "Выберите действие: \n"

<< "1. Заполнить матрицу с клавиатуры\n"

<< "2. Заполнить матрицу случайными числами\n"

<< "3. Транспонировать матрицу\n"

<< "4. Найти горизонтальную медиану\n"

<< "5. Вывести массив\n"

<< "0. Выход\n";

}

}

}