МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

(факультет)

Кафедра Графики, конструирования и информационных технологий в промышленном дизайне

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Основы программирования и алгоритмизации» .

Тема: «Реализация динамического многомерного массива» .

Расчетно-пояснительная записка

Разработал студент М.Е. Есаков

Подпись, дата Инициалы, фамилия

Руководитель О.В. Курипта

Подпись, дата Инициалы, фамилия

Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер О.В. Курипта

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

ВОРОНЕЖ

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра Графики, конструирования и информационных технологий в промышленном дизайне

ЗАДАНИЕ

на курсовой проект

по дисциплине: «Основы Программирования и Алгоритмизации» .

Тема: «Реализация динамического многомерного массива» .

. .

Студент бИД-241 Есаков Максим Евгеньевич .

Группа, фамилия, имя, отчество

Номер варианта 10 .

Технические условия:.

.

Содержание и объем работы (графические работы, расчеты и прочее):. .

. стр., иллюстр., табл. .

Сроки выполнения этапов:.анализ и постановка задачи (01.10-15.10); разработка пошаговой детализации программы (16.10-10.11); реализация программы (11.11-01.12); тестирование программы (01.12-10.12); оформление пояснительной записки (02.12-12.12).

.

Срок защиты курсового проекта: .

Руководитель . О.В. Курипта .

Подпись, дата Инициалы, фамилия

Задание принял студент . М.Е. Есаков .

Подпись, дата Инициалы, фамилия

Замечания руководителя

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc185466458)

[1. Постановка задачи 7](#_Toc185466459)

[2 Конструирование программы 8](#_Toc185466460)

[2.1 Пошаговая детализация программы 8](#_Toc185466461)

[2.2 Описание функций 10](#_Toc185466462)

[3 Тестирование программы 15](#_Toc185466463)

[ЗАКЛЮЧЕНИЕ 16](#_Toc185466464)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc185466465)

[ПРИЛОЖЕНИЕ 18](#_Toc185466466)

# ВВЕДЕНИЕ

В современном мире матричные вычисления имеют огромное значение в науке и технике. Они используются в машинном обучении, обработке изображений и решении систем уравнений. При работе с большими массивами данных крайне важен выбор эффективных структур и алгоритмов.

Эта работа актуальна, так как эффективное преобразование матриц - ключевой момент в производительности многих вычислительных задач. Использование динамических массивов, позволяющих гибко управлять памятью, позволяет создавать матрицы любого размера и оптимизировать работу программы.

Цель работы – разработать программу для работы с динамическими многомерными массивами и реализацией преобразования матриц. Для этого необходимо:

1. Выбрать подходящую структуру данных для представления матриц с динамическим выделением памяти.
2. Реализовать эффективный алгоритм преобразования матриц.
3. Создать пользовательский интерфейс для ввода и вывода матриц.

Таким образом, разработка программы для преобразования матриц с динамическими массивами важна для повышения производительности вычислительных задач в различных областях.

## 1. Постановка задачи

Программа по реализации динамического многомерного массива должна выполнять следующие функции:

1. Предоставление возможности пользователю инициализировать многомерный массив с консоли или случайным образом.
2. Интерфейс программы должен обеспечивать возможность пользователю вывести полученный результат по его выбору, завершить работу с программой или продолжить. Результаты должны сопровождаться комментариями.
3. Выделение памяти под массив с помощью функции malloc(..) или calloc(..).
4. Реализовать перестановку строк и столбцов таким образом, чтобы в левом верхнем углу и правом нижнем углу матрицы сформировались квадраты из четырех единиц.
5. Реализовать возможность повторного выполнения программы, чтобы пользователь мог вводить новые матрицы или генерировать их случайным образом без перезапуска программы.
6. Обеспечить корректное освобождение выделенной памяти после завершения работы программы, чтобы избежать утечек памяти.

Все задачи были поставлены. Далее начнётся конструирование самой программы, где наглядно будет показан алгоритм и ветвления программы. Алгоритм и ветвления будут проиллюстрированы с помощью блок-схем.

## 2 Конструирование программы

### 2.1 Пошаговая детализация программы

Подключение библиотек:

1. <stdio.h> для ввода и вывода данных;
2. <stdlib.h> для работы с динамической памятью и генерацией случайных чисел;
3. <locale.h> для корректного отображения русских букв;
4. <time.h> для работы с функцией генерации случайных чисел.

Объявление функций:

1. zapolnenie\_matrix - функция для заполнения матрицы вручную;
2. zapolnenie\_matrix\_randomno - функция для заполнения матрицы случайными значениями;
3. one\_or\_null - функция для проверки возможности формирования квадрата из единиц;
4. replace\_matrix - функция для замены нулей на единицы;
5. print\_matrix - функция для печати матрицы;

Интерфейс программы:

1. Главное меню:

1.1. Выбор операции (ввод матриц вручную или генерация случайных матриц или выход из программы) и запись ответа в переменную option с помощью функции scanf(), которая считывает написанную пользователем цифру.

int option – переменная, для записи выбора пользователя (цифра 1/2/3)

1. Выделяется память под матрицу. Матрица представлена как массив указателей на строки. Проверка выделения памяти (при ошибке выводится сообщение и завершается выполнение программы);
2. Выполнение операции:

3.1. Если пользователь выбрал 1, вызывается функция zapolnenie\_matrix для ввода матрицы вручную.

3.2. Если пользователь выбрал 2, программа генерирует случайную матрицу. Сначала инициализируется генератор случайных чисел с помощью srand(time(NULL)), а затем вызывается функция zapolnenie\_matrix\_randomno для заполнения матрицы случайными числами в диапазоне от 0 до 1, используя функцию rand().

3.3. Если пользователь выбрал 3, то осуществляется выход из программы.

3.3. Если введено некорректное значение, программа выводит сообщение «Неверный выбор» и возвращается к началу цикла.

1. Вызывается функция replace\_matrix, которая реализовывает перестановку строк и столбцов таким образом, чтобы в левом верхнем углу и правом нижнем углу матрицы сформировались квадраты из четырех единиц.
2. Вызывается функция one\_or\_null, которая проверяет на правильность выполнения функции replace\_matrix.
3. Программа выводит начальную и измененную матрицы с помощью функции print\_matrix. Каждая матрица выводится в виде таблицы, что позволяет пользователю легко воспринимать информацию;
4. После завершения всех операций программа освобождает выделенную память для матрицы.
5. Программа спрашивает у пользователя, хочет ли он продолжить работу. Если пользователь вводит 1, программа возвращается в главное меню, иначе программа завершается. Цикл do...while обеспечивает возможность многократного выполнения программы.

### 2.2 Описание функций

1. Функция int\*\* zapolnenie\_matrix(int\*\* matrix, int size) отвечает за ввод элементов матрицы с клавиатуры. Она принимает указатель на двумерный массив целых чисел (int \*\*matrix) и размер массива (int size).

int \*\*matrix – указатель на двумерный массив, в который будут записаны введённые пользователем значения.

int size – размерность матрицы

Функция использует вложенные циклы for для перебора всех элементов матрицы. Внутренний цикл отвечает за перебор элементов строки, а внешний за перебор строк (Рисунок 1).

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 1 – функция для заполнения матрицы

2. Функция zapolnenie\_matrix\_randomno(int\*\* matrix, int size) генерирует случайную матрицу с целыми числами в заданном диапазоне. Она принимает указатель на двумерный массив (int \*\*matrix) и размер массива (int size).

int \*\*matrix – указатель на двумерный массив, который будет заполнен случайными числами

int size – размерность матрицы

Функция использует вложенные циклы for для перебора всех элементов матрицы. Для каждого элемента она генерирует случайное число в диапазоне от 0 до 1, используя выражение matrix[i][j] = rand() % 2. Это выражение обеспечивает равномерное распределение случайных чисел в заданном диапазоне. Сгенерированное число записывается в соответствующий элемент матрицы (Рисунок 2).

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 2 – функция для заполнения матрицы случайными числами

3. Функция int\*\* one\_or\_null(int\*\* matrix, int size) выполняет проверку на правильность изменения матрицы. Она принимает указатель на двумерный массив целых чисел (int \*\*matrix) и размер массива (int size).

int \*\*matrix – указатель на двумерный массив, в который будут записаны введённые пользователем значения.

int size – размерность матрицы

Функция использует два цикла if для реализации алгоритма проверки реализации перестановки строк и столбцов таким образом, чтобы в левом верхнем углу и правом нижнем углу матрицы сформировались квадраты из четырех единиц (Рисунок 3).

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 3 – функция для проверки правильности реализации программы

4. Функция int\*\* replace\_matrix(int\*\* matrix, int size) пересталяет строки и столбцы таким образом, чтобы чтобы в левом верхнем углу и правом нижнем углу матрицы сформировались квадраты из четырех единиц. Она принимает указатель на двумерный массив целых чисел (int \*\*matrix) и размер массива (int size).

int \*\*matrix – указатель на двумерный массив, в который будут записаны введённые пользователем значения.

int size – размерность матрицы

Функция использует вложенные циклы for и if для перебора и перестановки элементов массива (Рисунок 4).

Изображение выглядит как снимок экрана, линия, Красочность, диаграмма

Автоматически созданное описание

Рисунок 4 – функция для преобразования матрицы

5. Функция void print\_matrix(int\*\* matrix, int size) выводит матрицу на экран. Она принимает указатель на двумерный массив (int \*\*matrix) и размер массива (int size).

int \*\*matrix – указатель на матрицу, которую нужно вывести

int size – размерность матрицы

Функция использует вложенные циклы for для перебора всех элементов матрицы (Рисунок 5).

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 5 – функция для печати матрицы

6. Функция void main() играет ключевую роль в программе, координируя весь процесс выполнения. Она управляет взаимодействием с пользователем, обработкой ввода данных, выполнением операций с матрицами и управлением памятью. Давайте подробнее рассмотрим каждый этап работы функции main().

Используются функции setlocale(LC\_CTYPE, "RUS") и srand(time(NULL)) для установки локали и генерации чисел. Происходит динамическое выделение памяти. Объявляются переменная option типа int.

Весь блок кода, начиная с вывода приветственного сообщения и заканчивая запросом о продолжении, заключен в цикл do...while. Это позволяет пользователю выполнить программу несколько раз, не перезапуская её.

Выводится приветственное сообщение с информацией о курсовом проекте.

Выводится главное меню, предлагающее пользователю три варианта: ввести матрицу вручную, сгенерировать ее случайным образом или завершить программу.

Используется free для освобождения памяти, выделенной под матрицу. Это предотвращает утечки памяти, что очень важно для корректной работы программы. Освобождение памяти происходит в обратном порядке выделения: сначала освобождаются строки матриц, затем сами массивы указателей.

Программа запрашивает у пользователя, хочет ли он продолжить работу.

## 3 Тестирование программы

При запуске программы выводится ФИО студента, группа, название программы и главное меню с функциями программы (Рисунок 5).

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание**

Рисунок 5 – меню программы

Навигация по главному меню программы осуществляется путем выбора пункта меню вводом с клавиатуры цифр в консоль. При выборе пункта 1, мы вводим матрицу вручную и в конце на экране появляется веденная матрица и преобразованная (Рисунок 6).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 6 – пункт 2 программы

При выборе пункта 2 происходит все тоже самое, только матрица генерируется случайными числами от 0 до 1.

При выборе пункта 3 происходит выход из программы.

# ЗАКЛЮЧЕНИЕ

Реализация динамического многомерного массива является важным аспектом программирования, позволяющим эффективно управлять памятью и создавать структуры данных произвольного размера. Динамические многомерные массивы обеспечивают гибкость в работе с данными, позволяя программистам выделять память в зависимости от потребностей приложения и избегать жестких ограничений, связанных с статическим выделением памяти.

В процессе создания динамического многомерного массива необходимо учитывать несколько ключевых моментов, таких как правильное выделение и освобождение памяти, а также управление доступом к элементам массива. Использование указателей и динамического выделения памяти позволяет создавать массивы с произвольными размерами и формами, что значительно расширяет возможности работы с данными.

Кроме того, динамические многомерные массивы находят широкое применение в различных областях, таких как обработка изображений, научные вычисления и моделирование, где требуется работа с большими объемами данных. Правильная реализация таких структур данных способствует повышению производительности и эффективности программных решений.

В заключение, динамические многомерные массивы представляют собой мощный инструмент в арсенале разработчиков, позволяя создавать гибкие и масштабируемые приложения, способные адаптироваться к изменяющимся требованиям и условиям работы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания

ПРИЛОЖЕНИЕ

#define \_USE\_MATH\_DEFINES

#define \_CRT\_SECURE\_NO\_WARNINGS

#define SIZE 6

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

int\*\* zapolnenie\_matrix(int\*\* matrix, int size); // Функция для заполнения матрицы вручную

int\*\* zapolnenie\_matrix\_randomno(int\*\* matrix, int size); // Функция для заполнения матрицы случайными значениями

int\*\* one\_or\_null(int\*\* matrix, int size); // Функция для проверки возможности формирования квадрата из единиц

int\*\* replace\_matrix(int\*\* matrix, int size); // Функция для замены нулей на единицы

void print\_matrix(int\*\* matrix, int size); // Функция для печати матрицы

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

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

printf("\* Курсовая работа: \*\n");

printf("\* Тема: Реализация динамического многомерного массива \*\n");

printf("\* Выполнил: Есаков М.Е \*\n");

printf("\* Группа: бИД - 241 \*\n");

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

// Выделение памяти для матрицы

int\*\* matrix = (int\*\*)malloc(SIZE \* sizeof(int\*));

// Проверка выделения памяти

if (matrix == NULL)

{

puts("Ошибка выделения памяти");

return -1;

}

// Проход в цикле по каждому указателю и выделение памяти для каждой строки – элементов одномерного массива

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

{

matrix[i] = (int\*)malloc(SIZE \* sizeof(int));

// Проверка выделения памяти

if (matrix[i] == NULL)

{

puts("Ошибка выделения памяти");

return -1;

}

}

int option;

do

{

printf("Выберите способ инициализации матрицы 6x6:\n");

printf("1. Ввести матрицу вручную\n");

printf("2. Сгенерировать матрицу случайно\n");

printf("3. Выход из программы\n");

printf("Ваш выбор: ");

scanf("%d", &option);

switch (option)

{

case 1:

// Ввод матрицы вручную

printf("Введите элементы матрицы 6x6 (0 или 1):\n");

zapolnenie\_matrix(matrix, SIZE);

break;

case 2:

// Заполнение матрицы рандомными числами

zapolnenie\_matrix\_randomno(matrix, SIZE);

printf("Сгенерированная матрица:\n");

print\_matrix(matrix, SIZE);

break;

case 3:

printf("Выход из программы\n");

break;

default:

printf("Неверный выбор\n");

}

if (option != 3)

{

// Изменение матрицы

replace\_matrix(matrix, SIZE);

printf("Преобразованная матрица:\n");

print\_matrix(matrix, SIZE);

// Проверка на правильность изменения

if (one\_or\_null(matrix, SIZE) == 1)

{

printf("Квадрат из 4 единиц в левом верхнем углу сформировать невозможно\n");

}

if (one\_or\_null(matrix, SIZE) == 0)

{

printf("Квадрат из 4 единиц в правом нижнем углу сформировать невозможно\n");

}

}

// Продолжение выполнения программы

printf("Хотите продолжить? (1 - Да, 0 - Нет): ");

scanf("%d", &option);

} while (option == 1);

// Освобождение памяти

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

{

free(matrix[i]);

}

free(matrix);

}

int\*\* zapolnenie\_matrix(int\*\* matrix, int size)

{

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

{

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

{

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

}

}

return \*\*matrix;

}

int\*\* zapolnenie\_matrix\_randomno(int\*\* matrix, int size)

{

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

{

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

{

matrix[i][j] = rand() % 2;

}

}

return \*\*matrix;

}

int\*\* one\_or\_null(int\*\* matrix, int size)

{

if (matrix[0][0] == 0 || matrix[0][1] == 0 || matrix[1][0] == 0 || matrix[1][1] == 0)

{

return 1;

}

if (matrix[4][4] == 0 || matrix[4][5] == 0 || matrix[5][4] == 0 || matrix[5][5] == 0)

{

return 0;

}

}

int\*\* replace\_matrix(int\*\* matrix, int size)

{

int ones\_needed = 4;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

if (matrix[i][j] == 0)

{

for (int k = 0; k < size; k++)

{

for (int l = 0; l < size; l++)

{

if (matrix[k][l] == 1)

{

int temp = matrix[i][j];

matrix[i][j] = matrix[k][l];

matrix[k][l] = temp;

ones\_needed--;

if (ones\_needed == 0) break;

}

}

if (ones\_needed == 0) break;

}

}

}

}

ones\_needed = 4;

for (int i = size - 2; i < size; i++)

{

for (int j = size - 2; j < size; j++)

{

if (matrix[i][j] == 0)

{

for (int k = 2; k < size; k++)

{

for (int l = 2; l < size; l++)

{

if (matrix[k][l] == 1)

{

int temp = matrix[i][j];

matrix[i][j] = matrix[k][l];

matrix[k][l] = temp;

ones\_needed--;

if (ones\_needed == 0) break;

}

}

if (ones\_needed == 0) break;

}

}

}

}

return \*\*matrix;

}

void print\_matrix(int\*\* matrix, int size)

{

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

{

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

{

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

}

printf("\n");

}

printf("\n");

}