****

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»**

Институт (факультет) ИИК

Кафедра Кафедра программных систем

**ОТЧЁТ**

**по лабораторной работе**

№01 «Работа с кучами процессов в ОС Windows»по дисциплине «Операционные системы»

Выполнил Гусева М. О., 6401

Проверил Каликин М.А.

Самара  
2025

Задание

Требуется реализовать код приложения со гласно варианту задания, структурно обозначив в нем (например, с помощью функций) три части: код для выделения в динамической памяти массивов средствами Windows API; код для выполнения об работки содержимого массивов; код для освобождения динамической памяти. Динамическая память (память в куче) должна выделяться при помощи функции HeapAlloc() и связанных с ней функций, а не путем использования функций или операторов стандартной библиотеки языков C/C++. Для выделения памяти может быть выбрана стратегия работы со стандартной кучей процесса или выделения памяти в специально созданной куче. Можно выделить память непосредственными обращениями к функциям HeapAlloc() или путем переопределения оператора new на языке C++.

Метод освобождения динамической памяти зависит от способа ее выделения и может заключаться в поочередном возврате блоков в кучу путем вызова HeepFree() или в удалении кучи вместе со всеми выделенными в ней блоками памяти.

Вариант №7. Транспонировать матрицу, размещенную в динамическом массиве размерности N×N, не используя дополнительного массива.

Демонстрация работы программы

Код представлен в приложении А. Программа разделена на заголовочный файл, где определены функции для работы с массивом и основной файл, в котором демонстрируется работа функций. Результат работы программы представлен на рисунках 1-2.

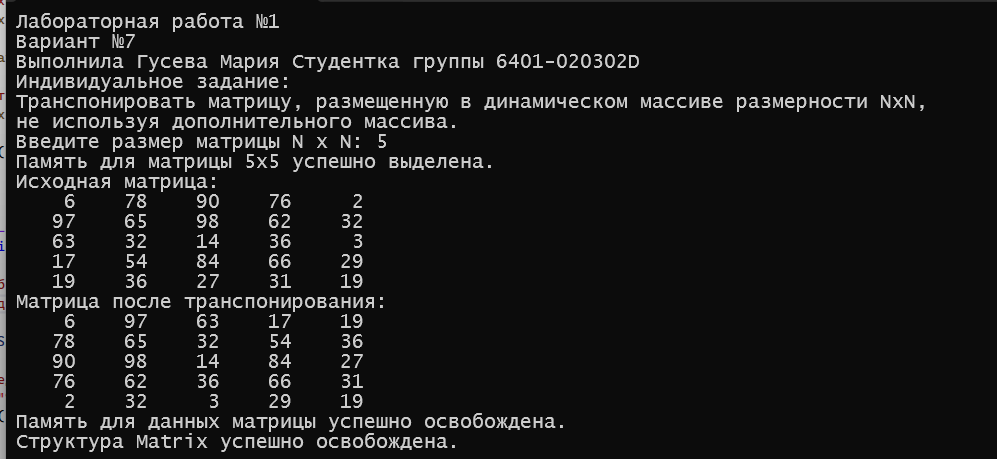


Рисунок 1 – Демонстрация работы программы с матрицей размерности 5х5

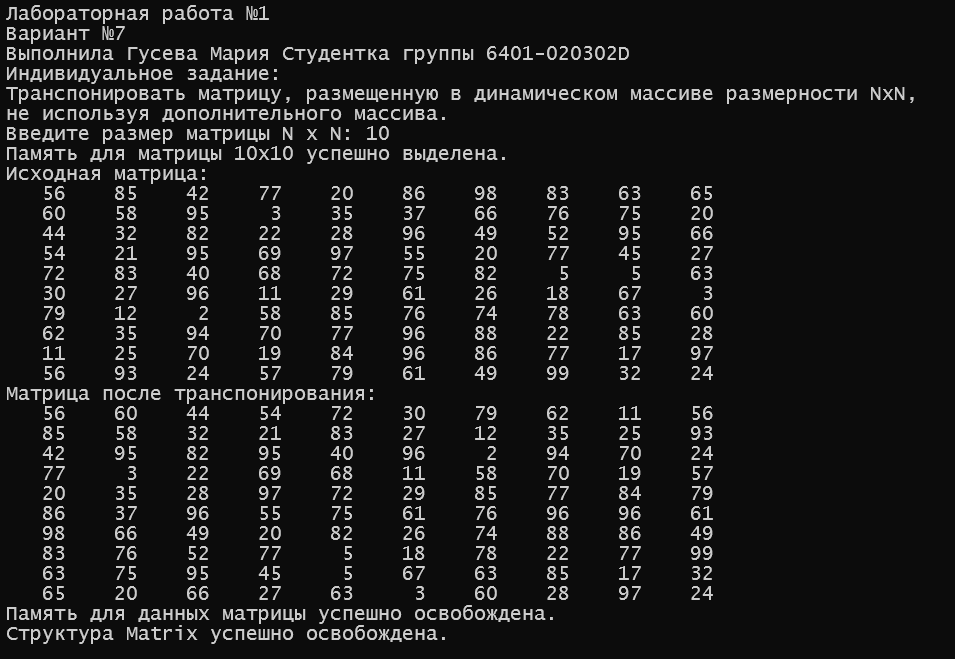


Рисунок 2 – Демонстрация работы программы с матрицей размерности 10х10

Приложение А

Листинг приложения

Приложение А.1 Код функций для структуры Matrix

#include <windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct Matrix {

int n;

int\* data; // Указатель на начало выделенной памяти

HANDLE heapHandle;

};

// Выделение памяти для матрицы

struct Matrix\* AllocateMatrix(int n, HANDLE heapHandle) {

if (n <= 0) {

printf("Ошибка: Размерность матрицы должна быть положительной.\n");

return NULL;

}

if (heapHandle == NULL) {

heapHandle = GetProcessHeap();

if (heapHandle == NULL) {

printf("Ошибка: Не удалось получить дескриптор кучи процесса.\n");

return NULL;

}

}

// Выделение памяти для данных матрицы

int\* allocatedMemory = (int\*)HeapAlloc(heapHandle, HEAP\_ZERO\_MEMORY, n \* n \* sizeof(int));

if (allocatedMemory == NULL) {

printf("Ошибка: Не удалось выделить память для матрицы размером %dx%d.\n", n, n);

return NULL;

}

// Выделение памяти для структуры Matrix

struct Matrix\* matrix = (struct Matrix\*)malloc(sizeof(struct Matrix));

if (matrix == NULL) {

printf("Ошибка: Не удалось выделить память для структуры Matrix.\n");

HeapFree(heapHandle, 0, allocatedMemory);

return NULL;

}

matrix->n = n;

matrix->data = allocatedMemory;

matrix->heapHandle = heapHandle;

printf("Память для матрицы %dx%d успешно выделена.\n", n, n);

return matrix;

}

// Освобождение памяти матрицы

void FreeMatrix(struct Matrix\* matrix) {

if (matrix == NULL) {

return;

}

if (matrix->data != NULL) {

if (!HeapFree(matrix->heapHandle, 0, matrix->data)) {

printf("Предупреждение: Не удалось освободить память для данных матрицы.\n");

}

else {

printf("Память для данных матрицы успешно освобождена.\n");

}

matrix->data = NULL;

}

free(matrix);

printf("Структура Matrix успешно освобождена.\n");

}

// Транспонирование матрицы

void transposeMatrix(struct Matrix\* matrix) {

if (matrix == NULL || matrix->data == NULL || matrix->n <= 1) {

printf("Матрица не содержит данных\n");

return;

}

int n = matrix->n;

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

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

int temp = matrix->data[i \* n + j];

matrix->data[i \* n + j] = matrix->data[j \* n + i];

matrix->data[j \* n + i] = temp;

}

}

}

// Вывод матрицы

void printMatrix(const struct Matrix\* matrix) {

if (matrix == NULL || matrix->data == NULL) {

printf("Матрица пустая\n");

return;

}

int n = matrix->n;

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

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

printf("%5d ", matrix->data[i \* n + j]);

}

printf("\n");

}

}

// Заполнение матрицы

void fillMatrixRandom(struct Matrix\* matrix) {

if (matrix == NULL || matrix->data == NULL) {

return;

}

int n = matrix->n;

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

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

matrix->data[i \* n + j] = rand() % 100;

}

}

}

Приложение А.2 Код для демонстрации работы функций

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include "matrix.h"

void demonstrateMatrixOperations(int size) {

struct Matrix\* matrix = AllocateMatrix(size, NULL);

if (matrix == NULL) {

return;

}

fillMatrixRandom(matrix);

printf("Исходная матрица:\n");

printMatrix(matrix);

transposeMatrix(matrix);

printf("Матрица после транспонирования:\n");

printMatrix(matrix);

FreeMatrix(matrix);

}

int main() {

setlocale(LC\_ALL, "Russian");

srand((unsigned int)time(NULL));

printf("Лабораторная работа №1\nВариант №7\nВыполнила Гусева Мария Студентка группы 6401-020302D\n");

printf("Индивидуальное задание: \nТранспонировать матрицу, размещенную в динамическом массиве размерности NxN, \nне используя дополнительного массива.\n");

int matrixSize;

printf("Введите размер матрицы N x N: ");

if (scanf("%d", &matrixSize) != 1 || matrixSize <= 0) {

printf("Ошибка: неверный размер матрицы.\n");

return 1;

}

demonstrateMatrixOperations(matrixSize);

return 0;

}