Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЁТ

к лабораторной работе №5

на тему

Динамическое выделение памяти.

ВАРИАНТ 7

Проверил Выполнил

асс. каф. ЭВМ студент гр. 318302 П.С. Жуковец В.С.Бахур

МИНСК 2023

**1** **ЗАДАНИЕ №1**

**1.1 Условие**

Заполнить массив натуральными числами с клавиатуры. Удалить из массива все двузначные числа.

**1.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении A»; Рисунок 1.1 (стр.8)

**1.3 Листинг программы**

/\*Заполнить массив натуральными числами с клавиатуры. Удалить из массива все двузначные числа.\*/

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

void fillArray(int\*\* array, int\* size) {

printf("Введите размер массива: ");

if (scanf\_s("%d", size) != 1 || \*size <= 0) {

printf("Ошибка ввода размера массива\n");

exit(EXIT\_FAILURE);

}

\*array = (int\*)malloc(\*size \* sizeof(int));

if (\*array == NULL) {

printf("Ошибка выделения памяти\n");

exit(EXIT\_FAILURE);

}

printf("Введите натуральные числа для заполнения массива:\n");

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

printf("Элемент %d: ", i + 1);

if (scanf\_s("%d", &(\*array)[i]) != 1) {

printf("Ошибка ввода элемента массива\n");

exit(EXIT\_FAILURE);

}

}

}

void removeTwoDigitNumbers(int\*\* array, int\* size) {

int newSize = 0;

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

if ((\*array)[i] < 10 || (\*array)[i] > 99) {

// Если число не двузначное, оставляем его в массиве

(\*array)[newSize++] = (\*array)[i];

}

}

if (newSize < \*size) {

// Если были удалены элементы, переаллоцируем память

\*array = (int\*)realloc(\*array, newSize \* sizeof(int));

if (\*array == NULL) {

printf("Ошибка перевыделения памяти\n");

exit(EXIT\_FAILURE);

}

\*size = newSize;

printf("Двузначные числа удалены из массива.\n");

}

else {

printf("В массиве нет двузначных чисел.\n");

}

}

void printArray(int\* array, int size) {

printf("Массив после удаления двузначных чисел:\n");

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

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

}

printf("\n");

}

void freeArray(int\* array) {

free(array); //ункция стандартной библиотеки языка Си, предназначенная для освобождения ранее выделенной динамической памяти.

}

int main() {

setlocale(LC\_ALL, "Rus");

int\* array = NULL;

int size;

fillArray(&array, &size);

removeTwoDigitNumbers(&array, &size);

printArray(array, size);

freeArray(array);

return 0;

}

**2 ЗАДАНИЕ №2**

**2.1 Условие**

В двумерной матрице размером NxM найти столбец с наибольшей суммой элементов и удалить его.

**2.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении A»; Рисунок 1.2(стр. 9,10)

**2.3 Листинг программы**

/\*В двумерной матрице размером NxM найти столбец с наибольшей суммой элементов и удалить его.\*/

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

void allocateMatrix(int\*\*\* matrix, int rows, int cols) {

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

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

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

}

}

void deallocateMatrix(int\*\*\* matrix, int rows) {

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

free((\*matrix)[i]);

}

free(\*matrix);

\*matrix = NULL;

}

void fillMatrix(int\*\* matrix, int rows, int cols) {

printf("Введите элементы матрицы %dx%d:\n", rows, cols);

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

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

// Проверяем, успешно ли произведено чтение

if (scanf\_s("%d", &matrix[i][j]) != 1) {

printf("Ошибка ввода. Пожалуйста, введите только целые числа.\n");

exit(EXIT\_FAILURE);

}

}

}

}

void printMatrix(int\*\* matrix, int rows, int cols) {

printf("Матрица %dx%d:\n", rows, cols);

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

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

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

}

printf("\n");

}

}

void findAndRemoveMaxSumColumn(int\*\* matrix, int rows, int\* cols) {

int maxSum = 0;

int maxSumColumn = -1; // Изменено на -1 для обозначения отсутствия максимального столбца

int countMaxSumColumns = 0; // Счетчик столбцов с максимальной суммой

// Находим столбец с наибольшей суммой

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

int currentSum = 0;

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

currentSum += matrix[i][j];

}

if (currentSum > maxSum) {

maxSum = currentSum;

maxSumColumn = j;

countMaxSumColumns = 1; // Сбрасываем счетчик, так как нашли новый максимум

}

else if (currentSum == maxSum) {

countMaxSumColumns++; // Увеличиваем счетчик столбцов с равной максимальной суммой

}

}

// Проверяем, есть ли более одного столбца с максимальной суммой

if (countMaxSumColumns > 1) {

printf("Есть несколько столбцов с одинаковой максимальной суммой. Ни один столбец не будет удален.\n");

return;

}

// Удаляем найденный столбец

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

for (int j = maxSumColumn; j < \*cols - 1; ++j) {

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

}

}

// Уменьшаем количество столбцов

\*cols -= 1;

printf("Столбец с наибольшей суммой (%d) удален.\n", maxSum);

}

int main() {

setlocale(LC\_ALL, "Rus");

int\*\* matrix = NULL;

int rows, cols;

printf("Введите количество строк матрицы: ");

if (scanf\_s("%d", &rows) != 1) {

printf("Ошибка ввода. Пожалуйста, введите только целое число.\n");

return EXIT\_FAILURE;

}

printf("Введите количество столбцов матрицы: ");

if (scanf\_s("%d", &cols) != 1) {

printf("Ошибка ввода. Пожалуйста, введите только целое число.\n");

return EXIT\_FAILURE;

}

allocateMatrix(&matrix, rows, cols);

fillMatrix(matrix, rows, cols);

printf("Исходная ");

printMatrix(matrix, rows, cols);

findAndRemoveMaxSumColumn(matrix, rows, &cols);

printf("Матрица после удаления столбца:\n");

printMatrix(matrix, rows, cols);

deallocateMatrix(&matrix, rows);

return 0;

}

**3 ЗАДАНИЕ №3**

* 1. **Условие**

Из двумерной матрицы размером NxM удалить элементы, находящиеся ниже побочной диагонали.

* 1. **Алгоритм решения**

Алгоритм предоставлен в «Приложении A»; Рисунок 1.3

**3.3 Листинг программы**

/\*Из двумерной матрицы размером NxM удалить элементы, находящиеся ниже побочной диагонали.\*/

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int\*\* allocateMatrix(int rows, int cols) {

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

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

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

}

return matrix;

}

void freeMatrix(int\*\* matrix, int rows) {

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

free(matrix[i]);

}

free(matrix);

}

void removeBelowSecondaryDiagonal(int\*\* matrix, int rows, int cols) {

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

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

if (i + j >= rows) {

matrix[i][j] = 0;

}

}

}

}

void printMatrix(int\*\* matrix, int rows, int cols) {

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

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

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

}

printf("\n");

}

}

int main() {

setlocale(LC\_ALL, "Rus");

int N, M;

printf("Введите размеры матрицы (NxM): ");

// Проверка на ввод символов

if (scanf\_s("%d %d", &N, &M) != 2 || N <= 0 || M <= 0) {

printf("Неверные размеры матрицы.\n");

return 1;

}

int\*\* matrix = allocateMatrix(N, M);

printf("Введите элементы матрицы:\n");

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

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

// Проверка на ввод символов

if (scanf\_s("%d", &matrix[i][j]) != 1) {

printf("Ошибка ввода. Введите целое число.\n");

freeMatrix(matrix, N);

return 1;

}

}

}

removeBelowSecondaryDiagonal(matrix, N, M);

printf("Матрица после удаления элементов ниже побочной диагонали:\n");

printMatrix(matrix, N, M);

freeMatrix(matrix, N);

return 0;

}

**Приложение А**

Блок-схема алгоритма 1

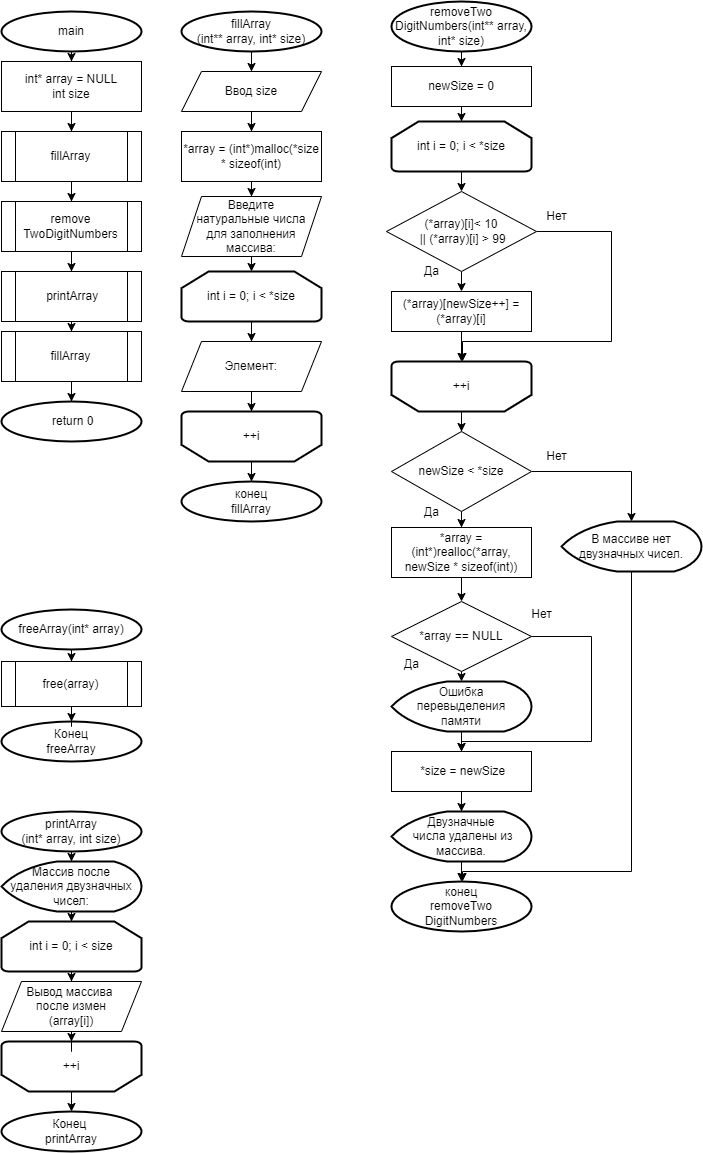
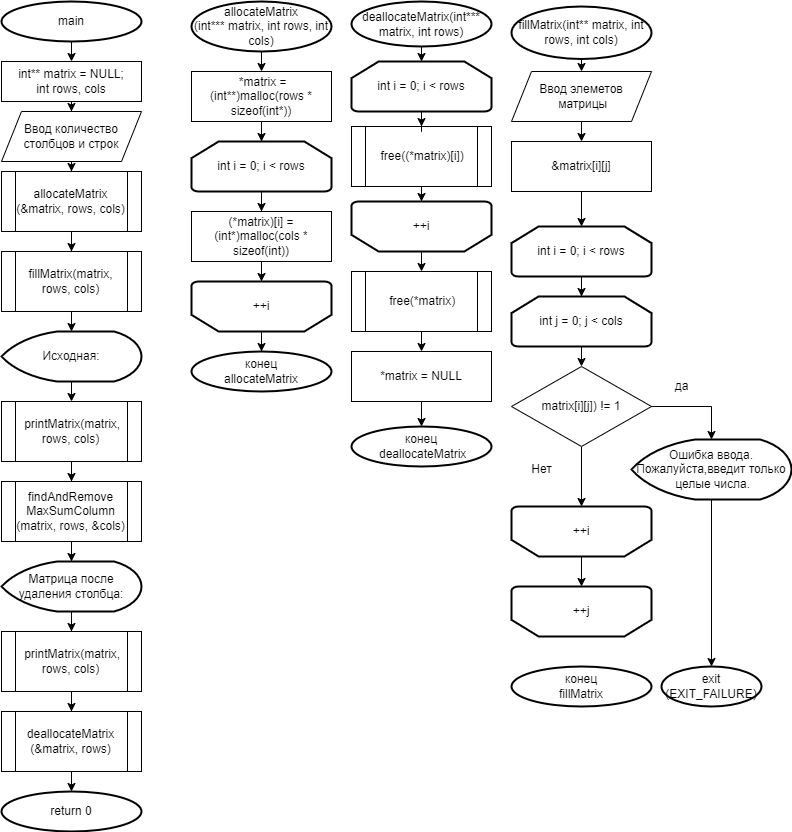


Рисунок 1.1 – Блок-схема №1

Блок-схема алгоритма 2



*продолжение на стр.10*

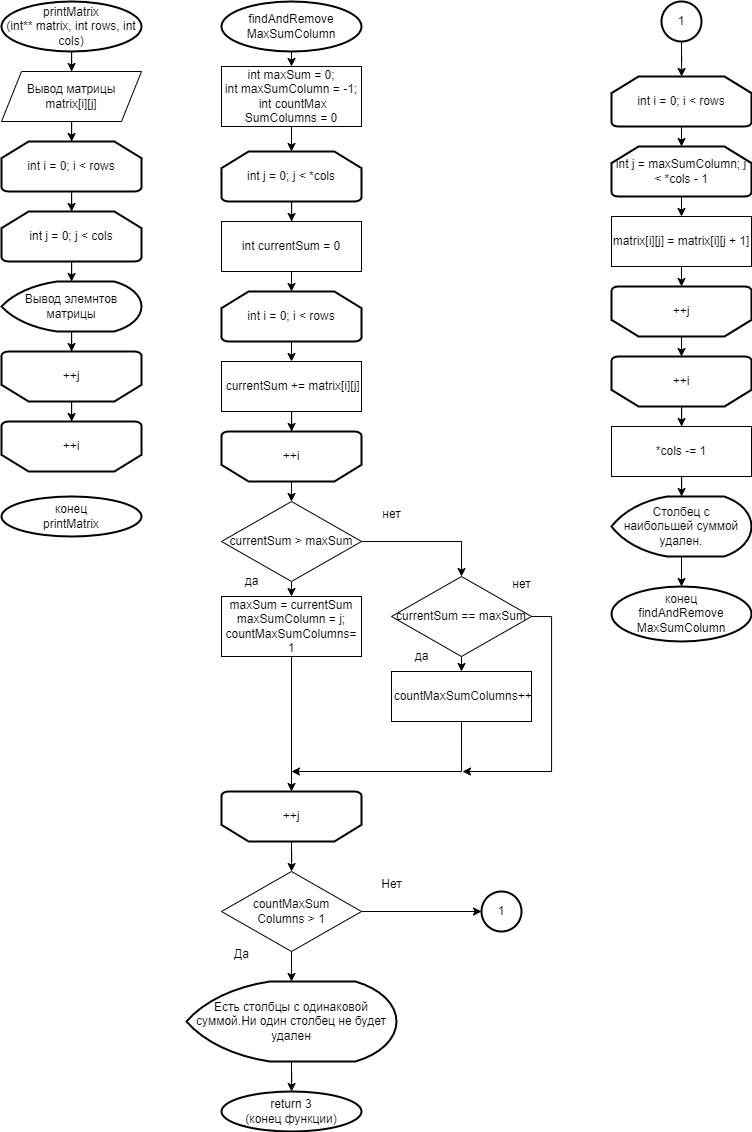


Рисунок 1.2 – Блок-схема №2

Блок-схема алгоритма 3

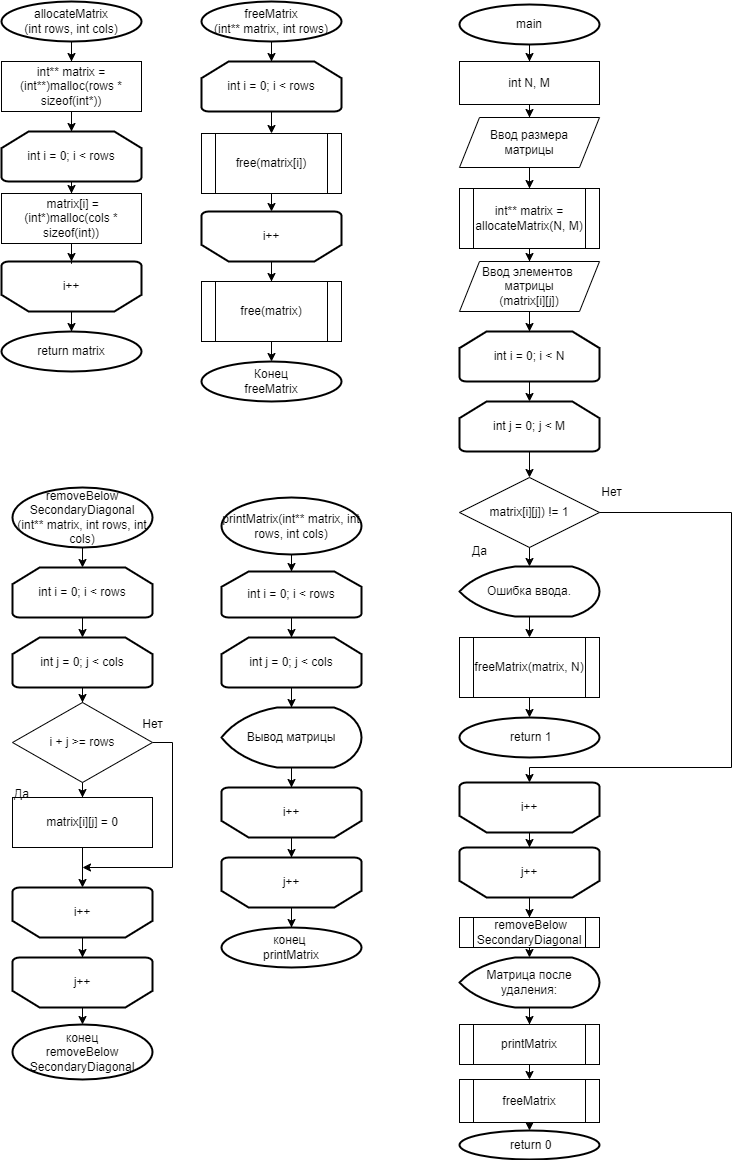


Рисунок 1.3 – Блок-схема №3