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 №6

на тему

СОРТИРОВКИ

ВАРИАНТ 5

Проверил Выполнил

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

МИНСК 2023

**1** **ЗАДАНИЕ №1**

**1.1 Условие**

В одномерном массиве выполнить сортировку элементов кратных k расположенных после максимального методом пузырька.

**1.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении A»; Рисунок 1.1

**1.3 Листинг программы**

/\*В одномерном массиве выполнить сортировку элементов кратных k расположенных

после максимального методом пузырька\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

void bubbleSort(int\* arr, int start, int end, int k);

int findMaxIndex(int\* arr, int start, int end);

void swap(int\* a, int\* b);

void displayArray(int\* arr, int n);

void manualInput(int\* arr, int n);

int main() {

setlocale(LC\_ALL, "Rus");

int n, k, choice;

// Ввод размера массива

printf("Введите размер массива: ");

while (scanf\_s("%d", &n) != 1 || n <= 0) {

printf("Ошибка ввода. Введите положительное целое число: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Выделение динамической памяти под массив

int\* arr = (int\*)malloc(n \* sizeof(int));

// Меню для выбора ввода

printf("Выберите способ ввода массива:\n");

printf("1. Ввести вручную\n");

printf("2. Инициализировать случайными значениями\n");

printf("Ваш выбор: ");

while (scanf\_s("%d", &choice) != 1 || (choice != 1 && choice != 2)) {

printf("Ошибка ввода. Выберите 1 или 2: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Ввод массива

if (choice == 1) {

manualInput(arr, n);

}

else {

// Инициализация массива случайными значениями

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

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

arr[i] = rand() % 100;

}

}

// Вывод исходного массива

printf("\nИсходный массив:\n");

displayArray(arr, n);

// Ввод k

printf("\nВведите значение k: ");

while (scanf\_s("%d", &k) != 1) {

printf("Ошибка ввода. Введите целое число: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Находим индекс максимального элемента

int maxIndex = findMaxIndex(arr, 0, n - 1);

// Сортировка элементов кратных k, расположенных после максимального

bubbleSort(arr, maxIndex + 1, n - 1, k);

// Вывод отсортированного массива

printf("\nОтсортированный массив:\n");

displayArray(arr, n);

// Вывод элементов кратных k, расположенных после максимального элемента

printf("\nЭлементы кратные %d, расположенные после максимального элемента:\n", k);

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

if (arr[i] % k == 0) {

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

}

}

printf("\n");

// Освобождение выделенной памяти

free(arr);

return 0;

}

//Сортировка

void bubbleSort(int\* arr, int start, int end, int k) {

for (int i = end; i >= start; --i) {

for (int j = start; j < i; ++j) {

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

if (arr[j] % k == 0 && arr[j] > arr[m] && arr[m] % k == 0) {

swap(&arr[j], &arr[m]);

break;

}

}

}

}

}

//Нахождение максимального элемента (Его позицию)

int findMaxIndex(int\* arr, int start, int end) {

// Инициализация индекса максимального элемента

int maxIndex = start;

// Поиск максимального элемента в диапазоне

for (int i = start + 1; i <= end; ++i) {

// Если текущий элемент больше максимального, обновляем индекс максимального элемента

if (arr[i] > arr[maxIndex]) {

maxIndex = i;

}

}

// Возвращаем индекс максимального элемента

return maxIndex;

}

//Свап для пузырька

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

//Вывод массива

void displayArray(int\* arr, int n) {

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

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

}

printf("\n");

}

//Заполнение массива

void manualInput(int\* arr, int n) {

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

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

while (scanf\_s("%d", &arr[i]) != 1) {

printf("Ошибка ввода. Введите целое число: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

}

}

**2 ЗАДАНИЕ №2**

**2.1 Условие**

В матрице размером NxM выполнить сортировку строк по возрастанию суммы положительных элементов методом Шелла

**2.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении Б»; Рисунок 1.2

**2.3 Листинг программы**

/\*В матрице размером NxM выполнить сортировку строк по возрастанию

суммы положительных элементов методом Шелла\*/

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

// Функция сортировки Шелла

void shellSort(int\*\* array, int\* sums, int size) {

int gap, i, j, temp;

// Начинаем с большего шага, постепенно уменьшая его до 1

for (gap = size / 2; gap > 0; gap /= 2) {

// Проходим по подмассивам, начиная с gap-ого элемента

for (i = gap; i < size; i++) {

// Сохраняем текущую сумму и соответствующую строку

temp = sums[i];

int\* tempRow = array[i];

// Перемещаем элементы влево, пока не находим правильную позицию для текущего элемента

for (j = i; j >= gap && sums[j - gap] > temp; j -= gap) {

sums[j] = sums[j - gap];

array[j] = array[j - gap];

}

// Вставляем текущий элемент на правильное место

sums[j] = temp;

array[j] = tempRow;

}

}

}

// Функция сортировки строк матрицы

void sortMatrixRows(int\*\* matrix, int numRows, int numCols) {

int\* sums = (int\*)malloc(numRows \* sizeof(int));

// Вычисляем суммы положительных элементов в каждой строке

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

int sum = 0;

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

if (matrix[i][j] > 0) {

sum += matrix[i][j];

}

}

sums[i] = sum;

}

// Вызываем функцию сортировки Шелла для отсортировки строк матрицы

shellSort(matrix, sums, numRows);

// Освобождаем память, выделенную для массива сумм

free(sums);

}

int main() {

int numRows, numCols, fillOption;

setlocale(LC\_ALL, "Rus");

// Ввод количества строк

printf("Введите количество строк: ");

while (scanf\_s("%d", &numRows) != 1 || numRows <= 0) {

printf("Ошибка ввода! Введите корректное количество строк: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Ввод количества столбцов

printf("Введите количество столбцов: ");

while (scanf\_s("%d", &numCols) != 1 || numCols <= 0) {

printf("Ошибка ввода! Введите корректное количество столбцов: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Выбор способа заполнения массива

printf("Выберите способ заполнения массива:\n");

printf("1. Ввод вручную\n");

printf("2. Заполнение случайными значениями\n");

while (scanf\_s("%d", &fillOption) != 1 || (fillOption != 1 && fillOption != 2)) {

printf("Ошибка ввода! Выберите 1 или 2: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

// Выделение памяти под матрицу

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

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

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

}

// Заполнение матрицы

if (fillOption == 1) { // Ввод вручную

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

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

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

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

printf("Ошибка ввода! Введите корректный элемент матрицы: ");

while (getchar() != '\n'); // Очистка буфера ввода

}

}

}

}

else if (fillOption == 2) { // Заполнение случайными значениями

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

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

matrix[i][j] = rand() % 100; // Генерация случайного числа от 0 до 99

}

}

}

// Сортировка строк матрицы

sortMatrixRows(matrix, numRows, numCols);

// Вывод отсортированной матрицы

printf("Отсортированная матрица:\n");

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

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

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

}

printf("\n");

}

// Освобождение памяти

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

free(matrix[i]);

}

free(matrix);

return 0;

}

**Приложение А**

Блок-схема алгоритма 1

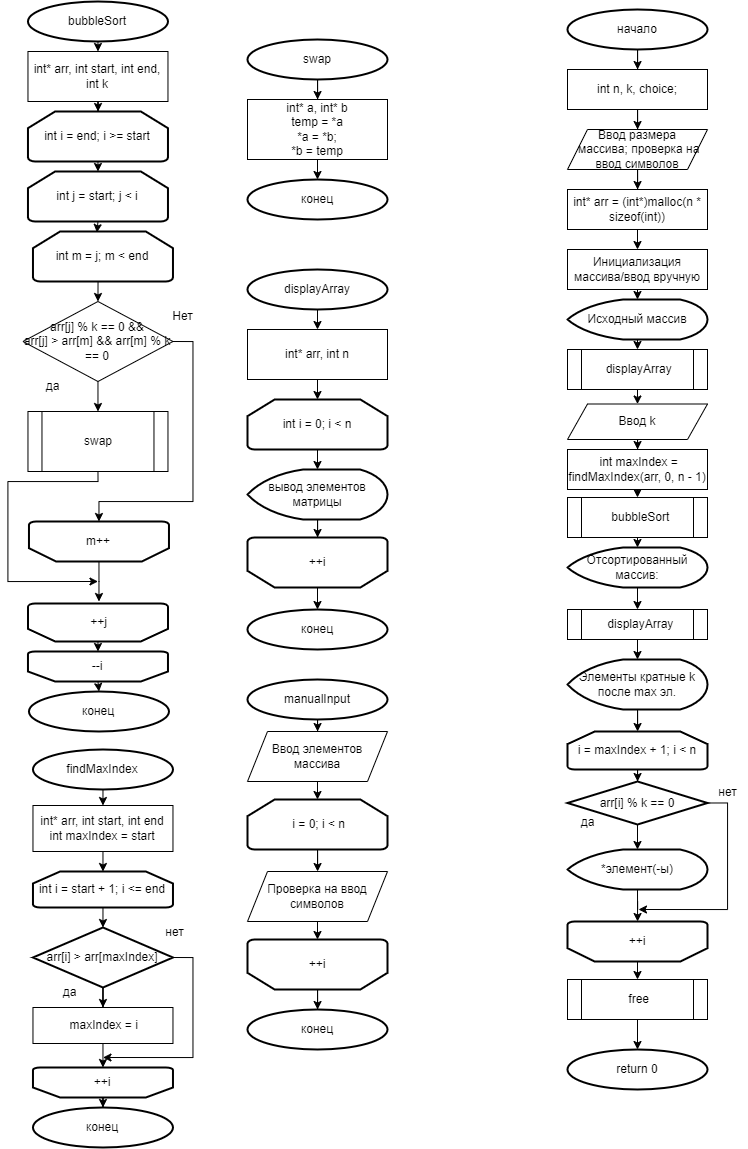


Рисунок 1.1 – Блок-схема №1

**Приложение Б**

Блок-схема алгоритма 2

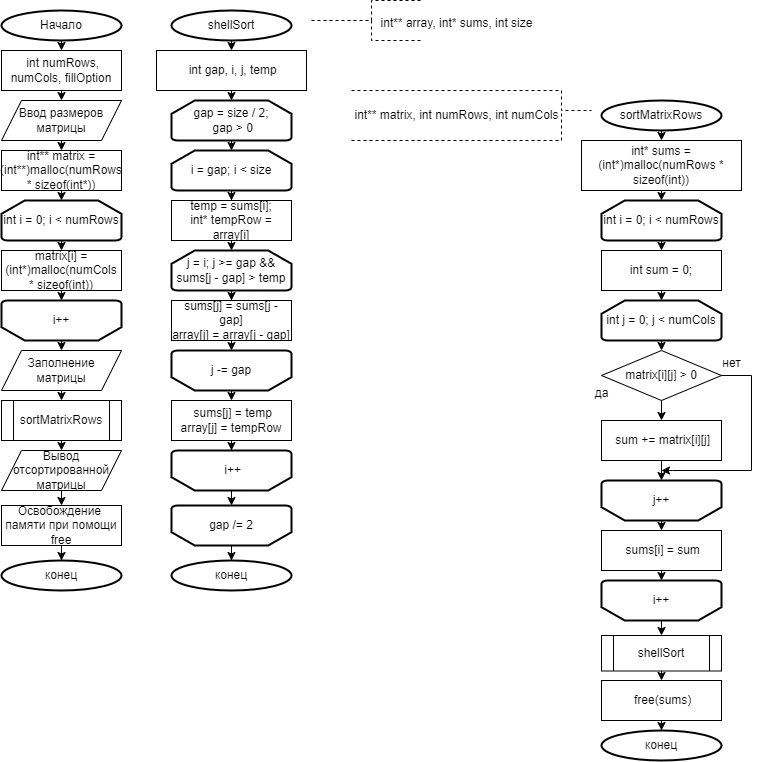


Рисунок 1.2 – Блок-схема №2