МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Технічна Кібернетика»

**Лабораторна робота №6**

**з дисципліни «Основи програмування»**

**на тему: «**Динамічні [масиви](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269) в мові С++**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Динамічні [масиви](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269) в мові С++.

**Мета.** Набути практичних навичок розробки алгоритмів, написання і налагодження програм для обробки різного типу структур динамічних матриць.

# **1) Постановка завдання (загальне завдання, індивідуальне завдання за обраним варіантом і рівнем, вимоги до програми);**

Загальне завдання:

Розробити два варіанти програми роботи з різними типами структури динамічної матриці. Кожна програма:

1) генерує динамічну дійсну матрицю розмірності N на М, де N – кількість рядків і M – кількість стовпців, з випадковими значеннями елементів з вказаного в індивідуальному завданні діапазону;

2) виводить форматовану матрицю на консоль (стовпці матриці повинні бути вирівняні);

3) виконує завдання з обробки матриці для обраного варіанту і рівня складності. Кожен рівень передбачає виконання вказаного завдання та завдань усіх попередніх рівнів.

В першому варіанті програми реалізувати динамічну матрицю такої структури:

*struct Matrix{*

*int n, m;                //розмірність матриці*

*float\* items  ;        //елементи матриці*

*}*

В другому варіанті:

*struct Matrix{*

*int n, m;                //розмірність матриці*

*float\*\* items;        //елементи матриці*

*}*

Індивідуальне завдання:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Діапазон значень елементів матриці | Рівень складності | | | | |
| Е \*  (достатньо) | D  (задовільно) | C  (добре) | B  (дуже добре) | A  (відмінно) |
| 13 | [-10.02; 17.5] | Підрахувати кількість елементів матриці, у яких дробова частина складається з двох однакових  цифр. | Обчислити суму додатних елементів матриці, в яких цифра в розряді десятих менше 5. | Виконати завдання D для кожного рядка матриці. Результати записати в динамічний масив. | Відсортувати рядки матриці за зростанням (непарний варіант) або за спаданням (парний варіант) відповідних елементів масиву, створеного у п.С. | Видалити з матриці всі парні рядки. |

Вимоги до програми:

- всі числа з завдання оголошуються як константи;

- розмірність матриці вводиться з клавіатури;

- передбачити обробку некоректного введення користувачем розмірності матриці;

- результати роботи програми виводяться на екран;

- для виділення і звільнення пам’яті для матриць і масиву використовуються оператори *new* і *delete*.

Вимоги до тексту програми:

- коментарі щодо призначення програми, її вхідних і вихідних даних;

- коментарі щодо призначення кожного блоку програми, дій окремих операторів для пояснення алгоритму;

- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

# **2) Зовнішні специфікації програми;**

Вхідні дані:

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до данних | Приклад |
| Кількість рядків матриці | Matrix1.n | Натуральне число | 3 |
| Кількість стовпчиків матриці | Matrix1.m | Натуральне число | 5 |

Вихідні дані:

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до данних | Приклад |
| Елементи матриці | Matrix1.items[i]  Або  Matrix1.items[i][j] | Дійсне число з діапазону [-10.02;17.5], не більше двох чисел у дробній частині | 10.54 |
| Кількість елементів з двома однаковими числами після коми(наприклад 7.55, 9.99 та ін.) | Matrix1.repdigit\_amount | Натуральне число або нуль | 1 |
| Сума додатніх чисел з дробною частиною меньше 0.6 для кожного рядка. і – номер рядка. | Matrix1.frac\_0\_6[i] |  | 25.57 |
| Сума додатніх чисел з дробною частиною меньше 0.6 для усіх стовпців. | Matrix1.frac\_0\_6\_sum |  | 164.56 |

# **3) Метод рішення завдання;**

1)Сгенерувати матрицю.

2)Вивести матрицю на екран.

3)Зробити масив, кожен елемент якого, зберігає суму додатних елементів з дрібною частиною меньше 0.6 для кожного рядку матриці.

4)Знайти суму усіх елементів масиву з попереднього пункту.

5)Знайти кількість елементів з двома однаковими цифрами після коми.

6)Відсортувати масив з третього пункту, та рядки матриці відповідно елементам масиву з п.3.

7)Видалити парні рядки матриці.

# **4) Набори тестів для перевірки виконання задач створення і обробки матриці;**

|  |  |  |  |
| --- | --- | --- | --- |
| Назва тесту | Вхідні дані | | Очікуваний результат |
|  | n | m |  |
| Генерація матриці 3х3 та виповнення над нею усіх функцій | 3 | 3 | 1)Виведення матриці 3х3 з числами, які відповідають умові у п.4  2)Виведення кількості елементів з двома однаковими числами у дробній частині.  3)Виведення сумми додатніх чисел із дробною частиною меньше 0.6 для кожного рядка.  4) Виведення сумми додатніх чисел із дробною частиною меньше 0.6 для усіх рядків.  5)Виведення матриці з рядками відсортованими за зростанням відповідних сум додатніх чисел із дробною частиною меньше 0.6 за зростанням зверху до низу.  6) виведення матриці з видаленими парними рядками. |
| Генерація матриці 4х4 та виповнення над нею усіх функцій | 4 | 4 | 1)Виведення матриці 4х4 з числами, які відповідають умові у п.4  2)Виведення кількості елементів з двома однаковими числами у дробній частині.  3)Виведення сумми додатніх чисел із дробною частиною меньше 0.6 для кожного рядка.  4) Виведення сумми додатніх чисел із дробною частиною меньше 0.6 для усіх рядків.  5)Виведення матриці з рядками відсортованими за зростанням відповідних сум додатніх чисел із дробною частиною меньше 0.6 за зростанням зверху до низу.  6) виведення матриці з видаленими парними рядками. |

# **5)Загальний алгоритм розв’язання(Діаграма Н-Ш головної функції);**



Рисунок 1

# **6) Алгоритми функцій першої структури**

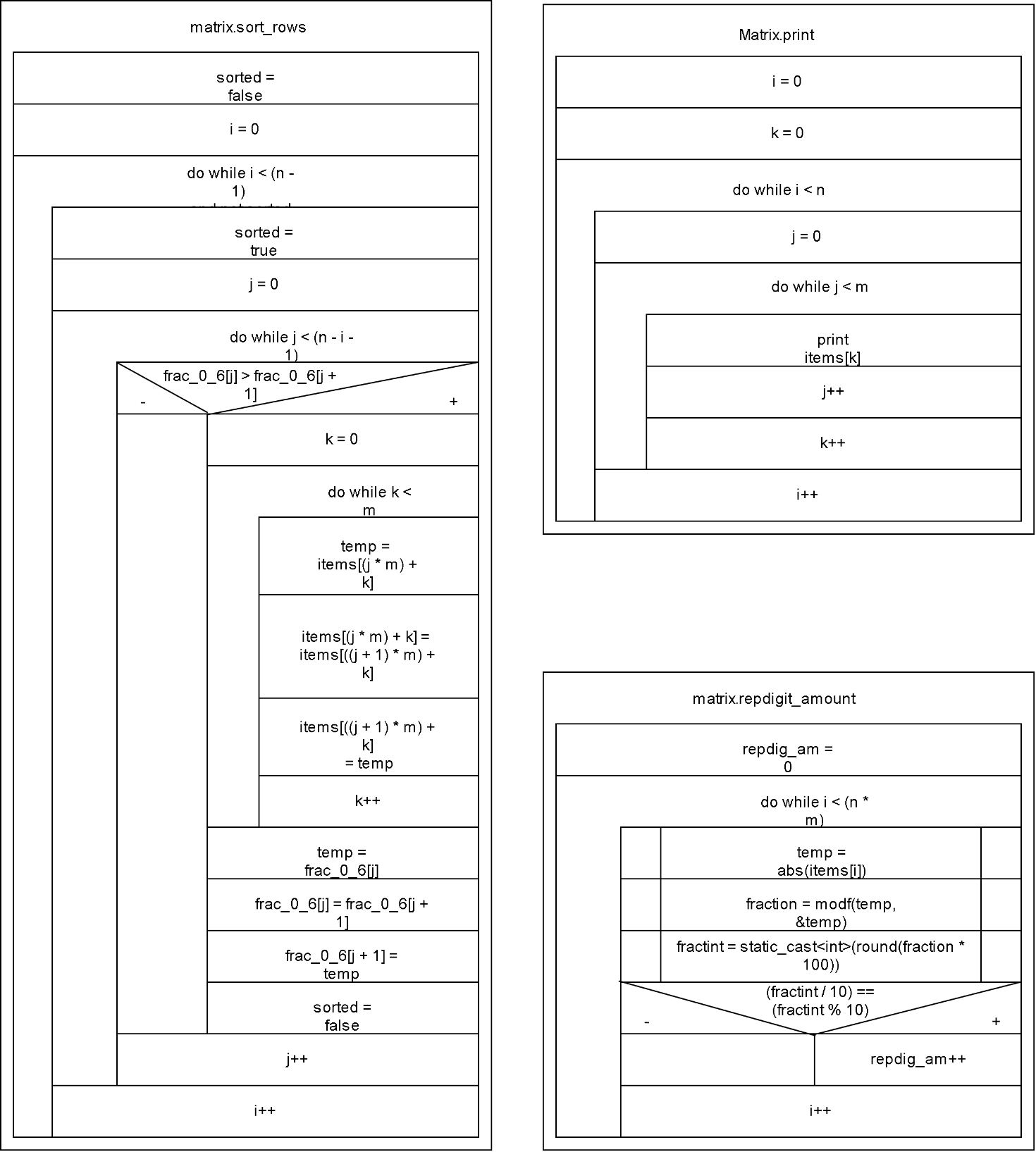


Рисунок 2

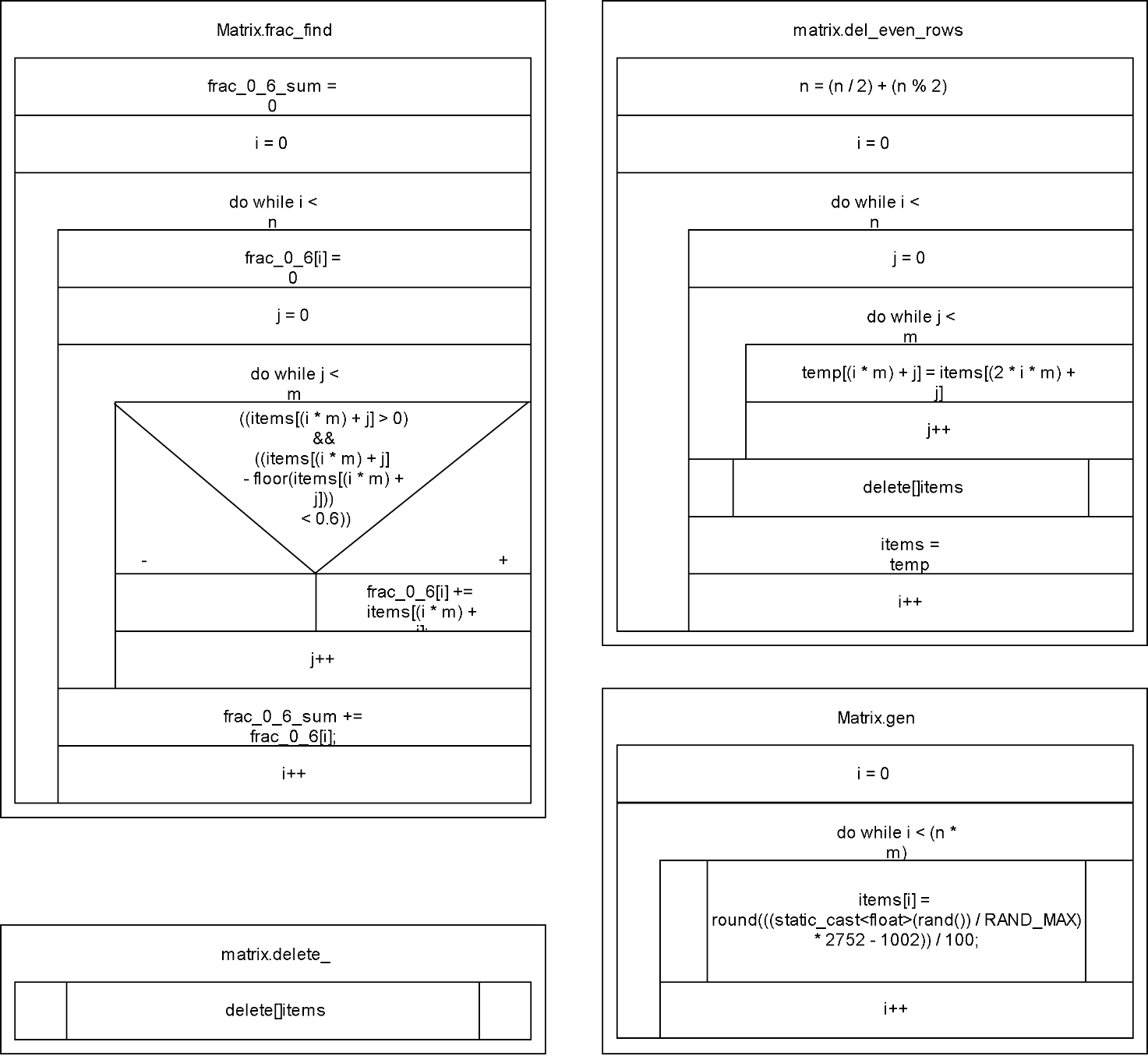


Рисунок 3

# **7)Алгоритми функцій другої структури**

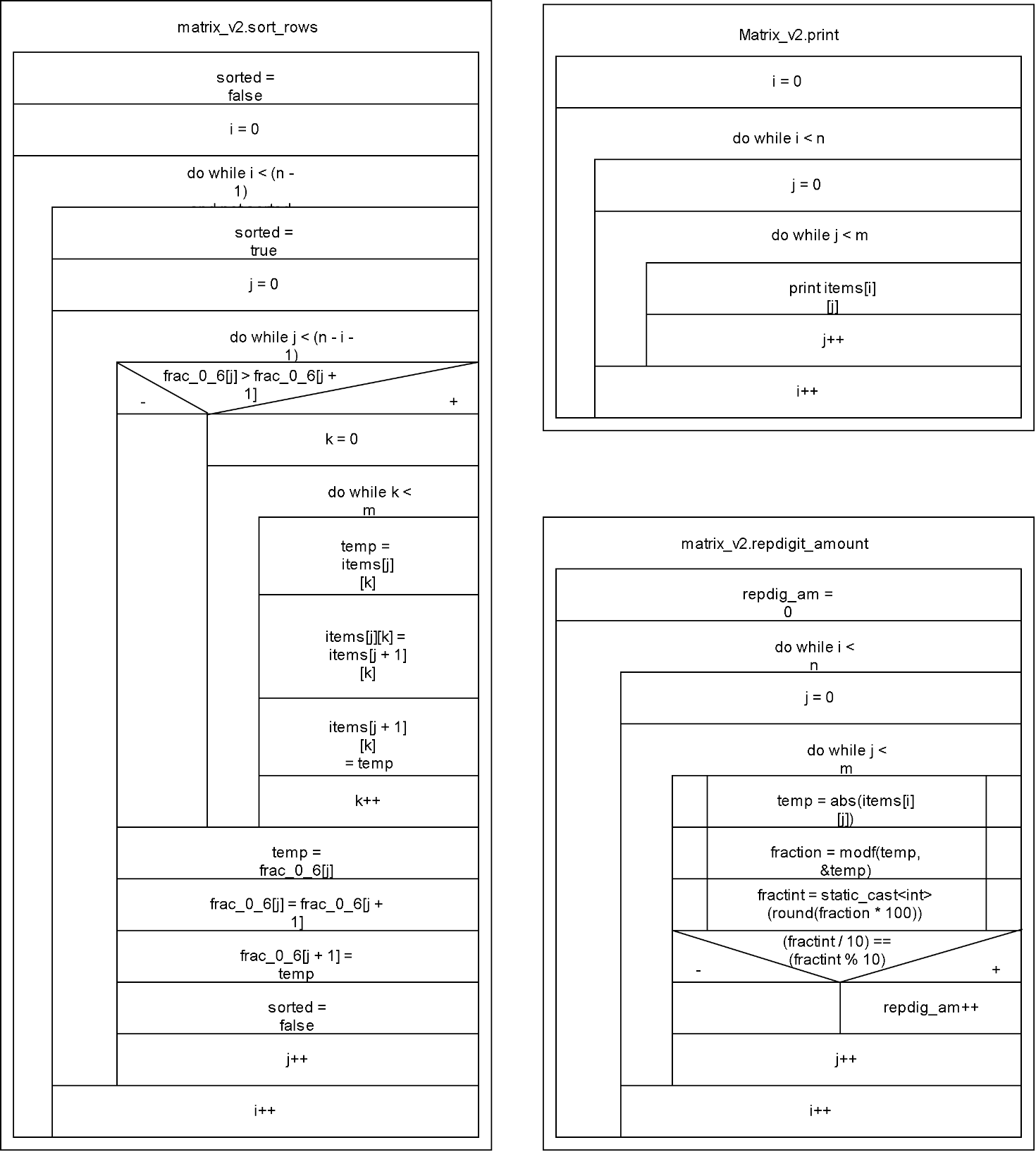


Рисунок 4

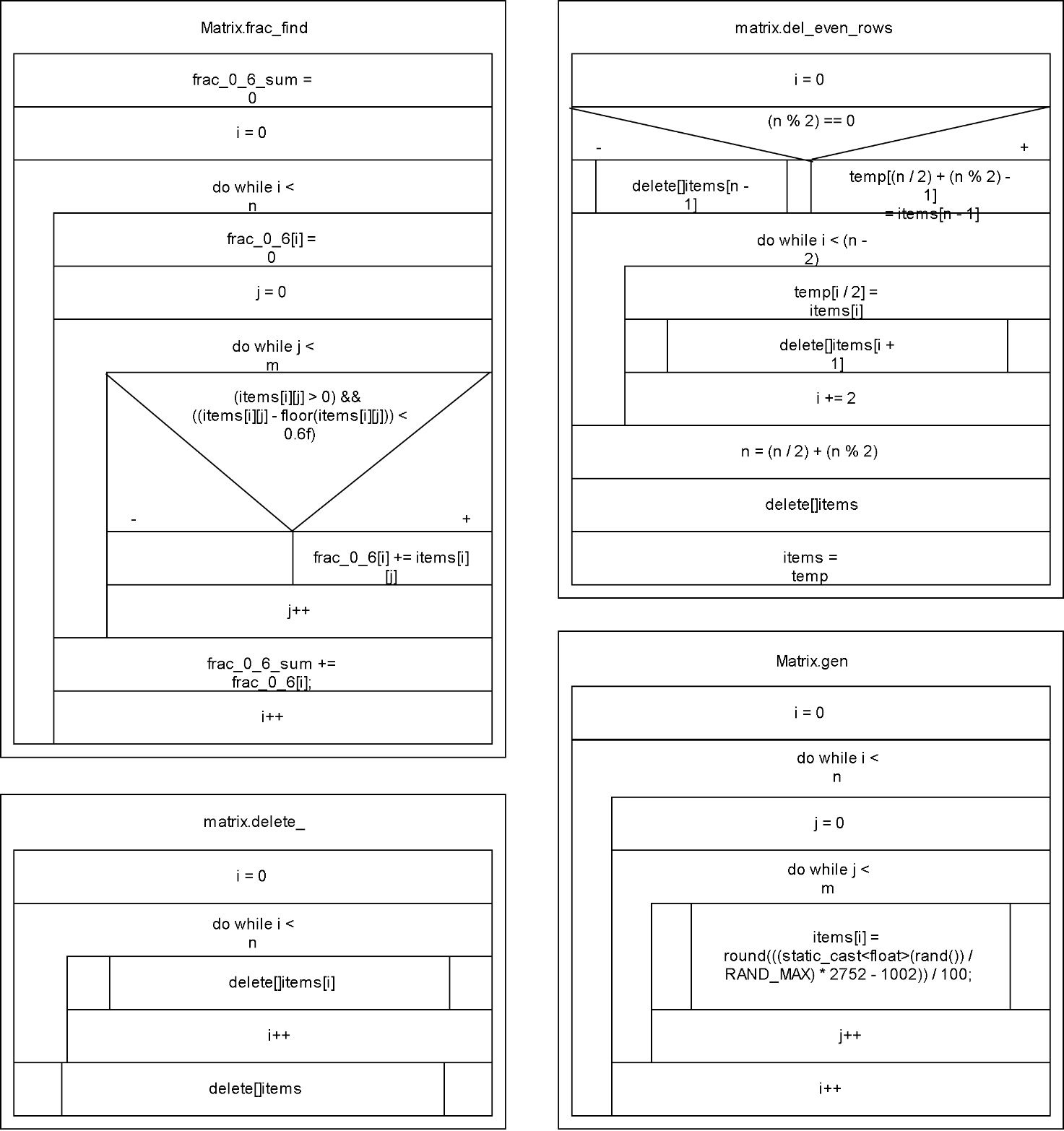


Рисунок 5

# **8) Текст програми(Містить обидва типи структур(треба зробити один рядок коментарем, а інший звичайною строкою(оголошення Матриці)));**

//application manipulates matrix NxM

//input:

//N - amount of rows

//M - amount of columns

//outputs:

//1. pseudo random generated matrix NxM

//2. amount of elements with two same numbers in fraction

//3. sum of additive elements with 5 or less tenth in fraction

//4. sum of additive elements with 5 or less tenth in fraction for every row(array)

//5. array from p.4 sorted ascending and matrix with rows sorted by respective elements of array(p.4)

//6. matrix with deleted even rows

#include <windows.h>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

#include "inp\_val6.h"

#include <cmath>

#include "structs\_.h"

int main()

{

std::srand(time(NULL));

int N, M;

std::cout << "input size of matrix:" << std::endl;

char msg\_invN[] = "N: ";

char msg\_invM[] = "M: ";

char msg\_error[] = "wrong input, try again";

N = cinum(msg\_invN, msg\_error, 1, 1, 1);

M = cinum(msg\_invM, msg\_error, 1, 1, 1);

Matrix matrix1;//v1

//Matrix\_v2 matrix1;//v2

matrix1.n = N;

matrix1.m = M;

matrix1.gen();

std::cout << "Matrix " << matrix1.n << "x" << matrix1.m << " after generation: " << std::endl;

matrix1.print();

std::cout << std::endl;

std::cout << "amount of elements with two same numbers in fraction: " << std::endl;

std::cout << matrix1.repdigit\_amount() << std::endl << std::endl;//E

matrix1.frac\_find();//D+C

std::cout << "sum of additive elements with 5 or less tenth in fraction:" << std::endl;

std::cout << matrix1.frac\_0\_6\_sum << std::endl << std::endl;

std::cout << "sum of additive elements with 5 or less tenth in fraction for every row:" << std::endl;

for (int i = 0; i < matrix1.n; i++) {

std::cout << "row" << (i + 1) << ": " << matrix1.frac\_0\_6[i] << ' ' << std::endl;

}

std::cout << std::endl;

matrix1.sort\_rows();//B

std::cout << "sum of additive elements with 5 or less tenth in fraction for every row(sorted):" << std::endl;

for (int i = 0; i < matrix1.n; i++) {

std::cout << "row" << (i + 1) << ": " << matrix1.frac\_0\_6[i] << ' ' << std::endl;

}

std::cout << std::endl;

std::cout << "Matrix " << matrix1.n << "x" << matrix1.m << " with rows sorted ascending up to down" << std::endl;

std::cout << "by sum of additive elements with 5 or less tenth in fraction, respectively: " << std::endl;

matrix1.print();

std::cout << std::endl;

std::cout << "Matrix " << matrix1.n << "x" << matrix1.m << " with deleted even rows: " << std::endl;

matrix1.del\_even\_rows();

std::cout << "New size of matrix: " << matrix1.n << "x" << matrix1.m << std::endl;

matrix1.print();

std::cout << std::endl;

std::system("pause");

return(0);

}

# **9) Результати тестування програми з першою структурою;**

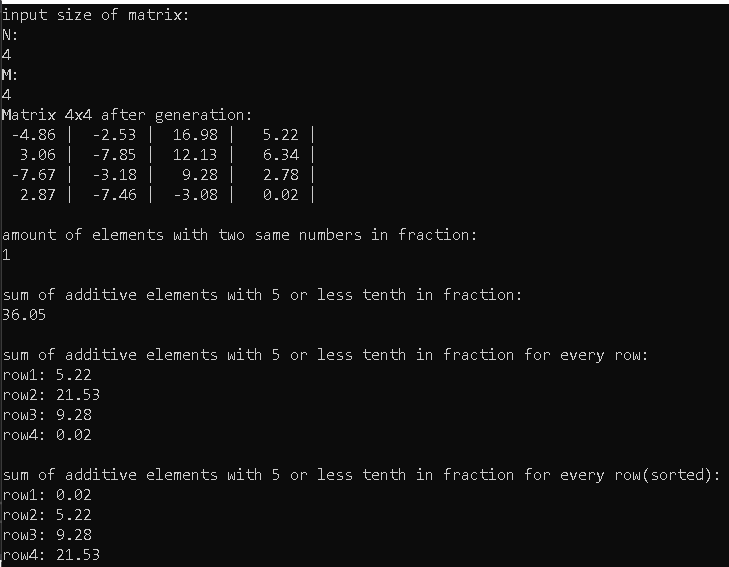


Рисунок 6

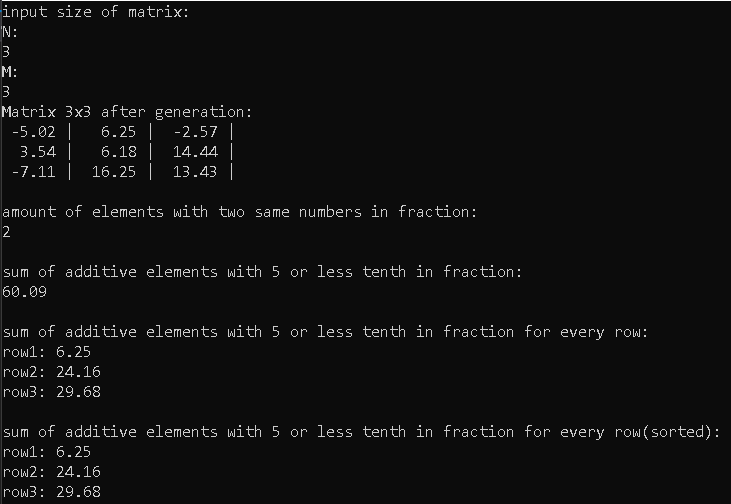


Рисунок 7

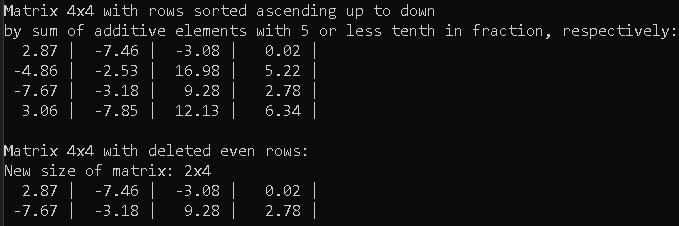


Рисунок 8

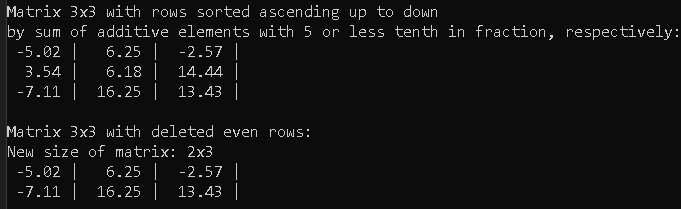


Рисунок 9

Програма працює коректно, результати відповідають очікуваням.

# **8) Текст заголовочного файлу з структурами(structs\_.h)**

struct Matrix {

int n, m;//n - rows, m - columns

float\* items;

float\* frac\_0\_6;//pointer for array with sum of positive elements that have fraction <.6 for every row

float frac\_0\_6\_sum;//sum of frac\_0\_6 array

void gen() {

items = new float[n \* m];

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

items[i] = round(((static\_cast<float>(rand()) / RAND\_MAX) \* 2752 - 1002)) / 100;

//items[i] = ((std::rand() % 2753) / 100.0) - 10.02;//gen rand num[-10.02;17.5]//old

}

}

void print() {

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

for (int j = 0; j < m; j++, k++) {

std::cout << std::setw(6) << items[k] << " | ";

}

std::cout << std::endl;

}

}

void frac\_find() {//make returnable?

frac\_0\_6\_sum = 0;

frac\_0\_6 = new float[n];

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

frac\_0\_6[i] = 0;

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

if ((items[(i \* m) + j] > 0) && ((items[(i \* m) + j] - floor(items[(i \* m) + j])) < 0.6)) {//floor returns whole part

frac\_0\_6[i] += items[(i \* m) + j];

}

}

frac\_0\_6\_sum += frac\_0\_6[i];

}

}

int repdigit\_amount() {

int repdig\_am = 0;

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

float temp = abs(items[i]);//abs returns absolute value of argument

double fraction = modf(temp, &temp);//modf returns fractional part, whole to temp

int fractint = static\_cast<int>(round(fraction \* 100));//round for kostil' fix

if ((fractint / 10) == (fractint % 10)) {//checking if both digits ==

repdig\_am++;

}

}

return repdig\_am;

}

void sort\_rows() {

bool sorted = false;

for (int i = 0; !sorted && (i < (n - 1)); i++) {//i - #of iteration

bool sorted = true;

for (int j = 0; j < (n - i - 1); j++) {//j - #element being compared to next

if (frac\_0\_6[j] > frac\_0\_6[j + 1]) {

for (int k = 0; k < m; k++) {//k - #number of element being swapped

float temp = items[(j \* m) + k];

items[(j \* m) + k] = items[((j + 1) \* m) + k];

items[((j + 1) \* m) + k] = temp;

}

float temp = frac\_0\_6[j];

frac\_0\_6[j] = frac\_0\_6[j + 1];

frac\_0\_6[j + 1] = temp;

sorted = false;

}

}

}

}

void del\_even\_rows() {

n = (n / 2) + (n % 2);

float\* temp = new float[n \* m];

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

for (int j = 0; j < m; j++) {//element iterator

temp[(i \* m) + j] = items[(2 \* i \* m) + j];

}

}

delete[]items;

items = temp;

}

void delete\_ () {//deletes matrix

delete[]items;

}

};

struct Matrix\_v2 {

int n, m;//n - rows, m - columns

float\*\* items;

float\* frac\_0\_6;//pointer for array with sum of positive elements that have fraction <.6 for every row

float frac\_0\_6\_sum;//sum of frac\_0\_6 array

void gen() {

items = new float\*[n];

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

items[i] = new float[m];

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

items[i][j] = round(((static\_cast<float>(rand()) / RAND\_MAX) \* 2752 - 1002)) / 100;

//items[i] = ((std::rand() % 2753) / 100.0) - 10.02;//gen rand num[-10.02;17.5]//old

}

}

}

void print() {

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

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

std::cout << std::setw(6) << items[i][j] << " | ";

}

std::cout << std::endl;

}

}

void frac\_find() {//make returnable?

frac\_0\_6\_sum = 0;

frac\_0\_6 = new float[n];

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

frac\_0\_6[i] = 0;

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

if ((items[i][j] > 0) && ((items[i][j] - floor(items[i][j])) < 0.6f)) {//floor returns whole part, 0.6f because 0.6 ~= 0.5(9) aka 0.6 - 0, and 0.6f ~= 0.6(0) aka 0.6 + 0// no casting to float from double, more consistent

frac\_0\_6[i] += items[i][j];

}

}

frac\_0\_6\_sum += frac\_0\_6[i];

}

}

int repdigit\_amount() {

int repdig\_am = 0;

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

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

float temp = abs(items[i][j]);//abs returns absolute value of argument

double fraction = modf(temp, &temp);//modf returns fractional part, whole to temp

int fractint = static\_cast<int>(round(fraction \* 100));//round for kostil' fix

if ((fractint / 10) == (fractint % 10)) {//checking if both digits ==

repdig\_am++;

}

}

}

return repdig\_am;

}

void sort\_rows() {

bool sorted = false;

for (int i = 0; !sorted && (i < (n - 1)); i++) {//i - #of iteration

bool sorted = true;

for (int j = 0; j < (n - i - 1); j++) {//j - #element being compared to next and row aswell

if (frac\_0\_6[j] > frac\_0\_6[j + 1]) {

for (int k = 0; k < m; k++) {//k - #number of element being swapped

float temp = items[j][k];

items[j][k] = items[j + 1][k];

items[j + 1][k] = temp;

}

float temp = frac\_0\_6[j];

frac\_0\_6[j] = frac\_0\_6[j + 1];

frac\_0\_6[j + 1] = temp;

sorted = false;

}

}

}

}

void del\_even\_rows() {

float\*\* temp = new float\* [(n / 2) + (n % 2)];

if ((n % 2) == 0) {

temp[(n / 2) + (n % 2) - 1] = items[n - 2];

}

else {

temp[(n / 2) + (n % 2) - 1] = items[n - 1];

}

for (int i = 0; i < (n - 2); i += 2) {

temp[i / 2] = items[i];

delete[]items[i + 1];

}

n = (n / 2) + (n % 2);

delete[]items;

items = temp;

}

void delete\_() {//deletes matrix

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

delete[]items[i];

}

delete[]items;

}

};

# **9) Текст inp\_val6.h**

//cinum - numerical only (R)

//char msg\_invite - your message for inviting user to input their number

//char msg\_error - if error input occures

//int m\_ - check if input is multiple of this number

//bool intFlag - =1(only integer input); =0(any input)

//double left - leftmost point in diapasone for input

//double right - rightmost point in diapasone for input

#include <iostream>

#include <windows.h>

#undef max

#include <cmath>

#include <climits>

using namespace std;

int cinum(char\* msg\_invite, char\* msg\_error, int m\_ = 1, bool intFlag = 0, float left = -2147483648.0, float right = 2147483647.0)

{

bool i = 0;

float aaa;

cout << msg\_invite << endl;

do

{

cin >> aaa;

int iaaa;

iaaa = (int)aaa;

if (cin.fail() || (aaa < left) || (aaa > right) || ((intFlag == 1) && (ceil(aaa) != floor(aaa))) || ((m\_ != 1) && ((iaaa % m\_) != 0))) {

cout << msg\_error << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else {

i = 1;

}

} while (!i);

return (int)aaa;

}

void val\_char\_lenght(char array[], int ary\_size) {//input to char array with lenght validation

bool correct = false;

do

{

std::cin.getline(array, 24);

if ((array[ary\_size - 1] != '\0') || cin.fail()) {

std::cout << "wrong input, try again" << std::endl;

cin.clear();

std::cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

else {

correct = true;

}

} while (!correct);

}

# **10) Результати тестування другої програми та їх аналіз;**

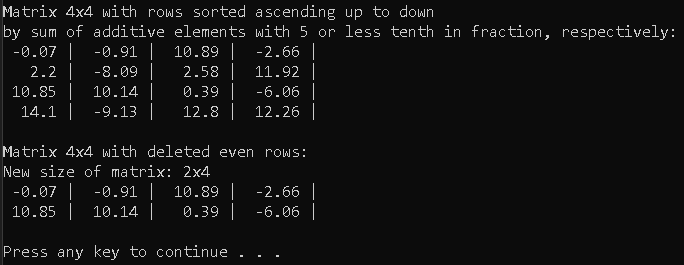


Рисунок 10

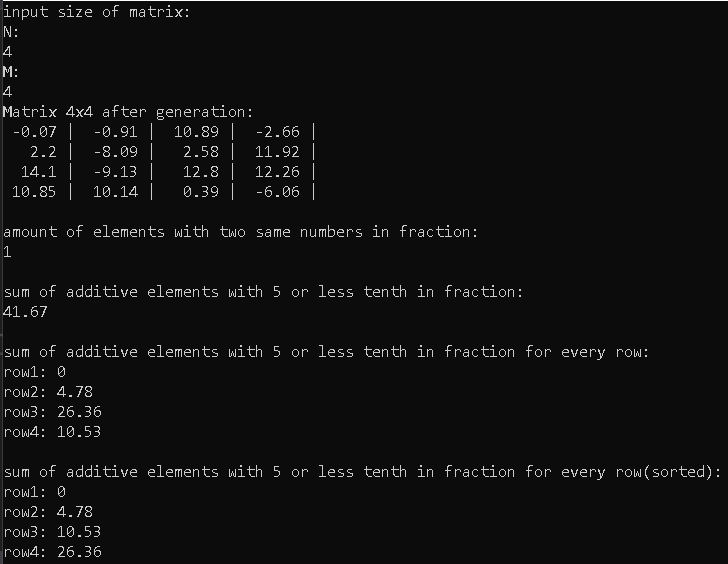


Рисунок 11

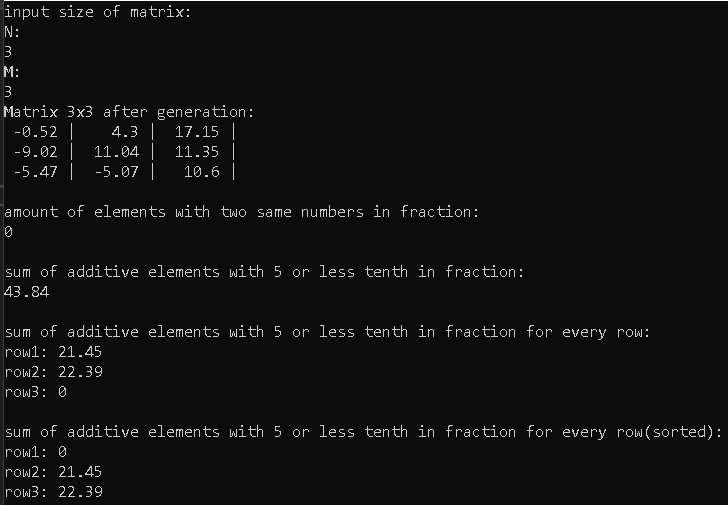


Рисунок 12

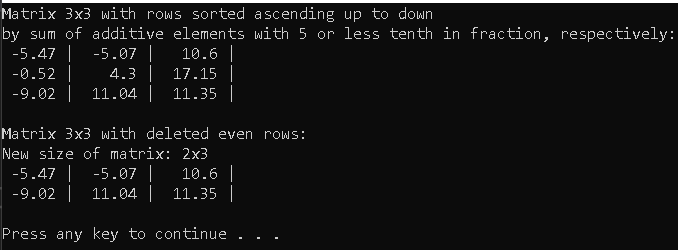


Рисунок 13

Програма працює коректно, результати відповідають очікуваням.

# **11) Висновки щодо залежності алгоритмів обробки динамічних матриць від їх структури.**

У обох структурах основні параметри – розміри матриці (кількість рядків та кількість стовпчиків) та її елементи, але у першому варіанті структури елементи матриці знаходяться у одновимірному масиві, а у другому варіанті – двовимірному. Через це майже усі варіанти обробки матриці відрізняються. По перше якщо потрібно перебрати усі елементи матриці для першого варіанту достатньо одного циклу з лічильником, а для другого – двох вкладених. Але для виводу обох матриць бук використаний майже однаковий алгоритм, для того, щоб матриця виводилася у привичному вигляді, а не у рядок(так як вона зберігається у першому варіанті). Відрізняється також індексація елементів, наприклад щоб звернутися до 3го елементу 4го рядку:

У першому випадку фрагмент коду виглядатиме так:

/\*matrix1.items[(3 \* m) + 2]\*/

//m – кількість стовпчиків у матриці.

У другому:

/\*matrix1.items[3][2]\*/

Індекси елементів на одиницю менше через те, що відлік починається з нуля(тож перший елемент першого рядку матиме індекс [0](перший варіант) або [0][0](другий варіант)).

У підсумок можна сказати, що перший варіант можна вважати ультимативним, для простих обробок які не спираються на індекс(однакові для усіх елементів), для інших випадків обидва варіанти потребують двох вкладених циклів для обробки, різниця тільки у індексації.