Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

**з дисципліни «Програмування алгоритмічних структур»**

**Тема «Програмування базових алгоритмів»**

Варіант № \_17\_\_

                 Виконав:

                                     Студент групи ТР-12

                                                                 Каркушевський Владислав Леонідович

Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ –2021

**РОЗРАХУНКОВО-ГРАФІЧНУ РОБОТУ**

**з дисципліни «Програмування алгоритмічних структур»**

***Тема****.* Програмування базових алгоритмів обробки масивів

***Мета роботи:***набуття практичних навичок розробки розгалужених та циклічних алгоритмів при створюванні програмних проектів для розв’язання практичних завдань.

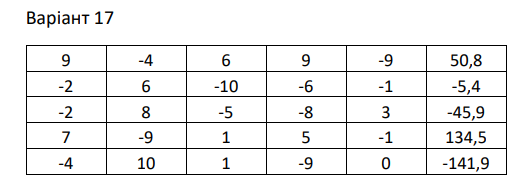
***Завдання на розрахунково-графічну роботу (РГР).***

1. Розробити алгоритми і написати програми мовою Java із застосуванням операторів циклів for, while, do while для розв’язання завдань, поданих в додатках 1, 2 відповідно до індивідуального варіанта. Варіант обрати за списком групи.

2. Розробити програмний проект в середовищі Intellij IDEA для реалізації написаних програм. Перевірити результати обчислень альтернативними розрахунками (наприклад, Excel, калькулятор, тощо).

3. Оформити РГР та вчасно надіслати викладачу на перевірку файл РГР та файл програми.

**Завдання1:**



**Завдання2:**



**Короткий опис завдань:**

Було створено клас ‘’’RgrTr12Drel’’ в якому створено методи “ main” , “task11”, “task12”, “task13” ,“task2”, “bubbleSort”

Також імпортовано бібліотеку “java.util.Scanner” для роботи з сканером.

У методі **“main”** вводимо дані для завдання 1(вводимо розширену матрицю). Викликаємо метод “task13” та виводимо на екран початкову розширену матрицю. Викликаємо метод “task11” в яку передаємо початкову матрицю. Викликаємо метод “task13” та виводимо на екран трикутникову матрицю. Викликаємо метод “task12” в яку передаємо трикутникову матрицю. Викликаємо метод “task13” та виводимо на екран вектор результатів.

Створюємо сканер, вводимо перше число, викликаємо метод “task2”, вводимо друге число, викликаємо метод “task2”, вводимо третє число, викликаємо метод “task2”.

У методі **“task11”** , в який передається початкова розширена матриця системи лінійних алгебраїчних рівнянь (СЛАР) та повертає трикутникову матрицю. В даному методі відбувається прямий хід методу Гауса для розв’язання СЛАР. Всі розрахунки та виведення результатів відбувається в циклі у вкладених циклах “for “ та ‘’ while’’

У методі **“task12”** , в який передається трикутникова матриця та повертає вектор результатів розв’язання СЛАР. В даному методі відбувається зворотний хід методу Гауса для розв’язання СЛАР. Всі розрахунки та виведення результатів відбувається в циклі у вкладених циклах “for “ та ‘’ while’’.

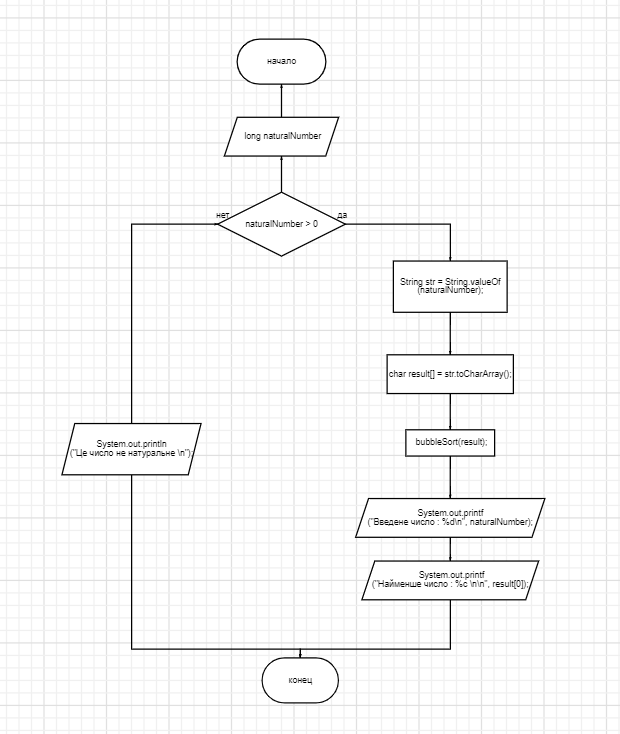
Метод **“task13”** викликається три рази task13, в який передавається по черзі початкову матрицю, трикутникову матрицю, вектор результатів. Метод “task13” є універсальним для виводу матриць і вектору.Метод task13 виконуємо за допомогою перегрузки методів . Виведення виконується за допомогою методу System.out.printf форматованого виводу значень. Для дробових чисел – не більше 3 знаків після коми. Цілі числа виводяться без коми та нулів після коми.

У методі **‘’task2’’**, в якому виконуються всі розрахунки завдання 2 та повернення результатів відбувається в ‘’main’’. Введення даних для завдання 2 зроблено для трьох різних варіантів вхідних даних, що обирає користувач та вводить с клавіатури. Введення даних зроблений за допомогою методів класу Scanner

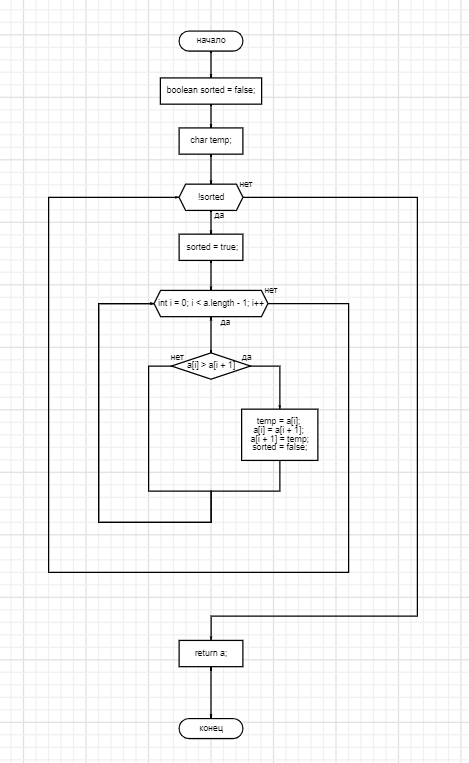
Також створено метод **“bubbleSort”** який нам згодиться для розв’язку завдання 2. Також створенна блок схема для цього методу.

Більш детальніше дії розписані в коментарях коду.

**Блок-схема алгоритму для другого завдання:**



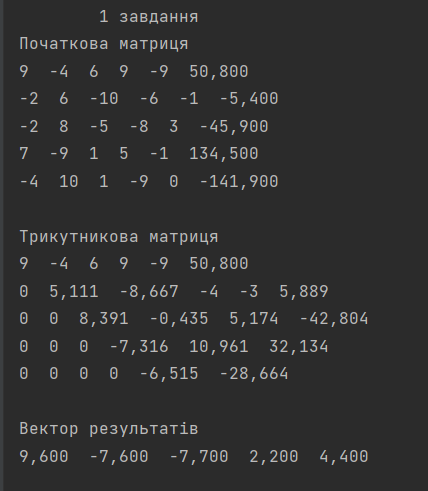
**Блок-схема алгоритму для сортування масиву методом бульбашки**



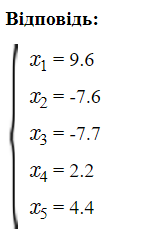
***Програму проекту:***

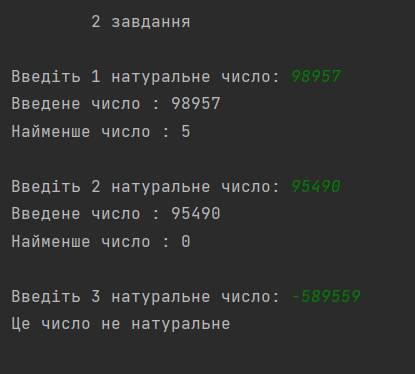
**import** java.util.Scanner;*//бібліотека для введення інформації з клавіатури***public class** RgrTr12Kark {*//створення класу* **public static void** main(String[] args) {*//створення головного метода main* System.***out***.println(**"\t\t1 завдання"**);  
 **float**[][] matrix = {{9, -4, 6, 9, -9, 50.8f},  
 {-2, 6, -10, -6, -1, -5.4f},  
 {-2, 8, -5, -8, 3, -45.9f},  
 {7, -9, 1, 5, -1, 134.5f},  
 {-4, 10, 1, -9, 0, -141.9f}*//створюємо двовимірний масив та заповняємо числами згідно з варіантом* };  
  
  
 System.***out***.println(**"Початкова матриця"**);  
 *task13*(matrix);*//викликаємо метод task13 та виводимо початкову матрицю* **float**[][] deltaMatrix = *task11*(matrix);*//створюємо змінну deltaMatrix якій присвоюємо значення метода task11* System.***out***.println(**"\nТрикутникова матриця "**);  
 *task13*(deltaMatrix);*//викликаємо метод task13 та виводимо трикутникову матрицю* **float**[] result = *task12*(deltaMatrix);*//створюємо змінну result якій присвоюємо значення метода task12* System.***out***.println(**"\nВектор результатів"**);  
 *task13*(result);*//викликаємо метод task13 та виводимо трикутникову матрицю* System.***out***.println(**"\n\n\t\t2 завдання\n"**);  
 Scanner sc = **new** Scanner(System.***in***);*//створюємо сканер* System.***out***.print(**"Введіть 1 натуральне число: "**);  
 **long** var1 = sc.nextLong();*//вводимо перше натуральне число з клавіатури  
 task2*(var1);*//викликаємо метод task2 для першого натурального числа* System.***out***.print(**"Введіть 2 натуральне число: "**);  
 **int** var2 = sc.nextInt();*//вводимо друге натуральне число з клавіатури  
 task2*(var2);*//викликаємо метод task2 для другого натурального числа* System.***out***.print(**"Введіть 3 натуральне число: "**);  
 **int** var3 = sc.nextInt();*//вводимо третє натуральне число з клавіатури  
 task2*(var3);*//викликаємо метод task2 для третього натурального числа* }  
  
 **public static float**[][] task11(**float**[][] a) {*//створюємо метод task11 в який ми будемо поміщати початкову матрицю* **float** m;  
 **int** k = 1;  
 **while** (k < a.**length**) {*//створюємо цикли for та while які будуть звичайну матрицю перетворювати в трикутникову* **for** (**int** i = k; i < a.**length**; i++) {  
 m = (a[i][k - 1] / a[k - 1][k - 1]);  
  
 **for** (**int** j = 0; j < a[0].**length**; j++) {  
 a[i][j] = a[i][j] - m \* a[k - 1][j];  
 }  
 }  
 k++;  
 }  
 **return** a;*//повернення трикутникової матриці* }  
  
 **public static float**[] task12(**float**[][] a) {*//створюємо метод task12 в який ми будемо поміщати трикутникову матрицю* **float**[] x = **new float**[a.**length**];*//створюємо саме двовимірний масив для подальшого виведення в task13* **int** k = 1;  
 **while** (k < a.**length**) {*//створюємо цикли for та while які будуть трикутникову матрицю перетворювати в вектор результатів* **for** (**int** i = a.**length** - 1; i >= 0; i = i - 1) {  
 x[i] = a[i][a.**length**] / a[i][i];  
 **for** (**int** c = a.**length** - 1; c > i; c = c - 1) {  
 x[i] = x[i] - a[i][c] \* x[c] / a[i][i];  
 }  
 }  
 k++;  
 }  
 **return** x;*//повернення вектора результатів* }  
  
 **public static void** task13(**float** [] a) {*///створення методу task13 для виводу одновимірних масивів* **for** (**int** i = 0; i < a.**length**; i++) {*//цикли для перебору елементів масиву* **if** (a[i] / (**int**) a[i] == 1) {  
 System.***out***.printf(**"%.0f "**, a[i]);*//якщо число ціле, то буде виводитися без знаків після коми* } **else if** (a[i] == 0) {  
 System.***out***.printf(**"%.0f "**, a[i]);*//якщо число дорівнює нулю, то буде виводитися без знаків після коми* } **else** {  
 System.***out***.printf(**"%.3f "**, a[i]);*//якщо число дробове, то буде виводитися три знаки після коми* }  
 }  
 }  
  
 **public static void** task13(**float** [][] a) {*//створення методу task13 для виводу двовимірних масивів* **for** (**int** i = 0; i < a.**length**; i++) {  
 **for** (**int** j = 0; j < a.**length** + 1; j++) {*//цикли для перебору елементів масиву* **if** (a[i][j] / (**int**) a[i][j] == 1) {  
 System.***out***.printf(**"%.0f "**, a[i][j]);*//якщо число ціле, то буде виводитися без знаків після коми* } **else if** (a[i][j] == 0) {  
 System.***out***.printf(**"%.0f "**, a[i][j]);*//якщо число дорівнює нулю, то буде виводитися без знаків після коми* } **else** {  
 System.***out***.printf(**"%.3f "**, a[i][j]);*//якщо число дробове, то буде виводитися три знаки після коми* }  
 }  
 System.***out***.println(**" "**);  
 }  
 }  
  
 **public static void** task2(**long** naturalNumber) {*//створюємо метод task2 в який ми будемо поміщати число* **if** (naturalNumber > 0) {*//якщо це число буде більше за нуль, тобто натуральне буде виконуватися такі дії* String str = String.*valueOf*(naturalNumber);*//перетворення типу long в тип String* **char** [] result = str.toCharArray();*//створення масиву символів зі строки str  
  
 bubbleSort*(result);*//сортування методом бульбашки* System.***out***.printf(**"Введене число : %d\n"**, naturalNumber);  
 System.***out***.printf(**"Найменше число : %c \n\n"**, result[0]);  
  
 } **else** {*//в іншому випадку буде виводитися на екран що число не натуральне* System.***out***.println(**"Це число не натуральне \n"**);  
 }  
 }  
  
 **public static char** [] bubbleSort(**char**[] a) {*//метод для сортування одновимірного масиву* **boolean** sorted = **false**;  
 **char** temp;  
 **while** (!sorted) {*// цикл завершиться коли sorted = true* sorted = **true**;  
 **for** (**int** i = 0; i < a.**length** - 1; i++) {*//створення циклу для перебору кожного елементу масиву* **if** (a[i] > a[i + 1]) {*//надання умови сортування* temp = a[i];  
 a[i] = a[i + 1];  
 a[i + 1] = temp;  
 sorted = **false**;  
 }  
 }  
 }  
 **return** a;*//повернення масиву* }  
}

***Результати обчислень:***



Перевірка СЛАР через онлайн калькулятор :





**Висновок:**

На цій графічній розрахунковій роботі було набуто практичні навички розробки розгалужених та циклічних алгоритмів при створюванні програмних проектів для розв’язання практичних завдань. Виконано СЛАР методом Гаусса та визначено найменше число натурального числа.При виконанні роботи було отрацьовано Debug. Також самостійно опрацьовано перегрузку методів.