МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ВІДОКРЕМЛЕНИЙ СТУКТУРНИЙ ПІДРОЗДІЛ

«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ   
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ   
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

**КУРСОВА РОБОТА**

з дисципліни «Основи програмування та АМ»

Дивовижний острів

на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Виконав: здобувач освіти 2 курсу,

25

Конопльов Б.О.

групи \_\_\_\_\_\_\_  
спеціалізації «Розробка програмного забезпечення»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали)

Олійник В.В.

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (підпис) (прізвище та ініціали)

Полтава – 2021

**ЗМІСТ**

[ВСТУП 3](#_Toc74859151)

[1. ПОСТАНОВКА ЗАДАЧІ 4](#_Toc74859152)

[1.1. Постановка задачі 4](#_Toc74859153)

[1.2. Основні вимоги до програми 4](#_Toc74859154)

[2. ОПИС ПРОЕКТНИХ РІШЕНЬ 5](#_Toc74859155)

[2.1. Опис програмних засобів та програмного забезпечення 5](#_Toc74859156)

[2.2. Алгоритм програми 5](#_Toc74859157)

[2.3. Опис специфікації функцій 10](#_Toc74859158)

[2.4. Тестування 15](#_Toc74859159)

[3. ІНСТРУКЦІЯ КОРИСТУВАЧА 19](#_Toc74859160)

[ВИСНОВКИ 22](#_Toc74859161)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 23](#_Toc74859162)

[ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ 24](#_Toc74859169)

# ВСТУП

Мета виконання КР полягає у закріпленні, поглибленні та узагальненні базових теоретичних знань з програмування та їх застосуванні до комплексного вирішення конкретного завдання. Основними завданнями написання курсової роботи є:

* закріплення, поглиблення та узагальнення знань;
* надбання досвіду роботи з літературними та фондовими матеріалами, вміння узагальнювати та аналізувати наукову інформацію, виробляти власне ставлення до наукової чи практичної проблеми;
* набуття навичок використання основ алгоритмізації та програмування на алгоритмічних мовах високого рівня;
* набуття теоретичних знань та практичних навичок в області використання сучасних систем створення програмного забезпечення та освоєння принципів та методів сучасних технологій програмування;
* вироблення вміння застосовувати методи обчислювальної математики та прикладного програмування для розв’язання прикладних задач;
* проведення ґрунтовного аналізу отриманих результатів і формування змістовних висновків стосовно їх якості.

Програма повинна знайти найменшу відстань між секторами кола, на яких знаходяться хижини в залежності від введених даних й вивести результат в файл або на екран. Користувач вводить кількість секторів і номера секторів на яких знаходяться хижини за допомогою вхідного файлу або через спеціальні поля вхідних даних.

# ПОСТАНОВКА ЗАДАЧІ

## Постановка задачі

Дивовижний острів. Якось на уроці географії Розумник почув про незвичайний острів, що має форму круга: посередині острова височіє скеля, а населення живе у хижах уздовж периметра острова і через прямовисність скелі може пересуватися від хижі до хижі також виключно по периметрі. Для зручності вважатимемо, що периметр острова розбито на кілька однакових частин, які умовно назвемо секторами, і з однієї такої частини в сусідню можна перейти рівно за хвилину. У деяких секторах розташовано по хижі (але не більш ніж одна хижа в секторі). Визначте, за який час можна подолати відстань між парою найвіддаленіших хиж на острові.

Вхідні дані: У першому рядку вхідного файлу вказано два натуральних числа *n* та *h* — кількість секторів та хиж на острові відповідно. Відомо, що 2 ⩽ *h* ⩽ *n* ⩽ 500 000. Сектори занумеровано числами від 1 до *n* у тому порядку, в якому вони йдуть на острові (при цьому сектори з номерами 1 та n замикають коло і також є сусідніми). У другому рядку в порядку зростання вказано номери секторів, у яких є хижі.

Вихідні дані: У вихідний файл виведіть єдине число — відстань між двома найвіддаленішими хижами острова, тобто час у хвилинах, за який можна дійти від однієї з цих хиж до іншої.

## 1.2. Основні вимоги до програми

Описати вимоги да програми, до інтерфейсу, до функціональності.

Програма повинна реалізувати можливість взаємодіяти з програмою таким чином, щоб користувач міг реалізовувати свої потреби Програма дає змогу переглядати на екрані постановку задачі, правила, вхідні та вихідні дані, переглянути графічно результати роботи програми. В програмі реалізовано роботу з файлами та введення даних у самій програмі, виведення даних у файл.

Програма реалізована на мові програмування JavaScript в html і css.

# 2. ОПИС ПРОЕКТНИХ РІШЕНЬ

## 2.1. Опис програмних засобів та програмного забезпечення

JavaScript - мультіпарадігменна мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний стилі. Є реалізацією специфікації ECMAScript (стандарт ECMA-262 [7]).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

JavaScript є об'єктно-орієнтованою мовою, але використання в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання - що додає мові додаткову гнучкість.

## 2.2. Алгоритм програми

Запуск програми. Спочатку показується суть програми, правила.

В програмі з’являються кнопки зі знаками питання, алгоритм їх роботи такий: при натисканні на кнопку знака питання, html теги для яких створені ці знаки питання стають видимі(спочатку в програмі вони за замовчуванням не видимі), а в цьому видимому html тегу є кнопка яка приховує цей видимий тег, щоб користувач міг відкривати й закривати htnl теги з потрібним текстом.

Користувач може ввести вхідні дані як через файл, додавши його в спеціальне поле для вхідних даних, так і через спеціальне текстове поле для вхідних даних.

Після введення даних, програма перевіряє чи введені дані коректні і відповідають правилам курсової роботи. Якщо знайдені помилки то програма виведе помилку на екран й зупиниться до нового введення даних користувачем. А якщо введені дані відповідають вимогам то на екран виведяться введені дані й кнопка запуску програми.

Після натискання на кнопку програма створе масив statisticArr й запустить цикл в яком кожну ітерацію будуть обраховуватись відстані між введеними номерами секторів за годинниковою стрілкою, за формулою кінцевий сектор мінус початковий, й проти годинникової стрілки, за формулою кількість секторів мінус кінцевй сектор(щоб дізнатись відстань від кінцевого сектора до початку секторів) й додати початковий сектор, в масив statisticArr записується масив в якому знаходиться 1) номер ітерації + 1(так як ітерації починаються з нуля), 2) номер початкового сектору, 3) номер кінцевого сектору, 4) відстань між секторами за годинниковою стрілкою, 5) відстань між секторами проти годинникової стрілки.

Після створення матриці(масива масивів) запускається функція що графічно зобажує сектора, їх номера, та хижі в введених користувачем номерах секторів.

Потім створюється таблиця, на основі матриці з даними, яка буде відображена на экрані, створюються функції сортування для таблиці й функції видилення секторів на які вказує користувач. Якщо користувач натискає ЛКМ на назву колонки таблиці, то матриця даних сортується по колонці на яку натиснув користувач від більшо значення до меншого стандартною функцієї сортування в JavaScript й допрацьовної спеціально для цієї програми й на екрані перестворюється таблиця з даними із матриці. А якщо натискається SHIFT + ЛКМ, то аналогічно, тільки сортування відбувається від меншого до більшого значення.

Якщо користувач наводить курсор на певний рядок в таблиці, то цей рядок виділяється, змінюється фонове забарвлення рядка й на полотні мальються сектори які знаходяться у виділеному рядку, потім ці сектори обводяться товстим контуром, тим самим й віділяються.

Якщо користувач відводить курсор від рядка, то забарвлення рядка змінюється на попереднє, а полотно перестворюється, й тим самим виділення секторів зкидається.

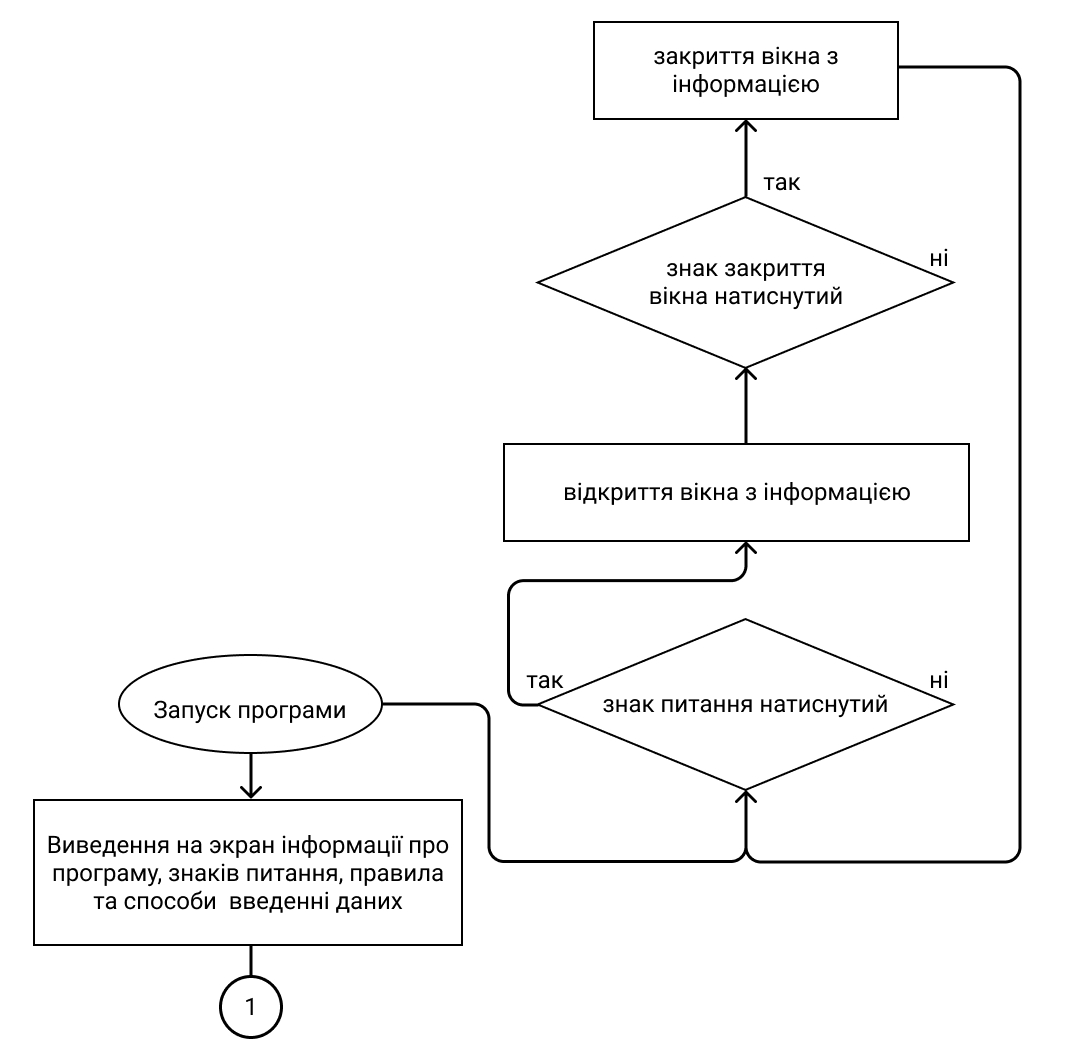


Рисунок 2.1 – Алгоритм роботи програми

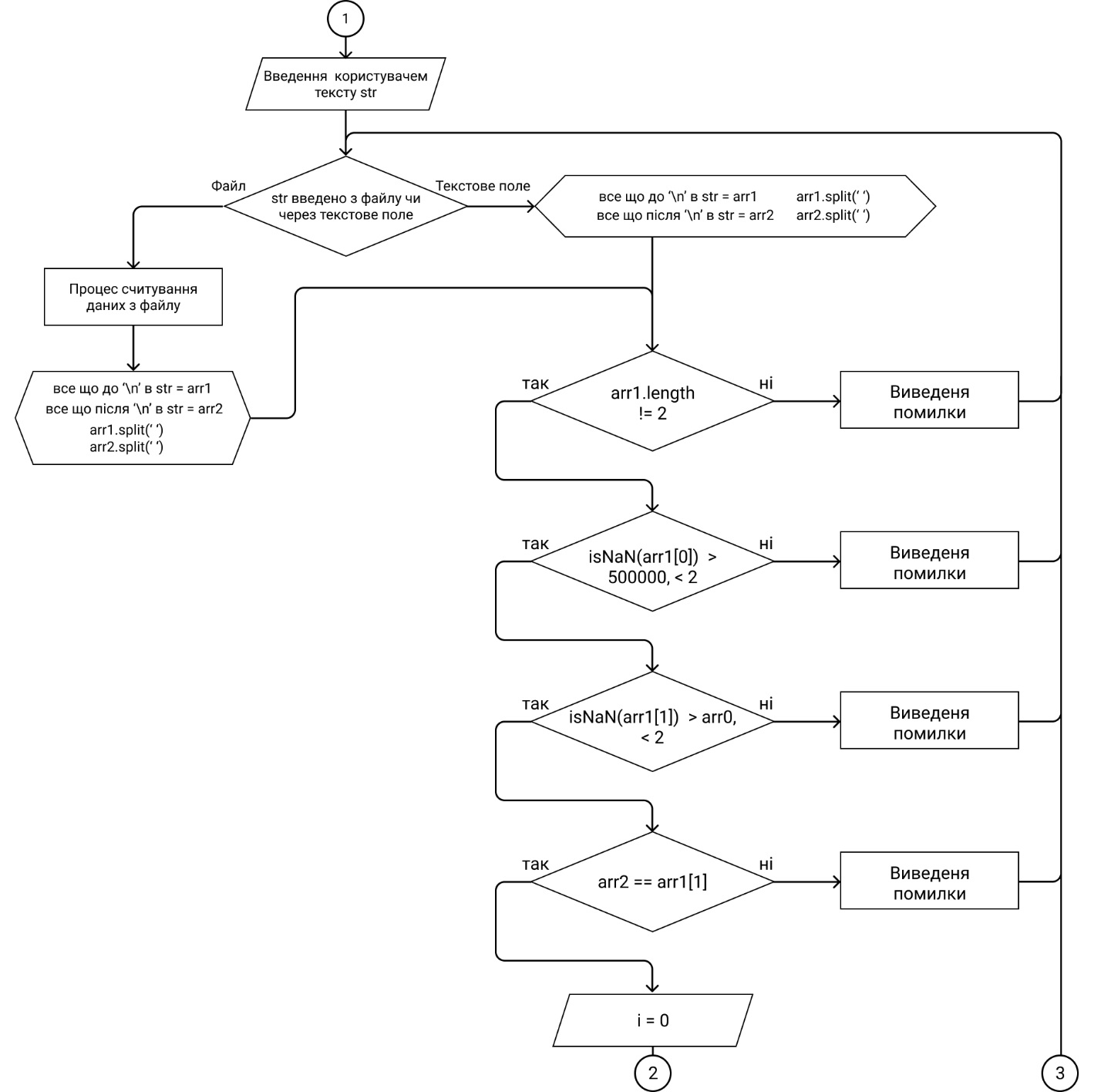


Рисунок 2.1 – Алгоритм роботи програми

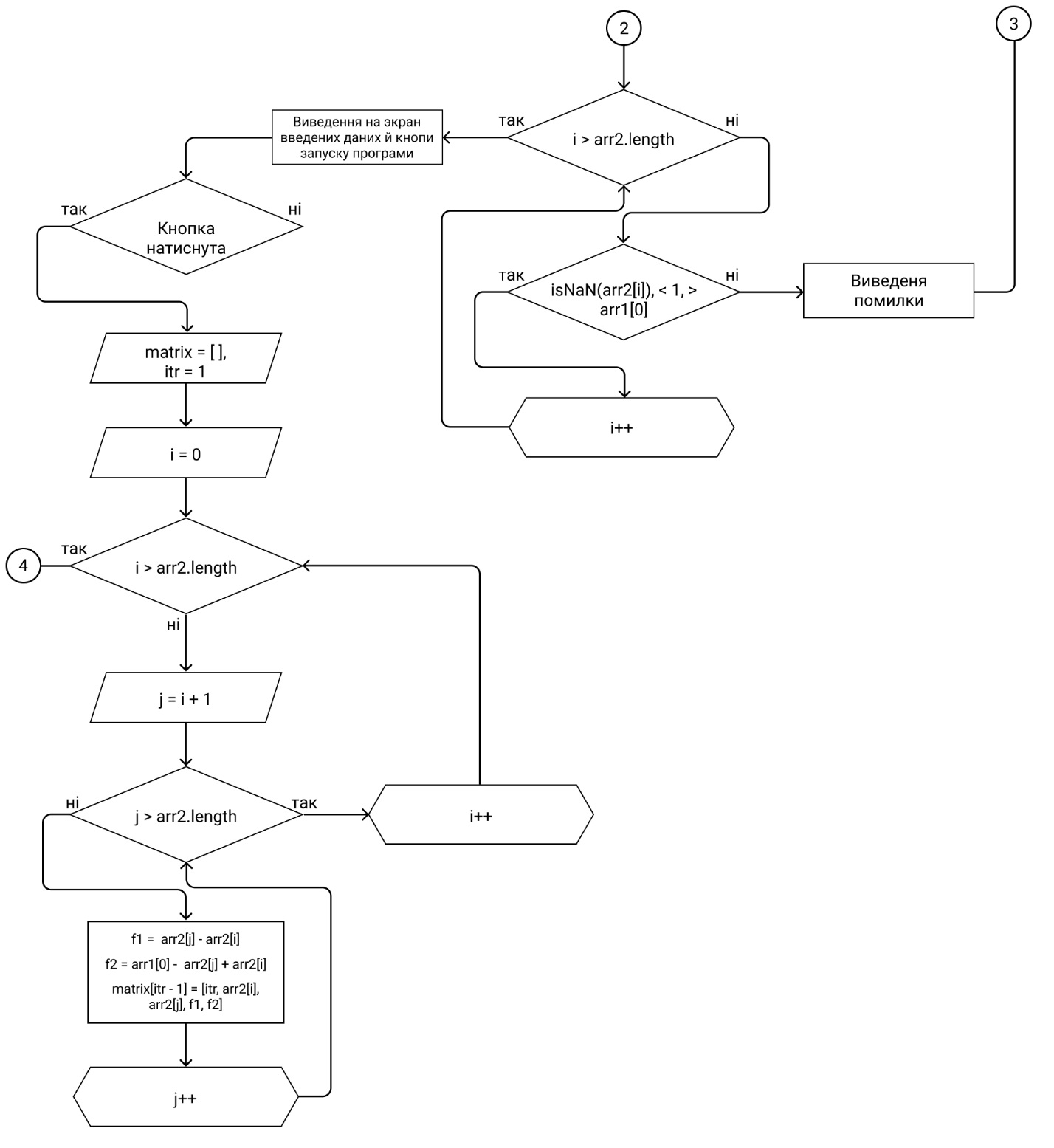


Рисунок 2.1 – Алгоритм роботи програми

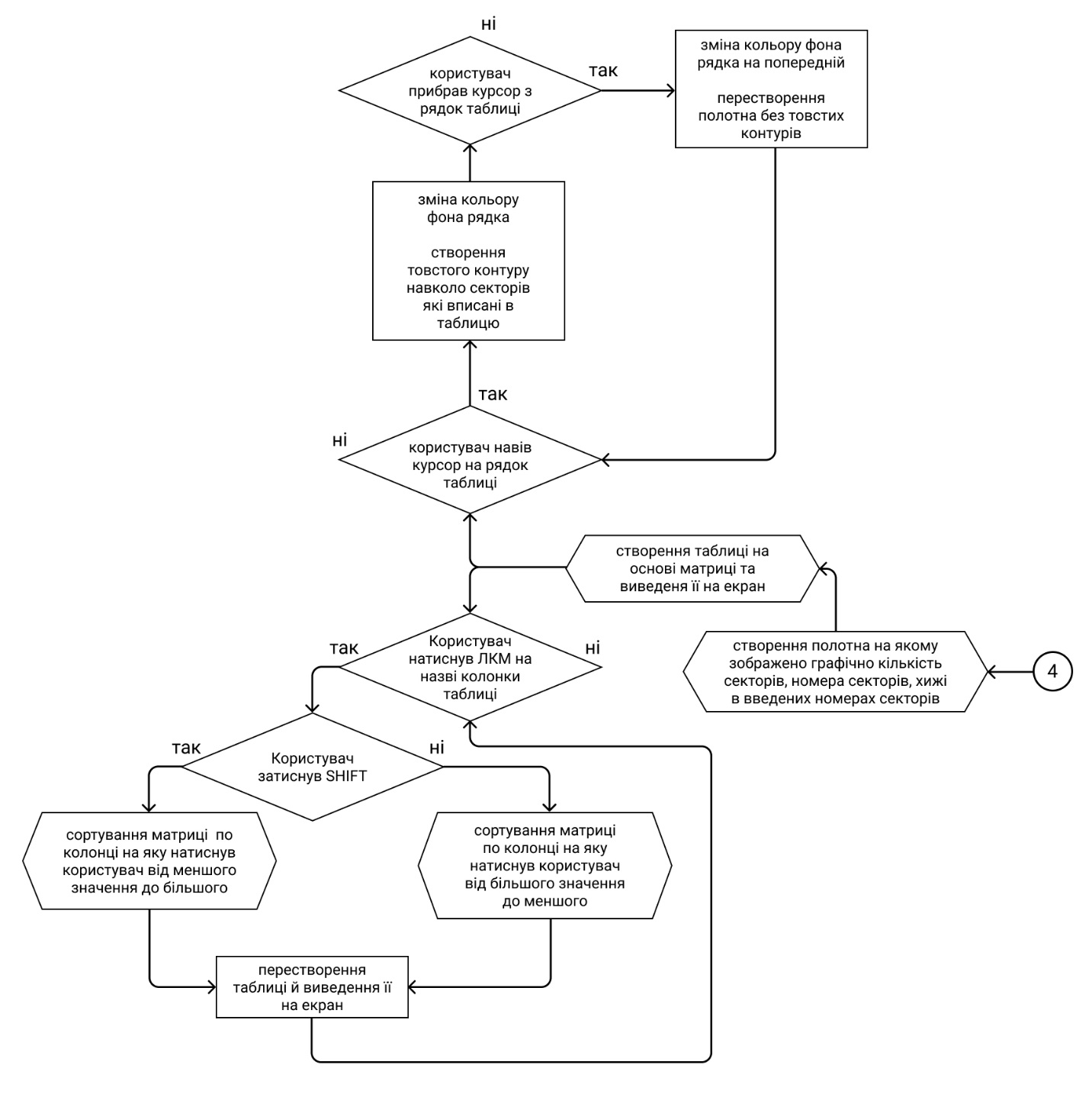


Рисунок 2.1 – Алгоритм роботи програми

## 2.3. Опис специфікації функцій

Таблиця 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | getNode | Повертає об’єкт html тега | node - Назва класу, індифікатору або назва html тега у вигляді тексту | html об’єкт | core.js |

Продовження Таблиці 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | getNode s | Повертає масив об’єктів html тегів | node - Назва класу, або назва html тега у вигляді тексту | Масив html об’єктів | core.js |
| 3 | createNode | Створює й повертає об’єкт html тега | node — назва тегу як текст, innerHTML — текст всередені html тега, cls — назва класу тега, ...attributes — масив масивів інших атребутів тега та їх значень | html об’єкт | core.js |
| 4 | onInputFile | Оброблює вхідні дані введені через файл та присвоєнні цих данних в змінні | немає | немає | input.js |
| 5 | inputedData | Оброблює вхідні дані введені через поля для введення даних та присвоєнні цих данних в змінні | thisNode — об’єкт html тега | немає | input.js |
| 6 | calcInputData | Перевірки що введені дані коректні та відповідають вимогам курсової роботи, вивід на екран даних та кнопки запуску програми або помилок, якщо вхідні дані недопустимі | немає | немає | input.js |

Продовження Таблиці 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | switchPopUp | відображає або приховує на екрані html тег | Type — тип дії, сховати або показати тег, node — html тег на якому буде проведено операції | немає | html.js |
| 8 | sortingStatistic | Сортує матрицю(таблицю) за певною колонкою від більшого до меньшо значення або навпаки | arr — матриця(таблиця) яку треба відсортувати, type — тип сортування, arrArg — номер колонки за якою сортується матриця починаючи з нуля | Сортована матриця | program.js |
| 9 | runProgram | Створення матриці відстаней між хижинами, на номерів хиж, номерів відстаней | немає | Заповнена матриця з даними | program.js |
| 10 | degToRad | Передворення градусів — 90 градусів в радіани | deg - градус | Значення в радіанах | canvas.js |

Продовження Таблиці 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 11 | lineToAngle | Обчислення значення координат через деяку дистанцію й на деякий кут | argX — поточне значення координати х,  argY — поточне значення координати y, angle — кут у градусах на який треба перемістити координати на деяку дистанції, distance — відстань на яку треба перемістити координати | Масив зі значенням координат x, y | canvas.js |
| 12 | runCanvas | Створює полотно на якому графічно зображаються дані | немає | немає | canvas.js |
| 13 | runStatistic | Створює об’єкт html тега, таблицю в якій будуть знаходитися дані функції runProgram та виведення таблиці на екран | немає | немає | statistic.js |
| 14 | sortFunc | Виведення на екран таблицю сортовану функцією sortingStatistic | e — об’єкт події де відбулося натискання ЛКМ | немає | statistic.js |
| 15 | highlightFunc | Виділення на полотні тих секторів на які наведений курсор користувача в таблиці даних та виділення рядка таблиці на який наведений курсор | node — об’єкт html тега | немає | statistic.js |

Продовження Таблиці 2.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 16 | outHighlightFunc | Перестворення полотна без виділення секторів та скасування виділення рядка таблиці на який був наведений курсор | node — об’єкт html тега | немає | statistic.js |

Так як в JavaScript немає бібліотек й їх файлів, то стандартні функції цієї мови належать певним об’єктам: String, Number, Math, ...

Таблиця 2.2 Опис специфікації функцій

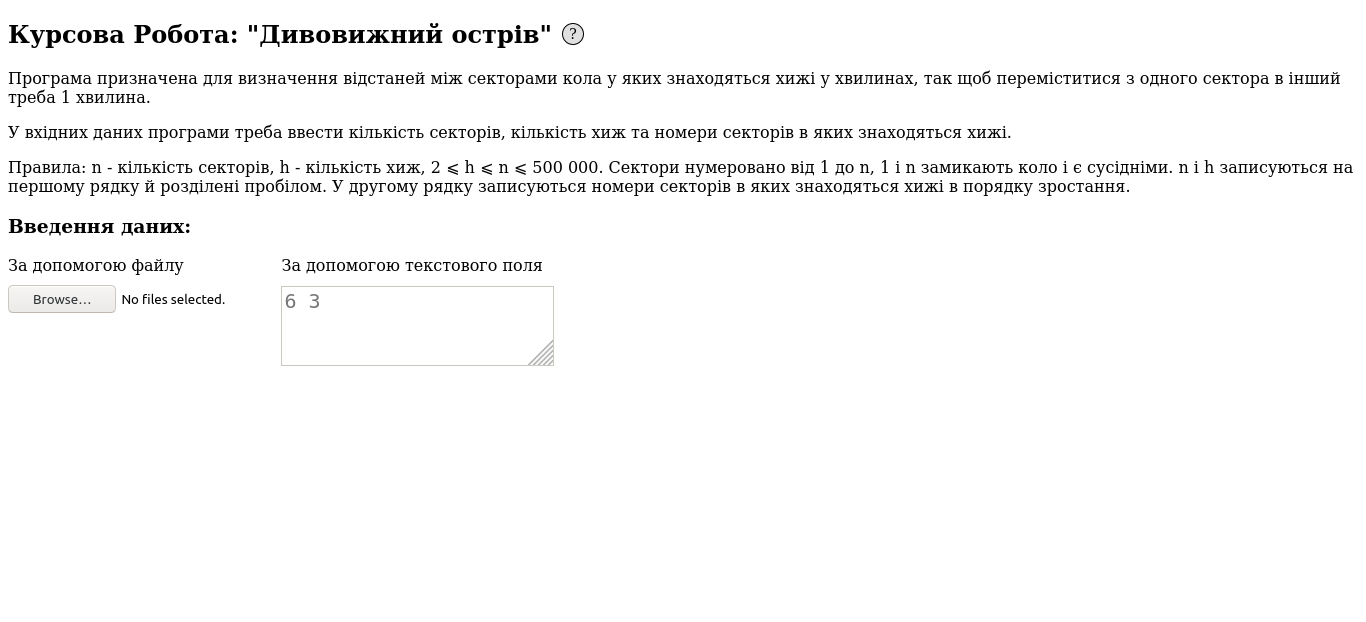
|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Заголовний файл |
| 1 | split | Повертає масив елементів з рядка які розділені аргументом | String |
| 2 | querySelector | Повертає перший об’єкт html тега вказавши в аргументі його клас, індифікатор або його назву | Object |
| 3 | querySelectorAll | Повертає масив об’єктів html тегів вказавши в аргументі їх клас, або назву | Object |
| 4 | createElement | Створює об’єкт html тега | Object |
| 5 | setAttribute | Додає атрибути з певними значеннями в html тег | Object |
| 6 | FileReader | Запускає подію зчитування даних з файлу | Object |
| 7 | readAsText | Зчитує данні файлу як текст | Object |
| 8 | addEventListener | Додає слухання події, якщо подія відбувається то запускається певна функція, вказана в аргументі | Object |
| 9 | remove | Видаляє html тег з html документа | Object |
| 10 | appendChild | Додає в html документ html тег вказаний в аргументі в кінець контенту тега до якого було викликано функцію | Object |
| 11 | isNaN | Перевіряє чи є аргумент функції числом або чи можна аргумент перетворити в число й повертає значення булевого типу, якщо аргумент не число то true, інакше false | Number |
| 12 | sort | Сортує масив за критеріями вказаними в функції переданій як аргумент цій функції сортування | Array |

Продовження Таблиці 2.2

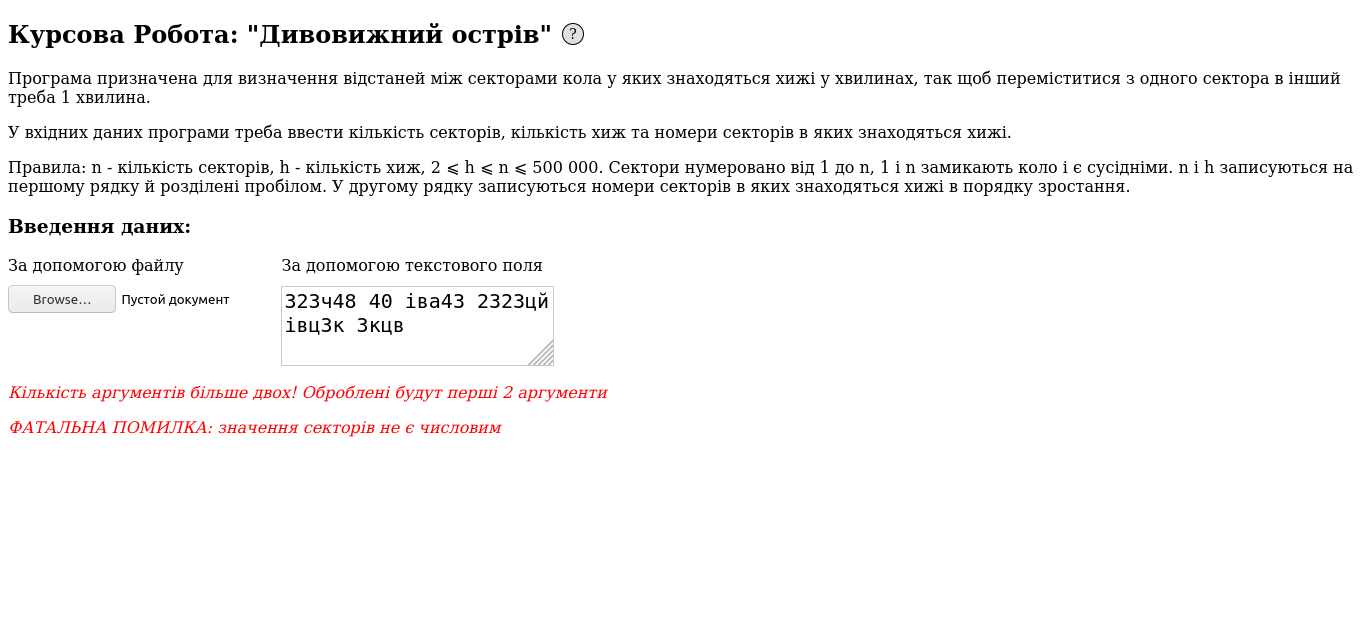
|  |  |  |  |
| --- | --- | --- | --- |
| 13 | cos | Повертає косинус аргументу переданого як радіани | Math |
| 14 | sin | Повертає синус аргументу переданого як радіани | Math |
| 15 | prepend | Додає в html документ html тег вказаний в аргументі в початок контенту тега до якого було викликано функцію | Object |
| 16 | getContext | Дозволяє виконувати функції малювання на полотні, аргументом вказується тип виміру полотна(‘2d’ або ‘3d’) | Object |
| 17 | fillRect | Дозволяє намалювати на полотні прямокутник й заповнити його кольором | Object |
| 18 | moveTo | Дозволяє перемістити “умовний пензлик” в певні координати | Object |
| 19 | arc | Дозволяє намалювати коло | Object |
| 20 | lineTo | Дозволяє намалювати лінію до певних координат | Object |
| 21 | fill | Дозволяє заповнити кольором намальовану фігуру | Object |
| 22 | stroke | Дозволяє намалювати контур намальованої фігури | Object |
| 23 | fillText | Дозволяє намалювати текст й зафарбувати його | Object |
| 24 | beginPath | Дозволяє створити новий шлях(почати нову фігуру) | Object |

## Тестування

При запуску програми, відкривається таке вікно(рисунок 2.4), користувач може ввести дані файлом або за допомогою текстового поля.

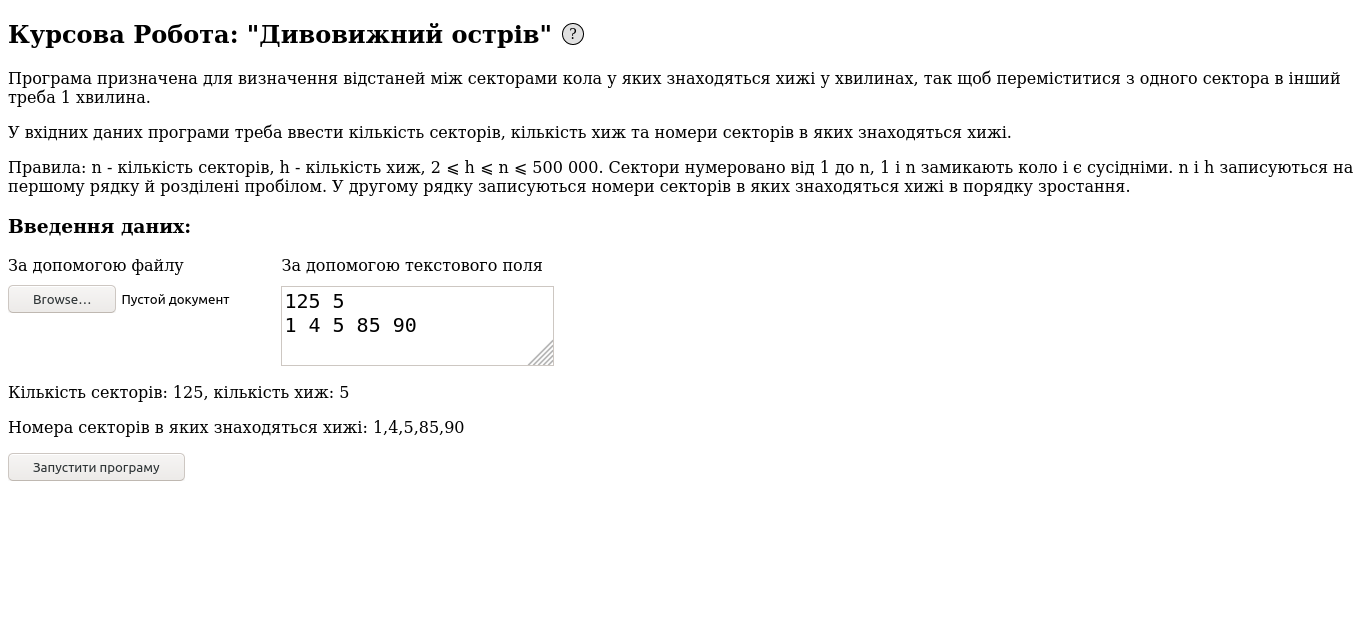
Рисунок 2.4 – Екран запущеної програми

При введені некоректних даних, програма буде повідомляти про помилки й не буде працювати далі, а чекати поки користувач введе коректні данні(рисунок 2.5).

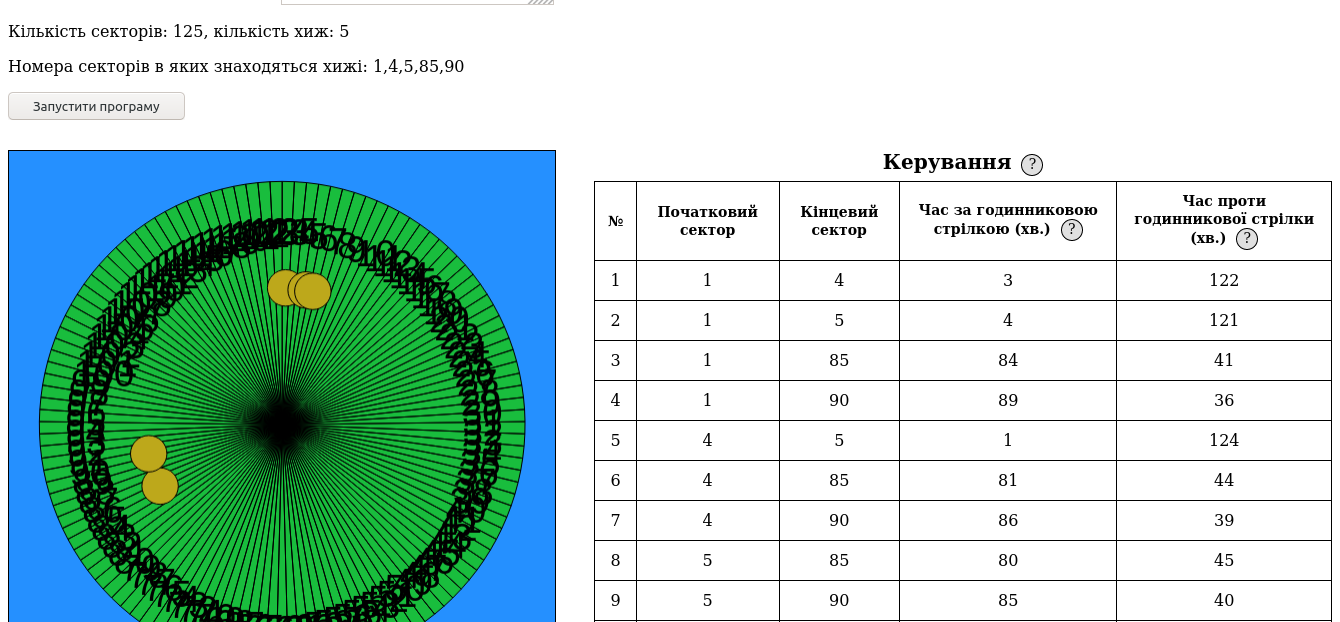
Рисунок 2.5 – Введення некоректних даних

Програма буде повідомляти про помилки у введених даних до тих пір поки не будуть введені коректні дані.

Коли ж користувач введе коректні дані, то програма виведе введені дані й кнопку запуска програми(рисунок 2.6).

Рисунок 2.6 – Введення коректних даних

Після натискання на кнопку програма зобразить графічно введені дані й побудує таблицю відстаней між секторами з хижинами(рисунок 2.7).

Рисунок 2.7 – Запущена програма

Оскілки введено було велике число секторів(125), й вони мають малу ширину, номери секторів накладаються одне на одне й хижі виходять за границі секторів. Якщо ввести значеня секторів менше, то графічне зображення буде більш зрозуміле(рисунок 2.8).

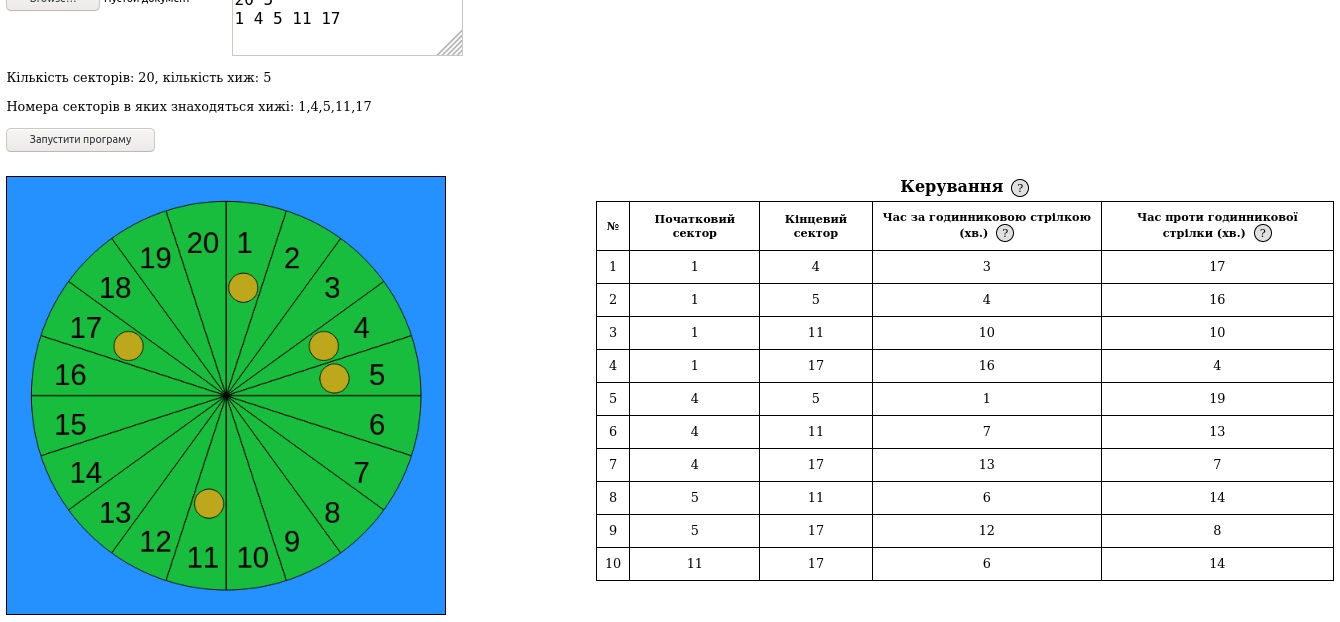
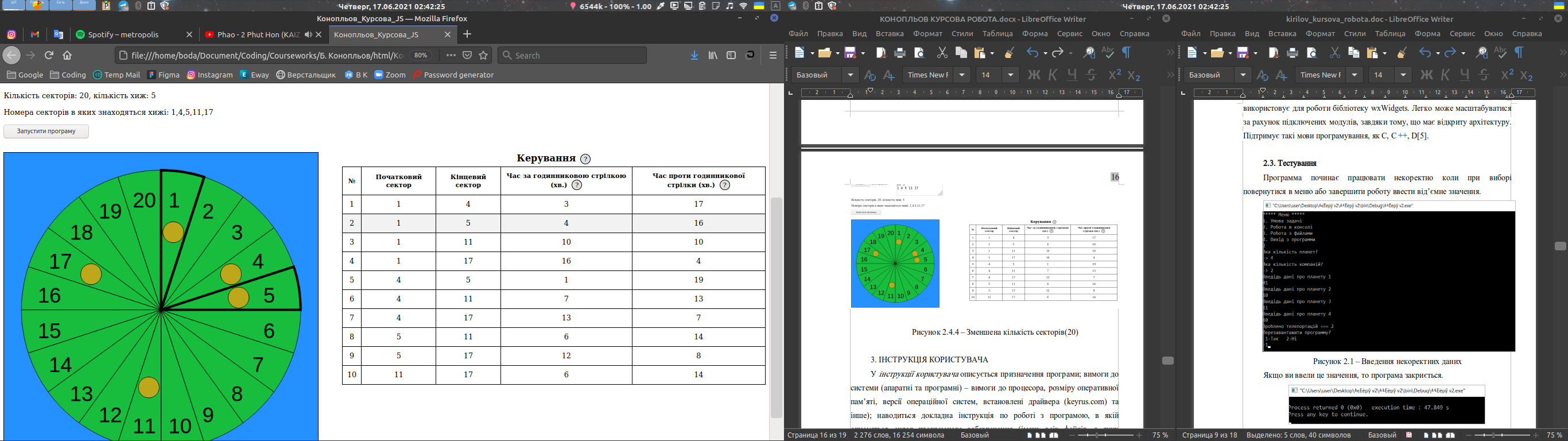


Рисунок 2.8 – Зменшена кількість секторів(20)

Якщо навести курсор в таблиці на рядок, то цей рядок виділиться й на графічному зображені сектори теж будуть виділені(рисунок 2.9).

Рисунок 2.9 – Виділення секторів й рядка

Якщо натиснути ЛКМ по назві колонки в таблиці, то таблиця сортується відносно цієї колонки від більшого значення до меншого, а якщо натискати SHIFT + ЛКМ то сортування відбувається від меншого значення до більшого.

При натисканні на кнопки знаків питання відкривається вікно з додатковою інформацією.

# 3. ІНСТРУКЦІЯ КОРИСТУВАЧА

Програма призначена для знаходження найбільшої відстані між хижами на острові.

Системні вимоги: процесор - intel celeron j1800 або гірше, оперативна пам’ять — 200 мб або більше, версії операційної системи — windows xp або новіші, linux, Chromium OS, Mac OS, Android, драйвери — стандартні драйвери які встановлюються зі встановленням OS.

Програма складається з восьми файлів:

1) Конопльов\_Курсова\_JS.html — 5.5кб головний файл який за допомогою інших файлів виводить програму на екран, дає змогу вводити й виводити дані;

2) Styles.css — 2.1кб файл стилів до файлу Конопльов\_Курсова\_JS.html, стилізує програму, робить її більш красивою;

3) core.js — 546б файл-ядро, в ньому зберігаються найважливіші функції, змінні, тощо, які використовуються у всій програмі;

4) inputs.js — 6.1кб в файлі зберігаються функції які дають змогу вводити дані в програму й первинно їх оброблювати;

5) html.js — 219б файл з функціями для більш комфортного користування те що виводиться на екран;

6) program.js — 822б в файлі зберігаються функції які оброблюють вхідні дані й вираховують час пересування між секторами;

7) canvas.js — 3.3кб створює полотно й виводить в нього графічно введені дані;

8) statistic.js — 2.2кб створює таблицю вихідних даних й взаємодії з таблицею.

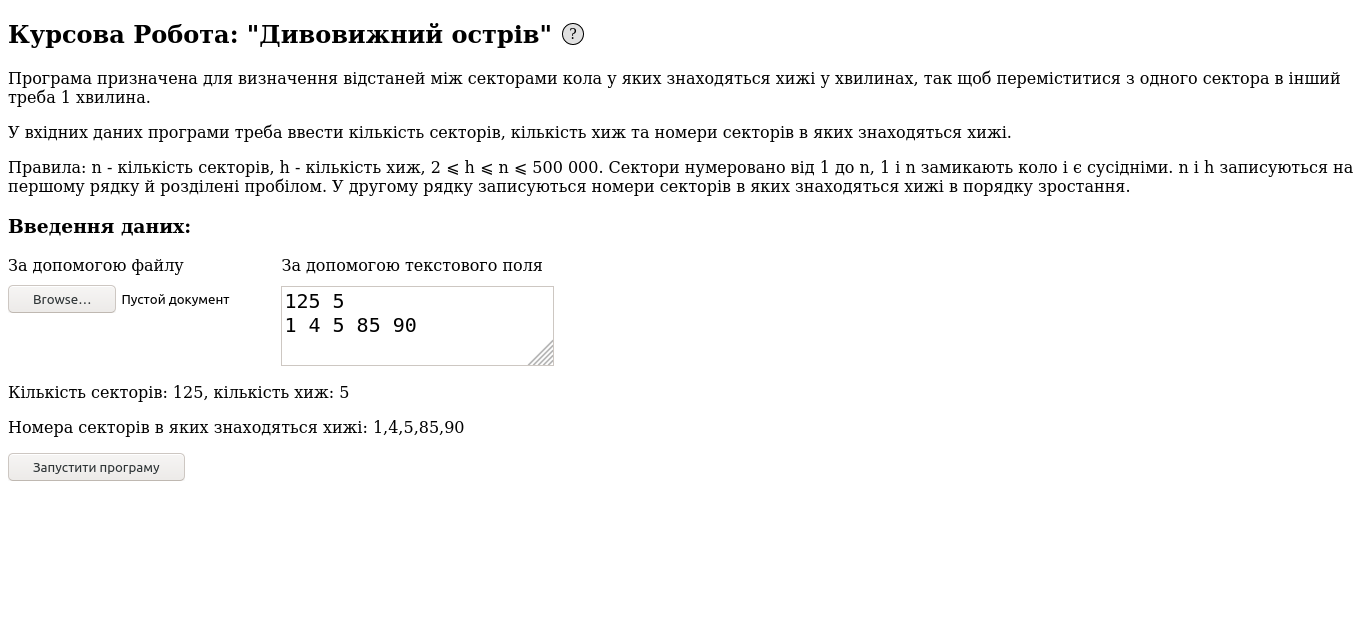
Інтерфейс програми складається з тексту, текстових полів, текстових полів для вводу даних, поля для введення файлу, полотна й таблиці.

Інструкція по використанню:

1) запустити програму;

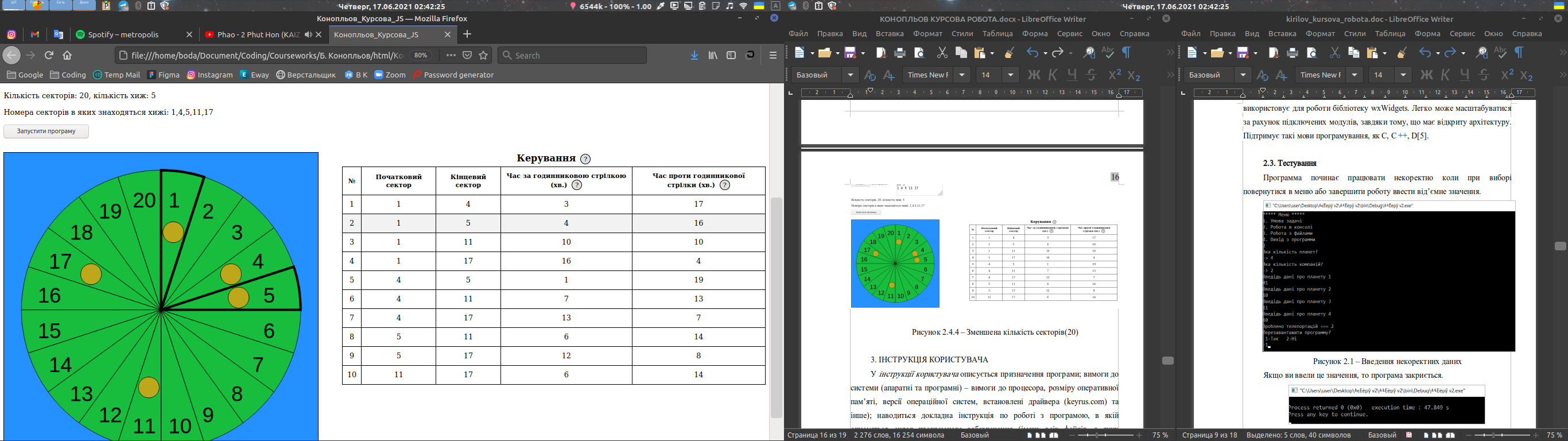
2)ввести дані, через файл або в текстовому полі;

3)Натиснути кнопку “запустити програму”(рисунок 3.1);

Рисунок 3.1 – Введення даних 

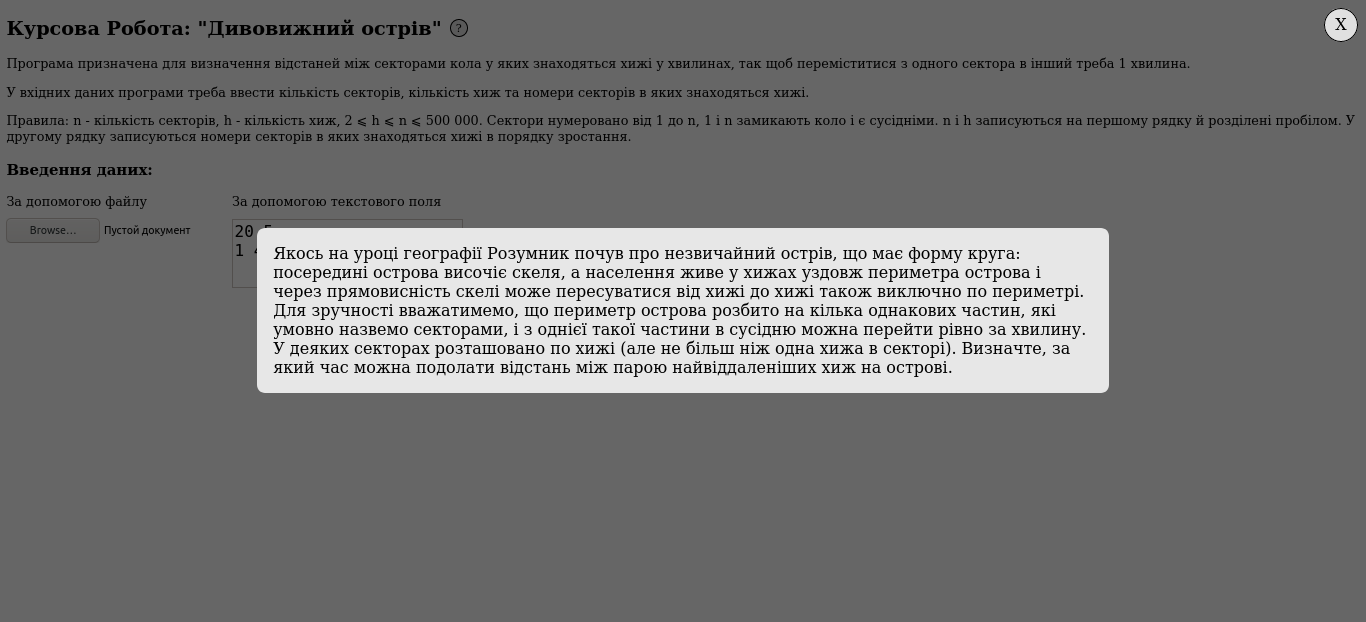
4)для сортування даних по певній колонці, треба натиснути на назву цією колонки;

5)для виділення початкового й кінцевого секторів, треба навести курсор на рядок таблиці з певними початковим й кінцевим секторами(рисунок 3.2);

Рисунок 3.2 – Виділелння секторів й рядка таблиці 

6)для відкриття вікна з інформацією, треба натиснути на кнопку з зображенням знака питання(Рисунок 3.3);

7)для закриття вікна з інформацією, треба натиснути на кнопку з зображенням хреста (Рисунок 3.3).

Рисунок 3.3 – Відкривання вікна з інформацією 

# ВИСНОВКИ

В ході створення курсової роботи була створена програма, задача якої знаходити відстані між секторами в яких знаходяться хижі. Програма створена на мові програмування JavaScript в html документі з додаванням css стилів. Середовище написання програми – редактор коду “Atom”.

В програмі реалізовано:

* введеня даних за допомогою файлу;
* введеня даних за допомогою текстового поля;
* обробка помилок при введені даних;
* виведеня введених даних у графічному вигляді;
* виведеня даних у табличному вигляді;
* взаємодія користувача з інтерфейсом програми;
* відкривання та закривання інформаційних вікон;
* швидкий й зручний спосіб зміни вхідних даних;

Під час написання курсової роботи був отриманий досвід написання курсових робіт з програмування, з програмування html canvas для виведення даних графічно, з програмування змін у html canvas у реальному часі, з реалізації даних у табличному виді, зі зміни таблиць даних у реальному часі, з сортування даних у таблиці за колонкою, з програмування спливаючих вікон, з обробки помилок в програмі. Отримав досвід з реалізації взаємодії файлової системи з JavaScript.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

### [JavaScript — Википедия](https://ru.wikipedia.org/wiki/JavaScript) [Електронний ресурс] – Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/JavaScript>

### learn.javascript.ru – [Введение в JavaScript](https://learn.javascript.ru/intro) Електронний ресурс] – Режим доступу до ресурсу: <https://learn.javascript.ru/intro>

### [JavaScript Основы для Начинающих – Полный Курс за 6 часов [2020]](https://www.youtube.com/watch?v=Bluxbh9CaQ0) [Електронний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=Bluxbh9CaQ0

### learn.javascript.ru – Строгий режим – "use strict" [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.javascript.ru/strict-mode>

### learn.javascript.ru – Условное ветвление: if, '?' [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.javascript.ru/ifelse>

### learn.javascript.ru – Циклы while и for [Електронний ресурс] – Режим доступу до ресурсу: [https://learn.javascript.ru/while-for](https://learn.javascript.ru/ifdelse)

# ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

*\* файл Конопльов\_Курсова\_JS.html головний файл який за допомогою інших файлів виводить програму на екран, дає змогу вводити й виводити дані.*

<!DOCTYPE html> <!-- говоримо браузеру що стрінка версії html5 -->

<html lang="en"> <!-- тег в якому зберігається весь документ -->

<head> <!-- тег в якому зберігається все те що не буде відображено на сторінкі, рім тегу title -->

<meta charset="UTF-8"> <!-- сторінка в кодуванні UTF-8 -->

<title>Конопльов\_Курсова\_JS</title> <!-- назва сторінки -->

<link rel="stylesheet" href="Styles.css"> <!-- підключення css стилів -->

</head> <!-- </назва тегу> це закриття парних тегів -->

<body width=100px> <!-- тег в якому зберігається весь відображаємий контент сторінки -->

<h2 class="curseWorkTitle">Курсова Робота: "Дивовижний острів"<div class="button" onclick="switchPopUp('show', getNode('#WonderIslandDescript'))">?</div></h2> <!-- створення заголока курсової -->

<div class="popUp" id="WonderIslandDescript"> <!-- створення popup контейнера з текстов всередені й кнопкою закриття цього попапа, пояснення курсової -->

<div>Якось на уроці географії Розумник почув про незвичайний острів, що має форму круга: посередині острова височіє скеля, а населення живе у хижах уздовж периметра острова і через прямовисність скелі може пересуватися від хижі до хижі також виключно по периметрі. Для зручності вважатимемо, що периметр острова розбито на кілька однакових частин, які умовно назвемо секторами, і з однієї такої частини в сусідню можна перейти рівно за хвилину. У деяких секторах розташовано по хижі (але не більш ніж одна хижа в секторі). Визначте, за який час можна подолати відстань між парою найвіддаленіших хиж на острові.</div>

<div class="button closeButton" onclick="switchPopUp('hide', getNode('#WonderIslandDescript'))">X</div> <!-- кнопа закриття попапа -->

</div>

<div class="popUp" id="clockwiseFuncInfo"> <!-- попап формули обчислення відстані за годдиниковою стрілкою-->

<div>Формула за якою обчислюється час переміщення з початкового сектора(s) в кінцевий(e) за годинниковою стрілкою(t-clockwise): кінцевий сектор відняти початковий. t-clockwise = s - e</div>

<div class="button closeButton" onclick="switchPopUp('hide', getNode('#clockwiseFuncInfo'))">X</div>

</div>

<div class="popUp" id="anticlockwiseFuncInfo"> <!-- попап формули обчислення відстані проти годдиникової стрілки-->

<div>Формула за якою обчислюється час переміщення з початкового сектора(s) в кінцевий(e) проти годинникової стрілки(t-anticlockwise): кількість секторів(a) відняти кінцевий сектор(щоб дізнатись час від кінцевого сектора до початку) і додати початковий сектор. t-anticlockwise = a - s + e</div>

<div class="button closeButton" onclick="switchPopUp('hide', getNode('#anticlockwiseFuncInfo'))">X</div>

</div>

<div class="popUp" id="statisticControls"> <!-- попап керування в таблиці статистики -->

<div>Натискання лівою кнопкою миші(ЛКМ) по назві колонки сортує колонку від найбільшого значення до найменшого, ЛКМ + SHIFT сортує колонку навпаки - від найменшого значення до найбільшого.</div>

<div class="button closeButton" onclick="switchPopUp('hide', getNode('#statisticControls'))">X</div>

</div>

<!-- просто текст -->

<p>Програма призначена для визначення відстаней між секторами кола у яких знаходяться хижі у хвилинах, так щоб переміститися з одного сектора в інший треба 1 хвилина.</p>

<p>У вхідних даних програми треба ввести кількість секторів, кількість хиж та номери секторів в яких знаходяться хижі.</p>

<p>Правила: n - кількість секторів, h - кількість хиж, 2 ⩽ h ⩽ n ⩽ 500 000. Сектори нумеровано від 1 до n, 1 і n замикають коло і є сусідніми. n і h записуються на першому рядку й розділені пробілом. У другому рядку записуються номери секторів в яких знаходяться хижі в порядку зростання.</p>

<h3>Введення даних:</h3>

<!-- введення даних через файл -->

<div class="dataInput">

<div>

<label for="inputFile">За допомогою файлу</label> <!-- назва інпута -->

<input type="file" multiple="true" id="inputFile" onchange="onInputFile(); setTimeout(calcInputData, 10)"> <!-- інпута типу файл -->

</div>

<!-- введення даних через текстове поле -->

<div>

<label for="inputTextArea">За допомогою текстового поля</label>

<textarea rows="2" cols="20" id="inputTextArea" oninput="inputedData(this); calcInputData()" placeholder="6 3"></textarea> <!-- інпута типу текстове поле -->

</div>

</div>

<!-- підключення JavaScript скриптів -->

<script src="core.js" charset="utf-8"></script>

<script src="inputs.js" charset="utf-8"></script>

<script src="html.js" charset="utf-8"></script>

<script src="program.js" charset="utf-8"></script>

<script src="statistic.js" charset="utf-8"></script>

<script src="canvas.js" charset="utf-8"></script>

<!-- <script>

'use strict'

firstSettings = '10 5'.split(' ')

numberSectors = '1 4 5 7 9'.split(' ')

calcInputData()

runProgram()

</script> -->

</body>

</html>

\*  *Styles.css файл стилів до файлу Конопльов\_Курсова\_JS.html, стилізує програму, робить її більш красивою.*

*/\* різні стилі, щоб сторінка виглядала красиво\*/*

*body {*

*/\* overflow: hidden; \*/*

*overflow-x: hidden;*

*}*

*/\* MESSAGES STYLES \*/*

*.errorMessage {*

*color: red*

*font-style: italic;*

*}*

*/\* CURSEWORK BODY STYLES \*/*

*.curseWorkTitle {*

*display: flex;*

*align-items: center;*

*}*

*.button {*

*border: 1px solid;*

*border-radius: 100px;*

*display: inline-flex;*

*justify-content: center;*

*align-items: center;*

*width: 20px;*

*height: 20px;*

*margin-left: 10px;*

*background-color: #e0e0e0;*

*cursor: pointer;*

*font-size: 14px;*

*font-weight: normal;*

*}*

*.button:hover {*

*font-weight: bold;*

*border: 2px solid;*

*margin-left: 9px;*

*}*

*.popUp {*

*position: fixed;*

*top: 0;*

*left: 0;*

*z-index: 1;*

*display: -webkit-flex;*

*display: -ms-flex;*

*display: none;*

*justify-content: center;*

*-ms-align-items: center;*

*align-items: center;*

*width: 100vw;*

*height: 100vh;*

*background-color: rgba(0, 0, 0, 0.6);*

*overflow-y: auto;*

*}*

*.popUp div:first-child {*

*width: 60%;*

*font-size: 20px;*

*padding: 20px;*

*background-color: #e7e7e7;*

*border-radius: 10px;*

*}*

*.closeButton {*

*position: fixed;*

*top: 10px;*

*right: 10px;*

*width: 40px;*

*height: 40px;*

*font-size: 20px;*

*margin-left: 0;*

*}*

*.closeButton:hover {*

*top: 9px;*

*right: 9px;*

*}*

*/\* INPUTS STYLES \*/*

*label {*

*margin-bottom: 10px;*

*display: block;*

*}*

*#inputTextArea {*

*font-size: 20px;*

*}*

*.dataInput {*

*width: 100vw;*

*display: -webkit-flex;*

*display: -ms-flex;*

*display: flex;*

*-webkit-flex-wrap: wrap;*

*-ms-flex-wrap: wrap;*

*flex-wrap: wrap;*

*}*

*.runButton {*

*display: block;*

*margin-bottom: 30px;*

*}*

*/\* PROGRAM STYLES \*/*

*.programNodesContainer {*

*display: -webkit-flex;*

*display: -ms-flex;*

*display: flex;*

*justify-content: space-between;*

*}*

*.statisticTable {*

*border-collapse: collapse;*

*height: min-content;*

*display: block;*

*text-align: center;*

*}*

*.statisticTable {*

*width: 54vw;*

*}*

*.statisticTable caption {*

*font-size: 20px;*

*font-weight: bold;*

*margin-bottom: 5px;*

*}*

*.statisticTable th {*

*font-size: 14px;*

*}*

*.statisticTable th:hover {*

*cursor: pointer;*

*background-color: #f2f2f2;*

*}*

*.statisticTable td, .statisticTable th {*

*border: 1px solid;*

*padding: 10px;*

*}*

*\* файл core.js файл-ядро, в ньому зберігаються найважливіші функції, змінні, тощо, які використовуються у всій програмі.*

*'use strict' // запуск скрипта в строгому режимі*

*// CORE FUNCTIONS*

*function getNode(node) { // функція яка повертає тег за назвою, node - назва тегу*

*return document.querySelector(node) // повертає хтмл тег за назвою*

*}*

*function getNodes(node) { // функція яка повертає масив тегів за назвою*

*return document.querySelectorAll(node)*

*}*

*function createNode(node, innerHTML = '', cls = '', ...attributes) { // створення хтмл тега, node - вид тегу, innerHTML - внутрішній контент тегу, cls - назва класу тегу, ...attributes - масив інших атрибутів тега*

*const tempNode = document.createElement(node) // створення тимчасової константи в якій зберігається хтмл тег*

*cls != 0 ? tempNode.classList.add(cls) : {} // якщо cls не пуста то тимчасовому тегу додається клас з таким ім'ям*

*innerHTML != 0 ? tempNode.innerHTML = innerHTML : {} // аналогічно як з cls тільки внутрішнім контентом тега*

*for(let atribute in attributes) { // цикл присвоєння інших атрибутів тегу з масива ...attributes*

*const attrb = attributes[atribute]*

*tempNode.setAttribute(attrb[0], attrb[1])*

*}*

*return tempNode // повернення тега*

*}*

*\* файл inputs.js файлі зберігаються функції які дають змогу вводити дані в програму й первинно їх оброблювати.*

*'use strict'*

*//firstSettings - перший рядок вхідних даних, numberSectors - другий рядок вхідних даних, sectors - кількість секторів, huts - кілкість хиж*

*let firstSettings = null, numberSectors = null, sectors = null, huts = null*

*// FILE INPUTS*

*const inputFile = getNode('#inputFile') // присвоєння змінній тега*

*function onInputFile() { // функція зчитування даних з фалу*

*const files = inputFile.files, // files масив файлів які були предані в інпут типу файл*

*reader = new FileReader() // запуск події зчитування даних з файла, це займе якийсь час*

*reader.readAsText(files[0]) // зчитування даних з файлу як текст*

*reader.addEventListener('load', () => { // запуск кода, коли зчитування даних з файлу завершиться*

*firstSettings = reader.result.split(' ') //THIS EVENT FILLS SOME TIME!!! AND NEXT CODE NEED CONTINUE VIE SOME TIME (10MS FIT)*

*firstSettings = reader.result.split('\n')[0] // розділення рядка reader.result на окремі елементи там де стоїть символ \n*

*numberSectors = reader.result.split('\n')[1]*

*firstSettings = firstSettings.split(' ') // розділення елементів масиву по символу ' '*

*numberSectors === undefined ? numberSectors = '' : numberSectors = numberSectors.split(' ') // якщо змінна numberSectors не дорівнює undefined, то вона буде оброблена*

*getNode('#inputTextArea').value = reader.result // значення текстового поля для введеня даних стає рівне введеним даним з файлу*

*})*

*}*

*// TEXTAREA INPUTS*

*function inputedData(thisNode) { // запис даних з текстового поля*

*firstSettings = thisNode.value.split('\n')[0]*

*numberSectors = thisNode.value.split('\n')[1]*

*firstSettings = firstSettings.split(' ')*

*numberSectors === undefined ? numberSectors = '' : numberSectors = numberSectors.split(' ')*

*}*

*// TREATMENT INPUTS VALUES, CHECK FOR TRUE*

*function calcInputData() { // перевірка що введені дані коектні*

*let fatalError = false // зміна для фатальних помилок*

*// REFRESHING*

*getNode('.inputOutStrContainer') === null ? {} : getNode('.inputOutStrContainer').remove() // якщо тег з класом inputOutStrContainer існує то він видаляється з хтмл документу*

*getNode('.programNodesContainer') === null ? {} : getNode('.programNodesContainer').remove()*

*// CREATING LOG CONTAINER*

*const inputOutStrContainer = createNode('div', '', 'inputOutStrContainer') // тег в який буде виводитись інформація про введені дані*

*document.body.appendChild(inputOutStrContainer) //створення тегу в документі*

*sectors = +firstSettings[0] // присвоєння числового значення змінній, "+"*

*huts = +firstSettings[1]*

*// CHECKS*

*if(firstSettings.length < 2) { // якщо кількість аргументів першого рядка(кількість елементів в масиві firstSettings) менше 2 то зпрацює умова*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Кількість аргументів менша двох!', 'errorMessage')) //створення й додавання тегу з повідомленням помилки а контейнер для інформіції про вхідні дані*

*fatalError = true // присвоєння значення*

*}*

*else if(firstSettings.length > 2 && fatalError == false) {*

*inputOutStrContainer.appendChild(createNode('p', 'Кількість аргументів більше двох! Оброблені будут перші 2 аргументи', 'errorMessage'))*

*}*

*if(isNaN(sectors) && fatalError === false) { // перевірка чи sectors не число*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: значення секторів не є числовим', 'errorMessage'))*

*fatalError = true*

*}*

*if(isNaN(huts) && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: значення хижин не є числовим', 'errorMessage'))*

*fatalError = true*

*}*

*if(sectors > 500000 && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Кількість секторів більша 500000!', 'errorMessage'))*

*fatalError = true*

*}*

*else if(sectors < 2 && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Кількість секторів менша 2!', 'errorMessage'))*

*fatalError = true*

*}*

*if(huts > sectors && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Кількість хижин більша за кількість секторів!', 'errorMessage'))*

*fatalError = true*

*}*

*else if(huts < 2 && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Кількість хижин менша 2!', 'errorMessage'))*

*fatalError = true*

*}*

*if((numberSectors === undefined || numberSectors == '') && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', 'ФАТАЛЬНА ПОМИЛКА: Не вказані номера секторів, в яких розміщено хижини!', 'errorMessage'))*

*fatalError = true*

*}*

*if(numberSectors.length != huts && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `ФАТАЛЬНА ПОМИЛКА: Кількість номерів секторів в яких знаходяться хижини, в яких розміщено хижини не зпівпадає з кількістю хижин(${huts} не дорівнює ${numberSectors.length})!`, 'errorMessage'))*

*fatalError = true*

*}*

*for(let i = 0; i < numberSectors.length; i++) {*

*if(isNaN(numberSectors[i]) && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `ФАТАЛЬНА ПОМИЛКА: ${i + 1}-й номер сектора не числового значення!`, 'errorMessage'))*

*fatalError = true*

*}*

*else if(numberSectors[i] > sectors && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `ФАТАЛЬНА ПОМИЛКА: ${i + 1}-й номер сектора більший за кількість секторів(${sectors})!`, 'errorMessage'))*

*fatalError = true*

*}*

*else if(numberSectors[i] < 1 && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `ФАТАЛЬНА ПОМИЛКА: ${i + 1}-й номер сектора менший за 1!`, 'errorMessage'))*

*fatalError = true*

*}*

*else if(+numberSectors[i] <= numberSectors[i - 1] && fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `ФАТАЛЬНА ПОМИЛКА: ${i + 1}-й номер сектора менший або дорівнює попередньому номеру сектора(${numberSectors[i - 1]})!`, 'errorMessage'))*

*fatalError = true*

*}*

*}*

*if(fatalError === false) {*

*inputOutStrContainer.appendChild(createNode('p', `Кількість секторів: ${sectors}, кількість хиж: ${huts}`))*

*inputOutStrContainer.appendChild(createNode('p', `Номера секторів в яких знаходяться хижі: ${numberSectors}`))*

*inputOutStrContainer.appendChild(createNode('button', 'Запустити програму', 'runButton'))*

*getNode('.runButton').onclick = runProgram // присвоєння функції при натисканні ЛКМ на тег кнопки*

*}*

*}*

*\* файл html.js файл з функціями для більш комфортного користування те що виводиться на екран.*

*'use strict'*

*// POPUPS SHOW/HIDE*

*function switchPopUp(type, node) { // відкривання, приховування тегів в документі, type - що зробити, node - тег з яким поводитиметься операція*

*switch (type) { // конструкція свіч*

*case 'show': // якщо type дорівнює show*

*node.style.display = 'flex' // відкривання тегу*

*break*

*case 'hide':*

*node.style.display = 'none' // приховування*

*break*

*}*

*}*

*\* файл program.js файлі зберігаються функції які оброблюють вхідні дані й вираховують час пересування між секторами.*

*'use strict'*

*function sortingStatistic(arr = statisticArr, type = '>', arrArg = 0) { // сортування матриці, arr - змінна матриці, type - тип сортування, arrArg - номер колонки за якою сортується матриця*

*arr.sort((a, b) => { // функція критеріїв сортування*

*if(type === '>') return b[arrArg] - a[arrArg]*

*if(type === '<') return a[arrArg] - b[arrArg]*

*})*

*}*

*let statisticArr = [] // змінна матриці в якій будуть всі оброблені дані*

*function runProgram() { // обробка даних й вивод їх в матрицю*

*statisticArr = [] // робить матрицю пустою*

*getNode('.programNodesContainer') === null ? document.body.appendChild(createNode('div', '', 'programNodesContainer')) : getNode('.programNodesContainer').remove()*

*let itr = 1 // номер відстані між секторами*

*for(let i = 0; i < numberSectors.length; i++) {*

*for(let j = i + 1; j < numberSectors.length; j++) {*

*statisticArr[itr - 1] = [itr++, numberSectors[i], numberSectors[j], numberSectors[j] - numberSectors[i], sectors - numberSectors[j] + +numberSectors[i]] // присвоєння масиву значень матриці значень*

*}*

*}*

*runCanvas() // визов функції*

*runStatistic()*

*}*

*\* файл canvas.js створює полотно й виводить в нього графічно введені дані.*

*'use strict'*

*function degToRad(deg) { // перетворення градусів - 90 градусів в радіани, deg - градуси*

*return (deg - 90) \* Math.PI / 180*

*}*

*function lineToAngle(argX, argY, angle, distance) { // повертає перміщенні координати на деяку відстань на деякий кут*

*argX = Math.cos(degToRad(angle)) \* distance + argX*

*argY = Math.sin(degToRad(angle)) \* distance + argY*

*return [argX, argY]*

*}*

*function runCanvas() { // створення полотна з графікою*

*// CREATING CANVAS AND CONFIGURING IT*

*getNode('canvas') === null ? {} : getNode('canvas').remove()*

*getNode('.programNodesContainer').prepend(createNode('canvas'))*

*const cnvs = getNode('canvas'), // полотно*

*cnvsWdth = window.innerWidth \* 0.4, // задання зміній значення ширини сторінки \* 0.4*

*cnvsSizeScale = 1,*

*cnvsScaledWdth = cnvsWdth \* cnvsSizeScale,*

*ctx = cnvs.getContext('2d')*

*// присвоєння значень полотну*

*cnvs.width = cnvsScaledWdth*

*cnvs.height = cnvsScaledWdth*

*cnvs.style.width = cnvsWdth + 'px'*

*cnvs.style.height = cnvsWdth + 'px'*

*cnvs.style.border = '1px solid #000'*

*// DRAWING ON CANVAS*

*ctx.fillStyle = '#2590ff'*

*ctx.fillRect(0, 0, cnvsScaledWdth, cnvsScaledWdth) // малювання прямокутника на все полотно*

*// CREATING DUBLICATE SECTORS AND PROTECT CANVAS FROM SECTORS OVER 2000*

*let sectorsDub = sectors*

*if(sectors >= 2000) sectorsDub = 2000*

*const x = cnvsScaledWdth / 2,*

*y = cnvsScaledWdth / 2,*

*degStep = 360 / sectorsDub*

*let currentDeg = 0*

*// DRAWING SECTORS*

*ctx.moveTo(x, y) // переміститися в певні координати*

*for(let i = 0; i < sectorsDub; i++) {*

*ctx.arc(x, y, cnvsScaledWdth / 2.25, degToRad(currentDeg), degToRad(currentDeg + degStep)) // малювати коло*

*ctx.lineTo(x,y) // малювати лінію*

*currentDeg += degStep*

*}*

*ctx.fillStyle = '#19bd3d' // колір зафарбовування*

*ctx.fill() // зафарбувати фігуру*

*ctx.stroke() // зафарбувати контур*

*// DRAWING NUMBERS SECTORS*

*let fontSize = cnvsScaledWdth / 15*

*ctx.font = `${fontSize}px arial`*

*ctx.fillStyle = '#000000'*

*currentDeg = 0*

*for(let i = 0; i < sectorsDub; i++) {*

*const tXY = lineToAngle(x, y, currentDeg + degStep / 2, cnvsScaledWdth / 2.75)*

*ctx.fillText(i + 1, tXY[0] - fontSize / 2, tXY[1] + fontSize / 2)*

*currentDeg += degStep*

*}*

*// DRAWING HUTS*

*ctx.beginPath()*

*ctx.fillStyle = '#bda81b'*

*currentDeg = 0*

*let numSectorsI = 0*

*for(let i = 0; i < sectorsDub; i++) {*

*const tXY = lineToAngle(x, y, currentDeg + degStep / 2, cnvsScaledWdth / 4)*

*if(i + 1 === +numberSectors[numSectorsI]) {*

*ctx.arc(tXY[0], tXY[1], cnvsScaledWdth / 30, 0, Math.PI \* 2)*

*ctx.fill()*

*ctx.stroke()*

*ctx.beginPath() // створення нового шляху(нової фігури)*

*numSectorsI++*

*}*

*currentDeg += degStep*

*}*

*}*

*function highlightSector(sectorNum) { // підсвічення секторів*

*const cnvs = getNode('canvas'),*

*cnvsWdth = window.innerWidth \* 0.4,*

*cnvsSizeScale = 1,*

*cnvsScaledWdth = cnvsWdth \* cnvsSizeScale,*

*ctx = cnvs.getContext('2d')*

*// CREATING DUBLICATE SECTORS AND PROTECT CANVAS FROM SECTORS OVER 2000*

*let sectorsDub = sectors*

*if(sectors >= 2000) sectorsDub = 2000*

*const x = cnvsScaledWdth / 2,*

*y = cnvsScaledWdth / 2,*

*degStep = 360 / sectorsDub*

*let currentDeg = (sectorNum - 1) \* degStep*

*ctx.moveTo(x, y)*

*ctx.arc(x, y, cnvsScaledWdth / 2.25, degToRad(currentDeg), degToRad(currentDeg + degStep))*

*ctx.lineTo(x,y)*

*// ctx.fillStyle = '#12962f'*

*ctx.lineWidth = 5*

*// ctx.fill()*

*ctx.stroke()*

*}*

*\* файл statistic.js створює таблицю вихідних даних й взаємодії з таблицею.*

*'use strict'*

*// STATISTIC PROGRAM*

*function runStatistic() { // створення таблиці й додавання її в хтмл документ*

*getNode('.statisticTable') !== null ? getNode('.statisticTable').remove() : {}*

*const table = createNode('table', `*

*<caption>Керування<div class="button" onclick="switchPopUp('show', getNode('#statisticControls'))">?</div></caption>*

*<tr onclick="sortFunc(event)">*

*<th data-statistic-Num="0">№</th>*

*<th data-statistic-Num="1">Початковий сектор</th>*

*<th data-statistic-Num="2">Кінцевий сектор</th>*

*<th data-statistic-Num="3">Час за годинниковою стрілкою (хв.)<div class="button" onclick="switchPopUp('show', getNode('#clockwiseFuncInfo'))">?</div></th>*

*<th data-statistic-Num="4">Час проти годинникової стрілки (хв.)<div class="button" onclick="switchPopUp('show', getNode('#anticlockwiseFuncInfo'))">?</div></th>*

*</tr>*

*`, 'statisticTable')*

*getNode('.programNodesContainer').appendChild(table)*

*for(let i = 0; i < statisticArr.length; i++) {*

*table.appendChild(createNode('tr', `<td>${statisticArr[i][0]}</td><td>${statisticArr[i][1]}</td><td>${statisticArr[i][2]}</td><td>${statisticArr[i][3]}</td><td>${statisticArr[i][4]}</td>`, '', ['onmouseover', 'highlightFunc(this)'], ['onmouseout', 'outHighlightFunc(this)']))*

*}*

*}*

*// SORTING*

*function sortFunc(e) { // функція сортування даних таблиці, e - об'єкт натискання ЛКМ*

*let error = false*

*for(let i = 0; i < getNodes('th .button').length; i++) {*

*if(e.target === getNodes('th .button')[i]) error = true*

*}*

*if(error == false) {*

*let statNum = e.target.dataset.statisticNum*

*if(e.shiftKey) {*

*sortingStatistic(statisticArr, '<', statNum)*

*}*

*else {*

*sortingStatistic(statisticArr, '>', statNum)*

*}*

*}*

*runStatistic()*

*}*

*// CANVAS SECTORS AND TABLE ROW HIGHLIGHTS*

*function highlightFunc(node) { // виділення рядка таблиці й секторів*

*node.style.backgroundColor = '#f2f2f2'*

*highlightSector(+node.childNodes[1].innerText)*

*highlightSector(+node.childNodes[2].innerText)*

*console.log(2)*

*}*

*function outHighlightFunc(node) { // зняття виділення рядка таблиці й секторів*

*node.style.backgroundColor = '#ffffff'*

*runCanvas()*

*}*