МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему
“ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА РЕАЛІЗАЦІЯ ВЕБ ВЕРСІЇ КОНСТРУКТОРА АЛГОРИТМІВ: МОДУЛЬ УЧНЯ”

Виконав: студент 2 курсу, групи МФ-62
спеціальність 122 «Комп’ютерні науки»
освітньо-професійна програма
«Інформатика»

Сергієнко І. А.

_____

Керівник к. ф. м. н. , доцент Зарецька І. Т.

Рецензент вчитель вищої категорії,
вчитель-методист,
заслужений вчитель України, голова
відділу інформатики
видавництва "Ранок"
Корнієнко М. М.

Харків – 2024 року

ЗМІСТ

ВСТУП

У сучасному світі, який стрімко розвивається завдяки технологіям, вміння програмувати та мислити алгоритмічно стає надзвичайно важливим для дітей. Конструктор алгоритмів виступає потужним інструментом, що сприяє розвитку логічного мислення, креативності та аналітичних навичок у молодих умів. Діти, ознайомлюючись із цим конструктором, не лише навчаються створювати алгоритми, а й вчаться представляти їх у графічній формі через блок-схеми.

Додаток розвиває їхній інтерес до програмування та логічного мислення. Його головна мета — поєднати навчання з ігровими елементами, щоб зробити процес засвоєння матеріалу цікавим та ефективним. У додатку передбачено простий і зручний інтерфейс, де діти можуть легко перетягувати блоки та з'єднувати їх, формуючи власні схеми. Кожен блок містить базові команди або питання, що сприяють вивченню основ алгоритмів і структур даних.

Блок-схеми є візуальним зображенням алгоритму, яке дозволяє легко та зрозуміло проілюструвати послідовність дій та логіку виконання завдань. Вони допомагають учням усвідомлювати структуру алгоритмів і візуалізувати взаємозв'язки між різними етапами процесу. Конструктор алгоритмів надає дітям можливість самостійно розробляти свої блок-схеми, експериментувати з ідеями та втілювати їх у життя.

Унікальна особливість цього застосунку - можливість налагодження створених схем, що дозволяє дітям бачити, як працюють їх алгоритми в реальному часі. Це не тільки забавляє, але й сприяє кращому розумінню логіки процесів. До того ж він пропонує різноманітність блоків для створення схем, від простих математичних операцій до складніших логічних умов, що дозволяє дітям експериментувати та вчитися, створюючи все складніші структури.

Основи побудови блок-схем були стандартизовані Американським національним інститутом стандартів (ANSI) у 1960-х роках, що забезпечило єдині правила їх створення. Такі елементи, як прямокутники для позначення операцій, ромби для рішень і стрілки для з'єднань, стали загальновизнаними та залишаються актуальними до сьогодні. Конструктор алгоритмів також дотримується цих стандартів.

Окрім навчання, блок-схеми знайшли своє застосування в багатьох інших сферах. Наприклад, у медицині вони використовуються для опису клінічних протоколів, в інженерії - для моделювання систем управління, а в бізнесі - для розробки процесів автоматизації. Особливий внесок вони зробили в розвиток програмування, ставши основою для проєктування алгоритмів у перших мовах програмування.

В результаті співпраці Кафедри теоретичної та прикладної інформатики з видавництвом «Ранок» було створено вебдодаток для побудови алгоритмів у вигляді блок-схем на платформі IZZI.

IZZI - це сучасна освітня онлайн-платформа, яку створили розробники з Хорватії. Вона пропонує вчителям інструменти для створення інтерактивних уроків, що можуть використовуватися як онлайн, так і офлайн. В Україні IZZI активно підтримується видавництвом "Ранок", яке забезпечує навчальні заклади підручниками та посібниками.

Щоб зробити навчання ще ефективнішим, видавництво доповнює свої навчальні матеріали інтерактивними ресурсами на базі IZZI. Ці ресурси складаються з готових шаблонів та інструментів, які дозволяють учням розв'язувати різноманітні логічні завдання у форматі вебдодатків. Одним з таких ресурсів є конструктор алгоритмів.

Метою роботи є розробка, проєктування та реалізація вебдодатка, який надає можливість створювати власні алгоритми, налагоджувати їх та задовільняє всім вимогам видавництва "Ранок" та платформи IZZI.

Актуальність теми: зумовлена необхідністю розширення інтерактивного контенту для платформи IZZI та видавництва «Ранок», що сприятиме покращенню навчального процесу та задоволенню зростаючих потреб користувачів.

Головні терміни:

Блок-схема — це графічне зображення, яке ілюструє послідовність дій, етапів або рішень у межах конкретного алгоритму, процесу чи системи, показуючи логіку їх виконання.

Завдання, що були поставлені до роботи:

1. Ознайомитися з технічними вимогами платформи та стандартами видавництва для розробки вебдодатка.
2. Описати та структурувати відомості щодо побудови блок-схем та їх основних компонентів.
3. Спроєктувати структуру, дизайн та архітектуру вебдодатка відповідно до заданих вимог.
4. Реалізувати вебдодаток на основі стандартів щодо побудови блок-схем.

В результаті виконання дипломної роботи:

1. Було проведено дослідження вимог і принципів побудови блок-схем.
2. Визначено та враховано вимоги видавництва та платформи.
3. Створено інтерактивний вебдодаток конструктор алгоритмів.

ОСНОВНА ЧАСТИНА

РОЗДІЛ 1. ПОРІВНЯННЯ З ІНШИМИ КОНСТРУКТОРАМИ АЛГОРИТМІВ

На просторах інтернету є чимало конструкторів алгоритмів, але більшість з них є лише саме конструкторами, які не мають функціоналу для виконання та налагодження створеного алгоритму.

Розглянемо перший додаток – draw.io

Додаток містить широкий вибір шаблонів уже готових діаграм та блок-схем, також є можливість побудувати власні схеми.

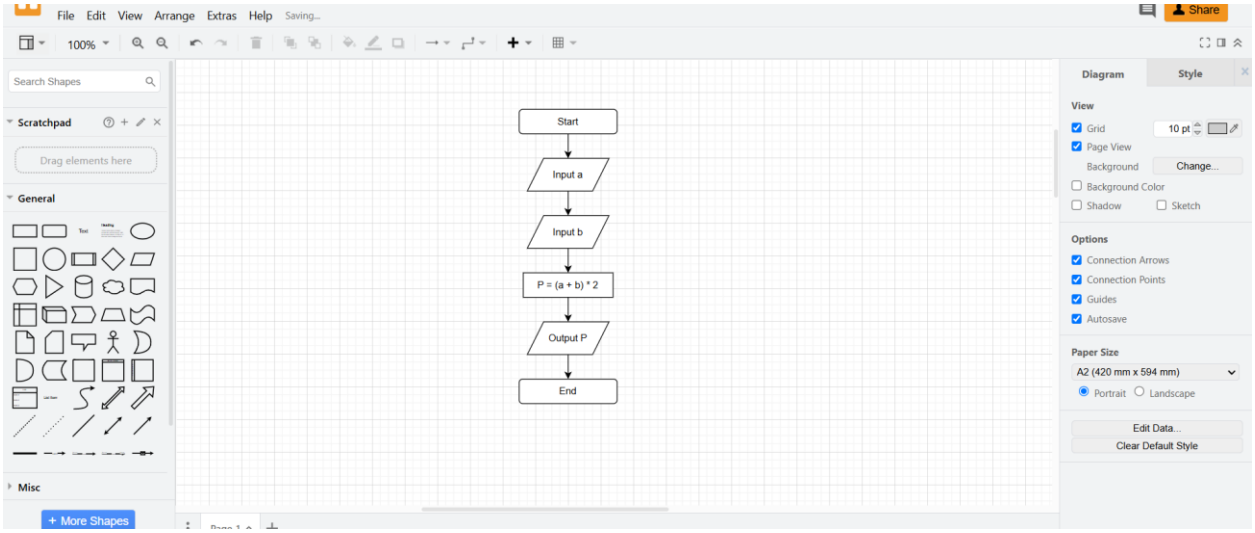

Рисунок 1.1 Додаток draw.io

Але немає функціоналу для запуску створеного алгоритму, тим паче покрокового проходження по блок-схемі.

Ще один онлайн додаток Diagram.Codes, який також має велику підбірку готових блок-схем та діаграм, які можна змінювати під свої потреби.

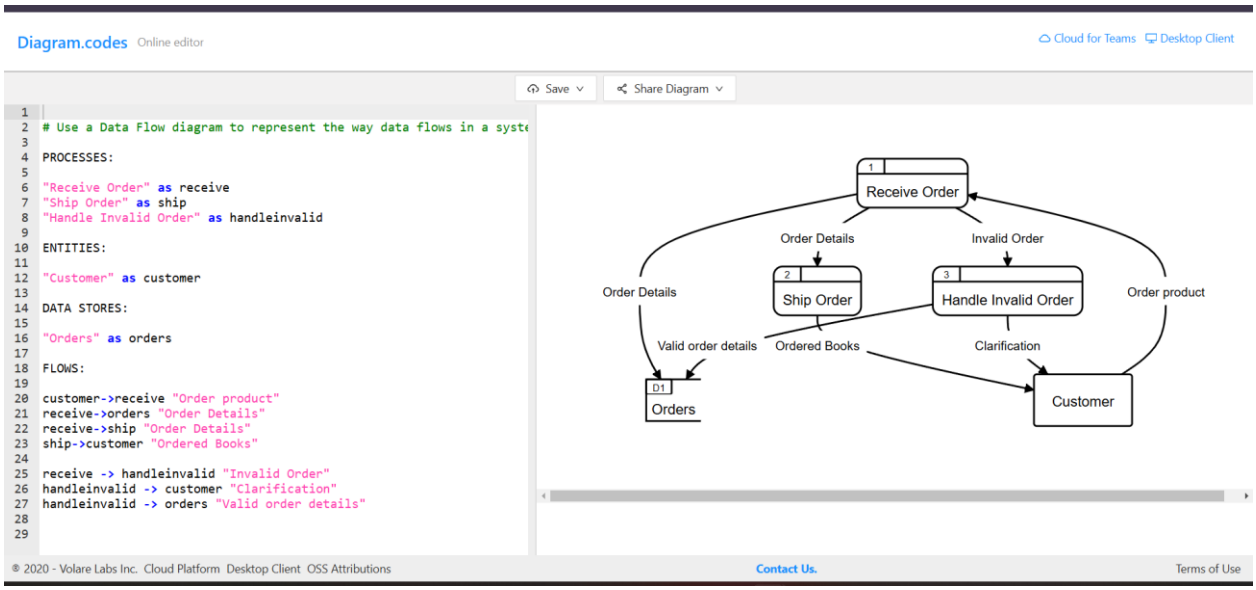

Рисунок 1.2 Додаток Diagram.Codes

Diagram.Codes не містить функціоналу для створення власноруч саме блок-схеми алгоритму, оскільки всі елементи задаються через код. Також відсутня можливість виконання введеного алгоритму, як і в попередньому додатку. Отже, цей додаток краще підходить для тих, хто вже має розвинене логічне мислення та вміє програмувати.

І розглянемо останній приклад — додаток Flowgorithm.

Для користування необхідно встановити додаток на комп’ютер з сайту *https://flowgorithm.org/*. Однак встановити його можна тільки на операційну систему Windows, інші операційні системи та веб версія не підтримуються.

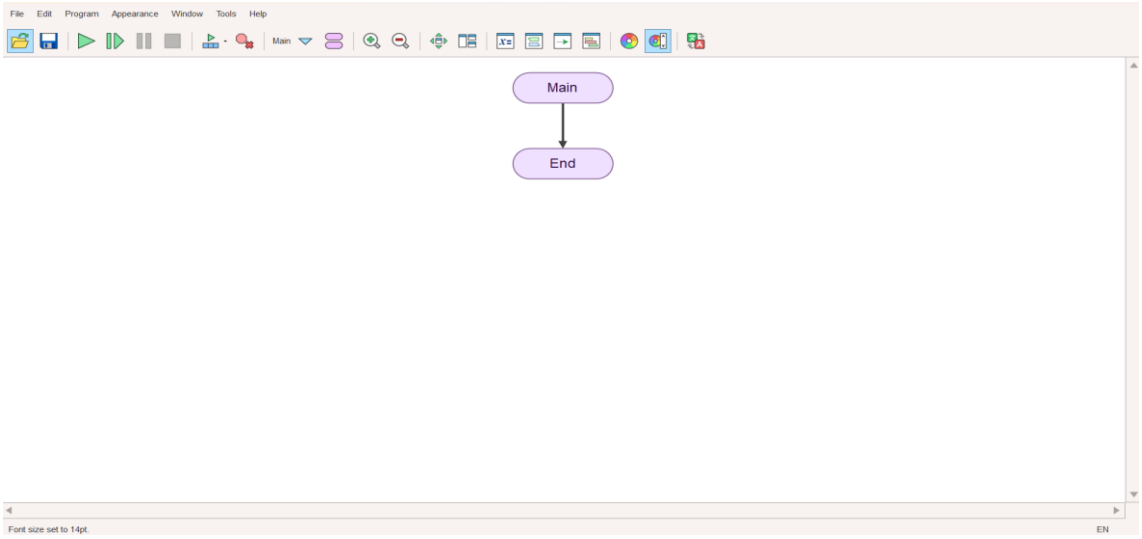

Рисунок 1.3 Початковий екран додатка з двома блоками, входу та виходу з алгоритму

З першого погляду Flowgorithm не є інтуїтивно зрозумілим, хоч і має багато налаштувань на верхній панелі.

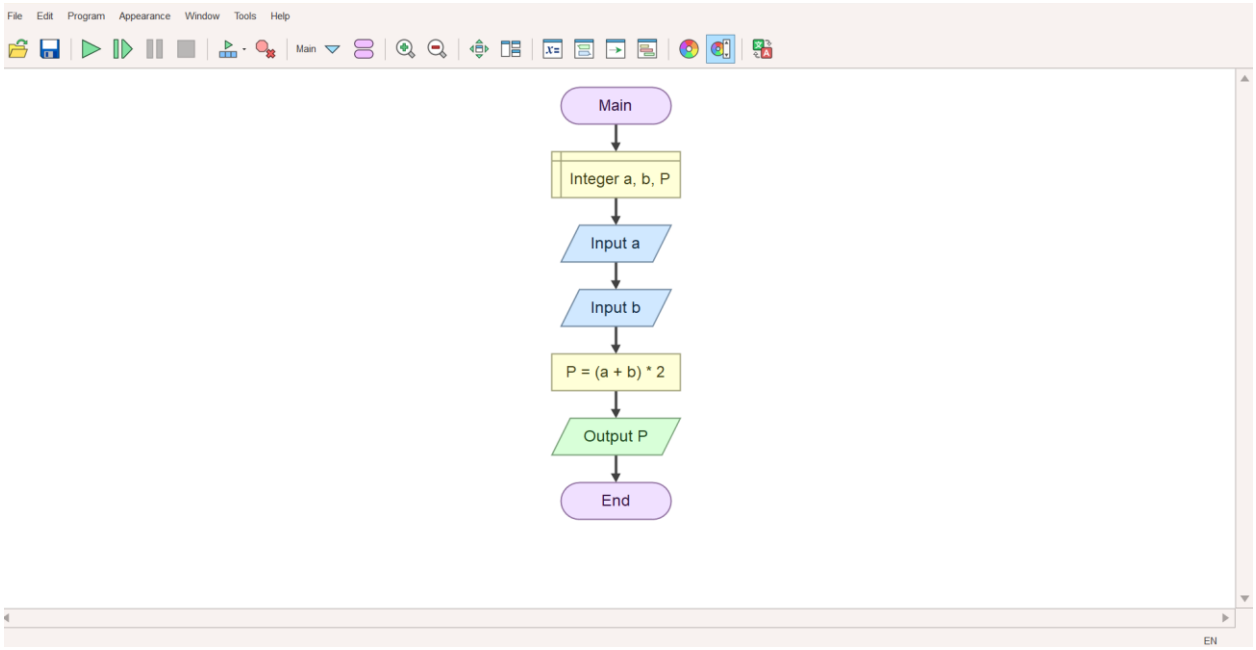

Рисунок 1.4 Додаток Flowgorithm

Додаток має багатий функціонал — відображення введеного алгоритму в коді, можливість визначати тип даних для змінних, створювати функції і т.д.

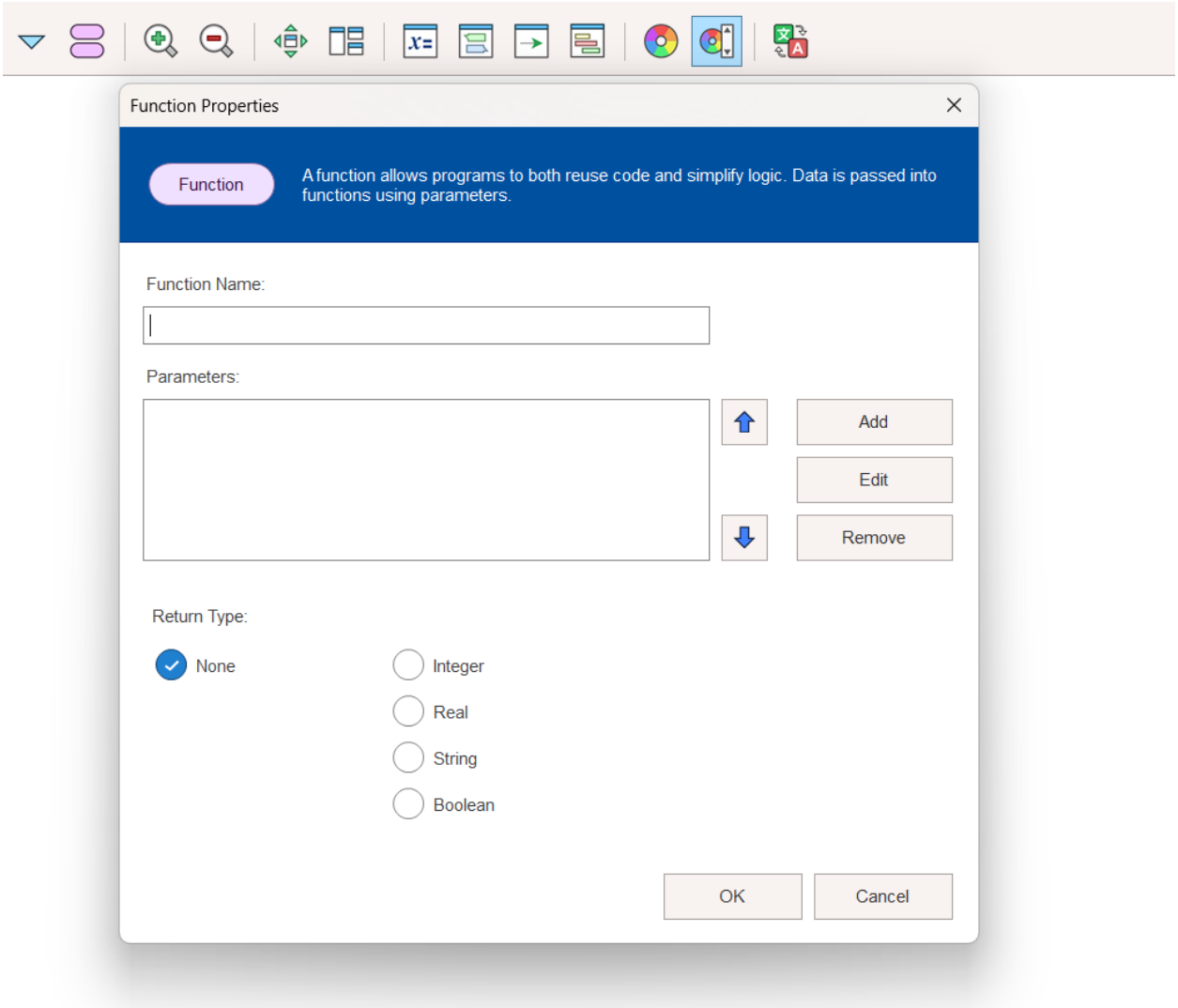

Рисунок 1.5 Можливість додавання функції у додатку Flowgorithm

Але цей додаток створений для дорослої аудиторії, з відповідним інтерфейсом та функціоналом.

РОЗДІЛ 2. ТЕОРЕТИЧНІ АСПЕКТИ БЛОК-СХЕМ

2.1 Основні визначення

Алгоритм – це набір кроків, що породжує скінченну послідовність елементарних обчислювальних операцій, що ведуть до розв'язання задачі. Текстовий опис алгоритму може бути занадто багатослівним та не зовсім зрозумілим. Тому замість алгоритму часто використовується його графічне зображення. Таке графічне зображення називається блок-схемою.
Блок-схема - це схематичне зображення кроків алгоритму. На блок-схемі використовуються елементи різної форми для позначення різних типів операцій, які називаються блоками блок-схеми. Ці блоки потім з'єднуються лініями зі стрілками, що позначають напрямок в якому слід рухатися, щоб дістатися наступного кроку.

Блок-схеми є двох типів:

1. Блок-схеми для зображення програм діють як дзеркальне відображення програмного забезпечення з погляду символів на блок-схемах. Вони містять кроки розв'язання проблемної одиниці для отримання конкретного результату.
2. Системні блок-схеми містять рішення багатьох проблемних одиниць разом які тісно пов'язані між собою і взаємодіють один з одним для досягнення мети.

2.2 Блок-схеми для зображення програм

Блок-схема для зображення програм є ефективним засобом візуалізації та аналізу алгоритмів програмного забезпечення. Вона дозволяє значно спростити процес виявлення помилок або пропущених елементів, оскільки надає графічне уявлення про структуру і логіку програми, що полегшує розуміння її роботи. Крім того, блок-схема забезпечує зручність у відстеженні послідовності виконання коду, що сприяє швидшому аналізу та

налагодженню. Також вона виконує функцію технічної документації, що є цінним для подальшої модифікації або підтримки програмного продукту.

При створенні блок-схем для зображення програм слід дотримуватися наступних п'яти правил:

1. У блок-схемах слід використовувати тільки стандартні символи.
2. Логіка програми повинна зображати потік зверху вниз або, рідше, зліва направо.
3. Кожен символ, що використовується в блок-схемі програми, повинен містити тільки одну точку входу і одну точку виходу, за винятком символу умови. Це правило відоме як *правило одного*.
4. Операції, показані в межах символу блок-схеми програми, повинні бути виражені незалежно від конкретної мови програмування.
5. Всі лінії зі стрілками повинні бути добре позначені.

Нижче наведено стандартні символи, що використовуються в блок-схемах для зображення програм:

1. Термінальні символи - використовуються для відображення початку та кінця блок-схеми.

Рисунок 2.1 Термінальний символ

2. Символи вводу/виводу - використовуються для відображення будь-якої операції вводу/виводу.

Рисунок 2.2 Символ вводу або виводу

3. Символ процесу - використовується для відображення декількох операцій або обробки даних.

Рисунок 2.3 Символ процесу

4. Символ процедури - використовується для позначення будь-якого процесу, спеціально не визначеного на блок-схемі. Тобто, в частині програми, що не візуалізуєтеся на даній блок схемі, або знаходиться в іншій підпрограмі (модулі).

Рисунок 2.4 Символ процедури

5. Символ коментаря - використовується для написання будь-якого пояснювального тексту.

Рисунок 2.5 Символ коментаря

6. Символи ліній зі стрілками – використовуються для з’єднання блоків блок-схеми.

Рисунок 2.6 Символ лінії зі стрілкою

7. Символи вводу/виводу з/до документа – використовується якщо вхідні дані надходять з документа, а вихідні до документа.

Рисунок 2.7 Символ вводу з файлу або виводу до файлу

8. Символ умови - використовується для позначення будь-якої точки процесу, в якій подальше виконання залежить від певних умов, прописаних в блоці.

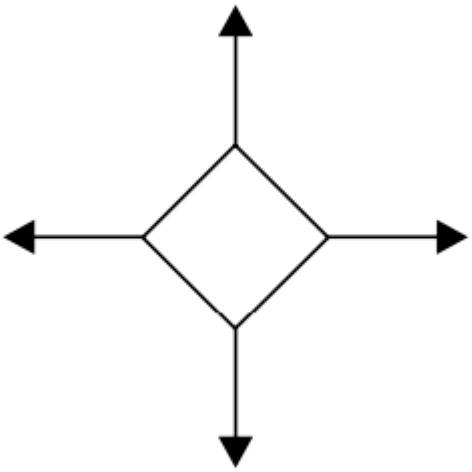

Рисунок 2.8 Символ умови

9. Символ внутрішньосторінкового з'єднання - використовується для з'єднання частин блок-схеми, що знаходяться на одній сторінці

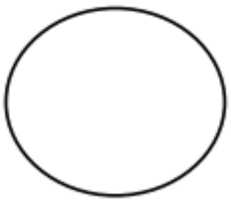

Рисунок 2.9 Символ внутрішньосторінкового з'єднання

10. Символ поза сторінкового з'єднання - використовується для з'єднання частин блок-схеми, які знаходяться на різних сторінках.

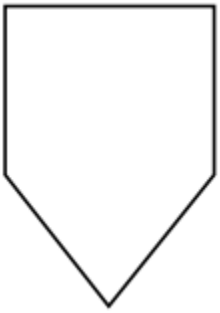

Рисунок 2.10 Символ поза сторінкового з'єднання

РОЗДІЛ 3. РЕАЛІЗАЦІЯ КОНСТРУКТОРА АЛГОРИТМІВ

3.1 Визначення функціональних та нефункціональних вимог до додатка

Перед тим, як розпочинати розробку архітектури та структури конструктора алгоритмів, необхідно встановити функціональні та нефункціональні вимоги до додатка.

Функціональні вимоги:

1. конструктор алгоритмів повинен мати функцію, яка надає можливість перетягувати та з'єднувати блоки блок-схеми;
2. конструктор алгоритмів повинен мати функцію вводу даних у відповідні блоки;
3. конструктор алгоритмів повинен мати функцію видалення блоків з блок-схеми;
4. конструктор алгоритмів повинен мати функцію, що запускає виконання з’єднаних блоків;
5. конструктор алгоритмів повинен мати режим налагодження;
6. конструктор алгоритмів повинен мати функцію видалення всіх блоків на блок-схемі;
7. конструктор алгоритмів повинен мати функцію перемикання між повноекранним режимом та стандартним виглядом;
8. конструктор алгоритмів має бути розроблений відповідно до критеріїв, встановленими видавництвом, а саме:
    1. розміри робочої області додатка мають становити 1050*850 пікселів;
    2. використовувати шрифт IBM Plex Sans для текстових елементів;
    3. інтегрувати у дизайн іконки з набору Google Material Icons;
    4. основним кольором дизайну, окрім стандартного чорного та білого, повинен бути волошково-синій (з HEX-кодом #4A90E2) або відтінки, близькі до нього;

5. для виділення тексту використовувати колір пряно-фіолетовий (з HEX-кодом #C30E60);
6. дизайн кнопок та перемикачів (радіокнопок) повинен відповідати стилю, який застосовує видавництво;

Нефункціональні вимоги:

1. Застосунок повинен бути адаптований українською мовою.
2. Інтерфейс додатка має бути доступним через веббраузер.
3. Застосунок не повинен залежати від пристрою використання.

Виходячи з функціональних та нефункціональних вимог до конструктора можемо сформулювати задачу наступним чином: необхідно розробити вебдодаток, який надає можливість будувати блок-схеми з різних типів блоків, можливість видалення їх з блок-схеми, видалення (очищення) всіх елементів блок-схеми та режим налагодження в якому можна покроково “ходити” по блок-схемі. Сам додаток повинен бути сумісним з платформою IZZI та відповідати стандартам видавництва “Ранок”.

3.2 Діаграма використання та діаграма діяльності

Відповідно до функціональних та нефункціональних вимог була розроблена діаграма використання та діаграма діяльності конструктора алгоритмів.

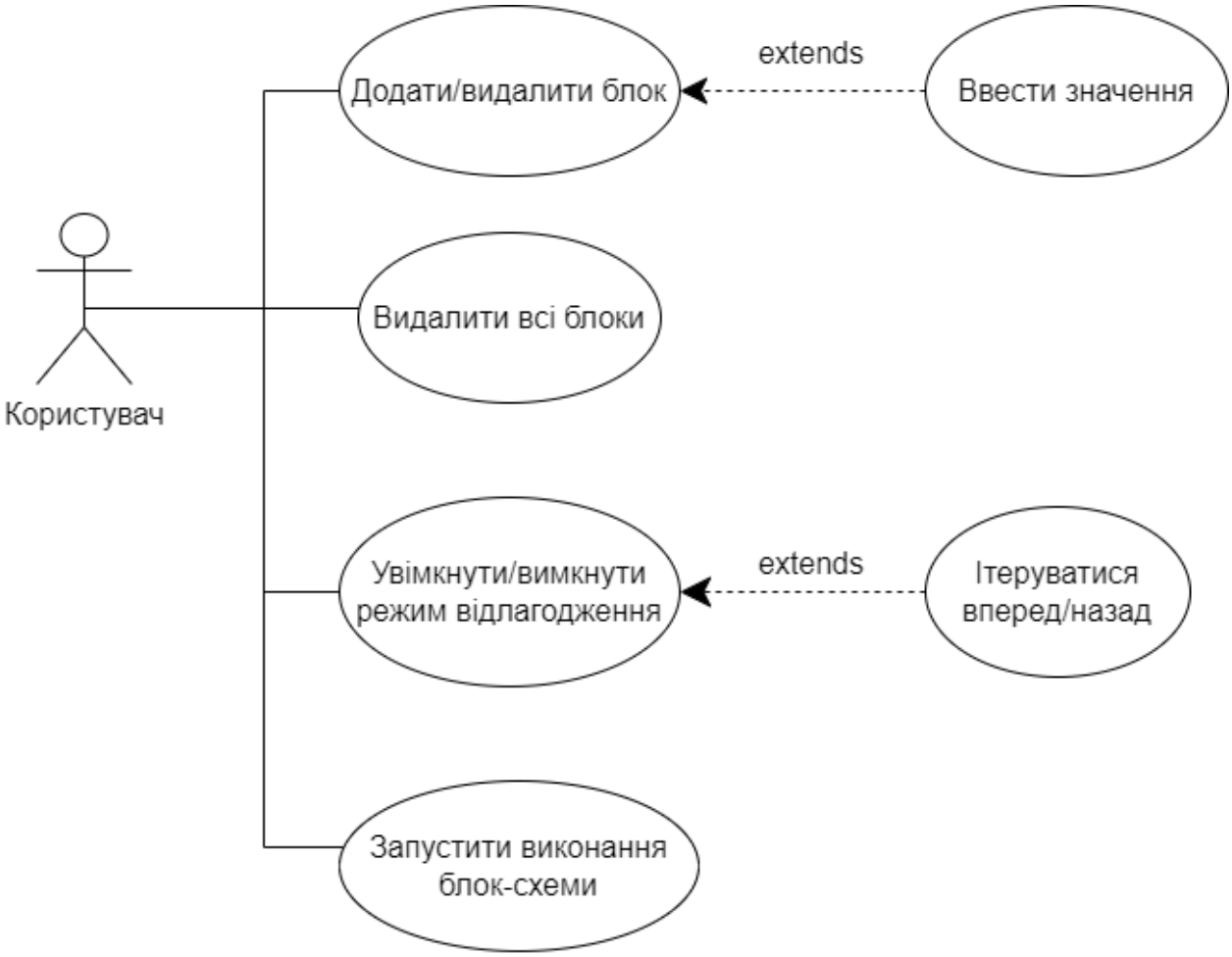

Рисунок 3.1 Діаграма використання

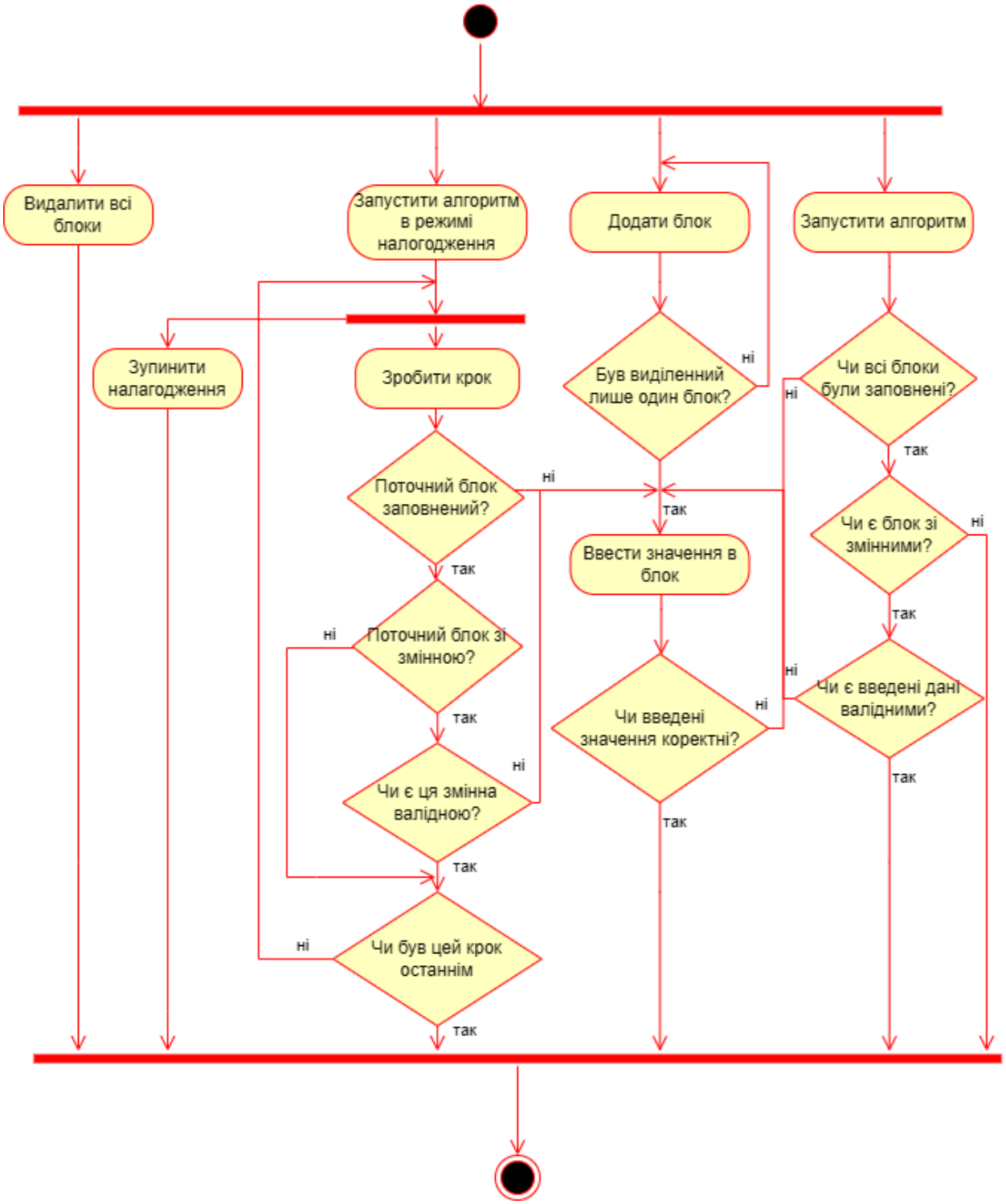

Рисунок 3.2 Діаграма діяльності

3.3 Архітектура конструктора алгоритмів та IZZI Server

При розробці вебдодатків одним з ключових моментів є вибір архітектури системи та визначення технологій, які будуть

використовуватися. Основна увага приділяється визначенню типу архітектури на основі вимог IZZI Server та видавництва “Ранок”. Враховуючи вимогу, що система повинна надавати доступ через веббраузер, вибір зупинився на розробці вебдодатка. Цей тип архітектури має ряд переваг:

1. незалежність від пристрою, що використовується для взаємодії з системою;
2. не потребує додаткової конфігурації на клієнтському пристрої;
3. доступ до додатка здійснюється через інтернет посилання, що забезпечує масштабованість і високу продуктивність системи.

Стандартна трирівнева модель архітектури вебдодатків охоплює рівень даних, рівень бізнес-логіки та рівень інтерфейсу і вважається найбільш поширеною на сьогодні. Однак, беручи до уваги той факт, що нема потреби зберігати дані протягом тривалих проміжків часу і для зменшення навантаження на системи і сервери, було прийнято рішення об'єднати рівень даних і рівень бізнес-логіки для створення дворівневої клієнт-серверної архітектури.

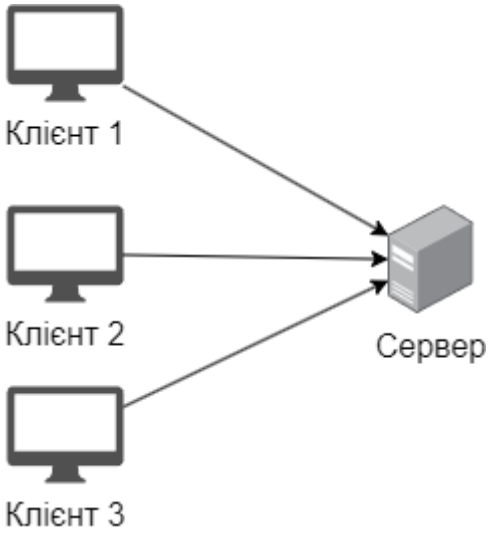

Рисунок 3.3 Дворівнева клієнт-серверна архітектура

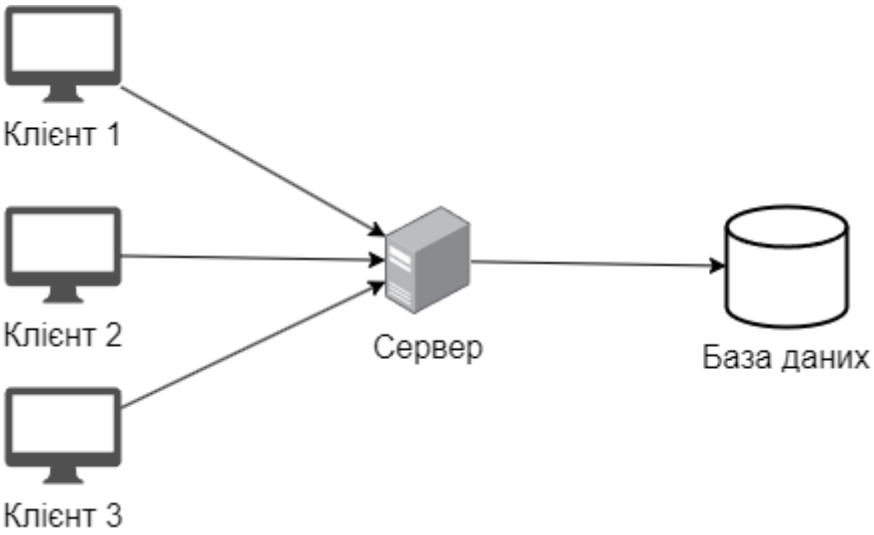

Рисунок 3.4 Трирівнева клієнт-серверна архітектура

IZZI Server, розташований у Хорватії. Цей сервер відповідає за надсилання даних на клієнтську сторону у вигляді JSON-файлів і використовує REST підхід.

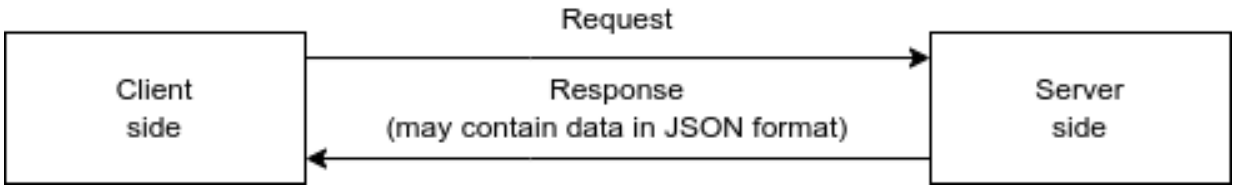

Рисунок 3.5 Взаємодія між клієнтською та серверною частинами

Такий підхід забезпечує наступні переваги та правила:

1. використовуються стандартні методи HTTP;
2. єдина угода про імена URL полегшує взаємодію між клієнтською та серверною частинами;
3. додаткові сервери або сервіси можуть бути додані без будь-яких змін на стороні клієнта;
4. можливість обмінюватися різними типами файлів між клієнтом і сервером;
5. кешування запитів;

6. використання стандартів безпеки та незалежність від мов програмування;

Проксі-сервер IZZI в Україні виступає в ролі сховища для локальних користувачів. Це дозволяє додатково підвищити рівень безпеки і відповідності регіональним вимогам. Важливо, що IZZI Server є гнучким, безпечним і масштабованим сервером, якій відповідає потребам сучасних вебдодатків.

До обов'язків на стороні клієнта входить створення інтерфейсу взаємодії, обробка отриманих від користувача даних та зміна відповідних інтерфейсів, які з'являються під час взаємодії з системою. Всі необхідні дані завантажуються на клієнтську сторону під час ініціалізації системи. Це виключає необхідність періодичного оновлення сторінок, оскільки всі важливі дані вже завантажені, а клієнт може самостійно здійснювати обмін інформацією.

Таким чином, переваги дворівневої архітектури в поєднанні з можливостями IZZI Server дають продуктивний, масштабований і гнучкий продукт, який відповідає сучасним технічним вимогам і бізнес-задачам. В результаті додаток не тільки відповідає поточним вимогам ринку, але й готовий до майбутніх розширень та оновлень.

3.4. Клієнтська частина конструктора алгоритмів

Клієнтська частина платформи IZZI – інтерактивний урок на певну навчальну тему або певного типу. Як описано в попередньому розділі, ця частина надсилає запит на сервер IZZI, який обробляє запит і відповідає, надсилаючи дані у форматі JSON, використовуючи архітектуру REST. Ці дані відображаються в зрозумілому і привабливому форматі на стороні клієнта, створюючи вебінтерфейс. Цей вебінтерфейс побудований за принципом SPA (Single Page Application), який використовує лише одну вебсторінку і

динамічно оновлює елементи без перезавантаження сторінки, тобто без додаткових запитів на сервер зі сторони клієнта.

Основна перевага SPA-архітектури полягає в тому, що всі зміни вносяться безпосередньо в браузері користувача після завантаження всього інтерфейсу. Це не тільки зменшує навантаження на сервер, але й покращує користувацький досвід, оскільки інтерфейс оновлюється плавно і швидко. Інтерактивність інтерфейсу створює враження постійної взаємодії, миттєво реагуючи на дії користувача. Такий підхід також підтримує легку масштабованість платформи, оскільки нові функції та оновлення можуть бути додані без зміни загальної структури сторінки.

Як результат, використання SPA-архітектури на платформі IZZI дозволяє створювати швидкі, ефективні та гнучкіші вебінтерфейси. Такий інтерфейс забезпечує глибшу інтеграцію з навчальним контентом, роблячи навчання цікавішим та інтерактивним. Використання JSON для передачі даних забезпечує швидкий і безперебійний обмін інформацією між сервером і клієнтом, що важливо для підтримки високої продуктивності і надійності системи. Як результат, платформа IZZI не тільки пропонує високу продуктивність і швидкість реагування, але також може бути легко розширена і модернізована, що робить її ідеальною для сучасних освітніх середовищ.

3.5 Реалізація

Основним завданням додатка є створення алгоритмів у вигляді блок-схем, що дозволяє користувачам візуально будувати логічні послідовності дій. Додаток підтримує широкий спектр функціональних блоків для створення алгоритмів різної складності, а також забезпечує можливість налагодження алгоритмів у покроковому режимі.
Додаток має декілька типів блоків, які користувач може додавати до схеми за допомогою простого перетягування:

1. Текстовий блок — використовується для опису дії.
2. Блок присвоєння значень — дозволяє створювати змінні та задавати їм певні значення.
3. Блок введення значень — запитує у користувача певні дані під час виконання алгоритму.
4. Блок виведення значень — відображає результати роботи алгоритму або проміжні значення змінних.
5. Умовні блоки — використовуються для створення умовних переходів. Додаток підтримує прості умови (наприклад, “A = B”) та складніші логічні вирази.

Додаток дозволяє користувачу:

- Будувати блок-схеми алгоритмів шляхом перетягування та з'єднання блоків. Блоки можна з’єднувати між собою для створення послідовностей дій та умовних переходів.
- Редагувати блоки. Користувач може змінювати текст та умови в блоках у будь-який момент.
- Налагоджувати алгоритм. Після створення алгоритму користувач може запустити його в режимі налагодження, який дозволяє виконувати алгоритм покроково та відстежувати змінні й логіку виконання.
- Зупиняти та перезапускати алгоритм для внесення змін та повторної перевірки.

Для зручності використання та відповідності вимогам платформи, додаток має такі обмеження:

- Максимальна кількість блоків обмежена 50 елементами, що забезпечує оптимальну продуктивність і зручність навігації в алгоритмах.
- Відсутність адаптивного розгортання у повноекранному режимі, що узгоджено з вимогами до інших додатків платформи.
- Скидання алгоритму. Користувач може скинути поточний алгоритм до початкового стану.

Основою архітектури додатка є використання класів і модулів в мові JavaScript, які відповідають за зберігання, обробку та візуалізацію алгоритмів. Базовим елементом для представлення будь-якого функціонального блоку є клас Block, що забезпечує структуру для зберігання ключових властивостей, таких як тип блоку, його координати на робочому полі та зв'язки з іншими блоками. Важливими методами цього класу є функції для створення та видалення з’єднань між блоками, а також для оновлення їхнього положення та рендерингу. На основі цього базового класу побудовані спеціалізовані підкласи, кожен з яких додає специфічну функціональність для різних типів блоків.

Текстові блоки (TextBlock) містять текстовий вміст, який можна редагувати в будь-який момент. Блоки присвоєння (AssignmentBlock) дозволяють створювати змінні та задавати їм значення. Це забезпечує гнучкість у роботі з даними, оскільки змінні можуть бути як числовими, так і текстовими. Важливою функцією таких блоків є можливість динамічного оновлення значень змінних у процесі виконання алгоритму. Але при виконанні алгоритму, користувач вже не може змінювати текст, формули, ім’я змінних тощо.

Інтерактивність додатка забезпечують блоки введення (InputBlock), які під час виконання алгоритму запитують у користувача необхідні дані. Це робить алгоритми універсальними для роботи з різними наборами даних. Аналогічно, блоки виведення (OutputBlock) відповідають за відображення результатів або проміжних значень змінних, що дозволяє користувачу спостерігати за динамікою змін у процесі виконання.

Якщо під час виконання сталася помилка, користувач буде про це проінформований у відповідному полі. До того ж якщо користувач не додав жодного блока виводу, додаток все одно проінформує його, чи був алгоритм виконанний успішно, чи ні.

Особливе місце займають умовні блоки (ConditionBlock), які забезпечують розгалуження логіки залежно від заданих умов. Вони можуть виконувати прості логічні перевірки або складні вирази із використанням операторів порівняння. Завдяки цьому алгоритми можуть адаптуватися до різних сценаріїв виконання.

Уся логіка алгоритму зберігається та обробляється в класі Algorithm, який відповідає за виконання блоків у визначеному порядку. Цей клас підтримує як покрокове виконання, так і повний запуск алгоритму. Його основні функції включають збереження списку блоків, їх з’єднань, запуск алгоритму, скидання до початкового стану та додавання або видалення блоків із схеми.

Застосунок використовує технологію SVG для створення масштабованих блоків, що забезпечує чіткість і якість зображення незалежно від розміру екрана. Підсвічування активних блоків під час виконання алгоритму додає зручності та прозорості у відстеженні ходу виконання.

В основі виконання алгоритмів лежить концепція орієнтованого графа, де вузли представляють блоки, а ребра - з'єднання між ними. Для послідовного виконання блоків застосовується підхід на основі черги (FIFO), що гарантує правильний порядок виконання.

Таким чином, архітектура додатка поєднує модульність, гнучкість та інтерактивність, що дозволяє користувачам легко створювати, редагувати та налагоджувати алгоритми різної складності. Використання сучасних технологій та інтуїтивно зрозумілий інтерфейс забезпечують комфортну роботу та високу продуктивність у процесі створення блок-схем.

3.6. Класифікація вебдизайну

Вебдизайн визначається як процес створення інтерфейсу вебсайту в інтернеті. Хоча створення всього вебсайту вимагає додаткових навичок і ресурсів, таких як програмування і розробка, аспект дизайну зазвичай фокусується на користувацькому інтерфейсі і взаємодії між користувачем і сайтом. Користувацький досвід складається зі зовнішнього вигляду,

функціональних вимог, макету і змісту вебсайту. Вебдизайнери намагаються знайти найефективніші способи представлення інформації на вебсайті, щоб користувачі вважали її привабливою і корисною. Для досягнення цієї мети часто використовуються різні дизайни і макети, залежно від призначення і завдань вебсайту.

Основними елементами вебдизайну є:

- шрифти
- кольори
- зображення
- макети
- форми
- символи
- логотипи
- іконки
- відео

Вебдизайн - це динамічна та багатогранна галузь, яка охоплює різноманітні підходи та техніки для створення привабливих, зручних та функціональних вебсайтів. Основні види вебдизайну такі:

1. Статичний вебдизайн. Статичні вебсайти складаються з фіксованих сторінок, вміст яких не змінюється, коли користувачі отримують доступ до них. Вони створюються за допомогою HTML і CSS і підходять для невеликих вебсайтів і сторінок з рідко оновлюваною інформацією. Раніше він був поширеним, але зараз використовується рідко.
2. Динамічний вебдизайн. Динамічні вебсайти використовують серверні скрипти для створення контенту «на льоту» у відповідь на запити користувачів. Це дозволяє створювати інтерактивні вебсайти з великою кількістю постійно оновлюваного контенту. Популярні технології динамічного вебдизайну включають PHP,

ASP.NET і JavaScript. Цей тип вебдизайну прийшов на зміну статичному дизайну, але зараз перевага надається адаптивному дизайну.

3. Адаптивний вебдизайн. Адаптивний вебдизайн використовує гнучку сітку і макет, які автоматично підлаштовуються під розмір екрана пристрою, на якому переглядається сайт. Це підвищує зручність доступу до контенту з будь-якого пристрою, включаючи комп'ютери, планшети та смартфони. Це також найпоширеніший тип дизайну сьогодні.
4. Мобільний дизайн. Цей підхід передбачає створення сайту, орієнтованого насамперед на мобільні пристрої. Дизайнери спочатку створюють макет для смартфонів і планшетів, а потім адаптують його для великих екранів. Це гарантує, що вебсайт оптимізований для користувачів, які переглядають його з мобільних пристроїв.
5. Односторінковий дизайн. Односторінкові вебсайти містять весь контент на одній сторінці з можливістю прокрутки. Підходить для презентацій, портфоліо, промо сторінок тощо. Ключовим моментом у цьому типі дизайну є забезпечення плавних переходів між різними частинами сторінки.
6. Матеріальний дизайн. Матеріальний дизайн був розроблений компанією Google і базується на принципах реального світу, використовуючи тіні, освітлення та рух для створення природного та інтуїтивно зрозумілого інтерфейсу. Він часто використовується в додатках для Android та вебсайтах із сучасним виглядом.
7. Паралакс-дизайн. Паралакс-дизайн використовує ефект паралакса, коли фон рухається повільніше, ніж передній план під

час прокрутки. Це створює ефект глибини і є дуже ефективним у створенні візуально привабливих та інтерактивних вебсайтів.

8. Графічний вебдизайн. Графічний дизайн фокусується на використанні зображень, іконок, типографіки та інших візуальних елементів для створення привабливого інтерфейсу. Він відіграє важливу роль у брендингу вебсайту та покращенні його естетичної привабливості.

Різні види вебдизайну можна комбінувати для створення сучасних, зручних і привабливих вебсайтів. Кожен вид вебдизайну має свої сильні сторони, і їх поєднання може допомогти досягти поставлених цілей максимально ефективно. Наприклад, статичний і динамічний дизайн можна комбінувати для створення статичних сторінок зі швидким завантаженням та інтерактивними елементами, такими як форми зворотного зв'язку та інтеграція з соціальними мережами. Паралакс можна використовувати для додавання глибини та динамічності, а графічний дизайн може забезпечити естетичну привабливість завдяки високоякісним зображенням та іконкам. Матеріальний дизайн можна використовувати для створення сучасного вигляду з реалістичними візуальними ефектами. Адаптивний вебдизайн гарантує зручність використання на всіх пристроях, від комп'ютерів до смартфонів. Тому його можна комбінувати з іншими типами вебдизайну, щоб зробити сайт більш привабливим і розширити цільову аудиторію.

Таким чином, комбінуючи різні типи вебдизайну, дизайнер може створити багатофункціональний і привабливий вебсайт, який відповідає потребам користувачів і бізнесу.

3.7. Принципи побудови візуально привабливого дизайну

Існує два основних підходи, які більшість людей використовують для оцінки того, чи є дизайн вебсайту “хорошим” чи “поганим”. Перший - це суворий юзабіліті підхід, який фокусується на функціональності, ефективному поданні інформації та економічності контенту. Другий - суто

естетичний підхід, який фокусується на художній цінності та візуальній привабливості дизайну. Деякі люди настільки захоплені естетикою та графікою, що забувають про користувача, тоді як інші дизайнери настільки занурюються у функціональність, що забувають про візуальну привабливість. Важливо максимально використовувати обидва аспекти, щоб привабити людей і зацікавити їх.

Дизайн - це комунікація. Дизайнер може створити вебсайт, який добре працює і надає інформацію, але якщо він не виглядає привабливо або не відповідає бренду клієнта, ніхто не захоче ним користуватися. Аналогічно, якщо дизайнер створить гарний вебсайт, але він буде складним у використанні або доступі, люди будуть триматися якомога далі від нього. Елементи та функції завершеного вебдизайну повинні працювати як єдина система.

Основні підходи до створення “хорошого” дизайну - це правильно розміщений контент, інтуїтивно зрозуміла навігація, дотримання певного макета та підходу (іконки, кольорові схеми, шрифти тощо) для всіх елементів сайту, а також загальна симетрія макета.

Користувачів приваблює дизайн, але вони залишаються на сайті заради контенту. Однією з найбільших проблем юзабіліті для вебдизайнерів є час, який користувачі витрачають на сканування сторінки, щоб знайти потрібну інформацію, чи то контент, чи то посилання на іншу сторінку, чи то поле форми. Дизайн повинен бути каналом між користувачем та інформацією, а не перешкодою.

Користувачі повинні мати можливість легко орієнтуватися за допомогою інтуїтивно зрозумілої навігації. Основні навігаційні блоки повинні бути добре помітні на сторінці, а кожне посилання повинно мати описову назву. Структура навігації, яка не тільки змінює вигляд при наведенні курсора, але й вказує на активну сторінку або розділ, допомагає

користувачам зрозуміти, де вони знаходяться і як дістатися до місця призначення.

Вторинна навігація, пошукові поля та зовнішні посилання не повинні домінувати на сторінці. Дизайнер повинен зробити дизайн таким чином, щоб всі елементи сайту було легко знайти, і візуально відокремити їх від основного контенту, щоб користувачі могли зосередитися на інформації, але знали, де шукати, коли захочуть перейти до іншого контенту.

Узгодженість усіх елементів сайту. Навіть якщо є значні відмінності між макетом головної сторінки та рештою вебсайту, всі сторінки повинні бути витримані в єдиній тематиці та стилі, щоб підтримувати єдність дизайну. Єдність досягається шляхом повторення елементів ідентифікації та навігаційних блоків. Послідовне використання обмеженої колірної палітри також допомагає уніфікувати сторінки.

Крім того, для створення візуально привабливого інтерфейсу необхідний баланс. У метафоричному сенсі концепція візуального балансу схожа на фізичну рівновагу. Подібно до того, як фізичні об'єкти мають вагу, елементи макета також мають вагу. Якщо елементи з обох боків макета мають однакову вагу, вони врівноважують один одного. Існує два основних типи візуального балансу: симетричний і асиметричний. Симетричний баланс, або формальний баланс, виникає, коли елементи композиції однакові по обидва боки від центральної лінії. Хоча для деяких вебсайтів це непрактично, горизонтальна симетрія може бути застосована в макеті вебсайту шляхом центрування контенту або балансування між колонками. Асиметричний або неформальний баланс трохи абстрактніший (і часто візуально цікавіший), ніж симетричний. Замість того, щоб розміщувати дзеркальні зображення з обох боків макета, асиметричний баланс включає об'єкти різних розмірів, форм, тонів і розташування. Наприклад, розміщення великого об'єкта з одного боку макета і декількох менших об'єктів з іншого боку виглядає збалансовано. На відміну від симетричного балансу,

асиметричний баланс є більш універсальним і тому частіше використовується у вебдизайні. У переважній більшості макетів сайтів з двома колонками, ширші колонки зазвичай світліші за кольором. Вужчі колонки навігації зазвичай темнішого кольору, їх обводять або виділяють іншими способами, щоб збалансувати макет.

3.8. Дизайн додатка

Розробка вебдизайну додатка спиралася на вимоги видавництва “Ранок” та можливості платформи IZZI.

Крім того, дизайн вебдодатка конструктора алгоритмів повинен бути інтуїтивно зрозумілим та естетично привабливим. Основна робоча область розміром 1050*850 пікселів оптимізована для зручного розміщення елементів блок-схеми.

Використання IBM Plex Sans шрифту гарантує, що текстові елементи, такі як заголовки, описи та коментарі, є чіткими і легкими для читання. Шрифт є сучасним, чистим та естетично привабливим і не викликає перенапруження очей навіть після тривалої роботи.

В інтерфейсі програми використовуються іконки з набору Google Material Icons. Ці іконки відомі своєю наочністю та простотою використання і дозволяють навіть новачкам швидко освоїти інтуїтивно зрозумілий інтерфейс. Використання цих іконок для основних інструментів і функцій робить навігацію в додатку більш інтуїтивно зрозумілою і надає інтерфейсу сучасного вигляду.

Основним кольором дизайну є волошковий синій (HEX-код #4A90E2), який створює спокійну та професійну атмосферу. Цей колір можна використовувати для виділення активних елементів, кнопок та індикаторів стану, що дозволяє користувачам легко ідентифікувати свою поточну позицію та основні дії в додатку. Волошковий добре поєднується з іншими кольорами, створюючи цілісний і привабливий інтерфейс.

Пряний фіолетовий (HEX-код #C30E60) використовується для виділення важливих текстових елементів і полегшує ідентифікацію важливої інформації. Пряний фіолетовий також додає візуального інтересу та контрасту до загального дизайну, допомагаючи зосередити увагу на ключових елементах.

Дизайн кнопок і перемикачів (радіокнопок) відповідає загальному дизайну видавництва, з використанням сучасних графічних елементів і м'якої анімації. Кнопки мають чіткі контури і є достатньо великими, щоб їх можна було легко ідентифікувати як на десктопних, так і на мобільних пристроях. Плавна анімація кнопок підвищує інтерактивність, покращує користувацький досвід і робить додаток цікавішим та захопливішим.

Крім того, додаток повинен підтримувати функцію масштабування і перетягування елементів блок-схеми для забезпечення високої ефективності. Це означає, що користувачі мають можливість змінювати, переміщати блоки в робочу область і пов'язувати блоки між собою, що сприяє швидкому і ефективному створенню блок-схем.

Таким чином, дизайн вебдодатка конструктор алгоритмів є функціональним, корисним і естетичним. Інтуїтивно зрозумілий інтерфейс і зручні функції масштабування і перетягування роблять додаток незамінним інструментом для всіх, хто працює з блок-схемами.

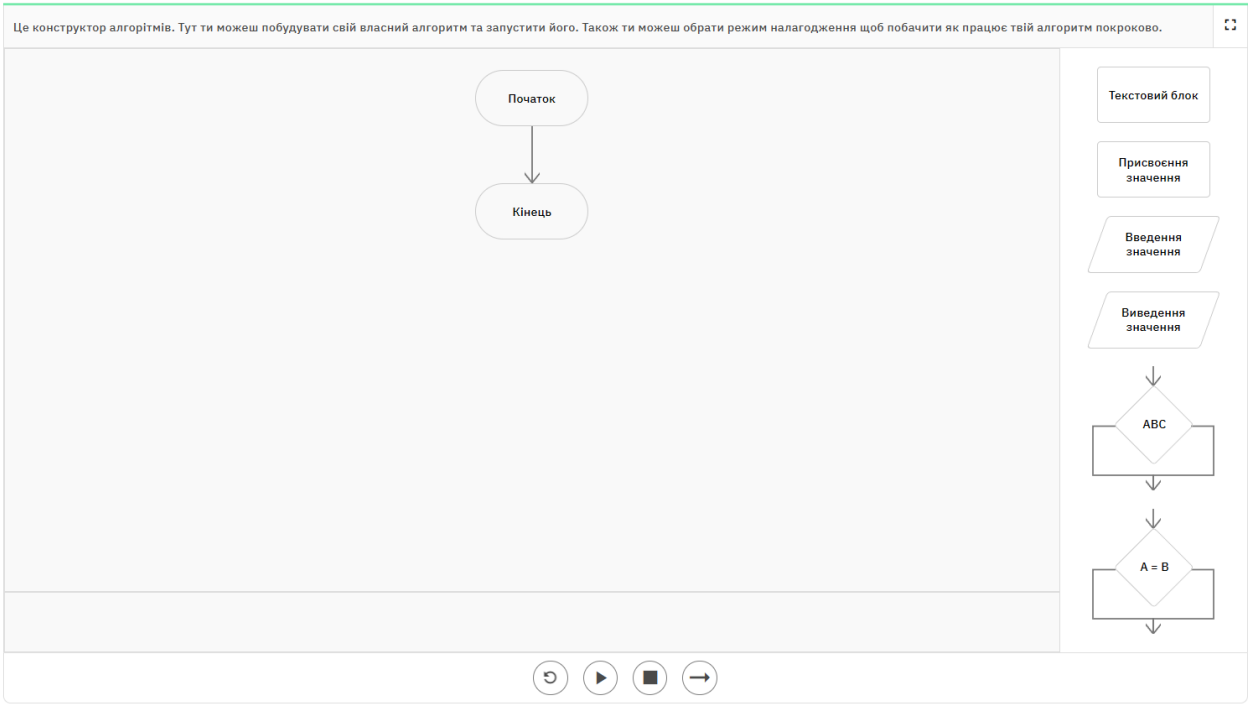

Рисунок 3.6 Загальний дизайн конструктора алгоритмів.

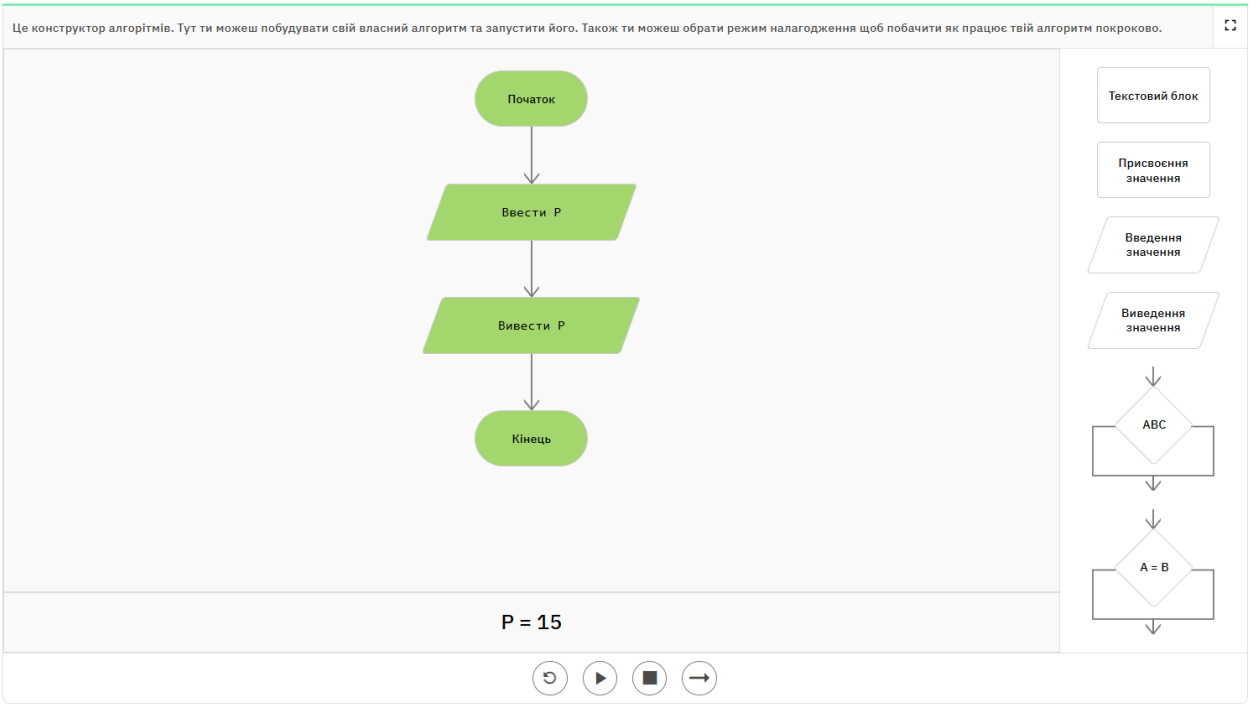

Рисунок 3.7 Дизайн конструктора алгоритмів із блоками вводу та виводу.

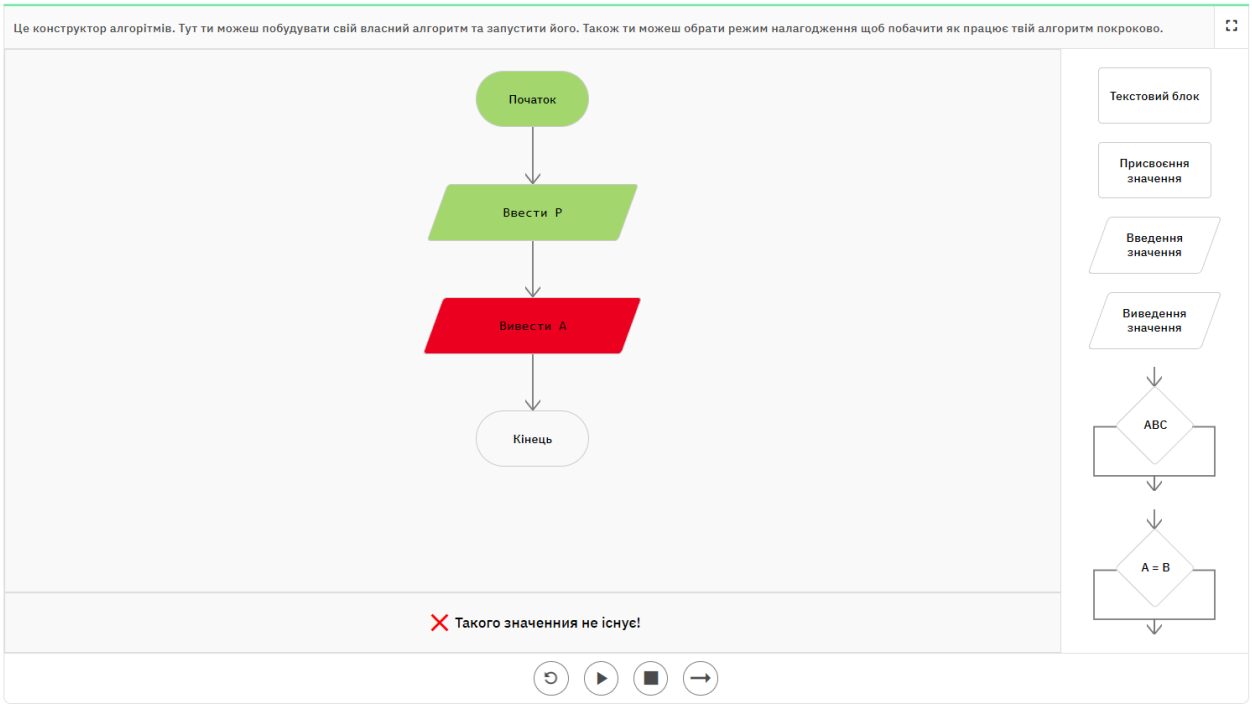

Рисунок 3.8 Приклад обробки не валідних алгоритмів. Обробка алгоритму в якому виводиться попередньо не задана змінна.

ВИСНОВКИ

В результати даної робити була розроблена архітектура та була реалізована вебверсія конструктора алгоритмів: модуль учня. У процесі роботи були вивчені основні поняття алгоритмів, блок-схем та принципи їх побудови. Ця теоретична база стала основою для реалізації функціональних можливостей додатка.

Порівняння з іншими конструкторами алгоритмів виявило їхні недоліки. Більшість з них не мають можливості для інтерактивного налагодження алгоритмів, що є одним із вирішальних переваг нового додатку.

Визначення функціональних і нефункціональних вимог до вебдодатка забезпечило його сумісність з вимогами видавництва “Ранок” та платформи IZZI. Це дозволило створити інтуїтивно зрозумілий інтерфейс, що спрощує навчальний процес, а також забезпечило інтеграцію вебдодатка в освітнє середовище.

Використання сучасних технологій дозволило досягти високої продуктивності та зручності використання додатка. Архітектура клієнт-сервер забезпечує ефективний обмін даними між серверами IZZI та кінцевими користувачами.

Вебдодаток може бути покращений шляхом включення нових функцій, таких як додаткові типи блоків, аналіз помилок алгоритмів та інтеграція з іншими освітніми платформами. Це дозволить охопити ще ширше коло користувачів і підвищити інтерактивність навчання.

Таким чином, успішна реалізація даного проєкту не лише відповідає на сучасні виклики освітнього процесу, а й закладає основи для подальшого розвитку інструментів навчання у сфері програмування та алгоритмічного мислення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М. Є. Яременко. *Розробка логічних завдань для інтерактивних навчальних ресурсів на платформі IZZI.*
2. І. А. Сергієнко. *Проєктування додатка (конструктор алгоритмів) на платформі IZZI*
3. І. А. Сергієнко. *Проєктування вебдизайна для додатка (конструктор алгоритмів) на платформі IZZI*
4. Sue Reynard. *Flowcharts: Plain and Simple*
5. A. B. Chaudhuri. *Flowchart and Algorithm Basics: The Art of Programming*
6. Jennifer Niederst Robbins. *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics 4th Edition*
7. Robert Damelio. *The Basics of Process Mapping 2nd edition*
8. Michel H. Boillot, Gary M. Gleason, Lister Wayne Horn. *Essentials of Flowcharting*
9. Paul McFedries. *Web Design Playground: HTML & CSS the Interactive Way 1st Edition*
10. Mark N Wayne. *Flowcharting concepts & data processing techniques: A self-instructional guide*
11. Bernard B. Bycer. *Flowcharting: Programming, Software Designing, and Computer Problem Solving*
12. John K. Lenher. *Flowcharting: An Introductory Text and Workbook*
13. A. B. Chaudhuri. *The Art of Programming Through Flowcharts & Algorithms*
14. Doogie Horner. *Everything Explained Through Flowcharts*
15. Farid Golnaraghi, Benjamin C. Kuo. *Automatic Control System Tenth Edition*
16. Tal Ater. *Building Progressive Web Apps: Bringing the Power of Native to the Browser 1st Edition*

17. *Flowchart Smart: Sets 1 – 3*
18. Jason Beaird, James George. *The Principles of Beautiful Web Design*
19. William R Parks. *Sets, Numbers and Flowcharts 1st edition*
20. Hazel Thornton. *Go With the Flow! The Clutter Flow Chart Workbook*
21. Irving Brangan. *Flowcharts Usage: Get An Introduction To The Use Of Flowcharts As A Process Design*
22. T. D. Graybeal. *Block Diagram Network Transformation*
23. Theo Farrington. *UX Design 2020: The Ultimate Beginner's Guide to User Experience*
24. draw.io [Електронний ресурс]. – Режим доступу : URL : *https://app.diagrams.net/*
25. Diagram.codes [Електронний ресурс]. – Режим доступу : URL : https://playground.diagram.codes/
26. Flowgorithm [Електронний ресурс]. – Режим доступу : URL : https://flowgorithm.org/