КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп’ютерних наук та кібернетики
Кафедра Теорії та Технології програмування

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп’ютерні науки : Інформатика
на тему:

РОЗРОБКА ЕЛЕКТРОННОГО ПІДРУЧНИКА З КУРСУ “ТЕОРІЯ АЛГОРИТМІВ”

Виконав студент 4-го курсу
Дмитро ПОЛІЩУК

(підпис)

Науковий керівник
доктор фізико-математичних наук, професор
Степан ШКІЛЬНЯК

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«05» червня 2023 р.
протокол № 18

Завідувач кафедри
Микола НІКІТЧЕНКО

(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи – 48 сторінок, 23 ілюстрації, 1 таблиця, 10 джерел посилань.

Тема: Розробка електронного підручника з курсу “Теорія алгоритмів”

Перелік ключових слів: електронний підручник, React js, node js, HTML, CSS, MHP-програма, MT-програма, Algorithmix.

Об’єктом роботи є процес інформатизації освіти з використанням сучасних інформаційних технологій. Зокрема, розробка електронних навчальних матеріалів, включаючи аналіз підходів до їх створення та реалізація алгоритмів для ілюстрації їх роботи.

Метою роботи є створення електронного підручника з курсу «Теорія алгоритмів» для надання чіткої і доступної інформації користувачам в будь-якому куточку земної кулі; реалізація можливості дистанційного навчання за допомогою цього проекту.

Задачі, поставлені для досягнення мети:

- Аналіз існуючих е-підручників;
- Аналіз особливостей створення е-підручників;
- Вибір технологій створення проекту;
- Розробка сервісу.

Результати роботи: створено сервіс-електронний підручник з різними методами надання інформації користувачу; опрацьовано інформацію стосовно створення електронних навчальних матеріалів; вироблено практичні вміння і навички щодо створення WEB-додатків.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

React js – це бібліотека мови JavaScript, яка слугує для створення користувальницьких інтерфейсів;

Node.js – оточення JavaScript, спроектоване для розробки великих мережевих додатків;

HTML – Hypertext Markup Language – це код, використання якого застосовується для структурування і відображення веб-сторінки з контентом;

CSS – (Cascading Style Sheets) мова стилів, що використовується для опису зовнішнього вигляду веб-сторінок, написаних мовою розмітки HTML;

VS Code – Visual Studio Code – редактор коду для розробки веб додатків;

МНР – машина з натуральнозначними регістрами;

МТ – машина Тьюрінга;

Е-підручник – електронний підручник;

Algorithmix – назва створеного електронного підручника.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Розвиток і популяризація інтернету дає людям змогу черпати більше корисної інформації. На допомогу цьому приходять різні веб-сервіси, блоги, електронні книги, тощо. Первинні веб-сервіси складались з вузькоспрямованої інформації, якої до того ж було не так багато, що насамперед пов’язано з малою кількістю користувачів і недостатнім розвитком інтернет-тренду.

Актуальність роботи та підстави для її виконання. Сучасна інтернет-мережа переповнена різноманітними електронними підручниками. Обираючи спосіб навчання або просто самовдосконалення, люди все частіше віддають перевагу електронній версії підручника, ніж звичайній. Переваги електронного носія видно одразу: такий носій – це можливість мати будь-який вид інформації кожен раз під рукою. Електронний підручник є програмним засобом для дистанційного навчання, яке охопило великий відсоток учнів та студентів по всій планеті. Роль електронної освіти та дистанційного навчання незмірно зростає в наш час, коли в Україні триває війна. Функціями електронного підручника є: здобуття довідкових знань, розвиваюча функція та закріплення набутого матеріалу.

Мета й завдання роботи. Мета роботи полягає в розробці електронного підручника з курсу «Теорія алгоритмів» за допомогою нових технологій веб-програмування. Вибудовування на цій основі вагомого підгрунтя для розширення сфери застосування комп'ютерних технологій навчання, а також:

- Ознайомитись з концепцією веб-програмування;
- Дослідити і знайти найкращий спосіб створення електронного підручника;
- Виконати практичну частину заданої теми;
- Проілюструвати роботу МНР програми;
- Проілюструвати роботу машини Тьюрінга.

- Підсумувати проведену роботу та визначити для себе особливості створення електронного носія для подальшої розробки власного продукту.

Об’єкт, методи й засоби розроблення Попередньо ознайомившись з великою кількістю електронних навчальних матеріалів, звернено увагу на сучасні тенденції щодо навігації по темах в наданих матеріалах сервісу. Інструментом розробки проекту було обрано бібліотеку React JavaScript для створення привабливого інтерфейсу довідника з метою утримання відвідувача. Node.js - оточення js для створення умов з реєстрації особистого кабінету через електронну пошту.

Можливі сфери застосування. Форма дистанційного навчання тягне за собою прогресивний формат взаємодії студентів з викладацькими матеріалами не напряму в тактильному варіанті, а за допомогою електронних ресурсів, до яких можна віднести розроблений продукт під назвою **електронний підручник з курсу «Теорія алгоритмів»**.

РОЗДІЛ 1 ЗАГАЛЬНИЙ ОГЛЯД ЕЛЕКТРОННИХ ПІДРУЧНИКІВ

1.1 Історія виникнення електронних книг

У сучасних умовах розвитку інформатизації суспільства, велика увага приділяється інформатизованому супроводу професійної діяльності у різних галузях економіки. Застосування нових засобів обміну інформацією дозволяє істотно спростити процес передавання інформації, обміну знаннями та досвідом. За рахунок активного впровадження інформаційних технологій, які допомагають ширше і швидше передавати матеріал з використанням засобів мультимедіа, а також зберігати великі обсяги інформації, істотно зростає значення використання електронних книг (підручників).

Електронна книга (e-Book, pocketbook) – це цифровий пристрій планшетного типу, призначений для зберігання і відображення оцифрованої інформації. Назва електронної книги походить від англійського слова" digital book", що в перекладі може означати" читалка"," цифрова книга" або" електронний папір".

Першим винахідником електронної книги виявився американський письменник Майкл Стерн Харт, який у 1971-му році здійснив оцифровку на електронному носії історичного документу "Декларацію незалежності США". Приводом для здійснення цієї роботи письменником виявилася зручність у користуванні документом у такому вигляді, можливість за допомогою електроніки розширити межі важливого документа.

Після оцифрування Майклом Хартом ще кількох книг на електронні носії, він почав працювати над проектом "Гутенберг". Головною метою цього проекту було створення електронної бібліотеки і розширення її можливостей. Результатом реалізації зазначеного проекту було оцифрування значної кількості книг. При цьому частина з них були занесені самим М. Хартом. Станом на 2005

рік електронний ресурс бібліотеки поміщав в собі понад 17 000 електронних матеріалів.

Хоча Майкла Харта по праву вважають винахідником електронної книги, проте пристрій для відображення текстової інформації на дисплеях першою придумала і винайшла в 1996-му році американська комп'ютерна компанія DEC. Під назвою " DEC Lectrice", що в перекладі з англійської означає "читач".

Перша електронна книга мала вигляд подібний до планшетного комп'ютера з залізним корпусом і кнопковою панеллю.

Висока собівартість перших електронних книг не сприяла масовому їх виробництву і поширенню. І лише після винаходу у 2007 році «електронного паперу», який дозволив текст, що зявляється на екрані робити більш помітним і менш яскравим значною мірою дало поштовх для популяризації електронних книг.

Першою моделлю електронної книги, випуск якої поставили на потік, стали девайси з LCD-екраном – їх виробництвом одночасно зайнялися Nuvo Media Softbook Press. Ще трохи – і ті ж компанії стали виробляти книги з кольоровими, рідкокристалічними екранами.

У структурі електронних книг використовують процесори ARM. Операційні системи пристроїв працюють на базі ядра Linux. За допомогою сучасних носіїв інформації можна не тільки читати текст, а й дивитися фотографії, завантажувати і слухати музику і навіть грати в прості ігри.

Дисплеї сучасних букридерів бувають двох типів: E-Ink Pear і LED-дисплей. Між собою вони відрізняються колірним відображенням і яскравістю. Сторінки на E-Ink Pear екрані мають жовтуватий відтінок, подібні до газетного формату. Водночас картинка на LED- дисплеї відображається більш яскраво і на білому тлі.

1.2 Основні задачі та принципи створення електронного підручника

Швидкі темпи накопичення та поширення інформації, обумовлені виникненням і розвитком комп'ютерних технологій, викликають необхідність пошуку нових підходів до вирішення нетрадиційних проблемних питань, які виникають при впровадженні цих технологій у різних сферах діяльності суспільства [2].

Можливості інформаційних технологій як інструменту діяльності людини і нового засобу навчання [3] приводить до появи нових методів, засобів, форм навчання і контролю. Сучасний навчальний процес неможливий без використання електронних навчальних видань, що є новим видом навчальної літератури. Інформаційні освітні ресурси, зокрема електронні підручники зайняли важливу нішу в освітньому процесі і сприяють підвищенню рівня науково-дослідницької та пошукової діяльності. Разом з тим, найбільший ефект від застосування електронного підручника досягається за умови його поєднання з іншими навчальними системами, у тому числі друкованими виданнями, які доповнюють одне одного.

Електронний підручник – це інтерактивний підручник, який існує в режимі онлайн та містить різні мультимедійні матеріали: відео, графіку, відео курси; він відзначається характерними особливостями, серед яких:

– багаторівневість подання інформації та поєднання інформаційних масивів різних типів на підставі асоціацій в єдине смислове ціле;

–можливість здійснення діяльнісного характеру навчання, що дозволяє включити до складу підручника структурні елементи, які надають можливість комплексного використання у навчанні як традиційних і нових видів навчальної діяльності,

– збагачення процесу навчання наочним високоякісним ілюстративним матеріалом – двохвимірними, об'ємними, статичними і динамічними зображеннями, звуковим супроводом відображеного на екрані матеріалу та дій того, хто навчається.

– забезпечення якісного зворотного зв’язку, завдяки чому створюються умови для ефективного самонавчання, самоконтролю, самокорекції та підвищення пізнавальної активності того, хто навчається.

– інтегрованість електронного підручника, що створює сприятливі умови для швидкого оновлення та модифікації представленого в підручнику навчального матеріалу, динамічного збагачення його новим змістом відповідно до рівня сучасної науки [4].

Електронний підручник (е-посібник), як різновид електронного навчального видання, незалежно від його змістового наповнення й типології, на думку науковців [1] має вирішувати низку задач і відповідати певним вимогам, а саме: оптимальному забезпеченню взаємодії оператора з комп’ютером; досягненню мети й завдань навчання; адаптації до індивідуальних особливостей суб’єктів навчання; проблемно-орієнтованій подачі матеріалу завдань; спрямованості на інтенсивне керування процесом пізнання. При застосуванні е-підручників слід враховувати такі чинники, як ступінь відповідності інформаційного й технологічного забезпечення підручника навчальній програмі з певного предмета; позитивність впливу мотиваційних орієнтацій на формування знань і вмінь більш високого рівня; варіативність індивідуалізованих та диференційованих навчальних завдань; інтенсивність використання інноваційних методів навчання.

Електронний підручник має певні переваги над традиційними видами посібників, зокрема вивчення матеріалу може бути не пов’язане із часовими рамками (аудиторними заняттями), він дає змогу розвивати навички самостійної роботи слухачів, його структура допомагає встановлювати контроль над вивченням відповідних блоків тем [2].

Розроблення сучасних електронних підручників, як відмічено в [5], має ґрунтуватися на відповідних принципах, а саме:

– відображення інформації з використанням різних даних: тексту, графіки, аудіо, відео, анімації (мультиплікації);

– забезпечення можливостей пошуку і вибору довідкової інформації;

– об’єктивність та різнобічність системи контролю знань;

– можливість інтерактивного зв’язку учня з учителем за допомогою мережевих технологій [5].

При розробці структури та змісту електронного підручника (посібника) слід враховувати наступні принципи та особливості:

– принцип пріоритетн ості педагогічного підходу;

– принцип модуля: розбиття матеріалу на розділи, що складаються з модулів, мінімальних за обсягом, але «замкнутих» за змістом

– принцип повноти, згідно з яким кожен модуль повинен складатися з відповідних компонентів;

– принцип наочності, за якого в основі створення електронного підручника є теорія мультисенсорного навчання;

– принцип розгалуження, згідно з яким кожен модуль повинен пов’язуватися гіперпосиланням з іншими модулями для того, щоб користувач міг вільно переходити з одного модуля в інший;

– принцип регулювання, що передбачає самостійну зміну кадрів, а також урахування всіляких елементів управління;

– принцип адаптивності, згідно з яким в електронному підручнику повинна враховуватися адаптація до потреб конкретного користувача в процесі навчання (див. [6]).

1.3 Розробка електронних підручників

Розробка електронного підручника базується на системному підході, який розглядає об’єкт як систему, що складається з низки взаємозалежних елементів, що утворюють певну цілісність і вимагає дотримання відповідних методичних вимог, за яких:

– навчальний матеріал має бути розбитий на блоки;

– кожний блок має містити детальні ілюстрації;

– ілюстрації мають підбиратися таким чином, щоб більш детально й просто пояснити матеріал, який важко сприймається слухачами;

– основний матеріал блоку має об’єднуватися в одне ціле за допомогою гіперпосилань [гіперпосилання (інколи гіперланка) – це активний (виділений кольором) текст, зображення чи кнопка на веб-сторінці, натиснення на яку (активізація гіперпосилання) викликає перехід на іншу сторінку чи іншу частину поточної сторінки]. Гіперпосилання можуть зв’язувати й окремі блоки електронного посібника;

– доцільно доповнити матеріал посібника підказками, що спливають [2].

Як правило, електронні навчальні підручники будуються за модульним принципом і містять всю необхідну інформацію, згруповану в декілька частин;

– теоретична частина, в основі якої міститься текст, графіка (статичні схеми, креслення, таблиці, рисунки), анімація, відеозаписи, а також інтерактивний блок;

– практична частина. Має бути представлено покрокове розв’язання типових задач (завдань) і вправ з певного навчального курсу, де містяться мінімальні пояснення;

– контрольна частина – містить набір текстів, контрольних питань з теоретичної частини, проміжні тести, що дають змогу оцінити одержані знання і відкрити доступ до подальшого рівня навчання (інших більш складних блоків навчального матеріалу), також рішення завдань і вправ із практики. При складанні проміжних і підсумкових тестів, що проводяться в рамках конкретного електронного посібника, рекомендується брати за основу педагогічні вимірювальні матеріали, які використовуються в системі дистанційного навчання;

– довідкова частина, яка може містити предметний покажчик, таблиці основних констант, розмірностей (значень), фізико-хімічних властивостей, основні формули й іншу необхідну інформацію у графічній, табличній чи будь-якій іншій формі.

РОЗДІЛ 2. ТЕОРІЯ АЛГОРИТМІВ

2.1 Поняття та визначення

Теорія алгоритмів є наукою про загальні властивості алгоритмів. Першочергова задача цієї науки полягає в дослідженні та вивченні обчислюваних функцій та моделей алгоритмів. Тому теорію алгоритмів слід виділити як галузь комп'ютерних наук, яка вивчає алгоритми, їх властивості та пов'язані з ними проблеми. Одна з центральних задач прикладної теорії алгоритмів – це аналіз часової та просторової складності алгоритмів, тобто оцінка кількості операцій або пам'яті, яку вони вимагають для виконання завдання. Це допомагає визначити, наскільки ефективним є алгоритм з точки зору ресурсів. Теорія алгоритмів є основою для розробки нових алгоритмів та вдосконалення існуючих. Вона використовується в різних галузях, включаючи програмування, штучний інтелект, обробку даних, оптимізацію, криптографію та багато інших.

Скінченна множина точно визначених правил для чисто механічного розв’язку завдань певного класу – саме це визначення являється стислим описом того, що є алгоритмом.

Характерними властивостями алгоритму є: фінітність, масовість, дискретність, елементарність та результативність, то ж давайте їх розглянемо.

Фінітність означає скінченність алгоритму – це необхідна умова його реалізовності.

На **масовість** вказує призначеність алгоритму на розв1язання певної вибірки (класу) однотипних задач, а не однієї конкретної. Це дає можливість обирати дані для алгоритму з множини початкових даних.

Дискретністю є поділ процесу роботи алгоритма на окремі кроки, що свідчить про роботу в дискретному часі.

Елементарність – кожен крок алгоритму має бути простим, елементарним, можливість виконання якого як людиною, так і машиною, не викликає сумнівів.

Результативністю можна назвати спрямованість алгоритму на отримання результату, а саме наявність можливості виокремлення результативних даних (див. [7]).

Таким чином, теорію алгоритмів можна охарактеризувати, як важливу складову комп'ютерних наук, яка досліджує загальні властивості алгоритмів. Теорія алгоритмів є теоретичним фундаментом програмування.

2.2 Машина Тьюрінга

Машину Тьюрінга запропоновано в 1936 році англійським математиком Аланом Тьюрінгом. Машина Тьюрінга є ідеалізованою моделлю обчислювальної машини. Вона була створена для вивчення можливостей обчислення та формалізації поняття алгоритму.

Будова машин Тьюрінга. Неформально машина Тьюрінга складається з нескінченної стрічки розділеної на комірки з обох боків, головки зчитування-записування та функцією переміщення по горизонтальній стрічці ліворуч та праворуч по коміркам, скінченний алфавіт – T, множини(скінченної) внутрішніх станів – Q, та множини станів q разом з початковим [8].

	#	$	0	1	a_0
q_1	←	←	←	←	→ q_2
q_2	0 →	0 →	→	→	q_0

Рис.1 Приклад програми Машини Тьюрінга

Формально *машина Тьюрінга* – це впорядкована п’ятірка (Q, T, δ, q_0, F), де:

– Q – скінченна множина внутрішніх станів;

– T – скінченний алфавіт символів стрічки, причому T містить спеціальний символ порожньої клітини λ;

– $\delta : Q \times T \to Q \times T \times \{R, L, \varepsilon\}$ – функція переходів;

– $q_0 \in Q$ – початковий стан;

– $F \subseteq Q$ – множина фінальних станів.

Функцію переходів зазвичай задають множиною команд одного з 3-х типів:

$qa \rightarrow pbR$,

$qa \rightarrow pbL$,

$qa \rightarrow pb$.

Наведемо приклади машин Тьюрінга.

$q_0| \rightarrow q_1 \lambda R$
$q_1| \rightarrow q_1 |R$
$q_1 \# \rightarrow q^* |$
$q_0 \# \rightarrow q^* \lambda$

$q_0| \rightarrow q_1 \lambda R$
$q_1| \rightarrow q_1 |R$
$q_1 \# \rightarrow q_1 \# R$
$q_1 \lambda \rightarrow q_2 \lambda L$
$q_2| \rightarrow q_3 \lambda L$
$q_3| \rightarrow q_3 |L$
$q_3 \# \rightarrow q_3 \# L$
$q_3 \lambda \rightarrow q_0 \lambda R$
$q_2 \# \rightarrow q^* |$
$q_0 \# \rightarrow q_4 \lambda R$
$q_4 \lambda \rightarrow q^* \lambda$

Рис.2-3 Приклади МТ для $x + y$ та $x - y$

$q_0| \rightarrow q_1 \lambda R$
$q_1| \rightarrow q_0 \lambda R$
$q_0 \lambda \rightarrow q^* |$
$q_1 \lambda \rightarrow q^* \lambda$

Рис.4 Приклади МТ програми, яка обчислює предикат “x парне”

Машина Тьюрінга має множину станів, які можуть змінюватись відповідно до внутрішніх правил. Задача машини Тьюрінга полягає в тому, щоб робити переходи між станами та змінювати символи на стрічці залежно від правил (команд), що визначені функцією переходів. Машина Тьюрінга може виконувати такі операції, як зчитування символу з поточної комірки, запис символу на поточну комірку, переміщення на наступну комірку або попередню комірку та зміна свого внутрішнього стану. За допомогою цих базових операцій машина Тьюрінга може моделювати будь-який алгоритм.

2.3 Машини з натуральнозначними регістрами

Машину з натуральнозначними регістрами природно вважати ідеалізованою моделлю комп’ютера. МНР складається з нескінченної кількості регістрів які вміщають натуральні числа. Головна ідея машини з натуральнозначними регістрами полягає у використанні набору регістрів, які здатні зберігати натуральні числа, тобто цілі невід'ємні числа. Кожен регістр має свою адресу, і до нього можна звернутися для зчитування або запису значення. Завдяки скінченому списку команд утворюється сама МНР-програма. Команди програми мають бути послідовно пронумерованими і відрізнятись особливими буквами, які несуть за собою тип команди. Номер команди називаємо адресою, в той час як тип команди відповідає за дію, яку виконує стрічка МНР. Команд є 4, розглянемо їх типи:

- Z(n) – обнулення n-го регістру Z(n).
- S(n) – збільшення регістру на 1.
- T(m,n) – копіювання вмісту m регістру у n (команда називається арифметичною).
- J(m,n,q) – команда умовного переходу, яка порівнює значення m та n регістрів. При однакових значеннях переходить до виконання q (адреса переходу) команди. При різних значеннях переходить до виконання наступної команди.

Розглянемо приклади МНР-програм для функцій:

- МНР-програма для функції x + y
    1) J(1,2,5)
    2) S(0)
    3) S(2)
    4) J(0,0,1)

- МНР-програма для функції x – y
    1) J(0,1,5)
    2) S(1)
    3) S(2)
    4) J(0,0,1)
    5) T(2,0)

- МНР-програма для функції [x/2]
    1) J(0,2,7)
    2) S(2)
    3) J(0,2,7)
    4) S(2)
    5) S(1)
    6) J(0,0,1)
    7) T(1,0)

- МНР-програма для обчислення максимального елемента з двох зазначених
    1) J(0,2,5)
    2) J(1,2,6)
    3) S(2)
    4) J(0,0,1)
    5) T(1,0)

- МНР-програма для всюди невизначеної функції
    1) J(0,0,1)

Розглянемо тепер приклад обчислення функції x + y зі значеннями аргументів x=1, y=2.

МНР-програма: (J(1,2,5), S(0), S(2), J(0,0,1))

Таблиця 1 – Покрокове виконання обчислення функції додавання МНР-програмою

№ Крок	**R_0**	**R_1**	**R_2**	**Опис**
1	1	2	0	J(1,2,5). Порівнюються регістри 1 та 2 (не однакові), перехід до наступної команди – S(0)
2	2	2	0	S(0) – Збільшення на 1. R_0 був 1, став 2. Перехід до наступної команди – S(2)
3	2	2	1	S(2) – Збільшення на 1. R_2 був 0, став 1. Перехід до наступної команди – J(0,0,1)
4	2	2	1	J(0,0,1) – проходить порівняння нульового регістру R_0 , так як порівнюються однакові числа, то переходимо до команди №1 – J(1,2,5)
5	2	2	1	J(1,2,5). Порівнюються регістри 1 та 2 (не однакові), перехід до наступної команди – S(0)
6	3	2	1	S(0) - Збільшення на 1. R_0 був 2, став 3. Перехід до наступної команди – S(2)
7	3	2	2	S(2) – Збільшення на 1. R_2 був 1, став 2. Перехід до наступної команди – J(0,0,1)
8	3	2	2	J(0,0,1) – проходить порівняння нульового регістру R_0 , так як порівнюються однакові числа, то переходимо до команди №1 – J(1,2,5)
9	3	2	2	J(1,2,5). Порівнюються регістри 1 та 2 (однакові), перехід до команди q5 – її немає, отже програма завершена. Результат = 3. Розрахунки 1 + 2 = 3 – виявились вірними. Програма працює коректно.

Машини з натуральнозначними регістрами є еквівалентними машинам Тьюрінга в тому сенсі, що вони здатні вирішувати ті ж самі обчислювальні проблеми. Це означає, що будь-який алгоритм, який може бути виконаний на машині Тьюрінга, може бути також реалізований на машині з натуральнозначними регістрами та навпаки. Машини з натуральнозначними регістрами і машини Тьюрінга є дуже важливими формальними моделями обчислення, які допомагають вивчати можливості обчислювальності.

РОЗДІЛ 3. РОЗРОБКА ПРОЄКТУ

3.1 Мета проєкту, огляд використаних технологій та хід розробки проєкту

Мета створення електронного підручника з курсу "Теорія алгоритмів" полягає у забезпеченні студентів інформаційним ресурсом, який допоможе їм освоїти основні концепції, принципи та методи розв'язання алгоритмічних задач. Основні цілі проекту включають:

1. Забезпечення доступності: Мета полягає у створенні електронного підручника, який буде легко доступним для студентів з будь-якого місця, в будь-який час. Це забезпечить можливість вивчення теорії алгоритмів у зручний для студентів спосіб.
2. Чіткість та систематичність: Підручник повинен пояснювати складні концепції та алгоритмічні підходи зрозумілим способом, мати демонстративні компоненти в поясненні підходів.
3. Візуалізація та приклади: Підручник повинен включати візуальні засоби та ілюстрації для кращого розуміння складних алгоритмів. Візуалізація допоможе студентам уявити та запам'ятати ключові концепції. Крім того, підручник повинен містити приклади реальних задач і розв'язків для практичного застосування.

Етап підбору технологій для створення власного продукту проходив з урахуванням основної мети розробки – створення зручного веб-підручника. Обраною мовою програмування стала JavaScript. Було прийнято рішення обрати саме такий стек технологій як: HTML, CSS, React та Node.js. Основними причинами були:

- Універсальність та популярність: HTML та CSS є основними мовами розмітки та стилізації веб-сторінок, що дозволяє легко створювати зручний та привабливий інтерфейс для підручника. React, у свою чергу, є однією з найпопулярніших JavaScript-бібліотек для розробки веб-додатків, яка

надає потужні можливості для організації компонентної структури та управління станом додатка. Node.js дозволить нам використовувати JavaScript як мову програмування як на фронтенді, так і на бекенді, що робить його універсальним рішенням для повноцінної розробки підручника.

- Динамічний та інтерактивний інтерфейс: Використання React дозволяє нам створити динамічний та інтерактивний інтерфейс підручника. Ми зможемо легко реалізувати різноманітні функції, такі як пошук, навігація по розділам, можливість позначати важливі розділи або додавати власні примітки. Це допоможе студентам знаходити необхідну інформацію швидко та зручно.
- Масштабованість та розширюваність: Завдяки використанню Node.js, ми зможемо створити бекенд-сервер для нашого підручника, що дозволить нам зберігати і керувати контентом. Наведемо приклади: додавати нові розділи, оновлювати матеріали або взаємодіяти з користувачами через коментарі чи форум.

Слід зазначити, що React - бібліотека, а не повноцінний фреймворк, що дозволяє мати більший контроль над потоком програми порівнюючи з іншими фреймворками.

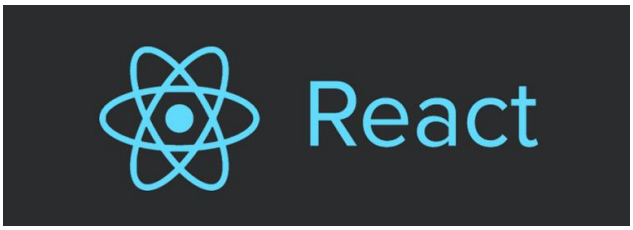

Рис.5 Логотип

Переваги:

- Швидкий розвиток
- Можливість переходу між версіями

- Синтаксис HTML-подібний
- Пришвидшення завдяки Virtual DOM

Недоліки:

- Великі розміри файлів

Також важливим аспектом розробки застосунку є зручність програміста, це залежить від багатьох факторів, але основним є середовище розробки, котре більш до вподоби розробнику. Обраним середовищем розробки став VSCode. VSCode розроблений Microsoft. Найбільш вагомим аргументом у виборі середовища на користь VSCode стала зручність в різних розширенях, налаштованість під кожного користувача персонально та легкістю використання.

Хід розробки мав насичений список етапів, котрі включали в себе основні:

- Вибір назви продукту

  Першим кроком було вибрати підходящу та актуальну назву для електронного підручника. Після проведення аналізу була обрана назва "Algorithmix". Ця назва відображала основну мету підручника, а саме - навчання алгоритмів.

- Створення Use-Case сценаріїв

  Для того, щоб краще зрозуміти, яким має бути кінцевий продукт і які функціональні можливості повинен мати підручник, було розроблено UseCase сценарії. Ці сценарії описували типові ситуації, в яких студенти або користувачі підручника можуть опинитися, та з якими завданнями або проблемами вони можуть зіткнутися. Це дало змогу зрозуміти краще реалізацію таких підпунктів як візуалізація алгоритмів та розподіл по темам.

- Розробка підручника

  Після аналізу UseCase сценаріїв розпочалася фактична розробка електронного підручника "Algorithmix".

3.2 Розробка та огляд реалізації

3.2.1 Розробка

Use-case сценарії:

1. **Історія**: Як користувач електронного підручника Algorithmix я хочу зареєструватись в е-підручник для подальшого користування

   Назва: Реєстрація в е-підручник

   Ціль: Реєстрація в е-підручник для користування

   Діюча особа: будь-які користувачі Algorithmix

   Попередні умови:

   - Користувач повинен мати доступ до інтернету

Успішний сценарій:

1. Користувач натискає кнопку «register»
2. Користувач має три поля для введення даних про аккаунт
   - Поле “name”
   - Поле “Email”
   - Поле “Password”

   в які вводить свій дані для подальшого створення аккаунту.

3. Користувач перевіряє точність введених даних та натискає кнопку “Log in to the account”

Неуспішний сценарій:

1. Користувач не має доступу до інтернету
   - Система не відповідає на запити і не завантажує е-підручник

Альтернативні шляхи:

Назва: користувач має створений аккаунт е-підручника

- Користувач на сторінці реєстрації “User account registration” має змогу повернутись на сторінку входу в систему
- Користувач натискає кнопку “log in” поруч з попереджувальним написом “If you have an account”

2. **Історія**: Як користувач електронного підручника Algorithmix я хочу увійти в систему

   Назва: Вхід в систему

   Ціль: Вхід в систему для подальшого користування

   Діюча особа: будь-які користувачі Algorithmix

   Попередні умови:

   - Користувач повинен мати доступ до інтернету

   Успішний сценарій:

   1. Користувач має 2 поля введення даних
      - Поле “Email”
      - Поле “Password”

      в які вводить свій дані для входу.

   2. Користувач перевіряє точність введених даних та натискає кнопку “Log in to the account”

   Неуспішний сценарій:

   1. Користувач не має доступу до інтернету
      - Система не відповідає на запити і не завантажує е-підручник
   2. Користувач ввів невірні пароль або логін
      - Система виводить повідомлення “Wrong login or password”

3. **Історія**: Як користувач електронного підручника Algorithmix я хочу швидко перейти на потрібний розділ з інформацією

   Назва: Навігація

   Ціль: Швидкий перехід між розділами

   Діюча особа: будь-які користувачі Algorithmix

   Попередні умови:

   - Користувач має бути зареєстрований в Algorithmix
   - Користувач повинен мати доступ до інтернету

   Успішний сценарій:

   1. Користувач ліворуч має навігаційну панель з усіма розділами підручника поділені на:
      - Основний розділ “Формальні моделі алгоритмів та алгоритмічно обчислювальних функцій” з підпунктами
      - Основний розділ “Канторові нумерації. Теза Чорча” з підпунктами
      - вправи для самоперевірки
      - розділ з візуалізацією роботи МНР
      - розділ з візуалізацією роботи Машини Тьюрінга, з яких може обрати потрібний.
   2. Користувач обирає розділ та натискає на кнопку переходу
   3. Основний екран е-підручника переміщає читача на сторінку з потрібною темою

   Неуспішний сценарій:

   1. Користувач не має доступу до інтернету
      - Система не відповідає на запити і не завантажує е-підручник
   2. Користувач має бути зареєстрований в Algorithmix
      - Система не відповідає на запити

4. **Історія**: Як користувач електронного підручника Algorithmix я хочу попрактикуватись у розділі Машина Тьюрінга та зробити деякі обчислення

   Назва: Робота з Машиною Тьюрінга

   Ціль: Зробити деякі обчислення

   Діюча особа: будь-які користувачі Algorithmix

   Попередні умови:

   - Користувач має бути зареєстрований в Algorithmix
   - Користувач повинен мати доступ до інтернету
   - Користувач має зайти у розділ “Машина Тьюрінга - приклад”

   Успішний сценарій:

   1. Користувач має можливість обирати операцію для обчислення.
   2. Користувач має змогу ввести в 2 поля числа та ініціалізувати їх на стрічці Машини Тьюрінга (кнопка “initialize”).
   3. Користувач має змогу покроково виконувати обраховування (кнопка “step”), почати обраховування програмою самостійно (кнопка “start”) , перезапустити обрахований вираз (кнопка “reset”)
   4. Користувач вводить, ініціалізує та обраховує вираз

   Неуспішний сценарій:

      1. Користувач має бути зареєстрований в Algorithmix
         - Система виводить повідомлення «Ви повинні зареєструватися в системі»
      2. Користувач не має доступу до інтернету
         - Система виводить повідомлення «Немає доступу до Інтернету. Перевірте інтернет підключення»
      3. Користувач має ініціалізувати вираз на стрічці Тьюрінга
         - Система не буде обраховувати вираз без ініціалізації

Огляд та пояснення коду

Розробка системи реєстрації:

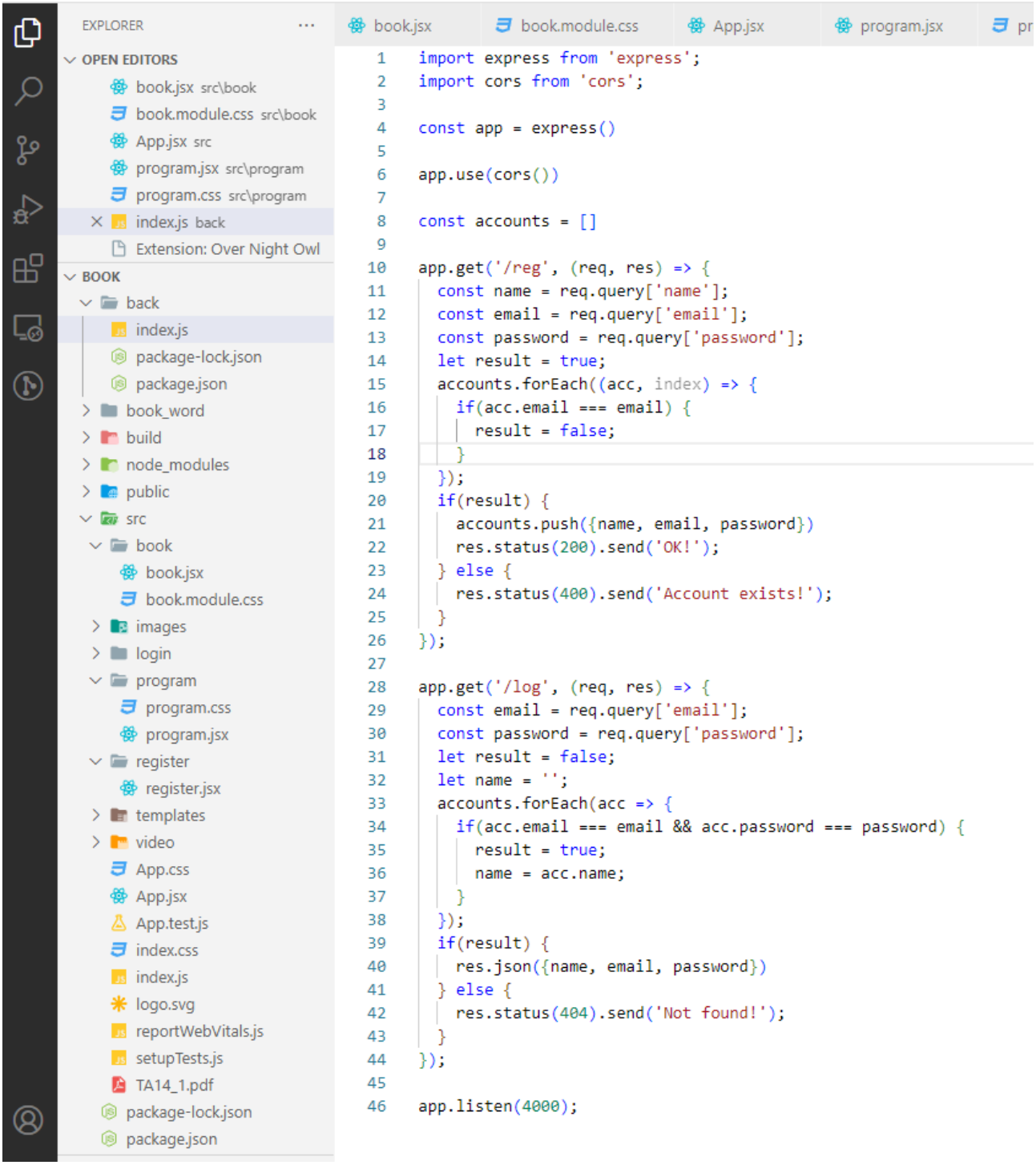

Рис.6 Фрагмент коду реєстрації

Оглянувши код, можна зрозуміти основні принципи створення реєстраційної функції підручника. Спочатку ми імпортуємо залежності. Потім

створюємо і налаштовуємо сервер, разом з цим оголошуємо масив для обліку облікових записів “accounts”, та обробляємо основні ендпоінти “/reg”, “/log”.

Окрім того, можемо побачити панель структурування проекту або іншими словами – організація папок і файлів для зручного управління розробки.

Опис компоненту Login для входу в систему:

```
import { NavLink } from 'react-router-dom';
import Descriptionlogin from '../templates/descriptionlogin';

const Login = () => {

  const log = (e) => {
    e.preventDefault()
    const email = document.querySelector('#emaill').value;
    const pass = document.querySelector('#pass').value;
    fetch('http://localhost:4000/log?email=' + email + '&password=' + pass)
      .then(data => {
        if (data.status === 200) {
          localStorage.setItem('user', data.json());
          window.location.href = 'http://localhost:3000/book'
        } else {
          alert('Wrong login or password!')
        }
      })
  }

  return (
    <div className="wrap-login">
      <Descriptionlogin />
      <div className="form_login">
        <form className="form_log" onSubmit={log}>
          <h3>Log in to your account</h3>
          <p>Email</p>
          <input type="email" id="emaill" placeholder="test@gmail.com" required />
          <p>Password</p>
          <input type="password" id="pass" placeholder="Password" required />
          <button>Log in to the account</button>
          <span>If you do not have an account -<NavLink to="./register">register</NavLink></span>
        </form>
      </div>
    </div>
  );
}

export default Login;
```

Рис.7 Фрагмент коду login

Тут описується сторінка входу в систему.

Переглянувши код можна побачити основну стрілкову функцію “Login”, функцію “log”. “Log” Const “email” та “pass” отримують значення поля відповідно зі своїми ідентифікаторами та зберігають їх в змінні.

book.jsx | book.module.css | App.jsx | login.jsx | register.jsx | program.jsx | program.css

```jsx
import '../App.css';
import Descriptionlogin from '../templates/descriptionlogin';


const Register = () => {

    const reg = (e) => {
        e.preventDefault()
        const name = document.querySelector('#namee').value;
        const email = document.querySelector('#emaill').value;
        const pass = document.querySelector('#pass').value;
        fetch('http://localhost:4000/reg?name='+name+'&email='+email+'&password='+pass)
            .then(data => {
                if(data.status === 200) {
                    alert('Account created!')
                    window.location.href = 'http://localhost:3000/'
                } else {
                    alert('Account exists!')
                }
            })
    }
    return (
        <div className="wrap-login">

            <Descriptionlogin />

            <div className="form_login">
                <form className="form_log" onSubmit={reg}>
                    <h3>User account registration</h3>
                    <p>Name</p>
                    <input type="text" id="namee" placeholder="Marry" />
                    <p>Email</p>
                    <input type="email" id="emaill" placeholder="test@gmail.com" required />
                    <p>Password</p>
                    <input type="password" id="pass" placeholder="Password" required />
                    <button>Log in to the account</button>
                    <span>If you have an account - <a href="./">log in</a></span>
                </form>
            </div>
        </div>
    )
}

export default Register;
```

Рис.8 Фрагмент коду register

Аналогічним принципом роботи можна визначити компонент “register”, котрий відповідає за реєстрацію.

3.2.2 Огляд реалізації алгоритмів.

Машина Тьюрінга

Для реалізації роботи МТ було вирішено використовувати мову JavaScript оскільки вона є потужною та широко використовуваною мовою програмування веб-розробки. Також я вирішив використати бібліотеку jQuery, яка є дуже популярною у веб-розробці. jQuery надає простий та зручний спосіб маніпулювання DOM-елементами, роботи з подіями і здійснення AJAX-запитів.

Використання jQuery спростило мені процес розробки, оскільки я міг швидко отримати доступ до елементів сторінки та виконувати з ними різні дії. Завдяки JavaScript і бібліотеці jQuery я зміг розробити алгоритм роботи Машини Тьюрінга зручно і ефективно. Вони дозволили мені створити функціональну інтерактивну модель Машини Тьюрінга, яка виконує обчислення на основі визначених правил та відображає результати на веб-сторінці.

Розбір основних принципів створеної МТ-програми:

```
var UniversalTuringMachine = function () {

    var me = this;
    me.headPosition = 0;
    me.state = "0";
    me.tape = [];
    me.stepCount = 0;
    me.stepInterval = 1000;

    me.successCallback = null;
    me.errorCallback = null;
    me.postStepCallback = null;
    me.startCallback = null;

    this.compare = function (a, b) {
        return a.toLowerCase() === b.toLowerCase();
    }

    this.moveLeft = function () {
        if (me.headPosition === 0) {
            me.tape.reverse().push('_');
            me.tape.reverse();
            // alert('not good');
        } else {
            me.headPosition--;
        }
    }

    this.moveRight = function () {
        me.headPosition++;

        if (!me.tape[me.headPosition])
            me.tape[me.headPosition] = BLANK;
    }

    this.preGo = function(input){
        me.tape = input.split('');
        if (me.startCallback) me.startCallback(me);

    }
```

Рис.9 Основний клас реалізованої Машини Тьюрінга

```
this.go = function (programm, cont) {
    (function loop() {
        setTimeout(function () {

            var ruleRan = false;
            var direction = "";

            for (var i = 0; i < programm.length; i++) {
                // Match actual me.state and actual value on me.tape
                if (me.compare(programm[i][1], me.tape[me.headPosition])
                && me.compare(programm[i][0], me.state)) {

                    me.tape[me.headPosition] = programm[i][2];

                    if (me.compare(programm[i][3], "R")){
                        me.moveRight();
                        direction = "R"
                    }
                    else if (me.compare(programm[i][3], "L")){
                        me.moveLeft();
                        direction = "L";
                    }

                    me.state = programm[i][4];
                    $('#lblState').text(me.state);

                    ruleRan = true;
                    me.stepCount++;

                    if (me.postStepCallback) me.postStepCallback(me, direction, programm[i][2]/* value to write */, programm[i]);
                    break;
                }
            }

            if (ruleRan) {
                if (cont === true) {
                    loop();
                }
            } else {
                if (me.compare(me.state, "END")) {
                    if (me.successCallback) me.successCallback();
                } else {
                    if (me.errorCallback) me.errorCallback();
                }
            }
        }, me.stepInterval)
    })();
}
}
```

Рис.10 Основний клас реалізованої Машини Тьюрінга

Основний клас UniversalTuringMachine оголошує різні властивості та методи для машини Тьюрінга. Основні властивості цього класу включають:

- headPosition: поточне положення головки на стрічці машини Тьюрінга.
- state: поточний стан машини Тьюрінга.
- tape: стрічка, яка служить як робоча пам'ять машини Тьюрінга.
- stepCount: кількість кроків, виконаних машиною Тьюрінга.
- stepInterval: інтервал часу (у мілісекундах) між кроками виконання машини Тьюрінга.

Основні методи цього класу включають:

- compare(a, b): функція, яка порівнює два рядки a і b без урахування регістру і повертає true, якщо вони ідентичні.
- moveLeft(): метод, який зсуває головку машини Тьюрінга вліво на одну позицію на стрічці.
- moveRight(): метод, який зсуває головку машини Тьюрінга вправо на одну позицію на стрічці.
- preGo(input): метод, який підготовлює машину Тьюрінга до виконання, розбиваючи вхідний рядок input на окремі символи та викликаючи початковий зворотний виклик (startCallback), якщо він визначений.
- go(programm, cont): метод, який починає виконання машини Тьюрінга з заданою програмою programm. Виконання програми відбувається у циклі, де машина Тьюрінга порівнює поточний стан me.state та поточний символ на стрічці me.tape[me.headPosition] з правилами програми programm. Якщо знайдено відповідне правило, машина Тьюрінга змінює символ на стрічці me.tape, переміщує головку у відповідному напрямку (R - праворуч, L - ліворуч) та змінює поточний стан me.state.
- Цикл виконання програми реалізований за допомогою рекурсивної функції loop(), яка викликається з певним інтервалом часу, встановленим значенням me.stepInterval. Це дозволяє візуалізувати кроки виконання машини Тьюрінга з певною затримкою.
- Після виконання кожного кроку, викликається зворотний виклик postStepCallback, якщо він визначений, передаючи поточний стан машини, напрямок руху (direction), символ, який був записаний на стрічку та саме правило, яке було застосовано.
- Якщо знайдено відповідне правило та виконано крок, то перевіряється умова cont === true, яка вказує, чи потрібно продовжувати виконання програми. Якщо cont === true, викликається знову функція loop() для

виконання наступного кроку. Це дозволяє виконувати програму по одному кроці за раз.

- Якщо жодне правило не підходить і поточний стан me.state не дорівнює "END", тоді викликається зворотний виклик errorCallback, якщо він визначений.

```
var Tools = function (turing) {
    var me = this;
    me.tm = turing;

    this.formatTape = function () {
        var result = "";

        for (var i = 0; i < me.tm.tape.length; i++) {
            var item = me.tm.tape[i];
            var lspacer = "";
            var rspacer = "";
            var itemSpacer = " ";

            if (i === me.tm.headPosition) {
                lspacer = "[";
                rspacer = "]";
            } else {
                lspacer = " ";
                rspacer = " ";
            }

            result = (result + lspacer + item + rspacer + itemSpacer);
        }
        return result;
    }

    this.logTape = function (text) {
        $('#outputField').prepend('<div>' + me.formatTape() + '</div>');
    }
}

Tools.init = function () {
    $('#outputField').empty();
    $('SPAN#status').text('idle');
    $('SPAN#stepCount').text('0');
    $('#tape').val('');
    $('#programm').text('');
    resetFancyTape();
}
```

Рис.11 Приклад коду взаємодії з інтерфейсом користувача

Функція Tools створює об'єкт, який містить різні інструменти, пов'язані з машиною Тьюрінга. Цей об'єкт отримує посилання на об'єкт машини Тьюрінга як параметр turing, щоб мати доступ до його властивостей та методів. У цьому об'єкті визначено два методи:

- formatTape(): метод, який форматує стрічку машини Тьюрінга для відображення в інтерфейсі користувача. Він проходиться по кожному символу на стрічці та додає відповідні теги HTML для позначення поточної позиції головки. Потім повертає отриманий рядок.
- logTape(text): метод, який додає відформатовану стрічку до вихідного поля в інтерфейсі користувача. Використовується jQuery для вставлення HTML-коду у відповідний елемент.

Функція Tools.init(): метод, який виконує початкову ініціалізацію інтерфейсу користувача. Вона очищає вихідне поле, встановлює деякі текстові значення та скидає декілька елементів у інтерфейсі.

Функція initializeUi()

- Змінні operation та tapeString отримують значення вибраної операції та введеної стрічки в інтерфейсі відповідно.
- Змінна tapeStringLength отримує довжину введеної стрічки.
- Змінна offset обчислює різницю між довжиною стрічки та очікуваною довжиною фантазійної стрічки (16 символів). Вона використовується для вирівнювання стрічки на фантазійній стрічці.
- Елементи списку ul#fancyTape в інтерфейсі, які необхідно змінити, видаляються або приховуються за допомогою методу remove() або addClass('hidden').
- Виконується цикл для побудови фантазійної стрічки. Кожен символ з введеної стрічки додається до списку ul#fancyTape у відповідному порядку. Перший символ поміщається в елемент з класом active, який вказує поточну позицію головки на стрічці. Якщо довжина стрічки більше 15 символів, то зайві символи після 16-го символу приховуються.

Далі йдуть звичайні обробники подій, для ініціалізації, кнопки покрокового виконання алгоритму, кнопки скидування наданої інформації.

```
$('#btnInitialize').click(function (e) {
    e.preventDefault();

    if (!isValid())
        return;

    // Reset all inputs
    $('#outputField').empty();
    $('SPAN#status').text('idle');
    $('SPAN#stepCount').text('0');
    $('#tape').val('');
    $('#programm').text('');
    resetFancyTape();

    var operation = $('input[name="programRadio"]:checked').val();
    var firstVal = parseInt($('#txtFirstVal').val());
    var secondVal = operation == 'faculty' ? 0 : parseInt($('#txtSecondVal').val());
    var tapeString = '';
    for (var i = 0; i < firstVal; i++) {
        tapeString += '0';
    }
    tapeString += '1';
    for (var i = 0; i < secondVal; i++) {
        tapeString += '0';
    }

    if (operation != 'faculty')
        tapeString += '1';

    $('#tape').val(tapeString);

    initializeUi();
});
```

Рис.12 Обробники подій

```
$('#btnStep').click(function (e) {
    if (_sl == false) {
        if (!isValid())
            return;

        _tm = new UniversalTuringMachine();
        _t = new Tools(_tm);

        var tape = $('#tape').val();
        var simulator = eval($('#programm').val());

        $(document).off('speedchange');
        $(document).on('speedchange', function (e, speed) {
            _tm.stepInterval = speed;
            $('#lblSpeed').text(speed);
        });

        _tm.startCallback = function () {
            _t.logTape();
            $('SPAN#status').text('running');
        }

        _tm.successCallback = function () {
            $('SPAN#status').text('success');
        };

        _tm.errorCallback = function () {
            $('SPAN#status').text('error!');
        };

        _tm.postStepCallback = function (me, direction, toWrite, rule) {
            _t.logTape();
            $('SPAN#stepCount').text(me.stepCount);
            $('#tape').val(me.tape.join(''));
            animateStep(direction, toWrite);
            _diagram.highlight(rule);
        };

        // Run
        _tm.preGo(tape);
        _tm.go(simulator, false);
        _sl = true;
    } else {
        var simulator = eval($('#programm').val());
        _tm.go(simulator, false);
    }

});
```

Рис.13 Обробники подій

МНР-програма

Рішенням, для створення візуалізації роботи МНР-програми стало створення емулятора роботи МНР-програми на мові C#

Основними фрагментами коду слід виділити клас CommandList, котрий включає в себе такі властивості:

```
1 reference
private CommandList()
{
    _commands = new List<Command>();
    _reverse = new Stack<ReverseCommand>();
}

7 references
public static CommandList Instance { get; } = _instance ??= new CommandList();

2 references
public void Add(params Command[] commands) => _commands.AddRange(commands);

0 references
public void Insert(int index, string rawCommand)
{
    var newCommand = FileManager.ParseCommand(index, rawCommand);
    _commands.Insert(index, newCommand);
    for (var i = index + 1; i < _commands.Count; ++i)
        ++_commands[i].Number;
}

1 reference
public void RemoveAt(int index)
{
    _commands.RemoveAt(index);
    for (var i = index; i < _commands.Count; ++i)
        --_commands[i].Number;
}

2 references
public void Swap(int index1, int index2)
{
    if (index1 == index2) return;

    var command = _commands[index1];
    _commands[index1] = _commands[index2];
    _commands[index2] = command;

    var number = _commands[index1].Number;
    _commands[index1].Number = _commands[index2].Number;
    _commands[index2].Number = number;
}
```

Рис.14 Клас CommandList

- Конструктор класу CommandList є приватним і ініціалізує поля _commands і _reverse новими порожніми об'єктами.
- Метод **Add** додає одну або декілька команд до списку _commands.
- Метод **Insert** додає нову команду до списку _commands на позицію з заданим індексом. Він також змінює номера інших команд, що знаходяться після вставленої команди.

- Метод **RemoveAt** видаляє команду зі списку _commands за заданим індексом.

Саме ці фрагменти коду відповідають за додавання команд МНР-програми, для визначення арифметичної операції для чисел.

3.3 Огляд створеного електронного підручника

Початком використання електронного підручника є сторінка реєстрації особистого акаунта за допомогою електронної пошти.

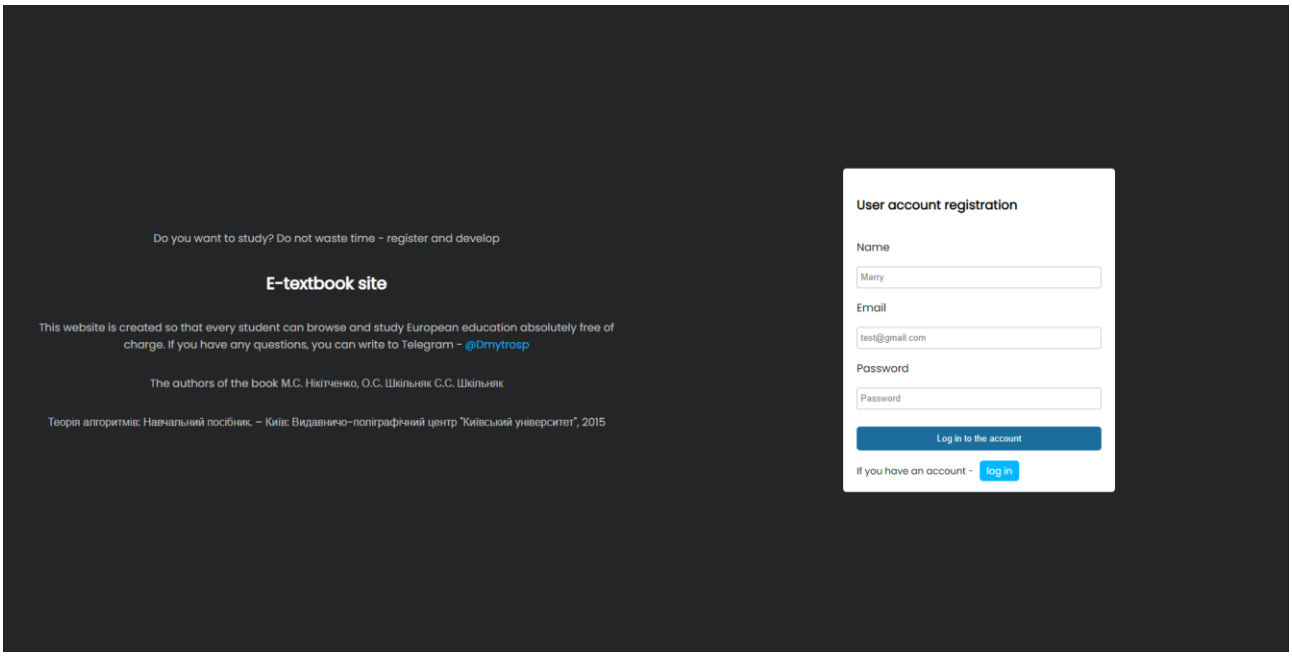

Рис.15 Сторінка реєстрації

Разом з полем для введення особистих даних є інформація про розробника електронної версії підручника та авторів цього посібника.

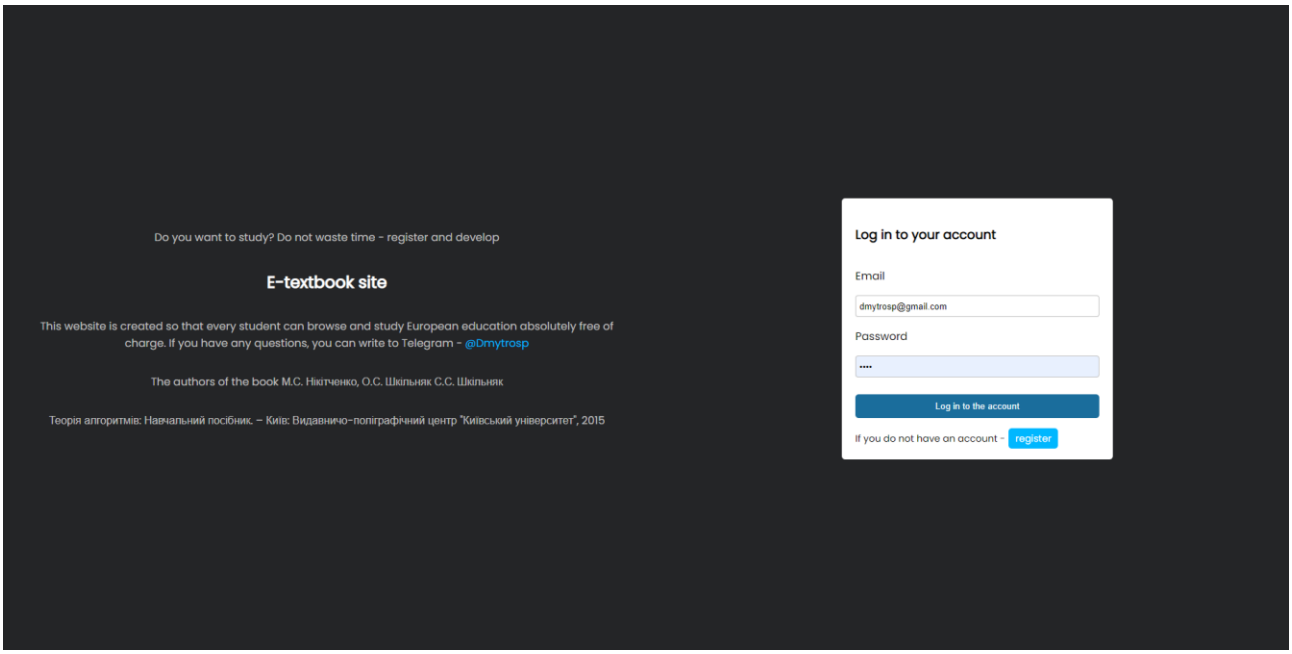

Рис.16 Вхід в акаунт

Зайшовши в електронний підручник ми бачимо титульну сторінку підручника, панель навігації по розділам, та зручну навігацію сторінок з поверненням на головну сторінку.

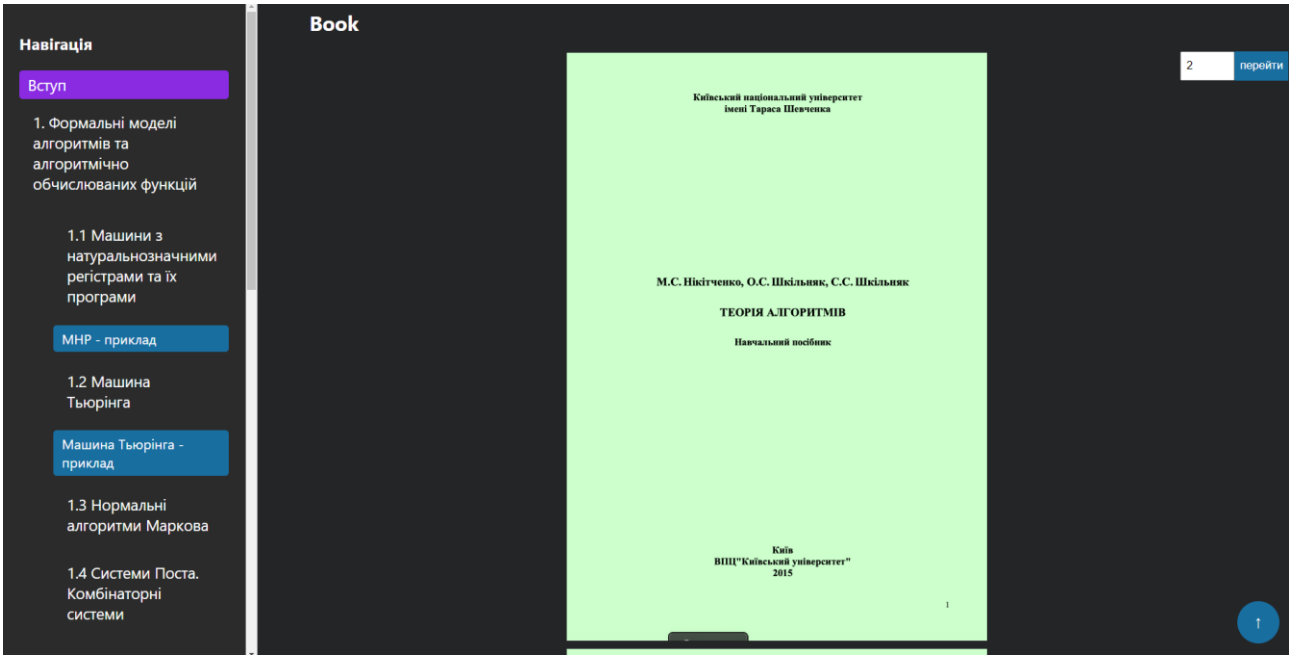

Рис.17 Головна сторінка

Зручний формат навігації по темах підручника дає змогу швидко перенаправляти користувача на вказану тему, просто натиснувши на її назву в панелі навігації ліворуч. Прикладом в рис.18 слугує перехід на тему Машини з натуральнозначними регістрами. При переході на тему, що вас цікавить, панель навігації залишається доступною для використання.

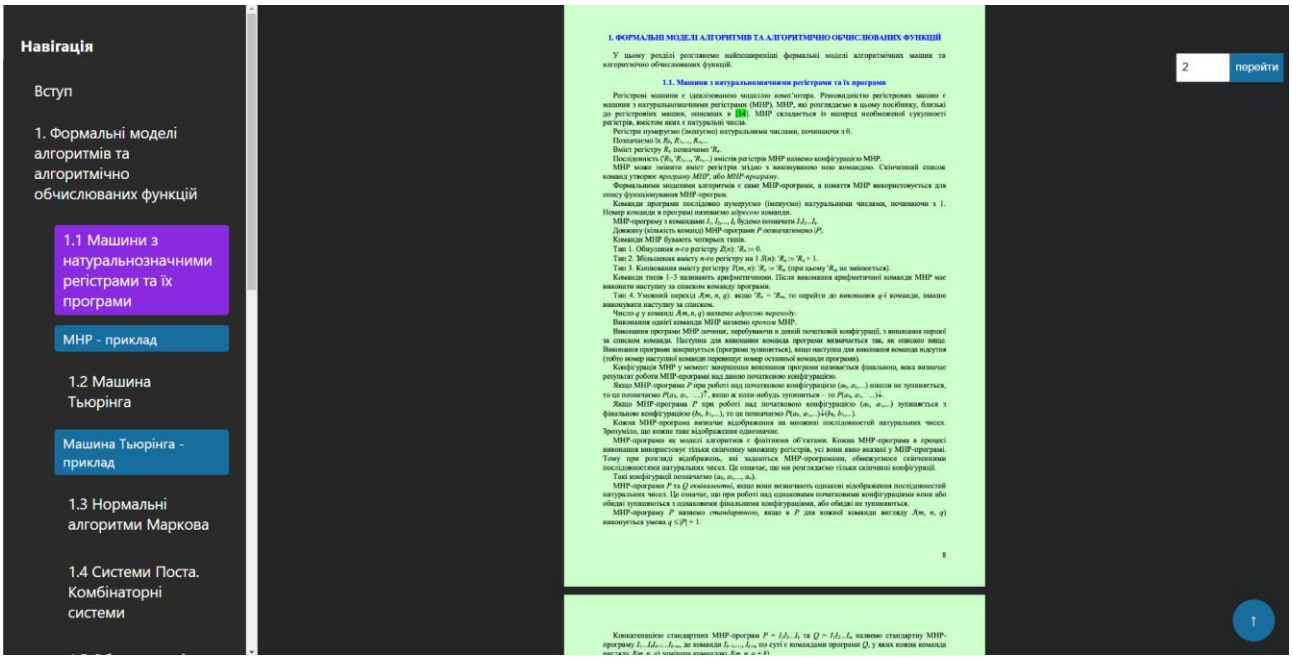

Рис.18 Реалізація навігації по темам підручника

Для зручності користування було вирішено додати поле для введення сторінок підручника разом з кнопкою повернення на титульну сторінку. Прикладом в рис.19 слугує перехід на сторінку 10 підручника.

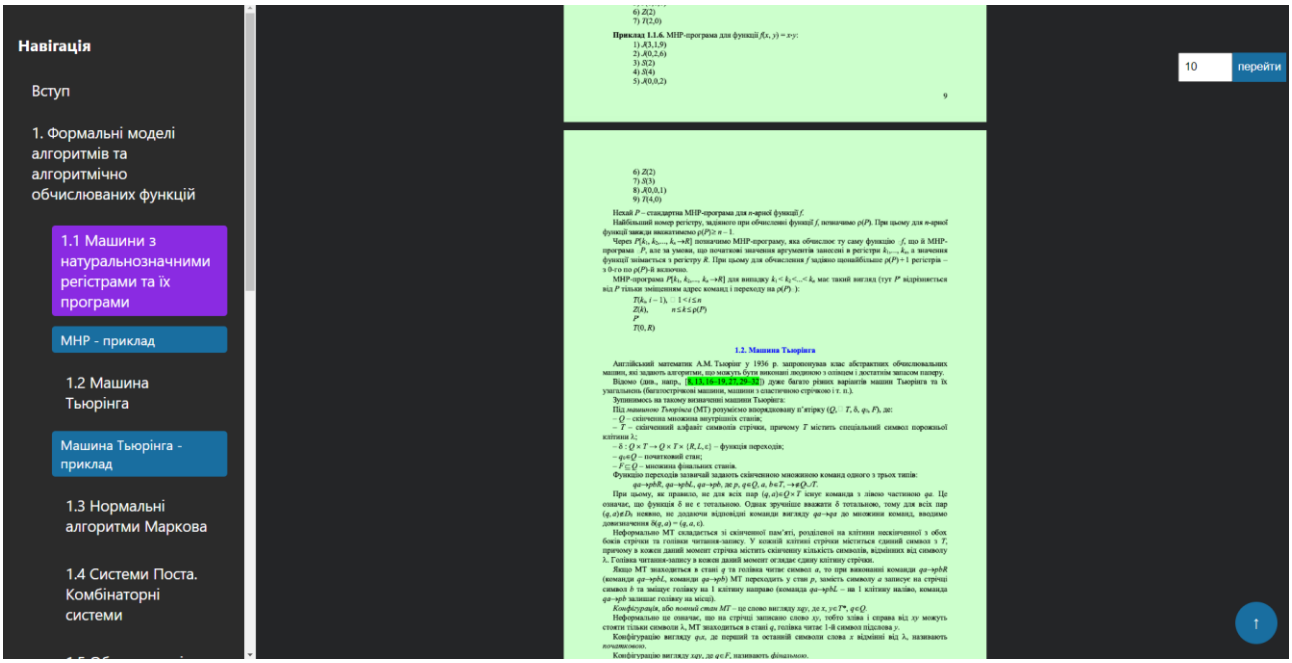

Рис.19 Перехід по сторінкам

3.4 Огляд ілюстрації роботи машини Тьюрінга

В дизайні сайту також створено контраст кольорів кнопок для деяких окремих тем в навігації, котрі мають назву «МНР – ілюстрація» та «Машина Тьюрінга – ілюстрація». Це зроблено з ціллю привернути увагу на окремі розділи з демонстрацією роботи моделей алгоритмів.

На рисунку 20 можемо побачити головний екран машини Тьюрінга. В наявності є головна стрічка з комірками для введених даних, поля для введення аргументів, консольне віконце з виведенням кожного відпрацьованого етапу обчислення стрічки та кнопки для початку підрахунків програми.

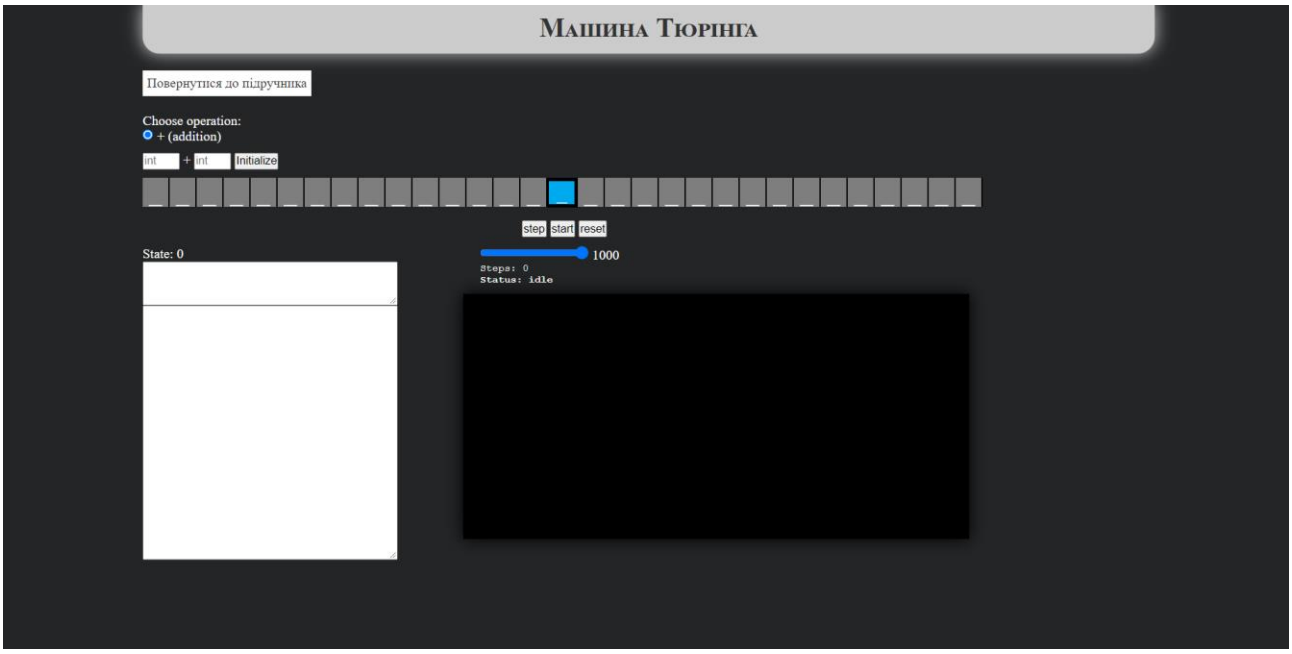

Рис.20 Сторінка з ілюстрацією роботи машини Тьюрінга

3.5 Огляд ілюстрації роботи МНР програми

Рис.9 та 10 показують нам головну сторінку розділу «МНР – приклад» з інформацією про саму МНР програму для функції (а нашому випадку це приклад додавання), аргументи функцій, які підраховуються в відео, саме відео роботи емулятора машини з натуральнозначними регістрами, та команди МНР-програми для функції додавання.

Функції, які ми підраховували за допомогою програми:

$f(2,3) = 2+3$, $f(4,5) = 4+5$.

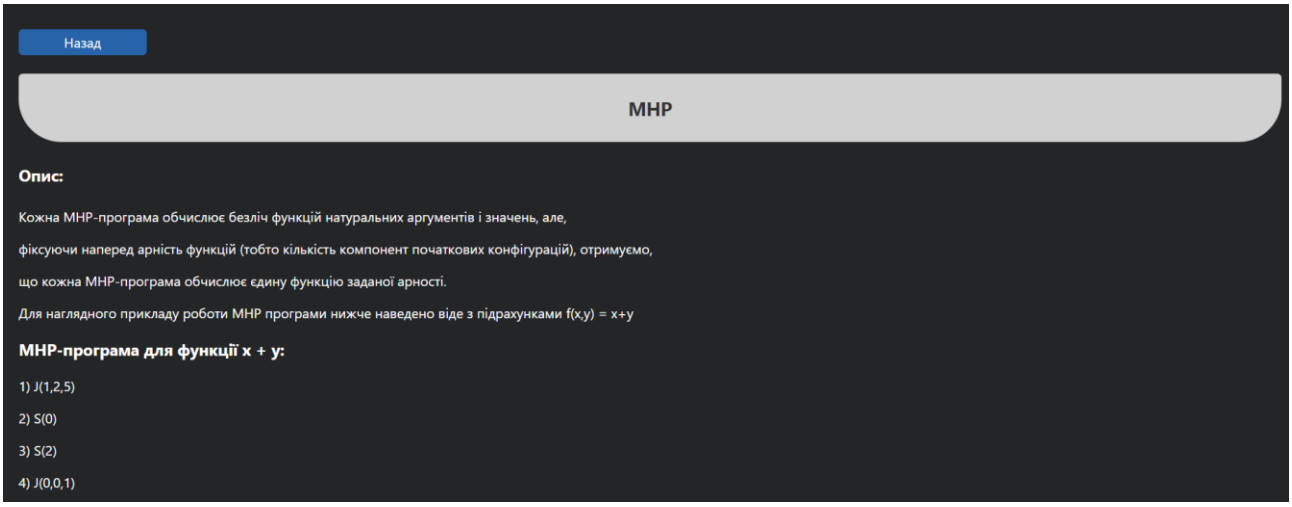

Рис.21 Верхня частина розділу МНР – приклад з інформацією

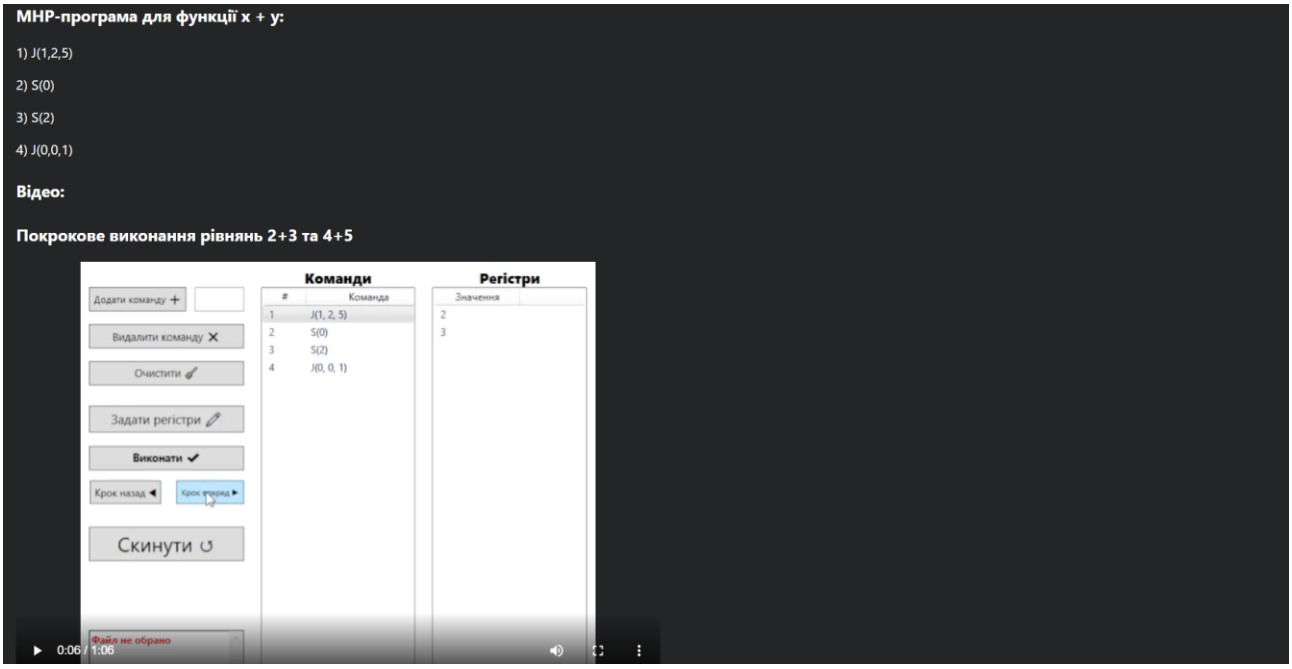

Рис.22 Відео роботи емулятора з підрахунком 2 функцій

Додаткова демонстрація самого емулятора МНР-програми:

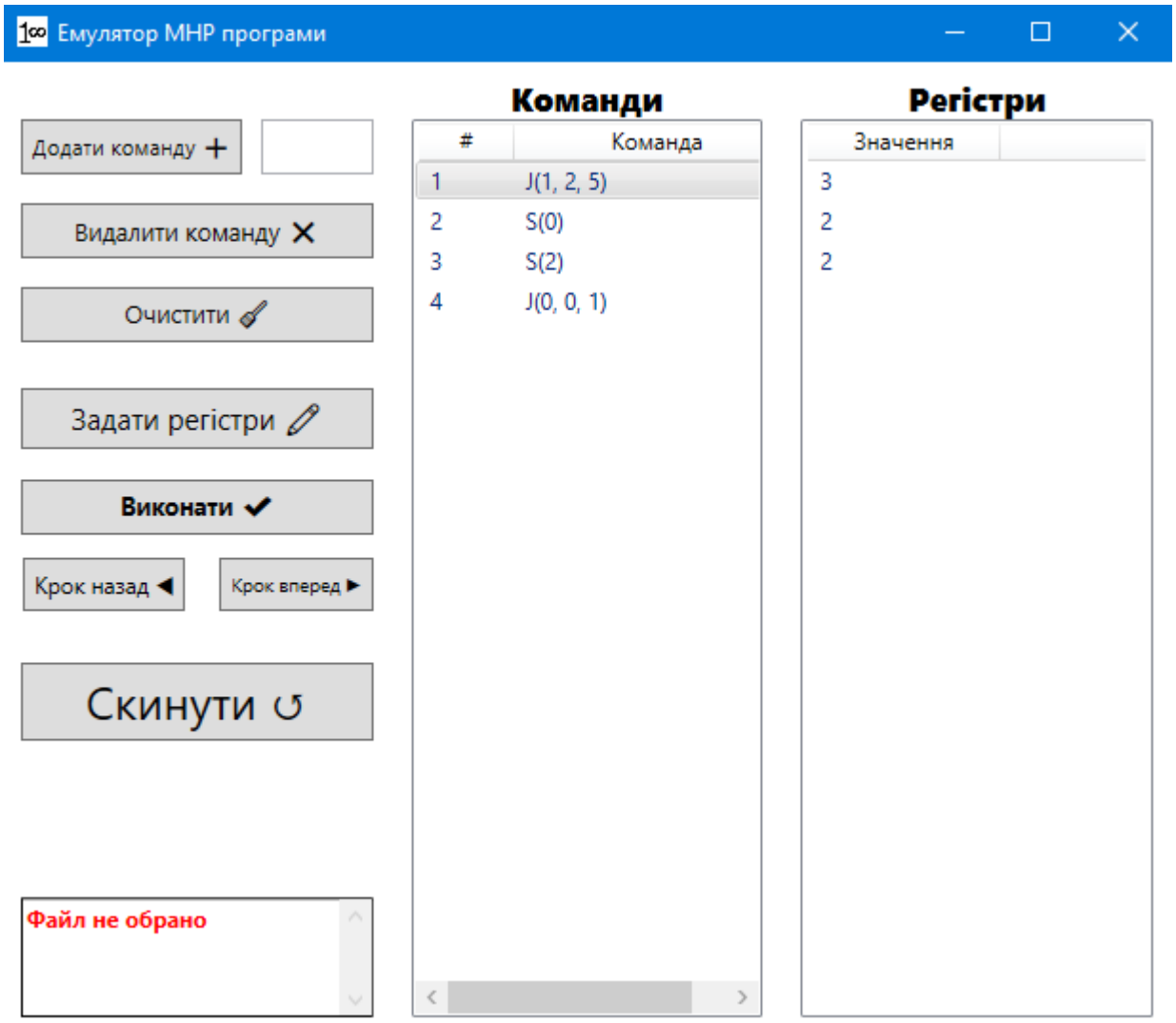

Рис.23 Емулятор МНР-програми

До основного функціоналу МНР-програми належить:

- “Додати команду” – додавання команд МНР-програми для визначення арифметичних операцій
- “Задати регістри” – задавання чисел для обчислення
- “Крок назад”, ”Крок вперед” – покрокове обчислення регістрів

ВИСНОВКИ

Під час виконання кваліфікаційної роботи мною було розроблено електронний підручник під назвою “Algorithmix”.

Основною метою роботи було створення чіткого, доступного та інтерактивного підручника, який надає інформацію про теорію алгоритмів широкому колу користувачів.

На початку роботи було проведено аналіз існуючих електронних підручників з тематики "Теорія алгоритмів" для вивчення підходів та особливостей їх створення. Використовуючи ці знання, було вибрано найбільш зручні для створення сервісу технології, зокрема React js, node js, HTML та CSS.

Наступним кроком було формулювання функціональних і нефункціональних вимог до системи. Це надало можливість більш чітко розуміти кінцевий вигляд продукту разом з його особливостями.

Результатом роботи є створений сервіс-електронний підручник "Algorithmix", який містить чітку структуру та логічний порядок подання матеріалу. Підручник включає різні методи надання інформації користувачу, зокрема ілюстрації та реалізацію алгоритмів для їх візуалізації.

У процесі розробки електронного підручника "Algorithmix" було отримано практичні вміння по створенню WEB-додатків та роботі з алгоритмами.

Таким чином, створений сервіс-електронний підручник "Algorithmix" відкриває можливості для дистанційного навчання студентів та надає зручний інструмент для вивчення теорії алгоритмів. Завдяки проробленій роботі по створенню продукту, мною було здобуто належні знання щодо розробки таких сервісів, отримано корисний практичний досвід, а також здобуто нові знання з теорії алгоритмів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Перша електронна книга [Електронний ресурс] // tribuna.com.ua. – 2019. – Режим доступу до ресурсу: https://tribuna.com.ua/576-persha-elektronna-kniga-rozvitok-tekhnologiyi-elektronnikh-knig.html#i (дата звернення: 20.05.2023).
2. Шлапак Ю. Електронний навчальний посібник як інноваційний вид програмно-педагогічних засобів / Ю. Шлапак // Наукові праці Національної бібліотеки України ім. В. І. Вернадського. - 2014. - Вип. 39. - С. 278-288. - Режим доступу: http://nbuv.gov.ua/UJRN/npnbuimviv_2014_39_25 (дата звернення: 20.05.2023).
3. Коміренко Г.Б. Електронний підручник – принципи та технологія створення [Електронний ресурс] / Галина Борисівна Коміренко – Режим доступу до ресурсу: http://nvd.luguniv.edu.ua/archiv/NN5/08kgbtts.pdf. (дата звернення: 20.05.2023).
4. Гризун Л.Е. Дидактичні основи створення сучасного комп'ютерного підручника : автореф. дис. на здобуття наук. ступеня канд. пед. наук: 13.00.09 / Харківський державний. пед. ун-т ім. Г.Сковороди.– Харків, 2002. – 13 с.
5. Степанов О.М. Основи психології і педагогіки / О.М. Степанов, М.М. Фіцула. – Київ: Київ Академвидав, 2006. – 265 с.
6. Стромило І. Технології та методологія розробки електронних посібників / І. Стромило // Нова педагогічна думка. - 2013. - № 2. - С. 182-185. - Режим доступу: http://nbuv.gov.ua/UJRN/Npd_2013_2_47 (дата звернення: 20.05.2023).
7. Нікітченко М.С. Теорія алгоритмів / М.С. Нікітченко, О.С. Шкільняк, С.С. Шкільняк. – К.: ВПЦ Київський університет, 2015.
8. Нікітченко М.С. Математична логіка та теорія алгоритмів. Підручник / М. С. Нікітченко, С. С. Шкільняк. – К.: ВПЦ Київський університет, 2008.

9. Шкільняк С.С. Теорія алгоритмів. Приклади й задачі / С. С. Шкільняк. – К.: ВПЦ Київський університет, 2012.
10. Turing Machine [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: *https://mathworld.wolfram.com/TuringMachine.html*. (дата звернення: 20.05.2023).