Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА АЛГОРИТМУ ОЦІНКИ ІНКЛЮЗИВНОСТІ МОБІЛЬНИХ ЗАСТОСУНКІВ

Текстова частина до кваліфікаційної випускної роботи

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник кваліфікаційної роботи,
доцент Афонін А.О.
(прізвище та ініціали)

_____
(підпис)
“_____”_____2023 р.

Виконала студентка ІПЗ-4
Гопцій Д.Р.
(прізвище та ініціали)
“_____”_____2023 р.

Київ 2023
Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Доцент
Афонін А.О.
(прізвище та ініціали)

(підпис)

“_____” _____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну випускну роботу

студентці Гопцій Дар’ї Романівни факультету інформатики 4-гокурсу
ТЕМА Розробка алгоритму оцінки інклюзивності мобільних застосунків

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання
Календарний план виконання роботи
Анотації
Вступ
1. Огляд стандартів та інструментів тестування
2. Розробка алгоритму тестування
3. Опис практичної частини та аналіз результатів
Висновки
Список літератури
Додатки

Дата видачі “_____” _____2023 р. Афонін А.О. _____
(підпис)

Завдання отримала _____
(підпис)

Тема: Розробка алгоритму оцінки інклюзивності мобільних застосунків

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми кваліфікаційної роботи	30.09.22	
2.	Вивчення аналогів інструментів тестування	31.10.22	
3.	Пошук літератури щодо стандартів тестування	30.11.22	
4.	Ознайомлення з літературою	31.12.22	
5.	Створення плану підпунктів алгоритму	31.01.23	
6.	Створення алгоритму та його опису в теоретичних розділах відповідно до підпунктів плану	15.03.23	
7.	Створення прикладу коду алгоритму	31.03.23	
8.	Огляд та аналіз результатів прикладу тестування	15.04.23	
9.	Створення презентації	30.04.23	
11.	Перегляд змісту роботи керівником та попередній захист	10.05.23	
12.	Внесення змін до роботи відповідно дозауважень наукового керівника	20.05.23	
13.	Захист кваліфікаційної роботи	01.06.23	

Студентка Гопцій Д.Р.
Керівник Афонін А.О.
“_____”_____ 2023 р.

ЗМІСТ

Анотація

В роботі представлено алгоритм тестування та аналізу інклюзивності мобільних застосунків, що спирається на стандарти WCAG 2.1 та загальні рекомендації для розробників для операційних систем iOS та Android. Було розглянуто наступні критерії оцінювання: присутність та правильне заповнення описових атрибутів, підтримка вбудованих програм-зчитувачів екрану, підтримка роботи фізичної клавіатури, рівень контрастності елементів вікна, присутність небезпечних спалахів. До кожної підтеми створено покроковий алгоритм тестування у вигляді блок-схем діаграм, приклад імплементації мовою програмування Java та приклади тестових випадків з очікуваним результатом.

Ключові слова: мобільна розробка, тестування, інклюзивність, елементи UI.

Вступ

В 2020 році в Україні було зареєстровано 2,7 мільйони громадян з різними формами інвалідності [1]. На жаль, станом на 2023 рік ця цифра значно зростає через отримані травми під час воєнного стану.

Незалежно від стану здоров’я людини, користування мобільними приладами стало невід’ємною частиною її життя і може значно полегшити виконання повсякденних задач. Проте, інтерфейс мобільних застосунків не завжди є коректно адаптованим до потреб людей з вадами зору, моторики рук чи епілептичними розладами. Саме тому в сьогочасній розробці додатків питання забезпечення належної доступності для всіх користувачів є дуже актуальним.

Метою даної роботи є створення загального алгоритму перевірки елементів інтерфейсу мобільних застосунків на відповідність задовільному рівню доступності для користувачів з різними формами інвалідності.

Об’єктом дослідження є вивчення наявних стандартів інклюзивності програмного забезпечення та його тестування на відповідність ним.

Робота складається з трьох розділів.

В першому розділі розглянуто наявні стандарти та рекомендації щодо інклюзивності програмного забезпечення. Також описано інструменти тестування, що вже присутні на ринку, їх переваги та недоліки.

Спираючись на це дослідження, в другому розділі розглядаються виокремлені стандарти, що можуть бути адаптовані для потреб мобільної розробки. Описується покроковий алгоритм тестування елементів вікна інтерфейсу, що спирається на них, як на критерії оцінки.

В третьому розділі наведено практичний приклад імплементації алгоритму мовою Java та розглянуто результати прикладу тестування вікон мобільних інтерфейсів.

Джерелами дослідження є загальноприйняті стандарти інклюзивності веб-застосунків запропоновані WAI та окремі рекомендації для розробників операційних систем iOS та Android.

Теоретичний аналіз, проведений в цій роботі, узагальнює наявні стандарти та методи забезпечення інклюзивності за їх можливістю використання в мобільній розробці, тому може бути використано для подальших наукових досліджень в цій сфері.

Практичне значення одержаних результатів полягає в тому, що алгоритм є більшою мірою абстрагованим від платформи мобільної розробки. Це дозволяє його подальшу імплементацію в межах будь-якої операційної системи.

РОЗДІЛ 1: Стандарти та інструменти тестування інклюзивності мобільних застосунків

1.1 Огляд наявних стандартів тестування

На теперішній час існують наступні загальноприйняті підбірки вказівок щодо покращення інклюзивності користувацьких інтерфейсів, запропоновані Web Accessibility International [49]. Вони являють собою набір рекомендацій щодо того, як зробити веб-вміст більш доступним, насамперед для людей з обмеженими можливостями:

а) Web Content Accessibility Guidelines (WCAG) – стосуються безпосередньо інформації, відображеної на веб-сторінках включно з текстом, елементами мультимедіа, розміткою тощо [48];

б) Authoring Tool Accessibility Guidelines (ATAG) – стосуються програм, що використовують розробники для створення веб-вмісту [6];

в) User Agent Accessibility Guidelines (UAAG) – стосуються розробки браузерів, розширення браузерів, медіаплеєрів, програм для читання та інші програми, які відтворюють веб-вміст [47].

Найбільш детальним переліком вказівок є WCAG. З моменту його створення було видано декілька оновлених версій. Останньою офіційно прийнятою і найбільш детальною з них є WCAG 2.1 2018 року. Багато вказівок з ATAG та UAAG повторюють або посилаються на стандарти WCAG, оскільки також пов’язані з відображенням або створенням вмісту веб-сторінок.

Оскільки всі перелічені підбірки вказівок стосуються веб-сторінок, в них відсутні деталі принципів розробки мобільних застосунків [19, с. 3]. В 2015 році WAI випустили перший варіант переліку вказівок для мобільних

застосунків, створений робочою групою з питань мобільної доступності Mobile Accessibility Task Force [46]. Проте, цей документ є скоріше визнаною приміткою до стандартів, що вже існують, і не набув окремого офіційного статусу.

Створення одного загальноприйнятого переліку стандартів для мобільних застосунків ускладнене відмінностями мобільних операційних систем. В той час як всі веб-сторінки створені за допомогою стандартизованої мови розмітки документів HTML та спеціальної мови стилю CSS, і відповідно мають спільний перелік елементів та їх атрибутів, структура елементів інтерфейсу мобільних застосунків та їх взаємодія відрізняється залежно від платформи розробки. Присутні також відмінності у взаємодії користувача з мобільним додатком та веб-сторінкою, наприклад в середньому менший розмір екрану, на якому переглядається вміст, використання сенсорного екрану замість клавіатури та миші тощо.

Однак, стандарти WCAG є узагальненими і стосуються кінцевого вигляду продукту, а не специфіки елементів, з яких створено інтерфейс. Оскільки цей перелік стандартів є наразі найбільш вичерпним, його можливо адаптувати для потреб розробників та користувачів мобільних застосунків. Тому загальні рекомендації для розробників iOS та Android щодо створення більш інклюзивного дизайну застосунків частково перекликаються з WCAG. У свою чергу ці рекомендації більш детально враховують особливості конкретних операційних систем [1; 8].

У цій роботі розглядатимуться як рекомендації для конкретних операційних систем, так і узагальнені стандарти створені WAI, які можуть бути використані в розробці мобільних застосунків. Алгоритм аналізу застосунку на відповідність цим стандартам є абстрагованим від конкретної операційної системи, проте розглядатимуться й певні приклади відмінностей

між ними. Це дозволить розробникам в майбутньому адаптувати алгоритм для потреб їх платформи розробки.

1.2 Огляд наявних інструментів тестування

Як і будь-яке тестування програмного забезпечення, тестування мобільних застосунків має відбуватися впродовж всього циклу розробки. Загалом тестування на відповідність стандартам інклюзивності можна розділити на три головні типи: автоматизоване, інструменти інспекції та мануальне.

Автоматизоване тестування відбувається до випуску програмного забезпечення. Перевагами інструментів такого тестування є можливість інтеграції їх в CI/CD конвеєри [50, с. 216].

Відмінністю інструментів інспекції від автоматизованого тестування є неможливість їх повноцінного використання без участі розробника. Вони можуть надавати більш детальну інформацію щодо застосунку, проте потребують мануального запуску і розгляду перевірок [50, с. 220]. Окрім того, їх не обов’язково використовуються виключно під час розробки застосунку. Деякі надають можливість провести його аналіз вже після встановлення на мобільний пристрій.

Мануальне тестування включає в себе ручну перевірку роботи застосунку та відповідності тим стандартам інклюзивності, що не можуть бути перевірені за допомогою автоматизованих алгоритмів. Вони застосовуються для оцінки елементів, характеристики яких залежать від вмісту вікна, контексту та суб’єктивного сприйняття.

Наведені в Додаток А Огляд інструментів тестування інклюзивності таблиці Таблиця А.1 Порівняння інструментів тестування інклюзивності частина 1 та Таблиця А.2 Порівняння

інструментів тестування інклюзивності частина 2містять приклади інструментів автоматизованого тестування та інспекції і за якими параметри вони здійснюють оцінку застосунку. Інформація зібрана за наявними у вільному доступі описами, документаціями або презентацією демоверсіями роботи [14; 15-18; 22; 28; 50, с. 216; 51].

Розглянуто наступні параметри оцінювання:

а) наявність описових атрибутів елементів;

б) відповідність вмісту та налаштувань описових атрибутів рекомендаціям Android або iOS;

в) обмеження мінімального розміру інтерактивних елементів;

г) перевірка правильності порядку фокусування на елементах;

д) наявність можливості динамічної зміни зовнішнього вигляду тексту;

е) перевірка відповідності співвідношення контрасту стандартам WCAG;

ж) перевірка відсутності небезпечних спалахів.

Із зібраної інформації стає зрозуміло, що незважаючи на ряд переваг наявних інструментів, вони не охоплюють всі параметри оцінювання достатньо вичерпно. Окрім того, більшість інструментів тестування не надає пояснень до використаних критеріїв оцінки цих параметрів, що не дає можливості перевірити, за якими стандартами чи рекомендаціями їх було обрано.

В цій роботі пропонується покроковий алгоритм оцінки інклюзивності мобільних застосунків, що ставить за мету найбільш детально охопити наявні стандарти та рекомендації інклюзивності спираючись на їх ретельний аналіз. Частково цей алгоритм може бути автоматизовано за винятком окремих

ситуацій, коли налаштування елементів залежать від контексту і потребують мануального розгляду розробником.

РОЗДІЛ 2: Розробка алгоритму перевірки інклюзивності мобільних застосунків

2.1 Підтримка роботи вбудованих програм інклюзивності

Головним інструментом покращення інклюзивності мобільних застосунків є підтримка роботи з програмами-зчитувачами екрану. Android та iOS пристрої мають такі програми встановлені за замовченням – TalkBack та VoiceOver відповідно [15; 25]. Вони озвучують інформацію, що міститься на екрані згенерованим голосом, а також надають можливість переключатись між елементами вікна за допомогою жестів або комбінації кнопок. Це дозволяє користувачу ознайомлюватись з вмістом додатку відмінним від зору способом.

Окрім того, Android та iOS обладнані вбудованими програмами полегшення навігації – SwitchAccess та SwitchControl відповідно[13; 5]. Ці програми дозволяють замінити навігацію жестами на альтернативну за допомогою більше легких жестів або фізичних кнопок.

Незалежно від операційної системи принцип роботи цих програм є однаковим. Вони доступаються до елементів вікна застосунку, що містять відповідні атрибути. Для забезпечення правильної взаємодії з застосунком, елементи його вікна мають відповідати переліку налаштувань. Частково ці критерії є подібними для різних операційних систем, але частково відрізняються через особливості збереження метаданих. Загалом налаштування включають в себе присутність описових атрибутів для елементів та атрибутів, що визначають порядок фокусування на них.

Окрім підтримки роботи програм-зчитувачів, правильні налаштування цих атрибутів допомагають задовільнити перелік наступних стандартів WCAG:

а) Стандарт 2.4.3 або «Порядок фокусування» [41]. Якщо по сторінці можна переміщатися послідовно, і послідовність навігації впливає на сприйняття вмісту або роботу, тоді компоненти, на яких можна сфокусуватись, отримують фокус у тому порядку, який зберігає значення вмісту та працездатність.

б) Стандарт 3.3.5 або «Довідка» [43]. Для вмісту сторінки доступна довідка, що залежить від контексту.

в) Стандарт 4.1.2 або «Ім'я, роль, значення» [44]. Для всіх компонентів інтерфейсу, ім’я та роль можна визначити програмним забезпеченням, включаючи допоміжні технології.

До схожих налаштувань відноситься визначення важливих для інклюзивності елементів. Елементи, що явно позначені як неважливі для інклюзивності будуть вважатись виключно декоративними та ігноруватись програмою-зчитування. Для Andoroid це параметр importantForAccessibility [21]. Для IOS це isAccessibleElement для UIKit та accessibility(hidden:) для SwiftUi відповідно[50, с.115]. Правильне використання цього атрибуту запобігає озвученню зайвої інформації, що не має практичної користі для користувача, а також зменшує кількість перемикань між елементами. Деякі елементи можуть мати цей параметр встановленим за замовченням[50, с.115]. Наприклад для зображень він зазвичай буде «false», для текстових та інтерактивних елементів «true». Проте, необхідна додаткова перевірка на доречність цього значення, а також чи не було воно хибно встановлено розробником або неправильно унаслідувано від батьківського класу.

Інтерактивні та текстові елементи, що не є прихованими на екрані не можуть бути пропущені програмою-зчитувачем. Для них параметр «важливий для інклюзивності» обов’язково має бути «true». Також елементи,

що є зазвичай декоративними або прихованими можуть бути важливими для контексту [50, с. 78]. Вони можуть мати заповнені описові атрибути або бути текстовими зображеннями. Описові атрибути та особливості їх заповнення розглянуті далі в цьому розділі. До них належать атрибут підказки, опису вмісту та специфічні лише до певної операційної системи.

Алгоритм перевірки не може оцінити контекст та важливість інформації, яку містять декоративні та приховані елементи. Кожен з них має бути додатково перевіреним розробником. Проте їх атрибути не мають суперечити один одному. Якщо такий елемент має описові атрибути, він має бути важливим для інклюзивності. Якщо описових атрибутів немає алгоритм запропонує розробнику додати їх, якщо на його думку цього вимагає контекст. Цей крок алгоритму зображений рисунку Рисунок 2.1.

Важливим зауваженням є те, що текстові зображення можуть порушувати певні стандарти інклюзивності. Цей випадок розглянуто в підрозділі 2.4 Колір. Контраст.

Наступним подібним налаштуванням є описовий атрибут підказки, що надає додатковий контекст для елемента. Для Andoroid це параметр hint [23], для IOS це accessibilityHint [26]. Атрибут підказки має бути заповнений для всіх елементів, що є важливими для інклюзивності. Це має бути короткий опис елементу або дії, що він виконує. Бажано, щоб він не містив назву елемента, наприклад «кнопка», оскільки це вже автоматично озвучується програмою-зчитувачем і таке пояснення буде зайвим. Підказки мають виглядати як закінчене лаконічне речення, починаючись з великої літери і закінчуючись крапкою [50, с.78, с.118].

Проте використання тільки атрибуту підказки є недостатнім. Користувачі IOS мають можливість вимкнути озвучення підказок, а в Android підказки не відображаються для елементів, що мають текстове значення [50,

с.118]. Тому важливо, щоб для елементів екрану були правильно заповнені атрибути опису вмісту.

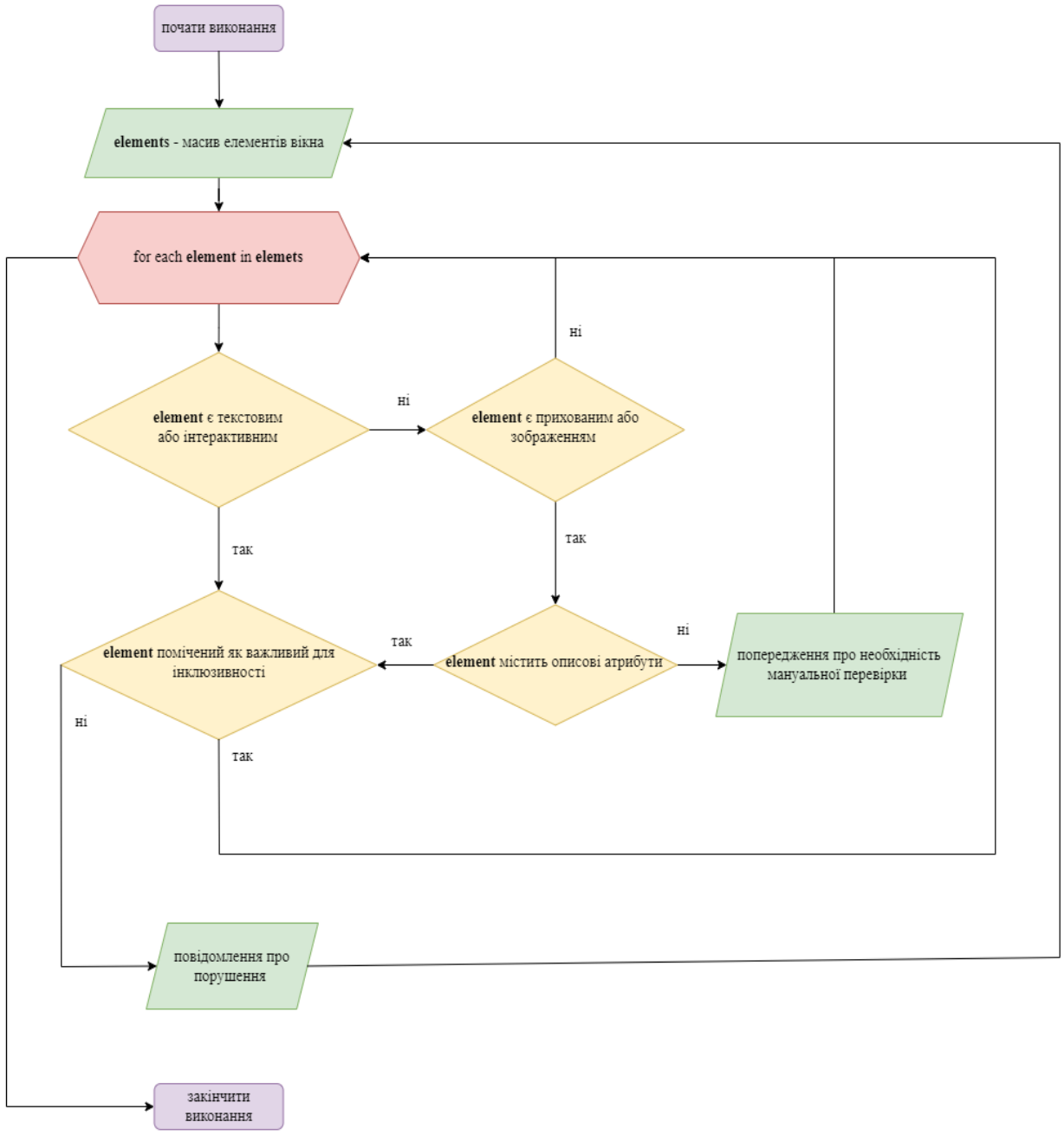


Рисунок 2.1

Ще одним подібним описовим атрибутом є безпосередньо атрибут опису вмісту. Для Andoroid це параметр contentDescription [21], для IOS це

accessibilityLabel [26]. Вимоги до заповнення цього атрибуту є подібними як і для атрибуту підказки. Відміною від значення для атрибуту підказки є те, що це має бути більш лаконічний опис, бажано в одне слово, який не закінчується крапкою [50, с.77, с.116]. Немає офіційного стандарту щодо довжини опису і це залишається на розсуд розробника, проте спираючись на рекомендацію про довжину будь-якого рядка на екрані максимум в 70 символів, можна використати це обмеження [19, с.7]. Всі атрибути опису вмісту мають бути унікальними [21], щоб користувач міг їх розрізняти при зчитуванні один за одним.

Використання атрибутів опису вмісту та підказки допомагає задовільнити стандарти 4.1.2 та 3.3.5 WCAG. Алгоритм перевірки елементів екрану на правильність їх використання наведено на рисунку Рисунок 2.2.

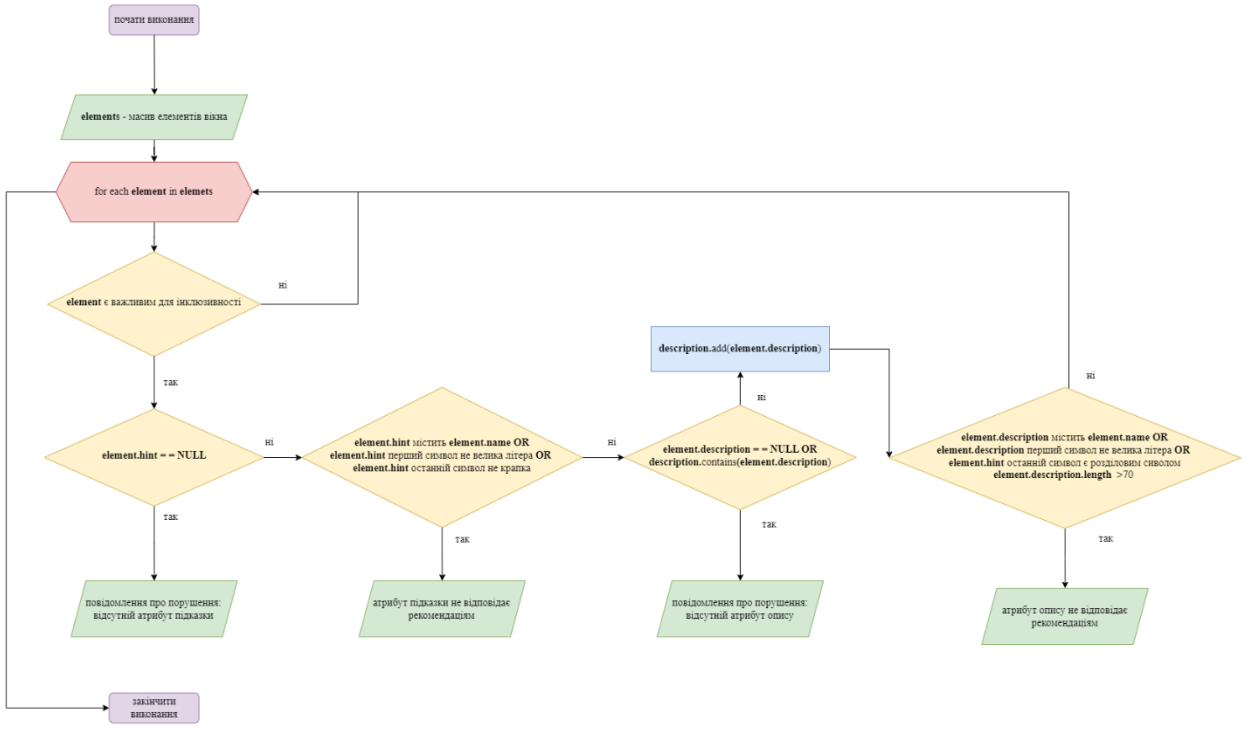


Рисунок 2.2

Наступні описові атрибути, що буде розглянуто, є характерними лише для конкретної операційної системи. В Andorid використовується атрибут labelFor [50, с.79]. Для елементів введення тексту атрибут підказки виконує роль значення введення, що відображається за замовченням [50, с.78]. Під час введення тексту, значення підказки зникає і відповідно втрачається контекст [23]. Для того, щоб уникнути цього, інший текстовий елемент з описом має бути прив’язаний до елементу введення тексту. Необхідно впевнитись, що для кожного елементу введення тексту існує такий неприхований описовий текстовий елемент, що містить його id в параметрі labelFor, як зображено на рисунку Рисунок 2.3.

В IOS використовується описовий атрибут accessibilityValue для UIKit та accessibility(value:) для SwiftUi, що показує поточне значення елементів, які можуть його змінювати, наприклад слайдери та текстові поля [50, с.117]. Зазвичай це значення налаштовується автоматично, проте якщо елемент було створено наслідуванням, цей атрибут можливо необхідно задати вручну. Як зображено на рисунку Рисунок 2.4, алгоритм перевіряє, що всі елементи, які можуть набувати різних значень окрім елементів введення тексту, мають мати цей атрибут заповненим.

Спільним критерієм налаштувань для Android та IOS є встановлення порядку проходження по елементам вікна під час їх зчитування. Допоміжні програми зчитують елементи на екрані в природньому порядку читання зліва направо зверху вниз [50, с.80], або справа наліво для мов, що читаються в цьому порядку [26]. Якщо логічно пов’язані елементи знаходяться знизу або зверху один від одного, програма може не прочитати їх в цьому порядку.

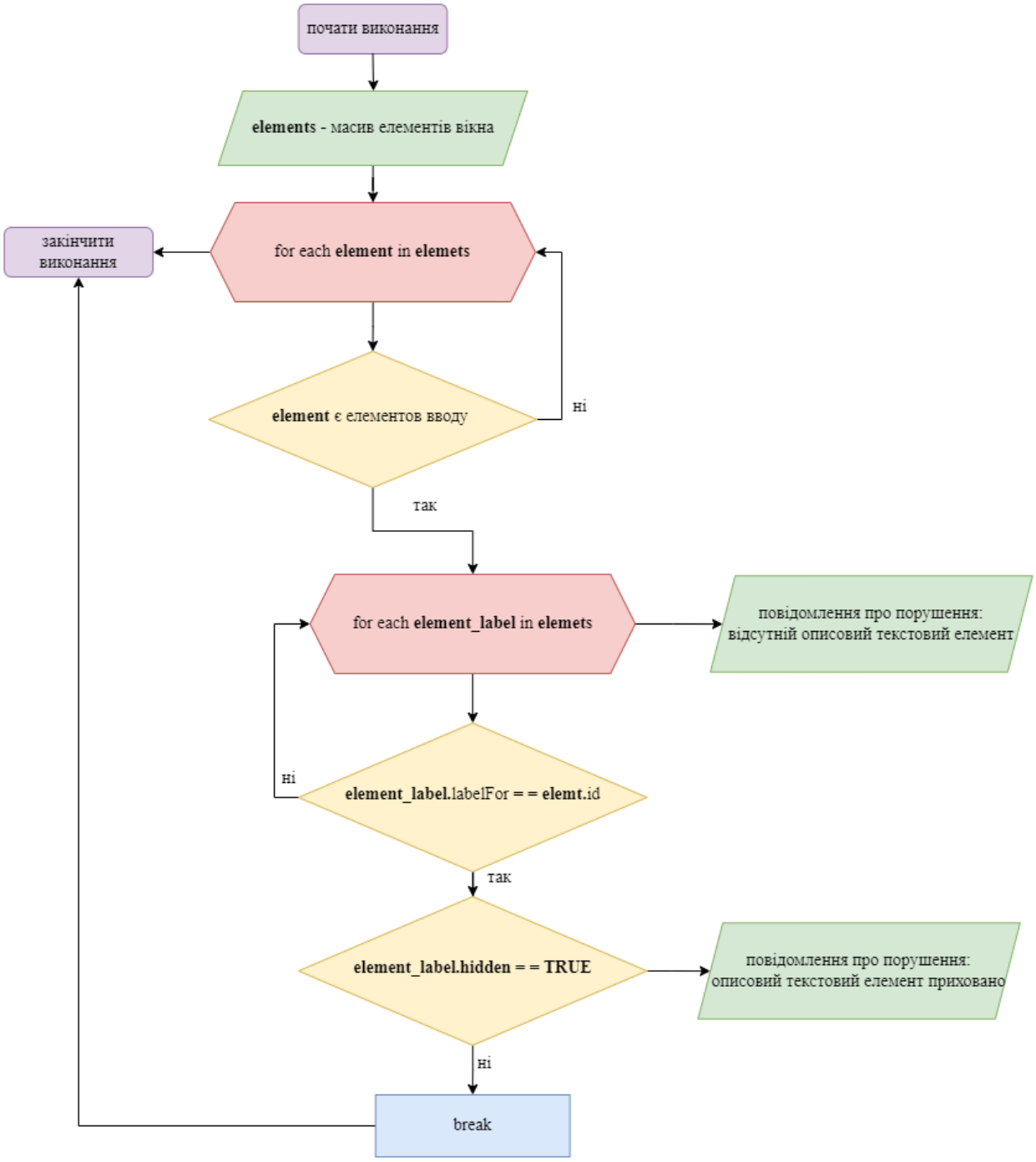

почати виконання
elements - масив елементів вікна
for each element in elemets
закінчити виконання
element є елементов вводу
ні
так
for each element_label in elemets
повідомлення про порушення: відсутній описовий текстовий елемент
ні
element_label.labelFor = = elemt.id
так
element_label.hidden = = TRUE
повідомлення про порушення: описовий текстовий елемент приховано
ні
break

Рисунок 2.3

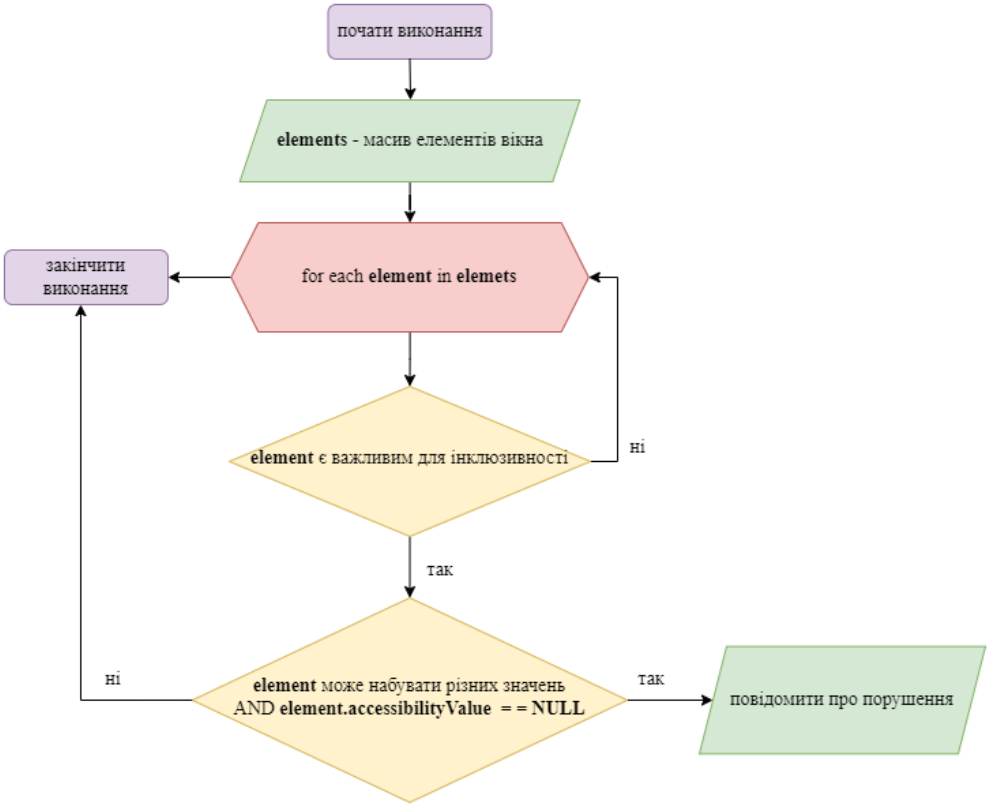


Рисунок 2.4

Для того, щоб контекст вмісту зберігався для користувача потрібно встановити атрибути порядку переходів від одного елементу до іншого, якщо такі наявні. Для Andoird це accessibilityTraversalAfter та accessibilityTraversalBefore, що визначають для поточного елементу попередній та наступний для зчитування. Також потрібно встановити атрибути порядку фокусування nextFocusUp, nextFocusDown, nextFocusLeft, nextFocusRight та nextFocusForward, що забезпечать правильну навігацію по сторінці при перемиканні за допомогою кнопок або клавіатури [50, с.80]. В Android операційна система намагається автоматично обрати найбільш вірогідний порядок зчитування, тому вказувати явно всі атрибути не обов'язково. Тоді можна припустити, що достатньо надати значення принаймні одному атрибуту з кожної з наступних груп: порядок зчитування,

вертикальна навігація, горизонтальна навігація, та наступний елемент nextFocusForward. Додаткові атрибути розробник може вказувати за потреби. Цей крок алгоритму зображено на рисунку Рисунок 2.5.

Для IOS такі атрибути не використовуються. Проте цю проблему можна вирішити логічним групуванням елементів, де необхідна для інклюзивності інформація надається батьківським контейнером. Для цього необхідно переписати масив accessibilityElements для батьківського контейнеру, що зазвичай заповнюється автоматично, і вписати в нього дочірні елементи у бажаному порядку зчитування [5; 26; 50, с.128]. Всі інші елементи, які не були додані до цього масиву, будуть проігноровані програмою-зчитувачем.

Групування можна використати і для Andoird. Для цього на батьківському контейнері дозволяється фокус, тобто значення атрибуту screenReaderFocusable встановлюється «true», а на всіх дочірніх елементах значення атрибуту focusable встановлюється «false». Таким чином програма-зчитувач послідовно оголосить всю інформацію про дочірні елементи один за одним за одне зчитування [23]. Це не вирішує проблему порядку зчитування, проте дозволяє його логічно згрупувати і використати більш ефективно.

Алгоритм не може перевірити, чи логічно пов’язані об’єкти на екрані між собою, і ця перевірка залишається мануальною для розробника. Проте логічно припустити, що групування знадобиться, якщо в одному батьківському контейнері знаходиться більше ніж два важливих для інклюзивності елементи. Алгоритм перевірки знаходить такі батьківські контейнери і пропонує розробнику використати групування, як зображено на рисунку Рисунок 2.6.

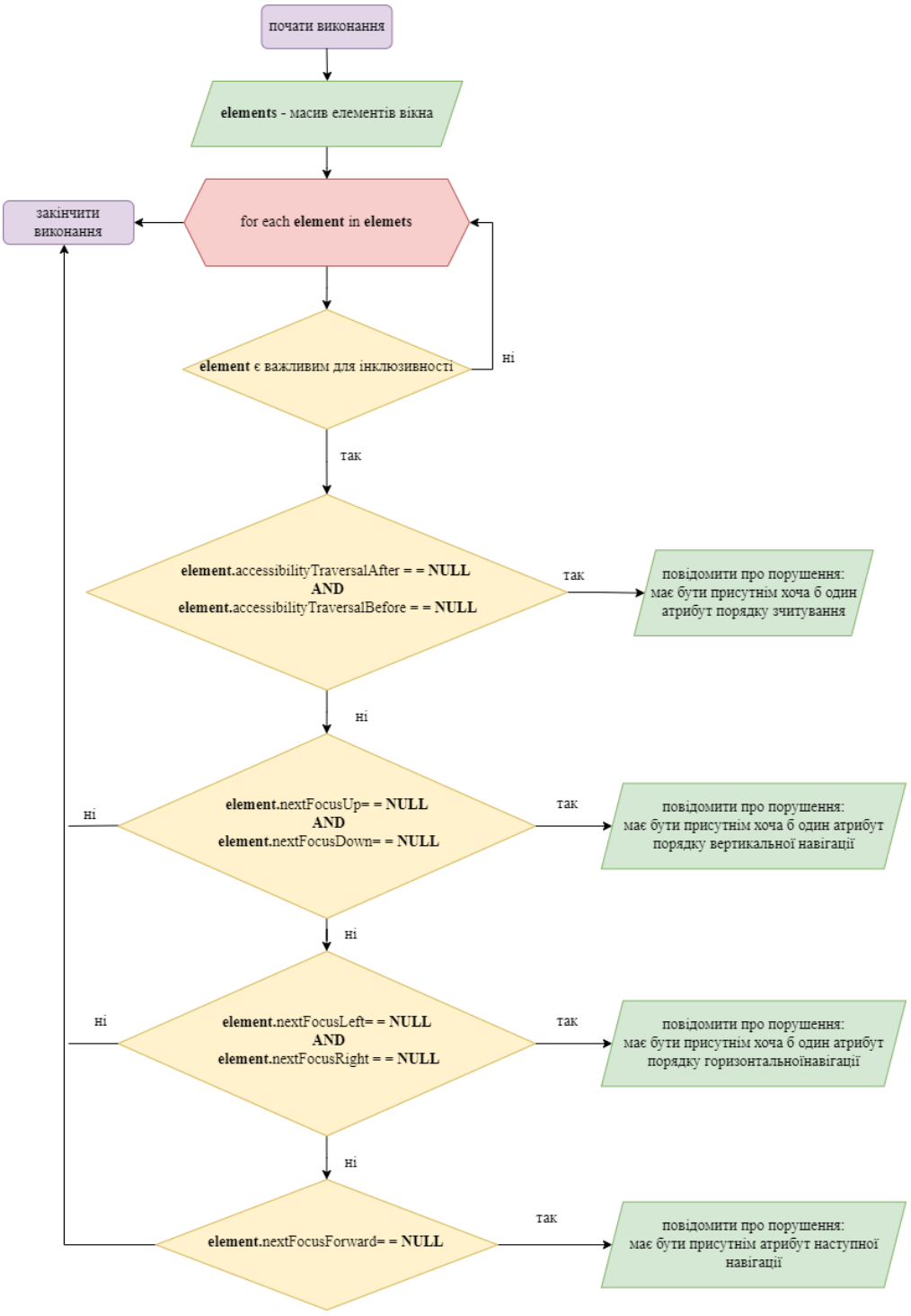

почати виконання
elements - масив елементів вікна
for each element in elemets
закінчити виконання
element є важливим для інклюзивності
ні
так
element.accessibilityTraversalAfter = = NULL
AND
element.accessibilityTraversalBefore = = NULL
так
повідомити про порушення:
має бути присутнім хоча б один атрибут порядку зчитування
ні
element.nextFocusUp= = NULL
AND
element.nextFocusDown= = NULL
ні
так
повідомити про порушення:
має бути присутнім хоча б один атрибут порядку вертикальної навігації
ні
element.nextFocusLeft= = NULL
AND
element.nextFocusRight = = NULL
ні
так
повідомити про порушення:
має бути присутнім хоча б один атрибут порядку горизонтальноїнавігації
ні
element.nextFocusForward= = NULL
так
повідомити про порушення:
має бути присутнім атрибут наступної навігації

Рисунок 2.5

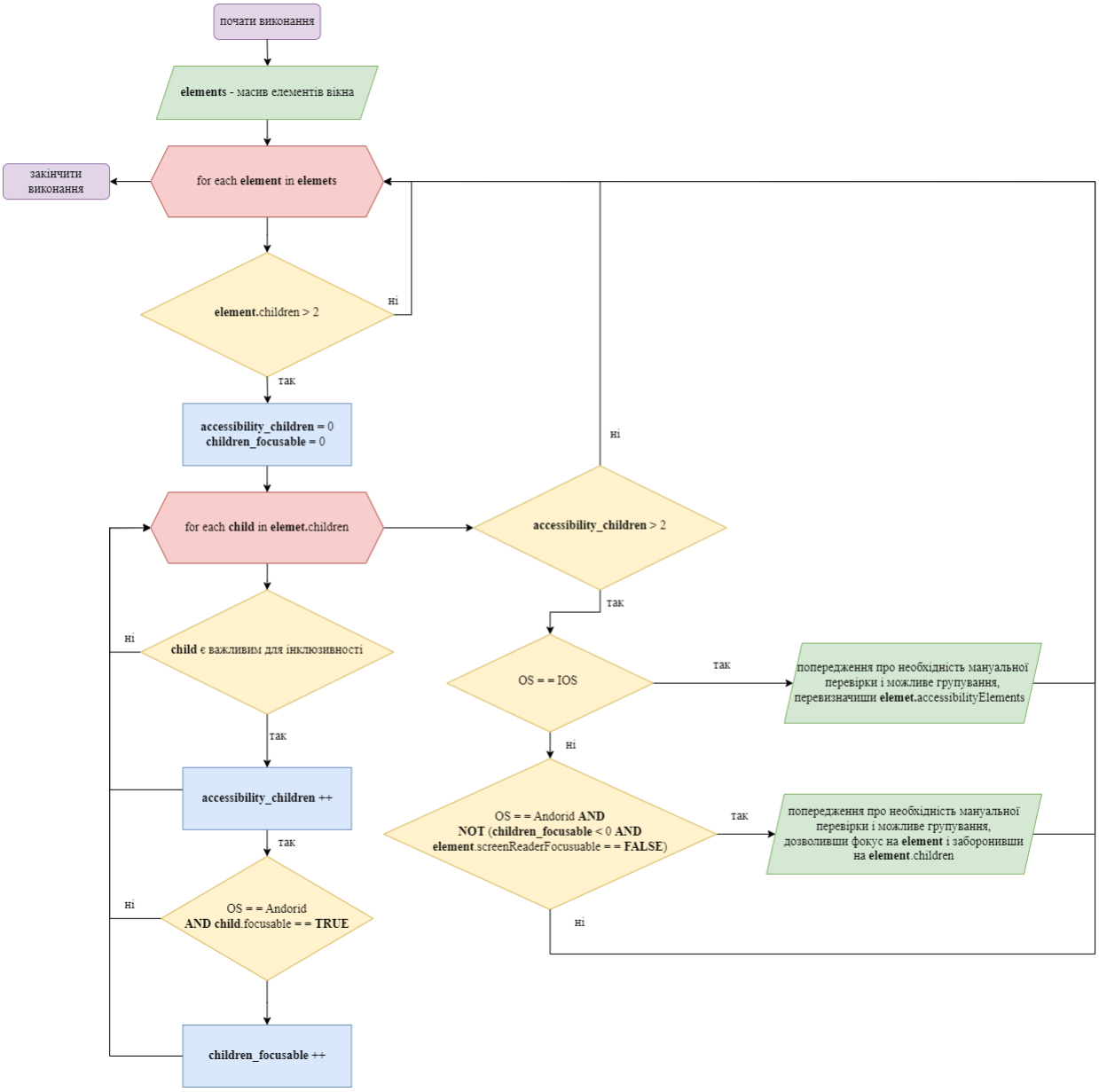


Рисунок 2.6

2.2 Підтримка роботи фізичної клавіатури

Підтримка роботи з фізичною клавіатурою є важливою, оскільки вона надає користувачам альтернативний спосіб взаємодії з програмою. Це особливо важливо для користувачів з обмеженою моторикою рук, які не

можуть використовувати навігацію сенсорним екраном. У WCAG це описано в стандарті 2.1.1 або «Клавіатура»: усі функціональні можливості вмісту можна використовувати через інтерфейс клавіатури [37].

Для забезпечення правильної роботи фізичної клавіатури з додатком необхідно перевірити, що всі інтерактивні елементи мають можливість фокусування на них та встановити правильний порядок фокусування як вже було розглянуто в підрозділі 2.1 Підтримка роботи вбудованих програм інклюзивності. Для Android це атрибут android:focusable, для IOS це accessibilityRespondsToUserInteraction [27, 3]. В переважній більшості випадків значення цих атрибутів встановляться автоматично, залежно від того чи є елемент інтерактивним та важливим для інклюзивності. Проте доречно зробити додаткову перевірку на помилкове встановлення неправильного значення.

Важливо зауважити, що заборона фокусування на елементі не обов’язково є помилкою. В прикладі з групуванням для Anodroid в підрозділі 2.1 Підтримка роботи вбудованих програм інклюзивностіце забезпечувало правильний порядок оголошення елементів програмою-зчитувачем. Окрім того, розробник може мати на меті обмеження взаємодії елемента з клавіатурою, але залишити звукове оголошення. Це зменшує час переключання між елементами і більш очевидно визначає для користувача, які елементи є інтерактивними. Наприклад, це доречно для текстових елементів, що є важливими для інклюзивності, бо містять інформацію, але не очікують дій користувача. Для IOS це можна досягнути встановленням accessibilityRespondsToUserInteraction значення «false» [25]. Для Android також можна приховати елемент від доступу клавіатурою але залишити видимим для VoiceOver встановивши android:focusable значення «false», а android: screenReaderFocusable значення «true» [45].

Алгоритм перевіряє, що всі інтерактивні елементи мають можливість фокусування на них, як показано на наступній діаграмі, а неінтерактивні ні, відповідно до атрибутів певної операційної системи. Важливі для інклюзивності елементи не можуть бути недоступні для програми-зчитувача. Якщо знайдена суперечність, видається попередження про необхідність мануальної перевірки розробником контексту елемента. Цей крок алгоритму зображено на рисунку Рисунок 2.7.

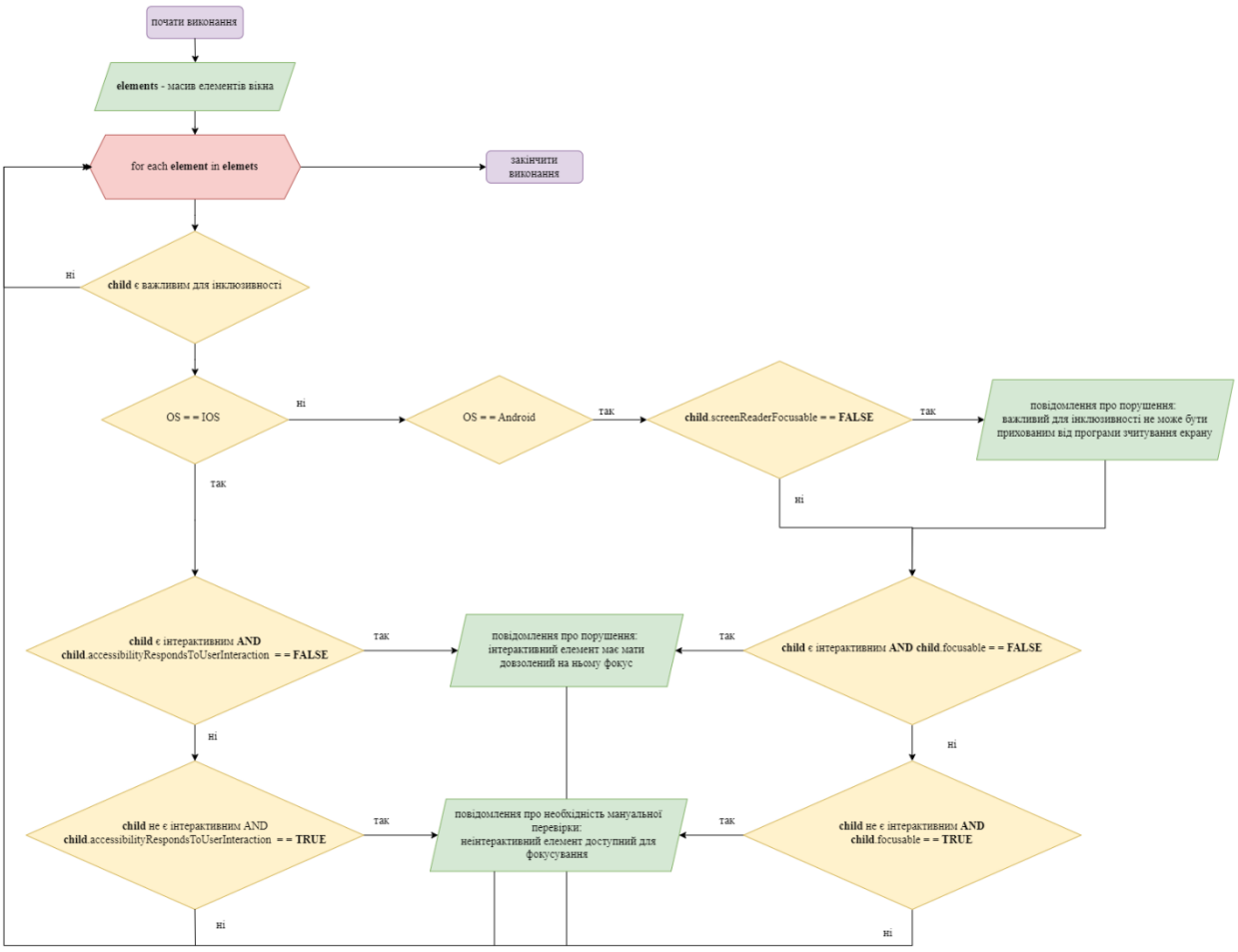


Рисунок 2.7

2.3 Особливості налаштування елементів на екрані маленького розміру

Особливість розробки додатків для мобільних пристроїв є розміщення елементів графічного інтерфейсу на екрані відносно невеликого розміру. Для людей з вадами зору або обмеженою моторикою рук може ускладнити сприйняття інформації, зображеною маленьким шрифтом, а також взаємодію з інтерактивними елементами маленького розміру.

З цих міркувань мінімальний розмір всіх інтерактивних елементів, що передбачають взаємодію з користувачем, особливо елементів введення інформації має бути обмеженим. За рекомендаціями Android площа взаємодії з елементом або площа фокусування на ньому має становити не менше прямокутника 48dp на 48dp[21], а за рекомендаціями IOS не менше 44dp на 44dp [1]. За стандартом WCAG 2.5.5. «Розмір цілі» розмір області, що взаємодіє з вказівником, має становити 44 на 44 пікселі, тому такий критерій оцінювання можна вважати достатнім [42]. Цей стандарт має наступні виключення:

а) елемент взаємодії знаходиться всередині текстового блоку;

б) ціль доступна через альтернативний елемент, що відповідає обмеженому в стандарті мінімальному розміру;

в) розмір елемента визначається агентом користувача;

г) обраний розмір елемента необхідний для контексту та передачі вмісту.

Спираючись на наведені рекомендації всі інтерактивні елементи, окрім посилань, що безпосередньо внесені в текст, мають мати налаштування, що відповідають формулі оскільки забезпечать необхідну мінімальну площу взаємодії [21]:

$$pad\_L + min\_W + pad\_R >= 44\ dp, \qquad (2.1)$$

де pad_L, pad_R – відступи від елемента зліва і справа відповідно;
min_W – мінімальна ширина елемента.

$$pad\_U + min\_H + pad\_D >= 44\ dp, \qquad (2.2)$$

де pad_U, pad_D – відступи від елемента знизу і згори відповідно;
min_H – мінімальна висота елемента.

Оскільки алгоритм не може визначити, чи впливає розмір елемента на контекст, необхідна додаткова мануальна перевірка розробником.

Окрім того, варто мінімізувати кількість фізичних дій, необхідних від користувача для взаємодії з додатком. За стандартом WCAG 1.4.10 «Перенесення тексту» вміст має відображатись без втрати інформації без необхідності двосторонньої прокрутки для елементів з обмеженою широтою чи висотою[32]. За цим стандартом вертикальне прокручування має бути вимкненим для елементів з висотою менше або рівною 256 пікселів, якщо горизонтальне вже дозволене. Аналогічно горизонтальне прокручування має бути вимкненим для елементів з широтою менше або рівною 320 пікселів, якщо вертикальне вже дозволене.

Для елементів, що містять текст, доречно встановити обмеження щодо максимальної кількості символів тексту на рядок, для того щоб уникнути необхідності в горизонтальній прокрутці елементів або нагромадження великої кількості інформації маленького шрифту. Слушним обмеженням буде 70 символів на рядок включно з символами пробілу, що у свій час не викликає задрібний поділ тексту на рядки[19, с.7]. Для цього необхідно розділити текст відповідно до того як його поділено на рядки і перевірити довжину кожної частини. Наприклад це можна зробити доступившись до

об’єкту layoutManager тексту в IOS або до об’єкту Layout в Android і використавши відповідні методи поділу стрічки на підстрічки [30; 29].

Окрім того, в багатьох випадках для кращого сприйняття текстової інформації користувачам необхідно мати можливість виділення тексту, наприклад для копіювання в програму-перекладач, аналізу тексту тощо. Для Andoroid це параметр textIsSelectable [29]. Для IOS це isSelectable для UIKit та allowsSelection для SwiftUi відповідно [20; 4]. Алгоритм перевіряє, що значення таких атрибутів встановлене як «true».

Використання динамічних типів для текстових елементів дозволяє автоматично підлаштовувати розмір шрифту під розмір екрану або додаткові налаштування інтерфейсу. Для елементів, що містять текст в IOS необхідно встановити параметру adjustsFontForContentSizeCategory значення «true», а в Android встановити значення autoSizeTextType як «uniform» [24; 7]. Окрім того для Andoird можна додати додаткові налаштування мінімального, максимального розмірів автоматично збільшеного тексту і крок переходу між розмірами, проте стандартних обмежень для них немає. Їх значення можуть бути встановлені на розсуд розробника. Алгоритм перевіряє лише наявність динамічного типу та чи наявні значення кількості рядків та довжини тексту не суперечать обмеженню кількості символів на рядок. Наведена на рисунку Рисунок 2.8 діаграма демонструє алгоритм перевірки на перелічені вище критерії оцінки.

Важливим зауваженням при створенні додатків для екранів невеликого розміру є можливість підтримки інтерфейсу для горизонтального положення екрану [19, с.7]. Це дозволяє забезпечити стандарт WCAG 1.3.4 «Орієнтація», що передбачає відсутність обмеження перегляду вмісту одним положенням екрану [31]. Дозвіл на зміну положення інтерфейсу встановлюється або в середовищі розробки для IOS Xcode, або встановленням в

AndroidManifest.xml параметру android:screenOrientation значення «sensor». Не зважаючи на те, що алгоритм перевірки не має можливості визначити, чи цей дозвіл надано, оскільки це неможливо встановити спираючись лише на значення елементів вікна, розробник має самостійно мануально перевірити, що такий дозвіл надано.

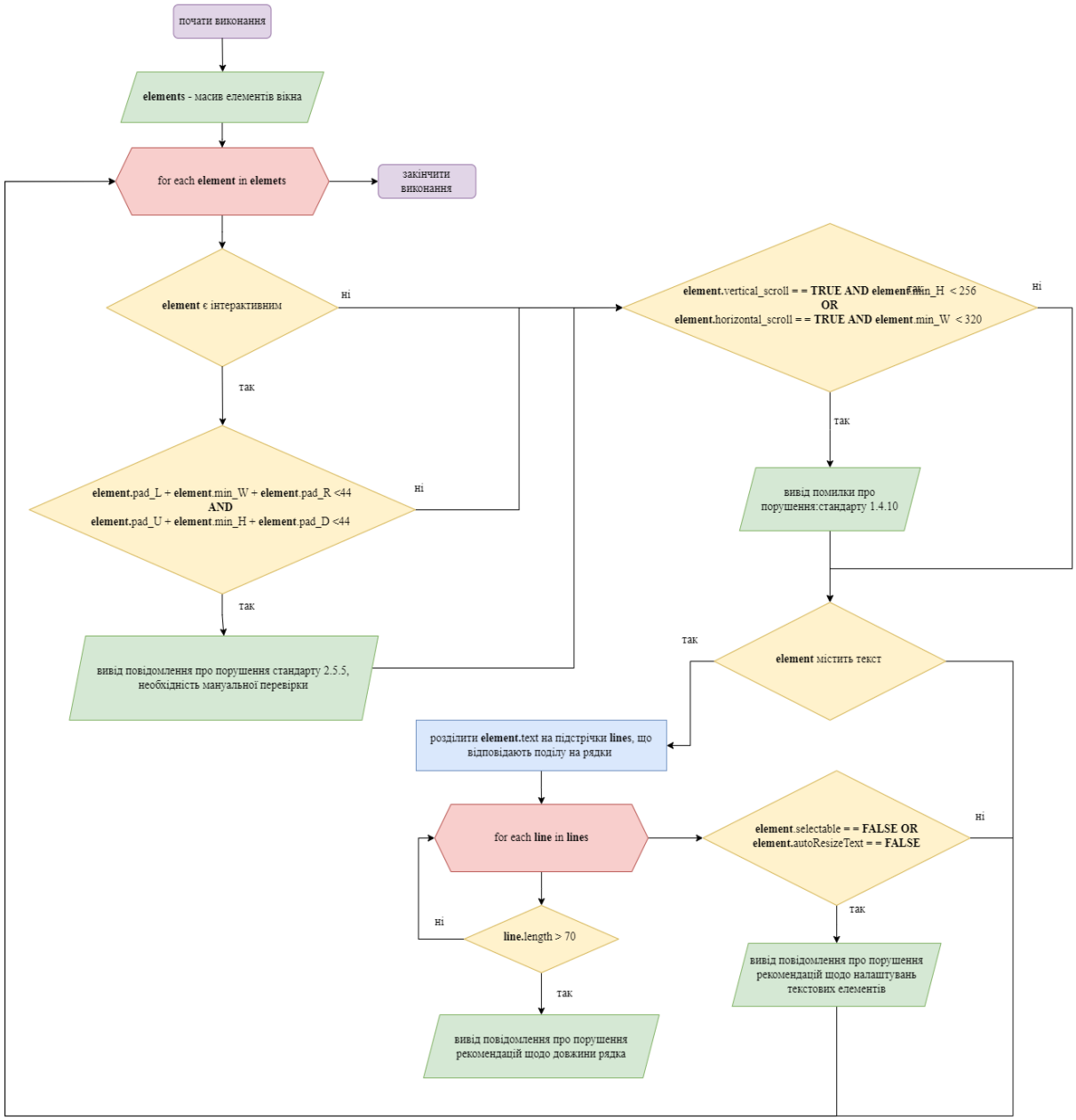


Рисунок 2.8

2.4 Колір. Контраст

Контрастність елементів графічного інтерфейсу додатку є одним з ключових компонентів його інклюзивності. Правильні значення контрастності є важливими для розуміння інформації людьми з порушенням зору та сприйняття кольору. За рекомендаціями WCAG є два основних стандарти до співвідношення контрасту елементів:

a) Стандарт 1.4.3 або «Контраст мінімальний» [33]. Текст та текстові зображення має контраст пропорційний 4.5:1, з наступними винятками:

1) великий текст або його зображення мають контрастність не менше 3:1;

2) текст або текстові зображення, які є декоративними або прихованими, або які є частиною зображення, яке містить інший візуальний зміст, можуть не відповідати вимогам контрасту;

3) текст, який є частиною логотипу чи назви бренду, може не відповідати вимогам контрасту.

b) Стандарт 1.4.6 або «Контраст покращений» [35]. Повторює критерії стандарту 1.4.3 з доповненням:

1) текст та текстові зображення має контраст пропорційний 7:1;

2) великий текст або його зображення мають контрастність не менше 4.5:1.

Очевидно, що краще віддавати перевагу більш жорстким критеріям оцінки, проте допустимим є забезпечення будь-якого з цих стандартів. Вибір залишається за розробниками залежно від потреб користувачів їх застосунку.

Доцільно також доповнити ці стандарти тим, що всі інтерактивні елементи інтерфейсу, які не передбачають розміщення тексту, мають також

відповідати пропорціям контрасту. До таких елементів можна віднести слайдери, прапорці тощо. Оскільки немає стандартних обмежень щодо їх контрасту, братимемо мінімальний критерій, такий як для великого тексту.

Великим текстом за стандартом WCAG вважатимемо той, що рівний або більший 18 pt та жирний від 14 pt [33; 35]. Окреме зауваження зроблено для шрифтів для ієрогліфічних мов, наприклад китайської, корейської та японської. Для них візуально еквівалентний розмір може відрізнятись в пунктах і має бути підібраний окремо.

Пропорційність контрасту або коефіцієнт контрастності обраховується за формулою [33; 35]:

$$(L1 + 0.05) / (L2 + 0.05), \qquad (2.3)$$

де L1 - відносна яскравість світлішого з кольорів;

L2 - відносна яскравість темнішого з кольорів.

Суттєвим недоліком стандартів WCAG є те, що за відсутності кольору фону вони пропонують сприймати його як білий [33; 35], оскільки вони припускають використання лише повністю непрозорих елементів. Кольори елементів графічних інтерфейсів мобільних застосунків задаються моделлю RGBa з додатковим параметром прозорості пікселя альфа [11]. Тому за відсутності кольору фону варто порівнювати колір переднього плану з кольором батьківського контейнеру. Тільки якщо батьківський елемент є коренем дерева ієрархії графічного інтерфейсу, тобто самим вікном, і також не має кольору заднього плану, колір вважатиметься білим.

Початковим завданням алгоритму є рекурсивний доступ до всіх дочірніх елементів батьківського контейнеру, починаючи з кореневого, і виконання необхідних обрахунків коефіцієнту контрастності для кожного з

них в циклі. Якщо поточний елемент, що розглядається, має дочірні елементи, виконується наступний рекурсивний крок. В нього за наявності передається колір фону цього елементу або за його відсутності – колір фону попереднього батьківського контейнеру. Цей крок алгоритму показано на рисунку Рисунок 2.9.

Як згадано вище, колір фону елементу, що оцінюється, може бути напівпрозорим і відповідно складатись з накладання власного значення та значення кольору батьківського контейнеру. Тому для того, щоб правильно обрахувати контраст, параметр альфа поточного елементу та колір фону батьківського контейнеру має бути врахованим за наступними формулами для червоного, зеленого та синього світла відповідно:

$$R\_background = (R8bit\_child * A + R8bit\_ parent * (1-A)) /255, \quad (2.4)$$
$$G\_background = (G8bit\_child * A + G\ 8bit\_ parent * (1-A)) /255,$$
$$B\_background = (B8bit\_child * A + B8bit\_ parent * (1-A))/255,$$

де A – параметр прозорості елементу, що оцінюється;

R8bit_ child, G8bit_ child, B8bit_ child – 8-бітне значення заданого кольору елементу, що оцінюється;

R8bit_ parent, G8bit_ parent, B8bit_ parent – 8-бітне значення заданого кольору батьківського елементу.

На вхід наступного кроку алгоритму, що виконує необхідні обрахунків коефіцієнту, подається поточний елемент та колір його батьківського контейнеру. Схема цього кроку зображена на рисунку Рисунок 2.10.

Якщо колір фону поточного елементу відсутній – за такий сприймається колір батьківського елементу. Якщо параметр альфа відмінний від одиниці – значення кольору перераховуються за наведеними вище формулами.

Якщо елемент містить текст порівнюються колір тексту та фону цього елементу. Аналіз результатів порівняння описаний нижче відповідно до стандартів щодо розміру тексту.

Для інтерактивних елементів доцільним буде порівнювати їх колір з кольором батьківського контейнеру, або колір переднього і заднього планів, якщо такі присутні. Забезпечення правильного коефіцієнту контрасту для одного з цих порівнянь буде достатньо, щоб елемент був добре видимим на екрані.

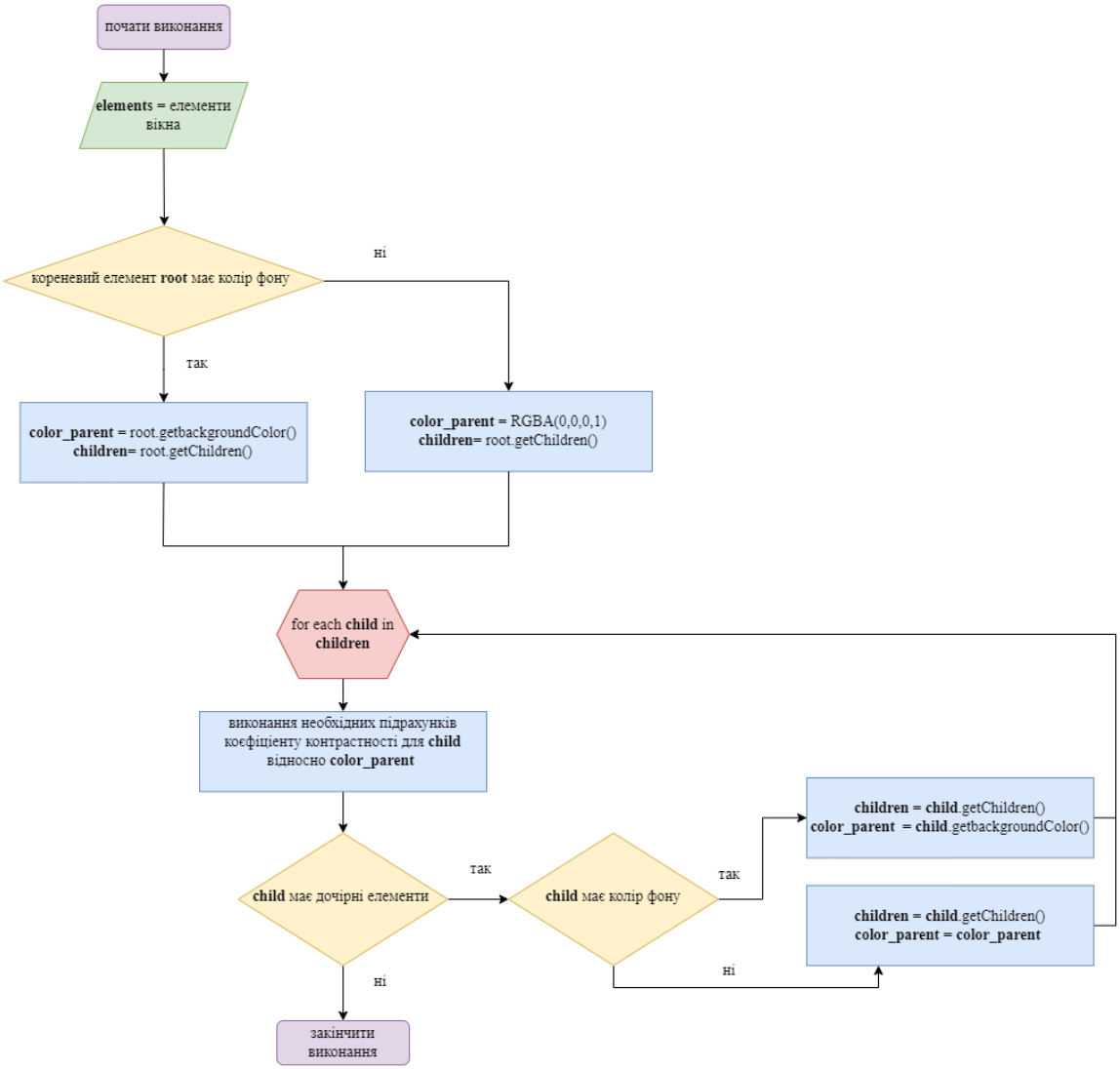


Рисунок 2.9

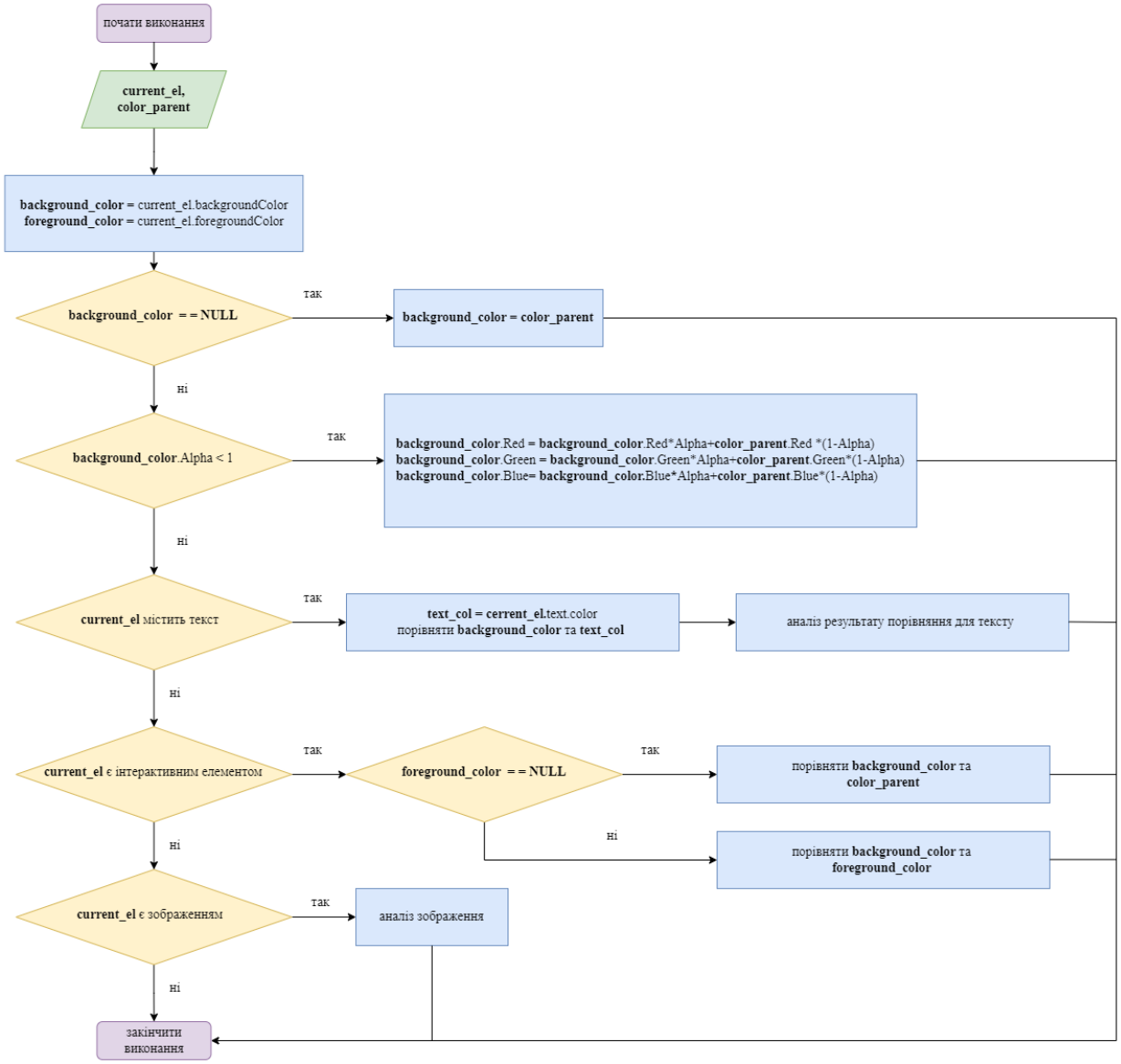


Рисунок 2.10

Якщо елемент є зображенням він потребує окремого аналізу, що є описаний далі в роботі.

Як зазначено вище, порівняння кольорів основане на порівнянні їх відносної яскравості. Її обрахунок відбувається за наступною формулою [33; 35]:

$$L = 0.2126 * R + 0.7152 * G + 0.0722 * B, \qquad (2.5)$$

де R – червоне світло;

G – зелене світло;

B – синє світло.

В свою чергу R, G, B це відповідні лінеарезовані значення кольору пікселя адитивної колірної моделі RGB. Формули для обрахунку цих параметрів також наведені в стандартах WCAG [1, 2]:

$$RsRGB = R8bit /255, \quad (2.6)$$
$$GsRGB = G8bit /255,$$
$$BsRGB = B8bit /255,$$

Якщо RsRGB <= 0.03928 то R = RsRGB/12.92 інакше R =
((RsRGB+0.055)/1.055) ^ 2.4,
Якщо GsRGB <= 0.03928 то G = GsRGB/12.92 інакше G =
((GsRGB+0.055)/1.055) ^ 2.4,
Якщо BsRGB <= 0.03928 то B = BsRGB/12.92 інакше B =
((BsRGB+0.055)/1.055) ^ 2.4.

де R8bit, G8bit, B8bit – 8-бітне значення заданого кольору.

Для кожної пари кольорів, що порівнюються підраховуються відносні яскравості L1 та L2. За їх значеннями обраховується коефіцієнт контрастності. Як зазначено вище, за L1 береться світліший колір, тобто той, чия відносна яскравість більша, як показано на рисунку Рисунок 2.11. Відносні яскравості для кожного кольору у свою чергу підраховуються за наведеними вище формулами, як показано на рисунку Рисунок 2.12.

Після підрахунку коефіцієнту контрастності його значення порівнюється з значенням стандарту для відповідного елемента. Як було обрано вище для інтерактивних елементів це значення буде таким, як для великого тексту, що наведено на рисунку Рисунок 2.13.

Для оцінки тексту на вхід окрім коефіцієнту контрастності, також подається сам текст для визначення його розміру та локалізації. Відповідно до стандартів він аналізується за наступною схемою, зображеною на рисунку Рисунок 2.14.

Аналіз зображень ускладнений тим, що залежно від контексту вони можуть бути інформативними або суто декоративними. Декоративні зображення, такі як логотипи звільнені від необхідності відповідати стандартам.

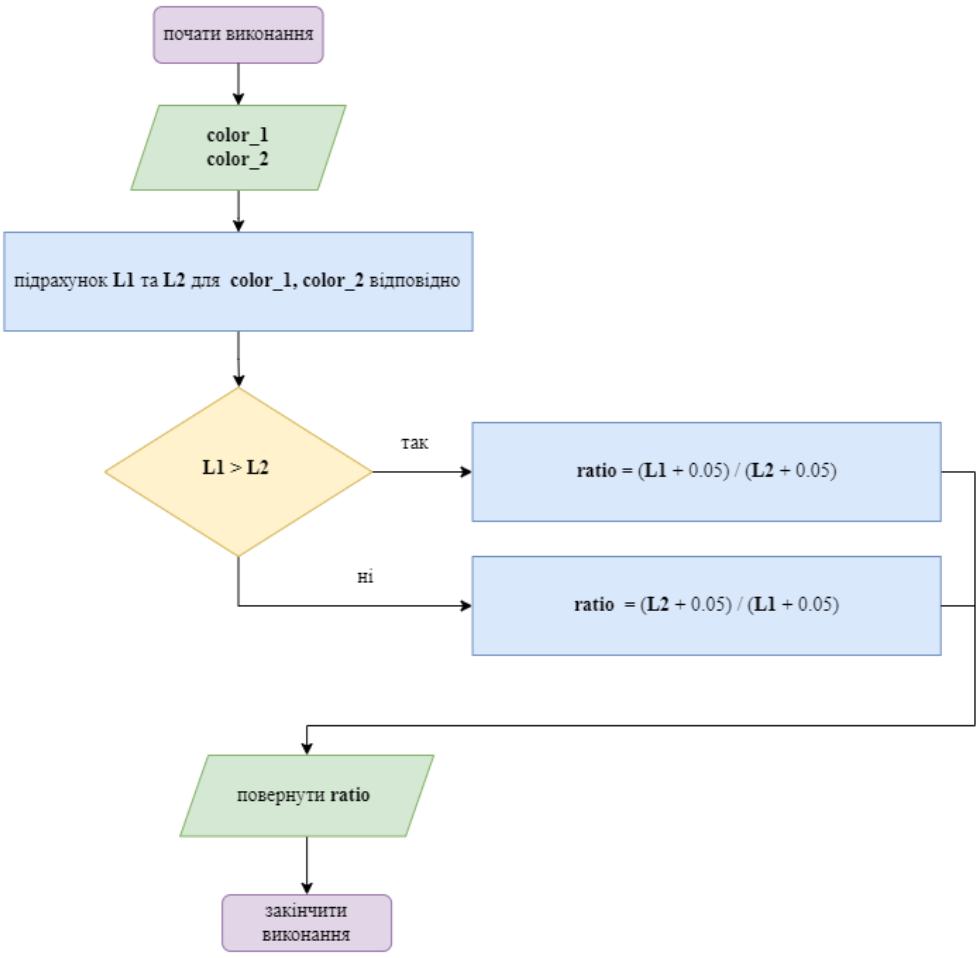


Рисунок 2.11

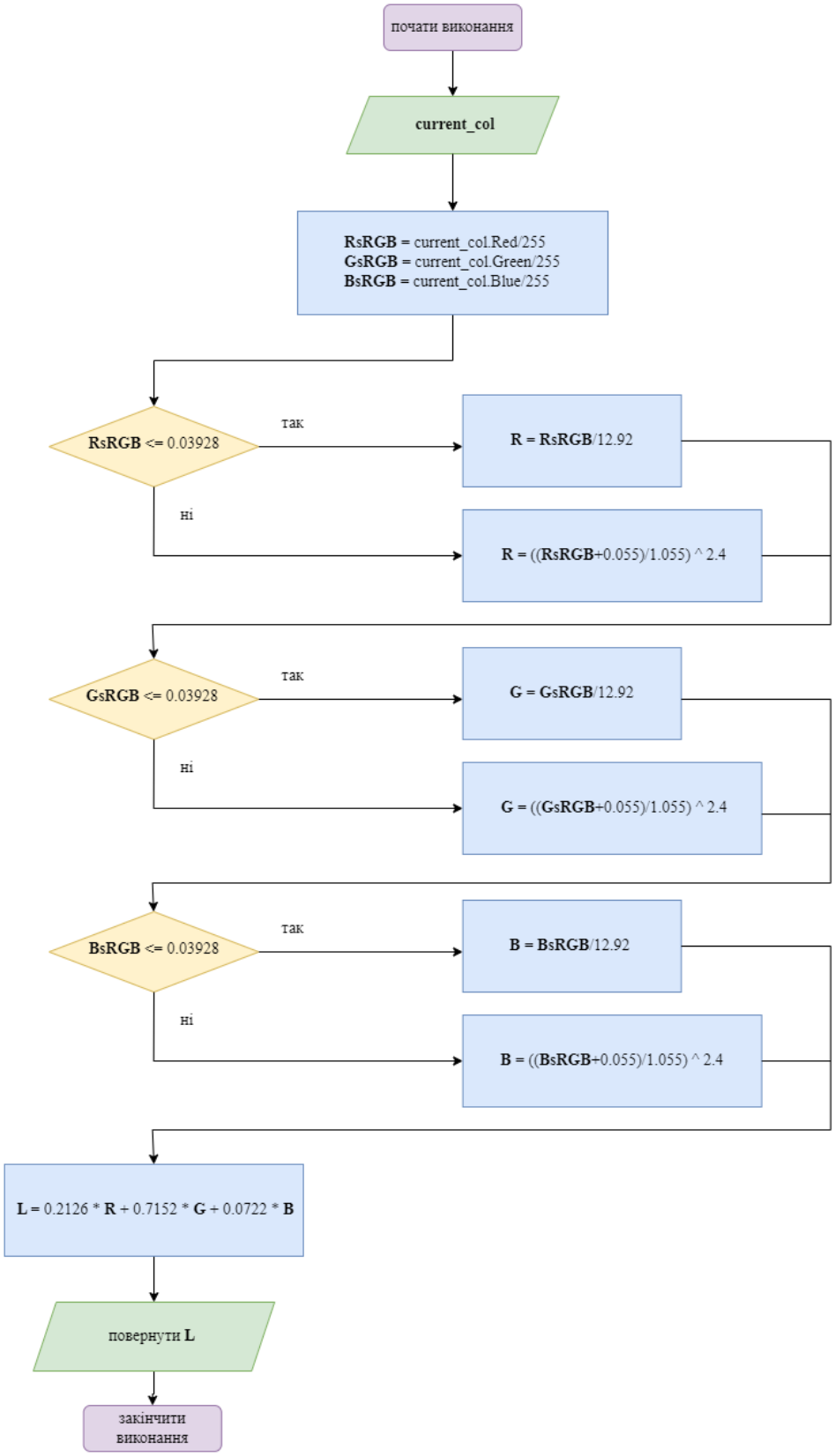

почати виконання
current_col
RsRGB = current_col.Red/255
GsRGB = current_col.Green/255
BsRGB = current_col.Blue/255
RsRGB <= 0.03928
так
ні
R = RsRGB/12.92
R = ((RsRGB+0.055)/1.055) ^ 2.4
GsRGB <= 0.03928
так
ні
G = GsRGB/12.92
G = ((GsRGB+0.055)/1.055) ^ 2.4
BsRGB <= 0.03928
так
ні
B = BsRGB/12.92
B = ((BsRGB+0.055)/1.055) ^ 2.4
L = 0.2126 * R + 0.7152 * G + 0.0722 * B
повернути L
закінчити виконання

Рисунок 2.12

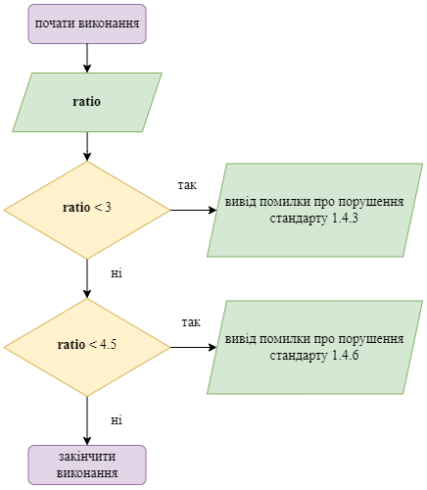


Рисунок 2.13

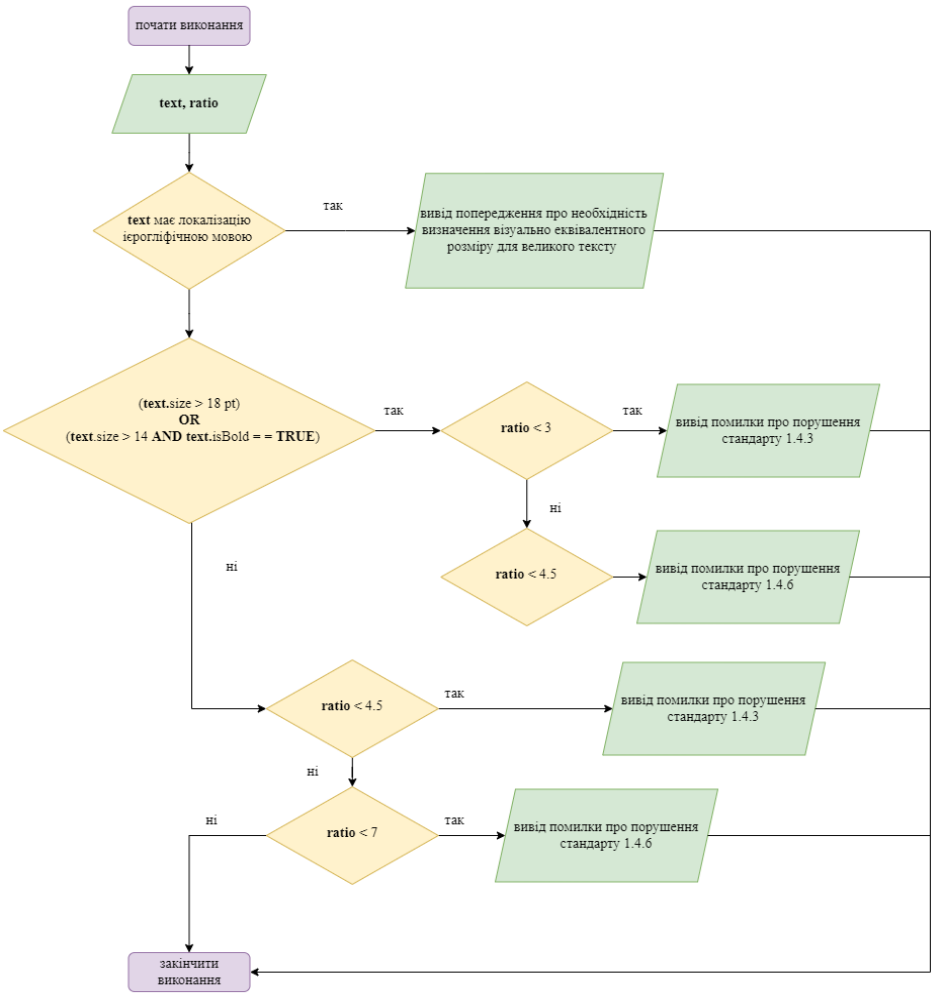


Рисунок 2.14

Загалом WCAG рекомендує утримуватись від використання текстових зображень для передачі інформації:

а) за стандартом 1.4.6 для передачі інформації використовується текст, а не текстове зображення з наступними винятками [34]:

1) є можливість налаштування розміру та кольору тексту в зображенні;

2) така подача є необхідною для відображення цієї інформації.

б) за стандартом 1.4.9 текстові зображення можуть використовуватись виключно як декоративні [36].

В мобільних застосунках елементи вікна, що є зображеннями, не передбачають можливість візуального налаштування користувачем, тому перший виняток з стандарту 1.4.6 не може бути забезпечений. Припускаємо, що подача інформації в цьому вигляді є все ж концептуально необхідною. Виникає необхідність мануальної перевірки розробником контексту зображення, чи є воно суто декоративним, чи все-таки містить корисну інформацію.

Припускаємо, що атрибути інклюзивності були правильно встановлені на всіх елементах вікна і декоративні зображення були позначені як неважливі для інклюзивності. В такому випадку якщо алгоритм знаходить зображення, що є важливим для інклюзивності, він повідомляє про порушення стандарту 1.4.9 та необхідність мануальної перевірки на відповідність стандарту 1.4.6.

За допомогою будь-якої допоміжної бібліотеки оптичного розпізнавання символів (optical character recognition) можливо виокремити текст від фону. Проте, текстові зображення також ускладнюють аналіз тим, що можуть містити в собі декілька кольорів тексту та фону, різні шрифти, накладання літер тощо. Заздалегідь передбачити всі такі нюанси неможливо. Загалом цьому немає необхідності, оскільки такі зображення ускладнюють

сприйняття тексту і суперечать стандартам інклюзивності, тому мають бути скоріше декоративними ніж інформативними.

З цього можливо зробити припущення, що текстове недекоративне зображення матиме чітко виражений текст та фон одного кольору або складається з обмеженої кількості кольорів подібних значень. Контраст зображень з неоднорідним кольором фону та переднього плану можна рахувати, спираючись на середнє значення множин пікселів, що їм відповідають [12, с.3].

Тоді алгоритм аналізу зображення відповідатиме схемі наведеній нижче. Обрахунок середнього значення кольору (функція average) це обрахунок середнього значення окремо для червоного, синього та зеленого світла моделі RGB. В даному випадку коефіцієнт альфа вираховується зі значення атрибуту прозорості самого зображення.

Після знаходження середнього значення кольору тексту і фону їх слід за такою ж схемою, як наведено на рисунку Рисунок 2.15 для текстових елементів.

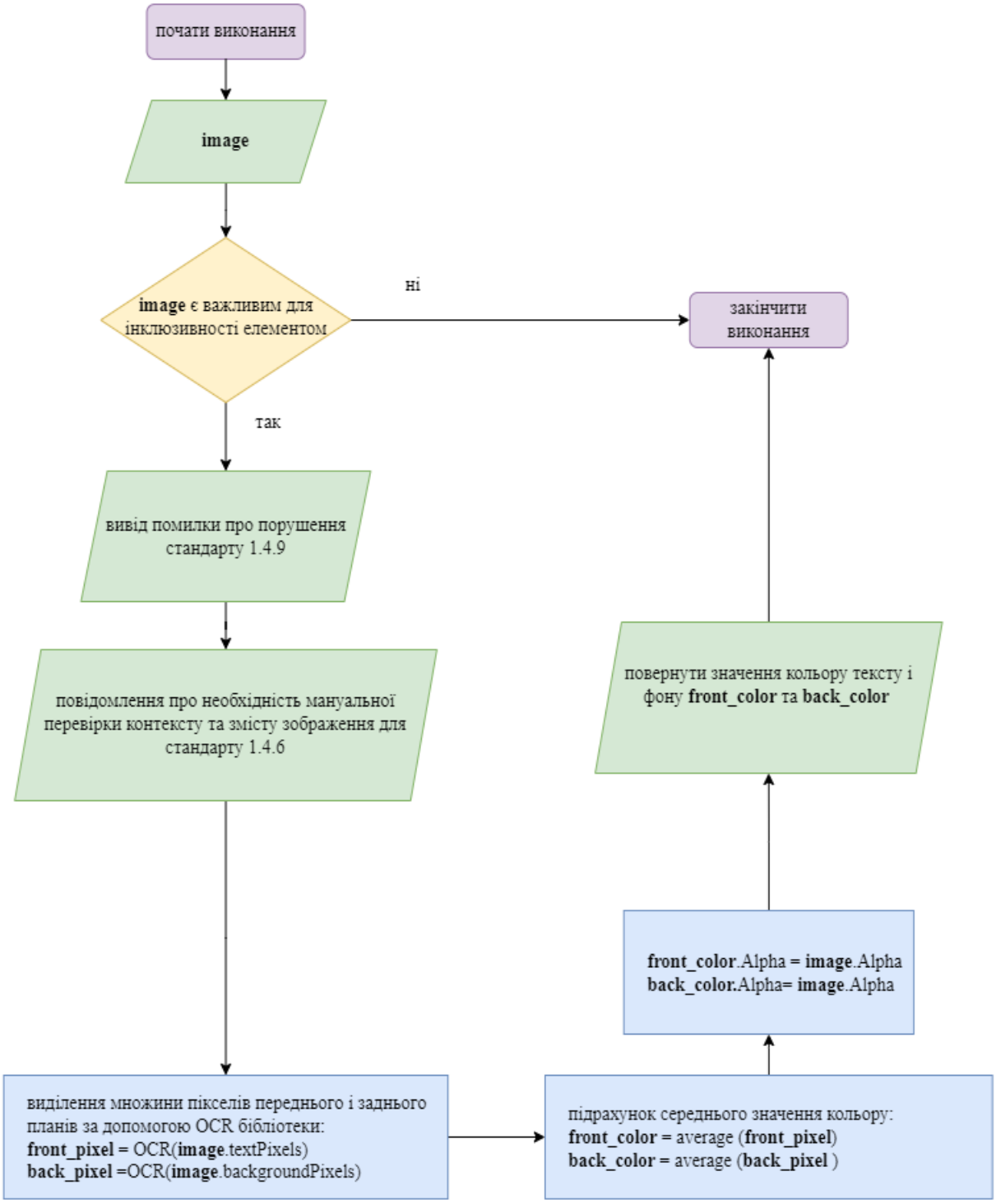

почати виконання
image
image є важливим для інклюзивності елементом
ні
закінчити виконання
так
вивід помилки про порушення стандарту 1.4.9
повідомлення про необхідність мануальної перевірки контексту та змісту зображення для стандарту 1.4.6
повернути значення кольору тексту і фону front_color та back_color
front_color.Alpha = image.Alpha
back_color.Alpha= image.Alpha
виділення множини пікселів переднього і заднього планів за допомогою OCR бібліотеки:
front_pixel = OCR(image.textPixels)
back_pixel =OCR(image.backgroundPixels)
підрахунок середнього значення кольору:
front_color = average (front_pixel)
back_color = average (back_pixel)

Рисунок 2.15

2.5 Спалахи. Контраст спалахів

Контраст динамічних контенту екрану, а саме коефіцієнт контрастності кадрів та швидкість їх зміни, є критичним параметром для людей з фоточутливістю до спалахів. Динамічний контент з великою кількістю спалахів може викликати судоми у людей з епілептичними розладами. За рекомендаціями WCAG є два основних стандарти до кількості спалахів на екрані:

а) Стандарт 2.3.1 або «Три спалахи або спалахи нижче порогу» [39]. Сторінки не містять нічого, що спалахує більше трьох разів протягом будь-якої однієї секунди, або спалах є нижчим від порогу розміру для звичайного спалаху та червоного.

б) Стандарт 2.3.2 або «Три спалахи» [40]. Сторінки не містять нічого, що спалахує більше трьох разів протягом будь-якої однієї секунди.

Спалах це пара протилежних різких змін відносної яскравості, яка потенційно може бути небезпечною для фоточутливих людей [39]. Тобто це послідовні переходи між трьома зображеннями від високої яскравості до низької і назад до високої, або навпаки – від низької до високої і назад до низької. Стандарти WCAG виділяють два типи спалахів – звичайний та червоний.

Звичайний спалах виникає при зміні відносної яскравості на десять або більше відсотків від максимальної, тобто на значення більше або рівне 0.1. Оскільки відносна яскравість може набувати максимального значення в 1.0 [39; 40]. При цьому серед зображень, що порівнюються при послідовних переходах, низькою яскравістю вважається та, що менша за 0.8.

Червоний спалах визначається як будь-яка пара протилежних переходів, де зустрічається насичений червоний колір [38-40]. Для нього мають справджуватись два наведені нижче критерії.

Першим критерієм є визначення насиченого червоного кольору. Червоний колір вважається насиченим, якщо для лінеарезованих значення кольору пікселя адитивної колірної моделі RGB, обрахованих за формулою RsRGB = *R8bit /255*, *(2.6*), справджується нерівність:

$$\frac{R}{R+G+B} \geq 0.8, \qquad (2.7)$$

де R – червоне світло;

G – зелене світло;

B – синє світло.

Другим критерієм є те, що для зображень, які порівнюються, має бути різниця станів на діаграмі кольоровості CIE 1976 UCS більше ніж 0.2 [38]. Різниця станів обраховується за формулою:

$$\sqrt{(u_1 - u_2)^2 + (v - v_2)^2}\ , \qquad (2.8)$$

$$X = 0.4124564 * R + 0.3575761 * G + 0.1804375 * B,$$

$$Y = 0.2126729 * R + 0.7151522 * G + 0.0721750 * B,$$

$$Z = 0.0193339 * R + 0.1191920 * G + 0.9503041 * B,$$

$$u = \frac{4 * X}{(X + 15 * Y + 3 * Z)},$$

$$v = \frac{9 * Y}{(X + 15 * Y + 3 * Z)},$$

де u_1, v_1 - кольорові координати початкового стану;

u_2, v_2 - кольорові координати кінцевого стану;

X, Y, Z – тристимульні значення кольору колірної моделі CIE XYZ;

R, G, B – значення обраховані за формулою RsRGB = R8bit /255, (2.6).

Переведення значень кольорів в значення моделі CIE XYZ є необхідним, оскільки вона визначає кількісні зв’язки між розподілами довжин хвиль в електромагнітному видимому спектрі та фізіологічно сприйнятими зором людини кольорами[10]. В свою чергу переведення до колірної моделі CIE 1976 UCS забезпечує сприйняття більш рівномірного інтервалу кольорів для кольорів із приблизно однаковою яскравістю [9]. Це дає можливість проаналізувати зміну кольорів в парі переходів відносно того, як їх розрізняє око людини.

Наведені вище формули стосуються безпосередньо пар переходів для кожного пікселя. Відповідно до стандарту 2.3.1 WCAG спалах вважається безпечним, якщо сумарний розмір області пікселів, що спалахують, не перевищує 25% від області, що відповідає десяти градусам поля зору людини в будь-якій частині екрану при типовій відстані перегляду. Типовою відстанню перегляду вважається 22-26 дюймів або 55-66 сантиметрів [39], тобто середня відстань перегляду екрану комп’ютера при роботі за столом.

Десять градусів поля зору відповідають розміру прямокутника приблизно 341 на 256 пікселів для екрану роздільної здатності 1024 на 768 пікселів [38;39]. Очевидно, що мобільні пристрої переважно мають значно менший розмір екрану. Проте, стандарти WCAG припускають, що при однаковій відстані перегляду цей прямокутник займатиме однаковий градус поля зору і відповідно забезпечить задовільну оцінку вмісту екрану[3].

Недоліком такого міркування є те, що перегляд екрану мобільних пристроїв відбувається зазвичай на значно меншій відстані від очей. Окрім того, якщо до групи потенційних користувачів додатку входять люди з вадами зору, вони можуть розміщувати екран яке значно ближче, так і навпаки далі для зручності перегляду. Варто взяти до уваги те, що розмір

об’єкта, який займає певний кут поля зору, є прямо пропорційним дистанції до нього:

$$\theta = \text{atan}(\frac{S}{D}), \qquad (2.9)$$

$$S = \tan(10°) * D,$$

де S – розмір зображення;

θ – кут поля зору;

D – відстань до зображення.

Окрім того, зі збільшенням роздільної здатності зображення того самого розміру в пікселях стає меншим в дюймах і відповідно безпечнішим. В стандартах WCAG розрахунку розміру прямокутника взято найменшу роздільну здатність, що використовується в сучасних пристроях, а саме 1024 на 768 пікселів на екрані 15-17 дюймів [39]. Це приблизно відповідає 85-75 ppi.

Отже, розмір прямокутника, що займає десять градусів поля зору, відрізнятиметься для кожного застосунку залежно від потреб його потенційних користувачів, а саме передбачуваної відстані перегляду, та роздільної здатності приладів, на які його планують встановити. Розробники можуть обраховувати його розмір для аналізу вмісту екрану відповідно до цих параметрів. Це дозволяє залишити гнучкість аналізу та врахувати різницю наприклад між планшетами та телефонами. Для формули обрахунку використана наступна пропорція. Вона спирається на найгірші початкові показники, використані WCAG (найменша роздільна здатність 75 ppi та найближча відстань перегляду 22 дюйми):

$$S_w = 341 * \frac{75\ ppi}{P} * \frac{22\ inch}{D_i}, \qquad (2.10)$$

$$S_h = 256 * \frac{75\ ppi}{P} * \frac{22\ inch}{D_i},$$

де S_w– ширина компоненту, що спалахує;

S_h – висота компоненту, що спалахує;

P – роздільна здатність екрану в пікселях на дюйм;

D_i - передбачувана відстань перегляду екрану.

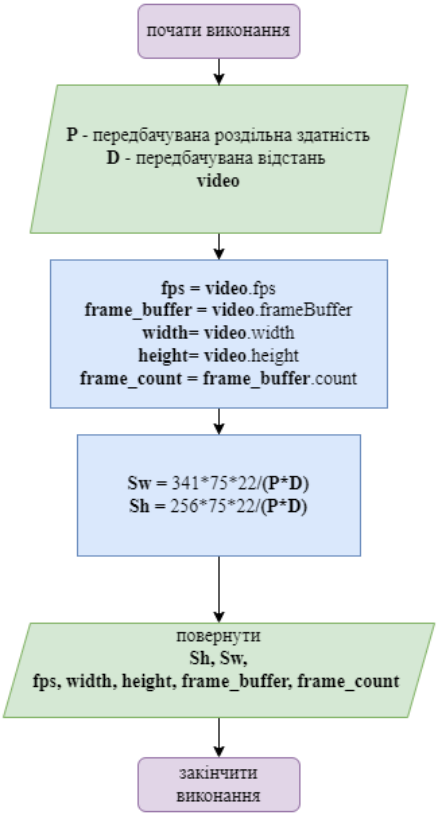
До динамічного контенту, що потребує аналізу на кількість спалахів, можна віднести відео-матеріали, анімовані зображення у форматах gif та apng, що можуть міститися в медіа плеєрах, контейнерах для зображень або як атрибути інших елементів.

На

На

Оскільки

Масиви


почати виконання
P - передбачувана роздільна здатність
D - передбачувана відстань
video
fps = video.fps
frame_buffer = video.frameBuffer
width= video.width
height= video.height
frame_count = frame_buffer.count
Sw = 341*75*22/(P*D)
Sh = 256*75*22/(P*D)
повернути
Sh, Sw,
fps, width, height, frame_buffer, frame_count
закінчити
виконання

Рисунок

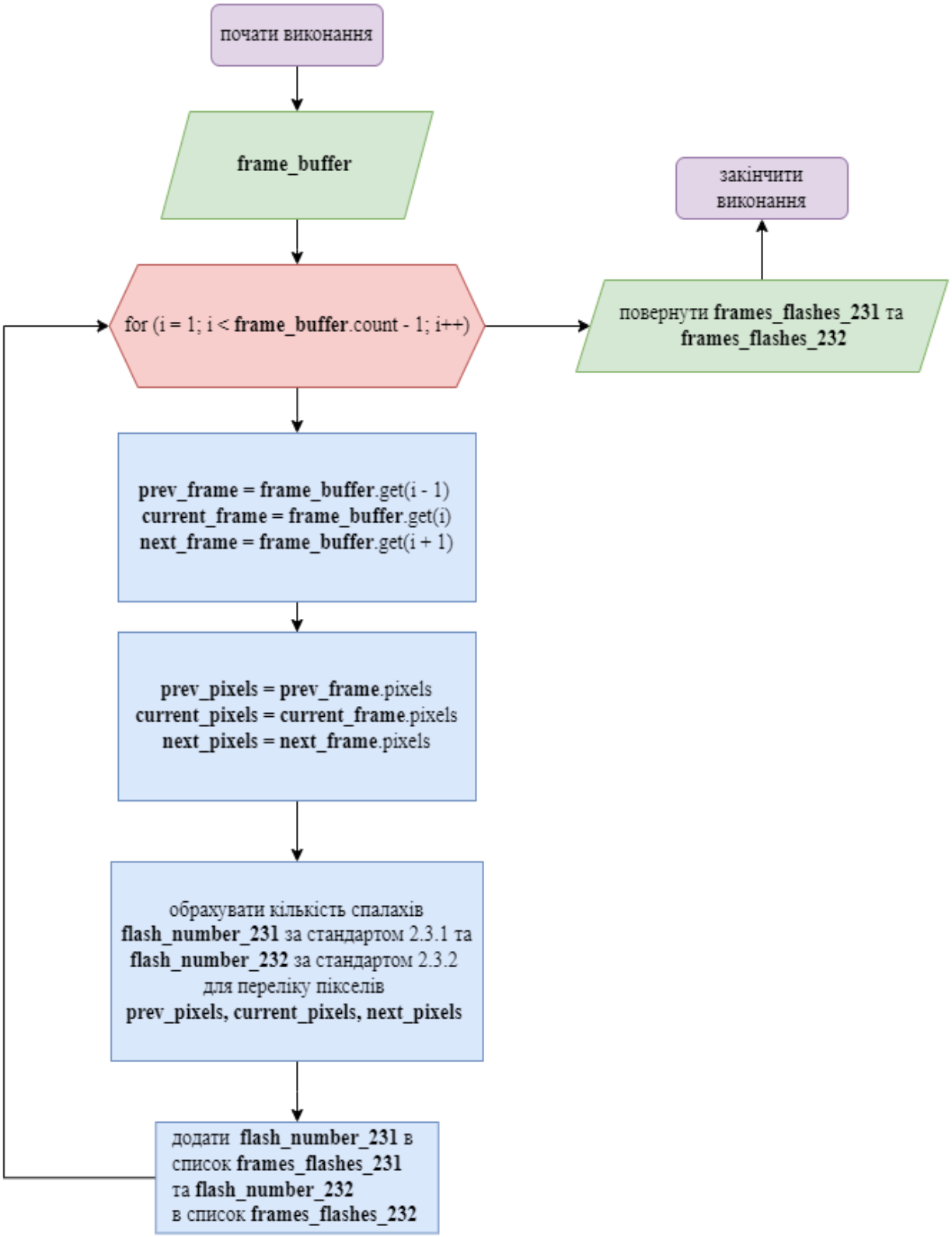
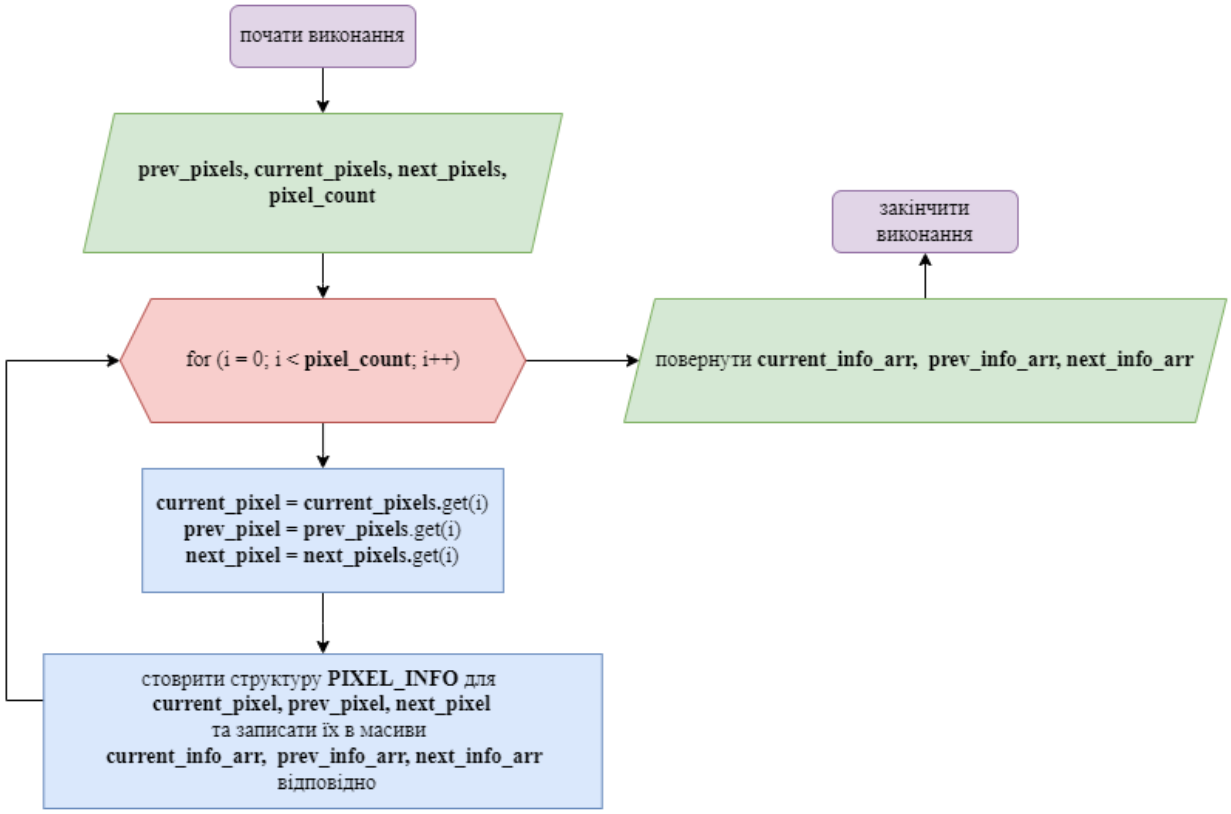

початати виконання
frame_buffer
for (i = 1; i < frame_buffer.count - 1; i++)
повернути frames_flashes_231 та frames_flashes_232
закінчити виконання
prev_frame = frame_buffer.get(i - 1)
current_frame = frame_buffer.get(i)
next_frame = frame_buffer.get(i + 1)
prev_pixels = prev_frame.pixels
current_pixels = current_frame.pixels
next_pixels = next_frame.pixels
обрахувати кількість спалахів flash_number_231 за стандартом 2.3.1 та flash_number_232 за стандартом 2.3.2 для переліку пікселів prev_pixels, current_pixels, next_pixels
додати flash_number_231 в список frames_flashes_231 та flash_number_232 в список frames_flashes_232

Рисунок
В



Рисунок

Для

а) L – відносна яскравість, обрахована за формулою L = *0.2126 * R + 0.7152 * G + 0.0722 * B,* *(2.5)*,

б) R, G, B – лінеаризовані значення моделі RGB обраховані за формулою RsRGB = *R8bit /255,* *(2.6)*;

в) u

, г) sR – значення насиченого червоного кольору, обраховане за формулою $RR+G+B \geq 0.8$, *(2.7)*;

v д) x, y – координати пікселя на екрані.

Алгоритм обрахунку полів структури PIXEL_INFO зображений на рисунку Рисунок 2.19.

к

о

д

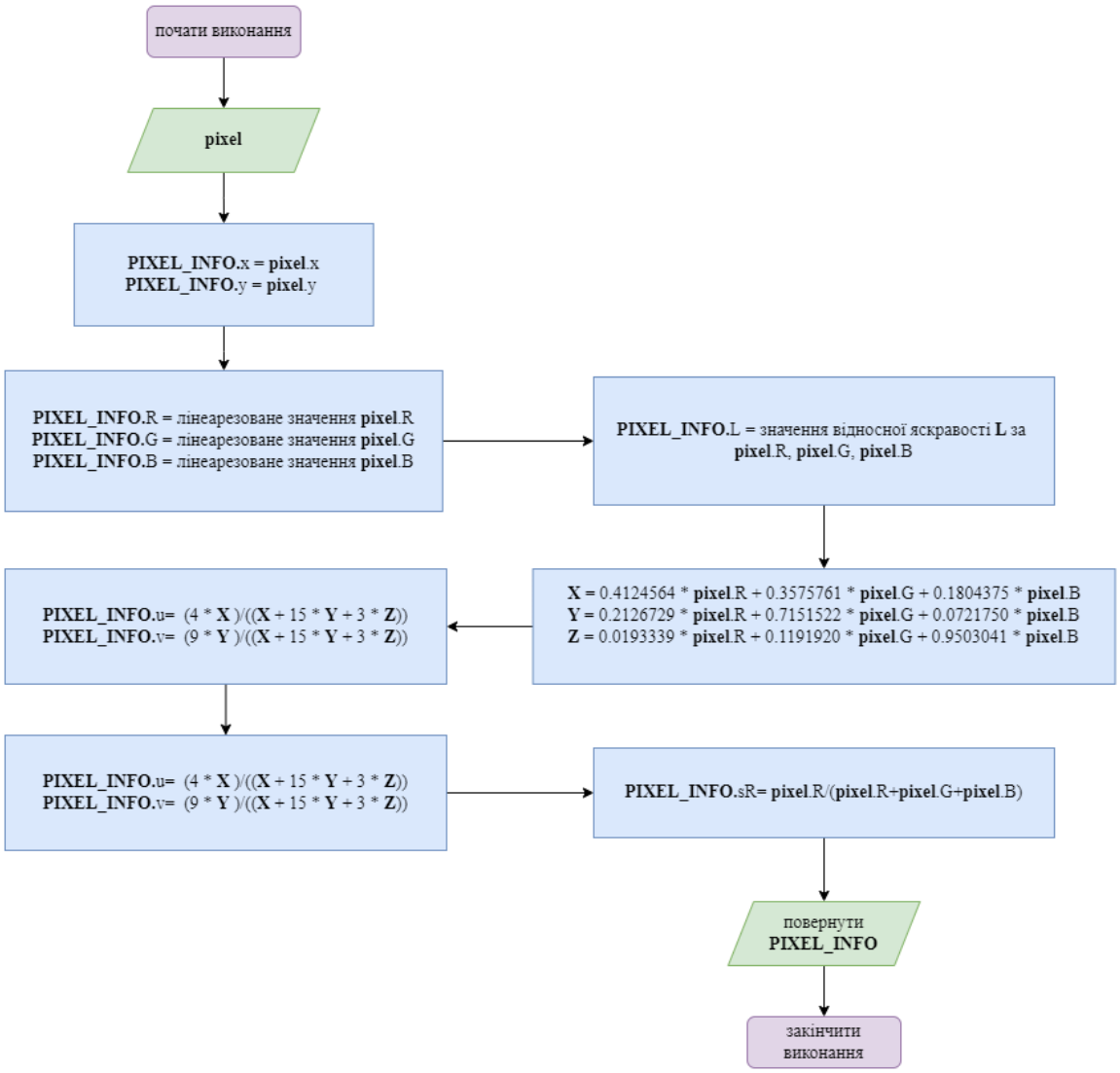


Рисунок 2.19

Структури PIXEL_INFO записуються в масиви current_info_arr, prev_info_arr, next_info_arr для пікселів поточного, попереднього та наступного кадрів відповідно. Наступним кроком є порівняння параметрів кожної трійки структур з цих масивів, як зображено на рисунку Рисунок 2.20.

В структурі PIXEL_FLASH зберігаються координати спалаху, що виник між трійкою пікселів. Ці структури записуються в масив flashes.

Оскільки стандарт 2.3.2 аналізує на наявність будь-яких спалахів, навіть розміром в один піксель, значення , яке виникло між трійкою кадрів, відповідає кількості елементів в списку flashes.

За стандартом 2.3.1 враховуються лише спалахи, що займають відповідну площу. Щоб виявити такі області, потрібно підрахувати кількість спалахів, що виникають в прямокутнику розміром Sw на Sh в будь-якому місці екрану. Для цього для кожного елементу в списку flashes підраховується кількість інших елементів-сусідів в цьому списку, що знаходяться на відстані менші або рівній Sw по абсцисі та меншій або рівній Sh по ординаті. Для того, щоб не аналізувати прямокутник з однаковими координатами повторно, для кожного елемента аналізуються тільки сусіди, що знаходяться нижче та правіше від нього. Якщо кількість таких елементів перевищує чверть прямокутника, спалах буде враховано в подальшому аналізі на частоту. Це зображено на рисунку Рисунок 2.21.

Останнім кроком є перевірка частоти виникнення спалахів, що не має перевищувати трьох на секунду. Масиви frames_flashes_231 та frames_flashes_232 зберігають кількість спалахів, що виникли між кожною трійкою кадрів. Відповідно до значення fps підраховується сума значень елементів цих масивів, що відповідають кадрам, відображеним за одну секунду в будь-який момент часу, як показано на рисунку Рисунок 2.22.

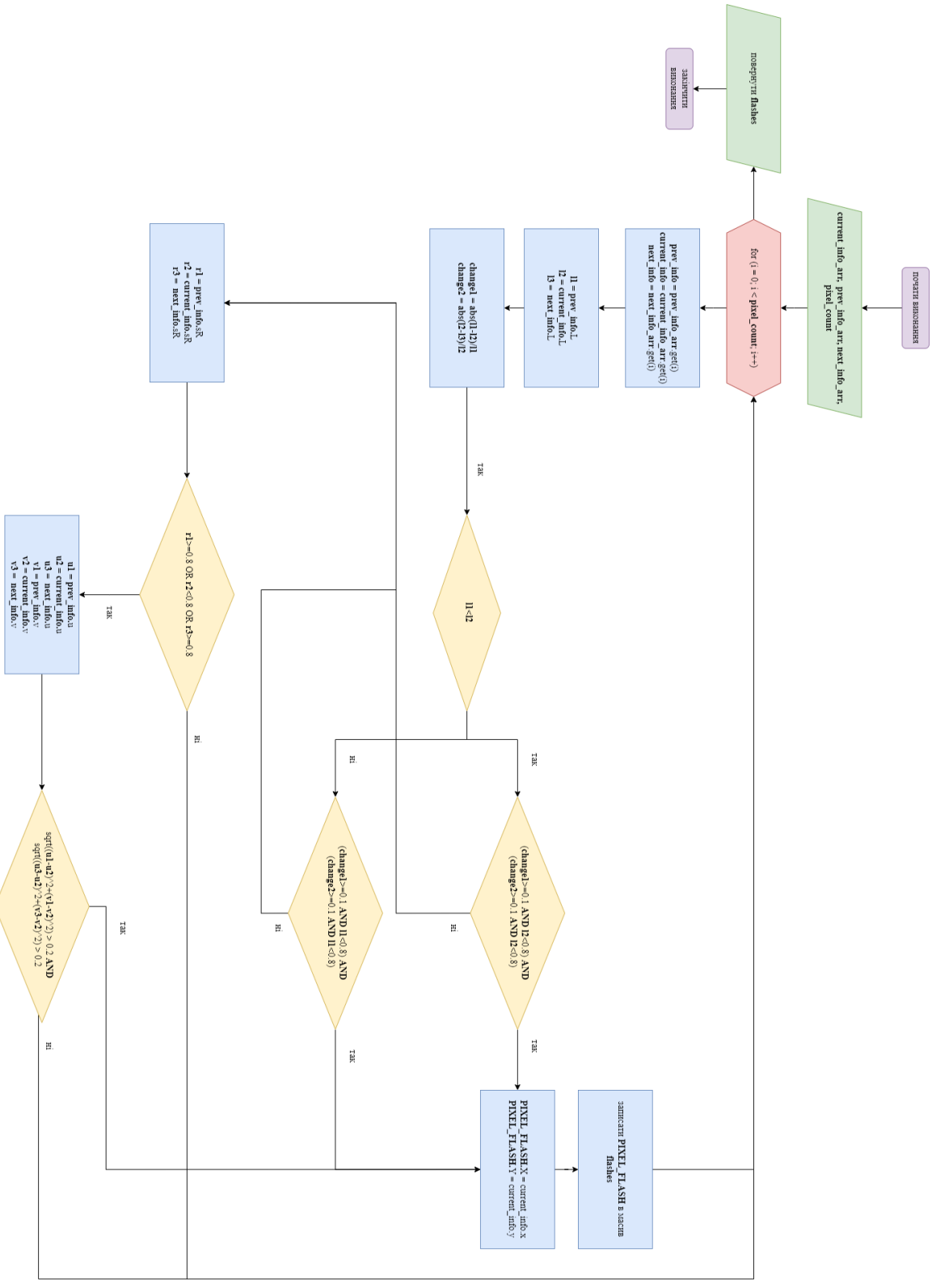

ПОЛУЧИТЬ ВЫПОЛНЕНИЯ
current_info_arr, prev_info_arr, next_info_arr, pixel_count
for (i = 0; i < pixel_count; i++)
prev_info = prev_info_arr.get(i)
current_info = current_info_arr.get(i)
next_info = next_info_arr.get(i)
l1 = prev_info.L
l2 = current_info.L
l3 = next_info.L
change1 = abs(l1-l2)/l1
change2 = abs(l2-l3)/l2
r1 = prev_info.sR
r2 = current_info.sR
r3 = next_info.sR
l1<l2
(change1>=0.1 AND l2<0.8) AND (change2>=0.1 AND l2<0.8)
(change1>=0.1 AND l1<0.8) AND (change2>=0.1 AND l1<0.8)
r1>=0.8 OR r2<0.8 OR r3>=0.8
u1 = prev_info.u
u2 = current_info.u
u3 = next_info.u
v1 = prev_info.v
v2 = current_info.v
v3 = next_info.v
sqrt((u1-u2)^2+(v1-v2)^2) > 0.2 AND sqrt((u3-u2)^2+(v3-v2)^2) > 0.2
PIXEL_FLASH.X = current_info.x
PIXEL_FLASH.Y = current_info.y
записать PIXEL_FLASH в массив flashes
ПОСЛАТЬ flashes
ЗАВЕРШИТЬ ВЫПОЛНЕНИЕ
ДА
НЕТ

Рисунок 2.20

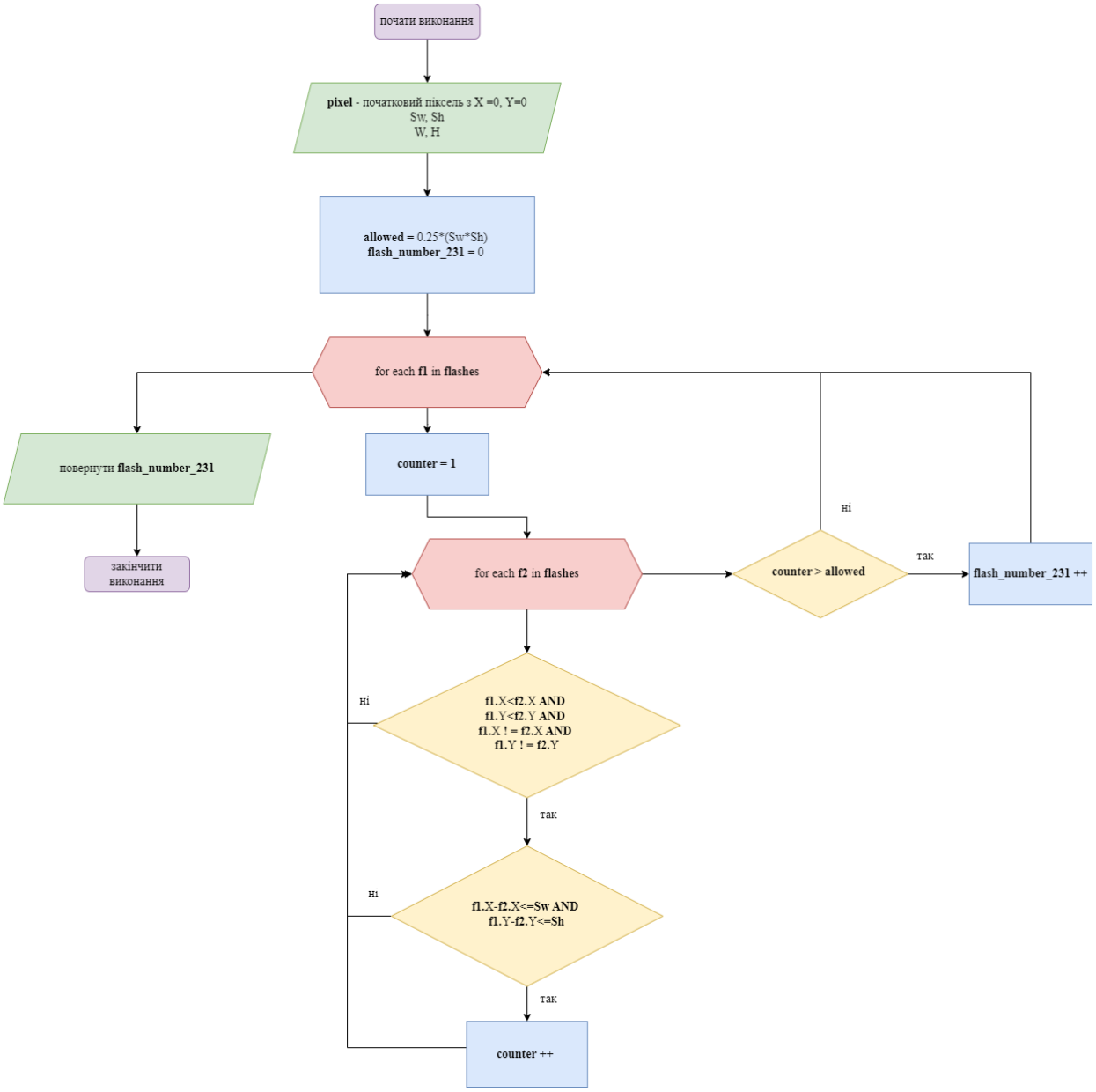

почати виконання
pixel - початковий піксель з X =0, Y=0
Sw, Sh
W, H
allowed = 0.25*(Sw*Sh)
flash_number_231 = 0
for each f1 in flashes
повернути flash_number_231
закінчити виконання
counter = 1
for each f2 in flashes
counter > allowed
ні
так
flash_number_231 ++
ні
f1.X<f2.X AND
f1.Y<f2.Y AND
f1.X ! = f2.X AND
f1.Y ! = f2.Y
так
ні
f1.X-f2.X<=Sw AND
f1.Y-f2.Y<=Sh
так
counter ++

Рисунок 2.21

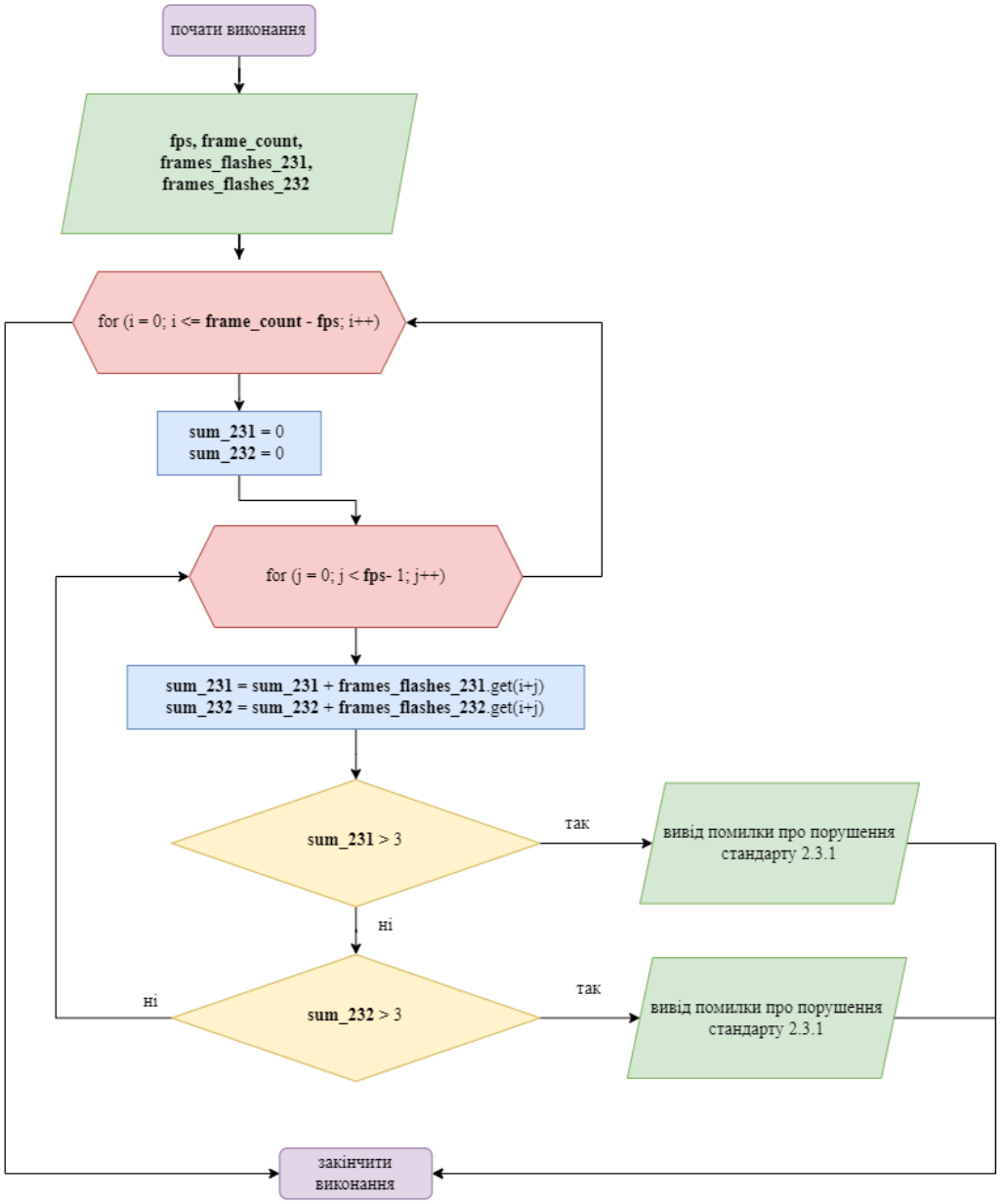


Рисунок 2.22

Суми спалахів обраховуються окремо для стандартів 2.3.1 та 2.3.2. Якщо сума спалахів перевищує три, виводиться повідомлення про порушення відповідного стандарту.

РОЗДІЛ 3: Опис практичної частини та аналіз результатів

3.1 Огляд прикладу імплементації алгоритму

Приклад імплементації алгоритму було створено мовою програмування Java. Він надає можливість наочної демонстрації кроків та результатів тестування, приймаючи на вхід дані, що імітують перелік елементів вікна з відповідними налаштуваннями.

Оскільки алгоритм є абстрагованим від певної платформи розробки, вибір мови програмування для створення прикладу оснований на її зручності використання представлення вхідних даних у вигляді переліку об’єктів.

Для імплементації алгоритму було створено наступні класи, як на рисунку:

а) клас «DescriptionAttributesCheck» – приклад імплементації алгоритму з підрозділу 2.1 Підтримка роботи вбудованих програм інклюзивності;

б) клас «FocusCheck» – приклад імплементації алгоритму з підрозділу 2.2 Підтримка роботи фізичної клавіатури;

в) клас «SmallScreenCheck» – приклад імплементації алгоритму з підрозділу 2.3 Особливості налаштування елементів на екрані маленького розміру;

г) клас «ContrastCheck» – приклад імплементації алгоритму з підрозділу 2.4 Колір. Контраст;

д) клас «FlashDetector» – приклад імплементації алгоритму з підрозділу 2.5 Спалахи. Контраст спалахів. Для покадрового аналізу відео було використано бібліотеку opencv.videoio;

Для створення тестових даних, було створено ієрархію класів «ScreenElement» та клас «DataGenerator», що виконує генерацію тестових

об’єктів, як зображено на UML діаграмах рисунках Рисунок 3.2 та Рисунок 3.3.

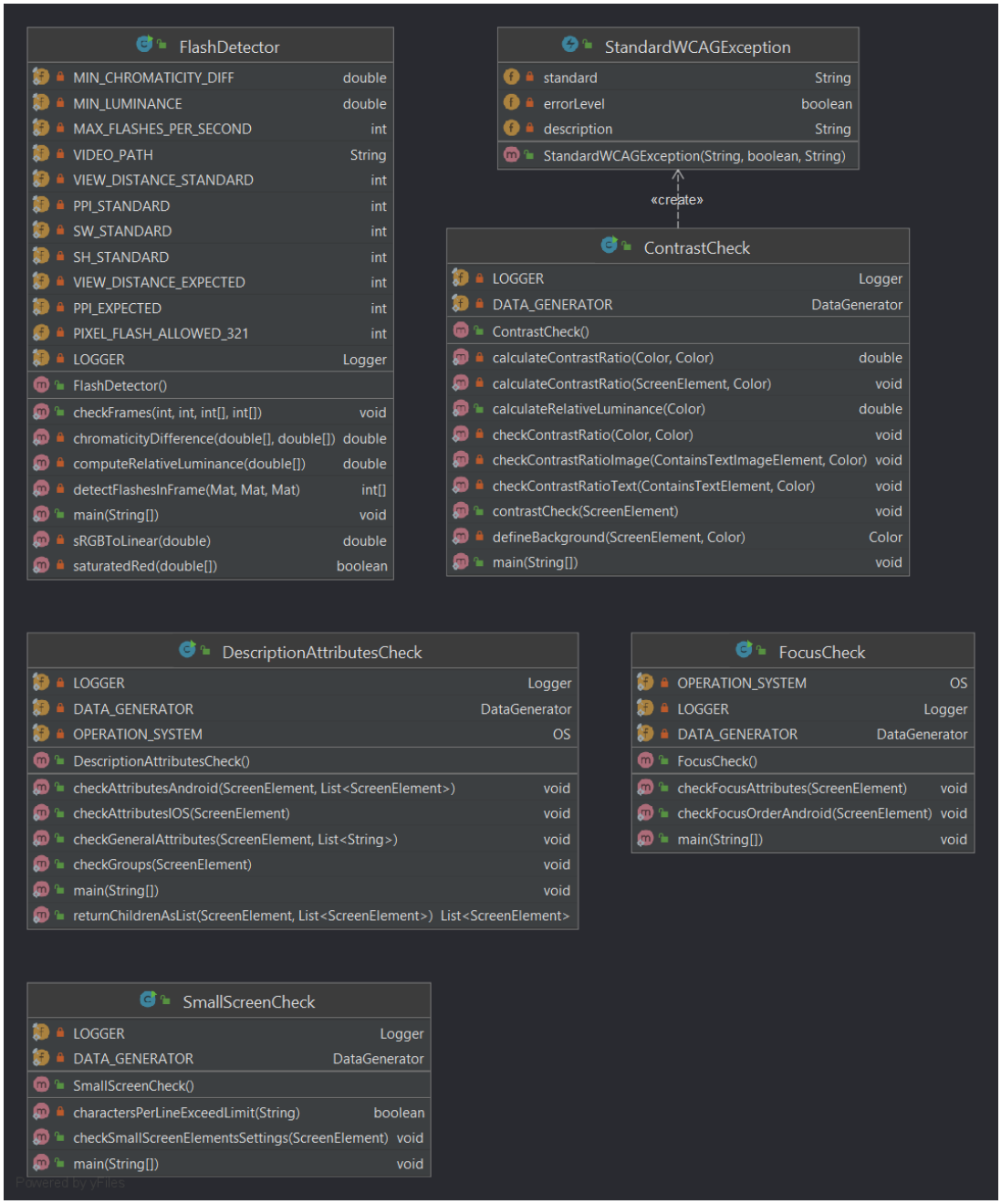


Рисунок 3.1

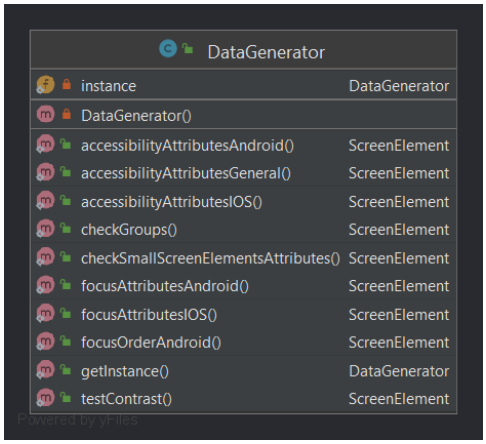


Рисунок 3.2

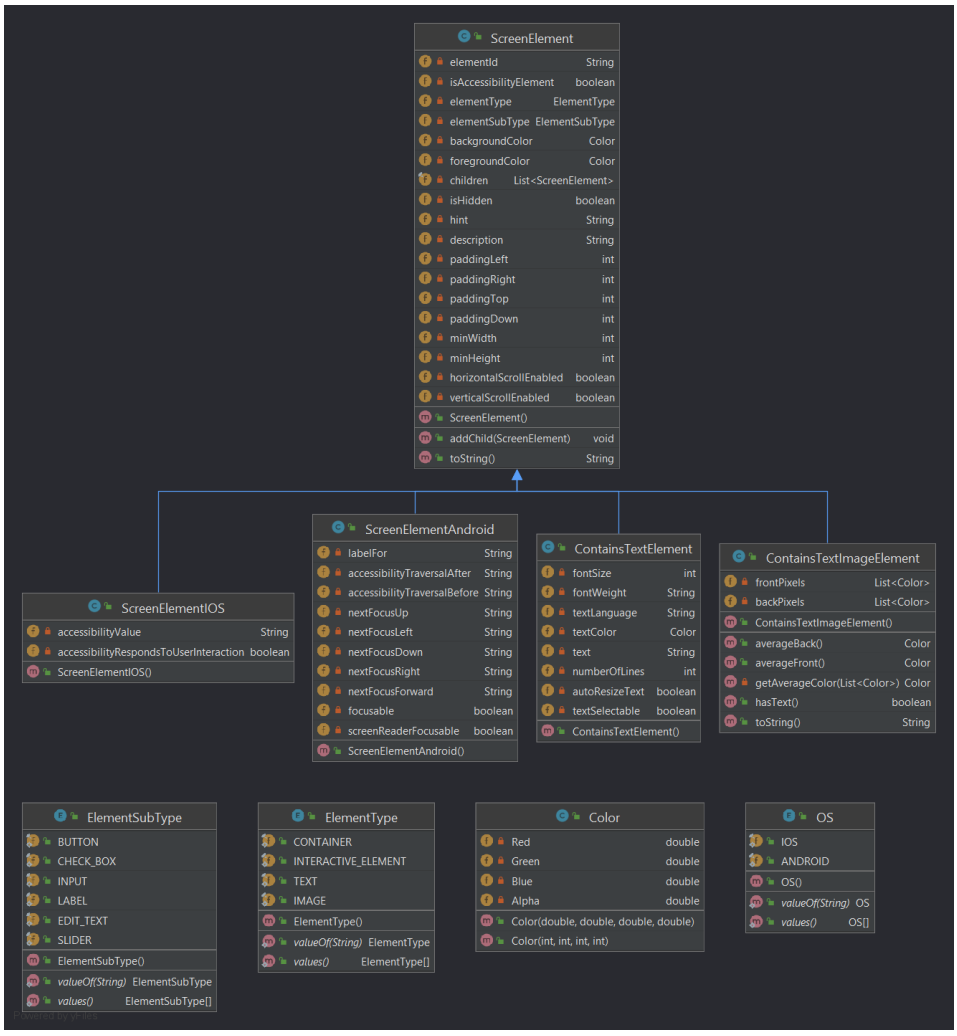


Рисунок 3.3

3.2 Опис результатів прикладу тестування

Для демонстрації роботи алгоритму, було створено наступні тестові випадки з використанням тестових даних та відповідними результатами виконання, що неведені в таблицях додатку Додаток Б Результати прикладу тестування.

1. Тестовий випадок : «Загальні описові атрибути»

Макет: Рисунок Б.1 – макет 1Рисунок Б.1 – макет 1

Вхідні дані: Таблиця Б.1 – вхідні дані для тесту «Загальні описові атрибути»

Результат виконання: Рисунок Б.7 – результат тесту «Загальні описові атрибути»

2. Тестовий випадок : «Описові атрибути Android»

Макет: Рисунок Б.1 – макет 1

Вхідні дані: Таблиця Б.2 – вхідні дані до тесту «Описові атрибути Android»

Результат виконання: Рисунок Б.8 – результат тесту «Описові атрибути Android»

3. Тестовий випадок : «Описові атрибути IOS»

Макет: Рисунок Б.2 – макет 2

Вхідні дані:Таблиця Б.3 – вхідні дані до тесту «Описові атрибути IOS»

Результат виконання:Рисунок Б.9 – результат тесту «Описові атрибути IOS»

4. Тестовий випадок : «Групування елементів»

Макет: Рисунок Б.3 – макет 3

Вхідні дані: Таблиця Б.6 – вхідні дані до тесту «Групування елементів»

Результат виконання:Рисунок Б.10 результат тесту «Групування елементів»

5. Тестовий випадок : «Порядок фокусування Android»

Макет: Рисунок Б.3 – макет 3

Вхідні дані: Таблиця Б.5 – вхідні дані до тесту «Порядок фокусування Android»

Результат виконання: Рисунок Б.11 – результат тесту «Порядок фокусування Android»

6. Тестовий випадок: «Атрибути фокусування. Android\IOS»

Макет: Рисунок Б.1 – макет 1

Вхідні дані: Таблиця Б.4 – вхідні дані до тесту «Атрибути фокусування. Android\IOS»

Результат виконання: Рисунок Б.12 – результат тесту «Атрибути фокусування. Android» Рисунок Б.13 – результат тесту «Атрибути фокусування. IOS»

7. Тестовий випадок: «Специфіка атрибутів маленького екрану»

Макет: Рисунок Б.4 – макет 4

Вхідні дані: Таблиця Б.7 – вхідні дані до тесту «Специфіка атрибутів маленького екрану» Таблиця Б.8 – вхідні дані до тесту «Специфіка атрибутів маленького екрану» Таблиця Б.9 – вхідні дані до тесту «Специфіка атрибутів маленького екрану»

Результат виконання: Рисунок Б.14 – результат тесту «Специфіка атрибутів маленького екрану»

8. Тестовий випадок: «Контраст»

Макет: Рисунок Б.4 – макет 4

Вхідні дані: Таблиця Б.10 – вхідні дані до тесту «Контраст»

Результат виконання:Рисунок Б.15 – результат тесту «Контраст»

9. Тестовий випадок: «Спалахи. Допустимі»

Вхідні дані: відео з почерговою зміною кадрів, що не містять спалахів – посекундна зміна неяскравого таймеру, як показано на рисунку Рисунок Б.6 – вхідні дані до тесту «Спалахи. Допустимі»

Результат виконання: Рисунок Б.16 – результат тесту «Спалахи. Допустимі»

10. Тестовий випадок: «Спалахи. Недопустимі»

Вхідні дані: відео з почерговою зміною кадрів, що містять спалахи, як показано на рисунку Рисунок Б.5 – вхідні дані до тесту «Спалахи. Недопустимі»

Результат виконання: Рисунок Б.17 – результат тесту «Спалахи. Недопустимі» Рисунок Б.18 – результат тесту «Спалахи. Недопустимі»

Висновки по роботі та рекомендації для подальших досліджень

У межах цієї роботи було проведено дослідження стандартів інклюзивності WCAG для веб-застосунків за можливістю їх використання для тестування мобільних додатків. Було розглянуто дев’ять інструментів тестування, що вже присутні на ринку, та проаналізовано їх критерії оцінки, переваги та недоліки.

Спираючись на цей аналіз, критерії тестування було розділено на п’ять головних підкатегорій, до кожної з яких було виокремлено стандарти, що мають виконуватись. Для кожної з підкатегорій було описано покроковий алгоритм тестування у вигляді блок-схем діаграм, наведено приклад імплементації мовою програмування Java та приклади десяти тестових випадків з очікуваним результатом. Сумарно робота охоплює 13 найголовніших стандартів WCAG, що стосуються забезпечення якості та безпеки сприйняття вмісту, як-от контраст, розмір та опис елементів вікна.

Практичним значенням роботи є можливість майбутньої імплементації алгоритму розробниками незалежно від вибору операційної системи застосунку з адаптацією до її потреб.

В подальших дослідження можуть бути розглянуті додаткові можливості забезпечення іклюзивності вмісту, не включені в цю роботу. До них відносяться особливості роботи OCR алгоритмів для визначення рівня контрасту текстових зображень, алгоритми перетворення мови в текст для автоматичного створення субтитрів тощо.

Список використаної літератури

1. Міністерство соціальної політики України. Особам з інвалідністю - Міністерство соціальної політики України. Міністерство соціальної політики України. URL: https://www.msp.gov.ua/timeline/invalidnist.html (дата звернення: 26.04.2023).

2. Accessibility - Foundations - Human Interface Guidelines - Design - Apple Developer. *Apple Developer*. URL: https://developer.apple.com/design/human-interface-guidelines/foundations/accessibility/ (дата звернення: 08.04.2023).

3. accessibilityRespondsToUserInteraction | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/objectivec/nsobject/3043551-accessibilityrespondstouserinter (дата звернення: 29.03.2023).

4. allowsSelection | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/uikit/uitableview/1614911-allowsselection (дата звернення: 30.03.2023).

5. App accessibility for Switch Control - WWDC20 - Videos - Apple Developer. *Apple Developer*. URL: https://developer.apple.com/videos/play/wwdc2020/10019/ (дата звернення: 09.02.2023).

6. Authoring Tool Accessibility Guidelines (ATAG) 2.0. *World Wide Web Consortium (W3C)*. URL: https://www.w3.org/TR/ATAG20/ (дата звернення: 08.04.2023).

7. Autosize TextViews | Android Developers. *Android Developers*. URL: https://developer.android.com/develop/ui/views/text-and-emoji/autosizing-textview (дата звернення: 30.03.2023).

8. Build accessible apps | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/topics/ui/accessibility (дата звернення: 08.04.2023).

9. CIE 1976 UCS Diagram / Color difference formula (CIE 1994) - Part IV - Precise Color Communication | KONICA MINOLTA. *Konica Minolta Deutschland | KONICA MINOLTA*. URL: https://www.konicaminolta.com/instruments/knowledge/color/part4/08.html (дата звернення: 26.02.2023).

10. CIE color spaces. *LuxaLight*. URL: https://www.luxalight.eu/en/cie-color-spaces (дата звернення: 26.02.2023).

11. Color | Android Developers. *Android Developers*. URL: https://developer.android.com/reference/android/graphics/Color#alpha-and-transparency (дата звернення: 11.02.2023).

12. Computing contrast ratio in images using local content information / B. Ortiz-Jaramillo та ін. *2015 20th Symposium on Signal Processing, Images and Computer Vision (STSIVA)*, м. Bogota, Colombia, 2–4 верес. 2015 р. 2015. URL: https://doi.org/10.1109/stsiva.2015.7330429 (дата звернення: 22.04.2023).

13. Control your Android device with Switch Access - Android Accessibility Help. *Google Help*. URL: https://support.google.com/accessibility/android/answer/6122836?hl=en (дата звернення: 09.02.2023).

14. Deque Systems. Accessibility Inspector Tutorial for iOS Native Mobile Apps, 2017. *YouTube*.

URL: https://www.youtube.com/watch?v=EkG5_kWkqwE (дата звернення: 09.04.2023).

15. Get started on Android with TalkBack - Android Accessibility Help. *Google Help*.
URL: https://support.google.com/accessibility/android/answer/6283677 (дата звернення: 09.02.2023).

16. GitHub - dequelabs/axe-android: WCAG Accessibility compliance library for Android Applications. *GitHub*. URL: https://github.com/dequelabs/axe-android (дата звернення: 09.04.2023).

17. GitHub - google/GTXiLib: Google Toolbox for Accessibility for iOS. *GitHub*. URL: https://github.com/google/GTXiLib (дата звернення: 09.04.2023).

18. GitHub - NAB/UBKAccessibilityKit: An iOS framework to help with accessibility development and testing. *GitHub*.
URL: https://github.com/NAB/UBKAccessibilityKit (дата звернення: 09.04.2023).

19. Guidelines for the development of accessible mobile interfaces. *Vi kan digital tillgänglighet - Funka*.
URL: https://www.funka.com/contentassets/9131835638b640cf96baf2ef62a2fba4/guidelines_for_the_development_of_accessible_mobile_interfaces.pdf (дата звернення: 09.02.2023).

20. isSelectable | Apple Developer Documentation. *Apple Developer Documentation*.
URL: https://developer.apple.com/documentation/uikit/uitextview/1618627-isselectable (дата звернення: 30.03.2023).

21. Make apps more accessible | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/topics/ui/accessibility/apps (дата звернення: 09.02.2023).

22. Mobile Flow Analyzer - Evinced, Inc. *Evinced*. URL: https://www.evinced.com/products/flow-analyzer-for-mobile (дата звернення: 09.04.2023).

23. Principles for improving app accessibility | Android Developers. *Android Developers*. URL: https://developer.android.com/guide/topics/ui/accessibility/principles (дата звернення: 09.02.2023).

24. Scaling Fonts Automatically | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/uikit/uifont/scaling_fonts_automatically (дата звернення: 30.03.2023).

25. Support Full Keyboard Access in your iOS app - WWDC21 - Videos - Apple Developer. *Apple Developer*. URL: https://developer.apple.com/videos/play/wwdc2021/10120/ (дата звернення: 29.03.2023).

26. Supporting VoiceOver in your app | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/accessibility/supporting_voiceover_in_your_app (дата звернення: 09.02.2023).

27. Support keyboard navigation | Android Developers. *Android Developers*. URL: https://developer.android.com/develop/ui/views/touch-and-input/keyboard-input/navigation (дата звернення: 29.03.2023).

28. Test your app's accessibility | Android Developers. *Android Developers*.

URL: https://developer.android.com/guide/topics/ui/accessibility/testing (дата звернення: 09.04.2023).

29. TextView | Android Developers. *Android Developers*. URL: https://developer.android.com/reference/android/widget/TextView (дата звернення: 22.04.2023).

30. UITextView | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/uikit/uitextview (дата звернення: 30.03.2023).

31. Understanding Success Criterion 1.3.4: Orientation | WAI | W3C. *World Wide Web Consortium (W3C)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/orientation.html (дата звернення: 30.03.2023).

32. Understanding Success Criterion 1.4.10: Reflow | WAI | W3C. *World Wide Web Consortium (W3C)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/reflow.html (дата звернення: 30.03.2023).

33. Understanding Success Criterion 1.4.3: Contrast (Minimum) | WAI | W3C. *World Wide Web Consortium (W3C)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum.html (дата звернення: 11.02.2023).

34. Understanding Success Criterion 1.4.5: Images of Text | WAI | W3C. *World Wide Web Consortium (W3C)*. URL: https://www.w3.org/WAI/WCAG21/Understanding/images-of-text.html (дата звернення: 11.02.2023).

35. Understanding Success Criterion 1.4.6: Contrast (Enhanced) | WAI | W3C. *World Wide Web Consortium (W3C)*.

URL: https://www.w3.org/WAI/WCAG21/Understanding/contrast-enhanced.html (дата звернення: 11.02.2023).

36. Understanding Success Criterion 1.4.9: Images of Text (No Exception) | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/images-of-text-no-exception (дата звернення: 11.02.2023).

37. Understanding Success Criterion 2.1.1: Keyboard | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/keyboard.html (дата звернення: 29.03.2023).

38. Understanding Success Criterion 2.3.1: Three Flashes or Below Threshold | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/three-flashes-or-below-threshold (дата звернення: 26.02.2023).

39. Understanding Success Criterion 2.3.1 | Understanding WCAG 2.0. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/TR/UNDERSTANDING-WCAG20/seizure-does-not-violate.html (дата звернення: 26.02.2023).

40. Understanding Success Criterion 2.3.2 | Understanding WCAG 2.0. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/TR/UNDERSTANDING-WCAG20/seizure-three-times.html (дата звернення: 26.02.2023).

41. Understanding Success Criterion 2.4.3: Focus Order | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/focus-order (дата звернення: 09.02.2023).

42. Understanding Success Criterion 2.5.5: Target Size | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/target-size.html (дата звернення: 30.03.2023).

43. Understanding Success Criterion 3.3.5: Help | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/help (дата звернення: 09.02.2023).

44. Understanding Success Criterion 4.1.2: Name, Role, Value | WAI | W3C. *World Wide Web Consortium (W3C)*.
URL: https://www.w3.org/WAI/WCAG21/Understanding/name-role-value (дата звернення: 09.02.2023).

45. View | Android Developers. *Android Developers*.
URL: https://developer.android.com/reference/android/view/View#attr_android:screenReaderFocusable (дата звернення: 29.03.2023).

46. (WAI) W. W. A. I. Mobile Accessibility at W3C. *Web Accessibility Initiative (WAI)*. URL: https://www.w3.org/WAI/standards-guidelines/mobile/ (дата звернення: 08.04.2023).

47. (WAI) W. W. A. I. User Agent Accessibility Guidelines (UAAG) Overview. *Web Accessibility Initiative (WAI)*.
URL: https://www.w3.org/WAI/standards-guidelines/uaag/ (дата звернення: 08.04.2023).

48. (WAI) W. W. A. I. WCAG 2 Overview. *Web Accessibility Initiative (WAI)*. URL: https://www.w3.org/WAI/standards-guidelines/wcag/ (дата звернення: 08.04.2023).

49. (WAI) W. W. A. I. Web Accessibility Initiative (WAI). *Web Accessibility Initiative (WAI)*. URL: https://www.w3.org/WAI/ (дата звернення: 08.04.2023).

50. Whitaker R. Developing Inclusive Mobile Apps. Berkeley, CA : Apress, 2020. URL: https://doi.org/10.1007/978-1-4842-5814-9 (дата звернення: 09.02.2023).

51. XCUITests for accessibility | Mobile A11y. *Mobile A11y*. URL: https://mobilea11y.com/guides/xcui/ (дата звернення: 09.04.2023).

Додаток А

Огляд інструментів тестування інклюзивності

Таблиця А.1 Порівняння інструментів тестування інклюзивності частина 1

Назва інструмента тестування	**Тип**	**Операційна система**	**Наявність описових атрибутів**	**Вміст описових атрибутів**
Google GTXiLib	автоматизоване	IOS	так	частково
XCUI Testing	автоматизоване	IOS	так	так
UBKAccessibilityKit	автоматизоване	IOS	так	-
Espresso Testing AccessibilityChecks	автоматизоване	Android	так	-
Accessibility Test Framework	автоматизоване	Android	так	частково
Axe Android	автоматизоване	Android	частково	-
Apple Accessibility Inspector	інспекція	IOS (в середовищі розробки)	так	виокремлює конфлікти між описовими атрибутами
Google Accessibility Scanner	інспекція	Android (після встановлення застосунку)	так	-
Flow Analyzer for Mobile	інспекція	IOS, Android (після встановлення застосунку)	так	частково

Таблиця А.2 Порівняння інструментів тестування інклюзивності частина 2

Назва	**Мінімальний розмір інтерактивних елементів**	**Порядок фокусування/ зчитування**	**Динамічна зміна вигляду тексту**	**Контраст. Колір**	**Присутність спалахів**
Google GTXiLib	так	-	так	тільки з елементом типу "label"	-
XCUI Testing	так	-	-	-	-
UBKAccessibilityKit	так	-	так	так	-
Espresso Testing AccessibilityChecks	-	так	-	так	-
Accessibility Test Framework	так	так	-	так	-
Axe Android	так	-	-	так	-
Apple Accessibility Inspector	так	імітує зчитування	так, але є хибнонегативні результати	так	-
Google Accessibility Scanner	так	-	-	так	-
Flow Analyzer for Mobile	так	-	так	так	-

Додаток Б
Результати прикладу тестування

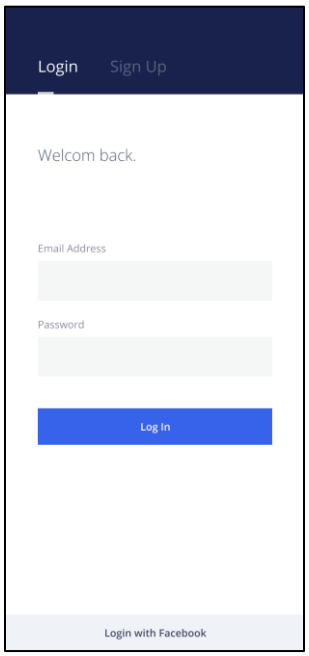


Рисунок Б.1 – макет 1

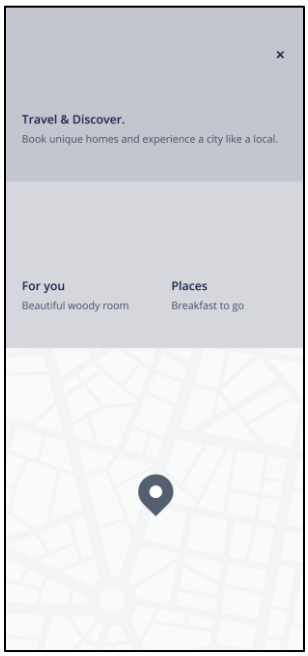


Рисунок Б.3 – макет 3

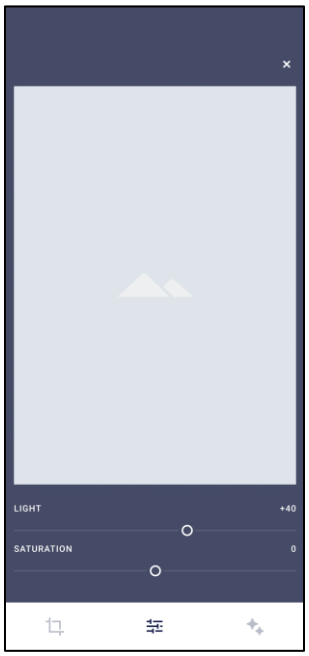


Рисунок Б.2 – макет 2

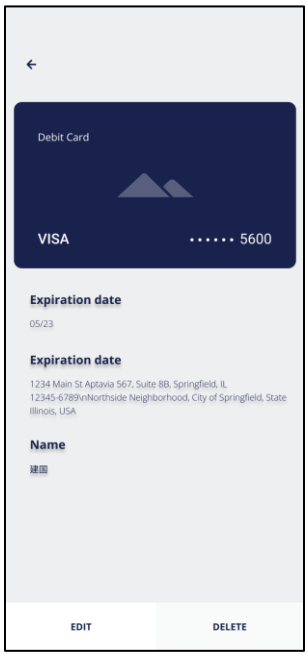


Рисунок Б.4 – макет 4

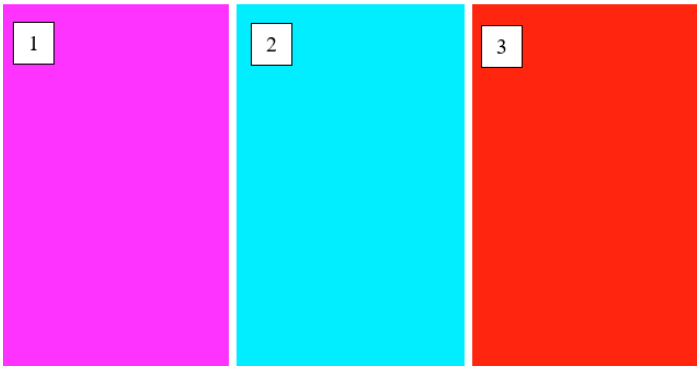


Рисунок Б.5 – вхідні дані до тесту «Спалахи. Недопустимі»

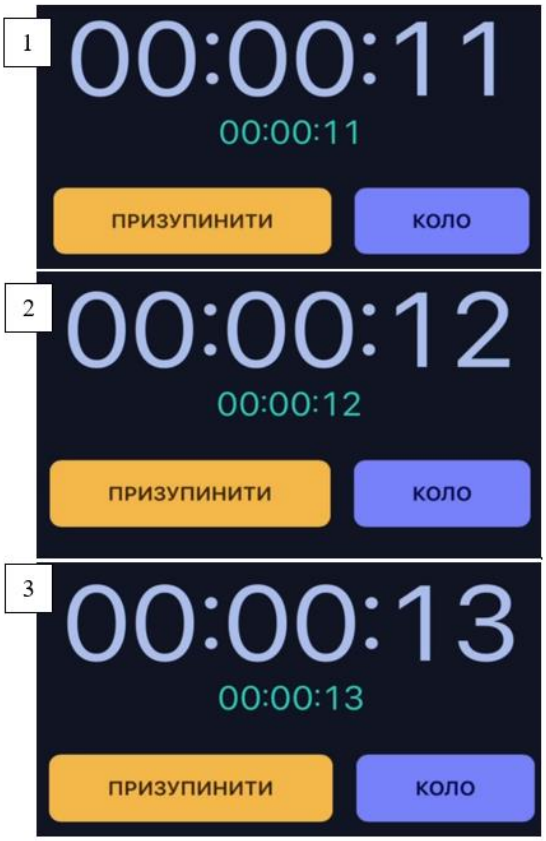


Рисунок Б.6 – вхідні дані до тесту «Спалахи. Допустимі»

Таблиця Б.1 – вхідні дані для тесту «Загальні описові атрибути»

Element	elementId	type	subtype	isAccessibilityElement	hint	description
Кнопка "Login"	loginButtonNav	interactive element	button	TRUE	shows login page	Login
Кнопка "Signup"	signUpButtonNav	interactive element	button	FALSE	-	-
Текстове зображення "Welcome back."	imageWelcome	image	-	TRUE	Shows welcome text.	Welcome back
Текст "Email address"	emailLabel	text	-	TRUE	Email label.	email address.
Поле вводу email	emailInput	interactive element	input	TRUE	Enter your email in this filed email.	Enter email
Текст "Password"	passwordLabel	text	-	TRUE	Enter your password in the filed below.	Password
Поле вводу password	passwordInput	interactive element	input	FALSE	-	-
Кнопка "Log In"	loginButtonBody	interactive element	button	TRUE	Shows login page.	Login
Кнопка "Login with Facebook"	loginWithFacebook	interactive element	button	TRUE	-	-

Таблиця Б.2 – вхідні дані до тесту «Описові атрибути Android»

Element	elementId	type	subtype	labelFor	isHidden
Текст "Email address"	emailLabel	text	-	emailInput	FALSE
Поле вводу email	emailInput	interactive element	input	-	FALSE
Текст "Password"	passwordLabel	text	-	-	FALSE
Поле вводу password	passwordInput	interactive element	input	-	FALSE

Таблиця Б.3 – вхідні дані до тесту «Описові атрибути IOS»

Element	elementId	type	subtype	accessibilityValue
Текст "light"	lightLabel	text	-	-
Слайдер вводу light	lightInput	interactive element	input	40%
Текст "saturation"	saturationLabel	text	-	-
Слайдер вводу saturation	saturationInput	interactive element	input	-

Таблиця Б.4 – вхідні дані до тесту «Атрибути фокусування. Android\IOS»

Element	elementId	type	subtype	focusable \ accessibility RespondsTo UserInteraction	Screen Reader Focusable (для Android)
Кнопка "Login"	loginButtonNav	interactive element	button	TRUE	TRUE
Кнопка "Signup"	signUpButtonNav	interactive element	button	FALSE	FALSE
Текстове зображення "Welcome back."	imageWelcome	image	-	FALSE	FALSE
Текст "Email address"	emailLabel	text	-	FALSE	TRUE
Поле вводу email	emailInput	interactive element	input	TRUE	TRUE
Текст "Password"	passwordLabel	text	-	TRUE	TRUE
Поле вводу password	passwordInput	interactive element	input	TRUE	TRUE
Кнопка "Log In"	loginButtonBody	interactive element	button	TRUE	FALSE
Кнопка "Login with Facebook"	loginWithFacebook	interactive element	button	TRUE	TRUE

Таблиця Б.5 – вхідні дані до тесту «Порядок фокусування Android»

Element	elementId	type	accessibility Traversal After	accessibility Traversal Before	nextFocus Up	nextFocus Down	nextFocus Left	nextFocus Right	nextFocus Forward
Верхній контейнер	container Top	container	-	container Middle	-	button Close	-	buttonClose	buttonClose
Середній контейнер	container Middle	container	-	container Bottom	-	textMiddle 1	-	textMiddle 1	textMiddle 1
Нижній контейнер	container Bottom	container	containerBottom	-	textMiddle 4	-	-	-	-
Кнопка "Travel & discount"	button Close	interactive element	-	textTop1	-	textTop1	-	textTop1	textTop1
Текст "Travel & discount"	textTop1	text	-	textTop2	-	textTop2	-	textTop2	textTop2
Текст "Book uinque..."	textTop2	text	textTop1	-	textTop1	-	textTop1	-	textMiddle 1
Текст "For you"	textMiddle 1	text	-	textMiddle2	-	textMiddle 2	-	textMiddle 2	textMiddle 2
Текст "Beautiful woodly..."	textMiddle 2	text	textMiddle1	-	-	textMiddle 3	-	textMiddle 3	textMiddle 3
Текст "Places"	textMiddle 3	text	-	textMiddle4	-	-	-	-	-
Текст "Breakfast to go"	textMiddle 4	text	textMiddle3	-	-	-	-	-	-

Таблиця Б.6 – вхідні дані до тесту «Групування елементів»

Element	elementId	type	focusable (для Android)	Screen Reader Focusable (для Android)	isAccessibility Element
Верхній контейнер	containerTop	container	FALSE	TRUE	FALSE
Середній контейнер	containerMiddle	container	FALSE	TRUE	FALSE
Нижній контейнер	containerBottom	container	FALSE	FALSE	FALSE
Кнопка "Travel & discount"	buttonClose	interactive element	TRUE	TRUE	TRUE
Текст "Travel & discount"	textTop1	text	FALSE	TRUE	TRUE
Текст "Book uinque..."	textTop2	text	FALSE	TRUE	TRUE
Текст "For you"	textMiddle1	text	TRUE	TRUE	TRUE
Текст "Beautiful woodly..."	textMiddle2	text	TRUE	TRUE	TRUE
Текст "Places"	textMiddle3	text	TRUE	TRUE	TRUE
Текст "Breakfast to go"	textMiddle4	text	TRUE	TRUE	TRUE

Таблиця Б.7 – вхідні дані до тесту «Специфіка атрибутів маленького екрану»

Element	element Id	type	subtype	min Width	min Height	pad Left	pad Right	pad Top	pad Bottom
Кнопка "Back arrow"	button Back	interactive element	button	23	23	0	0	0	0
Кнопка "Edit"	button Edit	interactive element	button	188	60	5	0	5	0
Кнопка "Delete"	utton Delete	interactive element	button	188	60	0	5	4	0

Таблиця Б.8 – вхідні дані до тесту «Специфіка атрибутів маленького екрану»

Element	elementId	type	minWidth	minHeight	Horizontal Scroll	Vertical Scroll
Зображення "Credit card"	imageCard	image	355	210	TRUE	TRUE

Таблиця Б.9 – вхідні дані до тесту «Специфіка атрибутів маленького екрану»

Element	**element Id**	**type**	**subtype**	**selectable**	**autoResize Text**	**length**	**numberOf Lines**
Текст "Expiration date"	label1	text	label	TRUE	TRUE	15	1
Текст "05/23"	text1	text	-	FALSE	TRUE	5	1
Текст "Billing Address"	label2	text	label	TRUE	TRUE	15	1
Текст "1234 West…."	text2	text	-	TRUE	FALSE	127	2

Таблиця Б.10 – вхідні дані до тесту «Контраст»

Element	elementId	type	subtype	backgroundColor	textColor	fontSize	fontWeight	textLanguage	isAccessibilityElement
Текст "Expiration date"	label1	text	label	NULL	rgba(31,40,81,255)	14	bold	English	TRUE
Текст "05/23"	text1	text	-	NULL	rgba(110,115,139,255)	17	regular	English	TRUE
Текст "Billing Address"	label2	text	label	NULL	rgba(31,40,81,255)	14	bold	English	TRUE
Текст "1234 Main…."	text2	text	-	NULL	rgba(110,115,139,255)	17	regular	English	TRUE
Текст "Name"	label3	text	label	NULL	rgba(31,40,81,255)	14	bold	English	TRUE
Текст "建国"	text3	text	-	NULL	rgba(110,115,139,255)	17	regular	Chinese	TRUE
Кнопка "Back arrow"	buttonBack	interactive element	button	NULL	rgba(39,47,86,255)	25	regular	English	TRUE
Кнопка "Edit"	buttonEdit	interactive element	button	rgba(255,255,255,255)	rgba(39,47,86,255)	20	regular	English	TRUE
Кнопка "Delete"	buttonDelete	interactive element	button	rgba(255,255,255,255)	rgba(39,47,86,255)	20	regular	English	TRUE
Вікно	window	container	-	rgba(238,239,241,255)	-	-	-	-	FALSE
Текстове зображення	imageText	container	-	rgba(88,95,124,255), rgba(25,34,77,255), rgb(140,148,167,255)	rgba(250,249,250,255)	-	-	-	TRUE

```
01:20:35.104 WARN  algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @70 - Hint does not match format recommendations. Should match regex: "^[A-Z].*\
 .$" . Element:loginButtonNav
01:20:35.107 ERROR algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @51 - Interactive or text element must be important for accessibility.
 Element:signUpButtonNav
01:20:35.108 WARN  algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @61 - Violates WCAG standard 1.4.9: Images of text are only used for pure
 decoration. Element:imageWelcome
01:20:35.109 WARN  algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @74 - Hint does not match format recommendations. Should not contain element name
 (type). Element:emailLabel
01:20:35.109 WARN  algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @82 - Description does not match format recommendations. Should match regex:
 "^[A-Z][^\p{P}]{0,68}[^\p{P}]$" . Element:emailLabel
01:20:35.109 ERROR algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @51 - Interactive or text element must be important for accessibility.
 Element:passwordInput
01:20:35.110 ERROR algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @91 - Description of the element must be unique. Element:loginButtonBody
01:20:35.110 ERROR algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @66 - Hint is missing. Element:loginWithFacebook
01:20:35.110 ERROR algorythm.accessibility_programs.AttributesCheck.checkGeneralAttributes() @78 - Description is missing. Element:loginWithFacebook
```

Рисунок Б.7 – результат тесту «Загальні описові атрибути»

```
01:20:35.125 ERROR algorythm.accessibility_programs.AttributesCheck.checkAttributesAndroid() @136 - Label for input element is missing. Element:passwordInput
```

Рисунок Б.8 – результат тесту «Описові атрибути Android»

```
01:20:35.132 ERROR algorythm.accessibility_programs.AttributesCheck.checkAttributesIOS() @154 - Accessibility value for input element is missing. Element:saturationInput
```

Рисунок Б.9 – результат тесту «Описові атрибути IOS»

```
03:18:33.691 WARN  algorythm.accessibility_programs.AttributesCheck.checkGroups() @189 - More than two important for accessibility elements found in container. A group might
 be needed. Element:containerTop
03:18:33.691 WARN  algorythm.accessibility_programs.AttributesCheck.checkGroups() @189 - More than two important for accessibility elements found in container. A group might
 be needed. Element:containerMiddle
```

Рисунок Б.10 результат тесту «Групування елементів»

```
01:38:01.624 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @59 - Missing at least one attribute of reading order. Element:textMiddle3
01:38:01.625 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @62 - Missing at least one attribute of vertical navigation order.
 Element:textMiddle3
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @65 - Missing at least one attribute of horizontal navigation order.
 Element:textMiddle3
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @68 - Missing attribute of next navigation order. Element:textMiddle3
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @62 - Missing at least one attribute of vertical navigation order.
 Element:textMiddle4
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @65 - Missing at least one attribute of horizontal navigation order.
 Element:textMiddle4
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @68 - Missing attribute of next navigation order. Element:textMiddle4
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @65 - Missing at least one attribute of horizontal navigation order.
 Element:containerBottom
01:38:01.626 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusOrderAndroid() @68 - Missing attribute of next navigation order. Element:containerBottom
```

Рисунок Б.11 – результат тесту «Порядок фокусування Android»

```
02:16:09.732 ERROR algorythm.accessibility_programs.AttributesCheck.checkFocusAttributes() @48 - Important for accessibility element has to ne screen reader focusable.
 Element:imageWelcome
02:16:09.732 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusAttributes() @45 - Non-interactive element is focusable. Element:passwordLabel
02:16:09.732 ERROR algorythm.accessibility_programs.AttributesCheck.checkFocusAttributes() @48 - Important for accessibility element has to ne screen reader focusable.
 Element:loginButtonBody
```

Рисунок Б.12 – результат тесту «Атрибути фокусування. Android»

```
02:17:36.574 WARN  algorythm.accessibility_programs.AttributesCheck.checkFocusAttributes() @39 - Non-interactive element is responsive to user interactions.
 Element:passwordLabel
```

Рисунок Б.13 – результат тесту «Атрибути фокусування. IOS»

```
19:03:00.999 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.checkSmallScreenElementsSettings() @28 - Interactive element does not meet requirements for minimal target size. Violates standard 2.5.5. Manual check required. Element:buttonBack
19:03:01.000 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.checkSmallScreenElementsSettings() @43 - Text element is not selectable. Element:text1
19:03:01.001 ERROR algorythm.accessibility_programs.DescriptionAttributesCheck.checkSmallScreenElementsSettings() @39 - Text element does not support dynamic size type. Element:text2
19:03:01.002 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.checkSmallScreenElementsSettings() @48 - Vertical scroll enabled for height less than 256. Violates standard 1.4.10. Element:imageCard
```

Рисунок Б.14 – результат тесту «Специфіка атрибутів маленького екрану»

```
20:19:25.733 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.checkContrastRatioImage() @94 - Violation of standard 1.4.9:  Images of text are only used for pure decoration. Element:imageCard
20:19:25.735 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.checkContrastRatioImage() @96 - Manual check for 1.4.6 required: image can be logo. Element:imageCard
20:19:25.743 ERROR algorythm.accessibility_programs.DescriptionAttributesCheck.calculateContrastRatio() @37 - Violation of standard: 1.4.6. Text does not meet color contrast ratio requirements. Element: text1
20:19:25.743 ERROR algorythm.accessibility_programs.DescriptionAttributesCheck.calculateContrastRatio() @37 - Violation of standard: 1.4.6. Text does not meet color contrast ratio requirements. Element: text2
20:19:25.744 WARN  algorythm.accessibility_programs.DescriptionAttributesCheck.calculateContrastRatio() @39 - Violation of standard: 1.4.5. Manual check is required for hieroglyphic languages. Element: text3
```

Рисунок Б.15 – результат тесту «Контраст»

```
---------- VIDEO INFO -------------
Num of frames: 122
FPS: 29
Width: 640 Height: 368
WCAG 3.2.1 flash size threshold: Sw:341 Sh:256 total:87296
----------------------------------
Frame: 0 General occurred: 5 Red occurred: 0
Frame: 1 General occurred: 0 Red occurred: 0
Frame: 2 General occurred: 14 Red occurred: 0
Frame: 3 General occurred: 0 Red occurred: 0
Frame: 4 General occurred: 0 Red occurred: 0
Frame: 5 General occurred: 0 Red occurred: 0
Frame: 6 General occurred: 0 Red occurred: 0
Frame: 7 General occurred: 0 Red occurred: 0
Frame: 8 General occurred: 0 Red occurred: 0
Frame: 9 General occurred: 0 Red occurred: 0
Frame: 10 General occurred: 0 Red occurred: 0
Frame: 11 General occurred: 0 Red occurred: 0
Frame: 12 General occurred: 1 Red occurred: 0
Frame: 13 General occurred: 0 Red occurred: 0
Frame: 14 General occurred: 3 Red occurred: 0
Frame: 15 General occurred: 21 Red occurred: 0
```

Рисунок Б.16 – результат тесту «Спалахи. Допустимі»

```
---------- VIDEO INFO -------------
Num of frames: 33
FPS: 30
Width: 464 Height: 848
WCAG 3.2.1 flash size threshold: Sw:341 Sh:256 total:87296
----------------------------------
Frame: 0 General occurred: 393472 Red occurred: 0
Frame: 1 General occurred: 393472 Red occurred: 393472
Frame: 2 General occurred: 393472 Red occurred: 393472
Frame: 3 General occurred: 393472 Red occurred: 0
Frame: 4 General occurred: 0 Red occurred: 0
Frame: 5 General occurred: 393472 Red occurred: 0
Frame: 6 General occurred: 393472 Red occurred: 393472
Frame: 7 General occurred: 393472 Red occurred: 393472
Frame: 8 General occurred: 393472 Red occurred: 0
Frame: 9 General occurred: 0 Red occurred: 0
Frame: 10 General occurred: 393472 Red occurred: 0
Frame: 11 General occurred: 393472 Red occurred: 393472
Frame: 12 General occurred: 393472 Red occurred: 393472
Frame: 13 General occurred: 393472 Red occurred: 0
Frame: 14 General occurred: 393472 Red occurred: 0
Frame: 15 General occurred: 393472 Red occurred: 393472
```

Рисунок Б.17 – результат тесту «Спалахи. Недопустимі»

```
13:26:59.339 ERROR algorythm.contrast.FlashDetector.checkFrames() @115 - Frame:0 2.3.1 Flashes more than 3 times per second. Flashes higher threshold. General flash: 24 Red
 flash: 12
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @118 - Frame:0 2.3.2 Flashes more than 3 times per second. General flash: 25 Red flash: 13
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @115 - Frame:1 2.3.1 Flashes more than 3 times per second. Flashes higher threshold. General flash: 24 Red
 flash: 12
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @118 - Frame:1 2.3.2 Flashes more than 3 times per second. General flash: 25 Red flash: 13
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @115 - Frame:2 2.3.1 Flashes more than 3 times per second. Flashes higher threshold. General flash: 24 Red
 flash: 12
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @118 - Frame:2 2.3.2 Flashes more than 3 times per second. General flash: 25 Red flash: 13
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @115 - Frame:3 2.3.1 Flashes more than 3 times per second. Flashes higher threshold. General flash: 24 Red
 flash: 12
13:26:59.341 ERROR algorythm.contrast.FlashDetector.checkFrames() @118 - Frame:3 2.3.2 Flashes more than 3 times per second. General flash: 25 Red flash: 13
```

Рисунок Б.18 – результат тесту «Спалахи. Недопустимі»

Додаток В

Перелік прийнятих скорочень

ATAG – Authoring Tool Accessibility Guidelines

CI/CD - continuous integration / continuous delivery

CSS – Cascading Style Sheets

dp – density-independent pixels

fps – frames per second

HTML – HyperText Markup Language

OCR – optical character recognition

PPI – pixel per inch

RGBA – Red, Green, Blue, Alpha

UAAG – User Agent Accessibility Guidelines

UML – Unified Modeling Language

WAI – Web Accessibility International

WCAG – Web Content Accessibility Guidelines