Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет прикладних інформаційних технологій та електроінженерії
(назва факультету)
Автоматизації технологічних процесів і виробництв
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної роботи

магістр
(освітній ступінь)

на тему: «Розробка та дослідження програмного модуля для автоматизованого створення макросів та чат-ботів»

Виконав: студент (ка) 6 курсу, групи КАм-61
спеціальності 151
“Автоматизація та комп’ютерно-інтегровані технології”
(шифр і назва спеціальності (напряму підготовки))

_____ Дем’янюк Т.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Савків В.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Козбур І.Р.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Савків В.Б.
(підпис) (прізвище та ініціали)

Рецензент _____ Стухляк Д.П.
(підпис) (прізвище та ініціали)

м. Тернопіль – 2022

АНОТАЦІЯ

Кваліфікаційна робота магістра складається з пояснювальної записки та графічної частини (ілюстративний матеріал – слайди).

Об’єм графічної частини роботи становить ___ слайдів.

Об’єм пояснювальної записки складає ___ друкованих сторінок формату А4 (210×297), об’єм додатків – ___ друкованих сторінок формату А4.

Робота складається з шести розділів, в яких нараховується ___ рисунків та ____ таблиць з даними.

В роботі використано ___ літературних джерел.

У роботі було розроблено програмний модуль для автоматизованого створення макросів та чат-ботів на базі об’єктно-орієнтованої мови програмування Python. Були побудовані алгоритми для створення макросів та ботів. Також у розробленому програмному модулю було досліджено параметри швидкості виконання макросів та точності ботів, під час їх пошуку об’єктів на екрані.

Ключові слова: МАКРОС, ПРОГРАМНИЙ МОДУЛЬ, БОТ, БІБЛІОТЕКА, ІНТЕРФЕЙС, ПОШУК, ОБ’ЄКТ.

ЗМІСТ

ВСТУП

У комп’ютерному програмуванні макрос (скорочення від «макроінструкція» від грецького μακρο — «довгий, великий») — це правило або шаблон, який визначає, як певний вхідний сигнал має бути зіставлений із замінним виходом. Застосування макросу до вхідних даних називається розширенням макросу. Вхід і вихід можуть бути послідовністю лексичних лексем або символів або синтаксичним деревом. Символьні макроси підтримуються програмними додатками, щоб полегшити виклик звичайних послідовностей команд.

Макроси використовуються, щоб зробити послідовність обчислювальних інструкцій доступною програмісту як єдиний оператор програми, що робить завдання програмування менш виснажливим і менш схильним до помилок. Тому їх називають «макросами», оскільки «великий» блок коду можна розширити з «маленької» послідовності символів. Макроси часто допускають позиційні або ключові параметри, які диктують, що генерує програма умовного асемблера, і використовувалися для цього. Створювати цілі програми або пакети програм відповідно до таких змінних, як операційна система, платформа чи інших факторів. Термін походить від «макроінструкцій» і такі розширення спочатку використовувалися для створення коду мови асемблера.

У середині 1950-х років, коли програмування на мові асемблера зазвичай використовувалося для написання програм для цифрових комп’ютерів, використання макрокоманд було розпочато з двома основними цілями: щоб зменшити кількість програмного коду, який потрібно було написати, генеруючи кілька операторів мови асемблера з однієї макроінструкції та для забезпечення дотримання стандартів написання програм, наприклад, визначення команд введення/виведення стандартними способами. Макроінструкції фактично були середнім кроком між програмуванням на мові асемблера та мовами

програмування високого рівня, такими як FORTRAN і COBOL. Дві з найперших інсталяцій програмування для розробки «макро-мов» для комп’ютера IBM 705 були в Dow Chemical Corp. у Делавері та Air Material Command, Управління логістики балістичних ракет у Каліфорнії.

Функціонал макросів можна покращити за допомогою ботів. Бот (англ. bot, скорочення від robot) — віртуальний робот або штучний інтелект, який функціонує на основі спеціальної програми, що виконує автоматично та/або за заданим розкладом будь-які дії через інтерфейси, призначені для людей. Зазвичай боти призначаються до виконання роботи, одноманітною і повторюваної, з максимально можливою швидкістю (очевидно, набагато вище можливостей людини).

Боти знаходять також застосування в умовах, коли потрібна найкраща реакція в порівнянні з можливостями людини (наприклад, ігрові боти, боти для інтернет-аукціонів тощо) або, рідше, для імітації дій людини (наприклад, боти для чатів тощо). Чат-бот може видати досить адекватну відповідь на питання, сформульоване правильною будь-якою мовою, робота з якою підтримується чат-ботом. Такі роботи часто застосовуються для повідомлення прогнозу погоди, результатів спортивних змагань, курсів валют, біржових котирувань, тощо. Вони знаходять застосування, наприклад, в системі SmarterChild AOL Instant Messenger і MSN messenger.

1. АНАЛІТИЧНА ЧАСТИНА

1.1. Опис макросів, їх переваги та недоліки

Макроси та їх використання

Макрос — це автоматична послідовність введення, яка імітує натискання клавіш або дії миші. Макрос зазвичай використовується для заміни повторюваних дій клавіатури та миші та часто використовується в електронних таблицях і програмах обробки текстів, таких як MS Excel і MS Word. Файл макросу зазвичай має розширення .MAC.

Концепція макросів також добре відома гравцям MMORPG (масові багатокористувацькі онлайн-рольові ігри) і спеціалістам з пошукової оптимізації (SEO). У світі програмування макроси — це сценарії програмування, які використовуються розробниками для повторного використання коду.

Термін «***макрос***» розшифровується як «макрокоманда» (довга інструкція). Запустивши макрос, користувачі можуть скоротити час, який зазвичай витрачається на повторювані завдання. Деякі макроси, наприклад у MS Excel, також можуть містити *функції*. Макрос Excel зазвичай створюється шляхом запису послідовності дій клавіатури та миші за допомогою засобу запису макросів. Його також можна створити за допомогою Visual Basic (оскільки навіть записаний макрос складається з коду Visual Basic).

Потім доступ до збереженого макросу можна отримати зі списку меню або з панелі інструментів і запустити, просто клацнувши. Ви також можете призначити гарячу клавішу макросу для ще швидшого доступу. Оскільки

макроси можна викликати автоматично, разом із запуском документа - вони використовувалися зловмисниками для створення макро-вірусів.

Макроси у відеоіграх

Макроси були широко популяризовані у відеоіграх за допомогою масових багатокористувацьких онлайн-рольових ігор (MMORPG) після появи Ultima Online у 1997 році. Спочатку вони використовувалися для того, щоб дозволити персонажу відеогри виконувати повторювані дії з метою тренування певних навичок у грі, коли гравець не сидить за комп’ютером та знаходиться далеко від клавіатури.

Проте в наступні роки макроси використовувалися як комбінації клавіш або миші в інших іграх. Замість натискання кількох клавіш у певному порядку, щоб активувати вміння персонажа, можна використати макрос для виконання цієї дії за допомогою однієї кнопки. У деяких іграх, таких як Fortnite або інших шутерах від першої особи (FPS), де час має суттєве значення, використання макросів часто розглядається як форма шахрайства (чітерства), особливо в змагальних іграх. Їх можна використати для безпомилкового виконання дій, які є надто складними для людини, щоб виконати їх вручну за такий же час в мілісекундах (ms).

Макроси в SEO

Макроси також використовуються спеціалістами SEO для автоматичного виконання повторюваних завдань. Наприклад, вони використовуються для форматування вмісту, який регулярно завантажується на веб-сайт, такі як дати подій, щотижневі нагадування, тощо. Вони також використовуються для

автоматичного оновлення електронних таблиць Excel або регулярних розсилок електронних листів на партнерські веб-сайти для розміщення посилань і обміну.

Макроси в програмуванні

У комп’ютерному програмуванні макроси — це фрагменти коду, які мають імена. Щоразу, коли використовується таке ім’я, виконується вміст макросу. Вони можуть нагадувати об'єкти або, навіть, виклики *функцій*.

Недоліки макросів

Макроси імітують записану послідовність дій, використовуючи одні й ті самі координати – все це чудово працюватиме, якщо не відбуватиметься жодних змін у фактичному знаходженні об'єктів на екрані. Однак, якщо якийсь із об'єктів хоч трохи змінить своє місце на екрані, то макрос зіб'ється зі шляху і продовжить "сліпо" виконувати записану послідовність дій.

Отже, нашим завданням є надання "зору" макросам, щоб виконання дій не було бездумним, а відбувався пошук необхідних об'єктів на екрані, незалежно від їхнього місцезнаходження. Для цього потрібно розробити програмний модуль для автоматизованого створення макросів та **ботів**.

1.2. Опис ботів. Їх види, можливості та переваги

Визначення та особливості

Бот (скорочення від robot) — це програма, запрограмована на виконання певних завдань у рамках іншої комп’ютерної програми або для імітації людської діяльності. Боти створені для самостійної автоматизації завдань без втручання людини, таким чином усуваючи громіздкі ручні процеси. Ці завдання часто повторюються і їх боти можуть виконати набагато швидше, надійніше та точніше, ніж людина.

Деякі боти корисні, наприклад, боти пошукової системи, які індексують вміст для пошуку, також боти служби підтримки клієнтів, які допомагають користувачам. Інші боти є «поганими» та запрограмовані на зламування акаунта користувачів, пошук контактної інформації в інтернеті для надсилання спаму або виконання інших шкідливих дій.

Зазвичай боти працюють у мережі. Вони спілкуються один з одним за допомогою інтернет-сервісів, таких як обмін миттєвими повідомленнями (IM), такі інтерфейси як Twitterbots або Internet Relay Chat. Відповідно до звіту дослідження 2021 року під назвою «Bot Attacks: Top Threats and Trends» охоронної компанії Barracuda, понад дві третини інтернет-трафіку складають боти. Крім того, 67% поганого трафіку ботів походить із публічних центрів обробки даних у Північній Америці.

Боти складаються з наборів алгоритмів, які допомагають їм виконувати поставлені завдання. Ці завдання включають розмову з людиною, де бот намагається імітувати людську поведінку, або збір інформації з різних веб-сайтів. Існує кілька різних типів ботів, призначених для виконання різноманітних завдань.

Наприклад, чат-бот використовує один із кількох методів роботи. Чат-бот, заснований на визначених правилах, взаємодіє з користувачем, надаючи йому попередньо визначені підказки для вибору. Інтелектуально незалежний чат-бот використовує машинне навчання, щоб навчатися на основі інформації, введеної людьми і шукати цінні ключові слова, які можуть ініціювати взаємодію. Чат-боти зі штучним інтелектом — це комбінація заснованих на правилах та інтелектуально незалежних чат-ботів. Чат-боти також можуть використовувати інструменти зіставлення шаблонів, обробки природної мови (NLP) і створення природної мови.

Організації та окремі особи, які використовують ботів, також можуть використовувати програмне забезпечення для управління ботами, яке допомагає керувати ботами та захищати від шкідливих ботів. Менеджери ботів також можуть входити до складу платформи безпеки веб-додатків. Менеджер ботів може дозволити використання одних ботів і заблокувати використання інших, які можуть завдати шкоди системі. Для цього менеджер ботів класифікує всі вхідні запити від людей і хороших ботів, а також відомих шкідливих і невідомих ботів. Тоді менеджер ботів перенаправляє будь-який підозрілий трафік ботів із сайту. Деякі базові набори функцій керування ботом включають обмеження швидкості IP та CAPTCHA. Обмеження швидкості IP обмежує кількість запитів на однакові адреси, тоді як CAPTCHA створює завдання, які допомагають відрізнити ботів від людей.

Види ботів

Існує багато різних видів ботів, усі з унікальними цілями та завданнями. Серед поширених ботів:

Чат-боти. Ці програми можуть імітувати розмову з людиною. Одною із перших та найвідоміших чат-ботів була Eliza, програма НЛП, розроблена в 1966 році як дослідницький проект Массачусетського технологічного інституту. Цей чат-бот прикидався психотерапевтом і відповідав на питання іншими запитаннями. Більш свіжі приклади чат-ботів включають віртуальних помічників, таких як Alexa від Amazon, Siri від Apple і Google Assistant.

Соціальні боти. Це боти, яких часто вважають ботами поглядів (переконання, думок), вони впливають на дискусії з користувачами на платформах соціальних мереж.

Шопботи. Багато з цих програм роблять покупки в інтернеті та знаходять найкращу ціну на продукт, який користувач зацікавлений придбати. Інші шоп-боти, такі як чат-бот Shopify, дозволяють власникам магазинів Shopify автоматизувати маркетинг і підтримку клієнтів.

Ноу-боти. Ці програми збирають знання для користувача шляхом автоматичного відвідування веб-сайтів для отримання інформації, яка відповідає певним заданим критеріям. Ноу-боти спочатку використовувалися як комп’ютеризований помічник, який виконував додаткові завдання.

Павуки або повзуни. Також відомі як веб-сканери, ці боти отримують доступ до веб-сайтів і збирають їх вміст для індексів у пошукових системах, таких як Google і Bing.

Веб-скребки. Вони схожі на сканери, але використовуються для збору даних і вилучення відповідного вмісту з веб-сторінок.

Моніторингові боти. Їх можна використовувати для моніторингу роботи веб-сайту чи системи.

Транзакційні боти. Ці боти розроблені, щоб спростити завдання, які інакше виконувала б людина по телефону, наприклад, блокувати викрадену кредитну картку або підтверджувати години роботи банку.

Ботів також можна класифікувати як хороших ботів і поганих ботів — іншими словами, боти, які не завдають шкоди, проти ботів, які становлять загрозу.

Приклади та використання ботів

Ботів можна використовувати в сферах обслуговування клієнтів, а також у таких сферах, як бізнес, планування, функції пошуку та розвагах. Боти в кожній області пропонують різні переваги. Наприклад, боти обслуговування клієнтів доступні 24/7 і підвищують доступність співробітників служби підтримки клієнтів. Ці програми також називаються віртуальними представниками або віртуальними помічниками, і вони звільняють людей, щоб ті могли зосередитися на більш складних питаннях.

Інші служби, які використовують ботів, включають:

- програми обміну миттєвими повідомленнями, такі як Facebook Messenger, WhatsApp, Viber і Telegram;
- новинні програми, такі як The Wall Street Journal, щоб показувати заголовки новин;
- Spotify, який дозволяє користувачам шукати та ділитися музичними треками через Facebook Messenger;

- Lyft, який дозволяє користувачам замовляти поїздки за допомогою програм для обміну миттєвими повідомленнями;
- послуги планування зустрічей;
- програми обслуговування клієнтів, які використовують чат-ботів для надсилання запитів клієнтів і опитування клієнтів.

Шкідливі боти

Шкідливі боти використовуються для автоматизації дій, які вважаються кіберзлочинами. Серед поширених типів шкідливих ботів:

- DoS-боти, які перевантажують ресурси сервера та перешкоджають роботі сервісів;
- спам-боти, які публікують рекламний вміст для залучення трафіку на певний веб-сайт;
- хакери, які розповсюджують зловмисне програмне забезпечення, атакують веб-сайти та збирають конфіденційну інформацію, таку як фінансові дані. Боти, створені хакерами, також можуть відкривати бекдори для встановлення більш серйозних шкідливих програм і вірусів.

Серед інших шкідливих типів ботів:

- засоби заповнення облікових даних;
- програмне забезпечення для збору адрес електронної пошти;
- засоби викрадення паролів та кейлоггери.

Переваги ботів

Використання ботів має багато переваг, а також недоліків, наприклад ризики, які можуть становити інші боти. Переваги ботів включають наступне:

- швидше за людей у повторюваних завданнях;
- економія часу замовників і клієнтів;
- доступний 24/7;
- організації можуть охоплювати велику кількість людей за допомогою додатків для обміну повідомленнями;
- можливість налаштування та навчання досвідом користувача

2. ТЕХНОЛОГІЧНА ЧАСТИНА

2.1. Побудова алгоритму створення макросів

Перш за все, побудуємо алгоритм програми для створення макросів:

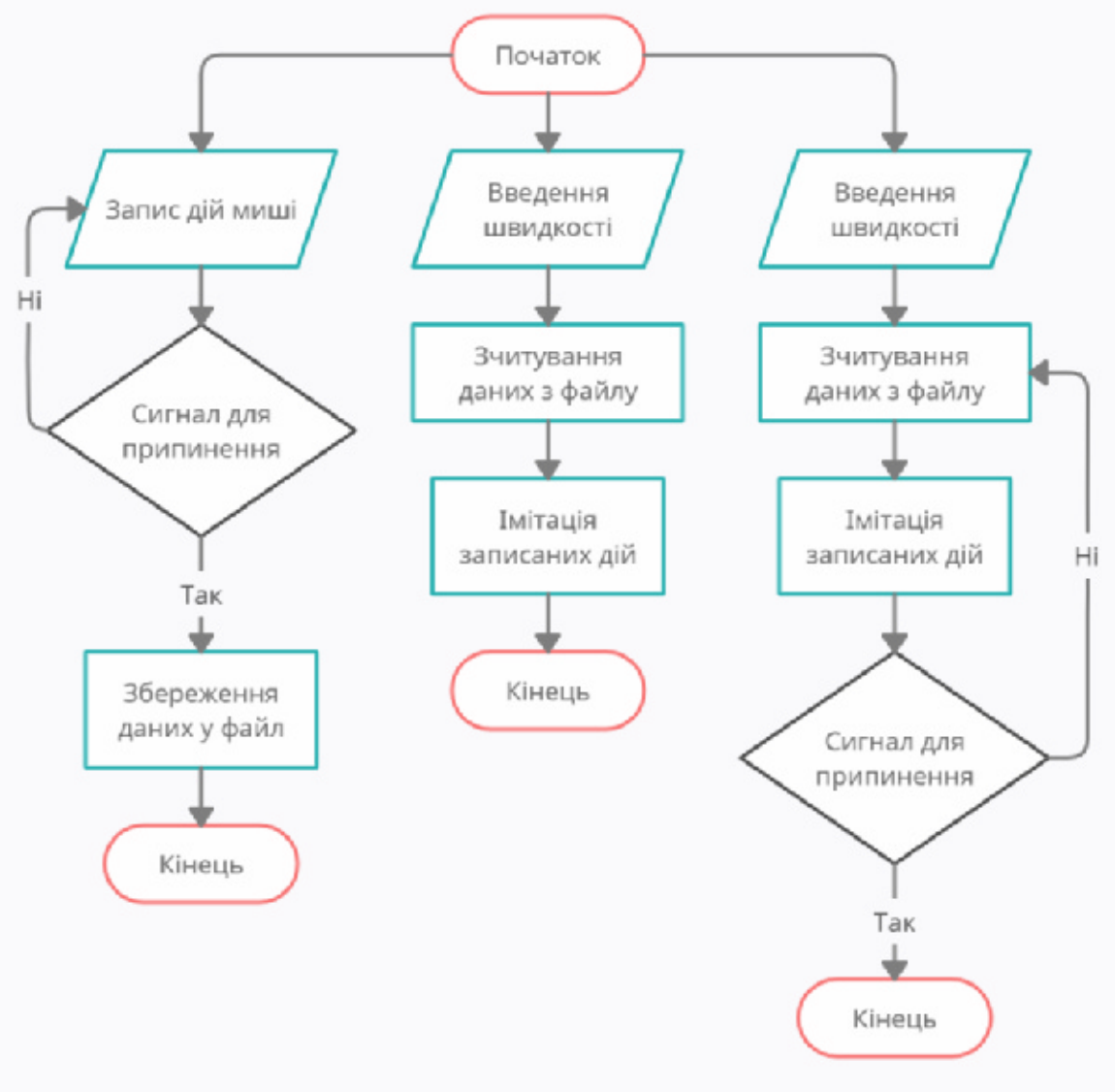


Рисунок 2.1 – Алгоритм автоматизованого створення макросів

Принцип роботи алгоритму макросів

Блок-схема складається з трьох гілок, тобто в даному випадку з трьох функцій програми.

У першій функції, при її викликанні здійснюється запис макросу. Спочатку програма записує послідовні дії комп’ютерної миші та її дані, такі як координати переміщення, координати натискання, натиснуті кнопки та відносні часові інтервали між цими подіями. Далі йде перевірка, чи був сигнал від користувача про зупинення запису, якщо ні – то програма продовжує записувати макрос, а якщо була команда припинити – програма створює файл, у який записує усі запам’ятанні події миші і завершає роботу.

У другій функції, при її запуску користувач може змінити швидкість виконання записаних інструкцій макросу, тобто часові інтервали між подіями. Далі програма зчитує дані зі створеного раніше файлу, в який були записані події комп’ютерної миші. Одразу ж після цього йде імітація дій миші, згідно записаних даних. Після виконання усіх подій функція завершає свою роботу

Викликаючи третю функцію, користувач може так само задати швидкість виконання макросу. Після цього програма зчитує дані з того ж самого створеного раніше файлу, в який були записані події комп’ютерної миші під час виконання першої функції. Після цього відбувається імітація дій миші, згідно записаних даних та заданою користувачем швидкістю, але після виконання усіх подій функція очікує сигнал відміни, якщо такого не було, то програма продовжує імітацію подій з початку, таким чином створюючи цикл до тих пір, поки не буде сигналу припинення

2.2. Побудова алгоритму для створення ботів

Тепер побудуємо алгоритм програми для автоматизованого створення ботів:

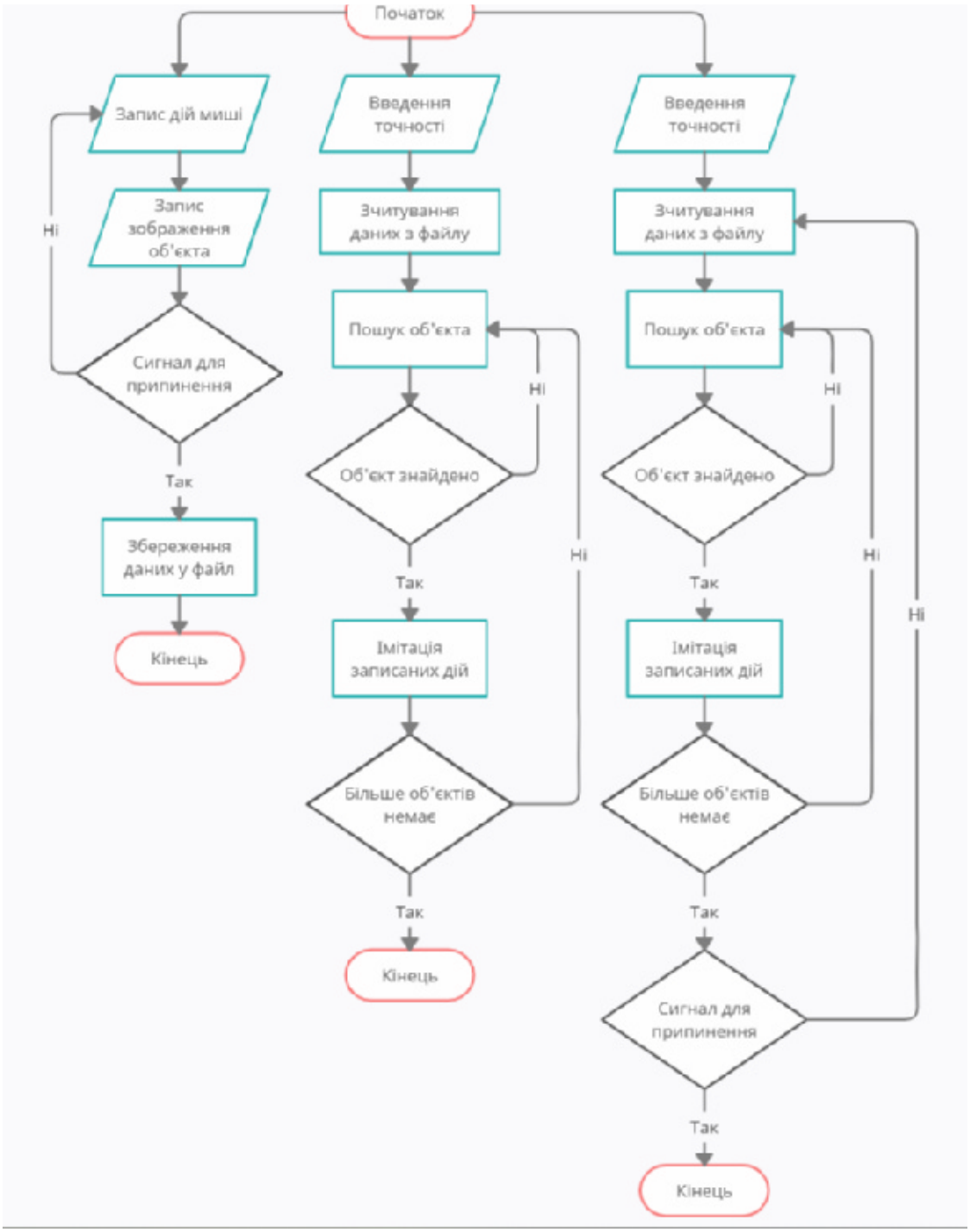


Рисунок 2.2 – Алгоритм програми для автоматизованого створення ботів

Принцип роботи алгоритму програми для створення ботів

Блок-схема також складається з трьох гілок - трьох функцій програми.

У першій функції, під час її викликанні здійснюється запис макросу. Перш за все, програма записує послідовні дії комп’ютерної миші та її дані, але цього разу лише натиснуті кнопки. Потім функція робить скріншот невеличкої області навколо курсору, тим самим записує об’єкт, який потім інші функції будуть шукати. Після цього відбувається перевірка чи необхідно далі записувати, якщо так – то цикл повертається на початок запису макросу, якщо була команда припинити запис – програма створює файл, у який відразу записує усі запам’ятанні кнопки миші та зображення об’єктів відповідно. Після чого завершає роботу.

Під час виклику другої функції, користувач може задати точність пошуку, наскільки записане зображення повинно співпадати з реальним на екрані. Програма зчитує дані з файлу, який ми створили та заповнили раніше і після цього починає пошук першого об’єкта зі списку на екрані користувача, якщо об’єкт знайдено, то функція виконує натискання кнопки миші, яка була записана разом з першим об’єктом, якщо нічого не знайдено – програма продовжує пошук. Далі, якщо у файлі були записані ще об’єкти, то функція виконує код до всіх по черзі, якщо всі об’єкти були знайдено та виконані натискання миші - функція завершає роботу.

При виконанні третьої функції відбуваються ті ж самі процеси, що й у другій, а саме користувач може задати точність пошуку, далі відбувається відкривання файлу із записаними об’єктами та кнопками, програма зчитує дані і закриває файл. Після чого розпочинає пошук першого об’єкта на екрані згідно списку, якщо знайдено – функція натискає відповідну кнопку, а якщо нічого не знайдено, то пошук продовжується. Далі ті ж самі дії відбуваються з наступним об’єктом, який було записано у файлі. Коли усі об’єкті знайдено, програма перевіряє наявність сигналу припинення. Якщо такого не було, то цикл

повертається на початок зчитування файлу і усі дії повторюються циклічно, поки не буде відповідного сигналу про відміну.

3. КОНСТРУКТОРСЬКА ЧАСТИНА

3.1. Вибір засобу для реалізації програмного модуля

Для реалізації програмного модуля для автоматизованого створення макросів та ботів було обрано мову програмування Python

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв’язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання як мови сценаріїв або з’єднувальної мови для з’єднання існуючих компонентів. Простий, легкий для вивчення синтаксис Python підкреслює читабельність і, таким чином, знижує вартість обслуговування програми. Python підтримує модулі та пакети, що сприяє модульності програми та повторне використання коду. Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Часто програмісти обирають Python через його підвищену продуктивність, яку він забезпечує. Оскільки етапу компіляції немає, цикл редагування-тестування-налагодження відбувається неймовірно швидко. Налагоджувати програми на Python легко: помилка чи неправильний вхід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, він викликає виняток. Якщо програма не вловлює виняток, інтерпретатор друкує трасування стека. Налагоджувач рівня вихідного коду дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати контрольні точки, покроково виконувати код по рядках і так далі. Сам налагоджувач написаний на Python, що свідчить про інтроспективну силу Python. З іншого боку, часто найшвидшим способом налагодження програми є додавання кількох операторів друку до джерела: швидкий цикл

редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

3.2. Вибір необхідних бібліотек для написання програмного модуля

Також нам знадобляться різні бібліотеки та модулі, для написання програмного модуля. А саме:

PyAutoGUI

PyAutoGUI дозволяє вашим сценаріям Python керувати мишею та клавіатурою для автоматизації взаємодії з іншими програмами. API розроблений таким чином, щоб бути простим. PyAutoGUI працює в Windows, macOS і Linux. PyAutoGUI має кілька функцій:

- Переміщення миші та клацання у вікнах інших програм.
- Надсилання натискань клавіш додаткам (наприклад, для заповнення форм).
- Робити скріншоти, надавати зображення (наприклад, кнопку або прапорець) і знаходити його на екрані.
- Знаходити вікно програми та переміщувати його, змінювати розмір, розгортати, згортати або закривати його (наразі лише для Windows).
- Відображати вікна сповіщень і повідомлень.

OpenCV

OpenCV (Open Source Computer Vision Library) – це відкрита бібліотека для роботи з алгоритмами комп'ютерного зору, машинним навчанням та обробкою зображень. Написана на C++, але існує також Python, JavaScript, Ruby

та інших мовах програмування. Працює на Windows, Linux та MacOS, iOS та Android.

OpenCV може використовуватися скрізь, де потрібний комп'ютерний зір. Ця галузь IT працює з технологіями, які дозволяють пристрою "побачити", розпізнати та описати зображення. Комп'ютерний зір дає точну інформацію про те, що зображено на зображенні, з описом, характеристиками та розмірами (з певним ступенем достовірності).

Також бібліотека працює з машинним навчанням - галуззю, яка навчає алгоритми діяти тим чи іншим чином.

Pillow

Бібліотека зображень Python додає можливості обробки зображень до вашого інтерпретатора Python.

Ця бібліотека забезпечує розширену підтримку форматів файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень.

Основна бібліотека зображень розроблена для швидкого доступу до даних, що зберігаються в кількох основних форматах пікселів. Вона забезпечує міцну основу для загального інструменту обробки зображень.

Mouse

Отримайте повний контроль над своєю мишею за допомогою цієї невеликої бібліотеки Python. Перехоплюйте глобальні події, реєструйте гарячі клавіші, імітуйте рух миші і клацання та багато іншого.

Особливості:

- Глобальний перехоплювач подій на всіх пристроях миші (фіксує події незалежно від фокусу).
- Прослуховування та посилання подій миші.
- Працює з Windows і Linux
- Працює з MacOS (потрібно надати дозвіл на доступ до терміналу/python у системних налаштуваннях -> Безпека та конфіденційність)
- Чистий Python, без потреби компіляції модулів C
- Включає API високого рівня (наприклад, запис і відтворення)
- Події автоматично фіксуються в окремому потоці, не блокує основну програму.
- Перевірено та задокументовано.
- Ця програма не намагається приховати себе, тому вона не підходить для кейлоггерів

Keyboard

Keyboard – бібліотека, яка повністю контролює вашу клавіатуру. За допомогою цієї бібліотеки ви можете друкувати будь-що, створювати гарячі клавіші, скорочення, блокувати клавіатуру, чекати введення з клавіатури та багато іншого.

Особливості:

- Глобальний перехоплювач подій на всіх клавіатурах (захоплює клавіші незалежно від фокусу).
- Прослуховування та посилання події клавіатури.
- Працює з Windows і Linux
- Чистий Python, без потреби компіляції модулів C

- Підтримка комплексних гарячих клавіш (наприклад, ctrl+shift+m, ctrl+пробіл) із контрольованим тайм-аутом.
- Включає API високого рівня (наприклад, запис і відтворення)
- Карти клавіш, як вони є у вашій розкладці, з повною підтримкою інтернаціоналізації (наприклад, Ctrl+ç).
- Події автоматично фіксуються в окремому потоці, не блокує основну програму.
- Перевірено та задокументовано.
- Не ламає акцентовані мертві клавіші

Pynput

Ця бібліотека дозволяє моніторити та контролювати пристрої введення.

Вона містить підпакети для кожного типу підтримуваного пристрою введення:

- Pynput.mouse - містить класи для керування та моніторингу миші або тачпада.
- Pynput.keyboard - містить класи для керування та моніторингу клавіатури.

Згадані вище модулі автоматично імпортуються в пакет Pynput. Щоб використовувати будь-який із них, імпортуйте їх із основного пакета.

Pickle

Модуль Pickle реалізує потужний алгоритм серіалізації та десеріалізації об'єктів Python. "Pickling" - процес перетворення об'єкта Python на потік байтів, а "unpickling" - зворотна операція, в результаті якої потік байтів перетворюється назад на Python-об'єкт. Так як потік байтів легко можна записати у файл, модуль Pickle широко застосовується для збереження та завантаження складних об'єктів у Python.

3.3. Розробка програмного модуля

Переходимо до написання коду. Перш за все, необхідно скачати і підключити необхідні нам бібліотеки та модулі:

```
import mouse
import pyautogui as pg
import pickle
import keyboard
from pynput import mouse as ms
```

Рисунок 3.1 – Підключення бібліотек

При підключенні модуля PyAutoGui оголошено його назву як «pg», для спрощення назви і таким чином спрощено процес написання. З бібліотеки Pynput імпортовано тільки модуль Mouse і оголошено його як «ms», щоб не було пересікання з бібліотекою Mouse.

Далі оголошуємо потрібні змінні:

```
step = 0
img_list = []
button_list = []
speedfact = 1
image_confidence = 0.8
```

Рисунок 3.2 – Оголошення змінних

Змінні speedfact та image_confidence – це початкові швидкість і точність відповідно, img_list та button_list – це змінні-списки з впорядкованою послідовністю елементів, які будуть заповнюватися зробленими зображеннями та натиснутими кнопками миші. А змінна step буде використовуватись для покрокового виконання коду.

Створюємо першу функцію, яка буде записувати всі дії миші:

```
def rec_macros():
    events = mouse.record(button='middle')
    try:
        with open('SavedMacros.bin', 'wb') as file1:
            pickle.dump(events, file1)
    except:
        print('Помилка')
```

Рисунок 3.3 – Функція запису подій миші

За допомогою бібліотеки Mouse та її методу .record у змінну events будуть записуватись усі події миші, координати переміщення, координати, по яким мишкою натиснули та якою кнопкою, а також відносні часові інтервали між цими подіями. Стоп-кнопкою назначимо середню кнопку, тобто коліщатко миші. Після того, як була нажата кнопка припинення, функція створює файл SavedMacros.bin та оголошує його змінною file1, після цього за допомогою модуля Pickle усі дані змінної events записуються у наш файл. Після чого одразу ж закриває створений файл, якщо виникне помилка, то функція повідомить користувача написом «Помилка». Файл буде закрито, навіть якщо виникне помилка, що дозволить уникнути витоку пам'яті - процес, при якому відбувається постійне зменшення доступної програмі оперативної пам'яті.

Тепер напишемо другу функцію, яка буде здійснювати імітацію дій миші, використовуючи раніше створений файл:

```
def play_macros(speedfact):
    try:
        with open('SavedMacros.bin', 'rb') as file1:
            mouse.play(pickle.load(file1), speed_factor=speedfact)
            mouse.click('middle')
    except:
        print('Помилка')
```

Рисунок 3.4 – Функція імітації дій миші

Функцію оголошуємо з параметром speedfact. Спочатку відкривається файл SavedMacros.bin, як змінна file1, потім, використовуючи метод .play, бібліотеки Mouse, а також Pickle для зчитування файлу, відбувається запуск імітації по заданим даним з вказаною швидкістю speedfact. Після чого файл закривається, а якщо виникла помилка, то виводиться на екран «Помилка».

Далі створюємо третю функцію, якщо буде ставити на цикл другу функцію:

```
def play_macros_cycle(speedfact):
    while not keyboard.is_pressed('Esc'):
        play_macros(speedfact)
```

Рисунок 3.5 – Функція циклічної імітації дій миші

Вказаний параметр швидкості – speedfact. З допомогою бібліотеки Keyboard та її методу .is_pressed ставимо функцію play_macros з параметром speedfact на циклічне виконання, поки не буде нажата клавіша клавіатури «Esc»

Таким чином, було створено функції для автоматизованого створення звичайних макросів з можливістю задати швидкість. Далі перейдемо до написання функцій для створення ботів.

Четвертою функцією буде запис дій користувача та скріншотів об’єктів, для подальшого їх розпізнавання:

```
def rec_bot():
    def on_clicked(x, y, button, pressed):
        pressed_status = 'Pressed' if pressed else 'Released'
        global image
        global step
        global img_list
        global button_list

        if button == button.left and pressed_status == 'Pressed' or button == button.right and pressed_status == 'Pressed':
            image = pg.screenshot('screenshots\my' + str(step) + '.png', region=(x - 15, y - 15, 30, 30))
            img_list.append(image)
            step = step + 1

        if pressed_status == 'Pressed':
            button_list.append(button)

        if button == button.middle:
            step = 0
            try:
                with open('SavedBot.bin', 'wb') as file2:
                    pickle.dump(img_list, file2)
                    img_list.clear()

                with open('SavedButtons.bin', 'wb') as file3:
                    pickle.dump(button_list, file3)
                    button_list.clear()
                return False
            except:
                print('Помилка')
                return False

    with ms.Listener(on_click=on_clicked) as listen:
        listen.join()
```

Рисунок 3.6 – Функція запису об’єктів екрану

У функції rec_bot створюємо функцію on_clicked з параметрами: x, y, button та pressed. В цій функції задаємо змінну pressed_status, яка буде визначати натискання кнопки миші, а також оголошуємо ряд глобальних змінних, які вже було оголошено поза функції, щоб дані цих змінних функція могла змінювати та використовувати для всієї програми, а не тільки локально для цієї функції.

Далі вказуємо умову за допомогою if, де якщо було натиснута ліва або права кнопка миші, то програма робить скріншот об’єкта навколо курсору миші

з областю 30x30 пікселів, після чого записує цей скріншот у список img_list. Теж саме відбувається з кнопкою миші, яка була натиснута, але записується у другу змінну button_list.

Якщо була натиснута середня кнопка, тобто коліщатко миші, то відбувається процес запису усіх даних з вище вказаних нами списків у різні файли, які функція створює, де зображення записуються у файл SavedBot.bin, а натиснуті кнопки миші у файл SavedButtons.bin, після чого змінні-списки очищаються від записаних даних. Також у функції rec_bot за допомогою методу .Listener бібліотеки Pynput запускається функція on_clicked для запису даних в окремий циклічний потік, не блокуючи основну програму.

Наступна, п’ята функція буде запускати імітацію дій користувача, шукаючи об’єкти на екрані, які були раніше записані:

```
def play_bot(image_confidence):
    def locate_on_screen_left(fil, lon):
        while True:
            butt = pg.locateOnScreen(fil[lon], confidence=image_confidence)
            if butt:
                pg.click(butt, duration=0.1)
                break

    def locate_on_screen_right(fil, lon):
        while True:
            butt = pg.locateOnScreen(fil[lon], confidence=image_confidence)
            if butt:
                pg.rightClick(butt, duration=0.2)
                break

    try:
        with open('SavedBot.bin', 'rb') as file2:
            img_list = pickle.load(file2)

        with open('SavedButtons.bin', 'rb') as file3:
            button_list = pickle.load(file3)
    except:
        print('Помилка')

    for i in range(len(img_list)):

        if str(button_list[i]) == str('Button.left'):
            locate_on_screen_left(img_list, i)

        elif str(button_list[i]) == str('Button.right'):
            locate_on_screen_right(img_list, i)
```

Рисунок 3.7 – Функція пошуку об’єктів екрану

У цій функції задається параметр image_confidence, тобто точність пошуку об’єкта, а також створюються дві інші функції – locate_on_screen_left та

locate_on_screen_right, де перша відповідає за натискання лівої кнопки миші, а друга – правої. Ці функції оголошуються з параметрами fil та lon, які будуть передавати змінну-список та її елемент відповідно. Спочатку функція play_bot відкриває створені раніше файли SavedBot.bin та SavedButtons.bin і переписує дані у змінні-списки img_list та button_list відповідно. Після цього, за допомогою циклу for відбувається ітерація кожного елемента списку зображень об’єктів, де якщо користувачем була нажата ліва кнопка миші на цей об’єкт, то виконується функція locate_on_screen_left, де відбувається пошук об’єкта з вказаною точністю на екрані і натискання на цей об’єкт лівою кнопкою, а якщо користувач натиснув правою кнопкою миші, то виконується функція locate_on_screen_right, в якій програма шукає об’єкт з вказаною точністю і натискає на нього правою кнопкою миші.

І наостанок, шоста функція, яка буде циклічно запускати функцію play_bot:

```
def play_bot_cycle(image_confidence):
    while not keyboard.is_pressed('Esc'):
        play_bot(image_confidence)
```

Рисунок 3.8 – Функція циклічного пошуку об’єктів екрану

Функція задається з параметром точності – image_confidence. Циклічне виконання функції запуска бота буде відбуватися до тих пір, поки користувач не натисне клавішу «Esc» з клавіатури.

Отже, було створено програмний модуль для автоматизованого створення макросів та ботів, з можливістю задавати швидкість для макросів та точність пошуку об’єктів для ботів

4. СПЕЦІАЛЬНА ЧАСТИНА

4.1. Вибір необхідних бібліотек для побудови інтерфейсу

Коли мова йде про програмне забезпечення, інтерфейс – це програма, яка дозволяє користувачеві взаємодіяти з комп’ютером особисто або через мережу. Інтерфейс може також стосуватися елементів керування, що використовуються в програмі, які дозволяють користувачеві взаємодіяти з програмою.

Для того, щоб спростити керування створеним програмним модулем, потрібно побудувати для нього інтерфейс, а для цього знадобляться наступні бібліотеки:

Tkinter

Tkinter – графічна бібліотека Python, яка призначена для створення програм з віконним інтерфейсом. Вона багатоплатформна, тобто з її допомогою можна писати програми для Windows, Linux, macOS.

ImageTk

Модуль ImageTk підтримує створення та зміну об’єктів Tkinter BitmapImage і PhotoImage із зображень бібліотеки PIL.

4.2. Розробка інтерфейсу для програмного модуля

Підключаємо необхідні для інтерфейсу бібліотеки:

```
from tkinter import *
from PIL import ImageTk
```

Рисунок 4.1 – Підключення модулів

Імпортуємо Tkinter та модуль ImageTk з бібліотеки PIL.

Далі створюємо вікно інтерфейсу та задаємо параметри:

```
root = Tk()
root.title('Auto Bot Maker')
root.resizable(width=False, height=False)
canvas = Canvas(root, width=610, height=630)
canvas.pack()

spdf = StringVar()
img_conf = StringVar()
```

Рисунок 4.2 – Створення вікна інтерфейсу

Оголошуємо клас Tk() як змінну root. Тоді задаємо назву нашому вікну «Auto Bot Maker», після цього забороняємо змінювати розмір вікна. Змінній canvas передаємо об’єкт класу TCanvas для виведення зображень, які можна змінювати і переміщати у процесі виконання програми, вказавши розмір полотна 610 пікселів в ширину та 630 пікселів у висоту і розміщуємо це полотно у нашому вікні root. Після чого створюємо два об’єкта класу StringVar(), за допомогою яких будемо передавати швидкість та точність.

Тепер створюємо зображення для кнопок:

```
imgbutton1 = ImageTk.PhotoImage(file='images/button_zapys_green.png')
imgbutton2 = ImageTk.PhotoImage(file='images/button_zapusk_green.png')
imgbutton3 = ImageTk.PhotoImage(file='images/button_povtor_green.png')
imgbutton4 = ImageTk.PhotoImage(file='images/button_zapys_blue.png')
imgbutton5 = ImageTk.PhotoImage(file='images/button_zapusk_blue.png')
imgbutton6 = ImageTk.PhotoImage(file='images/button_povtor_blue.png')
imgbutton7 = ImageTk.PhotoImage(file='images/button_zminity_green.png')
imgbutton8 = ImageTk.PhotoImage(file='images/button_zminity_blue.png')
```

Рисунок 4.3 – Оголошення змінних із зображеннями кнопок

Оголошуємо змінні та привласнюємо їм підготовлені для них зображення кнопок

Після цього створюємо зображення для заднього фону:

```
bg1 = ImageTk.PhotoImage(file='images/background1.jpg')
bg2 = ImageTk.PhotoImage(file='images/buttons_bg_green.png')
bg3 = ImageTk.PhotoImage(file='images/buttons_bg_blue.png')
bg4 = ImageTk.PhotoImage(file='images/buttons_bg_change_speed.png')
bg5 = ImageTk.PhotoImage(file='images/buttons_bg_change_confidence.png')
```

Рисунок 4.4 – Оголошення змінних із зображеннями фону

Оголошуємо змінні та привласнюємо їм підготовлені для них зображення заднього фону.

Далі створюємо самі кнопки інтерфейсу та елементи керування для введення користувачем тексту:

```
b1 = Button(command=rec_macros, image=imgbutton1, highlightthickness=0, bd=0)
b2 = Button(command=lambda: play_macros(speedfact), image=imgbutton2, highlightthickness=0, bd=0)
b3 = Button(command=lambda: play_macros_cycle(speedfact), image=imgbutton3, highlightthickness=0, bd=0)
b4 = Button(command=rec_bot, image=imgbutton4, highlightthickness=0, bd=0)
b5 = Button(command=lambda: play_bot(image_confidence), image=imgbutton5, highlightthickness=0, bd=0)
b6 = Button(command=lambda: play_bot_cycle(image_confidence), image=imgbutton6, highlightthickness=0, bd=0)
b7 = Button(command=change_button_speed, image=imgbutton7, highlightthickness=0, bd=0)
b8 = Button(command=change_button_confidence, image=imgbutton8, highlightthickness=0, bd=0)

e1 = Entry(textvariable=spdf, width=9, font=('Segoe', 29), justify='center')
e2 = Entry(textvariable=img_conf, width=9, font=('Segoe', 29), justify='center')
```

Рисунок 4.5 – Створення кнопок та елементів керування

Кожній змінній привласнюємо відповідну кнопку інтерфейсу. Задаємо параметри кнопкам, щоб вони викликали або виконували з відповідними параметрами функції, також передаємо раніше створені зображення кнопкам і робимо пустим власний задній фон кнопок. Також оголошуємо два елемента інтерфейсу для введення тесту.

Після цього змінюємо конфігурацію елементів:

```
b1.configure(bg='black', activebackground='black')
b2.configure(bg='black', activebackground='black')
b3.configure(bg='black', activebackground='black')
b4.configure(bg='black', activebackground='black')
b5.configure(bg='black', activebackground='black')
b6.configure(bg='black', activebackground='black')
b7.configure(bg='black', activebackground='black')
b8.configure(bg='black', activebackground='black')

e1.configure(bg='green')
e2.configure(bg='blue')
```

Рисунок 4.6 – Зміна конфігурації елементів

Для кнопок встановлюємо задній фон чорним кольором, а для елементів Entry задаємо зелений колір для швидкості та синій для точності.

Потім розміщуємо зображення фону на полотні нашого вікна:

```
canvas.create_image(0, 0, image=bg1)
canvas.create_image(135, 173, image=bg2)
canvas.create_image(480, 173, image=bg3)
canvas.create_image(135, 480, image=bg4)
canvas.create_image(480, 480, image=bg5)
```

Рисунок 4.7 – Розміщення фонових зображень

З допомогою методу .create_image створюємо раніше підготовлені зображення та розміщуємо їх на полотні з відповідними координатами

Під кінець розміщуємо елементи інтерфейсу на полотні та завершуємо інтерфейс:

```
b1.place(x=35, y=100)
b2.place(x=35, y=168)
b3.place(x=35, y=236)
b4.place(x=380, y=100)
b5.place(x=380, y=168)
b6.place(x=380, y=236)
b7.place(x=35, y=500)
b8.place(x=380, y=500)

e1.place(x=35, y=440)
e2.place(x=380, y=440)

root.mainloop()
```

Рисунок 4.8 – Розміщення кнопок

Розташовуємо кнопки та елементи введення тексту по координатам x та y. В кінці за допомогою функції mainloop() викликаємо нескінченний цикл опрацювання подій вікна, тому вікно буде чекати на будь-яку взаємодію з користувачем, поки не буде закрите.

У основній програми створюємо функції для зміни швидкості та точності:

```
def change_button_speed():
    global speedfact
    speedfact = spdf.get()
    speedfact = float(speedfact)


def change_button_confidence():
    global image_confidence
    image_confidence = img_conf.get()
    image_confidence = float(image_confidence)
```

Рисунок 4.9 – Функції зміни швидкості та точності

У функціях оголошено глобальні змінні speedfact та image_confidence, таким чином локальні зміни даних будуть змінюватись і для всієї програми. Цим змінним присвоюється значення створених елементів для введення тексту, куди користувач буде вводити необхідні значення, а далі функції перетворюють ці значення з типу string на тип float, тобто на числа з плаваючою точкою.

В результаті виконаних дій, ми отримуємо вікно інтерфейсу з таким виглядом:

Рисунок 4.10 – Інтерфейс керування

Тепер користувач може вільно керувати програмним модулем та автоматизовано створювати макроси та боти, задаючи власні параметри швидкості виконання для макросів та точності пошуку об’єкта для ботів.

5. НАУКОВО-ДОСЛІДНА ЧАСТИНА

5.1. Тестування параметра швидкості виконання макросів

Проведемо досліди параметра швидкості макроса та порівняємо їх з ботом.

Спершу викликаємо функцію створення макросів з допомогою зеленої кнопки «Запис» з меню «Макрос» та записуємо простий макрос

Рисунок 5.1 – Зміна швидкості на «1.0»

Пройдемо по наступному шляху: Мені Пуск – Панель керування - Брандмауер для Захисника Windows – Змінення параметрів сповіщення.

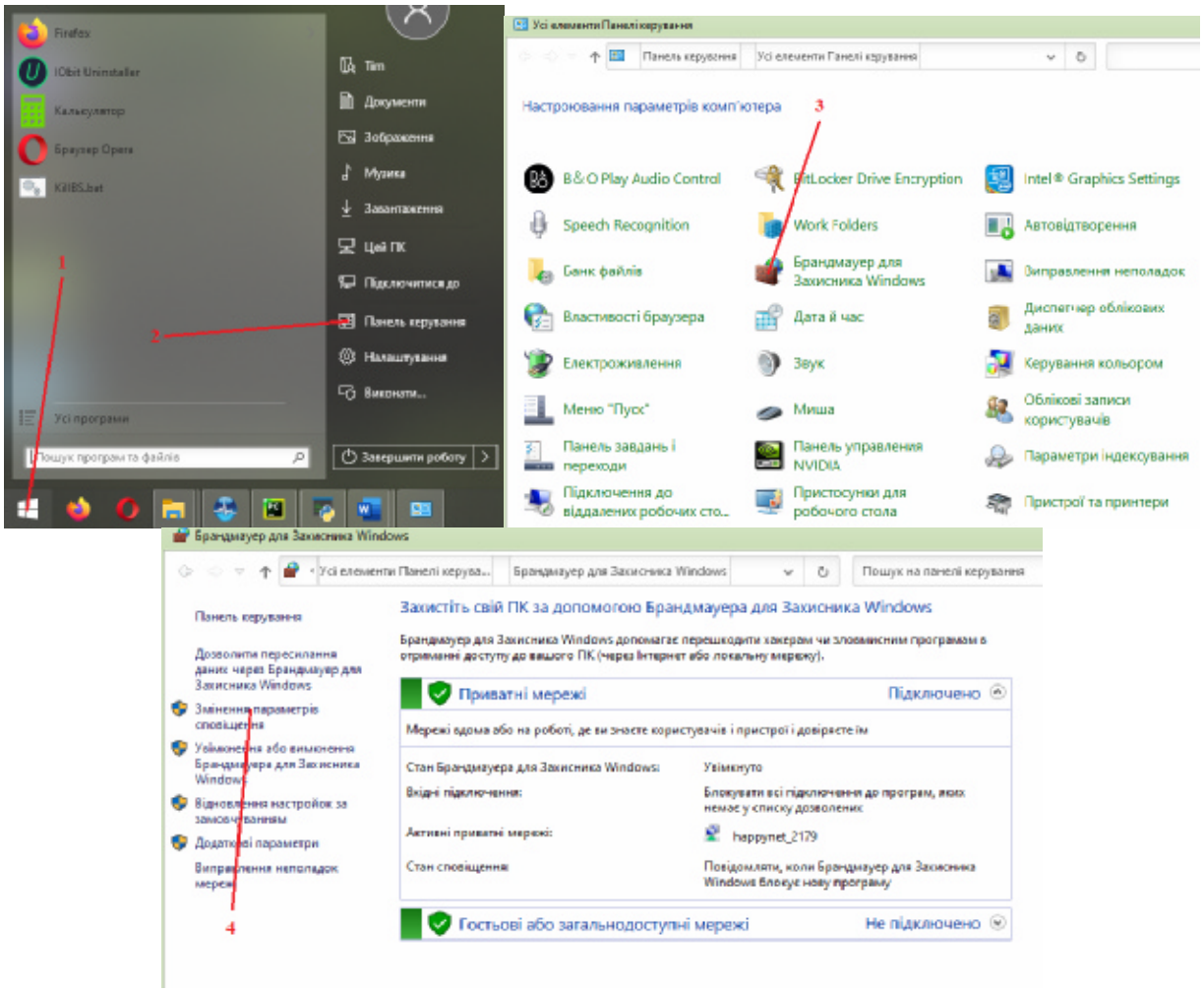


Рисунок 5.2-5.4 – Записаний шлях до налаштувань

В результаті, отримуємо вікно налаштування Брандмауера:

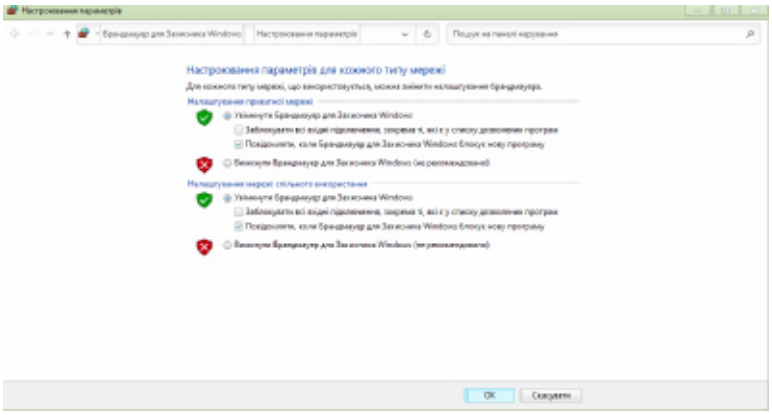


Рисунок 5.5 – Вікно налаштування Брандмауера

Тепер за допомогою зеленої кнопки «Запуск» з меню «Макрос» виконаємо функцію play_macros та заміряємо час, за який вдалось вручну дійти до налаштувань Брандмауера.

Отримуємо час: 9.07 секунди. Макрос повністю повторив усі дії, які були виконані вручну. Тепер спробуємо прискорити виконання макроса, змінивши швидкість виконання з 1.0 до 3.0 і запустимо макрос.

Рисунок 5.6 – Зміна швидкості на «3.0»

Отриманий час: 3.39 секунди. Якщо спробувати поставити швидкість 4.0, то макрос зіб’ється з курсу, оскільки комп’ютер не встигає відкривати вікна так швидко, тому макрос натискає раніше потрібного і йде далі по записаних подіях, не розбираючи елементів екрану.

А тепер застосуємо програму для створення бота, в якого є свій «зір». Для цього натискаємо синю кнопку «Запис» з меню «Бот» та пройдемо такий самий шлях, передаючи ботові необхідні об’єкти екрану.

Рисунок 5.7 – Меню бота

Автоматично записані ботом об’єкти екрану:

Рисунок 5.8

Тепер запускаємо виконання функції play_bot з допомогою синьої кнопки «Запуск» з меню «Бот» і заміряємо час, за який бот дійде до потрібного нам налаштування.

Отримуємо результат: 2.52 секунди. Бот виконав задані дії настільки швидко, наскільки це можливо, як тільки елементи з’являлись на екрані – бот одразу ж натискав на них.

Таким чином, створений автоматизований бот виявився ефективнішим, ніж макрос, коли необхідно виконати чітку послідовній дій. Однак макрос залишається ефективнішим для виконання абстрактних дій, наприклад, якщо намалювати якусь фігуру або малюнок і записати в макрос, то цю дію можна буде повторювати безліч раз, в прискореному режимі, а ось бот у даному програмному модулі такого не підтримує. Тому макроси та ботів слід використовувати для різних задач, щоб спрощувати роботу за ПК.

5.2. Тестування параметра точності пошуку об’єктів ботом

Перевіримо, як точність пошуку об’єктів впливає на роботу ботів. Для цього запишемо боту просту команду відкривання кошика на робочому столі:

Рисунок 5.9 – Кошик на робочому столі

Записані ботом об’єкти:

Рисунок 5.10 – Зображення кошика

Було здійснено подвійний клік мишкою, тому бот зберіг два однакових об’єкта і таким чином, буде виконувати так само два кліка лівою кнопкою миші на цей об’єкт.

Рисунок 5.11 – Зміна точності на «0.99»

Тепер виставимо точність 0.99 та запустимо бота.

Бот відразу находить кошик на робочому столі та відкриває його.

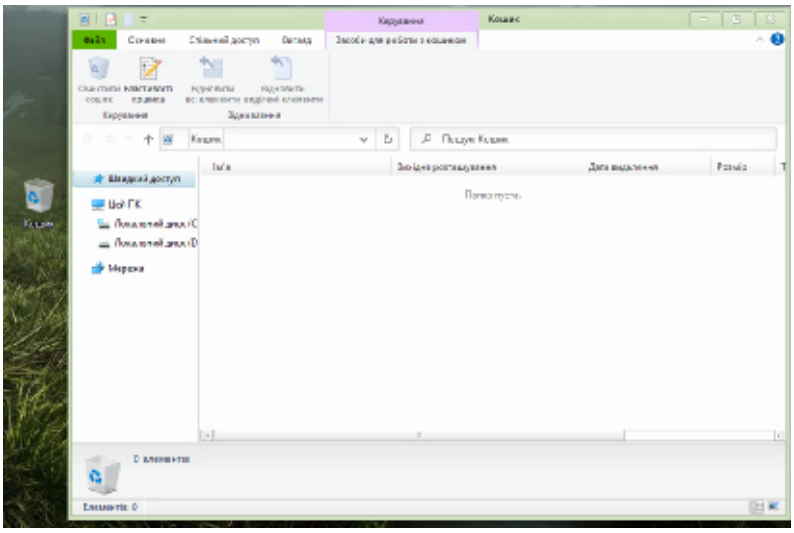


Рисунок 5.12 – Знайдений кошик

А тепер спробуємо перенести кошик на робочому столі в інакше місце і запустити бота:

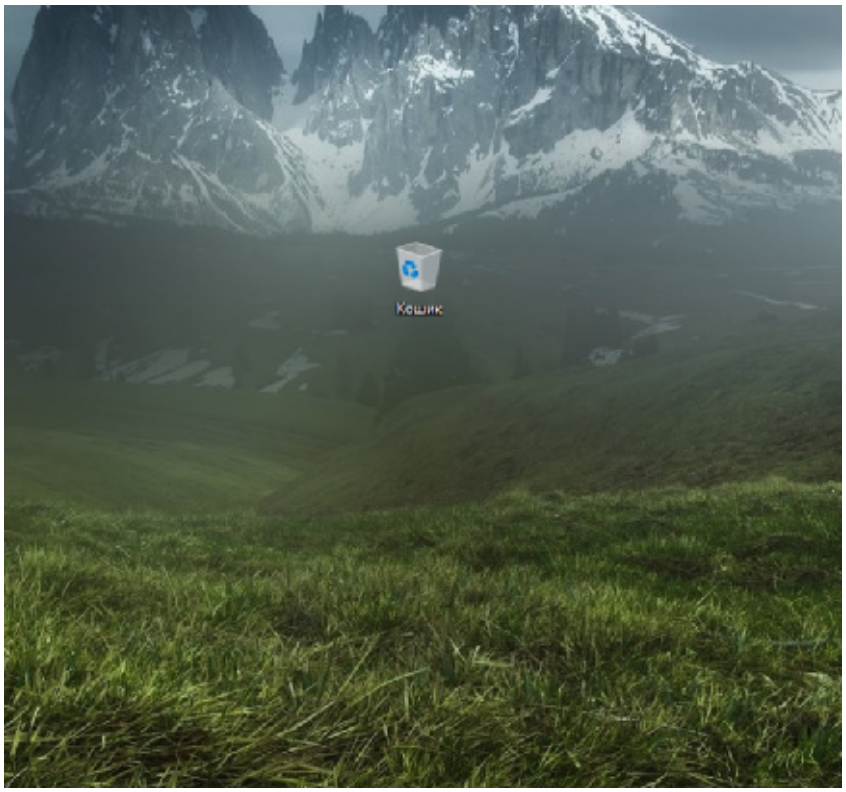


Рисунок 5.13 – Зміна місця кошика на робочому столі

В результаті, бот зависає під час пошуку об’єкта, оскільки вказана точність надто висока, а наш об’єкт – кошик вже змінив задній фон, тому він не відповідає на 100% об’єкту, який бот шукає:

Рисунок 5.14 – Бот в стані пошуку об’єкта

А тепер спробуємо змінити точність на 0.85 і запустимо бота:

Рисунок 5.15 – Зміна точності на «0.85»

Цього разу бот зразу найшов необхідний об’єкт, хоча спочатку він був в іншому місці робочого стола.

Однак, якщо точність вказати надто низьку, то бот буде находити схожі об’єкти на екрані, але не завжди той, що нам потрібен. Спробуємо поставити точність 0.35 і будемо спостерігати за діями бота:

Рисунок 5.16 – Зміна точності на «0.35»

Як і очікувалось, бот не знайшов кошик, а замість нього натиснув двома кліками по області на робочому столі, яка трохи схожа на кошик (вказано червоною стрілкою)

Рисунок 5.17 – Помилкове знаходження об’єкта

Отже, точність пошуку об’єкта слід ставити згідно місця виконання і задач, які необхідно виконати для максимальної ефективності автоматизованого бота. Якщо правильно встановити точність пошуку, то бот буде знаходити потрібні об’єкти незалежно від того, чи вони змінили своє місце на екрані, що є явною перевагою над макросами.

5.3. Дослідження швидкодії бота, під час пошуку об’єктів

Проведемо дослід, у якому проаналізуємо швидкість пошуку об’єктів користувачем та ботом з різними налаштуваннями, а для цього використаємо додаток «Aim Trainer» від «Human Benchmark».

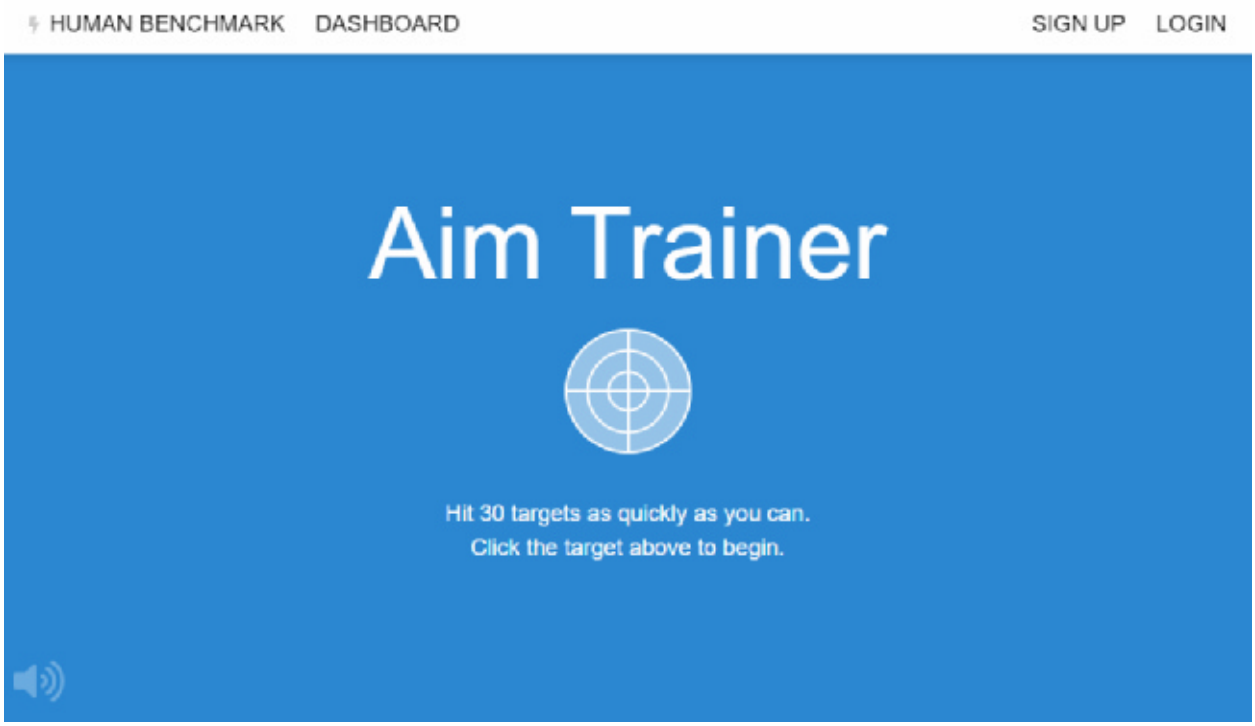


Рисунок 5.18 – Додаток «Aim Trainer»

В цьому додатку слід натискати на цілі якомога швидше й точніше, вони будуть з’являтися в різних випадкових місцях на екрані. Це перевіряє рефлекси та координацію рук і очей. Коли буде вражено 30 мішеней, відобразиться рахунок і середній час затрачений на мішень.

Спочатку вручну буде зроблено п’ять спроб, а потім з допомогою бота, після чого буде підраховано середній час всіх спроб кожного методу. Тоді буде можливість порівняти швидкості людини з ботом.

Під час першої спроби вручну, отриманий середній час на мішень становить: 585 мс

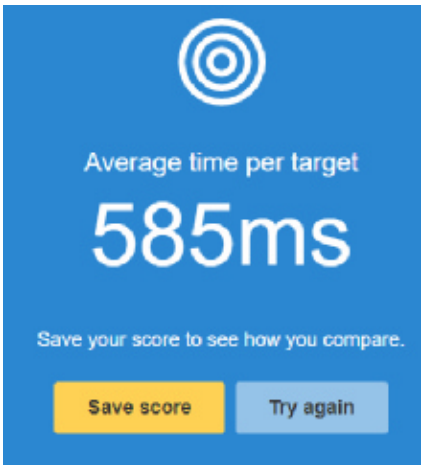


Рисунок 5.19 – Перша спроба тестування вручну

Далі наступні чотири спроби відповідно:

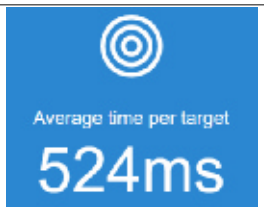
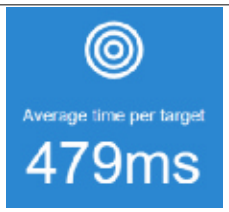
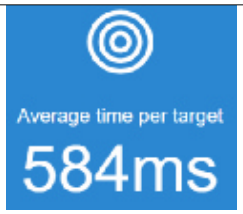
			

Рисунок 5.20 – Друга спроба вручну	Рисунок 5.21 – Третя спроба вручну	Рисунок 5.22 – Четверта спроба вручну	Рисунок 5.23 – П’ята спроба вручну

Отже, підрахуємо середній час, який було витрачено на одну мішень за п’ять спроб вручну: T_{avg} = (585 + 524 + 479 + 556 + 584) ÷ 5 = 545.6 мс

Далі створюємо бота за допомогою програмного модуля і запускаємо функцію play_bot_cycle синьою кнопкою «Повтор» з меню «Бот», щоб бот почав «ловити» мішені:

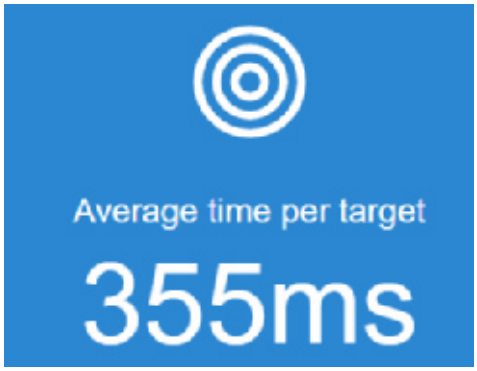


Рисунок 5.24 – Перша спроба тестування бота

За першу спробу бот отримав результат: 355 мс, що вже кращим результатом, ніж будь-який вручну. Продовжимо запускати бота до п’яти спроб:

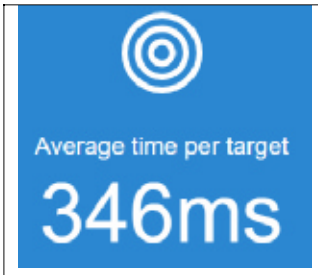


Рисунок 5.25 – Друга спроба бота

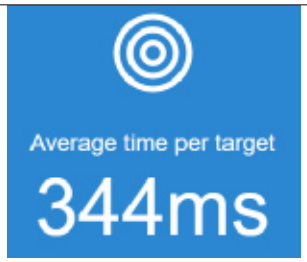


Рисунок 5.26 – Третя спроба бота

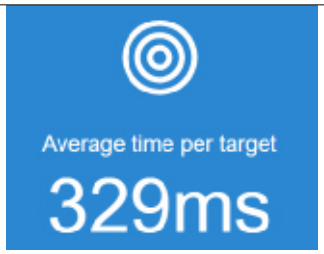


Рисунок 5.27 – Четверта спроба бота

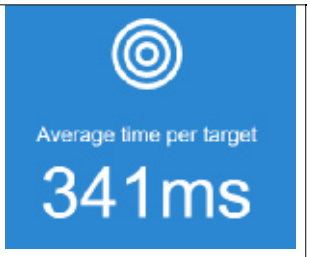


Рисунок 5.28 – П’ята спроба бота

Підрахуємо середній час, який бот витратив на пошук однієї мішені за п’ять спроб: T_{avg} = (355 + 346 + 344 + 329 + 341) ÷ 5 = 343 мс

Спробуємо внести деякі зміни в програмний код, а саме у функцію пошуку об'єктів додамо параметр grayscale=True, щоб бот здійснював зіставлення об'єктів за відтінками сірого кольору (чорно-білого). Це зменшує насиченість кольорів із зображень і знімків екрана, пришвидшуючи пошук:

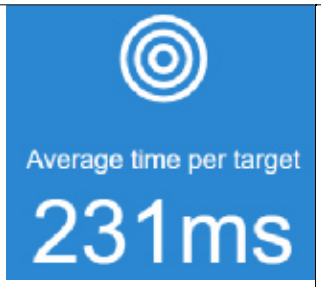
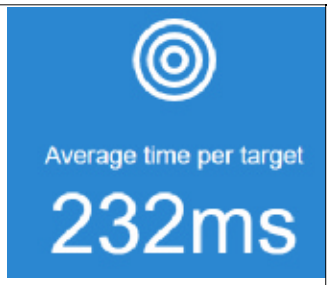
```
def play_bot(image_confidence):
    def locate_on_screen_left(fil, lon):
        while True:
            butt = pg.locateOnScreen(fil[lon], confidence=image_confidence, grayscale=True)
            if butt:
                pg.click(butt, duration=0.1)
                break
```

Рисунок 5.29 – Зміна функції пошуку об’єкта

Після чого протестуємо, як змінилась швидкодія бота, під час пошуку об’єктів. Для цього запускаємо знову функцію play_bot_cycle на п’ять спроб:

Рисунок 5.30 – Перша спроба тестування бота за сірими відтінками

Отриманий результат першої спроби – 240 мс. Бот почав шукати об’єкти на екрані набагато швидше. Продовжимо до здійснення п’яти спроб:

			
Рисунок 5.31 – Друга спроба бота за сірими відтінками	Рисунок 5.32 – Третя спроба бота за сірими відтінками	Рисунок 5.33 – Четверта спроба бота за сірими відтінками	Рисунок 5.34 – П’ята спроба бота за сірими відтінками

Підрахуємо середній час, який бот витратив на пошук однієї мішені за п’ять спроб за відтінками сірого кольору: T_{avg} = (240 + 239 + 229 + 231 + 232) ÷ 5 = 234.2 мс

Занесемо отримані дані у таблицю:

Таблиця 5.1

Спроба	Користувач (вручну), мс	Бот з врахуванням кольорів, мс	Бот без врахування кольорів (за відтінками сірого кольору), мс
1	585	355	240
2	524	346	239
3	479	344	229
4	556	329	231
5	584	341	232
Середня	545.6	343	234.2

Виходячи з отриманих результатів, швидкість знаходження об’єктів на екрані ботом значно вища, ніж людиною. Визначимо, наскільки відсотків різниця.

Бот з врахуванням кольорів: 545.6 ÷ 343 × 100 - 100 = 59.07%

Бот за відтінками сірого кольору: 545.6 ÷ 234.2 × 100 - 100 = 132.96%

Згідно розрахунків, бот з врахуванням кольорів на 59% відсотків швидше знаходив об’єкти на екрані, ніж користувач. А бот з пошуком за відтінками сірого кольору на 133% швидше.

Також проаналізуємо, наскільки швидше бот почав шукати об’єкти після зміни функції пошуку: 343 ÷ 234.2 × 100 - 100 = 46.46%

Таким чином, після задання параметру grayscale=True у функції пошуку об’єкта, бот почав находити об’єкти на екрані на 47% швидше.

6. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Питання охорони праці та безпеки в надзвичайних ситуаціях розглянуті для етапу проектування й розробки системи аналізу та візуалізації кліматичних даних.

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров’я і працездатності людини у процесі трудової діяльності. Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам законодавства. Працівник має право відмовитися від дорученої роботи, якщо створилася виробнича ситуація, небезпечна для його життя чи здоров’я або для людей, які його оточують, або для виробничого середовища чи довкілля. Він зобов’язаний негайно повідомити про це безпосереднього керівника або роботодавця. Факт наявності такої ситуації за необхідності підтверджується спеціалістами з охорони праці підприємства за участю представника профспілки, членом якої він є, або уповноваженої працівниками особи з питань охорони праці (якщо професійна спілка на підприємстві не створювалася), а також страхового експерта з охорони праці [12]. Завдання охорони праці – звести до мінімуму ушкодження та захворювання працівника з одночасним забезпеченням комфорту при максимальній продуктивності праці. Основними цілями охорони праці є формування в спеціалістів необхідних знань і практичних навичок по правових і організаційних питаннях охорони праці, виробничій санітарії, техніці безпеки, пожежній безпеці.

6.1. Загальна характеристика приміщення і робочого місця

Розробка системи виконується в приміщенні (рис. 6.1), яке знаходиться на четвертому поверсі восьмиповерхового будинку з загальним та місцевим освітленням. В приміщенні одностороннє освітлення, вікна орієнтовані на схід, на вікнах є ролети. Стеля білого кольору з коефіцієнтом відбиття 0,7, стіни цегляні світлого кольору з коефіцієнтом відбиття 0,5. В приміщенні працює 4 людини, відповідно до цього отримуємо вхідні дані для аналізу потенційно-небезпечних і шкідливих виробничих факторів, які наведено в табл. 6.1.

Таблиця 6.1

Вхідні дані

Параметри приміщення	**Значення**
Довжина х ширина х висота	6,6 х 6,1 х 2,7 м
Площа	40,26 м^2
Об’єм	108,70 м^3
Номер робочого місця	**Специфіка роботи**
I робоче місце	Front-end програміст (спеціаліст з розробки клієнтської частини веб- застосунків)
II робоче місце	Back-end програміст (спеціаліст з розробки серверної частини веб застосунків та проектування баз даних)
III робоче місце	Бізнес-аналітик (також виконує роль менеджера продукту)
IV робоче місце	UI-UX веб-дизайнер
Технічні засоби (кількість)	**Назва та характеристики**

Монітор (4 шт.)	HP 22Xi/21,5"/1920x1080px/IPS
Комп’ютер (4 шт.)	HP ProBook 440 G6, екран 14" IPS (1920x1080) Full HD, Intel Core i7-8565U (1.8 - 4.6 ГГц)/RAM 16 ГБ/SSD 256 ГБ
Підлоговий кулер (1 шт.)	CRYSTAL YLR3-5V208
Кондиціонер (1 шт.)	DEKKER DSH105R/G/26м^2/2,65кВт-2,9кВт/25x74,5x19,5см/9 кг
Світильники загального призначення (3 шт.)	Світильник растровий вмонтований 4x18W
Світильники місцевого призначення (4 шт.)	DeLux Décor TF-05 / 1 x 40Вт

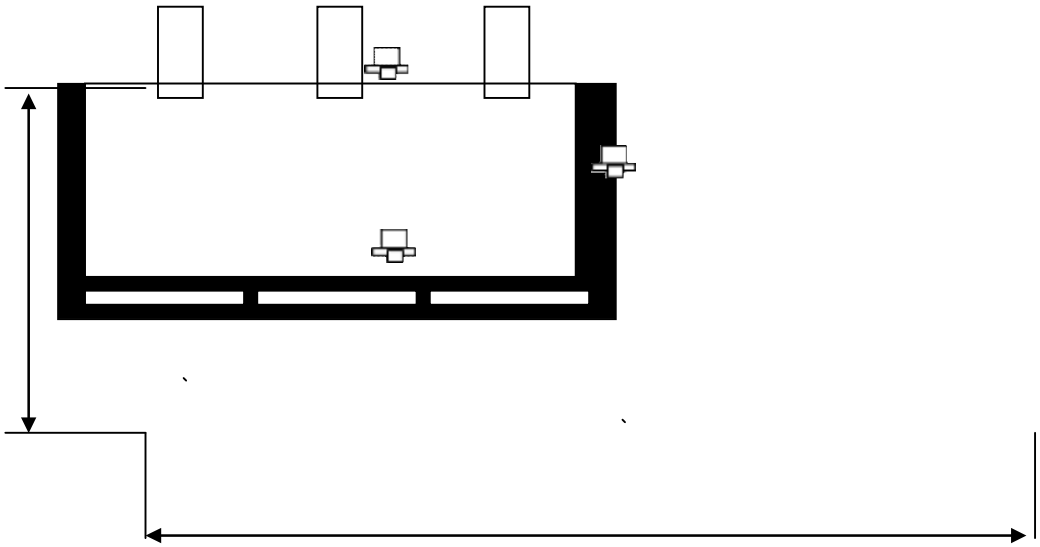


Рисунок 6.1 - Схема приміщення.

Згідно НПАОП 0.00-7.15-18 площа S$^{'}$, виділена для одного робочого місця з персональною ЕОМ, повинна бути не менше 6 м^2 і об’єм – не менше 20 м^3. У приміщенні розташовано 4 робочі місця, що повністю відповідає необхідним нормам.

Розрахуємо фактичні значення цих показників, розділивши об’єм приміщення та загальну площу на кількість працюючих.

Отже, виходячи з отриманих результатів за характеристиками площі та об’єму, приміщення відповідає нормам.

Можна зробити висновок, що розміри робочого місця програміста відповідають встановленим нормам, виходячи з заданих параметрів.

6.2. Аналіз потенційно небезпечних і шкідливих виробничих факторів на робочому місці

При створенні системи аналізу та візуалізації робота виконується сидячи без фізичних зусиль, тому відноситься до категорії легка Іа.

Під час роботи на працівника діє ряд небезпечних і шкідливих чинників, які наведені у табл. 6.3 та табл. 6.4.

Таблиця 6.2

Характеристики робочого місця

№	Найменування параметру	Значення	
		фактичне	нормативне
1.	Висота робочої поверхні, мм	780	680 – 800
2.	Ширина робочої поверхні, мм	1500	не менше 600
3.	Глибина робочої поверхні, мм	750	не менше 600
4.	Висота простору для ніг, мм	750	не менше 600

5.	Ширина простору для ніг, мм	800	не менше 500
6.	Глибина простору для ніг, мм	750	не менше 450
7.	Висота поверхні сидіння, мм	480	400 – 500
8.	Ширина сидіння, мм	500	не менше 400
9.	Глибина сидіння, мм	500	не менше 400
10.	Висота опорної поверхні спинки, мм	550	не менше 300
11.	Ширина поверхні спинки, мм	470	не менше 380
12.	Довжина підлокітників, мм	300	не менше 250
13.	Ширина підлокітників, мм	60	50 – 70
14.	Відстань від очей до екрану, мм	650	600 – 700

Таблиця 6.3

Шкідливі чинники на робочому місці

Фізичні	Психофізіологічні
Підвищений рівень шуму	Розумове перенапруження
Підвищений рівень електромагнітного випромінювання	Монотонність праці
Підвищений рівень статичної електрики	Перенапруження аналізаторів
Недостатній рівень освітленості	
Неоптимальний мікроклімат	

Таблиця 6.4

Аналіз шкідливих факторів, пов’язаних з мікрокліматом

№	Шкідливий фактор	Наслідки
1	Відхилення вологості повітря від оптимальних параметрів	Тимчасове погіршення самопочуття і зниження працездатності, хвороби, роздратованість
2	Відхилення t від оптимальних параметрів	Відсутність теплового комфорту, тимчасове погіршення самопочуття і зниження працездатності, хвороби
3	Відхилення V руху повітря від оптимальних параметрів	Тимчасове погіршення самопочуття і зниження працездатності, хвороби

У таблиці 6.5 та 6.6 наведені нормативні та фактичні показники мікроклімату.

Таблиця 6.5

Мікроклімат в теплий період року

Параметр мікроклімату			
Найменування	Значення		
	Фактичне		Оптимальне
t, ^{o}C	21	21 – 23	18 –27
w, %	55	60 – 40	до 75
V, м/с	0,2	0,3	0,4 – 0,2

Таблиця 6.6

Мікроклімат в холодний період року

Параметр мікроклімату			
Найменування	Значення		
	Фактичне		Оптимальне
t, ^{o}C	18	21 – 23	18 –27
w, %	70	60 – 40	до 75
V, м/с	0,4	0,3	0,4 – 0,2

Заходи для запобігання встановлених мікрокліматичних порушень норм подані в таблиці 6.7.

Таблиця 6.7

Запобіжні заходи в теплий та холодний періоди року

№	Технічні	Організаційні	ЗІЗ
1	Контроль параметрів за допомогою анемометра Extech AN100; використання кондиціонеру DEKKER DSH105R/G (для кондиціонування і провітрювання)	відсутні	відсутні
2	Контроль параметрів за допомогою термометра La Crosse WS8005; використання кондиціонеру DEKKER DSH105R/G (для кондиціонування і провітрювання)	Перерви в роботі з метою провітрювання кімнати; вологе прибирання на робочих місцях	відсутні

3	Контроль параметрів за допомогою психрометра Т-04; використання зволожувача повітря ZELMER AH1500	Перерви в роботі з метою провітрювання кімнати; вологе прибирання на робочих місцях	відсутні

Приміщення для роботи мають бути обладнані системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією відповідно до ДБН В.2.5-67:2013. Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин відповідають вимогам ДСН 3.3.6.042-99, ГН 2152-80, ГОСТ 12.1.005-88, ДСТУ ГОСТ 12.0.230:2008 та ДСТУ ГОСТ 12.4.041:2006. Під вентиляцією розуміють сукупність заходів та засобів, призначених для забезпечення на постійних місцях та зонах обслуговування приміщень метеорологічних умов та чистоти повітряного середовища, що відповідають гігієнічним та технічним вимогам. Основне завдання вентиляції – вилучити із приміщення забруднене, вологе або нагріте повітря та подати чисте свіже повітря.

Джерелами шуму в приміщенні є вентилятор системного блоку, ноутбуку та кондиціонер (табл. 6.8). Звук, що створюється вентилятором та кондиціонером, можна класифікувати як постійний.

Таблиця 6.8

Джерела шуму

Джерело шуму	Фактичний рівень шуму, дБ	Оптимальний рівень шуму, дБ	Час роботи, год.
Кондиціонер DEKKER SH105R/G	22	< 50	8
Кулер комп’ютеру HP Probook 4530s	20		8

Наслідки шуму та вібрації подано у таблиці 6.9.

Таблиця 6.9

Шум і вібрація

Шкідливий фактор	Наслідки
Підвищений рівень шуму	Погіршення слуху, підвищення ймовірності виникнення помилки, зниження продуктивності роботи
Вібрації на робочому місці	Роздратування, зниження працездатності, погіршення самопочуття

Запобіжні заходи, які здійснюються для уникнення наслідків шкідливих факторів, наведено в табл. 6.10.

Таблиця 6.10

Запобіжні заходи

№	Технічні	Організаційні	ЗІЗ
1	Контроль параметрів за допомогою приладу для виміру шуму DT-8852; якісний монтаж окремих вузлів комп’ютера	Проведення планового попереджувального ремонту (чищення від пилу і інших забруднень)	Відсутні
2	Контроль параметрів за допомогою приладу для виміру вібрацій TV260; встановлення спеціальної підставки під ноутбук	Проведення планового попереджувального ремонту (чищення від пилу й інших забруднень)	Відсутні

Відповідно до ДБН В.2.5-28:2018 робота відноситься до розряду зорових робіт. Передбачається використання природного, штучного та змішаного

освітлення. В табл. 6.11 наведені шкідливі фактори порушень норм яскравості світла.

Таблиця 6.11

Шкідливі фактори порушень норм яскравості світла

№	Шкідливий фактор	Наслідки
1	Недостатня освітленість робочої зони	Погіршення зору і самопочуття, втомлюваність, підвищення ризику здійснення помилки
2	Підвищена яскравість світла	

У таблиці 6.12 відображено фактичні та оптимальні значення для параметрів освітлення.

Таблиця 6.12

Параметри освітлення

Найменування	Значення	
	Фактичне	Оптимальне
При змішаному освітленні	450	400
При загальному освітленні	300	300
Коефіцієнт природного освітлення	1,23	1,2

Для уникнення наслідків неправильного освітлення вживаються такі запобіжні заходи (табл. 6.13).

ЕОМ є однофазним споживачем електроенергії, що живиться від змінного струму 220В від мережі із заземленою нейтраллю. ПК відноситься до

електроустановок до 1000В закритого виконання, всі струмопровідні частини знаходяться в кожухах. За способом захисту людини від ураження електричним струмом, ЕОМ і периферійна техніка повинні відповідати 1 класу захисту.

Технічні методи захисту від ураження струмом зводиться до застосування струму безпечної напруги, захисту у випадку випадкового доторкання до струмоведучих частин і від надмірних струмів, захисту у випадку переходу напруги на неструмоведучі металеві частини установки.

Безпечну напругу одержують від сітки підвищеної напруги (110-120 В) за допомогою знижувальних трансформаторів.

Таблиця. 6.13

Запобіжні заходи

№	Технічні	Організаційні	ЗІЗ
1	Контроль параметрів за допомогою люксметра DT-1308; використання нових світильників загального призначення ELSTEAD FINSBURY PARK FP6 POL NICKEL; урахування природного освітлення кімнати	Встановлення мінімального рівня освітлення; чищення скла вікон та світильників; заміна ламп, що перегоріли	Додаткове освітлення на робочих місцях (світильники DeLux Décor TF-05); окуляри для роботи з комп’ютером.
2	Контроль параметрів за допомогою люксметра DT-1308; використання регульованих пристроїв для відкривання вікон, а також жалюзі; використання світильників нового типу	Відсутні	Окуляри для роботи з комп’ютером.

Захисту від доторкання до струмоведучих частин установки досягають за допомогою ізоляції, відгородження застосування блокуючих пристроїв запобіжної сигналізації та неприступності розташування установок.

Розподільні щитки поміщають у закриті металеві кожухи-ящики.

Запобіжну сигналізацію застосовують у вигляді плакатів і надписів. Найкращими світловими сигналізаціями є подвійні, яких при наявності напруги горить червона лампочка, а при її відсутності - зелена.

Захист від надмірних струмів – короткого замикання і струмів перевантаження, які можуть спричинити займання ізоляції, здійснюється запобіжниками й автоматичними вимикачами, а захист від переходу напруги на струмоведучі частини за допомогою захисного заземлення і захисного вимикання.

В табл. 6.14 наведені небезпечні фактори ураження людини електричним струмом.

Таблиця 6.14

Небезпечні фактори ураження людини електричним струмом

№	Шкідливий фактор	Наслідки	Заходи
1	Небезпечний рівень напруги струмопровідних частин обчислювальної та побутової техніки	Зростання ризику ураження електричним струмом	Релейний захист струму дотику, захисні заземлюючі корпуси. Попереджувальні знаки про рівень напруги.

У таблиці 6.15 відображено фактичні та оптимальні значення для параметрів електропостачання.

Таблиця 6.15

Параметри електропостачання на робочому місці

Значення	Напруга, В	Частота, Гц	Тип розетки/вилки	Тип фази
Фактичне	220	50	F	Однафазна, трипровідна
Оптимальне	220	50	C, F	Однофазна, трипровідна

Вживаються такі запобіжні заходи для уникнення наслідків ураження людини електричним струмом (табл. 6.16):

Таблиця 6.16

Запобіжні заходи

№	Технічні	Організаційні	ЗІЗ
1	Релейний захист струму дотику, захисні заземлюючі корпуси	Проведення робіт з електричним обладнанням лише проінструктованим персоналом. Створення плану короткострокових відпочинків.	відсутні

Запобігання пожежі досягається виключенням утворення джерел загорянь і горючого середовища. У таблиці 6.17 приведено шкідливі фактори.

Таблиця 6.17

Шкідливі фактори, пов’язані з пожежною безпекою

№	Шкідливий фактор	Наслідок
1	Коротке замикання, електротравми, пожежі, летальні наслідки	Коротке замикання, пожежі, електротравми, летальні наслідки
2	Коротке замикання	Електротравми, пожежі, летальні наслідки
3	Порушення протипожежного режиму	Електротравми, пожежі, летальні наслідки

В цьому приміщенні можливі пожежі таких класів: А – горіння твердих речовин, Е – горіння електроустановок під напругою. Для забезпечення цих категорій застосовуються заходи, що вказані в таблиці 6.18.

Таблиця 6.18

Запобіжні заходи

№	Технічні	Організаційні	ЗІЗ
1	Контроль параметрів за допомогою термометра La Crosse WS8005; використання кондиціонеру DEKKER DSH105R/G (для кондиціонування і провітрювання)	Розвантаження електровузлів після виконання роботи; ознайомлення з інструкціями по використанню електроприладів;	відсутні
2	Наявність вогнегасника порошкового типу ОП-5 та автоматичної системи “ГАРАНТ-Р” (ПО-2), узгоджений план евакуації	Ознайомлення з інструкціями по використанню протипожежних засобів; узгоджений план евакуації	відсутні

3	Наявність вогнегасника порошкового типу ОП-5 та автоматичної системи “ГАРАНТ-Р” (ПО-2), узгоджений план евакуації	Ознайомлення з інструкціями по використанню протипожежних засобів; узгоджений план евакуації	відсутні

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розглянуто принцип роботи макросів та ботів. В ході аналізу принципу роботи макрокоманд було виявлено їх недоліки, для усунення яких були побудовані алгоритми для створення макросів та ботів. Також, згідно побудованих алгоритмів було розроблено програмний модуль для автоматизованого створення макросів та ботів на базі об’єктно-орієнтованої мови програмування Python. До того ж, було побудовано інтерфейс для зручного керування програмним модулем.

Також у розробленому програмному модулю було досліджено параметри швидкості виконання макросів та точності ботів, під час їх пошуку об’єктів на екрані. В результаті дослідження швидкодії ботів, та згідно розрахунків, створений бот може знаходити об’єкти на 133% швидше, ніж користувач.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://www.techopedia.com/definition/3833/macro

2. https://www.techtarget.com/whatis/definition/bot-robot

3. https://pyautogui.readthedocs.io/en/latest/index.html

4. https://humanbenchmark.com/tests/aim

5. https://creately.com/ru/lp/Программа-блок-схем-онлайн/

6. https://docs.python.org/3/library/tkinter.html

7. https://pypi.org/project/mouse/

8. https://pypi.org/project/keyboard/

9. https://pypi.org/project/pynput/

10. https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

11. https://pillow.readthedocs.io/en/stable/

12. https://docs.python.org/3/library/pickle.html