

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ
СІКОРСЬКОГО"**

Факультет електроніки

(повна назва інституту/факультету)

Акустичних та мультимедійних електронних систем

(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

 С.А. Найда

(ініціали, прізвище)

" 07 " 06 2021

р.

**Дипломна робота
на здобуття ступеня бакалавра**

зі спеціальності

171 Електроніка

(код і назва)

на тему:

"Розроблення програмного забезпечення для створення
колажів"

Виконав:

студент IV курсу, групи ДВ-72

(шифр групи)

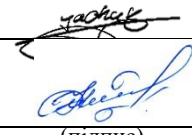
Дьячук Дмитро Вікторович

(прізвище, ім'я, по батькові)

Керівник

асистент, к.т.н., Філіпова Н.Ю.

(посада, вчене звання, науковий ступінь, прізвище, ініціали)



(підпис)

Консультант

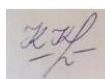
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище,
ініціали)

(підпис)

Рецензент

доцент каф. ЕПС, к.т.н., доц. Катерина КЛЕН

(посада, вчене звання, науковий ступінь, прізвище, ініціали)



(підпис)

Засвідчую, що у цій дипломній
роботі немає запозичень з праць
інших авторів без відповідних
посилань.

Студент



(підпис)

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут/факультет _____ Факультет електроніки
(повна назва)
Кафедра _____ Акустичних та мультимедійних електронних систем
(повна назва)
Рівень вищої освіти – перший (бакалаврський)
Спеціальність (спеціалізація) _____ 171 – Електроніка
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій НАЙДА

(ініціали, прізвище)

«07 ____» ____ 06 ____ 20 ____ р.

ЗАВДАННЯ

на дипломну роботу студенту

Дьячуку Дмитру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Розроблення програмного забезпечення для створення
колажів

Керівник роботи _____ Філіпова Наталія Юріївна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «24» травня 2021 р. №1316-с

2. Строк подання студентом роботи _____ 12.06.2021

3. Вхідні дані до роботи _____ Крос-платформовий інструментарій
розробки програмного забезпечення мовою програмування C++ QT

4. Зміст пояснювальної записки (перелік завдань, які потрібно розробити)

Історія, особливості та сучасні методи створення колажу. Методи, вимоги та особливості створення програмного забезпечення. Огляд середовища проектування. Розроблення програми для створення колажів.

5. Перелік графічного (ілюстративного) матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
Презентація з результатами розроблення програмного забезпечення

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

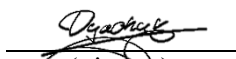
7. Дата видачі завдання 12.04.2021р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1	Затвердження плану дипломної роботи	14.04.2021	Виконано
2	Написання першого розділу	02.05.2021	Виконано
3	Написання другого розділу	15.05.2021	Виконано
4	Написання третього розділу	29.05.2021	Виконано
5	Підготовка та оформлення презентації для доповіді	13.06.2021	Виконано

Студент

Керівник роботи




(підпис)

Дьячук Д.В.

(ініціали, прізвище)

Філіпова Н.Ю.

(ініціали, прізвище)

УДК 004.4

РЕФЕРАТ

Дьячук Д.В. «Розроблення програмного забезпечення для створення колажів»: дипломна робота бакалавра: 171 Електроніка. - КПІ ім. Ігоря Сікорського, Київ, 2021. – 79 с.

Дипломна робота: 79 с., 25 рис., 1 дод., 13 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ФОТОКОЛАЖ, C++, QT CREATOR, КОМПОЗИЦІЯ, МОВА ПРОГРАМУВАННЯ.

Об'єктом дослідження є програмне забезпечення з відповідним функціоналом.

Предметом дослідження є мова програмування C++ з середовищем програмування Qt Creator.

Метою роботи є аналіз особливостей створення колажів та розробка програмного забезпечення з відповідним функціоналом.

У результаті виконання дипломної роботи проаналізовано техніки колажів. Досліджено методи, вимоги та особливості створення програмного забезпечення. Розглянуто середовище програмування Qt Creator. Виділено особливості мови програмування C++. А також розроблено програмне забезпечення для створення фотоколажів.

Галузь застосування: організація створення колажів у особистих та комерційних цілях.

ABSTRACT

Dyachuk Dmytro "Software development for creating collages": Bachelor's Thesis: 171 Electronics. - KPI them. Igor Sikorsky, Kyiv, 2021. - 79 p.

Thesis: 79 pages, 25 figures, 1 appendix, 13 sources.

SOFTWARE, PHOTO COLLAGE, C ++, QT CREATOR, COMPOSITION, PROGRAMMING LANGUAGE.

The object of study is software with appropriate functionality.

The subject of the study is the C ++ programming language with the Qt Creator programming environment.

The aim of the work is to analyze the features of creating collages and develop software with appropriate functionality.

As a result of the thesis work collage techniques are analyzed. Methods, requirements and features of software creation are investigated. The Qt Creator programming environment is considered. Features of the C ++ programming language are highlighted. And also developed software for creating photo collages.

Field of application: organization of collages for personal and commercial purposes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП	9
1 ІСТОРІЯ, ОСОБЛИВОСТІ ТА СУЧАСНІ МЕТОДИ СТВОРЕННЯ КОЛАЖУ	11
1.1 Історія виникнення мистецтва колажу	11
1.2 Особливості колажів та галузі застосування	13
1.3 Особливості техніки композиції	16
Висновки до розділу	19
2 МЕТОДИ, ВИМОГИ ТА ОСОБЛИВОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ОГЛЯД СЕРЕДОВИЩА ПРОЕКТУВАННЯ	20
2.1 Методологія розробки програмного забезпечення	20
2.1.1 Походження	21
2.1.2 Класифікація	21
2.1.3 Приклади методологій	22
2.2 Вимоги до проектування програмного забезпечення	24
2.2.1 Види вимог за рівнями	25
2.2.2 Види вимог за характером	25
2.2.3 Джерела та методи знаходження вимог	26
2.2.4 Документування вимог	26
2.2.5 Вимоги в процесах розробки	27
2.3 Процес розробки програмного забезпечення	27
2.4 Огляд середовища програмування	28
Висновки до розділу 2	29
3 РОЗРОБЛЕННЯ ПРОГРАМИ ДЛЯ СТВОРЕННЯ КОЛАЖІВ	30
3.1 Огляд мови програмування C++	30
3.1.1 Основи програмування на C++	31
3.1.2 Змінні в C++	32
3.1.3 Арифметичні операції в C++	32
3.1.4 Умовні вирази в C++	33

3.1.5 Умовні конструкції в C++	34
3.1.6 Цикли в C++	35
3.1.7 Масиви в C++	35
3.1.8 Декілька слів про функції, вказівники та класи в C++	35
3.2 Опис створення та огляд програмного забезпечення	36
Висновки до розділу 3	46
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	49
Додатки	51

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

MVS	– Microsoft Visual Studio
STL	– Standard Template Library
QML	– Qt Modeling Language
IDE	– Integrated Development Environment
SRS	– Software Requirements Specification
XP	– Extreme Programming
ООП	– Об’єктно орієнтоване програмування
ПЗ	– Програмне забезпечення

ВСТУП

Багатьом маркетологам важко швидко обрати зображення для просування товарів чи послуг без допомоги дизайнера. Влучного гасла та гарного опису часто буває недостатньо, щоб привести клієнта до кінцевої точки - покупки. Тому потрібен яскравий та особливий візуал.

Один з ефективних і доступних дизайнерських прийомів це створення колажів з фотографій. З їх допомогою можна одночасно вирішити багато завдань, а саме: привернути увагу за рахунок незвичайного дизайну; показати товар або послугу відразу з декількох сторін.

Також фотоколажі використовуються багатьма людьми в особистих цілях. Адже якісний фотоколаж можна подарувати рідним або близьким, він буде унікальним подарунком для будь-якої людини і прикрасить собою інтер'єр квартири або офісу.

Актуальність роботи полягає в необхідності створення багатофункціонального програмного забезпечення для створення фотоколажів.

Об'єктом дослідження є програмне забезпечення з відповідним функціоналом.

Предметом дослідження є мова програмування C++ з середовищем програмування Qt Creator.

Метою роботи є аналіз особливостей створення колажів та розробка програмного забезпечення з відповідним функціоналом

Для досягнення мети необхідно:

- виділити особливості колажів;
- проаналізувати техніки композицій;
- провести аналіз методів, вимог та особливостей створення програмного забезпечення;
- провести огляд середовища проектування;
- розглянути особливості мови програмування C++;
- розробити відповідне програмне забезпечення для створення фотоколажів.

Методом дослідження є теоретичне дослідження техніки композицій та особливостей мови програмування C++.

Новизна роботи полягає у пошуку кращих рішень написання програмного забезпечення з відповідним функціоналом для полегшення створення колажів.

Практична цінність полягає в тому, що є можливість більш глибоко зануритись у вивчення існуючих особливостей техніки композицій та розкрити великий функціонал мови програмування.

1 ІСТОРІЯ, ОСОБЛИВОСТІ ТА СУЧАСНІ МЕТОДИ СТВОРЕННЯ КОЛАЖУ

1.1 Історія виникнення мистецтва колажу

Існує закореніла дата народження мистецтва колажу, багато науковців схильні до думки, що саме як прийом в мистецтві, він був заснований, коли Пабло Пікассо створив картину «Натюрморт із плетеним стільцем» у 1912 році. Проте, в період з 1901 - 1906 рр., Михайло Врубель (рис. 1.1) та Іван Єфімов (рис. 1.2) в своїх роботах, для збільшення об'єму почали використовувати фольгу і папір інший за кольором та фактурою, які наклеювались на фундамент роботи. З'явлення цього прийому непоодинокі явище і є не суто європейським, від початку століття ідеєю колажу було пройняте мистецтво не тільки Європи, але і Росії [2].

Поняття «колаж» має узагальнюючий характер і вільно використовуються у різноманітних сферах мистецтва. Слово «колаж» французького походження від «collage» і перекладається як «наклеювання». Термін існує в межах: кіномистецтва, живопису та музики і його визначають по-перше, як термін живопису, запропонований представниками кубізму, а потім підхоплений сюрреалістами і футуристами для упорядкування власної художньої практики, по-друге - поєднання неоднорідних елементів. Великий енциклопедичний словник трактує колаж таким чином: як прийом в образотворчому мистецтві, суть якого в наліпленні на будь-яку основу елементів, що різняться з нею за фактурою і кольором. У ХХ ст. в сфері музики, колаж вважався винятковою формою використання композитором в своєму творі частин іншого твору: чужого або свого раніше написаного [1].

Доцільно звернути увагу, що техніка колажу з часом суттєво не змінювалась, але незважаючи на це, колаж пройшов певну еволюцію в мистецтві ХХ століття. У кубізмі колаж проявляється як засіб організації простору та руйнування традиційного зорового сприйняття, як вираз суб'єктивного світогляду художника.

У поп-арті і дадаїзмі колаж був засобом заперечення логічного мислення, руйнування матеріального світу, домінуючим було відчуття абсурдності і хаосу, переважала загадковість, незрозумілість. Провідним у колажах дадаїстів виявляється невиразна пустота в контрасті з райдужним злиттям фарб. Художники театрального-декораційного мистецтва 1910 - х рр. доцільно запроваджували метод колажу в мистецтво театру. Використовувались неформальні матеріали (залізо, мотузка, рогожа, скло, синтетичні матеріали) для створення ескізів декорацій. Колаж як мистецький прийом отримав «нове дихання» з розвитком технічного прогресу та появою потужних комп'ютерів вже у ХХІ столітті у вигляді комп'ютерного колажу, що говорить про його життєздатність та винятковість. Сьогодні техніка колажу має свою специфіку, хоча як прийом він застосовувався і попередніми поколіннями. Якщо раніше він був лише одним з можливих прийомів поруч з іншими, то в сучасному мистецькому просторі колаж займає центральну роль і невід'ємне значення в сучасній образотворчій культурі. Колаж має широкі виразні можливості, він може бути упорядкованим за техніками, часом створення, матеріалами, прийомами виконання та композиціями. Таким чином, вивчення колажу має перспективи, оскільки, цей прийом стає демонстрацією змін, що відбуваються в сучасному мистецтві [1].



Рисунок 1.1 - Приклад колажу створеного Михайлом Врубелем [3]



Рисунок 1.2 – Приклад колажу створеного Іваном Єфімовим [3]

1.2 Особливості колажів та галузі застосування

Популярні на сьогоднішній день колажі в наступних галузях:

У фотомистецтві: Д. Хілл і Р. Адамсон у 1843 р. створили композицію - фотомонтаж з 500 фотографій. У світі з'явився перший фотоколаж, який називався "Засідання Вільної церкви в Шотландії" (рис. 1.3). Завдяки особливому поєднанню багатьох фотографій в одній картинці вийшло захоплююче творіння [3].

Усякі фотоколажі можуть сильно відрізнятися за стилем і жанром. З появою цифрової фотографії фотоколажі почали створювати з використанням програмного забезпечення для редагування зображень, що називається фотомонтаж.



Рисунок 1.3 - "Засідання Вільної церкви в Шотландії" [3]

У кінематографі і сфері художньої обробки відео:

В кінематографі колаж як метод творчості використовував режисер Сергій Параджанов. Відеоколлаж - це ряд відео, створений з невеликих уривків з одного або декількох фільмів, інколи з додаванням фото зображень і тексту. За допомогою відеоколлажу створюється яскравий сюжет, що точно відображає творчу задумку автора [4].

У музиці:

У XX сторіччі, в період пошуку нових ідей в мистецтві, дуже відомі композитори в своїх роботах використовували елементи з творів колег. Такий технічний прийом актуальний і на сьогоднішній день. У музиці термін «колаж» означає особливу форму використання композитором в своєму творі фрагментів іншого твору композитором в своєму творі. Фрагменти з інших творів можуть бути різних стилів. Один з перших колажів створив Ч. Айвз. на початку XX століття. Надалі цю техніку використовували Д. Д. Шостакович та інші [5].

У літературі:

Шанувальники модернізму, футуризму і сюрреалізму, додають в свої твори чужі документи і цитати з робіт інших письменників. У літературі XX ст. колажем характеризуються напрямки, які користувались технікою фрагментації

тексту, склеєного з не пов'язаних зовнішньою логікою епізодів - цитат, документів і т.п. Це виводить читача на глибокі емоції і стає предметом бурхливого обговорення книги.

Особливості колажу, особливості техніки:

При створенні колажу використовуються найрізноманітніші декоративні елементи задля того, щоб вироби виконані в даній техніці були особливими та ефектними. Водночас, художник старається створити колаж так, щоб глядач сприймав його як єдине гармонійне творіння. Звичайно, при цьому потрібно дотримуватись всіх правил побудови композиції [10].

Стилі колажу:

На сьогодні нам відомо всього чотири основні стилі, які використовуються для виготовлення таких виробів [3].

- **Формо-лінійний.** Центральними елементами подібних колажів є геометричні фігури-квадрати, кола, ромби і т.д. Навіть фон в таких творах є поєднанням цих простих елементів.

- **Декоративний.** Дані колажі різняться фантастичним сюжетом. Основною метою, яку художник ставить перед собою, виконуючи подібну роботу, є неповторність і оригінальність. Глядач, дивлячись на таку картину, старається відгадати ідею автора, його настрій.

- **Вегетативний.** В ролі основних фрагментів використовуються рослини або їх елементи. В той же час художник повинен зробити готовий твір максимально реалістичним. В такому випадку рослини виступають не частиною пейзажу або загального сюжету картини, а як окремий елемент, на якому з акцентовано увагу.

- **Пейзажний.** Назва цього виду колажу говорить сама за себе. На таких зображено різного роду пейзажі - міські або природні. Це може бути ліс, будівлі, море і т.п.

1.3 Правила створення фото композицій та композицій в колажі.

Нюанси створення.

З латинської, слово "композиція" переводиться як твір, з'єднання, складання, тобто встановлення зв'язку між окремими його частинами або компонентами, що вкінці утворюють єдине ціле - завершене і закінчене фотографічне зображення [10].

Під композицією розуміють всю систему, весь образотворчий лад знімка, створений фотографом з метою розкриття ідеї та тематики, та з метою досягти виразності та чіткості художньої форми. В широкому розумінні композиція є поєднанням всіх фрагментів сюжету і образотворчої форми в фотознімку. Її визначають: напрямки основних ліній, напрямки, що відбуваються в кадрі руху, розміщення фігур і предметів. В процесі роботи над композицією кадру всі ці складові елементи з'єднуються в певну гармонійну систему, робиться чітка і ясною образотворча форма знімка, через яку зміст виражається зі всією необхідною повнотою [4].

Основи фотокомпозиції повинні бути викладені досить конкретно і пов'язані з технічними засобами фотографії образотворчими прийомами, що потім дозволяють отримати те чи інше композиційне рішення знімка. Тут не можливо обійтися формулюванням одних тільки загальних положень, з тієї причини, що тільки вони залишають достатній простір, необхідний для справжньої творчості. Основними прийомами в процесі створення колажу є: стилізація фотографій, заміна фонового зображення, ретуш, додавання спецефектів і нових предметів, робота з яскравістю і контрастністю зображення. Прийоми колажування можуть бути використані повністю для всього зображення або ж застосовуються тільки до окремих частин колажу.

Процес створення колажу включає [5]:

- Розробка ідеї та художньої концепції колажу;
- Зміна фону або ж його видалення;
- Додавання зайвих або видалення нових елементів;

- Використання спецефектів;
- Промальовування деталей.

Фотоколаж - одна з найбільш популярних різновидів класичного колажу. В даному випадку метою є створення креативної композиції з фотографій, розміщених химерним чином по відношенню один до одного. Під час створення колажу із фотографій потрібно завжди приділяти увагу наступним характеристикам [4]:

- Ракурс;
- Насиченість;
- Кольори та рефлекси;
- Освітлення, напрямки та наявність тіней;
- Відповідність співвідношень розмірів елементів;
- Контраст та яскравість;
- Різкість.

Основи створення фотоколажу:

- Вибір теми.

Перед тим як розпочати реалізацію фотоколажу, варто визначитися з його настроєм і темою, а також враженням, яке він повинен надавати на глядача [4].

- Вибір програми для створення колажів.

В нашому столітті ми можемо дозволити собі завантажити на ПК той чи інший графічний редактор і з його допомогою створити фотоколаж за невеликий проміжок часу.

- Техніки і прийоми, які використовуються в фотоколажі.

Як і будь-якому іншому творчому напрямку, фотоколажу притаманні характерні прийоми роботи, кожен з яких по-своєму особливий і специфічний.

- Накладення однієї фотографії на іншу.

Досить стандартна схема виконання колажу: як правило, на макет підготовлений заздалегідь, просто нашаровуються різноманітні зображення. Вони можуть бути

розташовані за годинниковою стрілкою, по вертикалі або діагоналі, групуватися або йти парами.

- Складання пазла з безладного набору фотографій.

Такий метод поширений в графічному дизайні - його також називають мозаїчним фотоколажем. Принцип дії полягає в наступному: вибирається заголовна фотографія, яка і буде служити пазлом, а також безліч картинок, які будуть цей пазл складати. Далі всі елементи фотомозаїка будуть впорядковані за розміром і кольором в точній відповідності з вихідним зображенням, в результаті виходить повноцінна композиція, що складається з дрібних шматочків.

- Поєднання декількох фотографій в одне ціле.

Просте заповнення макета зображеннями до упору-один з найбільш звичних методів створення будь-якого колажу. Бажано підбирати кожен кадр невеликого розміру - чим більше фотографій увійде, тим цікавіше буде розглядати їх в кінцевому підсумку. Колірна гамма фотознімків також відіграє велику роль. Аби вони не зливалися один з одним і виглядали яскраво, варто відібрати з поля зору невдалі, тьмяні, засвічені зображення.

Приклад створеного фотоколажу можна побачити на рисунку 1.4.



Рисунок 1.4 - Приклад готового фотоколажу

Висновки до розділу 1

У розділі проаналізовано історію виникнення мистецтва колажу, та розглянуто поняття «колаж». Висвітлено, що раніше при створенні колажу одночасно використовувалися декілька зовсім різних по факторі та походженню матеріалів. Проведено аналіз особливостей колажів. Розглянуто галузі застосування колажів, такі як: кінематографія і сфера художньої обробки відео, музика, література. Аналізовано стилі колажу, та виділено основні чотири: формолінійний, декоративний, вегетативний, пейзажний. Опрацьовано правила створення фотокомпозицій та композицій в колажі. А отже процес створення колажу включає розробку ідеї та художньої концепції колажу, зміну фону або ж його видалення, додавання зайвих або видалення нових елементів; використання спецефектів та промальовування деталей.

2 МЕТОДИ, ВИМОГИ ТА ОСОБЛИВОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. ОГЛЯД СЕРЕДОВИЩА ПРОЕКТУВАННЯ

2.1 Методологія розробки програмного забезпечення

На різноманітних етапах життєвого циклу розробки програмного забезпечення застосовується сукупність методів, що забезпечені методологією розробки програмного забезпечення. Вони пов'язані спільним філософським підходом та згідно до цього підходу, дозволяють гарантувати найвищу ефективність прогресу розробки [6].

Кожна методологія розробки програмного забезпечення наділена характеристиками за:

- Концепцією;
- Множиною моделей, що застосовуються для здійснення методології;
- Філософією підходу та провідними принципами.

Існує такий термін, як «парадигми програмування», що визначається як метод з врегульованими характеристиками, а точніше філософією, концепцією та множиною моделей .

На сьогоднішній день у світі програмування існують різні методології або парадигми програмування. Найбільш поширеними та відомими парадигмами є [6]:

- Спіральна модель (spiral model);
- Макетування (prototyping);
- Водоспадна модель (waterfall);
- Екстремальне програмування (extreme programming);
- Швидка розробка програмного забезпечення (rapid application development);
- Інкрементна та ітеративна розробка (incremental and iterative development);
- Різні види методологій гнучкої розробки (agile methodology).

2.1.1 Походження

Проаналізувавши методології можна дослідити три види їх формування [6].

- Перший вид: виникнення з досвіду розробки ПЗ.
- Другий вид: походження від імперативних моделей, таких як: нормальні алгоритми Маркова (логічне програмування), лямбда-числення Черча (функціональне програмування), рекурсивні функції Гільберта і Аккермана (структурне програмування), абстрактна машина Тюрінга (імперативне програмування).

- Третій вид: методології можна тлумачити через віддзеркалення однієї з трьох структур мови моделювання на структуру мови програмування. Компонентами можуть бути логіка, структура даних, структура управління. Кожне з дев'яти відображень характеризує або методологію, або дійсно серйозний метод програмування. Наприклад, відображення логіка-логіка є основою логічного програмування.

2.1.2 Класифікація

Щодо класифікації методологій розробки програмного забезпечення, їх розділяють за топологічною специфікою, ядрами, та за специфікою реалізації.

Класифікація методологій за топологічною специфікою - прийом вибору методів для уточнення ядра методології. Загальні витрати на розробку ПЗ можуть виступати критерієм якості тієї чи іншої типології. У свою чергу, витрати на розробку залежать від основних мовних абстракцій: абстракції управління, даних і модульності. В імперативній методології, наприклад можна досягти більш вигідної топології, з погляду мовних засобів, завдяки дотриманню методів структурного програмування. Результатом є методологія структурного програмування.

Класифікація методологій за ядрами:

- Методологія логічного програмування;
- Методологія функціонального програмування;
- Методологія ООП;
- Методологія імперативного програмування;
- Методологія програмування в обмеженнях.

Класифікація методологій за специфікою реалізацій проводиться згідно з архітектурою апаратного забезпечення.

Виділяють наступні види класифікації за специфікою реалізацій:

- Послідовна реалізація;
- Паралельна реалізація;
- Гібридна реалізація.

2.1.3 Приклади методологій

Перейдемо до таких прикладів методологій, як:

- Open Source;
- Екстремальне програмування;
- SCRUM.

Найбільш поширена методологія з усіх-методологій екстремального програмування (XP). В певній мірі це сталося завдяки тому, що лідери XP, особливо Кент Бек (Kent Beck) наділені перекрасним вмінням - привертати до себе увагу. Тут розкрився талант Кента, що відіграв немалу роль - знаходити і набирати шанувальників свого руху і вести їх за собою. В певному роді, можна сказати, що популярність XP навіть стала проблемою, оскільки ця методологія практично виштовхнула всі інші та разом з ними і ті цінні ідеї, які вони несуть [7].

Виділимо 4 основні компоненти XP: сміливість, комунікація, простота, зворотній зв'язок. Відповідно до них виокремлюють 12 практик, яких мають

дотримуватись проекти, які застосовують ХР. Велика частина з цих практик представляють з себе старі перевірені техніки, що, у всякому разі, більшість вже забула (охоплюючи переважну частину очікуваних процесів). ХР не лише «повертає» до життя дані техніки, а й скріплює їх так, що в результаті всі вони підтримують і підсилюють один одного [7].

На фоні цього ХР створює новаторський метод проектування. В основі цього методу лежить реорганізація коду системи в момент ітерації. Разом з цим, планується тільки та функціональність, яка має відношення до поточної ітерації, при цьому потреби що можуть з'явитися в майбутньому, не беруться до уваги. Процес, що вийшов в кінці має підтримуватися дисципліною розробників і водночас суміщати її з високим рівнем адаптивності. Підставою для думок, що ХР являється найрозвиненішою адаптивною методологією, є саме таке дивне поєднання описане вище.

Можливо виділити один або більше координаторів в проектах, яким властивий відкритий вихідний код. Людина, що являється лідером в проекті та яка може безпосередньо вносити корективи в сховище вихідного коду, зветься координатором. Окрім координатора, другі розробники теж можуть вносити свої зміни, різниця лише в тому, що вони мають спочатку надіслати їх координатору, що перевірить змінений код і вже потім вносить ці зміни до сховища. Найчастіше зміни такого характеру виглядають як патч-файли, це робить значно простішою процедуру такого типу. Отже, керівник проекту узгоджу патчі і слідує за тим, щоб вони збігались з тотальним планом розроблюваного ПЗ [6].

Нормалізувати стан програми можна паралельно, це є однією з властивостей розробки проектів з відкритим вихідним кодом. Завдяки цьому, в ній може брати участь велика кількість людей. Також, натрапляючи на помилку в програмі, вони можуть передати патч лідеру проекту. Виходить, що люди які не координують проектом, відіграють важливу роль, тому що багато часу марнують саме на пошук помилок. До того ж, до роботи такого типу придатні ті, хто не володіє достатнім рівнем навичок проектування.

Люди, що займаються об'єктно-орієнтованими розробками, досить обізнані методологією Scrum. На прикладі інших гнучких методологій, в ній акцентується увага на тому, що врегульований і одноманітний процес підходить лише для розв'язання певних повторюваних проблем у певному і повторюваному середовищі, в якому працюють і певні повторювані розробники.

Відповідно до методу Scrum, проект поділяється на ітерації (спринт), за 30 днів кожна. Команда розробників починають працювати після визначення функціональності на необхідному етапі, що виконується перед початком спринту. Незмінні вимоги протягом одного спринту є необхідною умовою. Кожного дня, розробники в команді збираються короткі збори, що зветься «scrum» (сутичка), під час яких обговорюється план роботи кожного на наступний день. Ще одним пунктом обговорення є введення керівників компаній в курс про проблеми, що перешкоджають просуванню проекту та які важливо вирішити, а також повідомлення про роботу що вже виконана, аби керівники усвідомлювали на якому етапі знаходиться розробка [6].

В методології SCRUM основний акцент робиться на ітеративне планування і відстеження процесу. Загалом SCRUM дуже подібний до інших гнучких методологій і гарно поєднується з правилами кодування XP.

2.2 Вимоги до проектування програмного забезпечення

Існує набір вимог щодо функцій, властивостей та якостей програмного забезпечення, що або перебуває в розробці або планується розробка в майбутньому. Ці вимоги зафіксовані в умовах вимог, графіках прецедентів, і визначені в процесі аналізу вимог [8].

Процес розробки вимог до програмної системи поділяється на декілька стадій:

- Виявлення вимог (збір, визначення потреб зацікавлених осіб та систем);
- Діагностика вимог (тестування на закінченість та цілісність);
- Умови (документування вимог);

- Перевірка вимог.

2.2.1 Види вимог за рівнями

За Карлом Вігерсом виокремлені три наступні рівні вимог до програмного забезпечення [8]:

- Бізнес-вимоги — розкривають призначення ПЗ, можуть бути відображені в документі про бачення (англ. vision) та документі про границі проекту (англ. scope).
- Вимоги користувача — характеризують комплекс доручень користувача, які мають бути вирішені програмою, а також сюжет їхнього вирішення в системі. Такі вимоги можуть бути у формі тверджень, сценаріїв взаємодії, історій користувача, варіантів використання.
- Функціональні вимоги — формулюють «що» має виконувати програмний продукт. Дані вимоги описані в документі «Специфікація вимог до програмного забезпечення» (англ. SRS).

2.2.2 Види вимог за характером

Функціональний характер-вимоги до поведінки системи [8]:

- Вимоги користувача;
- Бізнес-вимоги;
- Функціональні вимоги;
- Нефункціональний характер-вимоги до характеру поведінки системи.

Бізнес-правила-характеризують межі,що формуються відповідно до області дослідження.

Системні вимоги-вимоги до програмних інтерфейсів, надійності, обладнанню.

- Риси якості;
- Обмеження;

- Зовнішні системи та інтерфейси.

2.2.3 Джерела та методи знаходження вимог

Джерелами вимог можуть бути як бізнес-процеси, так і законодавство, також вимоги стандартів, а ще бачення та очікування користувачів системи.

До методів знаходження вимог відносять, як спілкування з потенціальним користувачем в вигляді інтерв'ю або анкетування, так і дослідження бізнес процесів та нормативної документації законодавства, а ще семінари та мозковий штурм [8].

2.2.4 Документування вимог

Щодо документування вимог, вимоги є засобом використання комунікації між зацікавленими особами. Виходить, що вимоги повинні бути легкими та дохідливими і для розробників, і для простих користувачів. Зазвичай представлені у вигляді одного з документів: деталі вимог до програмного забезпечення (англ. Software Requirements Specification, SRS), або технічне завдання [8].

В межах Уніфікованого процесу розробки вимоги представлені у вигляді кількох невизначених документів:

Зразок випадків використання (прецедентів) (англ. Use Case Model) - сукупність характерних сюжетів використання системи. Зображає функціональні (поведінкові) вимоги.

Доповняльна специфікація (англ. Supplementary Specification) - включає нефункціональні вимоги, наприклад вимоги до підтримки та надійності, ліцензування і документування, ще й продуктивності.

Словник термінів (англ. Glossary) виділяє вагомі терміни та визначення. Припустима версія зберігання концепції словника даних, яким зафіксовані вимоги, що мають певний зв'язок з даними, такими як норми верифікації та прийнятні значення.

Терміном «бачення» (англ. Vision) можна узагальнити найсуттєвіші ідеї високого рівня та вимоги, що лежать в основі розробки системи. Так можна назвати стислий оглядовий документ аби швидше увійти в курс функціонування проекту.

Бізнес-правила (англ. Business Rules) або, як ще називають, правила предметної області зображають вимоги або політики, що виходять за межі одного проекту, наприклад, державні норми оподаткування, політика компанії, закони. Можуть виглядати у формі додаткової специфікації або бути окремим артефактом.

2.2.5 Вимоги в процесах розробки

Відмінні методології розробки програмного забезпечення мають різний принцип роботи з вимогами. В застарілій, і мало актуальній моделі водоспаду (waterfall англ.) стадія аналізування та розробки вимог є першочерговою. Відмінною рисою є те, що вона цілком завершується до початку проектування та розробки ПЗ, а останні не починаються до закінчення аналізу вимог. В ітеративних діях розробки, період аналізу та розробки вимог в різноманітному об'ємі присутній на будь-якій ітерації [8].

2.3 Процес розробки програмного забезпечення

Процес виробництва програмного забезпечення (англ. software development process, software process) - операція, що складається зі зв'язних дій, що цілеспрямовані на розробку програмного забезпечення (ПЗ).

Розрізняють декілька прототипів даного процесу, кожен з них зображає власний підхід, у формі завдань або/і діяльності, які виявляються протягом процесу. Вибрати ту чи іншу модель можна згідно з обраною методологією розробки програмного забезпечення. В ході розробки задіяні безліч дисциплін чи підпроцесів, кілька з них продемонстровані нижче. У прототипі водоспаду вони

розташовані одна за одною, в інших подібних процесах їх послідовність або будова змінюється [7].

- Аналіз вимог → Специфікація програмного забезпечення
- Проектування програмного забезпечення
- Програмування
- Тестування програмного забезпечення
- Системна інтеграція
- Впровадження програмного забезпечення
- Супровід програмного забезпечення

2.4 Огляд середовища програмування

Qt Creator - є інтегрованою сферою розробки (IDE) для додатків C ++, QML і JavaScript. Це комбінована кроссплатформенна IDE, за допомогою якої користувачі мають можливість масштабуватися вище коду і формувати підключені пристрої, додатки і призначені для користувача інтерфейси [11].

ПЗ представляє з себе цілісну складну консоль, за допомогою якої розробники можуть створювати додатки для: настільних систем та численних фіксованих платформа а також мобільних пристроїв,що включають Android та iOS. У додатку можна знайти редактор коду, який легко об'єднується з іншими інструментами для: тестування, проектування, моніторингу, кодування та розгортання.

Процес навчання проходить більш гладко завдяки зрозумілому та простому інтерфейсу Qt Creator. Також серед функцій, що забезпечують розробку додатків швидко і легку, є: конструктор форм, інструменти профілювання та інтегрована розмітка графічного інтерфейсу, а ще візуальна налагодженість і створення інтерфейсу що призначений для користувача.

Переваги Qt Creator IDE: дизайн форм, також візуальна налагодженість, керування проектом, інструменти профілювання та інтегрований графічний інтерфейс, ще перетягування і дизайн інтерфейсу для користувачів, і останнє- редактор коду з підсвічуванням та автозаповненням синтаксису.

Висновки до розділу 2

Досліджено та проаналізовано аспекти розробки програмного забезпечення на різних стадіях. Розглянуто методології розробки ПЗ, їх походження, класифікацію, характеристики та приклади. Визначено вимоги для розробки ПЗ, а саме види вимог за рівнями, за характером та за джерелами. Показано, як правильно знаходити та документувати вимоги ПЗ. Також розглянуто процес та тонкості розробки ПЗ. Наостанок досліджено середовище програмування та більшість можливостей такого додатку.

3 РОЗРОБЛЕННЯ ПРОГРАМИ ДЛЯ СТВОРЕННЯ КОЛАЖІВ

3.1 Огляд мови програмування C++

Мова програмування C++ - це мова програмування високого рівня (рис. 3.1). Ця мова програмування ґрунтується на трьох китах - трьох парадигмах програмування, таких як: об'єктно орієнтованість, узагальненість та процедурність. Мова C++ була похідною мовою від мови програмування C, початкова назва мови C++ - "C з класами", та вже була перейменована на C++ у 1983. Творцем цієї мови програмування є Б'ярн Страуструп (рис.3.2) [12].



Рисунок 3.1 - Емблема мови програмування C++ [12]

Треба відзначити, що ця мова програмування розвивається достатньо швидко та є однією з найуживаніших мов програмування. Про це свідчать кількість стандартів, що постійно удосконалюються та випускаються. Наразі останнім стандартом є стандарт C++20.

Якщо говорити про функціонал C++, обов'язково слід відзначити момент наявності стандартної бібліотеки шаблонів (STL), та ще одної, досить потужної бібліотеки, що була створена фанатами мови програмування C++ - бібліотеки Boost, яка відіграє досить значну роль у житті програмістів .

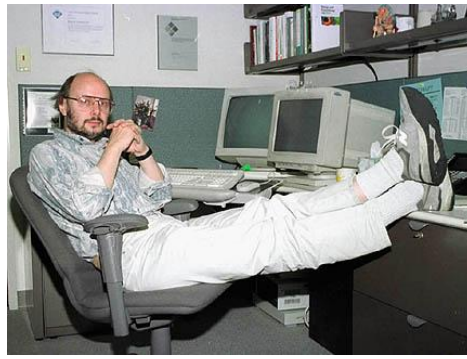


Рисунок 3.2 - Б'ярн Страуструп [12]

3.1.1 Основи програмування на C++

Для того, щоб програмувати на C++ потрібно мати деякі інструменти та знання. Щодо інструментів, для компіляції коду C++ потрібні: текстовий редактор, та відповідний компілятор, прикладами можуть бути Sublime Text (текстовий редактор для написання коду) та MinGW (компілятор коду) [12]. Це дві незалежні програми, але наразі є багато середовищ програмування (IDE), що об'єднують в собі не тільки функціонал вищевказаного ПЗ, а ще й більш зручні функції: відладник програм, автоматизатор написання коду, інспектор класів та об'єктів та інші. До таких середовищ відносяться: NetBeans, CodeBlocks, MVS, QT Creator, Eclipse, Xcode. В моєму випадку буде використовуватись IDE Qt Creator.



Рисунок 3.3 - Емблема IDE Qt Creator [11]

Щоб програмувати на мові C++ також потрібно розуміти основні поняття синтаксису та структури коду. Стейтмент, функція, вираз, бібліотека - поняття, що супроводжують кожного програміста в момент написання програми, тож розберемося в кожному слові [13]:

- Стейтмент - це так звана інструкція для обчислювальної машини. Стейтменти поділяють на: оголошення(декларація), присвоювання та виведення.
- Вираз - створений математичний об'єкт, для проведення обчислення та отримання результату.
- Функція - блок послідовних стейтментів. Треба відмітити, що кожна програма на мові програмування C++ починається саме з функції - головної функції `main()`, в яку починають записувати стейтменти.

3.1.2 Змінні в C++

Змінна - це найпростіший контейнер для зберігання даних. Щоб створити змінну потрібно для початку визначитись з типом даних, які будуть зберігатися [14]. Після типу даних повинна йти назва змінної та в кінці, як і для кожного стейтменту на C++, повинен йти символ “;”. Доречі, назви змінних не повинні повторюватись. Щоб змінна містила в собі деякі дані, її потрібно проініціалізувати, тобто надати цій змінній деяке значення. Це значення може бути різне, в залежності від типу змінної. Як ми вже говорили, у мові програмування є різні типи змінних, деякі з них: `bool`, `char`, `int`, `float`, `double`, `short`, `long`, `void`. Також, окрім змінної, можна створити константу, що характеризується словом “`const`” на початку стейтменту.

3.1.3 Арифметичні операції в C++

Арифметичні операції проводять над числами, а змінні або значення, що приймають участь в цих операціях, називають операндами [17]. В C++ є два типи арифметичних операцій: унарні та бінарні. Унарна операція проводиться над

одним операндом, а бінарна - над двома. C++ підтримує всі типи найпростіших математичних операцій: додавання, віднімання, ділення, множення, а також операція для отримання остачі від ділення, що виражається символом “%”. Слід згадати також унарні операції, такі як: префіксний/постфіксний інкремент/декремент (збільшення/зменшення значення на одиницю). Пріоритети дуже схожі на пріоритети звичайних арифметичних операцій (рис. 3.4).

Пріоритет	Оператор	Описание
1	::	Область видимости
2	++ --	Суффиксальный/постфиксный инкремент и декремент
	()	Вызов функции
	[]	Обращение к массиву по индексу
	.	Выбор элемента по ссылке
3	->	Выбор элемента по указателю
	++ --	Префиксный инкремент и декремент
	+ -	Унарный плюс и минус
	! ~	Логическое НЕ и побитовое НЕ
	(type)	Приведение к типу type
	*	Indirection (разыменование)
	&	Адрес
	sizeof	Размер
	new, new[]	Динамическое выделение памяти
	delete, delete[]	Динамическое освобождение памяти
4	.* ->*	Указатель на член
5	* / %	Умножение, деление и остаток
6	+ -	Сложение и вычитание
7	<< >>	Побитовый сдвиг влево и сдвиг вправо
8	< <=	For relational operators < and ≤ respectively
	> >=	For relational operators > and ≥ respectively
9	== !=	For relational = and ≠ respectively
10	&	Побитовое И
11	^	Побитовый XOR (исключающее или)
12		Побитовое ИЛИ (inclusive or)
13	&&	Логическое И
14		Логическое ИЛИ
15	?:	Тернарное условие
	=	Прямое присваивание (предоставляемое по умолчанию для C++ классов)
	+= -=	Присвоение с суммированием и разностью
	*= /= %=	Присвоение с умножением, делением и остатком от деления
	<<= >>=	Assignment by bitwise left shift and right shift
	&= ^= =	Assignment by bitwise AND, XOR, and OR
16	throw	Throw оператор (выброс исключений)
17	,	Запятая

Рисунок 3.4 - Пріоритети операцій в C++

3.1.4 Умовні вирази в C++

Умовні вирази поділяють на вирази відношення та логічні операції. До операцій відношення прилягають наступні операції [18]:

- “==” - операція “дорівнює”;
- “>” - операція “більше ніж”;
- “<” - операція “менше ніж”;
- “>=” - операція “більше або дорівнює”;
- “<=” - операція “менше або дорівнює”;

- “!=” - операція “не дорівнює”.

До логічних операцій відносять:

- “!” - операція “заперечення”.
- “&&” - операція “логічне множення”.
- “||” - операція “логічне додавання”.

Всі ці операції дають лише дві відповіді, “так” чи “ні”. Тому результат виконання цих операцій потрібно зберігати у типі даних bool.

3.1.5 Умовні конструкції в C++

Як і в будь-якій мові програмування, у C++ також присутні умовні конструкції такі як “if - else if - else”. Сутність даних конструкцій полягає в перевірці вказаного в умовах виразу на відповідність до “true/false”, коли перевірка виразу призводить до результату “true” - усі стейтменти, що знаходяться в тілі умовної конструкції починають виконуватись. Коли ж результат перевірки видає як результат значення “false” - починає виконуватись або перевірка наступної умови (блок “else if”), або тіло блоку “else” [13].

Ще одною умовною конструкцією в C++ є конструкція “switch-case-default”. Подібно до “if - else if - else”, в умові “switch” подається шукане значення, що перевіряється у випадках “case”. Коли умова “switch” співпадає з випадком “case”, починається виконання прописаних стейтментів для конкретного “case”. Якщо ж ні один “case” не співпадає з заданою умовою “switch” - починає виконуватись випадок “default”, якщо такий випадок прописаний. Потрібно вказати, що для дострокового виходу з даної умовної конструкції (якщо це потрібно) використовують ключове слово “break”.

3.1.6 Цикли в C++

Існує три типи циклів у C++: “while”, “for”, “do-while”

Принцип циклів, як і принцип умовних операторів, полягає у перевірці даного виразу [19]. Цикл буде виконуватись, а саме всі стейтменти, що вказані у тілі циклу будуть виконуватись, до тих пір, поки результат перевірки даного виразу не дасть негативний результат “false”. Потрібно відзначити, що на відміну від циклів “while”, “for”, які починають виконуватись тоді і тільки тоді, коли результат перевірки буде “true”, цикл “do-while” завжди починає своє перше виконання.

3.1.7 Масиви в C++

Якщо говорити про масиви даних, можна сказати, що це один великий контейнер для однотипних змінних, пам'ять для яких є статичною, знаходиться поруч та виділяється під час запуску програми. Доступ до даних надається по індексу. Для створення масиву потрібно вказати тип даних, які будуть зберігатися в масиві, назву масиву, у квадратних дужках кількість елементів. Якщо масив не проініціалізований у момент створення - ініціалізація проходить методом перебору всього масиву через цикл. Відзначаю, що індексація елементів починається з 0 [20].

3.1.8 Декілька слів про функції, вказівники та класи в C++

Функція представляє собою набір стейтментів, що проводять операції над даними, що передаються у цю функцію. Також функція може повертати результат проведених операцій, але тільки у заздалегідь вказаному типі даних. Простими словами, функція - окремий модуль, що може бути виконаний неодноразово [21].

Вказівники - це змінні, що зберігають в собі адресу пам'яті, в яку може бути збережено значення. Щоб створити вказівник, потрібно перед назвою

змінної прописати знак “*”. Для запису у комірку пам’яті вказівника даних, потрібно також перед операцією присвоєння вказати символ розйменування “*”.

Класи являють собою тип даних, що створений користувачем. Цей тип даних може містити в собі простіші типи даних, які є стандартними для мови програмування C++, адже все майже все можливо описати числами або символами. Для класів характерні наступні властивості:

- Конструктор, що має автоматично ініціалізувати користувацький тип даних;
- Деструктор, що має звільняти динамічно виділену пам’ять(у разі необхідності);
- Функції, що можуть проводити операції з внутрішніми даними класу;
- Специфікатори інкапсуляції, що визначають відкритість доступу до функцій класу, чи його елементів.
- Перевантаження операторів, щоб виконувати операції над класами, як операндами.

3.2 Опис створення та огляд програмного забезпечення

В процесі розроблення програми першим чином потрібно було визначитись з головним меню. У середовищі Qt було достатньо легко створити форму вікна та додати відповідні кнопки для реалізації функціоналу, так як це середовище вже надає готові інструменти для формування повноцінного програмного забезпечення (рис. 3.5).

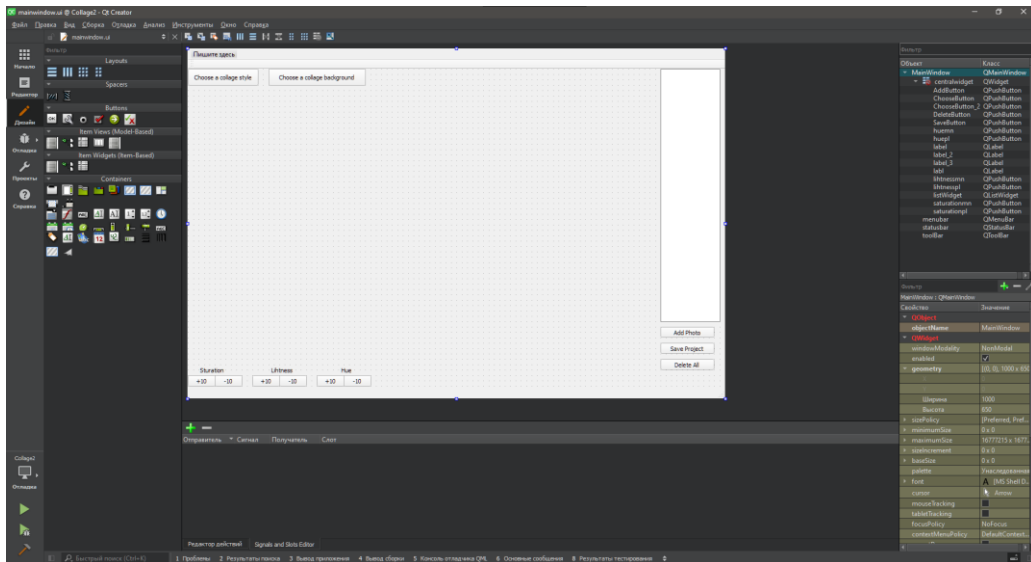


Рисунок 3.5 - Вікно розроблення графічного інтерфейсу

Як можна побачити, головне меню складається з вікна “QMainWindow”, поля для списку “QListWidget”, текстових полів “QLabel” та кнопок “QPushButton”. Також у головному вікні за замовчуванням створюється “QMenuBar”, “QStatusBar”, “QToolBar”, але задля простоти у використанні ці віджети не використовувались повноцінно.

Створивши таку форму, розмістивши відповідні віджети ми можемо скомпілювати програму та відразу отримати реальне вікно, але без будь-якої реакції на дії користувача (рис. 3.6).

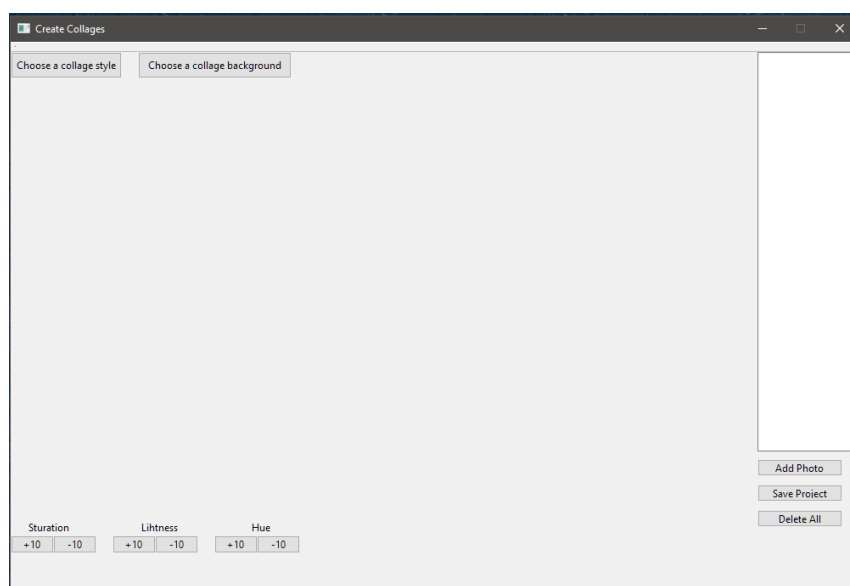


Рисунок 3.6 - Вікно програми на виході

Тепер потрібно реалізувати функціонал кожної кнопки в готовому вікні.

Для цього у середовищі Qt існує метод сигналів та слотів (рис. 3.7):

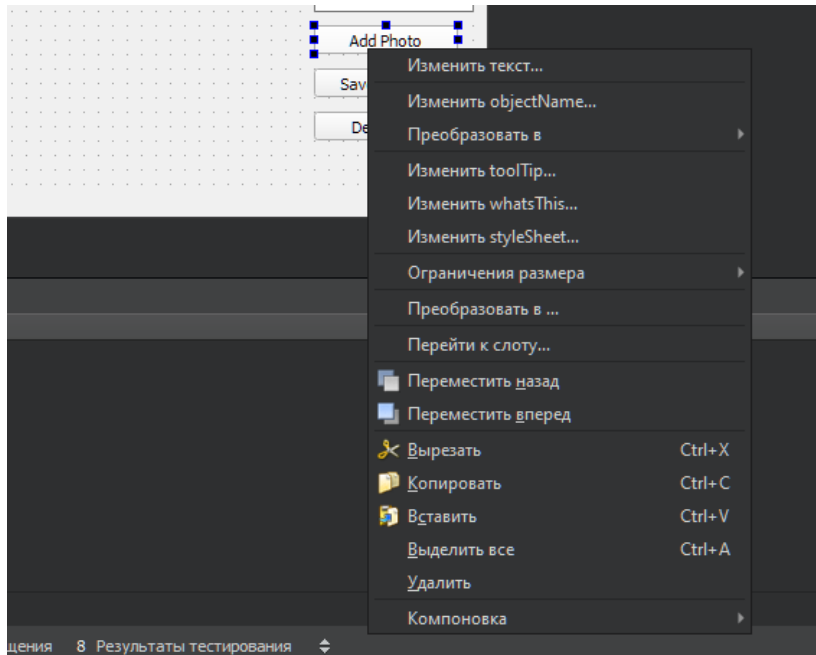


Рисунок 3.7 - Перехід до методу сигналів та слотів

Нам потрібно створити сигнал кліку на кнопку, тому саме цей сигнал вибираємо як дію користувача (рис. 3.8). Далі відкривається текстовий редактор, у якому вже безпосередньо програмуємо, що повинно статися, коли користувач тисне кнопку “Add Photo” (рис. 3.9).

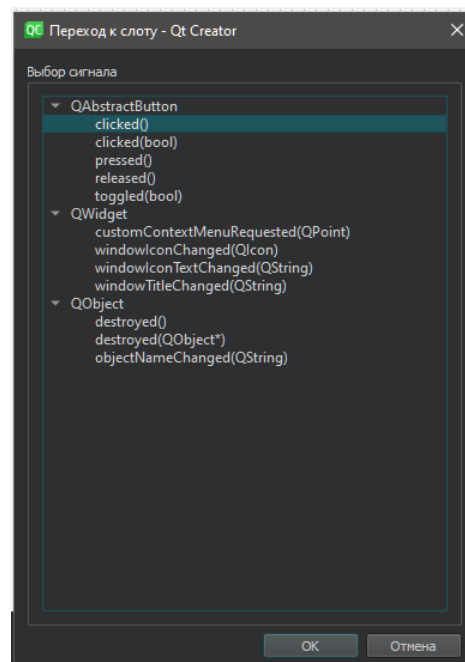


Рисунок 3.8 - Перехід до сигналу

```

void MainWindow::on_AddButton_clicked()
{
    if(counter != 9)
    {
        QIcon pht;
        QString fileName = QFileDialog::getOpenFileName();
        pht.addFile(fileName);
        photos[counter].load(fileName);
        photolist[counter] = new QListWidgetItem (pht,fileName);
        ui->listWidget->setIconSize(QSize(100,50));
        ui->listWidget->addItem(photolist[counter]);
        photolist[counter]->setSizeHint(QSize(100,50));
        counter++;
    }else
    {
        QMessageBox warn;
        warn.setText("To much photos!");
        warn.exec();
    }
}

```

Рисунок 3.9 - Програмування реакції на натискання кнопки “Add Photo”

Написаний код складається з вибору файлу фотографії з директорії комп’ютера (рис. 3.10), додавання його до масиву фотографій, створення нового об’єкту для поля списку, та встановлення іконки та шляхом розміщення файлу (для відображення і опису у списку) (рис. 3.11).

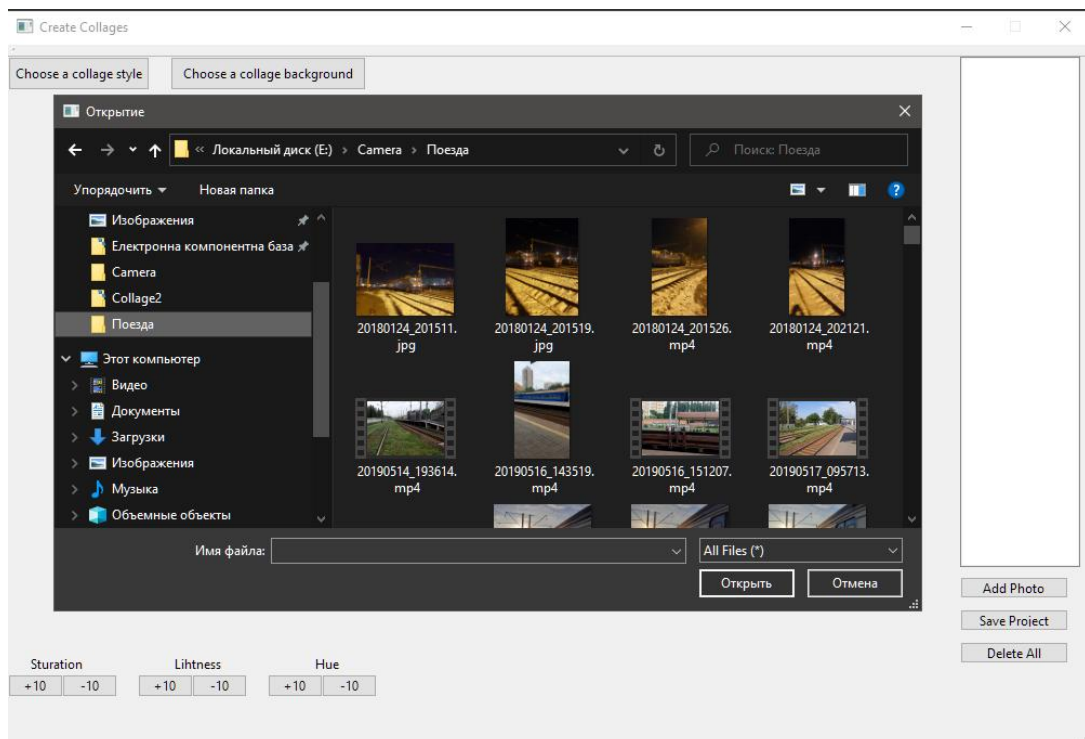


Рисунок 3.10 - Вибір файлу з директорії комп’ютера



Рисунок 3.11 - Додавання файлу до поля списку з іконкою та шляхом

Для повноцінного відображення фотографії у робочому полі потрібно було реалізувати метод сигналів та слотів для натискання на компонент поля списку (рис. 3.12).

```
void MainWindow::on_listWidget_itemClicked()
{
    ui->labl->clear();
    currentImg = ui->listWidget->currentRow();
    ui->labl->setPixmap(QPixmap::fromImage(photos[currentImg]));
    double hI = photos[currentImg].height();
    double hL = ui->labl->height();
    double f = hL/hI;
    ui->labl->setFixedWidth(f * ui->labl->pixmap().width());
    ui->labl->setFixedHeight(f * ui->labl->pixmap().height());
    ui->labl->show();
}
```

Рисунок 3.12 - Програмування реакції на натискання компоненту поля списку

Так як одним із методів відображення фотографії є розміщення його у віджеті “QLabel”, саме ця процедура і виконується в приведеній частині коду, фотографія завантажується саме з масиву даних, куди зберігалась. Також

проходить процедура підгонки “QLabel” під зображення для показу без втрати пропорцій (рис. 3.13).

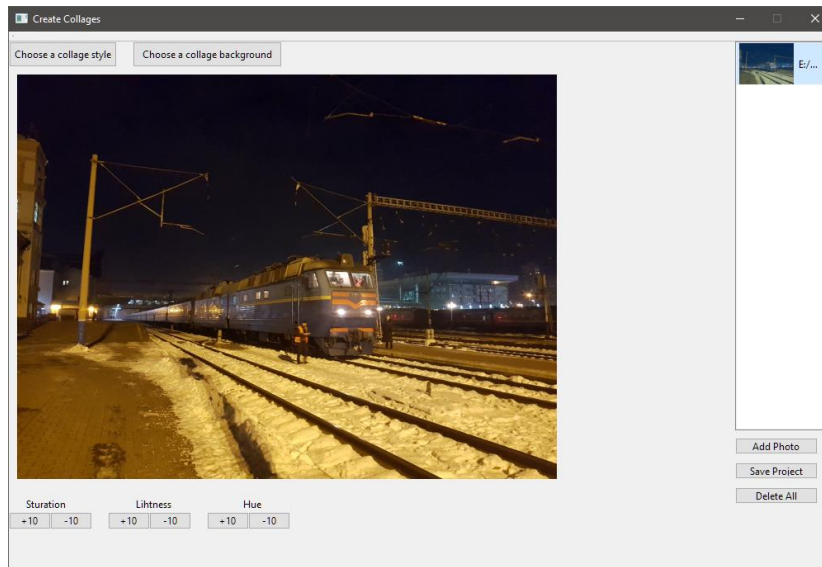


Рисунок 3.13 - Реакція програми на натискання по елементу списку

Для редагування фотографії реалізовано три функції: зміни насиченості, зміни яскравості, зміни відтінку (у дві сторони кожна). Дані функції дуже схожі між собою по алгоритму. Різниця лише в тому, яку властивість з властивостей HSL ми опрацьовуємо та редагуємо. Для того, щоб змінити будь-яку властивість зображення потрібно розділити зображення по пікселям та відредагувати цю властивість саме на кожному пікселі (рис. 3.14).

```
void MainWindow::on_saturationpl_clicked()
{
    ui->labl->clear();
    ui->labl->show();
    int curr = ui->listWidget->currentRow();
    QImage img = photos[curr];
    for(int x = 0; x < img.width(); x++)
    {
        for(int y = 0; y < img.height(); y++)
        {
            QColor clr = img.pixelColor( x, y);
            qreal ns = clr.hslSaturationF() + 0.05;
            if(ns > 1) ns = 1;
            clr.setHslF(clr.hslHueF(), ns, clr.lightnessF());
            img.setPixelColor(x,y,clr);
        }
    }
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
    photos[curr] = img;
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
    ui->labl->show();
}
```

Рисунок 3.14 - Програмування реакції на збільшення насиченості

В програму вбудовані дві основні функції для створення колажу: вибір фону (рис 3.16) а також вибір розміщення фотографій (рис 3.15). За це відповідають кнопки “Choose collage style”, “Choose collage background”. Дані кнопки створені таким чином, щоб викликати додаткове вікно, де можна підібрати бажані характеристики майбутнього колажу. Іконки вибору реалізовані за допомогою кнопок з накладеними на них фотографіями варіантів вибору. Варіанти фону колажу завантажені та збережені до ресурсів проекту. При виборі того чи іншого варіанту розташування чи варіанту фону, спрацьовує метод сигналу слоту, який записує відповідне число у одиницю пам'яті комп'ютера для обробки при зберіганні проекту.

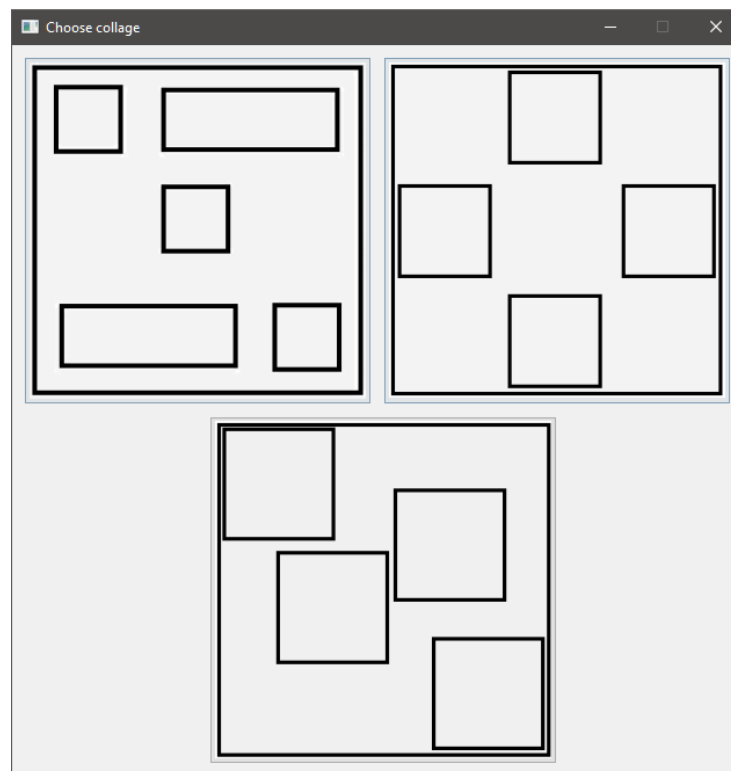


Рисунок 3.15 - Вікно вибору розташування фотографій в колажі

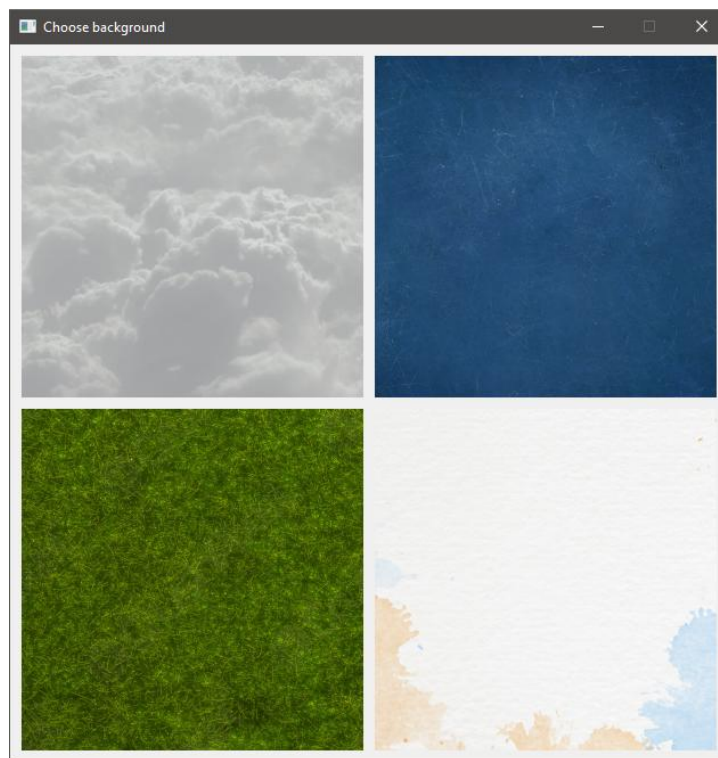


Рисунок 3.16 - Вікно вибору фону

Були створені допоміжні реакції на натискання кнопки “Delete all” та на подвійне натискання на об’єкт поля списку для видалення всіх та довільних фотографій відповідно.

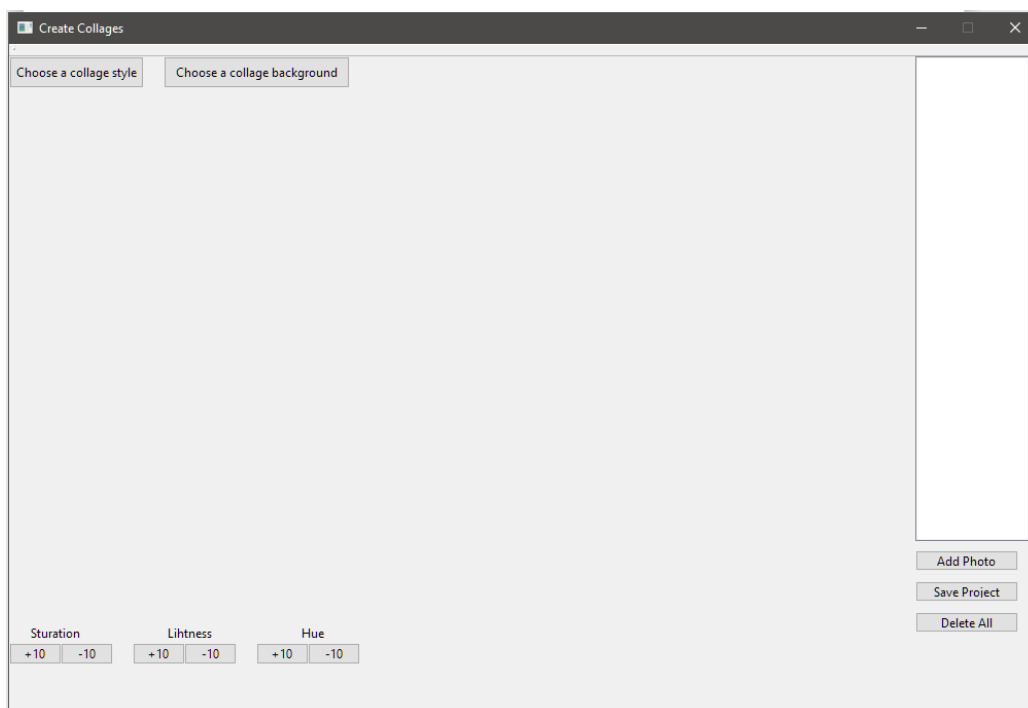


Рисунок 3.17 - Спрацювання подвійного натискання або кнопки “Delete all”

При натисканні кнопки “Delete all” спрацьовує функція, що видаляє всі елементи зі списку, та заповнює масив фотографій звичайним білим фоном для подальшого перезапису (рис. 3.18). При подвійному натисканні на елемент списку видаляється поточний елемент з поля списку і також заповнюється відповідна комірка фотографії звичайним білим фоном (рис. 3.19).

```
void MainWindow::on_DeleteButton_clicked()
{
    ui->labl->clear();
    ui->labl->show();
    if(counter > 0)
    {
        ui->listWidget->clear();
        for(int i = 0; i < counter; i++)
            photos[i].load(":/white.jpg");
        counter = 0;
    }else
    {
        QMessageBox warn;
        warn.setText("Nothing to delete!");
        warn.exec();
    }
}
```

Рисунок 3.18 - Програмування реакції натискання кнопки “Delete all”

```
void MainWindow::on_listWidget_itemDoubleClicked()
{
    int curr = ui->listWidget->currentRow();
    delete ui->listWidget->takeItem(curr);
    photos[curr].load(":/white.jpg");
    ui->labl->clear();
    ui->labl->show();
    counter -= 1;
}
```

Рисунок 3.19 - Програмування реакції подвійного натискання на поле списку

При натисканні на кнопку “Save Project” проходить аналіз вхідних даних розташування фотографій в колажі та вибраного фону, відповідно до вибору користувача. Також розраховується масштаб фотографій, що будуть накладені поверх фону та зберігається проект (рис. 3.20).

```

void MainWindow::on_SaveButton_clicked()
{
    if(counter < 3)
    {
        QMessageBox warn;
        warn.setText("Nothing to save!");
        warn.exec();
    }
    else
    {
        QImage collage;
        if (*backgr == 0)
            collage = photos[0];
        else if (*backgr == 1)
            collage.load(":/sky.jpg");
        else if (*backgr == 2)
            collage.load(":/bricks.jpg");
        else if (*backgr == 3)
            collage.load(":/Grass.jpg");
        else if (*backgr == 4)
            collage.load(":/water.jpg");
        QPixmap var;
        QPainter overPhoto(&collage);
        if(*form == 2)
        {
            int pixWidth = collage.width();
            int pixWidthT = pixWidth/3;
            int pixHeight = collage.height();
            int pixHeightT = pixHeight/3;
            var = QPixmap::fromImage(photos[1]);
            overPhoto.drawPixmap(pixWidthT, 0, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[2]);
            overPhoto.drawPixmap(0, pixHeightT, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[3]);
            overPhoto.drawPixmap(pixWidthT*2, pixHeightT, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[4]);
            overPhoto.drawPixmap(pixWidthT, pixHeightT*2, pixWidthT, pixHeightT, var);
            overPhoto.end();
            collage.save("newCollage.jpg");
        }
        else if(*form == 1)
        {
            int pixWidth = collage.width();
            int pixWidthT = pixWidth/3;
            int pixHeight = collage.height();
            int pixHeightT = pixHeight/3;
            var = QPixmap::fromImage(photos[1]);
            overPhoto.drawPixmap(0, 0, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[2]);
            overPhoto.drawPixmap(pixWidthT*2, 0, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[3]);
            overPhoto.drawPixmap(pixWidthT, pixHeightT, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[4]);
            overPhoto.drawPixmap(0, pixHeightT*2, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[5]);
            overPhoto.drawPixmap(pixWidthT*2, pixHeightT*2, pixWidthT, pixHeightT, var);
            overPhoto.end();
            collage.save("newCollage.jpg");
        }
        else if(*form == 3)
        {
            int pixWidth = collage.width();
            int pixWidthT = pixWidth/3;
            int pixHeight = collage.height();
            int pixHeightT = pixHeight/3;
            var = QPixmap::fromImage(photos[1]);
            overPhoto.drawPixmap(0, 0, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[2]);
            overPhoto.drawPixmap(pixWidthT, pixHeightT, pixWidthT, pixHeightT, var);
            var = QPixmap::fromImage(photos[3]);
            overPhoto.drawPixmap(pixWidthT*2, pixHeightT*2, pixWidthT, pixHeightT, var);
            overPhoto.end();
            collage.save("newCollage.jpg");
        }
    }
}

```

Рисунок 3.20 - Програмування реакції на натискання кнопки “Save Project”

В результаті роботи програми можна отримати чудовий результат (рис. 3.21).

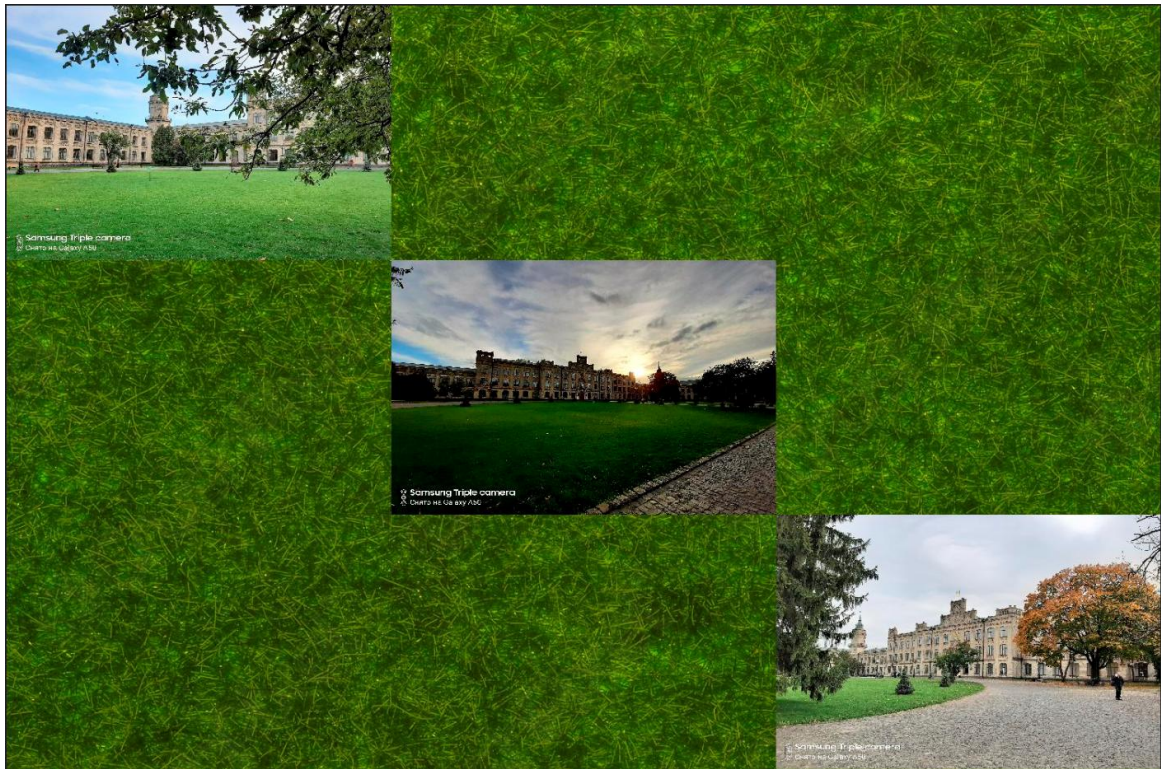


Рисунок 3.21 - Результати роботи програми

Висновки до розділу 3

Перш за все, розібрано основні моменти, що стосуються програмування на мові C++. До таких моментів відносяться поняття змінної, масиву, арифметичних та логічних операцій, циклів, умовних конструкцій, функцій, вказівників та класів. У цьому розділі розглянуто процес розробки програмного забезпечення для створення колажів та його функціонал. Також було підтверджено працездатність написаного коду. У програмі розроблено наступні функції: функція додавання фотографії, два методи видалення (видалення всіх доданих фото та видалення окремих фото), можливість корегування трьох параметрів зображення (насиченість, яскравість, відтінок), функції вибору типу колажу (розташування фото на основному, вихідному зображенні) та вибору фону, на який будуть накладатися додані зображення. Приведені рисунки робочої програми та рисунки частин коду з поясненнями до них.

ВИСНОВКИ

Опрацьовано питання особливостей та сучасних методів створення колажу. Розглянуто історію виникнення мистецтва колажу. Досліджено особливості колажів та галузі їх застосувань. Виконано опис техніки композиції. Простежено характеристики створення колажу із фотографій та сам процес і основи створення колажу.

Проаналізовано методологію розробки програмного забезпечення. Розглянуто найбільш поширені та відомі парадигми програмування. Досліджено три види формування методологій. Виконано класифікації методології за ядрами та за специфікою реалізацій. Виділено приклади методологій такі як: Open Source, екстремальне програмування, SCRUM.

Опрацьовано набір вимог щодо функцій, властивостей та якостей програмного забезпечення, що перебуває у розробці а саме види вимог за рівнями, за характером та за джерелами. Розглянуто процес та тонкості розробки ПЗ. Досліджено середовище програмування та більшість можливостей такого додатку.

Опрацьовано питання розроблення програми для створення колажів. Для цього виконано огляд мови програмування C++, що є однією з найуживаніших мов програмування. Обрано середовище програмування IDE Qt Creator.

Розглянуто основні поняття синтаксису та структури коду для C++, а саме стейтмент, функція, вираз, бібліотека. Виконано аналіз змінних, арифметичних операцій, умовних виразів, циклів, а також масивів для мови програмування C++.

Відповідно гарній теоретичній підготовці виконано розроблення програмного забезпечення з відповідним функціоналом за допомогою Qt та C++. Проведено тестування та опис створення програмного забезпечення.

Виконано опис основних функцій написаної програми. До них відноситься функція додавання фотографії, два методи видалення, можливість корегування трьох параметрів зображення, функції вибору типу колажу та вибору фону, на який будуть накладатися додані зображення.

Рекомендовано до використання розроблене програмне забезпечення з відповідним функціоналом для створення колажів у особистих та комерційних цілях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Колаж - техніка сучасного мистецтва. URL: <https://veryimportantlot.com/ru/news/blog/kollazh> (дата звернення 05.05.2021).
2. Історія дослідження колажу. URL: <http://kmbu.com.ua/?p=1449> (дата звернення 06.05.2021).
3. Основні відомості про колаж. URL: <https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%B0%D0%B6> (дата звернення 08.05.2021).
4. Композиція фотографії. URL: <https://photoschool.kiev.ua/fotoklub/blog/1369-photography-composition-three-rules> (дата звернення 08.05.2021).
5. Композиція в колажі. URL: https://studbooks.net/580480/kulturologiya/kompozitsiya_tsvet_kollazhe (дата звернення 09.05.2021).
6. Методологія розробки ПЗ. URL: <https://uk.wikipedia.org/wiki/> (дата звернення 16.05.2021).
7. Процес розробки ПЗ. URL: <https://uk.wikipedia.org/wiki/%D0%9F> (дата звернення 16.05.2021).
8. Вимоги до програмного забезпечення. URL: <https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%> (дата звернення 18.05.2021).
9. Огляд середовища програмування. URL: <https://linuxvsem.ru/programs/qt-creator> (дата звернення 18.05.2021).
10. Композиція в колажі. URL: <http://www.xserver.ru/computer/graphic/obsh/66/> (дата звернення 09.05.2021).
11. QT: інтерфейс, и не только... URL: <http://citforum.ck.ua/programming/qt/> (дата звернення 25.05.2021).
12. Уроки программирования на языке C++. URL: <https://ravesli.com/uroki-cpp/> (дата звернення 24.05.2021).

13. Metanit C++. URL: <https://metanit.com/cpp/tutorial/2.7.php> (дата звернения 26.05.2021).
14. C++ | Переменные. URL: <https://metanit.com/cpp/tutorial/2.2.php> (дата звернения 26.05.2021).
15. C++ | Типы данных. URL: <https://metanit.com/cpp/tutorial/2.3.php> (дата звернения 26.05.2021).
16. C++ | Константы. URL: <https://metanit.com/cpp/tutorial/2.5.php> (дата звернения 26.05.2021).
17. C++ | Арифметические операции. URL: <https://metanit.com/cpp/tutorial/2.6.php> (дата звернения 27.05.2021).
18. C++ | Условные выражения. URL: <https://metanit.com/cpp/tutorial/2.7.php> (дата звернения 27.05.2021).
19. C++ | Циклы. URL: <https://metanit.com/cpp/tutorial/2.13.php> (дата звернения 27.05.2021).
20. C++ | Массивы. URL: <https://metanit.com/cpp/tutorial/2.15.php> (дата звернения 27.05.2021).
21. C++ | Определение и объявление функций. URL: <https://metanit.com/cpp/tutorial/3.1.php> (дата звернения 28.05.2021).

Додаток А.

SUMMARY

SUMMARY

Many marketers find it difficult to quickly choose an image to promote a product or service without the help of a designer. A good slogan and a good description are often not enough to lead the customer to the end point - the purchase. That's why you need a bright and special visual.

One of the effective and affordable design techniques is to create collages from photos. With their help it is possible to solve many tasks at the same time, namely: to attract attention at the expense of unusual design; show a product or service from several sides.

Photo collages are also used by many people for personal purposes. After all, a quality photo collage can be given to family or friends, it will be a unique gift for anyone and will decorate the interior of the apartment or office.

In this bachelor's work 3 sections are considered:

- history, features and modern methods of collage creation.
- Methods, requirements and features of software creation. Overview of the design environment.
- Program development for collaging.

In the first chapter it was considered that this visual genre, as a collage, has its own, long and interesting story, which continues to take place before our eyes. It is mentioned that earlier, when creating a collage, several materials of completely different factors and origins were used at the same time. Also with an emphasis on the areas of application of collages, such as: photography, cinema, music, literature. The first wave of collage popularity in art vanished by the late 1930s. But after 20 years, interest in this technique reappeared in the cultural environment. Since then, collage has taken a worthy place in modern culture. The works of Russian artists of the XIX-XX centuries were also illustrated. The peculiarities of the collage technique and its styles are considered in detail, among which: Vegetative, decorative, landscape, form-linear. The characteristics of creating a collage of photos and the process and basics of creating a collage are traced.

The second section explores and analyzes aspects of software development at different stages. The methodologies of software development, their origin, classification, characteristics and examples were considered. The requirements for software development have been identified, namely the types of requirements by levels, by nature and by sources. Learn how to properly find and document software requirements. The process and subtleties of software development were also considered. Finally, the programming environment and most of the capabilities of such an application were explored.

In the third section, first of all, the main points concerning programming in C ++ are analyzed. Such points include the concepts of variable, array, arithmetic and logical operations, cycles, conditional constructions, functions, pointers and classes.

This section discusses the process of developing software for creating collages and its functionality. The workability of the written code was also confirmed. The program has the following functions: the function of adding a photo, two methods of deletion (deleting all attached photos and deleting individual photos), the ability to adjust three image parameters (saturation, brightness, hue), collage type selection functions (photo location on the main, original image) and select the background on which the added images are superimposed. Figures of the working program and drawings of parts of the code with explanations to them are resulted.

Додаток Б.

ЛІСТИНГИ

ЛІСТИНГИ

Файл “form.h”

```
#ifndef FORM_H
#define FORM_H

#include <QWidget>

namespace Ui {
class Form;
}

class Form : public QWidget
{
    Q_OBJECT

public:
    explicit Form(QWidget *parent = nullptr);
    ~Form();
    int *result;

private slots:
    void on_collage1_clicked();

    void on_collage2_clicked();

    void on_collage3_clicked();

private:
    Ui::Form *ui;
};

#endif // FORM_H
```

Файл “form2.h”

```
#ifndef FORM2_H
#define FORM2_H

#include <QWidget>

namespace Ui {
```

```

class Form2;
}

class Form2 : public QWidget
{
    Q_OBJECT

public:
    explicit Form2(QWidget *parent = nullptr);
    ~Form2();
    int *result;

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_4_clicked();

private:
    Ui::Form2 *ui;
};

#endif // FORM2_H

```

Файл “mainwindow.h”

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QListWidgetItem>
#include <QListWidget>
#include <QIcon>
#include <QFileDialog>
#include <QLabel>
#include <QPushButton>
#include <QMessageBox>
#include <QPainter>
#include <QStatusBar>
#include <QWidget>

```

```
#include <QAbstractButton>
#include <QBitmap>
#include <QColor>
#include <QImage>
#include <QtGlobal>
#include "form.h"
#include "ui_form.h"
#include "form2.h"
#include "ui_form2.h"
#include "Global.h"
```

```
QT_BEGIN_NAMESPACE
```

```
namespace Ui { class MainWindow; }
```

```
QT_END_NAMESPACE
```

```
class MainWindow : public QMainWindow
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    MainWindow(QWidget *parent = nullptr);
```

```
    ~MainWindow();
```

```
    int *backgr;
```

```
    int *form;
```

```
private slots:
```

```
    void on_AddButton_clicked();
```

```
    void on_listWidget_itemClicked();
```

```
    void on_SaveButton_clicked();
```

```
    void on_DeleteButton_clicked();
```

```
    void on_listWidget_itemDoubleClicked();
```

```
    void on_listWidget_entered();
```

```
    void on_ChooseButton_clicked();
```

```

void on_saturationpl_clicked();

void on_saturationmn_clicked();


void on_lihtnesspl_clicked();

void on_lihtnessmn_clicked();


void on_huepl_clicked();

void on_huemn_clicked();


void on_ChoseButton_2_clicked();


private:
    Ui::MainWindow *ui;
    int counter;
    int currentImg;
    int *result;
    QLabel *lbl;
    QImage photos[9];
    QListWidgetItem *photolist[9];

};

#endif // MAINWINDOW_H

```

Файл “form.cpp”

```

#include "form.h"
#include "ui_form.h"
#include "mainwindow.h"
// #include "Global.h"

Form::Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Form)
{
    ui->setupUi(this);
    result = new int(0);
}

Form::~Form()

```

```

{
    delete ui;
}

void Form::on_collage1_clicked()
{
    *result = 1;
    close();
}

void Form::on_collage2_clicked()
{
    *result = 2;
    close();
}

void Form::on_collage3_clicked()
{
    *result = 3;
    close();
}

```

Файл “form2.h”

```

#include "form2.h"
#include "ui_form2.h"
#include "mainwindow.h"

Form2::Form2(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Form2)
{
    ui->setupUi(this);
    result = new int(0);
}

Form2::~Form2()
{

```

```

        delete ui;
    }

void Form2::on_pushButton_clicked()
{
    *result = 1;
    close();
}

void Form2::on_pushButton_2_clicked()
{
    *result = 2;
    close();
}

void Form2::on_pushButton_3_clicked()
{
    *result = 3;
    close();
}

void Form2::on_pushButton_4_clicked()
{
    *result = 4;
    close();
}

```

Файл “main.cpp”

```

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setFixedSize(1000,650);
    w.setWindowTitle("Create Collages");
}

```

```

        w.show();
        return a.exec();
    }

```

Файл "mainwindow.cpp"

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

```

```

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)

```

```

{
    ui->setupUi(this);
    this->setMouseTracking(true);
    counter = 0;
    currentImg = 0;
    lbl = new QLabel(this);
    backgr = new int(0);
    form = new int(0);
}

```

```

MainWindow::~MainWindow()

```

```

{
    delete ui;
}

```

```

void MainWindow::on_AddButton_clicked()

```

```

{
    if(counter != 9)
    {
        QIcon pht;
        QString fileName = QFileDialog::getOpenFileName();
        pht.addFile(fileName);
        photos[counter].load(fileName);
        photolist[counter] = new QListWidgetItem(pht, fileName);
        ui->listWidget->setIconSize(QSize(100, 50));
        ui->listWidget->addItem(photolist[counter]);
        photolist[counter]->setSizeHint(QSize(100, 50));
        counter++;
    } else

```

```

    {
        QMessageBox warn;
        warn.setText("To much photos!");
        warn.exec();
    }
}

void MainWindow::on_listWidget_itemClicked()
{
    ui->labl->clear();
    currentImg = ui->listWidget->currentRow();
    ui->labl->setPixmap(QPixmap::fromImage(photos[currentImg]));
    double hI = photos[currentImg].height();
    double hL = ui->labl->height();
    double f = hL/hI;
    ui->labl->setFixedWidth(f * ui->labl->pixmap().width());
    ui->labl->setFixedHeight(f * ui->labl->pixmap().height());
    ui->labl->show();
}

void MainWindow::on_SaveButton_clicked()
{
    if(counter < 3)
    {
        QMessageBox warn;
        warn.setText("Nothing to save!");
        warn.exec();
    }
    else
    {
        qDebug() << *backgr;
        QImage collage;
        if (*backgr == 0)
            collage = photos[0];
        else if (*backgr == 1)
            collage.load(":/sky.jpg");
        else if (*backgr == 2)
            collage.load(":/bricks.jpg");
        else if (*backgr == 3)
            collage.load(":/Grass.jpg");
        else if (*backgr == 4)

```

```

        collage.load(":/water.jpg");
QPixmap var;
QPainter overPhoto(&collage);
if(*form == 2)
{
    int pixWidth = collage.width();
    int pixWidthT = pixWidth/3;
    int pixHeight = collage.height();
    int pixHeightT = pixHeight/3;
    var = QPixmap::fromImage(photos[1]);
    overPhoto.drawPixmap(pixWidthT, 0, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[2]);
    overPhoto.drawPixmap(0, pixHeightT, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[3]);
    overPhoto.drawPixmap(pixWidthT*2, pixHeightT, pixWidthT, pixHeightT,
var);

    var = QPixmap::fromImage(photos[4]);
    overPhoto.drawPixmap(pixWidthT, pixHeightT*2, pixWidthT, pixHeightT,
var);

    overPhoto.end();
    collage.save("    newCollage.jpg");
}
else if(*form == 1)
{
    int pixWidth = collage.width();
    int pixWidthT = pixWidth/3;
    int pixHeight = collage.height();
    int pixHeightT = pixHeight/3;
    var = QPixmap::fromImage(photos[1]);
    overPhoto.drawPixmap(0, 0, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[2]);
    overPhoto.drawPixmap(pixWidthT*2, 0, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[3]);
    overPhoto.drawPixmap(pixWidthT, pixHeightT, pixWidthT,
pixHeightT,var);

    var = QPixmap::fromImage(photos[4]);
    overPhoto.drawPixmap(0, pixHeightT*2, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[5]);
    overPhoto.drawPixmap(pixWidthT*2, pixHeightT*2, pixWidthT, pixHeightT,
var);

    overPhoto.end();
    collage.save("newCollage.jpg");
}

```

```

else if(*form == 3)
{
    int pixWidth = collage.width();
    int pixWidthT = pixWidth/3;
    int pixHeight = collage.height();
    int pixHeightT = pixHeight/3;
    var = QPixmap::fromImage(photos[1]);
    overPhoto.drawPixmap(0, 0, pixWidthT, pixHeightT, var);
    var = QPixmap::fromImage(photos[2]);
    overPhoto.drawPixmap(pixWidthT, pixHeightT, pixWidthT,
pixHeightT,var);
    var = QPixmap::fromImage(photos[3]);
    overPhoto.drawPixmap(pixWidthT*2, pixHeightT*2, pixWidthT, pixHeightT,
var);
    overPhoto.end();
    collage.save("newCollage.jpg");
}
}
}

```

```

void MainWindow::on_DeleteButton_clicked()

```

```

{
    ui->labl->clear();
    ui->labl->show();
    if(counter > 0)
    {
        ui->listWidget->clear();
        for(int i = 0;i < counter; i++)
            photos[i].load(":/white.jpg");
        counter = 0;
    }else
    {
        QMessageBox warn;
        warn.setText("Nothing to delete!");
        warn.exec();
    }
}
}

```

```

void MainWindow::on_listWidget_itemDoubleClicked()

```

```

{
    int curr = ui->listWidget->currentRow();
    delete ui->listWidget->takeItem(curr);
    ui->labl->clear();
}

```

```

        ui->labl->show();
        counter -= 1;
    }

void MainWindow::on_listWidget_entered()
{
    QString helpmessage = "Click to display photo or double click to delete";
    ui->statusbar->showMessage(helpmessage, 2000);
}

void MainWindow::on_ChoseButton_clicked()
{
    Form *f = new Form();
    f->setFixedSize(630,630);
    f->setWindowTitle("Choose collage");
    f->show();
    form = f->result;
}

void MainWindow::on_saturationpl_clicked()
{
    ui->labl->clear();
    ui->labl->show();
    int curr = ui->listWidget->currentRow();
    QImage img = photos[curr];
    for(int x = 0; x < img.width(); x++)
    {
        for(int y = 0; y < img.height(); y++)
        {
            QColor clr = img.pixelColor( x, y);
            qreal ns = clr.hslSaturationF() + 0.05;
            if(ns > 1) ns = 1;
            clr.setHslF(clr.hslHueF(), ns, clr.lightnessF());
            img.setPixelColor(x,y,clr);
        }
    }
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
    photos[curr] = img;
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
    ui->labl->show();
}

void MainWindow::on_saturationmn_clicked()

```

```

{
    ui->labl->clear();
    ui->labl->show();
    int curr = ui->listWidget->currentRow();
    QImage img = photos[curr];
    for(int x = 0; x < img.width(); x++)
    {
        for(int y = 0; y < img.height(); y++)
        {
            QColor clr = img.pixelColor( x, y);
            qreal ns = clr.hslSaturationF() - 0.05;
            if(ns < 0) ns = 0;
            clr.setHslF(clr.hslHueF(), ns, clr.lightnessF());
            img.setPixelColor(x,y,clr);
        }
    }
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
    photos[curr] = img;
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
    ui->labl->show();
}

void MainWindow::on_lihtnesspl_clicked()
{
    ui->labl->clear();
    ui->labl->show();
    int curr = ui->listWidget->currentRow();
    QImage img = photos[curr];
    for (auto x = 0; x < img.width(); x++)
    {
        for (auto y = 0; y < img.height(); y++)
        {
            QColor clr = img.pixelColor(x, y);
            qreal ns = clr.lightnessF() + 0.05;
            if(ns > 1) ns = 1;
            clr.setHslF(clr.hslHueF(), clr.hslSaturationF(), ns);
            img.setPixelColor(x, y, clr);
        }
    }
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
    photos[curr] = img;
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
    ui->labl->show();
}

```

```
}
```

```
void MainWindow::on_lihtnessmn_clicked()
```

```
{
```

```
    ui->labl->clear();
```

```
    ui->labl->show();
```

```
    int curr = ui->listWidget->currentRow();
```

```
    QImage img = photos[curr];
```

```
    for (auto x = 0; x < img.width(); x++)
```

```
    {
```

```
        for (auto y = 0; y < img.height(); y++)
```

```
        {
```

```
            QColor clr = img.pixelColor(x, y);
```

```
            qreal ns = clr.lightnessF() - 0.05;
```

```
            if(ns < 0) ns = 0;
```

```
            clr.setHslF(clr.hslHueF(), clr.hslSaturationF(), ns);
```

```
            img.setPixelColor(x, y, clr);
```

```
        }
```

```
    }
```

```
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
```

```
    photos[curr] = img;
```

```
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
```

```
    ui->labl->show();
```

```
}
```

```
void MainWindow::on_huemn_clicked()
```

```
{
```

```
    ui->labl->clear();
```

```
    ui->labl->show();
```

```
    int curr = ui->listWidget->currentRow();
```

```
    QImage img = photos[curr];
```

```
    for (auto x = 0; x < img.width(); x++)
```

```
    {
```

```
        for (auto y = 0; y < img.height(); y++)
```

```
        {
```

```
            QColor clr = img.pixelColor(x, y);
```

```
            qreal ns = clr.hslHueF() - 0.05;
```

```
            if(ns < 0) ns = 0;
```

```
            clr.setHslF(ns, clr.hslSaturationF(), clr.lightnessF());
```

```
            img.setPixelColor(x, y, clr);
```

```
        }
```

```
    }
```

```
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
```

```

        photos[curr] = img;
        ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
        ui->labl->show();
    }

void MainWindow::on_huepl_clicked()
{
    ui->labl->clear();
    ui->labl->show();
    int curr = ui->listWidget->currentRow();
    QImage img = photos[curr];
    for (auto x = 0; x < img.width(); x++)
    {
        for (auto y = 0; y < img.height(); y++)
        {
            QColor clr = img.pixelColor(x, y);
            qreal ns = clr.hslHueF() + 0.05;
            if(ns > 1) ns = 1;
            clr.setHslF(ns, clr.hslSaturationF(), clr.lightnessF());
            img.setPixelColor(x, y, clr);
        }
    }
    photolist[currentImg]->setIcon(QPixmap::fromImage(img));
    photos[curr] = img;
    ui->labl->setPixmap(QPixmap::fromImage(photos[curr]));
    ui->labl->show();
}

void MainWindow::on_ChoseButton_2_clicked()
{
    Form2 *f = new Form2();
    f->setFixedSize(630,630);
    f->setWindowTitle("Choose background");
    f->show();
    backgr = f->result;
}

```

Файл “Collage2.pro”

```
QT += core gui
```

```
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
```

```

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs
deprecated before Qt 6.0.0

SOURCES += \
    form.cpp \
    form2.cpp \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    form.h \
    form2.h \
    mainwindow.h

FORMS += \
    form.ui \
    form2.ui \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    source.qrc

```

Файл “form.ui”

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>Form</class>
    <widget class="QWidget" name="Form">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>630</width>

```

```

        <height>630</height>
    </rect>
</property>
<property name="windowTitle">
    <string>Form</string>
</property>
<widget class="QPushButton" name="collage1">
    <property name="geometry">
        <rect>
            <x>10</x>
            <y>10</y>
            <width>300</width>
            <height>300</height>
        </rect>
    </property>
    <property name="text">
        <string/>
    </property>
    <property name="icon">
        <iconset>

<normaloff>../../../../Desktop/Collage1.png</normaloff>../../../../Desktop/Collage1.png</iconset>

    </property>
    <property name="iconSize">
        <size>
            <width>290</width>
            <height>300</height>
        </size>
    </property>
</widget>
<widget class="QPushButton" name="collage2">
    <property name="geometry">
        <rect>
            <x>320</x>
            <y>10</y>
            <width>300</width>
            <height>300</height>
        </rect>
    </property>
    <property name="text">
        <string/>
    </property>

```

```

    <property name="icon">
        <iconset>

<normaloff>../../Desktop/Collage2.png</normaloff>../../Desktop/Collage2.png</icons
et>
    </property>
    <property name="iconSize">
        <size>
            <width>290</width>
            <height>310</height>
        </size>
    </property>
</widget>
<widget class="QPushButton" name="collage3">
    <property name="geometry">
        <rect>
            <x>170</x>
            <y>320</y>
            <width>300</width>
            <height>300</height>
        </rect>
    </property>
    <property name="text">
        <string/>
    </property>
    <property name="icon">
        <iconset>

<normaloff>../../Desktop/Collage3.png</normaloff>../../Desktop/Collage3.png</icons
et>
    </property>
    <property name="iconSize">
        <size>
            <width>290</width>
            <height>300</height>
        </size>
    </property>
</widget>
</widget>
<resources/>
<connections/>
</ui>

```

Файл “form2.ui”

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>Form2</class>
  <widget class="QWidget" name="Form2">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>630</width>
        <height>630</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Form</string>
    </property>
    <widget class="QPushButton" name="pushButton">
      <property name="geometry">
        <rect>
          <x>10</x>
          <y>10</y>
          <width>300</width>
          <height>300</height>
        </rect>
      </property>
      <property name="text">
        <string/>
      </property>
      <property name="icon">
        <iconset resource="source.qrc">
          <normaloff>:/sky.jpg</normaloff>:/sky.jpg</iconset>
        </property>
        <property name="iconSize">
          <size>
            <width>600</width>
            <height>600</height>
          </size>
        </property>
      </widget>
    <widget class="QPushButton" name="pushButton_2">
      <property name="geometry">
        <rect>
```

```

        <x>320</x>
        <y>10</y>
        <width>300</width>
        <height>300</height>
    </rect>
</property>
<property name="text">
    <string/>
</property>
<property name="icon">
    <iconset resource="source.qrc">
        <normaloff>:/bricks.jpg</normaloff>:/bricks.jpg</iconset>
    </property>
<property name="iconSize">
    <size>
        <width>600</width>
        <height>600</height>
    </size>
</property>
</widget>
<widget class="QPushButton" name="pushButton_3">
    <property name="geometry">
        <rect>
            <x>10</x>
            <y>320</y>
            <width>300</width>
            <height>300</height>
        </rect>
    </property>
    <property name="text">
        <string/>
    </property>
    <property name="icon">
        <iconset resource="source.qrc">
            <normaloff>:/Grass.jpg</normaloff>:/Grass.jpg</iconset>
        </property>
    <property name="iconSize">
        <size>
            <width>600</width>
            <height>600</height>
        </size>
    </property>
</widget>

```

```

<widget class="QPushButton" name="pushButton_4">
  <property name="geometry">
    <rect>
      <x>320</x>
      <y>320</y>
      <width>300</width>
      <height>300</height>
    </rect>
  </property>
  <property name="text">
    <string/>
  </property>
  <property name="icon">
    <iconset resource="source.qrc">
      <normaloff>:/water.jpg</normaloff>:/water.jpg</iconset>
    </property>
  <property name="iconSize">
    <size>
      <width>600</width>
      <height>600</height>
    </size>
  </property>
</widget>
</widget>
<resources>
  <include location="source.qrc"/>
</resources>
<connections/>
</ui>

```

Файл “mainwindow.ui”

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>1000</width>
        <height>650</height>
      </rect>

```

```
</property>
<property name="windowTitle">
  <string>MainWindow</string>
</property>
<widget class="QWidget" name="centralwidget">
  <widget class="QListWidget" name="listWidget">
    <property name="enabled">
      <bool>true</bool>
    </property>
    <property name="geometry">
      <rect>
        <x>880</x>
        <y>0</y>
        <width>111</width>
        <height>471</height>
      </rect>
    </property>
    <property name="mouseTracking">
      <bool>true</bool>
    </property>
  </widget>
  <widget class="QPushButton" name="AddButton">
    <property name="geometry">
      <rect>
        <x>880</x>
        <y>480</y>
        <width>100</width>
        <height>20</height>
      </rect>
    </property>
    <property name="text">
      <string>Add Photo</string>
    </property>
  </widget>
  <widget class="QPushButton" name="SaveButton">
    <property name="geometry">
      <rect>
        <x>880</x>
        <y>510</y>
        <width>100</width>
        <height>20</height>
      </rect>
    </property>
```

```
<property name="text">
  <string>Save Project</string>
</property>
</widget>
<widget class="QPushButton" name="DeleteButton">
  <property name="geometry">
    <rect>
      <x>880</x>
      <y>540</y>
      <width>100</width>
      <height>20</height>
    </rect>
  </property>
  <property name="text">
    <string>Delete All</string>
  </property>
</widget>
<widget class="QPushButton" name="ChooseButton">
  <property name="geometry">
    <rect>
      <x>0</x>
      <y>0</y>
      <width>131</width>
      <height>31</height>
    </rect>
  </property>
  <property name="text">
    <string>Choose a collage style</string>
  </property>
</widget>
<widget class="QPushButton" name="saturationpl">
  <property name="geometry">
    <rect>
      <x>0</x>
      <y>570</y>
      <width>51</width>
      <height>21</height>
    </rect>
  </property>
  <property name="text">
    <string>+10</string>
  </property>
</widget>
```

```

<widget class="QPushButton" name="saturationmn">
  <property name="geometry">
    <rect>
      <x>50</x>
      <y>570</y>
      <width>51</width>
      <height>21</height>
    </rect>
  </property>
  <property name="text">
    <string>-10</string>
  </property>
</widget>
<widget class="QLabel" name="label">
  <property name="geometry">
    <rect>
      <x>-10</x>
      <y>550</y>
      <width>111</width>
      <height>20</height>
    </rect>
  </property>
  <property name="text">
    <string>Sturation</string>
  </property>
  <property name="alignment">
    <set>Qt::AlignCenter</set>
  </property>
</widget>
<widget class="QPushButton" name="lihtnesspl">
  <property name="geometry">
    <rect>
      <x>120</x>
      <y>570</y>
      <width>51</width>
      <height>21</height>
    </rect>
  </property>
  <property name="text">
    <string>+10</string>
  </property>
</widget>
<widget class="QPushButton" name="lihtnessmn">

```

```
<property name="geometry">
  <rect>
    <x>170</x>
    <y>570</y>
    <width>51</width>
    <height>21</height>
  </rect>
</property>
<property name="text">
  <string>-10</string>
</property>
</widget>
<widget class="QLabel" name="label_2">
  <property name="geometry">
    <rect>
      <x>120</x>
      <y>550</y>
      <width>111</width>
      <height>20</height>
    </rect>
  </property>
  <property name="text">
    <string>Lihtness</string>
  </property>
  <property name="alignment">
    <set>Qt::AlignCenter</set>
  </property>
</widget>
<widget class="QLabel" name="label_3">
  <property name="geometry">
    <rect>
      <x>240</x>
      <y>550</y>
      <width>111</width>
      <height>20</height>
    </rect>
  </property>
  <property name="text">
    <string>Hue</string>
  </property>
  <property name="alignment">
    <set>Qt::AlignCenter</set>
  </property>
```

```
</widget>
<widget class="QPushButton" name="huepl">
  <property name="geometry">
    <rect>
      <x>240</x>
      <y>570</y>
      <width>51</width>
      <height>21</height>
    </rect>
  </property>
  <property name="text">
    <string>+10</string>
  </property>
</widget>
<widget class="QPushButton" name="huemn">
  <property name="geometry">
    <rect>
      <x>290</x>
      <y>570</y>
      <width>51</width>
      <height>21</height>
    </rect>
  </property>
  <property name="text">
    <string>-10</string>
  </property>
</widget>
<widget class="QLabel" name="lab1">
  <property name="geometry">
    <rect>
      <x>10</x>
      <y>40</y>
      <width>511</width>
      <height>491</height>
    </rect>
  </property>
  <property name="text">
    <string/>
  </property>
  <property name="scaledContents">
    <bool>true</bool>
  </property>
  <property name="alignment">
```

```

        <set>Qt::AlignHCenter|Qt::AlignTop</set>
    </property>
</widget>
<widget class="QPushButton" name="ChooseButton_2">
    <property name="geometry">
        <rect>
            <x>150</x>
            <y>0</y>
            <width>181</width>
            <height>31</height>
        </rect>
    </property>
    <property name="text">
        <string>Choose a collage background</string>
    </property>
</widget>
</widget>
<widget class="QMenuBar" name="menubar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>1000</width>
            <height>20</height>
        </rect>
    </property>
</widget>
<widget class="QStatusBar" name="statusbar">
    <property name="mouseTracking">
        <bool>true</bool>
    </property>
    <property name="autoFillBackground">
        <bool>true</bool>
    </property>
</widget>
<widget class="QToolBar" name="toolBar">
    <property name="windowTitle">
        <string>toolBar</string>
    </property>
    <attribute name="toolBarArea">
        <enum>TopToolBarArea</enum>
    </attribute>
    <attribute name="toolBarBreak">

```

```
    <bool>false</bool>
  </attribute>
</widget>
</widget>
<resources/>
<connections/>
</ui>
```