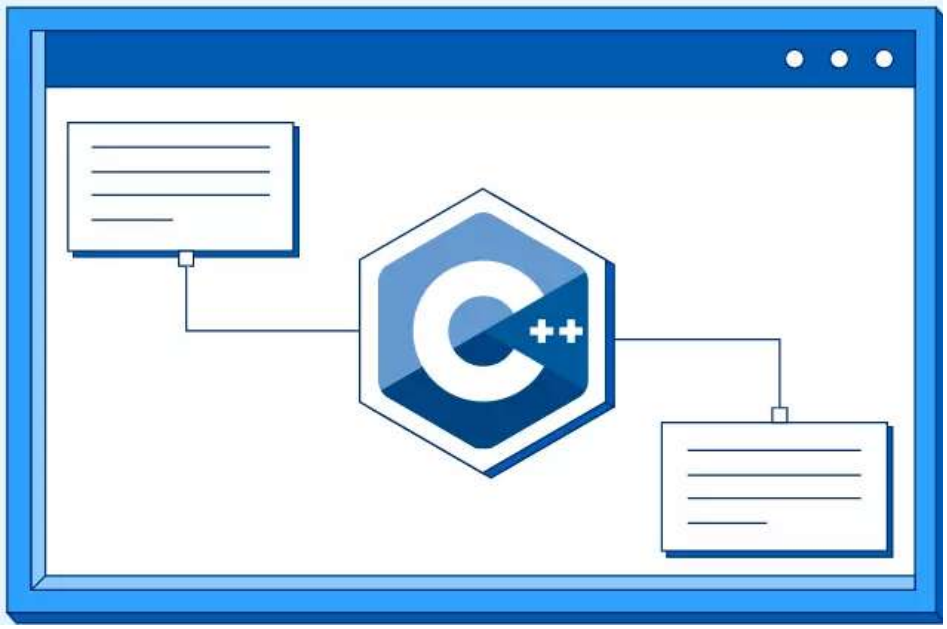
О. М. Рейда, А. В. Денисюк

Розробка програмних застосунків ОС Windows до виконання курсової роботи з дисципліни «Операційні системи» для здобувачів спеціальності 121 «Інженерія програмного забезпечення»



Міністерство освіти і науки України
Вінницький національний технічний університет

О. М. Рейда, А. В. Денисюк

Розробка програмних застосунків ОС Windows до виконання курсової роботи з дисципліни «Операційні системи» для здобувачів спеціальності 121 «Інженерія програмного забезпечення»

Електронний навчальний посібник

‘

Вінниця
ВНТУ
2025

УДК 004.42
Р35

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 13 від 25.06.2024 р.)

Рецензенти:
І. З. Лютак, доктор технічних наук, професор
С. В. Павлов, доктор технічних наук, професор
О. К. Колесницький, кандидат технічних наук, професор

Рейда, О. М.

Р35 Розробка програмних застосунків ОС Windows до виконання курсової роботи з дисципліни «Операційні системи» для здобувачів спеціальності 121 «Інженерія програмного забезпечення» : електронний навчальний посібник [Електронний ресурс] / О. М. Рейда, А. В. Денисюк. – Вінниця : ВНТУ, 2025. – (PDF, 93 с.)

Посібник присвячений матеріалам до виконання курсової роботи з дисципліни «Операційні системи» для здобувачів, що навчаються за спеціальністю 121 «Інженерія програмного забезпечення» денної та заочної форм навчання.

Мета посібника – надати здобувачам можливість більш детально ознайомитися з процесом розробки програмних застосунків, що охоплює: аналіз аналогів, методів і засобів розробки, розробку моделей, структур, діаграм і схем, проведення тестування роботи застосунків.

Перелік та зміст тем відповідає програмі вказаної вище дисципліни.

УДК 004.42

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

Курсова робота з дисципліни «Операційні системи» (ОС) виконується згідно з індивідуальним завданням і є самостійною роботою здобувача, призначеною для закріплення, розширення, узагальнення та практичного використання знань, умінь і навичок, одержаних під час навчання. У процесі виконання курсової роботи здобувачі здобувають навички проєктування та розробки програмних застосунків.

1.1 Загальні вимоги щодо організації курсового проєктування

Організація курсового проєктування здійснюється відповідно до «Положення про курсове проєктування», рекомендованого до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 12 від 26.06.2018 р.).

Відповідальність за правильність прийнятих рішень, обґрунтувань, розрахунків та якість оформлення несе здобувач – автор роботи.

Тематика курсових робіт може бути типовою та спеціалізованою. Завдання на спеціалізовані курсові роботи незалежно від об’єкта проєктування має передбачати розробку програмних застосунків, бути узгодженим з керівником і консультантом курсової роботи і затвердженим завідувачем кафедри.

Курсова робота має задовольняти такі вимоги:

- обсяг текстової частини визначається кількістю кредитів СРС, які виділяються на курсову роботу (1,5 кредиту, але не менше 45 годин) та не має перевищувати 35 сторінок формату А4;
- графічна частина може подаватися в тексті пояснювальної записки у вигляді відповідних рисунків або виноситись в додатки з обов’язковим конкретним зазначенням графічного матеріалу в індивідуальному завданні;
- у випадку повного збігання тем курсової роботи індивідуальне завдання має містити не тільки різні числові вихідні дані, але й передбачати самостійне викладення здобувачем тексту пояснювальної записки з метою уникнення використання одного і того ж електронного варіанта.

Індивідуальне завдання в перелік змісту не вноситься та має бути другою сторінкою після титульного аркуша. Зразок індивідуального завдання до курсової роботи наведено в додатку А.

Будь-яке переписування матеріалів літературних джерел або електронних документів (електронних книг, INTERNET-сайтів) неприпустимо. Якщо здобувач вважає за необхідність наведення певної кількості описових матеріалів, то вони розміщуються у додатках.

Згідно з затвердженим графіком здобувач зобов’язаний своєчасно подавати керівникові результати роботи над курсовим проєктом.

1.2 Порядок захисту курсової роботи

До захисту допускаються курсові роботи, виконані в повному обсязі згідно з затвердженим індивідуальним завданням.

Захист робіт проводиться публічно за встановленим графіком перед комісією з двох–трьох викладачів, склад якої затверджується завідувачем кафедри, за безпосередньої участі керівника проєкту у присутності здобувачів групи. Здобувач робить доповідь з теми до 5–10 хвилин з використанням ілюстративного матеріалу у вигляді електронної презентації або плакатів. Після доповіді члени комісії ставлять запитання за темою роботи.

За результатами захисту комісія на закритому засіданні визначає оцінку, яка потім оголошується здобувачу. У результаті захисту курсова робота оцінюється п’ятибальною оцінкою і відповідною їй модульною оцінкою за кредитно-модульною системою залежно від якості виконання та оформлення роботи і рівня відповідей на запитання при захисті роботи.

Курсові роботи, виконані не за своїм варіантом завдання, або не в повному обсязі чи з суттєвими помилками, виконані несамостійно (про що свідчить некомпетентність у рішеннях та матеріалах), до захисту не допускаються і направляються керівником роботи на доопрацювання. У цьому випадку у заліковій відомості робиться запис «не допущений», що еквівалентно одержаній оцінці «незадовільно», тобто свідчить про появу академічної заборгованості, яка може бути ліквідована на загальних підставах.

1.3 Структура курсової роботи

За змістом курсова робота (КР) має відповідати індивідуальному завданню. КР містить низку обов’язкових складових частин, перелік та вимоги до яких конкретизуються керівником. Кожну складову частину потрібно починати з нового аркуша.

Пояснювальна записка (ПЗ) до курсової роботи – основний документ, що відображає всі етапи та результати виконання роботи. Порядок подання складових частин ПЗ і їх рекомендований обсяг:

1) титульний аркуш ПЗ;
2) індивідуальне завдання на КР;
3) анотація (до 1 с.);
4) зміст;
5) вступ (до 2 с.);
6) основна (технічна) частина та її розділи (до 30 с.):
- аналіз сучасного стану питання та обґрунтування задачі;
- розробка інтерфейсу програмного застосунку;
- розробка моделі програмного застосунку;
- тестування програмного застосунку;
7) висновки (до 2 с.);
8) література (до 2 с.).

Кожний розділ ПЗ може складатися з підрозділів, пунктів і т. д.

Обсяг основної частини має складати до 30 с. друкованого тексту на аркушах формату А4, причому обсяг технічної частини має складати не менше 70% всієї пояснювальної записки.

2 ВИМОГИ ЩОДО ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

2.1 Загальні вимоги

Текст роботи оформлюється відповідно до державного стандарту України ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання [1]; Наказу Міністерства освіти і науки України № 40 від 12.01.2017 «Про затвердження Вимог до оформлення дисертації» [2]; ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання [3]; Етичного кодексу ученого України (Бюлетень ВАК України, № 11, 2011) [4]; ДСТУ 3582:2013. Інформація та документація. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила [4]; ДСТУ 6095:2009. Система стандартів з інформації, бібліотечної та видавничої справи. Правила скорочення заголовків і слів у заголовках публікації [5]; ДСТУ 7093:2009. Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Скорочення слів і словосполучень, поданих іноземними європейськими мовами [6].

Бібліографічний опис у списку використаних джерел, списку скорочень назв лексикографічних джерел та списку скорочень назв джерел ілюстративного матеріалу складається згідно з вимогами державного стандарту до бібліографічного опису наукової літератури (таблиця Д.1).

Вимоги до оформлення тексту пояснювальної записки:

- текст друкувати українською мовою з одного боку аркуша білого паперу формату А4 (210 · 297 мм);
- поля тексту: *вгорі та внизу – 20 мм, з лівого боку – 25 мм, з правого боку – 10 мм;*
- абзац починається з 5-го знаку (1,25 мм);
- шрифт – Times New Roman, кегль – 14 пунктів, міжрядковий інтервал – 1,5;
- текст друкувати без переносів;
- примітки чи виноски на сторінці не робити.

Друкарські помилки, описки, графічні неточності, які виявили під час написання роботи, можна виправляти підчищенням або зафарбуванням коректором та нанесенням на тому ж місці або між рядками виправленого тексту (фрагмента рисунка) машинописним способом. Допускається наявність не більше двох виправлень на одній сторінці.

Текст основної частини поділяють на розділи, підрозділи, пункти та підпункти.

Заголовки структурних частин роботи АНОТАЦІЯ, ЗМІСТ, ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ВСТУП, РОЗДІЛ 1, РОЗДІЛ 2, РОЗДІЛ 3, РОЗДІЛ 4, ВИСНОВКИ, ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ, ДОДАТКИ друкують великими літерами напівжирним шрифтом (окрім висновків до розділів) симетрично до тексту з нової сторінки, без крапки, наприклад: заголовки

висновків до розділів друкують великими літерами, без виділення напівжирним шрифтом симетрично до тексту з нової сторінки, назву розділу потрібно друкувати з нового рядка з номером розділу. Крапка після номера розділу не ставиться. Заголовки підрозділів, пунктів і підпунктів звіту потрібно друкувати напівжирним шрифтом з абзацного відступу з великої літери без крапки в кінці. Заголовки підрозділів друкують маленькими літерами (крім першої великої) з абзацного відступу, виділяючи напівжирним шрифтом. Відстань між заголовком, приміткою, прикладом і подальшим або попереднім текстом має бути не менше ніж два міжрядкових інтервали. Відстань між основами рядків заголовка, а також між двома заголовками приймають такою як у тексті звіту.

Не можна починати новий підрозділ чи пункт у кінці сторінки, якщо після назв підрозділу чи пункту наводиться менше 2-х рядків основного тексту. У такому випадку примусово починають новий підрозділ чи пункт з нової сторінки.

До обсягу основного тексту не входять: список використаних джерел, список скорочень назв лексикографічних джерел, але всі сторінки зазначених елементів роботи підлягають наскрізній нумерації і входять до загального обсягу роботи.

Заголовки структурних елементів та розділів друкують великими літерами напівжирним шрифтом без крапки у кінці посередині рядка рисунок 2.1.

ВСТУПУ

Актуальність теми зумовлена потребою підвищення ефективності роботи операційної системи у зв'язку з низьким рівнем оптимізації записів реєстру, що впливає на збільшення ресурсів і, як наслідок, зменшення швидкодії роботи ПК.

Рисунок 2.1 – Приклад оформлення заголовків структурних елементів та розділів

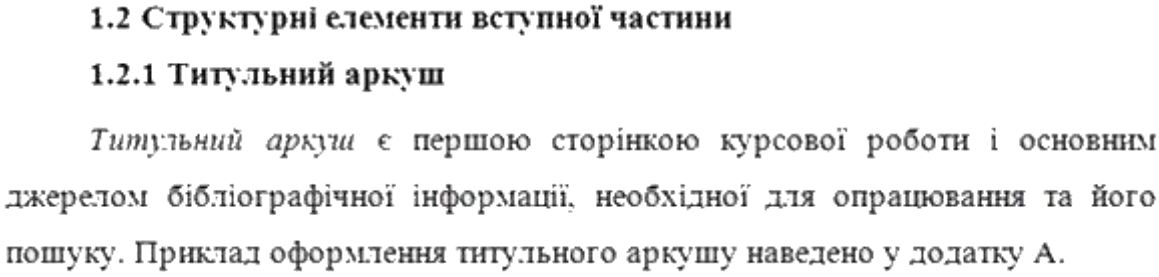
Заголовки підрозділів, пунктів і підпунктів друкують з абзацного відступу звичайним шрифтом з великої літери без крапки у кінці (рисунок 2.2).

1 ВИМОГИ ДО СТРУКТУРИ ТА ЗМІСТУ КУРСОВОЇ РОБОТИ

Курсова робота з навчальної дисципліни – це індивідуальне завдання, що передбачає створення сукупності документів (розрахунково-пояснювальної записки, текстового матеріалу, при необхідності графічного матеріалу) і являє собою творче або репродуктивне вирішення конкретного завдання, спрямованого на об'єкти діяльності фахівця (пристрої, устаткування, механізми, апаратні і програмні засоби, матеріали, процеси, явища, властивості тощо),

Рисунок 2.2 – Приклад оформлення заголовків підрозділів, пунктів і підпунктів

Заголовки відокремлюються від тексту одним пустим рядком (до і після тексту). Відстань між заголовками, як у тексті 1,5 інтервала (рисунок 2.3).


1.2 Структурні елементи вступної частини

1.2.1 Титульний аркуш

Титульний аркуш є першою сторінкою курсової роботи і основним джерелом бібліографічної інформації, необхідної для опрацювання та його пошуку. Приклад оформлення титульного аркушу наведено у додатку А.

Рисунок 2.3 – Приклад відокремлення заголовків підрозділів, пунктів і підпунктів

Не дозволено розміщувати назву розділу, підрозділу, а також пункту й підпункту на останньому рядку сторінки.

2.1.1 Нумерація

Нумерація сторінок, розділів, підрозділів, пунктів, підпунктів, рисунків, таблиць подається арабськими цифрами без знака «№».

Першою сторінкою курсової роботи є титульний аркуш, який вноситься до загальної нумерації сторінок, але номер сторінки не проставляється. На наступних сторінках (починаючи з 2-ої) номер розташовують у правому верхньому кутку сторінки без крапки в кінці.

Після номера розділу крапка не ставиться, заголовок розділу друкується після номера. Номер розділу і його назва друкуються великими літерами, напівжирним шрифтом й відцентровуються. Такі структурні частини дипломної роботи, як ЗМІСТ; ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ВСТУП; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ не мають порядкового номера.

Підрозділи (параграфи) нумеруються у межах кожного розділу. Номер підрозділу складається з номера розділу й порядкового номера підрозділу (параграфа), між якими ставиться крапка. В кінці номера підрозділу крапка не ставиться і далі, в цьому ж рядку, йде заголовок підрозділу. Крапка після назви розділу не ставиться.

2.1.2 Ілюстрації

Якщо в тексті роботи використовуються ілюстрації (рисунки, графіки, схеми, фотографії, діаграми) і таблиці (необхідний елемент наукової праці), то вони подаються безпосередньо після тексту, де вони згадуються вперше, або у додатках. Якщо ілюстрація чи таблиця мають розмір більший за формат А4, то їх розміщують у додатках.

Кожна ілюстрація має відповідати тексту, а текст – ілюстрації.

Назви ілюстрацій розміщують після їхніх номерів. За необхідності ілюстрації доповнюють пояснювальними даними (підрисунковий підпис).

Підпис під ілюстрацією зазвичай має такі елементи:

- найменування графічного сюжету позначають словом «Рисунок»;

- порядковий номер ілюстрації, який вказується без знака «№» арабськими цифрами. Нумерують ілюстрації послідовно в межах розділу (за винятком ілюстрацій, поданих у додатках). Номер ілюстрації складається з номера розділу і порядкового номера ілюстрації, між якими ставиться крапка.

Наприклад: «Рисунок 1.3» (третій рисунок першого розділу). Назву рисунка друкують з великої літери та розміщують під ним посередині рядка, (наприклад, Рисунок 1.3 – Назва рисунка).

Якщо ілюстрація виноситься у додаток, її номер складається з літери додатка і порядкового номера, між якими ставиться крапка, наприклад: «Рисунок А.1» (перший рисунок додатка А). Номер ілюстрації, її назва і пояснювальний підпис розміщується у додатку так само, як і у тексті роботи, безпосередньо під ілюстрацією відцентровано, без виділень (рисунок 2.4)

Посилання на рисунок розміщують у тексті в круглих дужках «(рисунок 2.4)» або зворот типу «…як це видно з рисунка 2.4», або «…як це показано на рисунку 2.4».

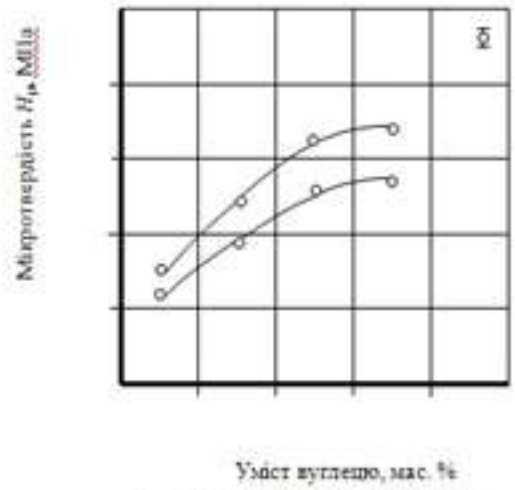


Рисунок 2.1 – Залежність мікротвердості композиційного матеріалу від умісту вуглецю спеченого за різних температур

Рисунок 2.4 – Приклад оформлення рисунка

За ДСТУ 3008:2015 пояснювальні дані до рисунка подають безпосередньо після графічного матеріалу перед назвою рисунка, яку друкують з великої літери

Пояснювальні дані і назву рисунка розміщують посередині рядка без абзацного відступу.

2.1.3 Таблиці

Таблиці нумерують послідовно в межах розділу. Назву таблиці друкують з великої літери і розміщують над таблицею з абзацного відступу. Номер таблиці складається з номера розділу і порядкового номера таблиці, між якими

ставиться крапка (рисунок 2.5). Наприклад, «Таблиця 1.2» (друга таблиця першого розділу).

Якщо таблицю винесено у додаток, її номер має складатися з літери додатка і порядкового номера таблиці, між якими ставиться крапка. Наприклад, «Таблиця А.1» (перша таблиця додатка А).

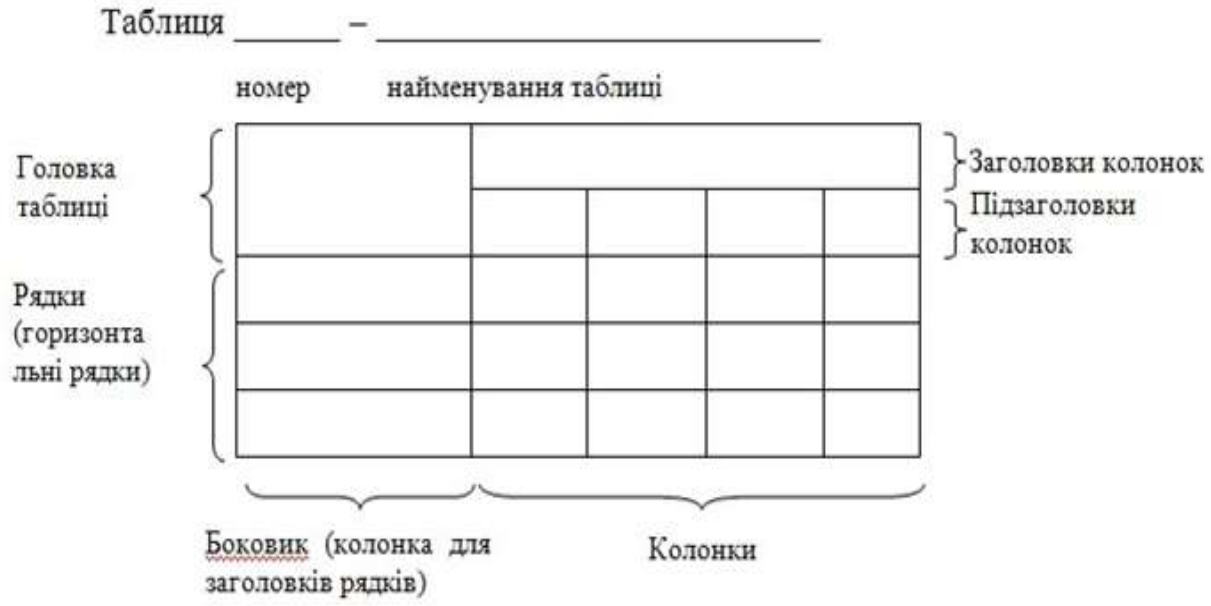


Рисунок 2.5 – Приклад оформлення таблиці

Назву таблиці друкують з великої літери і розміщують над таблицею з абзацного відступу.

Назву і слово «Таблиця» починають з великої літери. Приклад побудови назви таблиці: Таблиця 1.2 – Назва таблиці.

Якщо таблиця переноситься на наступну сторінку, то над таблицею з абзацного відступу пишеться «Продовження таблиці» і вказується її номер. Заголовки граф пишуть з великої літери, підзаголовки – з малої, якщо вони становлять одне речення з заголовком, і з великої, якщо вони є самостійними. Заголовки (як підпорядковані, так і головні) мають бути максимально точними та простими. В них не має бути слів, що повторюються. Висота рядків має бути не меншою 8 мм.

Таблицю розміщують після першого згадування про неї в тексті таким чином, щоб її можна було читати без повороту переплетеного блока роботи або з поворотом за годинниковою стрілкою. Таблицю з великою кількістю рядків можна переносити на інший аркуш, у цьому разі назву вміщують тільки над її першою частиною (рисунок 2.6). Таблицю з великою кількістю граф можна ділити на частини і розміщувати одну під одною в межах тієї самої сторінки.

Усі наведені в таблицях дані мають бути достовірними, однорідними і такими, що можуть зіставлятися, в основу їх групування покладають лише суттєві ознаки.

Таблиця 2.1 – Характеристики зразків з композиції Fe-самофлюсівні сплави, спечених за температури 1150 °С упродовж 60 хв.

Тиск пресування, *МПа*	Розміри зразків, *мм*		Вага зразка, *г*	Щільність, *г/см³*	Відносна щільність, %	Усадка, %
	Висота	Діаметр				
1	2	3	4	5	6	7
300	17,6	11,35	10,693	6,03	77,34	0
400	12,77	25,42	39,542	6,119	78	0,012

Продовження таблиці 2.1

1	2	3	4	5	6	7
500	12,43	25,43	39,331	6,233	79,5	0,014
500	1,705	1,142	10,981	6,292	80,6	0,006
700	1,134	2,548	37,612	6,51	83,5	0,0139
700	1,56	1,143	10,806	6,76	86,7	0,005
800	0,942	1,619	12,9025	6,664	85,44	0,001

Рисунок 2.6 – Приклад оформлення продовження таблиці

2.1.4 Оформлення лістингів

В основній частині роботи для ілюстрації викладеного теоретичного матеріалу мають наводитися лістинги фрагментів програм, які потрібно розташовувати безпосередньо після тексту, в якому вони вперше згадуються. На всі лістинги мають бути дані посилання в тексті роботи.

Лістинг програми – це подання коду комп'ютерної програми певною мовою програмування. Лістинг використовується для подання коду програми з метою відображення функціональних частин коду. Він має чітко відображати структуру коду, охоплюючи класи, функції, змінні та інші елементи програми. Лістинг має бути організований з використанням відступів, коментарів та інших засобів форматування, щоб полегшити читання та розуміння коду.

Лістинги мають мати порядкову нумерацію в межах кожного розділу. Номер лістинга має складатись з номера розділу та порядкового номера лістинга, розділених крапкою, наприклад «Лістинг 3.2» – другий лістинг третього розділу. При посиланні на лістинг потрібно писати слово «лістинг» із зазначенням його номера. Лістинги, розміщені в додатках, нумерують у межах кожного додатка, наприклад: «Лістинг А.1.2» – другий лістинг першого розділу додатка А. Рекомендується відокремлювати смислові блоки пустими рядками, а також візуально позначати вкладені конструкції (програмний код у фігурних дужках) за допомогою відступів.

Назва лістинга друкується тим же шрифтом, що і основний текст, та розміщується над лістингом зліва, без абзацного відступу через тире після номера лістингу. Крапка після назви не ставиться.

На рисунку 2.7 зображено приклад оформлення лістинга.

Лістинг 1 – Функція отримання кількості елементів вектора

```
int getlength() {
        int datalength = 0;
         if (data) datalength = data->size();
         return datalength;}
```

Рисунок 2.7 – Приклад оформлення лістинга

2.1.5 Формули та рівняння

Формули та рівняння потрібно розташовувати безпосередньо після тексту, де вони згадуються, посередині рядка.

Вище і нижче кожного рівняння або формули потрібно залишити один пустий рядок.

Рівняння і формули в роботі (за винятком формул і рівнянь, наведених у додатках) потрібно нумерувати послідовно в межах даного розділу.

Номер рівняння або формули складається з номера розділу і порядкового номера формули в межах даного розділу, що відокремлені крапкою, наприклад, формула (2.20) – двадцята формула другого розділу роботи.

Номер рівняння або формули зазначають на рівні рівняння або формули в дужках у крайньому правому положенні. У багаторядкових формулах або рівняннях номер проставляють на рівні останнього рядка.

За ДСТУ 3008-2015 пояснення познак до формули треба подавати без абзацного відступу з нового рядка у тій послідовності, в якій вони наведені у формулі чи рівнянні. Перший рядок пояснення починають словом «де» без двокрапки (рисунок 2.8).

$$E = mV^2/2, \qquad (5.2)$$

де E – кінетична енергія, *Дж*;

m – маса матеріальної точки, що дорівнює 0,5 *кг*;

V – швидкість руху, що дорівнює 30 *м/с*.

Допускається позначення одиниць фізичних величин у поясненнях позначень величин.

Рисунок 2.8 – Приклад оформлення формули

2.1.6 Переліки

За потреби переліки можуть бути наведені всередині пунктів або підпунктів. Перед переліком ставлять двокрапку.

Перед кожною позицією переліку потрібно ставити малу літеру української абетки з дужкою, або, не нумеруючи, – дефіс (перший рівень деталізації). Для подальшої деталізації переліку потрібно використовувати арабські цифри з дужкою (другий рівень деталізації).

Приклад:

а) форма і розмір клітин;

б) живий склад клітин:

1) частини клітин;

2) неживі включення протопластів;

в) утворення тканини».

Переліки першого рівня деталізації друкують малими літерами з абзацного відступу, другого рівня – з відступом відносно місця розташування переліків першого рівня (рисунок 2.9).

Підпорядкованість у переліку позначають малими літерами української абетки, далі – арабськими цифрами, далі – через знаки «тире». Відступ кожного наступного рівня деталізації 1,25 см Літери ґ, є, з, і, ї, й, о, ч та ь НЕ використовуються!!!

Рисунок 2.9 – Приклад оформлення переліку

2.1.7 Посилання

У тексті звіту можна робити посилання на структурні елементи самого звіту та інші джерела.

У разі посилання на структурні елементи самого звіту зазначають, відповідно, номери розділів, підрозділів, пунктів, підпунктів, позицій переліків, рисунків, формул, рівнянь, таблиць, додатків.

Посилаючись, потрібно використовувати такі вирази: «у розділі 4», «див. 2.1», «відповідно до 2.3.4.1», «(рисунок 1.3)», «відповідно до таблиці 3.2», «згідно з формулою (3.1)», «у рівняннях (1.23)–(1.25)», «(додаток Г)» тощо.

Дозволено в посиланні використовувати загальноприйняті та стандартизовані скорочення згідно з ДСТУ 3582, наприклад, «згідно з рис. 10», «див. табл. 3.3» тощо.

Під час роботи з літературою потрібно користуватися різними видами каталогів: систематичним (назви джерел розташовані за галузями знань), алфавітним (назви творів розташовані в алфавітному порядку) та предметним

(назви праць з конкретних проблем чи галузей науки). Крім того, потрібно звертатися також до періодичних видань.

У процесі опрацювання наукових джерел бажано робити записи, які можуть бути повними і точними (дослівними) або скороченими (коротким викладом матеріалів, що вивчаються) [17, С. 40]. Так, наприклад, можна зробити:

- детальний конспект тобто запис основних положень робіт, фактичного матеріалу, власних зауважень тощо;
- короткі записи зі своїми роздумами або без них;
- виписки у формі цитат.

Під час роботи над літературою здобувач виписує цитати. Вони можуть фіксуватися у зошитах. При написанні наукових робіт здобувач має обов'язково посилатися на авторів і джерела, з яких запозичив матеріали або окремі результати. Посилання бажано робити на останні публікації. На більш ранні видання можна посилатися лише в тих випадках, коли праці, в яких міститься потрібний матеріал, не перевидавалися.

Для підтвердження власних аргументів посиланнями на авторитетне джерело або для критичного аналізу того чи іншого друкованого твору потрібно наводити цитати. Науковий етикет вимагає точно відтворювати цитований текст, бо найменше скорочення наведеного витягу може спотворити зміст, викладений автором. У теоретичній частині на одній сторінці тексту рекомендовано наводити 2–3 цитати різних авторів.

Загальні вимоги до цитування такі:

1. Текст цитати починається і закінчується лапками і наводиться в граматичній формі, в якій він поданий в джерелі, зі збереженням особливостей авторського написання. Наукові терміни, запропоновані іншими авторами, не виділяються лапками, за винятком тих, що викликали загальну полеміку. У цих випадках використовується вираз «так званий»;

2. Цитування має бути повним, без довільного скорочення авторського тексту і без перекручень думок автора. Пропуск слів, речень, абзаців при цитуванні без перекручення авторського тексту позначається трьома крапками. Вони ставляться у будь-якому місці цитати (на початку, всередині, в кінці). Якщо перед випущеним текстом або за ним стояв розділовий знак, то він не зберігається;

3. Кожна цитата обов'язково супроводжується посиланням на джерело;

4. При непрямому цитуванні (переказі, викладі думок інших авторів своїми словами), що дає значну економію тексту, потрібно бути максимально точним у викладі думок автора, коректним щодо оцінювання його результатів, і робити відповідні посилання на джерела;

5. Цитування не має бути ні надмірним, ні недостатнім, бо це знижує рівень наукової праці: надмірне цитування створює враження компілятивності праці, а недостатнє – знижує наукову цінність викладеного матеріалу.

Коли використовують відомості, матеріали з монографій, оглядових статей, інших джерел, що мають велику кількість сторінок, тоді в посиланні

потрібно точно вказати номери сторінок, ілюстрацій, таблиць, на які дано посилання в роботі.

Посилання в тексті курсової роботи оформлюються за такими зразками: «...у праці Т. А. Казакової [12] ...», «В. М. Комісаров [15, С. 64] увів поняття ...», «Поняття функціональної граматики перехрещується, але не збігається з поняттям семантичної граматики» [2, С. 24]. Посилання в дужках на роботи різних авторів подаються через крапку з комою: [7; 15; 29].

Посилання на ілюстративний матеріал оформлюються у круглих дужках із зазначенням скорочення та вказанням сторінки (за наявності), наприклад: на всі таблиці роботи мають бути посилання в тексті, при цьому слово «таблиця" пишуть скорочено, наприклад: «… в табл. 1.2». При роботі з науковими джерелами потрібно пам'ятати, що власне дослідження передбачає, передусім, критичний аналіз цих джерел, а не механічне переписування чужих думок без відповідних посилань.

2.1.8 Оформлення переліку джерел посилання

Використані наукові джерела подаються у рубриці ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ (відцентрована назва ВЕЛИКИМИ ЛІТЕРАМИ, напівжирним шрифтом), в алфавітному порядку, з абзацу (1,25 см). Розмір шрифту – 14 пт, міжрядковий інтервал – 1,5.

Джерела, що використані для написання наукових робіт, потрібно розміщувати у списку літератури в алфавітному порядку за прізвищами авторів (якщо авторів декілька, то за прізвищами перших), а також заголовків праць. До списку літератури вносяться всі публікації вітчизняних і зарубіжних авторів, на які є посилання в роботі. Всі джерела вказуються тією мовою, якою вони видані. Спочатку наводиться в алфавітному порядку література за кириличним алфавітом (українською мовою). Література, опублікована іноземними мовами, які використовують латинський шрифт (англійська, німецька, французька), наводиться згодом наприкінці списку використаної літератури (без пропусків рядків) також за алфавітом.

Відомості про джерела, що внесені до списку, подаються згідно з вимогами державного стандарту до бібліографічного опису наукової літератури ДСТУ 8302:2015 [7].

Приклади оформлення бібліографічного опису у списку використаних джерел з урахуванням Національного стандарту України ДСТУ 8302:2015 наведено у таблиці Д.1.

2.1.9 Оформлення додатків

Додатки оформлюють як продовження основного дослідження, на наступних сторінках.

Кожен додаток починається з нової сторінки. Найчастіше додаток має свій заголовок, надрукований угорі малими літерами з першої великої симетрично відносно тексту на сторінці. Посередині рядка над заголовком великими літерами друкують слово «ДОДАТОК» і відповідну велику літеру української абетки, крім літер Ґ, Є, З, І, І, Й, О, Ч, Ь. Наприклад, ДОДАТОК А, ДОДАТОК Б і т. п.

Текст кожного додатка, за потреби, може поділятися на розділи й підрозділи, які нумерують у межах кожного додатка, наприклад: А.2 – другий розділ додатка А; В.2.1 – перший підрозділ другого розділу додатка В.

Подібним чином нумеруються ілюстрації, таблиці та формули, розміщені у додатках. Наприклад: Рисунок А.1.1 – перший рисунок першого розділу додатка А.

2.2 Зміст

Структурний елемент «Зміст» розташовують після анотації, починаючи на наступній сторінці.

У «Змісті» наводять такі структурні елементи: «Скорочення та умовні познаки», «Вступ», послідовно перелічено назви всіх розділів, підрозділів і пунктів (якщо вони мають назву) змістовної частини звіту (суті звіту), «Висновки», «Перелік джерел посилання», «Додатки» з їх назвою та зазначенням номера сторінки початку структурного елемента.

Нумерація сторінок має бути наскрізною.

Під час виконання робіт обсяг пояснювальної записки враховується до додатків. За умови, що додатки курсової роботи підтверджують цінність результатів розробки, обсяг пояснювальної записки і додатки мають мати наскрізну нумерацію. У додатку В наведено приклад типового змісту курсової роботи.

2.3 Складові частини пояснювальної записки

Пояснювальна записка курсової роботи має відповідати індивідуальному завданню й оформлюватися відповідно до чинних стандартів, яких потрібно дотримуватися при виконанні розробки з урахуванням всіх чинних офіційних змін. Зміст пояснювальної записки роботи (вихідні дані та перелік питань, що підлягають розробці) визначає керівник. При цьому можна виділити такі обов’язкові складові:

1) вступ;
2) аналіз сучасного стану питання та обґрунтування задачі;
3) розробка інтерфейсу програмного застосунку;
4) розробка моделі програмного застосунку;
5) тестування програмного застосунку;
6) висновки;
7) перелік джерел посилання;
8) додатки (за потреби).

«ВСТУП», «ВИСНОВКИ», «ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ», «ДОДАТКИ» як розділи не нумеруються.

Текст пояснювальної записки роботи має бути обґрунтованим і лаконічним.

2.4 Вступ

У вступі стисло викладають:

- оцінку сучасного стану об’єкта дослідження або розробки, розкриваючи практично розв’язані завдання провідними науковими установами та організаціями, а також провідними вченими й фахівцями певної галузі;
- світові тенденції розв’язання поставлених проблем і/або завдань;
- актуальність роботи та підстави для її виконання;
- мету роботи й можливі сфери застосування;
- предмет, об’єкт роботи;
- задачі і цілі.

Актуальність теми розкриває її значущість, тобто необхідність та невідкладність розгляду для потреб розвитку економіки держави, галузі, підприємства. Головним критерієм актуальності теми виступає можливість забезпечення найбільшого ефекту.

Для обґрунтування актуальності потрібно: обґрунтувати значення проблеми та наголосити на ступені її дослідження, довести практичну цінність дослідження, навести лаконічні, переконливі аргументи, які не викликають запитань, визначити цільову аудиторію.

Актуальність визначається на підставі таких факторів: недоліки аналогів, поява нових методів, засобів або принципів розробки, розвиток науково-технічного процесу, поява нових методів дослідження.

При формуванні актуальності можна використовувати такі фрази:

- «Актуальність теми зумовлена ...»;
- «На актуальність теми вказують такі фактори, як ...»;
- «Досліджувана тема актуальна, тому що…»;
- «... цим можна довести актуальність цього дослідження».

Виходячи з назви роботи, визначеного об’єкта та предмета, формується мета дослідження, що характеризує, яку найбільш важливу проблему або завдання має намір вирішити дослідник.

Мета дослідження – це очікуваний кінцевий результат, який зумовлює загальну спрямованість і логіку дослідження (теоретичного або прикладного).

Мета визначається відповіддю на запитання: «Для чого проводиться дослідження?». Мета дослідження полягає у вирішенні інженерної проблеми шляхом удосконалення вибраної сфери діяльності конкретного об’єкта. На завершальному етапі досліджень потрібно перевірити, чи відповідають висновки поставленій меті. Мета конкретизується та розвивається у завданнях дослідження.

Об’єктом прийнято називати те, на що спрямована пізнавальна діяльність дослідника, процес або явище, яке породжує проблемну ситуацію, обрану для дослідження.

Як об’єкт визначаються лише ті зв’язки, відносини, властивості реального об’єкта, які внесені до процесу пізнання. Будь-який об’єкт дослідження – це певна сукупність властивостей та відносин, яка існує незалежно від дослідника, але ним відображається.

Предмет дослідження – досліджувані з певною метою властивості об’єкта.

При визначенні предмета та об’єкта потрібно з’ясувати: предмет і об’єкт дослідження є новими чи традиційними.

Співвідношення об’єкта та предмета дослідження можна коротко охарактеризувати так: об’єкт об’єктивний, а предмет суб’єктивний (до речі, предмет англійською – subject).

Предмет – не частина, відрізана від об’єкта, а спосіб, аспект його вивчення. Об’єкт розглядається весь, цілісно. Предмет дослідження – все те, що знаходиться в межах об’єкта дослідження у визначеному аспекті розгляду.

Завдання дослідження:

- вирішення теоретичних питань, які пов’язані з проблемою дослідження (введення до обігу нових понять, розкриття їх сутності і змісту; розроблення нових критеріїв і показників; розроблення принципів, умов і факторів застосування окремих методик і методів);
- виявлення, уточнення, поглиблення, методологічне обґрунтування суттєвості, природи, структури об’єкта, що вивчається; виявлення тенденцій і закономірностей процесів; аналіз реального стану предмета дослідження, динаміки, внутрішніх протиріч розвитку;
- виявлення шляхів і засобів удосконалення явища, процесу, що досліджується (практичні аспекти роботи); обґрунтування системи заходів, потрібних для вирішення прикладних завдань;
- експериментальна перевірка розроблених пропозицій щодо розв’язання проблеми, підготовка методичних рекомендацій для їх використання на практиці.

Завдання мають розглядатись як основні етапи інженерного дослідження. Найчастіше формулювання таких завдань здійснюється у вигляді певного набору підпитань. Наприклад, «виявити…», «розробити…», «експериментально перевірити…» тощо.

2.5 Аналіз сучасного стану питання та обґрунтування задачі

Цей підрозділ є обов’язковим. Порівняльний аналіз та обґрунтування задачі має здійснюватися на рівні інженерного мислення з усебічним використанням сучасних досягнень науки та техніки. В цьому підрозділі потрібно описати та проаналізувати основну специфіку предметної області та поставленої задачі і сучасних підходів до вирішення такої чи аналогічних задач. Визначити основні функціональні властивості, параметри програмного застосунку, що розробляється, й провести аналіз аналогів відповідно до визначених параметрів.

Рекомендований обсяг цього підрозділу 5–7 сторінок тексту.

2.6 Основна частина пояснювальної записки

Ця частина пояснювальної записки курсової роботи містить розробку інтерфейсу, структурних схем, моделей програмного застосунку, алгоритмів і структур даних. Як правило, обсяг пояснювальної записки встановлюється в межах годин, що їх передбачено для вивчення дисципліни, і складає 25 сторінок разом з теоретичною частиною КР.

У цій частині пояснювальної записки роботи наводяться методики тестування програмного за стосунку і його опис.

Для курсових робіт, що пов’язані з математичним моделюванням і теоретичними дослідженнями, основна частина може складати 60% обсягу пояснювальної записки.

При цьому основна частина має бути логічно пов’язана з теоретичними відомостями з теми роботи, демонструватись ілюстративним матеріалом або таблицями з обов’язковим посиланням на ці рисунки або таблиці за текстом пояснювальної записки роботи.

Під час викладення тексту записки забороняється переписування матеріалів літературних джерел, сканування рисунків, що стосуються технічної частини. Але допускається використання сканованих рисунків, взятих з довідкової літератури, що вказується в оглядовій частині роботи, з обов’язковим посиланням на джерела. Як правило, додатки використовуються для розміщення частини описового змісту або розрахунків (таблиць), графічної інформації.

У тексті пояснювальної записки мають бути посилання на рисунки, таблиці, додатки, які входять до змісту роботи.

Графічна частина може подаватись як інформація в тексті пояснювальної записки або додатків, які чітко визначаються керівником в індивідуальному завданні роботи.

Якщо під час розробки виникає потреба в експериментальному дослідженні або моделюванні, то ця частина має бути детально обґрунтована і містити аналіз отриманих результатів досліджень або моделювання.

Під час виконання роботи рекомендується надавати перевагу використанню програмного продукту друку або машинному друку.

2.7 Висновки

Оформляють з нової пронумерованої сторінки з абзацу. Висновки є останньою частиною, підсумком рішення, що приймається, у виконаній роботі з зазначенням результатів, що досягнуто, і переваг предмета порівняно з існуючими аналогами. Також надаються можливі рекомендації прикладного застосування та перспективи удосконалення спроєктованого предмета.

При цьому бажано у тексті пояснювальної записки давати висновки в кожному розділі роботи, що є постановкою задачі до наступного розділу.

3 ЗМІСТ РОБОТИ

Курсова робота з дисципліни «Операційні системи» передбачає розробку прикладного програмного застосунку для роботи в операційній системі «Windows».

3.1 Аналіз завдань на курсове проєктування

Кожен із етапів виконання курсової роботи має передбачати багатоваріантний аналіз, обґрунтованість рішень, порівняльну характеристику та оцінювання відповідних параметрів.

Кількість варіантів завдань усуває можливість повного повторювання завдання як в межах групи, так і в суміжних групах навчального потоку. Варіанти завдань щорічно оновлюються.

Індивідуальне завдання на виконання курсової роботи (додаток А) передбачає розробку програмного застосунку для роботи в операційній системі «Windows».

Титульний аркуш наведено у додатку А.

Після титульного аркуша розміщується аркуш з індивідуальним завданням, виконаний відповідно до додатка Б.

Далі розміщується аркуш із анотацією і змістом.

Текст вступу, який розміщується після змісту, має бути коротким і висвітлювати питання актуальності, значення, сучасний рівень і призначення курсової роботи. У вступі і далі за текстом не дозволяється використовувати скорочені слова, терміни, крім загальноприйнятих.

Вступ (1–2 с.) має висвітлювати сучасний стан технологій проєктування програмних застосунків, мету та загальну постановку задачі, актуальність, яка має наводитись в першому абзаці вступу, з метою стислого викладення суті розробки. Потрібно визначити об’єкт та предмет розробки, основні задачі розробки і цілі, що були досягнуті. Схему вступу наведено у додатку Г.

3.2 Рекомендації щодо викладення розділу «Аналіз сучасного стану питання та обґрунтування задачі»

Розділ «Аналіз сучасного стану питання та обґрунтування задачі» передбачає аналіз відомих аналогів (програмний засіб такого ж призначення, що й досліджуваний, схожий з ним за технічною суттю та за результатом, що досягається при їх використанні). Для проведення аналізу потрібно визначити 3–4 аналоги.

Для кожного аналога потрібно навести його опис, що охоплює:

- назву програмного застосунку або системи;
- поточну версію і дату останнього випуску;
- розробника або власника продукту;
- короткий опис;
- перелік переваг і недоліків;
- зображення користувацького інтерфейсу головного вікна.

Після наведеного опису аналогів потрібно провести їх аналіз. Для проведення аналізу потрібно визначити 4–5 функціональних характеристик, які будуть відображати ефективність використання аналогів і програмного застосунку, що розробляється.

Функціональність – це сукупність властивостей, які визначають спроможність програмного забезпечення виконувати в заданому середовищі перелік функцій за вимогами до обробки і загальносистемних засобів [8, С. 147].

Аналіз здійснюється на основі вимог, що висуваються до створюваної системи та з урахуванням особливостей предметної області. Результати мають лягти в основу обґрунтування вибору платформи, технологій та середовища розробки програмного забезпечення. Для обґрунтування вибору застосовують методи прийняття рішень. Приклад аналізу аналогів застосунку редагування тексту регулярними виразами наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз функціональності

	Notepad++	Sublime	Word	REedit
Підтримка регулярних виразів	1	1	1	1
Використання шаблонів виразів	1	0	1	1
Створення шаблонів на основі регулярних виразів	0	0	0	1
Генератор регулярних виразів	0	0	0	1
Підсвічування результату	1	0	1	1
Сумарний коефіцієнт	3	1	3	5

Після проведення аналізу аналогів потрібно провести аналіз методів і засобів розробки. Аналіз методів розробки передбачає проведення ґрунтовного аналіз мов розробки застосунку.

Мова програмування – це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії. Розроблено багато різних мов програмування. Кожна мова програмування орієнтована на певний клас задач.

Для проведення аналізу потрібно визначити 3–4 мови програмування.

Для кожної мови потрібно навести опис, що містить:

- назву;
- поточну версію і дату останнього випуску;
- розробника або власника;
- короткий опис;
- перелік переваг і недоліків;

Результати аналізу мов програмування подати у таблиці. Приклад аналізу мов програмування наведено у таблиці 3.2.

Таблиця 3.2 – Порівняльний аналіз мов програмування

	Java	Python	C#	C++
Статична типізація	1	0	1	1
Використання оперативної пам’яті	0	0	1	1
Інструкція go to	0	0	1	1
Множинне наслідування	0	1	0	1
Перевантаження функцій	1	0	1	1
Значення параметрів за замовчуванням	0	1	1	1
Сумарний коефіцієнт	2	2	5	6

Проведення аналізу засобів розробки мовою програмування C++

Розглянемо такі засоби розробки: Microsoft Visual Studio, Eclipse CDT, NetBeans, QT Creator, CLion.

Microsoft Visual Studio – інтегроване середовище розробки C++, яке дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, зокрема з підтримкою технології Windows Forms. Підходить для створення веб-сайтів, веб-додатків та веб-служб для всіх підтримуваних платформ: Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone, .NET Compact Framework та Silverlight.

Переваги:

- є безкоштовна версія Visual Studio Community;
- вбудований інтерфейс командного рядка;
- API для підключення додаткових інструментів налаштування;
- повний набір інструментів розробника для створення та клонування Git-репозиторіїв, управління гілками та вирішення конфліктів злиття прямо в інтегрованому середовищі розробки C++;
- великий набір доповнень розширення базової функціональності.

Недоліки:

- висока вартість платних версій Professional та Enterprise (від 45 доларів на місяць);
- високі вимоги до «заліза»;
- немає версії для Linux.

Eclipse CDT (C/C++ Development Tooling) – безкоштовне інтегроване середовище розробки модульних кросплатформових додатків, яке набуло великої популярності серед розробників C++. Це середовище має всі потрібні інструменти, безкоштовне та працює під різними операційними системами: Windows/Linux/macOS.

Переваги:

- безкоштовне використання;
- автозавершення та інші можливості, що допомагають швидше писати код ;
- великий набір плагінів для розширення функціональності;
- розвинена спільнота розробників, докладна документація;
- вбудоване юніт-тестування, оптимізація тестів;
- налаштовний графічний інтерфейс.

Недоліки:

- повільний запуск, споживання великого обсягу пам’яті;
- проблеми зі зворотною сумісністю;
- можливі конфлікти плагінів;
- потреба завантажувати eclipse cdt з офіційного сайту.

NetBeans – безкоштовне інтегроване середовище розробки. Дозволяє створювати програми мовами програмування Java, Python, PHP, JavaScript, C, C++ та іншими. У цьому інтегрованому середовищі програмування C++ є дистрибутива для платформ Microsoft Windows, Linux, FreeBSD, macOS, OpenSolaris і Solaris, а для решти платформ є можливість зібрати NetBeans самостійно з вихідних джерел.

Переваги:

- безкоштовне інтегроване середовище розробки C++;
- кросплатформна підтримка;
- великий вибір плагінів;
- автозавершення коду, інструменти для рефакторингу;
- розвинена спільнота розробників.

Недоліки:

- повільний запуск;
- проблеми з власним кешем під час складання готових програм;
- для роботи потрібно jdk;
- потреба завантажувати netbeans з офіційного сайту.

Qt Creator – інтегроване середовище розробки C++, доступне для Windows, Linux і macOS. Пропонує повний набір інструментів розробника, призначених для створення та розгортання додатків.

Переваги:

- підтримує налаштування, профілювання, автозавершення коду та рефакторинг;
- можливість компіляції проєктів щодо різних ОС.

Недоліки:

- велика вага додатків;
- не завжди працює автозавершення коду;

- вартість платної версії висока;
- потрібна реєстрація для завантаження безкоштовної версії.

CLion – кросплатформне середовище програмування мовою C++ від компанії JetBrains. Містить сучасні стандарти C++, libc++ і Boost. Підтримує також інші мови програмування Kotlin, Python, Rust і т. д. – «З коробки» або за допомогою плагінів.

Переваги:
- зручні механізми налаштування програм;
- автозавершення коду ;
- підтримка VIM.

Недоліки:
- немає безкоштовної версії – лише демо на 30 днів;
- немає вбудованого компілятора;
- виникають проблеми зі встановленням компілятора.

Приклад аналізу засобів програмування наведено у таблиці 3.3. Рейтинг засобів розробки мовою програмування C++ показано на рисунку 3.1.

Таблиця 3.3 – Порівняльний аналіз засобів програмування

IDE	Eclipse CDT	NetBeans	CLion	MS Visual Studio	Qt Creator
Debugger	1	1	1	1	1
Конструктор GUI	1	1	0	1	1
Інтегрований ланцюжок інструментів	1	1	1	1	1
Профайлер	1	0	0	1	1
Покриття коду	1	0	0	1	0
Автозаповнення	1	1	1	1	1
Статичний аналіз коду	1	0	1	1	1
Дизайн на основі GUI	1	1	0	1	1
Браузер класу	1	1	1	1	1
Сумарний коефіцієнт	9	9	5	9	8

Рисунок 3.1 – Рейтинг засобів розробки мовою програмування C++

3.3 Рекомендації щодо викладення розділу «Розробка інтерфейсу програмного застосунку»

Інтерфейс користувача (ІК) – це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (з комп'ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача – це такий різновид інтерфейсів, в якому з одного боку – людина, з іншого – машина (пристрій, програмне забезпечення) [1]. За дефініцією Національного банку стандартизованих науково-технічних термінів, інтерфейс користувача – це комплекс апаратних і програмних засобів, що забезпечує взаємодію користувача з комп'ютером [2, 3].

Процес розробки ІК має бути ітераційним, оскільки робочий інтерфейс неможливо одержати без взаємодії з попередніми етапами розробки. Критерієм для завершення ітераційної розробки є факт задоволення вимог користувача і відповідність продукту вимогам ТЗ. При використанні ітераційного процесу початкові варіанти розробки допомагають створити прототипи, які можуть використовуватись в наступних ітераціях. На сьогодні найбільш ефективним підходом розроблення ІК, орієнтованим на користувача, є ітераційний підхід з такими етапами:

1. Розробка програмного застосунку, орієнтованого на використання графічного інтерфейсу та забезпечення функціональних характеристик. Такий

підхід враховує календарні терміни для кожного з етапів процесу розробки ІК, визначає основні ризики, об’єднує методи, встановлює цілі та критерії оцінювання графічного інтерфейсу.

2. На етапі визначення вимог до розробки вирішуються такі задачі: опис цільової аудиторії, постановка задач, оцінювання поточного рівня реалізації, аналіз можливостей, аналіз тенденцій розвитку.

3. Розробка концептуального проєкта, що є сукупністю високорівневих описів, абстракцій та оглядової інформації, яка дає розробникам та кінцевим користувачам загальне уявлення про програмний застосунок

4. Проєкт користувацького інтерфейсу є сукупністю характеристик програмного застосунку, які сприймаються користувачем (вхідні дані, взаємодія користувача та системи, вихідні дані).

5. Створення прототипів та моделювання – засоби оцінювання проєкту на ранній стадії розробки.

6. Документування проєкту програмного застосунку, що описує дії користувачів, а також вигляд та поведінку в специфічних ситуаціях.

7. Написання програмного коду та тестування. Методи оцінювання пов’язані з залученням потенційних користувачів програмного застосунку. Загальні критерії досягнення цілей створення інтерфейсу користувача мають бути чітко визначені, зрозумілі й прийняті замовниками та розробниками; досягнення поставлених цілей може вимагати багатократних ітерацій.

8. На етапі впровадження здійснюється: оцінювання програмного засобу за участю користувачів, які не залучались до розробки, тестування, виконання задач, які не оцінювались або не були передбачені під час проєктування та розробки.

Графічний інтерфейс користувача (GUI – Graphical User Interface) – тип інтерфейсу користувача, в якому елементи інтерфейсу (меню, кнопки, значки, списки), відображені на екрані і виконані у вигляді графічних примітивів.

У графічному інтерфейсі користувача (ГІК) забезпечено доступ до активних екранних об’єктів (елементів інтерфейсу) та управління ними. Найчастіше елементи інтерфейсу в ГІК реалізовані на основі прототипів і відображають свої призначення та властивості, що полегшує розуміння й засвоєння ПЗ недосвідченими користувачами.

Основною характеристикою графічного інтерфейсу користувача є використання стандартних елементів інтерфейсу для управління даними або процесом обробки, відображення даних або результатів.

Характеристики ГІК [9]:

1) стандартизований для сумісності між застосунками;

2) користувачі можуть бачити графічні зображення і текст на екрані в тому вигляді, в якому вони подані в процесі кодування;

3) використовує концепцію інтерактивної взаємодії «об’єкт-дія»;

4) дозволяє переміщувати дані між програмними застосунками;

5) дозволяє інтерактивно управляти об’єктами та інформацією на екрані;

6) використовує стандартні елементи інтерфейсу (меню та діалогові вікна, кнопки, поля, списки);

7) забезпечує візуальне відображення інформації та об’єктів (іконки і вікна, текст);

8) забезпечує візуальний зворотний зв’язок в процесі виконання користувачами дій та задач;

9) надає візуальне відображення дій користувача/системи, а також режимів (меню, палітри);

10) використовує графічні елементи керування, які дозволяють користувачам робити вибір та вводити дані;

11) надає користувачам можливість налагодити та персоналізувати інтерфейс та інтерактивні дії;

12) забезпечує гнучкість переходу від клавіатури до іншого пристрою введення даних.

Одним з недоліків ГІК є його орієнтація на використання застосунку з візуальним інтерфейсом. При роботі з ГІК користувачі оперують із візуальною інформацією на екрані. Вони обирають застосунок, вказують використовуваний файл даних, організовують застосунки та дані на комп’ютерах у вигляді графічних структур.

ГІК передбачає проблемно-орієнтований підхід – перш ніж виконувати операції з файлами, користувачі мають запустити програмне забезпечення, яке працює з таким типом файлів. Користувачі працюють з застосунками у «вікнах», які мають панель меню з варіантами маршрутів, відображуваними спадними меню. Такі меню містять варіанти дій і маршрутів, пов’язаних з об’єктами у вікні, або застосунком, який надає саме вікно.

Будь-який вдало розроблений ІК має знижувати навантаження на пам’ять користувача. ГІК здатні запропонувати наочні підказки та потрібні відомості, використовуючи комп’ютерні потужності для зберігання та пошуку інформації.

ГІК об'єднують три основних стилі інтерфейсу: командний рядок, меню та пряме маніпулювання.

Рядок статусу та поля інформації визначені як елементи вікна і використовуються для інформування користувачів про те, що відбувається всередині програми, тобто є візуальним зворотним зв’язком.

Кожен елемент керування, іконка, колір мають мати певне призначення. Семантичний зворотний зв’язок нагадує про змістовну характеристику об’єкта або виконання дії. Схожі елементи мають поводитись аналогічним чином.

Методи інтерактивної взаємодії ГІК передбачають використання клавіатури або вказівних пристроїв для переміщення, вибору та маніпулювання інформацією на екрані.

До методів інтерактивної взаємодії ГІК належать [9]:

1) можливість використання клавіатури або миші залежно від переваг користувача;
2) робота з різними типами меню;
3) два режими редагування: вставка або заміна;
4) технологія drag-and-drop;
5) буфер обміну.

Орієнтовані на користувача вдосконалення розробники будуть продовжувати впроваджувати в операційні системи та прикладне ПЗ зі зростанням конкуренції серед розробників.

Структурна схема – схема, яка визначає основні функціональні частини програмного застосунку, їх взаємозв'язки та призначення. Як структурні компоненти можна вносити підпрограми, модулі, бібліотеки, ресурси. Розробку структурної схеми програми зазвичай виконують методом покрокової деталізації.

Структурний підхід до програмування в тому вигляді, в якому він був сформульований в 70-х роках XX ст., пропонував здійснювати декомпозицію програм методом покрокової деталізації. Метод був спочатку запропонований Е. Дейкстрою, а потім доповнений Н. Віртом. Результатом декомпозиції є структурна схема програми, яка являє собою багаторівневу ієрархічну схему взаємодії підпрограм стосовно керування. Мінімально така схема відображає два рівні ієрархії, тобто показує загальну структуру програми. Проте цей же метод дозволяє отримати структурні схеми з великою кількістю рівнів. Метод покрокової деталізації реалізує низхідний підхід і базується на основних конструкціях структурного програмування. Він припускає покрокову розробку алгоритму. Кожен крок при цьому містить розкладання функції на підфункції. Так, на першому етапі описують вирішення поставленої задачі, виділяючи загальні підзадачі, на наступному аналогічно описують вирішення підзадач, формулюючи при цьому підзадачі наступного рівня. Таким чином, на кожному кроці відбувається уточнення функцій майбутнього програмного забезпечення. Процес продовжують, поки не доходять до підзадач, алгоритми вирішення яких очевидні. При декомпозиції програми методом покрокової деталізації, потрібно дотримуватися основного правила структурної декомпозиції, який з принципу вертикального керування: в першу чергу деталізувати процеси керування самого компонента декомпозиції, залишаючи уточнення операцій з даними насамкінець. Це пов'язано з тим, що пріоритетна деталізація процесів керування істотно спрощує структуру компонентів всіх рівнів ієрархії і дозволяє не відокремлювати процес ухвалення рішення від його виконання: так, визначивши умову вибору деякої альтернативи, зразу ж викликають модуль, що її реалізовує.

Деталізація операцій зі структурами дозволить відкласти уточнення їх специфікацій і забезпечить можливість модифікації цих структур за рахунок скорочення кількості модулів, залежних від цих даних.

Окрім цього, доцільно дотримуватися нижченаведених рекомендацій:

- не відокремлювати операції ініціалізації та завершення від відповідної обробки, оскільки модулі ініціалізації та завершення мають погану зв'язаність (тимчасову) і сильне зчеплення (щодо управління);
- не проєктувати спеціалізованих або універсальних модулів, оскільки проєктування надмірно спеціальних модулів збільшує їх кількість, а проєктування надмірно універсальних модулів підвищує їх складність;
- уникати дублювання дій в різних модулях, оскільки при їх зміні виправлення доведеться вносити до всіх фрагментів програми, де вони

виконуються, в цьому випадку доцільно просто реалізувати ці дії в окремому модулі;

− групувати повідомлення про помилки в один модуль за типом бібліотеки ресурсів, тоді легше узгоджувати формулювання, уникнути дублювання повідомлень, а також перекласти повідомлення іншою мовою

3.4 Рекомендації щодо викладення розділу «Розробка програмного застосунку»

3.4.1 Розробка класу програмного застосунку

У мові програмування С++ поняття «клас» лежить в основі об'єктно-орієнтованого програмування (ООП). Об'єктно-орієнтоване програмування виникло як удосконалення процедурно-орієнтованого програмування.

В основі ООП лежать поняття «об'єкт» та «клас». У мові програмування об'єкт – це змінна типу «клас». Клас описує дані та методи (функції), які будуть використовуватись об'єктом цього класу. Кожен клас описує логічно-завершену одиницю програми. Інкапсуляція даних і методів їх обробки в межах класу дозволяє покращити структурованість програмних систем. Це, в свою чергу, зменшує ризик виникнення «невидимих» логічних помилок. Використання спадковості та поліморфізму у класах дозволяє уникнути повторюваності програмного коду та зручно впорядкувати складні виклики методів, що об'єднані між собою в ланцюг.

Клас визначає формат (опис) деяких даних та роботу (поведінку) над цими даними. З оголошення класу можна отримати різну кількість об'єктів класу (змінних типу «клас»). Кожен об'єкт класу визначається конкретним (на даний момент) значенням внутрішніх даних, який називають станом об'єкта.

У класі оголошуються дані (внутрішні змінні, властивості) та методи (функції), що оперують цими даними (виконують роботу над даними).

На рисунку 3.2 показано базову структуру класу С++.

Рисунок 3.2 – Базова структура класу С++

У середовищі CLR підтримуються два види класів:

- некеровані (unmanaged) класи. Для виділення пам'яті під об'єкти таких класів можуть бути використані некеровані покажчики (*) та операція new;

- керовані (managed) класи. Для виділення пам’яті в таких класах можуть бути використані керовані покажчики (^) та операція gcnew.

Ця тема висвітлює особливості використання некерованих (*) класів.

Загальна форма оголошення unmanaged-класу

Ключове слово «class». У найпростішому випадку (без спадковості) загальна форма оголошення unmanaged-класу має нижчеописаний вигляд.

Ім’я_класу – безпосередньо ім’я типу даних «клас». Це ім’я використовується при створенні об’єктів класу.

Ключове слово class повідомляє про те, що оголошується новий клас (тип даних). Всередині класу оголошуються члени класу: дані та методи. Ключове слово private визначає члени класу, що мають бути приховані (закриті) від зовнішніх методів, оголошених за межами класу, а також об’єктів класу. Члени даних, оголошені з ключовим словом private, доступні тільки іншим членам цього класу.

Ключове слово public визначає дані (змінні) та методи (функції) класу, що є загальнодоступними.

Ключове слово protected визначає захищені дані та методи класу, які є:

- доступними для методів успадкованих класів, якщо даний клас є батьківським щодо інших класів;
- недоступними для інших методів, що реалізовані в інших частинах програми;
- недоступними для об’єктів (екземплярів) класу.

В межах опису класу секції (розділи) private, protected, public можуть слідувати в будь-якому порядку та в будь-якій кількості.

Термін «інкапсуляція даних» означає, що для членів класу (даних та методів) можна встановлювати ступінь доступності з інших частин програмного коду (інших методів, об’єктів класу). Таким чином, виникає поняття приховування даних (методів) у класі.

Інкапсуляція забезпечує покращення надійності зберігання даних у класі шляхом введення додаткових методів перевірки цих даних на допустимі значення. Як правило, доступ до прихованих даних в класі відбувається не напряму, а через виклики спеціальних методів доступу чи властивостей. Безпосередньо дані розміщуються у прихованій частині класу, а методи доступу до цих даних розміщуються у загальнодоступній частині класу.

Класична мова C++ дозволяє встановлювати доступ до членів класу з допомогою трьох специфікаторів: private, protected, public.

Члени класу можуть мати три основні типи доступу, які визначаються відповідними ключовими словами:

- private – члени класу є прихованими. Це означає, що доступ до них мають тільки методи, що оголошені в класі. Private-члени класу є недоступними з породжених класів та об’єктів цього класу;
- protected – члени класу є захищеними. Це означає, що доступ до protected-членів мають методи класу, «дружні функції» та методи успадкованих класів. Protected-члени класу є недоступними для об’єктів цього класу;

- public – члени класу є відкритими (доступними) для усіх методів та об’єктів з усіх інших частин програмного коду (рисунок 3.3).

```
class ClassName {
   private:
   // приховані дані та методи (функції)
   // ...
                              public:
   // відкриті дані та методи
   // ...
   protected:
   // захищені дані та методи
   // ...
};
```

Рисунок 3.3 – Структура класу з приватними і публічними ідентифікаторами

Клас може бути оголошений без методів. Такі класи містять тільки дані. Щоб доступитися до даних у класі, що не містить методів, потрібно дані оголосити в розділі public. Класи без методів майже не застосовуються. Якщо оголосити дані в розділі private, то доступитись до членів-даних класу буде неможливо.

Пустий клас доцільно створювати у випадку, коли під час створення великого проєкту потрібно протестувати його ранню (початкову) версію, в якій деякі класи ще не розроблені і не реалізовані. Замість реального класу вказується пустий клас – заглушка. Цей клас розроблений під потреби майбутнього проєкту таким чином, щоб компілятор не видавав повідомлення про помилку і можна було протестувати вже написану частину коду. У цьому випадку ім’я пустого класу вибирається таким, яким воно має бути в майбутньому проєкті.

Основні переваги класів виявляються за наявності методів – членів класу. За допомогою методів доступу до даних в класах можна зручно:

- ініціалізувати дані початковими значеннями;
- здійснювати перевірку на коректність внутрішніх даних при їх задаванні;
- реалізовувати різні варіанти перетворення (конвертування) даних;
- реалізовувати організацію виділення пам’яті для складених типів даних, що реалізовані у класах;
- виконувати різноманітні види обчислень над даними.

Реалізацію методів класу можна оголошувати в класі та за межами класу.

Наприклад. У наведеному нижче програмному коді оголошується клас CustomTime (рисунок 3.4). Клас містить два методи SetTime1() та SetTime2(), які виконують ту саму роботу: встановлюють новий час. Тіло (реалізація) методу SetTime1() описується в класі CustomTime. Реалізація методу SetTime2()

описується за межами класу (рисунок 3.5). В класі описується тільки прототип (декларація) методу SetTime2().

```
class CustomTime{
  private:
  int sec; // секунда
  int min; // хвилина
  int hour; // година
  public:
  // реалізація методу всередині класу
  void SetTime1(int h, int m, int s)   {
    // реалізація
    hour = h;
    min = m;
    sec = s;
  }
  // тільки декларація (прототип) методу (метод SetTime2 є членом класу)
  void SetTime2(int h, int m, int s); // реалізація методу - за межами класу
};
```

Рисунок 3.4 – Опис класу CustomTime

```
void CustomTime::SetTime2(int h, int m, int s)
{
  hour = h;
  min = m;
  sec = s;
}
```

Рисунок 3.5 – Реалізація методу SetTime2() за межами класу

Тіло методу, що описується за межами класу, може бути описане в іншому модулі. Як правило, у системі Microsoft Visual Studio цей модуль має розширення *.cpp. Сам же клас описується в модулі з розширенням *.h.

Програмний код методів – членів класу можна описувати в самому класі та за його межами. Якщо потрібно описати код методу, що є членом класу, то для цього використовується оператор розширення області видимості «::» . Оператор «::» визначає ім’я члена класу разом з іменем класу, в якому він реалізований.

Оголошення класу – це опис формату типу даних. Цей тип даних «клас» описує дані та методи, що оперують цими даними. Опис класу – це тільки опис (оголошення). У цьому випадку пам’ять для класу не виділяється.

Пам‘ять виділяється тільки тоді, коли клас використовується для створення об‘єкта. Цей процес ще називають створенням екземпляра класу, що являє собою фізичну сутність класу.

Оголошення об’єкта класу (екземпляра) нічим не відрізняється від оголошення змінної: «ім’я_класу» «ім’я_об’єкта» (Sample sample).

За допомогою імені ім’я_об’єкта можна здійснити доступ до загальнодоступних (public) членів класу. Це здійснюється з допомогою символу «.» (крапка).

Можливий варіант оголошення покажчика на клас. Якщо це unmanaged-клас, то оголошення має вигляд: «Ім’я_класу» * «ім’я_об’єкта» (Sample* sample).

Після такого оголошення потрібно виділяти пам’ять для об’єкта класу з допомогою оператора new. Доступ до даних за покажчиком здійснюється з допомогою комбінації символів «->» так само як і у випадку зі структурами.

Об’єкт класу – це змінна типу «клас». При оголошенні об’єкта класу виділяється пам’ять для цього об’єкта (змінної). З об’єкта можна мати доступ тільки до public-членів класу. Це можна здійснювати з допомогою символу «.» (крапка) або доступу за покажчиком «->» (рисунок 3.6).

```
Worker w1;
w1.SetName(«Johnson J.»); // виклик public-методу
Worker * w2; // покажчик на клас Worker
w2 = new Worker(); // динамічне виділення пам’яті для класу
w2->SetName(«Jackson M.»); // виклик public-методу з класу за покажчиком
```

Рисунок 3.6 – Використання класів і їх методів

За замовчуванням, члени класу мають доступ private. Тому, при оголошенні класу, якщо потрібно вказати private-члени класу, це слово можна опустити. Як правило, private-члени класу є закритими. Це є основна перевага інкапсуляції. Щоб змінювати значення private-членів класу, використовують методи класу, що оголошені в public-секції. У цих методах можна змінювати значення private-членів. Такий підхід використовується для забезпечення надійності збереження даних у private-членах. У public-методах, що мають доступ до private-членів, можна реалізувати додаткові перевірки на допустимість значень.

Конструктор класу – це спеціальний метод (функція) класу. Конструктор викликається при створенні об’єкта класу. Як правило, конструктор використовується для:

- виділення пам’яті для об’єкта класу;
- початкової ініціалізації внутрішніх даних класу.

Конструктор призначений для формування екземпляра об’єкта класу. Ім’я конструктора класу збігається з іменем класу. Виклик конструктора здійснюється при створенні об’єкта класу. Конструктор класу викликається компілятором. Конструктор може мати будь-яку кількість параметрів. Також конструктор може бути без параметрів (конструктор за замовчуванням).

У разі створення об’єкта класу, що не містить жодного конструктора, буде викликатись так званий конструктор за замовчуванням (default constructor), який виділить пам’ять для об’єкта класу. Конструктор за замовчуванням – це

конструктор класу, що оголошується без параметрів. Якщо в класі немає явно визначеного конструктора, тоді у разі створення об’єкта автоматично викликається конструктор за замовчуванням. Конструктор за замовчуванням просто виділяє пам’ять для об’єкта класу, коли він оголошується, але в класі можна оголосити власний конструктор за замовчуванням. Такий конструктор називається: «явно заданий конструктор за замовчуванням». Кожен клас може мати тільки один конструктор за замовчуванням. Це пов’язано з тим, що у класі не може бути двох методів (функцій) з однаковою сигнатурою. Конструктор не може повертати значення (навіть значення void). Якщо в конструкторі написати повернення значення з допомогою оператора return, то компілятор видасть помилку. У випадку, коли в класі оголошено об’єкт іншого класу (підоб’єкт), першим викликається конструктори класу, що є підоб’єктом основного класу, наступним викликається конструктор основного класу. Якщо є два класи, один з яких базовий а інший – успадкований від базового, то в цьому випадку послідовність викликів така:

- спочатку викликається конструктор базового класу;
- наступним викликається конструктор успадкованого класу.

В класі може бути оголошено два конструктори, що приймають однакову кількість параметрів, однак за умови, що типи параметрів будуть відрізнятись. Для класу має виконуватись правило: у класі не може бути двох методів (функцій) з однаковою (такою, що збігається) сигнатурою.

Параметризований конструктор – це конструктор класу, що має параметри.

Деструктор, як і конструктор, належить до спеціальних функцій класу. Деструктор – це спеціальний метод, що викликається при видаленні об‘єкта. Як правило, деструктор використовується для звільнення пам‘яті, динамічно виділеної під внутрішні дані класу. Можуть бути й інші випадки застосування деструктора. Деструктор не може мати параметрів. У разі використання в класі між конструктором і деструктором можна визначити такі основні відмінності:

- деструкторам не можна передавати параметри, конструкторам можна;
- деструктори можуть бути віртуальні, конструктори – ні;
- коли оголошується клас, можна оголосити тільки один деструктор.

3.4.2 Діаграма класів

Діаграма класів – статичне подання структури моделі [9]. Відображає такі статичні (декларативні) елементи, як класи, типи даних, їх зміст та взаємодію. Діаграма класів може містити позначення для пакетів а також позначення для вкладених пакетів. Ще діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для подання статичної структури моделі системи в термінології класів об’єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об’єкти й кооперації, а також їхні відносини (зв’язки).

Асоціація показує, що об’єкти однієї сутності (класу) пов’язані з об’єктами іншої сутності. Якщо між двома класами визначена асоціація, то можна переміщатися від об’єктів одного класу до об’єктів іншого. Цілком припустимі

випадки, коли обидва кінці асоціації належать до одного і того ж класу. Це означає, що з об’єктом деякого класу дозволено зв’язати інші об’єкти з того ж класу. Асоціація, що зв’язує два класи, називається бінарною. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв’язують відразу кілька класів; вони називаються парними. Графічно асоціація зображується у вигляді лінії, що зв’язує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносин. Зазвичай ім’я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає потреба посилатися на них і відрізняти один від одного. Ім’я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, відіграє в ній деяку роль. Насправді – це «обличчя», яким клас, що знаходиться з одного боку асоціації, звернений до класу з іншого її боку. Ви можете явно позначити роль, яку клас відіграє в асоціації.

Агрегація – проста асоціація між двома класами, де один з класів має вищий ранг (ціле) і складається з декількох менших за рангом (частин). Становище такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об’єкт-ціле має кілька об’єктів-частин). Агрегація є окремим випадком асоціації і її зображують як просту асоціацію з незафарбованим ромбом з боку «цілого». За потреби можна вказувати кратність агрегації. Агрегація не накладає жорстких умов на термін існування об’єктів. Наприклад, «частини» можуть існувати тоді, коли «ціле» зникає. Графічно агрегація подана пустим ромбом на блоці «цілого», і лінією, яка проведена від цього ромба до класу, що міститься в ньому («частин»).

Композиція – більш строгий варіант агрегації. Відома також як агрегація за значенням. Композиція має жорстку залежність часу існування екземплярів класу контейнера та екземплярів класів, що містяться в ньому. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно подається як і агрегація, але з зафарбованим ромбиком.

3.4.3 Алгоритми роботи програмного застосунку і базових модулів

Алгоритм – це набір інструкцій, що описує порядок дій виконавця для досягнення певного результату [10]. Перед початком розробки алгоритму потрібно чітко уяснити, що програма має робити, яка інформація потрібна програмі (які дані є в наявності та які існують обмеження на ці дані), які обчислення та інші дії програма має виконати та яку інформацію видати користувачу як результат роботи. Після цього треба вирішити, як програма буде це робити, а саме: яким буде інтерфейс користувача, як має бути побудована програма, як будуть подані дані в програмі та які методи потрібно використати для обробки даних, щоб отримати остаточний результат. Алгоритм повинен має бути універсальним, тобто придатним для широкого класу вхідних даних [3].

При графічному поданні алгоритм зображають у вигляді послідовності зв’язаних між собою функціональних блоків, кожний з яких відповідає виконанню однієї або декількох дій.

Потоки даних або потоки керування на схемах показують лініями. На схемах потрібно уникати перехрещення ліній. Лінії завжди мають бути направлені до центра блока. Лінії на схемах потрібно розривати для того, щоб уникнути зайвих перехрещень, дуже довгих ліній, а також у випадках, коли схема розміщена на декількох сторінках.

Декілька виходів із блока можна показати кількома лініями до інших блоків або однією лінією, яку потім розгалужують на відповідну кількість ліній.

Окремі алгоритми розробляють для різних рівнів деталізації задачі. Кількість рівнів залежить від її розмірів та складності. Рівень деталізації має бути таким, щоб різні частини та взаємозв’язок між ними були зрозумілі в цілому.

Зазначені правила діють відповідно до ДСТУ ISO 5807:2016 «Символи та угоди щодо документації стосовно даних, програм та системних блок-схем, схем мережних програм та схем системних ресурсів» і розповсюджуються на схеми, які використовують для відображення різних видів задач обробки даних та способів їх рішень [4].

Окремі функції алгоритмів і програм, з урахуванням ступеня їх деталізації, відображаються у вигляді умовних графічних позначень – символів (див. таблицю 3.1).

Співвідношення геометричних розмірів символів – розмір a (див. табл. 3.1) має вибиратися з ряду 10, 15, 20 мм. Допускається збільшувати розмір a на число, кратне 5. Розмір b дорівнює 1,5a.

Примітка. При ручному виконанні схем алгоритмів і програм для символів 1–4, 7 допускається встановлювати b рівним 2a.

При виконанні умовних графічних позначень автоматизованим методом розміри геометричних елементів символів округляються до значень, обумовлених технічними можливостями використовуваних пристроїв.

Записи усередині символа або поруч з ним мають виконуватися машинописом або креслярським шрифтом. Записи мають бути короткими.

Скорочення слів або абревіатури, за стандартних і загальноприйнятих, мають бути розшифровані в нижній частині поля схеми або в документі, до якого ця схема належить. Записи усередині символа мають бути подані так, щоб їх можна було читати зліва направо, справа наліво і зверху вниз, незалежно від напрямку потоку.

У схемі символ присвоюється номер, що має міститися ліворуч над символом (наприклад, для посилання в інших частинах документації) (рисунок 3.7).

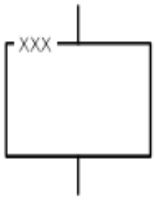


Рисунок 3.7 – Схематичне зображення блока

Символи нумеруються, починаючи з 1, порядок нумерації визначається напрямком лінії потоку, а також напрямком зліва направо.

3.5 Рекомендації щодо викладення розділу «Тестування програмного застосунку»

3.5.1 Методики тестування

Тестування програмного забезпечення – техніка контролю якості, що перевіряє відповідність між реальною та очікуваною поведінками програми завдяки кінцевому набору тестів, які вибираються певним чином.

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування (як процес своєчасного виявлення помилок та дефектів) не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому потрібно розрізняти тестування програмного застосунку й забезпечення його якості, до якого входять всі складові ділового процесу, а не тільки тестування.

Тестування – це структурована технічна діяльність [11]. План тестування (Test Plan) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінювання ризиків з варіантами їх дозволу .

Тестовий випадок (Test Case) – це артефакт, що описує сукупність кроків, конкретних умов і параметрів, потрібних для перевірки реалізації функції, що тестується, або її частини.

Основні атрибути Test Case:

ID (номер),

1) Name (ім'я),
2) Preconditions (умови і параметри),
3) Steps (кроки до відтворення),
4) Expectedresult (очікуваний результат),
5) Actualresult (фактичний результат),
6) Postconditions (постумови).

[3]

Тест дизайн (Test Design) – це етап процесу тестування ПЗ, на якому проєктуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріїв якості і цілей тестування.

Баг / Дефект Репорт (Bug Report) – це документ, що описує ситуацію або послідовність дій, що призвела до некоректної роботи об’єкта тестування, з причин і очікуваного результату.

Тестове Покриття (Test Coverage) – це одна з метрик оцінювання якості тестування, що являє собою щільність покриття тестами.

Специфікація Тест Кейсів (Test Case Specification) – це рівень деталізації опису тестових кроків і потрібного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття

Час Проходження Тест Кейса (Test Case Pass Time) – це час від початку проходження кроків тест кейса до отримання результату тесту.

Класифікація видів тестування за ознаками

За ступенем автоматизації:

- ручне тестування (manual testing) – це процес ручної перевірки програмного забезпечення на помилки. Тестувальник має відігравати роль користувача програми й використовувати властивості програми для знаходження помилок у роботі програми. Для професійного тестування тестувальник часто користується написаним планом тестування з варіантами тестування (test cases);
- автоматизоване тестування (automated testing) – частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно використовує програмні засоби для виконання тестів і перевірки результатів виконання, що допомагає скоротити час тестування та спростити його процес.

Існує два основних підходи до автоматизації тестування: тестування на рівні коду і GUI-тестування. До першого типу належить, зокрема, модульне тестування. До другого – імітація дій користувача за допомогою спеціальних тестових фреймворків. Найпоширенішою формою автоматизації є тестування додатків через графічний інтерфейс користувача. Популярність такого виду тестування пояснюється двома факторами: по-перше, додаток тестується тим же способом, яким його буде використовувати людина, по-друге, можна тестувати додаток, не маючи при цьому доступу до вихідного коду.

GUI-автоматизація розвивалася протягом 4 поколінь інструментів і технік:

1. Утиліти запису і відтворення (capture/playback tools) – записують дії тестувальника під час ручного тестування. Вони дозволяють виконувати тести без прямої участі людини протягом тривалого часу, значно збільшуючи продуктивність і усуваючи «безглузде» повторення одноманітних дій під час ручного тестування. У той же час, будь-яка мала зміна ПЗ, що тестується, вимагає перезапису ручних тестів. Тому це перше покоління інструментів неефективне і не масштабоване;
2. Сценарії (Scripting) – форма програмування мовами, спеціально розробленими для автоматизації тестування ПЗ – пом’якшує багато проблем capture/playback tools. Але розробкою займаються програмісти високого рівня, вони працюють окремо від тестувальників, які безпосередньо запускають тести. До того ж скрипти найбільше підходять для тестування GUI і не можуть бути

впровадженими, пакетними або взагалі будь-яким чином об’єднаними в систему. Нарешті, зміни в ПЗ, яке тестується, вимагають складних змін у відповідних скриптах, і підтримка все більш зростаючої бібліотеки тестувальних скриптів стає зрештою непереборним завданням.

Види тестування:

− Інсталяційне тестування. Інсталяційне тестування запевняє, що система встановлена правильно і коректно працює на апаратному забезпеченні конкретного клієнта.

- Тестування сумісності. Основною метою його є перевірка коректної роботи продукту в певному середовищі. Середовище може містити такі елементи: апаратна платформа, мережні пристрої, периферія (принтери, CD/DVD-приводи, веб-камери та ін.), операційна система (Unix, Windows, MacOS, …), бази даних (Oracle, MS SQL, MySQL, …), системне програмне забезпечення (веб-сервер, фаєрвол, антивірус, …), браузери (Internet Explorer, Firefox, Opera, Chrome, Safari).

− «Смоук» тестування (Smoke testing). Мінімальний набір тестів на явні помилки. Цей тест зазвичай виконується самим програмістом. Програму, що не пройшла такий тест, немає сенсу передавати на глибше тестування. Регресивне тестування виявляє помилки в уже протестованих ділянках початкового коду. Такі помилки – коли після внесення змін до програми перестає працювати те, що мало б працювати, – називають регресивними помилками.

- Регресивне тестування (за деякими джерелами) охоплює:
  1) new bug-fix – перевірка виправлення знайдених дефектів;
  2) old bug-fix – перевірка, що виявлені раніше й виправлені дефекти не відтворюються в системі знову;
  3) side-effect – перевірка того, що не порушилася працездатність працюючої раніше функціональності, якщо її код міг бути зачеплений під час виправлення деяких дефектів в іншій функціональності.

- Функціональне тестування. Перевіряє, чи реалізовані функціональні вимоги, тобто можливості ПЗ в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить продукт, які завдання вирішує. Функціональні вимоги охоплюють:
  1) функціональну придатність;
  2) точність;
  3) можливість до взаємодії;
  4) відповідність стандартам та правилам;
  5) захищеність.

- Модульне тестування. Належить до тестів, які перевіряють функціональність певного розділу коду, зазвичай на функціональному рівні. В об’єктно-орієнтованому середовищі, це, як правило, тестування на рівні класу, а мінімальні модульні тести містять у собі конструктори та деструктори.

Такі типи тестів зазвичай пишуться розробниками під час роботи над кодом (стиль «білої скриньки»), щоб впевнитись, що дана функція працює так, як очікувалося. Одна функція може мати кілька тестів, щоб переглянути всі випадки використання коду. Модульне тестування, саме по собі, не може

перевірити функціонування частини ПЗ, а використовується щоб гарантувати, що основні блоки ПЗ працюють незалежно один від одного.

Модульне тестування – це процес розробки ПЗ, що містить синхронізовані застосування широкого спектра для запобігання дефектів та для виявлення стратегій з метою зниження ризиків розробки ПЗ, часу та витрат. Воно виконується розробником ПЗ або інженером, у процесі будівельної фази життєвого циклу розробки ПЗ. Модульне тестування спрямоване на усунення помилок проєктування. Ця стратегія спрямована на підвищення якості одержуваного ПЗ до такого рівня, як того вимагає процес контролю якості.

Залежно від очікуваної організації розробки ПЗ модульне тестування може охоплювати статичний аналіз коду, аналіз потоку даних аналізу метрик, експертні оцінки коду, аналізу покриття коду та інші методи перевірки ПЗ.

Інтеграційне тестування. Інтеграційне тестування є типом тестування ПЗ, яке прагне перевірити інтерфейси між компонентами від програмного дизайну. Програмні компоненти можуть бути інтегровані як в рамках ітеративного підходу, так і всі разом.

Інтеграційне тестування працює над виявленням дефектів у інтерфейсах та взаємодії інтегрованих компонентів (модулів). Воно проводиться до тих пір, поки великі групи тестованих компонентів ПЗ, які відповідають потрібній архітектурі, починають працювати як система.

Рівні інтеграційного тестування:

1) компонентний інтеграційний рівень (Component Integration testing). Перевіряється взаємодія між компонентами системи після проведення компонентного тестування;

2) системний інтеграційний рівень (System Integration Testing). Перевіряється взаємодія між різними системами після проведення системного тестування.

Існує багато прийомів тестування ПЗ. Деякі з них перевершують інші, деякі можна використовувати разом з іншими для кращих результатів. Нижче наведено список основних прийомів тестування.

1. Ручне тестування – тести виконуються людьми з наперед складеними, або визначеними для кожного випробування тестовими даними.

2. Автоматизоване тестування – тести виконуються спеціальними інструментами або самостійними процесами і можуть повторюватися багато разів. Тестові дані попередньо вводяться або генеруються.

3. Регресивне тестування – тести, зазвичай автоматизовані, мають на меті виявлення негативного впливу змін в програмі на функції, що пройшли попередні перевірки.

4. Димове тестування – тести, спрямовані на швидку перевірку базової функціональності з метою виявлення, чи вартий тестування новий білд (версія програми чи її певного модуля).

5. Дослідницьке тестування – тести, що виконуються за відсутності специфікацій. Тестер розроблює власну систему тестування, яка базується на накопиченому ним досвіді та оцінює ризики, створюючи сценарії тестування.

3.5.2 Тестовий випадок

Тестовий випадок/ситуація (Тест Кейс /Test Case) – це артефакт, що описує сукупність кроків, конкретних умов і параметрів, потрібних для перевірки реалізації функції, що тестується чи її частини.

Під тест кейсом мається на увазі структура Action – Expected Result – Test Result

Тест кейси поділяються за очікуваним результатом на позитивні та негативні:

1. Позитивний тест кейс використовує лише коректні дані і перевіряє, чи правильно застосунок виконує функцію, що викликається;

2. Негативний тест кейс оперує як коректними, так і некоректними даними (мінімум 1 некоректний приклад) і ставить за мету перевірку виняткових ситуацій (спрацювання валідаторів), а також перевіряє, чи не виконується при спрацюванні валідатора функція, яка викликається застосунком.

Структура тестової ситуації:

- опис (Description) – відображає мету перевірки;
- передумова (PreConditions) – список дій, які приводять систему в стан, потрібний для основної перевірки;
- кроки (Steps) – метод виконання тесту, описаний покроково;
- очікуваний результат (Expected Result) – передбачувана поведінка системи після проходження по кроках;
- статус кейсу (Status) – вказується відповідно до того, чи відповідає фактичний результат очікуваному;
- Postcondition – список дій, що переводить систему в початковий стан (стан до проведення тесту) і є не обов'язковою частиною.

Обов'язкові вимоги до тест-кейсів

1. Відсутність залежності один від одного, оскільки тести можуть доповнюватися, змінюватися, втратити свою актуальність і бути видаленими. Крім того взаємозв'язок може ввести в оману, що робота продукту відповідає очікуванням.

2. Чіткі формулювання та висока ймовірність виявлення помилки.

3. Наявність детальної та не надлишкової інформації. Якщо перевірці підлягає процес авторизації, тест-кейс має містити логін та пароль.

4. Легка діагностика помилок. Виявлена помилка має бути очевидною.

5. Дослідження відповідної (безпосередньо тієї, що потрібно) галузі застосування, виконання потрібних дій.

Чек-лист (Check list) – це набір тестових ідей; простий, іноді поверхневий, лаконічно (але інформативно) описаний список ідей для перевірок. В більшості випадків чек-лист використовується тестувальником як «чернетка», щоб задокументувати всі ідеї та думки в голові, тому що дуже часто продукти достатньо складні, з великою кількістю функцій, а кількість перевірок може досягати сотень.

Тест-кейси розробляються з метою перевірки системи оптимальною кількістю тестових випадків.

Що потрібно зробити до написання тест-кейсів:

- уважно прочитати і дослідити вимоги до продукту;
- чітко розуміти мету розробки продукту;
- проаналізувати та розробити дизайн потрібних сценаріїв для максимально оптимального покриття системи тестовими випадками;
- визначити середовище для проведення тестів;
- розробити стратегію тестування та визначитись із видами тестування;
- з'ясувати функціональні та інші залежності у системі;
- передбачити поведінку системи на випадок некоректної поведінки користувача чи інші можливі некоректні умови.

Параметри якісних тест-кейсів

1. Результати легко визначаються. Очікуваний результат має бути чітким, помітним, зрозумілим та його можна легко інтерпретувати.

2. Тести виконуються швидко. Тестові випадки мають містити якомога меншу кількість кроків, якщо можливо, та швидко проводитися.

3. Зрозумілий кожному опис кроків. Тест-кейси мають бути описаними у достатньо зрозумілий та доступний спосіб, щоб інший виконавець зміг виконати його без вникання у вимоги до продукту.

3.5.3 Дефекти

Система відстеження помилок (bugtracking system) – програма, розроблена з метою допомоги розробникам ПЗ враховувати і контролювати помилки (баги), знайдені в програмах, побажання користувачів, а також стежити за процесом усунення цих помилок і виконання або невиконання побажань.

Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти:

1) номер (ідентифікатор об'єкта);
2) дата і час, коли був виявлений дефект;
3) хто повідомив про дефект;
4) хто відповідальний за усунення дефекту;
5) короткий опис проблеми (обов'язкове поле);
6) критичність дефекту (обов'язкове поле) і пріоритет рішення;
7) опис кроків для виявлення дефекту (відтворення неправильної поведінки програми) (обов'язкове поле);
8) очікуваний результат (обов'язкове поле);
9) фактичний результат (обов'язкове поле);
10) обговорення можливих рішень і їх наслідків;
11) статус дефекту;
12) версія продукту, в якій дефект був знайдений;
13) версія продукту, в якій дефект виправлений;
14) скріншот.

Критерій оцінювання проблеми, що характеризує вплив дефекту на працездатність програми:

- блокувальна (blocker) – розробка або використання продукту неможливо. Потрібне негайне виправлення проблеми;
- критична (critical) – серйозні проблеми (не працює критичний блок функціонала, спостерігаються серйозні помилки, пов'язані з втратою даних тощо);
- значна (major) – проблема, пов'язана з важливим функціоналом продукту;
- незначна (minor) – проблема, пов'язана з другорядним функціоналом продукту або є легкий обхідний шлях для цієї проблеми;
- тривіальна (trivial) – проблема косметичного рівня («недопрацьований» інтерфейс: друкарські помилки, різнобій з кольорами).

Пріоритет – це атрибут, який вказує на черговість виконання завдання або усунення дефекту:

- високий (high) – помилка має бути виправлена якомога швидше, тому що її наявність є критичною для проєкту;
- середній (medium) – помилка має бути виправлена, її наявність не є критичною, але вимагає обов'язкового рішення;
- низький (low) – помилка має бути виправлена. Її наявність не є критичною, і не вимагає термінового вирішення.

3.6 Керівництво користувача

«Керівництво користувача» – документ, призначений для організації ефективної роботи користувача з програмним продуктом. При викладенні матеріалу доцільно використовувати два розділи: інструкція (навчальний) і в довідковий. Інструкція орієнтована на допомогу новим користувачам, довідка призначена для досвідчених користувачів.

Матеріал інструкції має бути впорядкований відповідно до послідовності дії, починаючи з найпростіших і найбільш потрібних операцій, що з'являються в першу чергу, і закінчуючи більш складними, які з'являються пізніше. У довідковій секції подаються основні операції, впорядковані для зручності використання, наприклад, за алфавітом. Інформація подана в довідці є формальною, точною і детальною. Доцільно додавати ілюстрації у вигляді екранів з описом особливостей маніпуляцій на клавіатурі. Керівництво користувача має містити такі розділи:

- загальні відомості;
- опис застосування;
- вимоги до процедур.

У першому, вступному, розділі подається опис прикладної області і описуються основні функції програмного застосунку, а також умови його функціонування.

У другому розділі розглядаються основні функції, більш докладно описуються призначення програмного продукту. Значна увага приділяється опису умов експлуатації програмних засобів. Структура програмного продукту подається з описом ролі кожного компонента, функціональних можливостей з

зазначенням кількісних параметрів вхідних і вихідних потоків, часу реакції тощо. Окремо описуються дані з зазначенням кожного файлу та його призначення. Особлива увага приділяється опису потоків даних, що обробляються.

3.7 Висновки

В основній частині пояснювальної записки до КР викладаються структурні і графічні схеми, проводиться розробка базового класу з поданням діаграми класу, описом її основних полів і методів, проводиться розробка загальної блок-схеми алгоритму роботи застосунку і детальної блок-схеми алгоритму одного з методів з відповідними поясненнями.

Розробка програмного застосунку має бути логічно пов’язана з теоретичними відомостями з теми роботи, супроводжуватись ілюстративним матеріалом (схемами, діаграмами) або таблицями з обов’язковим посиланням на рисунки (таблиці) за текстом пояснювальної записки.

Розробка моделі є основною частиною пояснювальної записки за обсягом та змістом. При виконанні цієї частини КР потрібно дотримуватись обґрунтованого і аргументованого стилю викладення та врахувати можливі варіанти розв’язання поставленої задачі на підставі проведеного аналізу відомих розв’язків. Аргументація щодо тексту має містити відповідні схеми, діаграми, рисунки, таблиці тощо.

Для демонстрації працездатності розробленого застосунку потрібно провести тестування, результати тестування і проведений аналіз потрібно навести у роботі.

Висновки оформляють з нової пронумерованої сторінки. Вони є підсумковою частиною роботи. Потрібно відобразити послідовність виконання роботи, охарактеризувати розроблений застосунок. Також потрібно відобразити те, які навички здобуті під час виконання курсового проєкту.

Перелік посилань містить перелік літературних джерел, на які мають бути обов’язкові посилання в тексті пояснювальної записки. До переліку рекомендовано внести таких 5−9 літературних джерел з датою публікації не пізніше 5 років до дати захисту, як підручники, навчальні посібники, методичні матеріали, довідники, періодичні видання, інтернет-посилання. Література в загальний список записується в порядку посилання на неї в тексті.

Посилання на літературні джерела наводять в квадратних дужках […], вказуючи порядковий номер за списком.

Літературні джерела записують мовою оригіналу відповідно до ДСТУ ГОСТ 7.1:2006. У списку кожне джерело записують з абзацу, нумерують арабськими цифрами, починаючи з одиниці.

4 ЗАВДАННЯ НА КУРСОВУ РОБОТУ

У вступі потрібно визначити актуальність роботи, навести мету, об’єкт і предмет. Актуальність роботи характеризується потребою проведення розробки для підвищення ефективності певних процесів, оптимізації існуючих методів, покращення результатів. Мета роботи полягає у потребі розробки програмного застосунку для підвищення або покращення певних функціональних характеристик. Об’єктом дослідження є процес розробки програмного застосунку, предметом дослідження є методи і засоби розробки. Далі потрібно навести п’ять задач, що будуть розв’язані в процесі розробки. Задачі – це процеси, які потрібно виконати для досягнення цілей. Під задачами потрібно навести відповідні цілі виконання задач. Схему вступу наведено у додатку Г.

4.1 Аналіз сучасного стану питання та обґрунтування задачі

Потрібно провести аналіз відомих аналогів (програмний засіб такого ж призначення, що й досліджуваний, схожий з ним за технічною суттю та за результатом, що досягається при їх використанні). Для проведення аналізу потрібно визначити 3–4 аналоги.

Для кожного аналога потрібно навести його опис, що містить в собі:

- назву програмного застосунку або системи;
- поточну версію і дату останнього випуску;
- розробника або власника продукту;
- короткий опис;
- перелік переваг і недоліків;
- зображення користувацького інтерфейсу головного вікна.

Після наведеного опису аналогів потрібно провести їх аналіз. Для проведення аналізу потрібно визначити 4–5 функціональних характеристик, які будуть відображати ефективність використання аналогів і програмного застосунку, що розробляється. Функціональні характеристики визначають спроможність програмного застосунку виконувати в заданому середовищі функції з переліку, описують внутрішню роботу системи, її поведінку: обчислення даних, маніпулювання даними, обробку даних та інші специфічні функції, які має виконувати система.

Функції програмного застосунку є унікальними можливостями або атрибутами, які забезпечують його корисність для користувачів. При описі функцій продукту потрібно відповісти на такі питання: «Яка цільова аудиторія?» (оцінити групи користувачів, їх поведінку та проблеми), «У чому основна мета роботи програмного застосунку?» (визначити функції, що є ключовими).

Після проведення аналізу аналогів потрібно провести аналіз методів і засобів розробки. Аналіз методів розробки передбачає проведення ґрунтовного аналізу мов розробки застосунку. Опис методів і засобів розробки має містити: назву, поточну версію і дату останнього випуску, розробника або власника

продукту, короткий опис, зображення користувацького інтерфейсу або логотип. Для проведення аналізу потрібно визначити 4–5 функціональних характеристик, які будуть відображати ефективність використання методів і засобів для розробки.

4.2 Розробка програмного застосунку

Розробити структурну схеми застосунку, як показано на рисунку 4.1. Структурна схема має містити блоки, що відображають модулі, які використовуються при роботі програмного застосунку. До таких модулів відносять системні модулі, модулі компонентів, класів та методів (рис. 4.1).

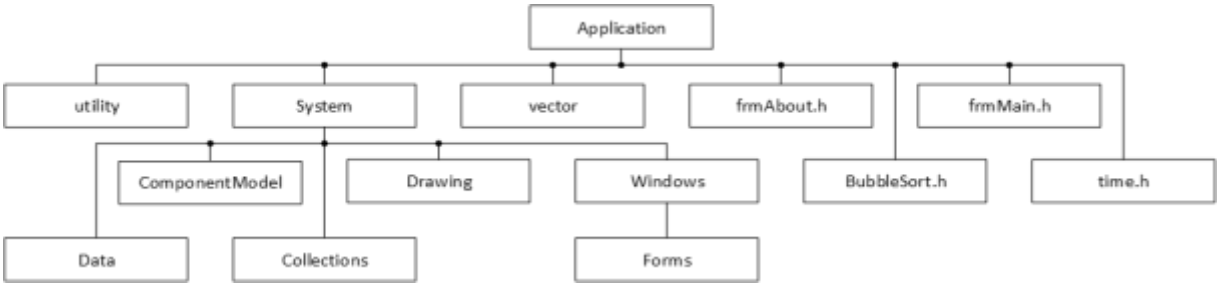


Рисунок 4.1 – Структурна схема за стосунку

4.3 Розробка графічного інтерфейсу програмного застосунку

Графічний інтерфейс має бути поданий мінімум двома вікнами (формами): головна форма (рисунок 4.2) і форма «Про за стосунок» (рисунок 4.3).

Головна форма має містити компоненти для управління процесом вирішення задач, що визначені у індивідуальному завданні. У формі «Про застосунок» має бути відображена загальна інформація про застосунок і розробника. На головній формі обов’язково мають міститися такі компоненти: головне меню, меню швидкого доступу, графічні компоненти управління і візуалізації, поле статусу.

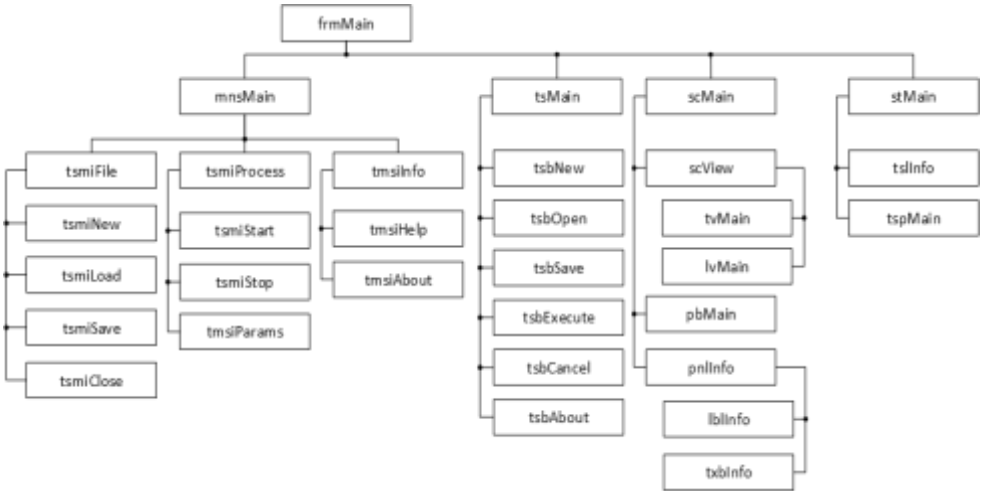


Рисунок 4.2 – Структурна схема головного вікна програми

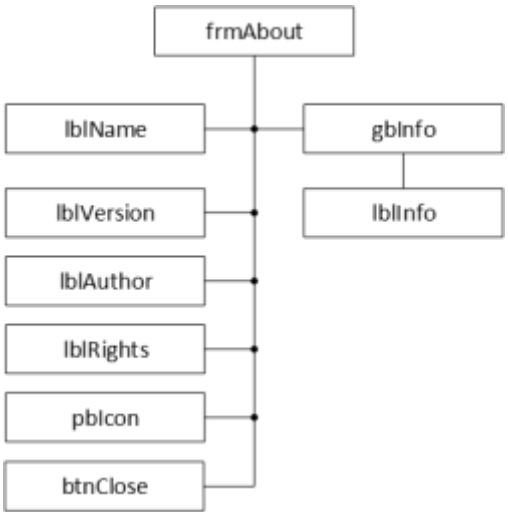


Рисунок 4.3 – Структурна схема вікна «Про застосунок»

Графічна схема головного вікна зображена на рисунку 4.4. Графічна схема відображає основні елементи управління та візуалізації. На схемі умовними блоками показано місце розташування візуальних компонентів.

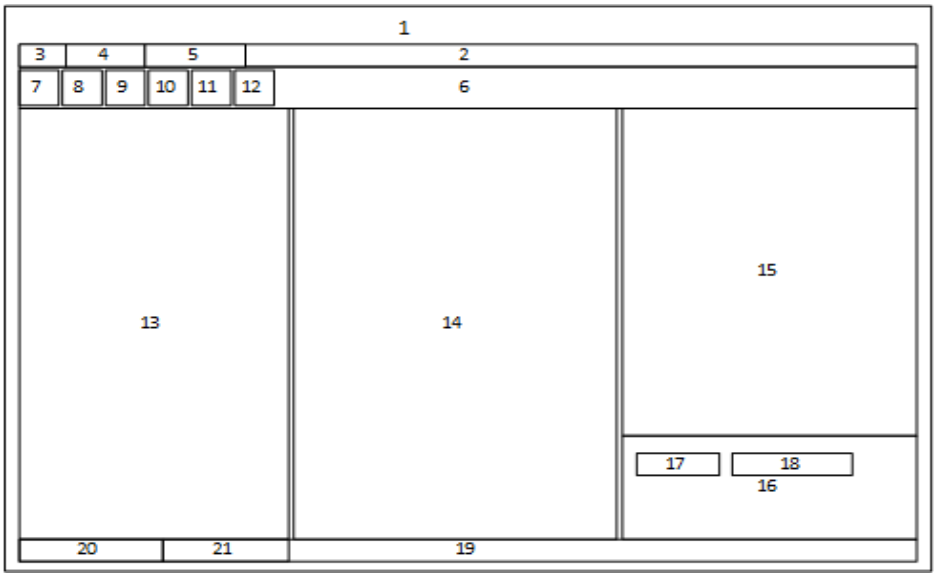


Рисунок 4.4 – Графічна схема інтерфейсу головного вікна

Елементи головно вікна:

1) frmMain – головна форма, на якій розташовується елемент управління і візуалізації;
2) mnsMain – головне меню, в якому містяться елементи управління роботою застосунку;
3) tsmiFile –низхідне меню управління процесом роботи з даними;
4) tsmiProcess –низхідне меню управління процесом обробки даних;
5) tsmiInfo –низхідне меню відображення пунктів інформація і «Про застосунок»;
6) tsMain – панель швидкого доступу;
7) tsbNew – кнопка створення нового проєкту;
8) tsbOpen – кнопка завантаження даних проєкту;

9) tsbSave – кнопка збереження даних проєкту;
10) tsbExecute – кнопка виходу з застосунку;
11) tsbCancel – кнопка запуску процесу обробки даних;
12) tsbAbout – кнопка зупинки процесу обробки даних;
13) tvMain – кнопка виведення на екран форми про застосунок;
14) lvMain – візуальний компонент дерево даних;
15) pbMain – візуальний компонент список;
16) pnlInfo – панель контейнер;
17) lblInfo – мітка, що містить текстову інформацію;
18) txbInfo – текстове поле введення, що визначає обсяг даних для обробки;
19) scMain – панель статусу, що розташована внизу форми;
20) tslInfo – текстове поле, що виводить додаткову інформацію про компоненти управління;
21) tspMain – панель прогресу.

На рисунку 4.5 зображено головне вікно застосунку.

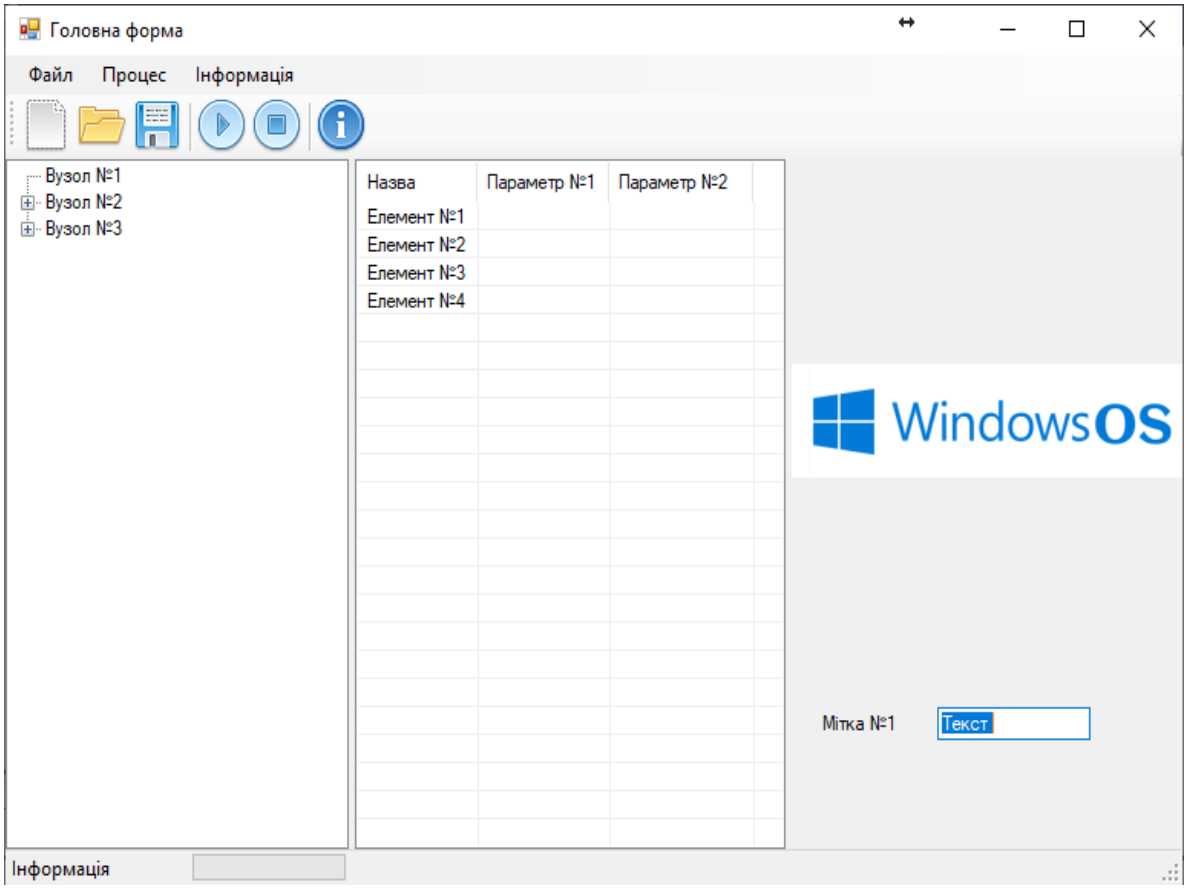


Рисунок 4.5 – Зображення головного вікна

Діалогове вікно «Про застосунок» містить інформацію про застосунок та розробника . На рисунку 4.6 зображено графічну схему вікна про програму.

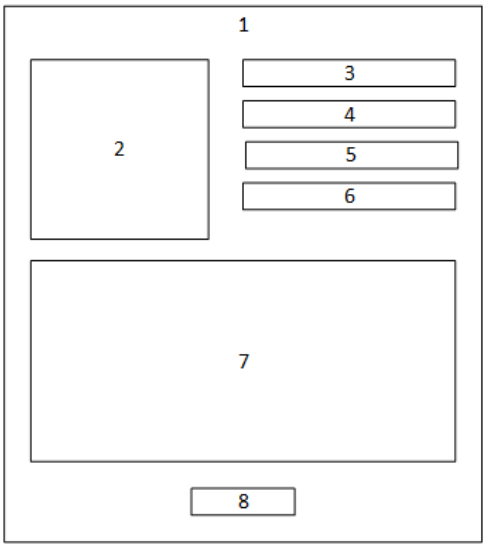


Рисунок 4.6 – Графічна схема інтерфейсу вікна «Про за стосунок»

Елементи вікна «Про застосунок» зображено на рисунку 4.7:

1) frmAbout – форма «Про застосунок»;
2) pbIcon – компонент із зображенням логотипу;
3) lblName – текстовий компонент відображення назви застосунку;
4) lblVersion – текстовий компонент відображення версії застосунку;
5) lblAuthor – текстовий компонент відображення автора застосунку;
6) lblRights – текстовий компонент відображення прав на застосунок;
7) lblInfo – текстовий компонент відображення інформації про застосунок;
8) btnClose – кнопка для закриття вікна.

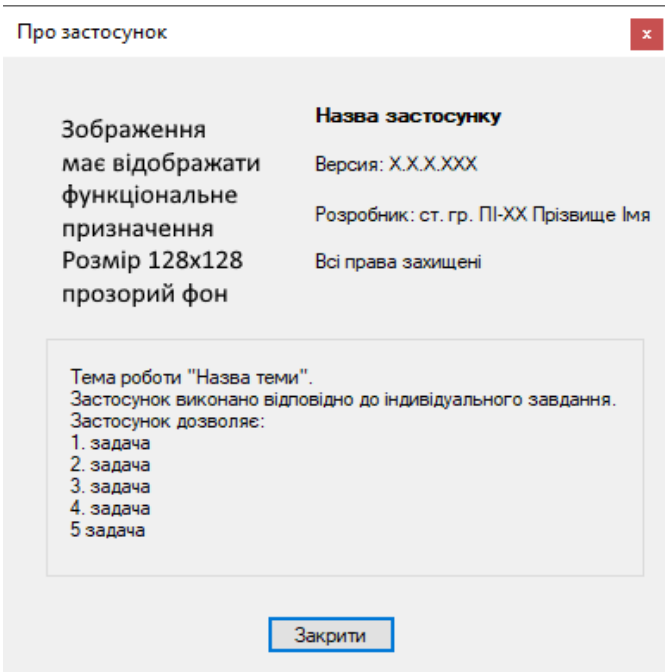


Рисунок 4.7 – Зображення вікна «Про застосунок»

4.4 Розробка моделі і алгоритму застосунку

Для проєктування, зміни та рефакторингу класів й інших типів потрібно додати до проєкту C++ діаграму класів. Для візуалізації різних частин коду у проєкті додайте до проєкту кілька діаграм класів.

Встановлення компонента «Конструктор класів». Якщо компонент «Конструктор класів» не встановлено, виконайте такі дії, щоб встановити його:

- відкрийте Visual Studio Installer з меню «Пуск» або вибравши у рядку меню Visual Studio Інструменти>Отримати інструменти та функції;
- відкриється Visual Studio Installer;
- виберіть вкладку «Окремі компоненти», а потім прокрутіть вниз до категорії «Інструменти для роботи з кодом»;
- виберіть «Конструктор класів» та натисніть Редагувати.

На рисунку 4.8 зображено вікно додавання компонента «Діаграма класів».

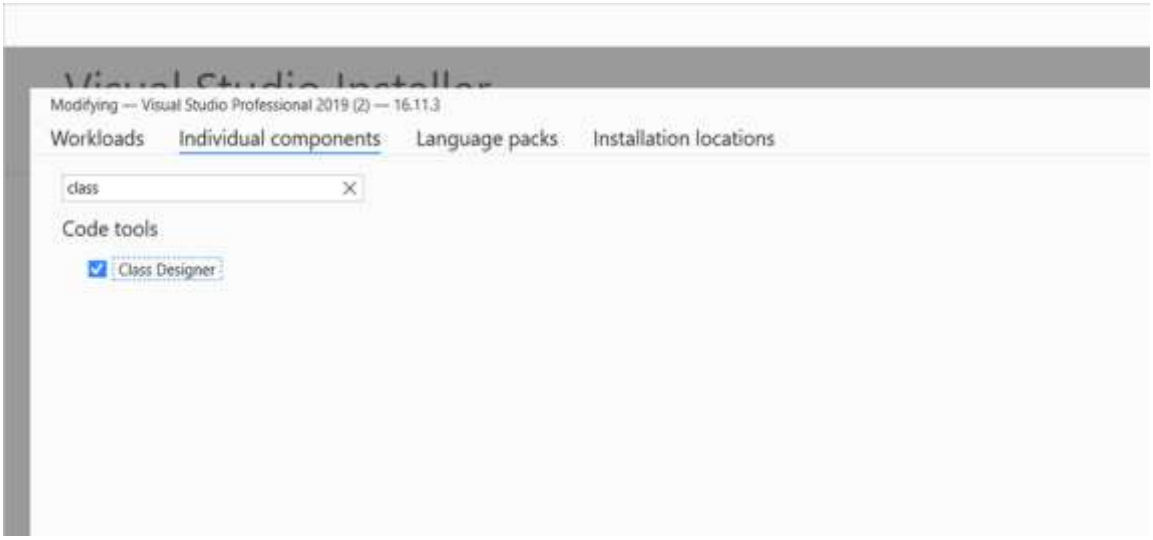


Рисунок 4.8 – Вікно додавання компонента «Діаграма класів»

4.4.1 Додавання порожньої схеми класів у проєкт

У браузері рішень клацніть вузол проєкту. Розгорніть вузол «Загальні», а потім у списку шаблонів виберіть пункт «Схема класів». «Схема класів» для проєктів Visual C++ шаблон знаходиться в категорії «Службові програми». Якщо шаблон «Схема класів»» відсутній, виконайте дії з інсталяції компонента «Конструктор класів» у Visual Studio. У «Конструкторі класів» відкриється схема класів, і в браузері рішень з’явиться файл із розширенням «.cd». Можна перетягувати фігури та лінії на діаграму з панелі елементів (рисунок 4.9). Щоб додати кілька схем класів, потрібно повторити кроки даної процедури.

У браузері рішень або подання класів клацніть проєкт правою кнопкою миші та виберіть Перегляд, а потім – Перегляд схеми класу. Буде створено автоматично заповнювану діаграму класів.

У роботі розроблено клас «Bubblesort» для сортування даних. Діаграму класу наведено на рисунку 4.10.

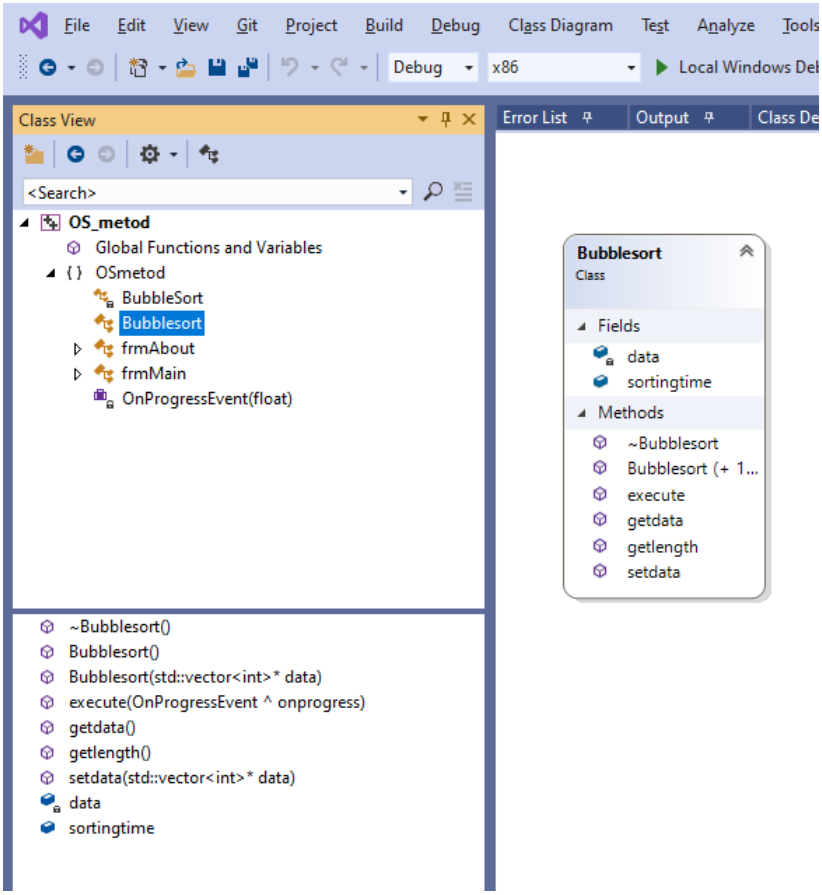


Рисунок 4.9 – Додавання класу у діаграму класів

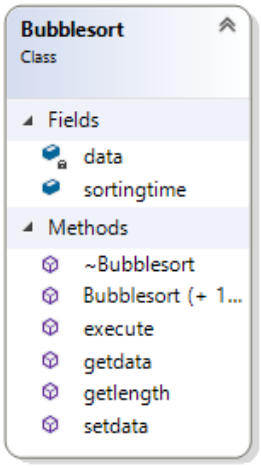


Рисунок 4.10 – Діаграма класу «Bubblesort»

Клас «Bubblesort» містить поля:

- data - вказівник на вектор даних типу integer (`std::vector<int>*`);
- sortingtime - час обробки даних (`time_t`).

Конструктор за замовчуванням ініціює дані вектору даних і час обробки (рисунок 4.11).

```
Bubblesort() {
        this->data = new std::vector<int>;
        sortingtime = 0;
    }
```

Рисунок 4.11 – Конструктор за замовчуванням

Конструктор з параметрами копіює дані у вектор даних (рисунок 4.12). Параметр конструктора – вказівник на вектор, кожний елемент якого є ціле типу «int»;

```
Bubblesort(std::vector<int>* data) : Bubblesort() {
    this->data->assign(data->begin(), data->end());
}
```

Рисунок 4.12 – Конструктор з параметрами

Деструктор виконує функцію очищення даних класу після видалення (рисунок 4.13).

```
~Bubblesort() {
        delete data;
    }
```

Рисунок 4.13 – Деструктор

Метод «getlands» повертає довжину вектора даних (рисунок 4.14).

```
int getlength() {
        int datalength = 0;
        if (data) datalength = data->size();
        return datalength;
    }
```

Рисунок 4.14 – Метод «getlength»

Метод «getdata» повертає вказівник на вектор даних (рисунок 4.15).

```
std::vector<int>* getdata() {
        return data;
    }
```

Рисунок 4.15 – Метод «getdata»

Метод «setdata» дата копіює дані з параметра у вектор даних (рис. 4.16). Параметр методу – вказівник на вектор, кожний елемент якого є ціле типу «int».

```
        void setdata(std::vector<int>* data) {
                    this->data->assign(data->begin(),         data-
>end());
```

Рисунок 4.16 – Метод «setdata»

Метод «execute» виконує сортування даних. Параметр методу – делеговане посилання «OnProgressEvent» на метод «doprogress», що дозволяє передавати метод як параметр (рисунок 4.17).

```
delegate void OnProgressEvent(float);
private: void doprogress(float value) {
      if (tspMain->Value != (int)value) tspMain->Value =
int(value);}
execute (gcnew OnProgressEvent(this, &frmMain::doprogress));
```

Рисунок 4.17 – Делеговане посилання «OnProgressEvent»

Реалізація методу «execute» наведена на рисунку 4.18. Параметром методу є делегат для асинхронного виклику методу doprogress, який належить класу frmMain. OnProgressEvent – це делегат (delegate) або подія, яка використовується для передачі методу, що буде викликано асинхронно. Функція дозволяє викликати метод doprogress у відповідь на події прогресу в процесі асинхронного виконання, що використовується для оновлення інтерфейсу користувача під час виконання тривалих операцій, а саме: візуального компонента tspMain, що розташований на головній формі застосунку.

```
bool execute(OnProgressEvent^ onprogress)
        {
            bool result = false;
            vector<int> arr(*data);
            try {
                sortingtime = time(NULL);
                int i, j, n;
                n = arr.size();
                bool swapped;
                for (i = 0; i < n - 1; i++) {
                    swapped = false;
                    for (j = 0; j < n - i - 1; j++) {
                        if (arr[j] > arr[j + 1]) {
                            swap(arr[j], arr[j + 1]);
                            swapped = true;
                        }
                    }
                    if (swapped == false)
                        break;
                    if (onprogress) onprogress(float(i + 1) * 100.0f / (float)n);
                };
                data->assign(arr.begin(), arr.end());
                result = true;
            }
            catch (...) {
                result = false;
            }
            sortingtime = difftime(time(NULL), sortingtime);
            return result;
        };
```

Рисунок 4.18 – Метод «execute»

4.4.2 Розробка блок-схеми алгоритму

Блок-схема роботи застосунку наведена на рисунку 4.19, вона відображає процес управління елементами керування. Відповідні елементи керування виконують функції, що реалізують виконання задач, відповідно до індивідуального завдання.

Основні етапи роботи

- Ініціалізація frmMain – створення головного вікна програми frmMain.
- Запуск програми – Application::Run(% frmMain) запускає головний цикл обробки повідомлень програми з вікном frmMain.
- Перевірка активного компонента інтерфейса – програма перевіряє яка дія активна та відповідно реагує (виконує такі методи: tsbNew_Click, tsbLoad_Click, tsbSave_Click, tsbStart_Click, tsbCancel_Click, tsbAbout_Click, tsbHelp_Click, tsbClose_Click). Процес перевірки проводиться у циклі до тих пір, поки не натиснута кнопка «tsbClose».
- Завершення роботи – програма завершує роботу, коли всі дії завершено.

Для вибору функції, що буде виконана, використовується активний компонент управління, ідентифікатор якого порівнюється з компонентами управління, що розташовані у вікні. Виконання методу активізується, якщо активний компонент збігається з компонентом управління для виконання певних задач. Після виконання задачі управління повертається до головної форми і очікується активізація наступного компонента управління.

Блок-схему алгоритму роботи функції сортування даних наведено на рисунку 4.20. Для сортування використовується вектор даних класу і обробник виняткових ситуацій Try-Catch.

Для визначення часу обробки використовуються блоки «6» і «21». Операції даних блоків дозволяють визначити різницю між початковим і кінцевим часом обробки даних. Сортування виконується за допомогою вкладених циклів блоків «8» і «10». Умовні блоки «11» і «12» дозволяють сформувати стан параметра «swapped» залежно від значень елементів масиву, що порівнюється у блоці «12».

Блок «16» дозволяє візуалізовувати процес сортування елементів масиву. Підсумкові дані передаються у вектор вхідних даних у блоці «17». Результат роботи алгоритму формуються у блоках «18», «20» і «3». У разі виняткової ситуації виконується блок «20», що дозволяє продовжити роботу застосунку без генерації помилки системи.

Необхідно звернути увагу на те, що блоки обов’язково мають бути нумерованими, нумерація має відповідати порядку операцій, що виконуються.

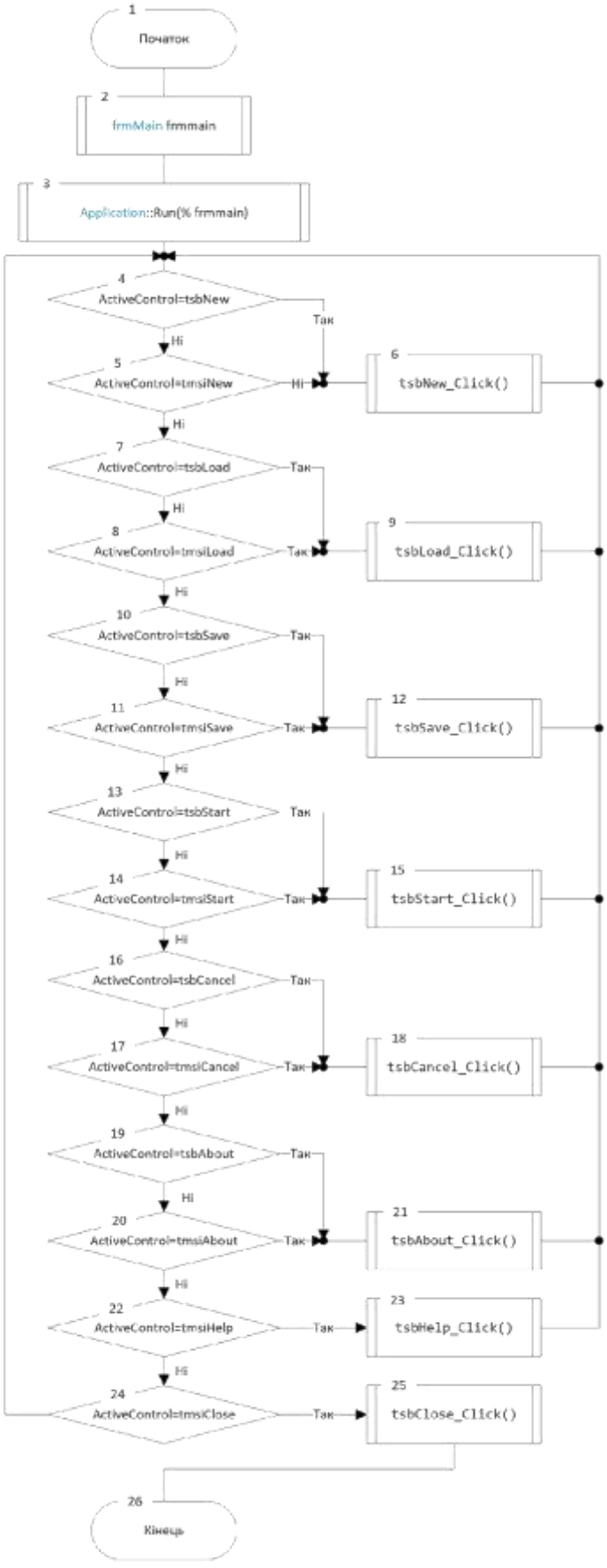


Рисунок 4.19 – Блок-схема роботи застосунку

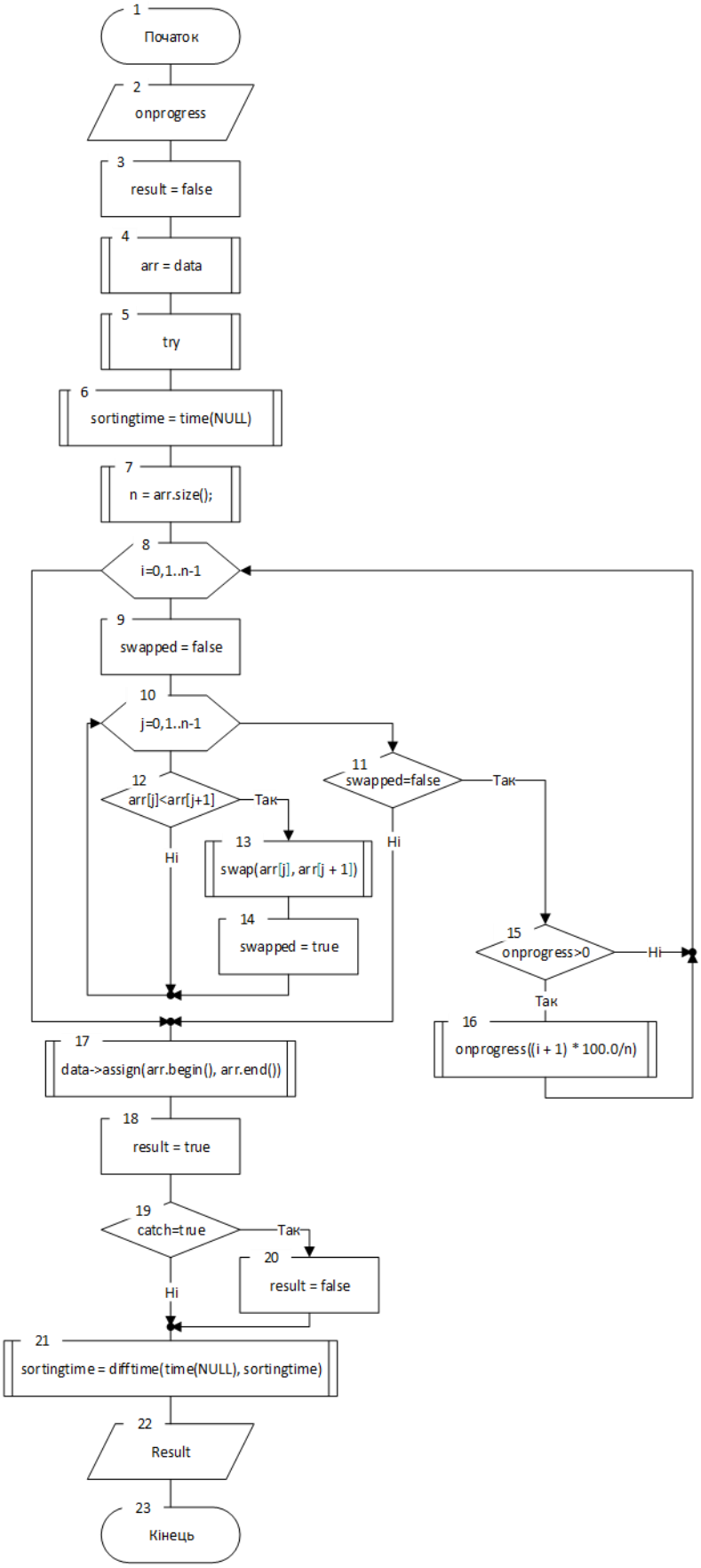


Рисунок 4.20 – Блок-схема алгоритму сортування даних

4.5 Тестування програмного за стосунку

Для проведення тестування використовується таблиця з тестовими випадками (таблиця 4.1), що відображають основні етапи роботи програмного застосунку, методи проведення тестування і очікувані результати

Таблиця 4.1 – Тестові випадки

Іденти-фікатор	Назва	Методика проведення тестування	Очікуваний результат	Результат
ТВ-1	Перевірка відображення елементів головного вікна	1. Запустити застосунок, відкривши файл «OS_method.exe	1. Відображається логотип та назва застосунку. 2. Відображається меню та панель інструментів.	Виконано
ТВ-2	Перевірка відображення діалогових вікон	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Натиснути «Про програму» 3. Натиснути «Виконати	1. Відкривається діалог «Про програму». 2 Виконується обробка даних	Виконано
ТВ-3	Перевірка алгоритму сортування даних	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Запустити сортовані дані та натиснути «Виконати» 3. Очікування завершення сортування	1. Сортовані дані мають збігатися з завантаженими даними.	Виконано
ТВ-4	Перевірка упинки процесу	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Натиснути «Виконати». 3. На панелі інструментів натиснути «Зупинити»	1. Сортування має припинитися. 2. В рядку статусу з'являється повідомлення «Сортування припинено користувачем» 3. Кнопка «Виконати» на панелі інструментів стає активною.	Виконано
ТВ-5	Перевірка перезапуску процесу	1. Повторити кроки 1–5, вказані в ТВ-4. 2. На панелі інструментів натиснути «Виконати»	1. Сортування завантажених даних.	Виконано
ТВ-6	Перевірка збереження даних у файл	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Натиснути «Виконати». 3. Після завершення сортування натиснути «Зберегти».	1. В обраній папці з'являється файл із сортованими даними з розширенням .txt	Виконано
ТВ-7	Перевірка кнопки «Допомога»	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Натиснути «Допомога».	1. Відкривається вкладка «Допомога» у обраному за за мовчуванням браузері.	Виконано
ТВ-8	Перевірка кнопки «Вихід»	1. Запустити застосунок, відкривши файл «OS_method.exe». 2. Натиснути «Вихід».	1. Програма успішно завершиться.	Виконано

Отже, за допомогою вищенаведених тестових випадків було проведено тестування графічного інтерфейсу та функціонала програмного застосунку, також підтверджено його працездатність і коректність виконання дій.

4.6 Інструкція користувача

Інструкція користувача має складатися з двох розділів: інструкція користування застосунком і довідкова інформація. Інструкція має описувати послідовності дій, починаючи з найпростіших і найбільш необхідних операцій, що виконуються в першу чергу, і закінчуючи більш складними, які з'являються пізніше. У довідковій секції подаються основні операції, впорядковані для зручності використання. Інструкція обов'язково має містити зображення інтерфейсу з описами.

Типовий зміст інструкції користувача програмного застосунку

1. Загальні відомості (охоплюють: назву програми, версію програми, інформацію про призначення програми, системні вимоги).
2. Встановлення програми (охоплюють: завантаження інсталяційного файлу, опис процесу встановлення крок за кроком, налаштування після встановлення).
3. Початок роботи (охоплює: запуск програми, опис процесу авторизація, огляд інтерфейсу користувача, основні функції).
4. Опис основних функцій та їх використання (охоплює: інструкції з виконання базових операцій, додаткові функції).
5. Опис додаткових можливостей програми (охоплює: налаштування додаткових параметрів, налаштування програми).
6. Вирішення проблем (охоплює: типові проблеми та їх вирішення, контактну інформацію служби підтримки).
7. Оновлення програми (охоплює: перевірку наявності оновлень, процедуру оновлення програми, список запитань (FAQ)).
8. Відповіді на типові запитання користувачів.
9. Додаткова інформація.

Список рекомендацій для написання інструкції

1. Чіткість і зрозумілість: використовуйте просту та зрозумілу мову, уникайте складних технічних термінів або пояснюйте їх.
2. Структура та форматування: розділіть текст на логічні розділи та підрозділи, використовуйте заголовки та підзаголовки для полегшення навігації, використовуйте списки, номери, таблиці та ілюстрації для покращення сприйняття інформації.
3. Інструкції крок за кроком: надання інструкцій у вигляді послідовних кроків з ілюстраціями або скріншотами, уникайте великих блоків тексту.
4. Ілюстрації та приклади: додайте скріншоти, діаграми та інші візуальні матеріали для пояснення складних моментів, використовуйте приклади, щоб показати, як виконувати конкретні завдання.
5. Консистентність: дотримуйтесь єдиного стилю написання та форматування по всій інструкції, використовуйте одні й ті ж терміни та назви для одних і тих самих функцій або об'єктів.
6. Оновлення та актуальність: регулярно оновлюйте інструкцію, щоб вона відповідала новим версіям програми, додавайте інформацію про нові функції або зміни у програмі.

Рекомендації допоможуть розробити ефективну та зручну для користувачів інструкцію, яка полегшить роботу з програмою та зменшить кількість запитів до служби підтримки.

5 ПОРЯДОК ЗАХИСТУ

Порядок захисту визначається робочим планом-графіком виконання курсового проєкту, підписаним викладачем, завідувачем кафедри та затверджується деканом факультету інформаційних технологій та комп'ютерної інженерії. Графік подається до деканату ФІТКІ за місяць до захисту КР.

Попередньо здійснюється:

− нормоконтроль КР;

− виправлення помилок, які стосуються оформлення та відповідності курсового проєкту нормативно-технічним документам;

− перевірка проєкту.

Будь-яке переписування матеріалів літературних джерел або електронних документів не допускається. За потреби можна навести певну кількість описових матеріалів у додатках.

Для захисту робіт кафедрою ПЗ призначається комісія у складі не менше двох викладачів кафедри, один з яких є керівником роботи.

До захисту допускаються роботи, які виконані в повному обсязі згідно з затвердженим індивідуальним завданням, а також перевірені керівником.

Здобувачі, які виконали навчальний план з дисципліни «Операційні системи», допускаються до складання екзамену незалежно від захисту КР.

Захист роботи проводиться публічно за встановленим графіком перед комісією, склад якої затверджується завідувачем кафедри, і здійснюється в такому порядку:

− здобувач робить доповідь з теми роботи обсягом до 5–10 хвилин (при цьому комісія може визначити іншу форму прийому роботи);

− після доповіді члени комісії ставлять запитання щодо теми проєкту;

− за результатами захисту комісія на закритому засіданні визначає оцінку, що потім оголошується здобувачу;

− якщо керівником проєкту чи членами комісії виявлено факт несамостійного виконання роботи, то здобувач до захисту курсової роботи не допускається.

Робота оцінюється за лінгвістичною системою на підставі критеріїв виконаної та захищеної КР з дисципліни ОС, затвердженої кафедрою ПЗ.

5.1 Обов'язки кафедри

Кафедра ПЗ несе повну відповідальність за хід курсового проєктування у навчальному процесі, в зв'язку з цим:

1) кафедра розроблює відповідні методичні вказівки (МВ) до виконання курсових проєктів на кафедрі, критерії оцінювання та іншу необхідну методичну документацію. Щороку переглядає їх на початку навчального року, повідомляє здобувачів через керівників на початку курсового проєктування;

2) заздалегідь формує та затверджує тематику курсових робіт на засіданні кафедри;

3) вирішує питання організації та проведення виконання курсових робіт відповідно до навчального плану;

4) регулярно заслуховує на засіданнях кафедри ПЗ питання організації, виконання, захисту курсових робіт; подає до деканату ФІТКІ інформацію про порушення здобувачами графіка курсового проєктування;

5) здає захищені роботи до архіву, де їх зберігають у встановленому порядку;

6) формує комісії для захисту курсових робіт й організовує їх роботу;

7) обговорює на засіданнях підсумки курсового проєктування і заходи підвищення якості;

8) питання затвердження тематики курсових робіт, організації, виконання, захисту, підсумків відображаються в протоколах засідання кафедри ПЗ.

5.2 Обов'язки деканату

Деканат здійснює загальний контроль за організацією та ходом курсового проєктування кафедрою, у зв'язку з чим:

1) своєчасно інформує кафедру про недопуск до проєктування здобувачів, які не виконали навчальний план з дисциплін, що є базовими для виконання відповідних курсових робіт;

2) разом з робочими планами розглядає, коригує і затверджує графік курсового проєктування, а у необхідних випадках розробляє і погоджує з керівником курсової роботи індивідуальний план роботи здобувача;

3) має інформацію щодо графіка захисту курсових робіт;

4) виносить на розгляд Вченої Ради факультету підсумки виконання курсових робіт та питання вдосконалення курсового проєктування.

5.3 Обов'язки керівника курсової роботи

Керівництво курсовим проєктування здійснюється найбільш кваліфікованими викладачами, які ведуть лекційні, практичні та лабораторні заняття з дисципліни. Керівник:

1) готує індивідуальні завдання на курсові роботи, у яких визначає коло питань, що мають висвітлюватися у курсовій роботі;

2) заздалегідь розробляє графік виконання курсових робіт і контролює його виконання кожним здобувачем;

3) контролює виконання здобувачем поетапного індивідуального графіка курсового проєктування;

4) організовує і проводить консультації з питань курсового проєктування;

5) веде журнал, відзначаючи не тільки стан успішності та відвідування консультацій за семестр, але й заповнює в кінці журналу за списком поетапний графік виконання КР;

6) проміжні етапи позначаються відповідною оцінкою;

7) перевіряє і візує до захисту (чи відхиляє) виконану, оформлену і підписану здобувачем курсову роботу;

8) після завершення графіка проєктування продовжує консультування, але переглядає і перевіряє вже повністю закінчену і оформлену курсову роботу.

Суттєвим порушенням вимог до виконання курсової роботи є відсутність:

1) аркуша індивідуального завдання з відповідними підписами;

2) графічної частини, специфікації, переліків, які мають, згідно з індивідуальним завданням, подаватись в додатках до проєкту.

5.4 Критерії оцінювання

Оцінювання якості виконання та захисту КР здійснюється за критеріями, наведеними в таблицях 5.1 та 5.2:

1) достовірність отриманих результатів у КР;
2) якість оформлення КР відповідно до чинних вимог;
3) якість подання результатів КР на захисті.

Таблиця 5.1 – Критерії оцінювання якості виконання та захисту КР

Орієнтовні критерії оцінювання	Кількість балів
Вагомість отриманих результатів: − точність та коректність завдань і висновків; − повнота обґрунтування актуальності обраної теми; − чіткість постановки мети та завдань, повнота їх реалізації; − правильність обраних методів і підходів до вирішення поставленого завдання; − дотримання науково-технічного стилю викладу.	до 50 балів
Якість оформлення відповідно до чинних вимог.	до 20 балів
Якість подання результатів КР на захисті (якість доповіді та презентації, а також відповідей на запитання).	до 30 балів
Максимальна оцінка	100 балів

Підсумкову оцінку захисту КР визначає комісія. Рішення комісії є остаточним.

Таблиця 5.2 – Критерії оцінювання знань, умінь і навичок здобувачів

Рівень компетентності	За бальною шкалою	За шкалою ЕКТС	Критерії оцінювання
1	2	3	4
IV Високий (творчий)	90–100	A	Виставляється, якщо при відповіді на питання виявлено всебічні, систематизовані, глибокі знання матеріалу, який виноситься на контроль, уміння вільно виконувати завдання, передбачені програмою дисципліни, знання основної і додаткової літератури на рівні творчого використання.
III Достатній (конструктивний)	82–89	B	Повні знання з питань і задач, що стоять перед здобувачом. Уміння викладати основні ідеї. Вміння профе-сійно відстоювати свою точку зору. Допускаються несут-тєві неточності у викладенні матеріалу та у відповідях.
	75–81	C	Достатньо повні знання з поставлених питань і задач. Вміння викладати основні ідеї. Здатність самостійно застосовувати вивчений матеріал на рівні стандартних си-туацій, наводити окремі власні приклади на підтвердження власних тверджень. Вміння доводити правильність своїх рішень. Несуттєві неточності у відповідях та деякі нераціональності при програмуванні задач.
II Середній (репролуктивний)	64–74	D	Здобувач може відтворити значну частину теоретичного матеріалу, виявляє знання та розуміння основних поло-жень, з допомогою викладача може аналізувати матеріал, робити висновки та розробляти програмні блоки. Пояснення неповні, нелаконічні, не завжди точні. Відповіді на питання неповні, містять неточності, при програмуванні застосовуються не найраціональніші рішення.
	60–63	E	Задовільні знання програмного матеріалу на рівні вищому за початковий. Здатність за допомогою викладача логічно відтворювати значну частину матеріалу. При відповіді на запитання виникають труднощі у деяких положеннях, відповіді неповні, програми пишуться нераціонально, не використовуються всі ефективні засоби програмування.
I Низький	35–59	FX	Теорію знає на рівні фрагментів, викладає матеріал уривчасто. Утруднюється в обґрунту-ванні рішень, на запитання викладача дає неправильні відповіді (40–60%), пояснення не до ладу. Самостійно, без допомоги викладача, не може сформувати алгоритм рішення задачі. Програми нераціональні неефективні, при програмуванні використовуються лише прості конструкції.
	0–34	F	Теорію знає на рівні фрагментів, викладає матеріал уривчасто. Утруднюється в обґрунту-ванні рішень, на запитання викладача дає неправильні відповіді (60–100%). Самостійно, без допомоги викладача, не може сформувати алгоритм рішення задачі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2017-07-01]. Вид. Київ, ДП «УкрНДНЦ». 2016. 26 с.

2. Про затвердження Вимог до оформлення дисертації: наказ Міністерства освіти і науки від 12.01.2017. № 40. *Офіційний вісник України*. 2017. № 20. 136 с.

3. ДСТУ 8302::2015 БІБЛІОГРАФІЧНЕ ПОСИЛАННЯ Загальні положення та правила складання. [Чинний від 2016-07-01]. Вид. Київ, ДП «УкрНДНЦ», 2016. 20 с.

4. Малахов В. Етичний кодекс ученого: аксіологія і прагматика. *Вісник Національної академії наук України*. 2009. № 5. С. 12–19. Бібліогр.: С. 19.

5. ДСТУ 6095:2009 (ГОСТ 7.88–2003, MOD) Правила скорочення заголовків і слів у заголовках публікацій. Чинний від 2009–07–01. Київ : Держспоживстандарт України, 2009. 10 с. (Національний стандарт України).

6. ДСТУ 7093:2009 (ГОСТ 7.11–2004, MOD; ISO 832:1994, MOD) Скорочення слів і словосполук, поданих іноземними європейськими мовами. На заміну ГОСТ 7.11–78 ; введ. в дію від 2010–04–01. Київ : Держспоживстандарт України, 2009. VI, 82 с. (Межгосударственный стандарт) (Національний стандарт України).

7. Бабенко Л. П., Лаврішева К. М. Основи програмної інженерії : навч. посіб. К. : Т-во «Знання», КОО, 2001. 269 с. (Вища освіта XXI століття).

8. Бабенко Л. П., Лаврішева К. М. Основи програмної інженерії : навч. посіб. К. : Т-во «Знання», КОО, 2001. 269 с. (Вища освіта XXI століття).

9. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.) . Addison Wesley Longman Inc. ISBN 0-201-30998-X.

10. Т. Кормен; Ч. Лейзерсон; Р. Рівест; К. Стайн (2009) [1990]. Вступ до алгоритмів (вид. 3rd). MIT Press і McGraw-Hill. ISBN 0-262-03384-4.

11. Тестування программного забезпечення: [Електрон. ресурс]. / Режим доступу:http://moodle.chdu.edu.ua/course/view.php?id=1021

ДОДАТКИ

ДОДАТОК А

ЗРАЗОК ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедри ПЗ, проф., д.т.н.

_____ О. Н. Романюк
(підпис)
«___» _____ 20__ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу з дисципліни «Операційні системи»
здобувачу групи _ПІ-___ ПІБ (повністю)
«_____»

Вихідні дані:
Розробити програмний додаток для ОС Windows, використовуючи мову програмування C++, IDE _____

1 Провести аналіз сучасного стану питання та обґрунтування задачі:
1.1 провести аналіз аналогів і їх основних характеристик;
1.2 обґрунтувати вибір методу та засобу розробки.
2 Розробити програмний застосунок:
2.1 розробити блок-схему і структурну схему застосунку;
2.2 розробити структурну схему інтерфейсу;
2.3 розробити графічний інтерфейс.
3 Розробити моделі програмного застосунку:
3.1 розробити функціональний клас;
3.2 розробити алгоритм роботи програмного застосунку і детальний алгоритм функціонального методу класу;
3.3 розробити діаграму функціонального класу.
4 Провести тестування програмного застосунку:
4.1 описати методики тестування;
4.2 розробити інструкцію тестування програмного застосунку;
4.3 розробити інструкцію користувача програмного застосунку.

Дата видачі «___» _____ 20__ р. Керівник _____

Завдання отримав _____

ДОДАТОК Б

ЗРАЗОК ТИТУЛЬНОГО АРКУША КУРСОВОЇ РОБОТИ

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки України
29.03.2012 N 384

Форма N Н-6.01

Міністерство освіти і науки України
Вінницький національний технічний університет
Кафедра програмного забезпечення

КУРСОВА РОБОТА
з дисципліни «Операційні системи»
на тему: « »

Здобувача 3-го курсу групи __ ПІ-___
спеціальності 121 «Інженерія програмного забезпечення»
_____

Керівник доцент кафедри ПЗ, Рейда О. М.

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Вінниця – 20___ рік

ДОДАТОК В

ТИПОВИЙ ЗМІСТ

ДОДАТОК Г

ПРИКЛАД ВСТУПУ

Актуальність теми зумовлена потребою підвищення ефективності роботи операційної системи у зв’язку з низьким рівнем оптимізації записів реєстру, що впливає на збільшення ресурсів і, як наслідок, зменшення швидкодії роботи ПК.

Мета роботи полягає у розробці програмного застосунку аналізу даних і структур реєстру для підвищення ефективності роботи операційної системи, оптимізації ресурсів.

Об’єкт дослідження – процес розробки програмного застосунку аналізу даних і структур реєстру.

Предмет дослідження – методи і засоби розробки програмного продукту, принципи об’єктно-орієнтованого програмування мови С++

В процесі розробки потрібно розв’язати такі задачі:

- обґрунтувати вибір методу та засобу розробки.
- розробити блок-схему і структурну схему застосунку;
- розробити структурну схему інтерфейсу;
- розробити графічний інтерфейс;
- розробити функціональний клас.

Потрібно досягти таких цілей:

- мова програмування і засіб розробки;
- блок-схема і структурна схема застосунку;
- структурна схема інтерфейсу;
- графічний інтерфейс;
- функціональний клас, що виконує основну задачу програмного застосунку.

Робота подана на __ сторінках, складається з __ розділів і містить __ рисунків та __ таблиці.

ДОДАТОК Д

ПРИКЛАДИ ОФОРМЛЕННЯ ДЖЕРЕЛ ПОСИЛАНЬ

Таблиця Д.1 – Приклади оформлення бібліографічного опису

Характеристика джерела	Приклад оформлення
1	2
Книги: Один автор	1. Дичківська О. О. Інноваційний менеджмент : конспект лекцій. Київ : ДІА, 2018. 82 с. 2. Бондаренко В. Г. Історія України. Львів, 2017. 153 с. 3. Лазор О. Я. Державне управління у сфері реалізації екологічної політики в Україні: організаційно-правові засади : монографія. Львів : Ліга-Прес, 2003. 542 с. 4. Ваш О. М. Етика : навч.-метод. посіб. Запоріжжя : ЗНУ, 2018. 104 с. 5. Гурманова Л. І. Релігієзнавство : навч. посіб. Вид. 2-ге, переробл. та допов. Київ : ЦУЛ, 2017. 193 с.
Два автори	1. Мартиненко З. Е., Макар І. В. Управління підприємством: теоретико-методичні засади : монографія. Харків : Щедра садиба плюс, 2017. 296 с. 2. Палеха В. І., Карпова П. В. Менеджмент організацій : навч. посіб. Запоріжжя : ЗНУ, 2015. 120 с. 3. Білоус С. І., Корнійчук В. П. Філософія освіти : навч.-метод. посіб. Переяслав-Хмельницький, 2016. 176 с. 4. Мороз І. С., Василенко Н. Ю. Маркетинг : конспект лекцій. Київ : Молодь, 2016. 102 с. 5. Вердіна С. А., Волков А. А. Контролінг : навч. посіб. Запоріжжя : ЗНУ, 2016. 131 с.
Три автори	1. Тарнавська Г. Я., Марценюк Н. С., Герасимова Т. М. Фінанси : навч. посіб. Львів : Магнолія 2006, 2017. 412 с. 2. Пустовенко В. В., Максименко І. Л., Яким А.С. Безпека життєдіяльності : монографія. Харків : ХНПУ, 2017. 348 с.
Чотири автори	1. Інновації : навч. посіб. / Гуревич Д. Т., Чекан О. С., Грибан О. М., Макарова В. В. Запоріжжя : ЗНУ, 2016. 389 с. 2. Вища математика : конспект лекцій / Ткачук Т.С. та ін. Київ, 2015. 82 с.
П’ять і більше авторів	1. Операційний менеджмент : підручник / С. М. Поплавська та ін. Київ : ЦУЛ, 2011. 267 с. 2. Охорона праці : навч. посіб. / О. І. Подольська та ін. Вид. 2-ге. Київ : ЦУЛ, 2017. 264 с. 3. Науково-практичний коментар Цивільного кодексу України : станом на 10 жовт. 2017 р. / К. І. Мягченко та ін. ; за заг. ред. І. М. Ливанова. Київ : ЦУЛ, 2017. 428 с.
Автор(и) та редактор(и)/ упорядники	1. Веретенко В. В. Міжнародний маркетинг : монографія / за заг. наук. ред. В. М. Марценюка. Київ, 2015. 374 с. 2. Бутенко М. П., Качур В. П., Петренко С. В. Психологія : навч. посіб. / за ред. М. П. Дутко. Київ : ЦУЛ, 2017. 332 с.

Продовження таблиці Д.1

1	2
Без автора	1. 30 років історичному факультету: історія та сьогодення (1986-2016) : ювіл. вип. / під заг. ред. В. В. Черепані. Запоріжжя : ЗНУ, 2016. 340 с. 2. Етнографія : конспект лекцій / за заг. ред. В. І. Гарапка; уклад. А. І. Гарапко. Київ : ЦУЛ, 2018. 320 с. 3. Міжнародні відносини : монографія / за ред. М. А. Березовського. Київ : ЦУЛ, 2016. 162 с. 4. Міжнародні економічні відносини : навч. посіб. / за ред.: П. О. Бедрія, О. О. Петренка. Одеса : ОНУ, 2015. 306 с. 5. Науково-практичний коментар Цивільного кодексу України / за заг. ред. Т. А. Тарнавського. Київ : ЦУЛ, 2016. 186 с. 6. Підготовка фахівців у ВНЗ в умовах реформування вищої освіти : матеріали Всеукр. наук.-практ. конф., м. Мукачево, 4-5 жовт. 2018 р. Мукачево : МДУ, 2018. 226 с. 7. Освіта в Україні: виклики модернізації : зб. наук. пр. / редкол.: П. М. Марценюк (відп. ред.) та ін. Київ : Ін-т всесвітньої історії НАН України, 2017. 319 с. 8. Товарознавство / упоряд. В. Олексик. Київ, 2014. 804 с.
Багатотомні видання	Енциклопедія рослин / редкол.: І. М. Деркач та ін. Київ : ЦУЛ, 2016. Т. 8. 812 с. Новицкий О. М. Сочинения : в 4 т. / ред. изд. : Н. Г. Мозговая, А. Г. Волков ; авт. вступ. ст. Н. Г. Мозговая. Киев ; Мелитополь: НПУ им. М. Драгоманова ; МГПУ им. Б. Хмельницкого, 2017. Т. 1. 382 с. Бюджетна система України: історія, стан та перспективи : у 3 т. / Акад. прав. наук України. Львів : Право, 2012. Т. 2 : Бюджетний менеджмент / заг. ред. Ю. П. Бубряка. 476 с. Кучеренко Н. П. Казначейська справа : в 6 т. Київ : Право, 2016. Т. 3 : Контроль у системі Державного казначейства. 432 с. Дендрофлора України. В 12 т. Т. 2. Дикорослі та культивовані дерева і кущі. Вип. 1. Покритонасінні / Л.І. Перхоменко. Київ : Наукова думка, 2012. 200 с.
Автореферати дисертацій	Петров О. Г. Музикотерапія : автореф. дис. на здобуття наук. ступеня канд. психол. наук : 12.00.06. Київ, 2009. 40 с.
Дисертації	Євдоченко О.О. Європейське бізнес-середовище в розвитку міжнародної економічної діяльності : дис...канд. екон. наук : 08.05.01 / Київський національний економічний ун-т. Київ, 2005. 235 с.

Продовження таблиці Д.1

1	2
Законодавчі та нормативні документи	1. Конституція України : офіц. текст. Київ : КМ, 2015. 98 с. 2. Конституція України : станом на 1 жовтня 2017 р. / Верховна Рада України. Київ : Право, 2017. 93 с. 3. Про вищу освіту : Закон України від 05.09.2016 р. № 2145-VIII. *Голос України*. 2016. 27 верес. (№ 178–179). С. 10–22. 4. Податковий кодекс України : Закон України від 19.05.2011 р. № 3393-VI. *Відомості Верховної Ради України*. 2011. № 48-49. Ст. 536. 5. Про освіту : Закон України від 01.07.2014 р. № 1556-VII. Дата оновлення: 28.09.2018. http://zakon2.rada.gov.ua/laws/show/1556-18 (дата звернення: 15.11.2018). 6. Питання соціального забезпечення : Постанова Кабінету Міністрів України від 28.12.2017 р. № 1060. *Офіційний вісник України*. 2018. № 5. С. 430–443. 7. Про інформування громадськості з питань євроатлантичної інтеграції України на 2019–2020 роки : Указ Президента України від 21.02.2018 р. № 43/2018. *Урядовий кур'єр*. 2018. 23 лют. (№ 35). С. 10. 8. Про затвердження Вимог до оформлення кандидатської дисертації : наказ Міністерства освіти і науки від 12.01.2018 р. № 50. *Офіційний вісник України*. 2018. № 25. С. 139–141. 9. Інструкція щодо порядку оформлення і ведення особових справ отримувачів усіх видів соціальної допомоги : затв. наказом М-ва. праці та соц. політики від 19.09.2006 р. № 156. *Баланс-бюджет*. 2006. 19 верес. (№ 18). С. 15–16.
Архівні документи	1. Лист Голови Спілки «Первоцвіт» Г. Ф. Петренка на ім'я Голови Ради Міністрів УРСР В. А. Поповича щодо реєстрації Статуту Спілки та сторінки Статуту. 14 грудня 1989 р. ЦДАГО України (Центр. держ. архів громад. об'єднань України). Ф. 1. Оп. 32. Спр. 2612. Арк. 63, 64 зв., 71.
Патенти	1. Зернозбиральний комбайн: пат. 25742 Україна: МПК6 С09К11/00, G01T1/28, G21H3/00. № 200701472; заявл. 12.02.07; опубл. 27.08.07, Бюл. № 13. 4 с. 2. Спосіб лікування гіперактивності у дітей: пат. 76509 Україна. № 2004042416; заявл. 01.04.2004; опубл. 01.08.2006, Бюл. № 8 (кн. 1). 120 с.
Препринти	1. Марченко М. І., Кополович А. Д., Яким Б. М. Про точність визначення радіоактивних відходів гамма-методами. Чорнобиль : Ін-т з проблем безпеки АЕС НАН України, 2006. 7, [1] с. (Препринт. НАН України, Ін-т проблем безпеки АЕС; 06-1). 2. Федорченко Б. А., Смотрич В. Н. Радиационное повреждения материалов нейтронами источника ННЦ ХФТИ / ANL USA с подкритической сборкой, управляемой ускорителем электронов. Харьков : ННЦ ХФТИ, 2006. 19 с.: ил., табл. (Препринт. НАН Украины, Нац. науч. центр «Харьк. физ.-техн. ин-т»; ХФТИ2006-4).

Продовження таблиці Д.1

1	2
Стандарти	1. ДСТУ 7152:2010. Видання. Оформлення публікацій у журналах і збірниках. [Чинний від 2010-02-18]. Вид. офіц. Київ, 2010. 16 с. (Інформація та документація). 2. ДСТУ ISO 6107-1:2004. Якість води. Словник термінів. Частина 1 (ISO 6107-1:1996, IDT). [Чинний від 2005-04-01]. Вид. офіц. Київ : Держспоживстандарт України, 2006. 181 с. 3. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та правила (ISO 4:1984, NEQ; ISO 832:1994, NEQ). [На заміну ДСТУ3582-97; чинний від 2013-08-22]. Вид. офіц. Київ : Мінекономрозвитку України, 2014. 15 с. (Інформація та документація).
Каталоги	1. Прокопенко И. П. Каталог растений для работ по экодизайну / Донец. ботан. сад НАН Украины. Донецк : Лебедь, 2005. 228 с. 2. Історична спадщина України : кат. вист. / Харків. держ. наук. б-ка ім. В. Г. Короленка; уклад.: Л. І. Петров, О. В. Олійник. Харків, 2000. 64 с. 3. Пам'ятки історії та мистецтва Закарпатської області : кат.-довід. / авт.- упоряд.: М. Петрик та ін.; Упр. культури Закарпат. облдержадмін., Закарпат. іст. музей. Ужгород, 2003. 160 с.
Бібліографічні покажчики	1. Боротьба зі злочинністю: нагальна проблема сучасності : бібліогр. покажч. Вип. 3 / уклад.: О. В. Куріпта, відп. за вип. Н. М. Щур; Запорізький національний університет. Запоріжжя, 2017. 60 с. 2. Іван Марченко : біобібліогр. покажч. / уклад. В. Петрик. Львів : Вид. центр ЛНУ ім. І. Франка, 2003. 356 с. (Українська біобібліографія ; ч. 9).

Аналітичний бібліографічний запис складова частина видання (глави, розділу, статті) розділовий знак «дві навскісні риски» («//») можна замінювати крапкою, а відомості про документ (його назву), виділяти шрифтом (наприклад, *курсивом)*.

Частина видання: книги	1. Петренко М. А. Международное право и роль Конституционного Суда Украины // Максим Петренко: право як буття вченого : зб. наук. пр. до 60- річчя проф. М. А. Петренко / упоряд. та відп. ред. Ю. О. Волошин. К., 2009. С. 477–493. 2. Якса А. П. Економічна політика держави. *Двадцять п'ять років з економічним правом* : вибрані праці. Харків, 2017. С. 205–212. 3. Корнійчук Т. О. Методи активізації навчально-пізнавальної діяльності. *Педагогіка* : навч. посіб. / за заг. ред. Т. О. Корнійчука. Київ, 2017. С. 195–197.
Частина видання матеріалів конференцій (тези доповіді)	1. Лалак Н. В. Шляхи підвищення мотивації молодших школярів до навчання // Анотовані результати науково-дослідної роботи Інституту педагогіки за 2011 рік : збірник тез повідомлень. Київ, 2012. С. 202–203. 2. Максименко Д. В. Методи оперативної діагностики виробничої діяльності підприємства // Зростання ролі бухгалтерського обліку в сучасній економіці : збірник тез та доповідей І Міжнародної науково-практичної конференції (м. Київ, 21 лютого 2013 р.) / відпов. за випуск Мельничук Б. В. Київ, 2013. С. 331–335. 3. Цехмістров І. І., Перець І. П. Про бюджет. Дослідження проблем в Україні очима молодих вчених : матеріали Міжнар. наук.-практ. конф., м. Запоріжжя, 3–4 берез. 2016 р. Запоріжжя, 2016. С. 50–53.

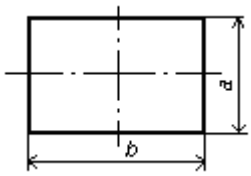
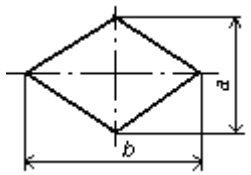
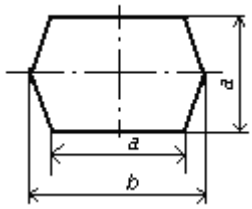
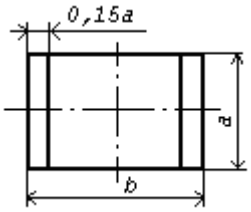
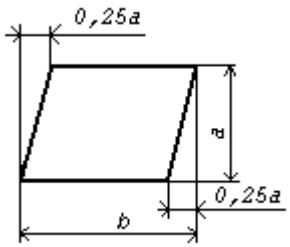
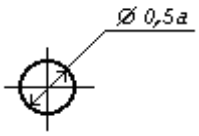
Кінець таблиці Д.1

1	2
Частина довідкового видання	1. Павлик І. М. Право інтелектуальної власності. *Великий енциклопедичний юридичний словник* / ред. Ю. С. Шемшученко. Київ, 2007. С. 683. 2. Дичківська І. М. Інноваційні педагогічні технології. *Основи педагогіки освіти :* словник термінів / за ред.: Т. О. Дмитрука, В. К. Колпакова. Київ, 2014. С. 54–55. 3. Попович Н. І. Початкова освіта. *Педагогічна енциклопедія*. Київ, 2003. Т. 5. С. 699.
Частина видання: продовжуваного видання	– Куцінко Т. О. Адміністративне законодавство України: реалії та перспективи формулювання. *Вісник Запорізького національного університету. Юридичні науки*. Запоріжжя, 2017. № 1. С. 36–46. – Безруков С. А., Хмельов А. А. Дослідження циліндричних оболонок. *Вісник Запорізького національного університету*. Фізико-математичні науки. Запоріжжя, 2015. № 3. С. 153–159. – Хорошилова С. А., Малафіїк Л. О., Хмельов А. А. Моделювання складеної конструкції за допомогою матриць типу Гріна. *Проблеми обчислювальної механіки і міцності конструкцій.* Дніпропетровськ, 2012. Вип. 19. С. 212–218.
Частина видання: періодичного видання (журналу, газети)	1. Кучеренко О. О. Конституційні права людини і громадянина. *Часопис Київського університету права.* 2007. № 4. С. 88–92. 2. Коваль Л., Коваль П. Переваги дистанційної роботи. *Урядовий кур'єр*. 2017. 1 листоп. (№ 205). С. 5. 3. Bletskan D. I., Glukhov K. E., Frolova V. V. Electronic structure of 2H-SnSe2. *Semiconductor Physics Quantum Electronics & Optoelectronics*. 2017. Vol. 18, No 2. P. 109–118.
Електронні ресурси	1. Україна очима дітей : фотовиставка. URL: http://www.kmu.gov.ua/control/uk/photogallery/gallery?galleryId=15725757& (дата звернення: 15.11.2017). 2. Хміль А. А. Функції державної служби за законодавством України. *Юридичний науковий електронний журнал*. 2017. № 5. С. 115–118. URL: http://lsej.org.ua/5_2017/32.pdf. 3. Хміль І. О. Шляхи подолання правового нігілізму в Україні. *Вісник Запорізького національного університету. Юридичні науки*. Запоріжжя, 2016. № 3. – С. 20–27. – URL:http://ebooks.znu.edu.ua/files/Fakhovivydannya/vznu/juridichni/VestUr20 15v3/5.pdf. (дата звернення: 15.11.2017). 4. Куцкір Я. С., Махно Б. А., Борислав С. Г. Трансформація науково-педагогічної системи України протягом 90-х років XX століття: період переходу до ринку. *Наука та інновації*. 2016. Т. 12, № 6. С. 6–14. DOI:https://doi.org/10.15407/scin12.06.006.

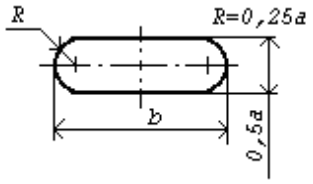
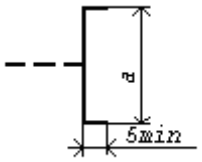
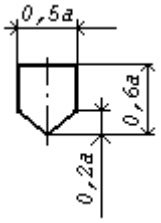
ДОДАТОК Е

УМОВНІ ПОЗНАЧЕННЯ НА БЛОК-СХЕМАХ

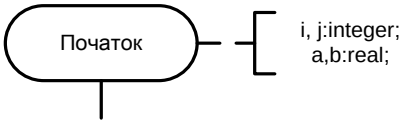
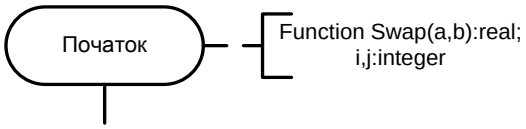
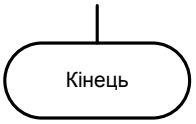
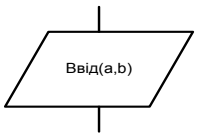
Таблиця Е.1 – Умовні позначення на блок-схемах

Тип	Зображення блока	Опис
1	2	3
1. Процес (операція)		Виконання операції або групи операцій, у результаті яких змінюється значення, форма подання або знаходження даних
2. Умовний блок		Вибір напрямку виконання алгоритму або програми залежно від деяких змінних умов
3. Циклічний блок		Виконання операцій, які змінюють команди, або групи команд, що змінюють програму
4. Визначений процес (процедурний блок)		Використання створених раніше і окремо описаних алгоритмів або програм
5. Введення-Виведення		Перетворення даних у форму, є найзручнішою для обробки (ввід) або відображення результатів обробки (введення / виведення)
6. З’єднувач		Зазначення зв’язку між перерваними лініями потоку, які з’єднують символи

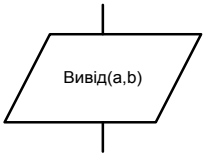
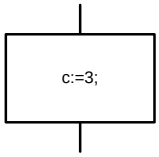
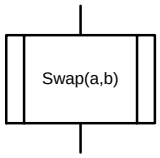
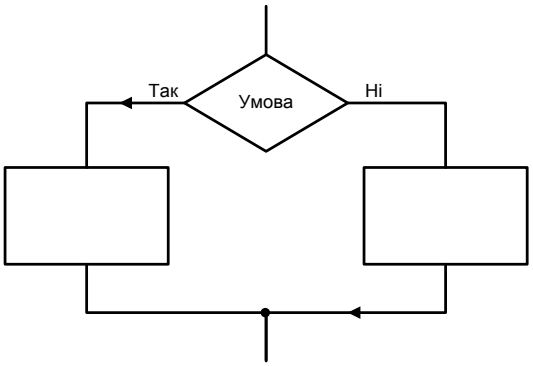
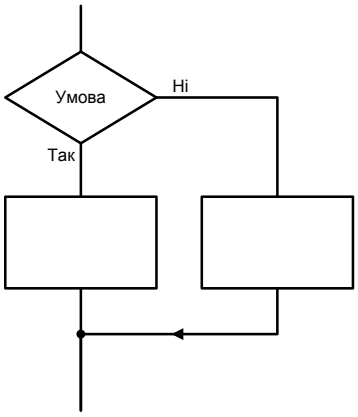
Кінець таблиці Е.1

1	2	3
7. Початок / Кінець		Початок, кінець, переривання процесу обробки даних або виконання програми
8. Коментар		Зв’язок між елементом схеми і поясненням
9. Міжсторінковий з’єднувач		Зазначення зв’язку між роз’єднаними частинами схем алгоритмів і програм, розташованих на різних аркушах

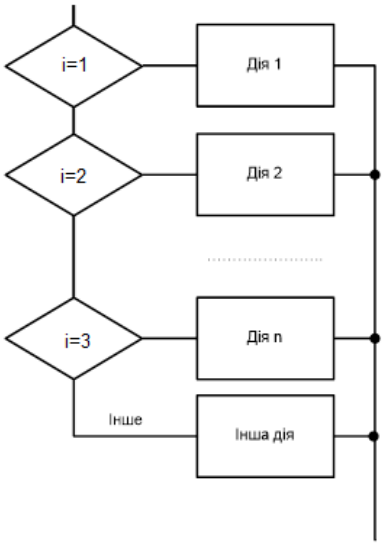
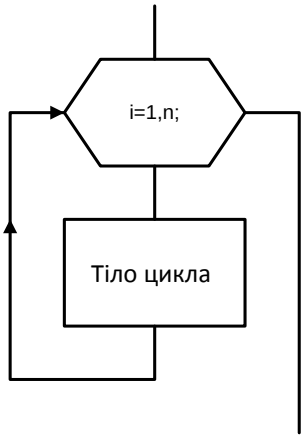
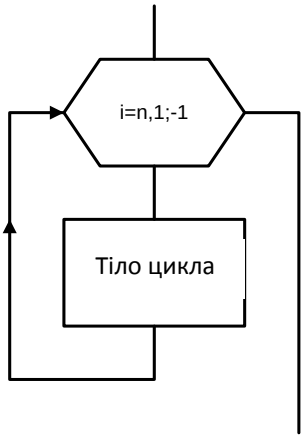
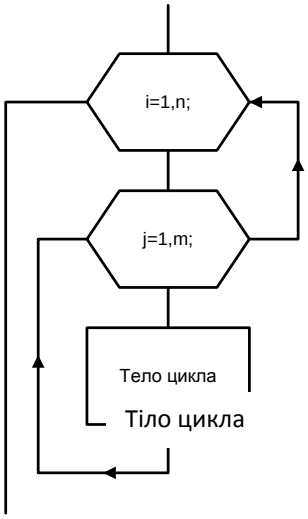
Таблиця Е.2 – Застосування символів у схемах алгоритмів

Тип операції	Зображення на блок-схемі
1	2
1. Початок програми	
2. Початок підпрограми	
3. Кінець (програми/ підпрограми)	
4. Ввід Read[ln](a,b)	

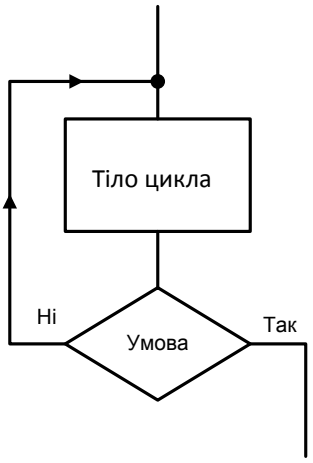
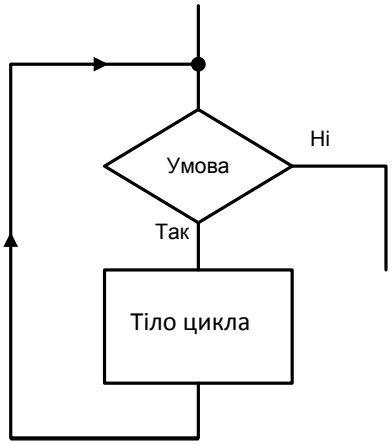
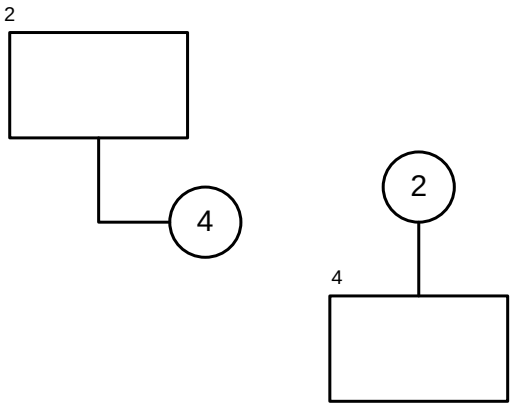
Продовження таблиці Е.2

1	2
5. Вивід Write[ln](a,b)	
6. Процес	
7. Підпрограма	Даний блок використовується у випадках виклику підпрограми «Процедури» або «Функція». Якщо підпрограма «Функція» не викликається, а повертає своє значення, то треба використовувати блок «Процес». (Наприклад, a:=Factor(N); тоді використовується блок «Процес»). 
4. Оператор If..Then..Else	Варіант1  Варіант2 

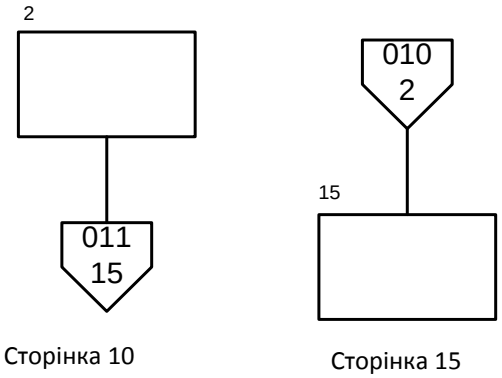
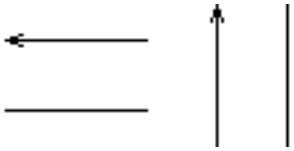
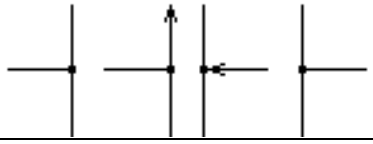
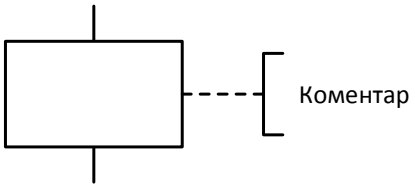
Продовження таблиці Е.2

<table>
<tr><th>1</th><th>2</th></tr>
<tr><td>5. Оператор
switch case..default</td><td> 
</td></tr>
<tr><td rowspan="2">6. Оператор циклу
For</td><td>Варіант1:
for (int i=1,i<=n,i++)
 
</td><td>Варіант2:
for (int i=n,i>0,i--)
 
</td></tr>
<tr><td colspan="2">Варіант3:
for (int i=1,i<=n,i++)
for (int j=1,j<=m,j++)
 
</td></tr>
</table>

Продовження таблиці Е.2

1	2
7. Оператор циклу з після умовою do…while	
8. Оператор циклу з передумовою while..do	
9. З'єднувач	При великій насиченості схеми символами окремі лінії потоку між віддаленими один від одного символами допускається обривати. При цьому наприкінці (початку) обриву має бути поміщений символ «З'єднувач». 

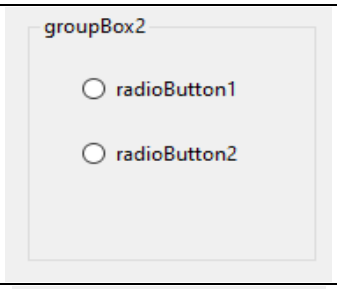
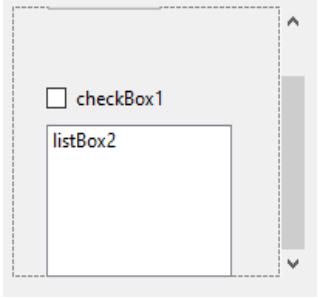
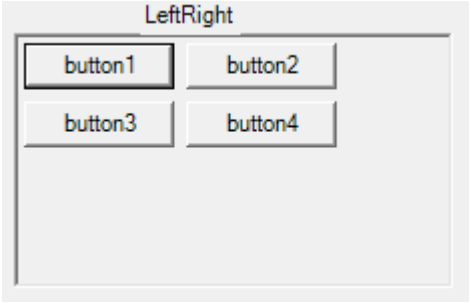
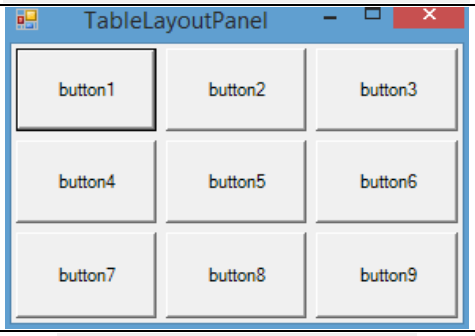
Кінець таблиці Е.2

1	2
10. Міжсторінковий з'єднувач	
11. Лінії потоку	Застосовують для зазначення напрямку лінії потоку: без стрілки, якщо лінія спрямована ліворуч, праворуч і зверху вниз; зі стрілкою – в інших випадках. Відстань між паралельними лініями потоку має бути не менше 3 мм, між іншими символами схеми – не менше 5 мм.  Злиття ліній потоку застосовується у випадку коли, кожна з дінйій спрямована до того самого символа на схемі. Місце з'єднання ліній потоку позначено крапкою. 
12. Коментарі	Застосовується, якщо пояснення не поміщається усередині символа (для пояснення характеру параметрів, особливостей процесу, ліній потоку й ін.) Коментар записують паралельно основному напису. Коментар поміщають у вільному місці схеми алгоритму на цьому аркуші і з'єднують з пояснювальним символом. 

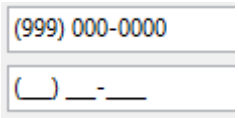
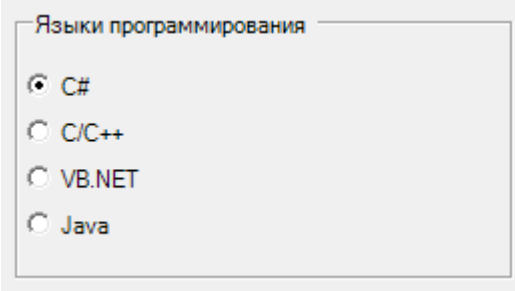
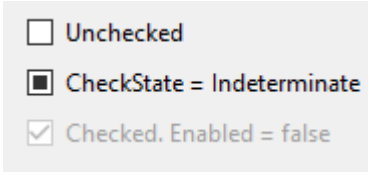
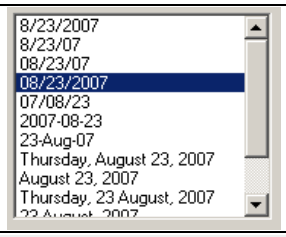
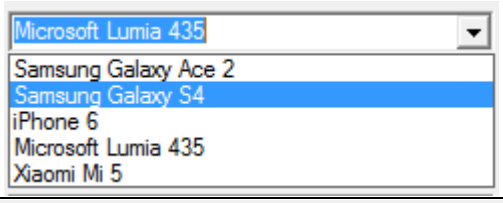
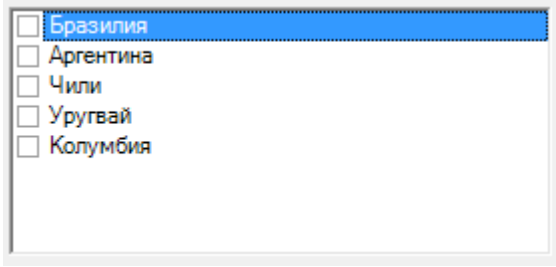
ДОДАТОК Ж

КОМПОНЕНТИ ГРАФІЧНОГО ІНТЕРФЕЙСУ

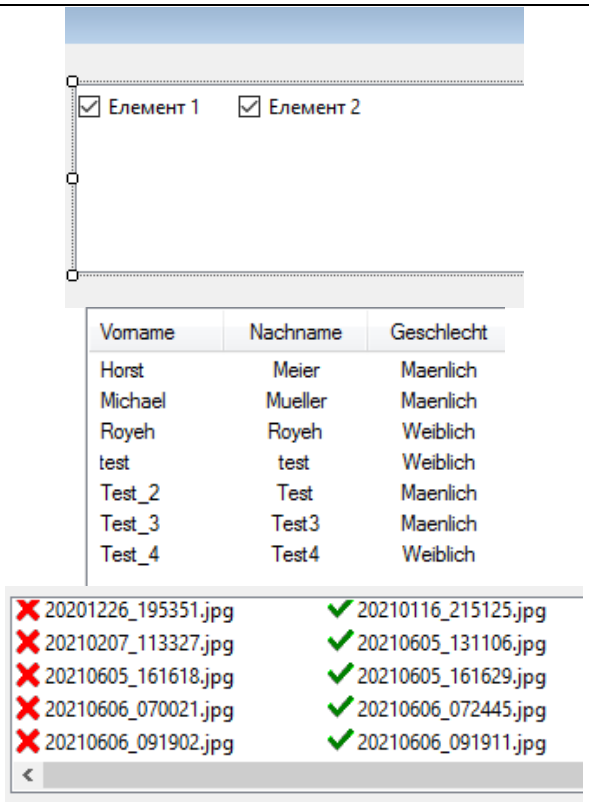
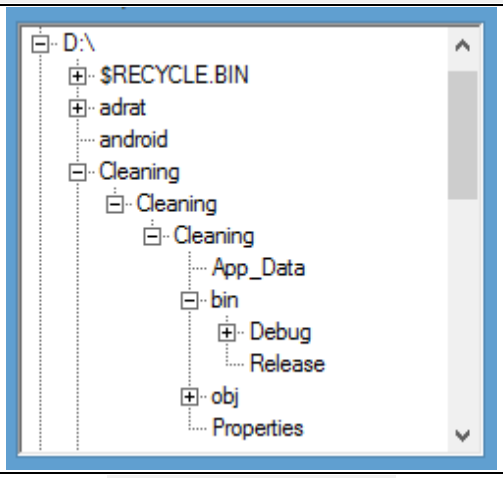
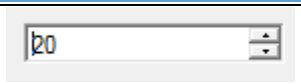
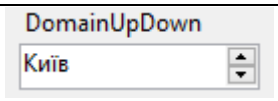
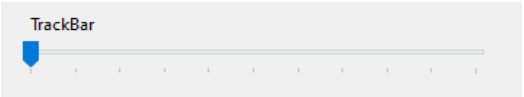
Таблиця Ж.1 – Компоненти графічного інтерфейсу «Toolbox»

Назва	Опис	Зображення
1	2	3
GroupBox	Спеціальний контейнер, який відділений від іншої форми межею.	
Panel	Є прямокутною зоною для розташування елементів інтерфейсу	
FlowLayoutPanel	Дозволяє змінювати позиціонування та компонування дочірніх елементів при зміни розмірів форми.	
TableLayoutPanel	Елемент перевизначає панель і має дочірні елементи управління у вигляді таблиці, де для кожного елемента є своя комірка.	
Button	Елемент управління діями миші і клавіатури	
TextBox	Введення та редагування багаторядкового тексту	

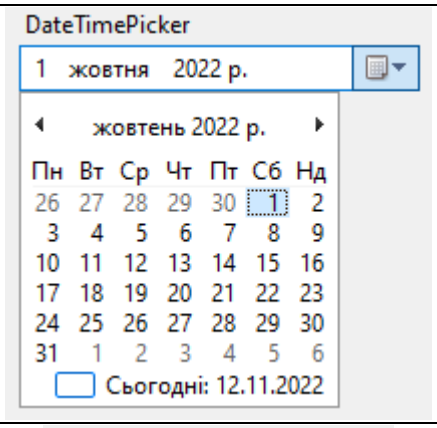
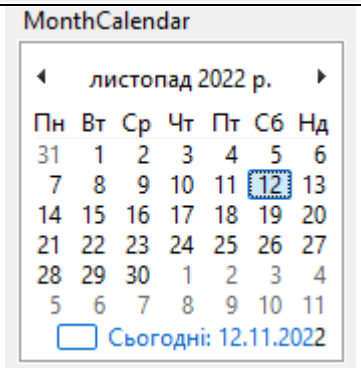
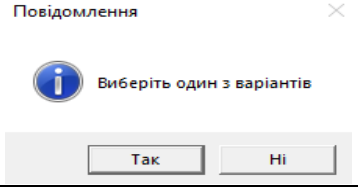
Продовження таблиці Ж.1

1	2	3
MaskedTextBox	Текстове поле для контролю введення користувачем текстових даних	
RadioButton	Перемикачі розташовуються групами, і вмикання одного перемикача означає вимикання решти.	
CheckBox	Призначений для встановлення одного з двох значень: відмічено (позначено) або не позначено (не відмічено)	
ListBox	Елемент ListBox є простим списком	
ComboBox	Утворює випадний список і поєднує функціональність компонентів ListBox і TextBox	
CheckedListBox	Симбіоз компонентів ListBox та CheckBox	

Продовження таблиці Ж.1

1	2	3
ListView	Елемент ListView є списком, але з розширеними можливостями, порівняно з ListBox. У ListView можна відображати складні дані по-різному, можна задавати зображення та піктограми.	
TreeView	TreeView є візуальним елементом у вигляді дерева. Дерево містить вузли, які є об'єктами TreeNode.	
NumericUpDown	Надає користувачеві вибір числа з певного діапазону значень	
DomainUpDown	Призначений для введення текстової інформації	
TrackBar	TrackBar є елементом, який за допомогою переміщення повзунка дозволяє вводити числові значення.	

Продовження таблиці Ж.1

1	2	3
Timer	Timer є компонентом запуску дій, повторюваних через певний проміжок часу.	Не візуальний
ProgressBar	ProgressBar слугує для того, щоб надати користувачеві інформацію про хід виконання будь-якої задачі.	
DateTimePicker	DateTimePicker є календарем, що розкривається за натисканням, в якому можна вибрати дату	
MonthCalendar	За допомогою MonthCalendar також можна вибрати дату, тільки в цьому випадку цей елемент є календарем, який не потрібно розкривати	
PictureBox	PictureBox призначений для показу зображень. Він дозволяє відобразити файли у форматі bmp, jpg, gif, а також метафайли зображень та іконки.	
MessageBox	Для виведення повідомлень використовується елемент MessageBox.	

Кінець таблиці Ж.1

1	2	3
OpenFileDialog	Виведення діалогового вікна для завантаження даних файлів	
SaveFileDialog	Виведення діалогового вікна для збереження даних у файл	
ImageList	ImageList не є візуальним елементом управління, однак він є компонентом, який використовується елементами управління. Він визначає набір зображень, які можуть використовувати такі елементи, як ListView або TreeView.	

ДОДАТОК И

ТИПОВИЙ ШАБЛОН ІНСТРУКЦІЇ КОРИСТУВАЧА

ІНСТРУКЦІЯ КОРИСТУВАЧА
1. Загальні відомості:
Назва програми: [Назва програми]
Версія: [Версія програми]
Призначення: [Короткий опис призначення програми]
Системні вимоги:
Операційна система: [ОС]
Процесор: [Процесор]
Оперативна пам'ять: [ОП]
Місце на диску: [МБ/ГБ]
2. Встановлення програми:
2.1 Завантаження інсталяційного файлу: перейдіть на офіційний сайт за посиланням [URL], завантажте інсталяційний файл для вашої операційної системи.
2.2 Процес встановлення: відкрийте завантажений файл, дотримуйтесь інструкцій інсталятора, оберіть папку для встановлення програми, натисніть «Встановити».
2.3 Налаштування після встановлення: запустіть програму, здійсніть первинне налаштування (авторизація, вибір мови тощо).
3. Початок роботи:
3.1 Запуск програми: запустіть програму, двічі клацнувши на її іконку на робочому столі або через меню «Пуск».
3.2 Авторизація: введіть логін та пароль (якщо потрібно), натисніть «Увійти».
3.3 Огляд інтерфейсу користувача: Опис основних елементів інтерфейсу (меню, панелі інструментів, робоча область).
4. Основні функції:
4.1 Функція 1 – опис функції 1, покрокова інструкція з використання.
4.2 Функція 2 – опис функції 2, покрокова інструкція з використання.
5. Додаткові функції:
5.1 Функція 3 – опис додаткової функції, налаштування та використання.
5.2 Функція 4 – опис додаткової функції, налаштування та використання.
6. Налаштування програми:
6.1 Основні налаштування – доступ до меню налаштувань, опис основних параметрів.
6.2 Користувацькі налаштування – як змінити налаштування під власні потреби.
6.3 Резервне копіювання та відновлення налаштувань – створення резервної копії налаштувань, відновлення налаштувань з резервної копії.
7. Вирішення проблем:
7.1 Типові проблеми та їх вирішення:
Проблема 1: [Опис проблеми], рішення: [Опис рішення]
Проблема 2: [Опис проблеми], рішення: [Опис рішення]
7.2 Контактна інформація служби підтримки
Email: [Електронна пошта підтримки]
Телефон: [Телефон підтримки]
Час роботи: [Години роботи]
8. Додаткова інформація:
Посилання на офіційний сайт: [URL]
Інформація про ліцензію: [Опис ліцензії]
Авторські права: [Інформація про авторські права]

ДОДАТОК К

СПИСОК ТЕМ КУРСОВИХ РОБІТ

1 Розробка програмного застосунку для розширеного пошуку даних у реєстрі, використовуючи регулярні вирази.

2 Розробка програмного застосунку для аналізу даних і структур реєстру.

3 Розробка програмного застосунку для створення резервної копії реєстру, і перевірки її цілісності.

4 Розробка програмного застосунку для імпорту/експорту даних реєстру.

5 Розробка програмного застосунку для перевірки валідності записів реєстру.

6 Розробка програмного застосунку для редагування записів реєстру.

7 Розробка програмного застосунку для запису змін робочого столу у медіафайл.

8 Розробка програмного застосунку для управління режимами роботи екрана.

9 Розробка програмного застосунку для відтворення медіафайлів (медіапрогравач).

10 Розробка програмного застосунку для керування і запису даних із зовнішнього мультимедійного пристрою типу Web-камера.

11 Розробка програмного застосунку аналізу і пошуку медіафайлів.

12 Розробка програмного застосунку для редагування відеофайлів.

13 Розробка програмного застосунку для створення графічних зображень.

14 Розробка програмного застосунку для перегляду графічних форматів файлів.

15 Розробка програмного застосунку для редагування графічних зображень.

16 Розробка програмного застосунку для запису дій з екрана (desktop graber).

17 Розробка програмного застосунку для побудови 3D-зображень за допомогою OpenGL.

18 Розробка програмного застосунку для побудови 3D-зображень за допомогою DirectX.

19 Розробка програмного застосунку корегування фону зображення.

20 Розробка програмного застосунку для визначення і візуалізації однакових зображень на жорсткому диску.

21 Розробка програмного застосунку «Електронна лупа».

22 Розробка програмного застосунку для конвертації зображень.

23 Розробка програмного застосунку для афінного трансформування графічних зображень.

24 Розробка програмного застосунку для управління регіональними налаштуваннями ОС.

25 Розробка програмного застосунку «Віртуальна клавіатура».

26 Розробка програмного застосунку для планування завдань ОС.

27 Розробка програмного застосунку для управління поточними процесами.

28 Розробка програмного застосунку для управління режимами заставки робочого столу.

29 Розробка програмного застосунку для управління завданнями ОС.

30 Розробка програмного застосунку для управління автозапуском додатків ОС.

31 Розробка програмного застосунку для блокування ОС у разі несанкціонованого доступу.

32 Розробка програмного застосунку для управління службами ОС Windows.

33 Розробка програмного застосунку для редагування режиму завантаження ОС.

34 Розробка програмного застосунку для оптимізації роботи ОС шляхом видалення тимчасових файлів.

35 Розробка програмного застосунку для керування робочими столами ОС.

36 Розробка програмного застосунку для управління панеллю швидкого доступу «Старт».

37 Розробка програмного застосунку для управління налаштуваннями панелі завдань ОС.

38 Розробка програмного застосунку для управління шрифтами ОС.

39 Розробка програмного застосунку для керування налаштуваннями перегляду провідника ОС Windows.

40 Розробка програмного застосунку для управління режимами енергозбереження.

41 Розробка програмного застосунку для візуалізації і аналізу стану процесора.

42 Розробка програмного застосунку для аналізу режиму процесора.

43 Розробка програмного застосунку для аналізу завантаження ЦП, ОЗП й мережі.

44 Розробка програмного застосунку для управління режимом відновлення ОС.

45 Розробка програмного застосунку для відображення продуктивності роботи ПК.

46 Розробка програмного застосунку для управління персональними налаштуваннями.

47 Розробка програмного застосунку для управління обліковими записами ОС.

48 Розробка програмного застосунку для блокування дій ОС за паролем (батьківський контроль)

49 Розробка програмного застосунку для створення бази даних файлів і папок.

50 Розробка програмного застосунку для оптимізації даних на жорсткому диску шляхом пошуку однакових файлів і знищення пустих файлів.

51 Розробка програмного застосунку для розширеного пошуку даних на жорсткому диску, використовуючи регулярні вирази.

52 Розробка програмного застосунку для управління блокуванням / розблокуванням файлів і папок.

53 Розробка програмного застосунку для аналізу дискового простору.

54 Розробка програмного застосунку для аналізу файлів, папок на жорсткому диску.

55 Розробка програмного застосунку для управління властивостями папок і файлів.

56 Розробка програмного застосунку для пошуку та сортування файлів на жорсткому диску.

57 Розробка програмного застосунку для перегляду інформації про файл і зміни його атрибутів.

58 Розробка програмного застосунку для архівуванню файлів.

59 Розробка програмного застосунку для управління доступом до файлів.

60 Розробка програмного застосунку для групового перейменування файлів і папок за допомогою регулярних виразів.

61 Розробка програмного застосунку для групового керування файлами і папками за допомогою регулярних виразів.

62 Розробка програмного застосунку файлового менеджера.

63 Розробка програмного застосунку для визначення подібності файлів.

64 Розробка програмного застосунку аналізу файлів на диску і побудова звіту.

65 Розробка програмного застосунку для обробки і візуалізації Excel файлів.

66 Розробка програмного застосунку для керування базою даних Postgres.

67 Розробка програмного застосунку для управління базами даних засобами SQLite.

68 Розробка програмного застосунку для редагування даних засобами SQLite.

69 Розробка програмного застосунку для пошуку даних засобами SQLite за допомогою регулярних виразів.

70 Розробка програмного застосунку для управління запам’ятовувальними пристроями.

71 Розробка програмного застосунку для створення звіту про використання користувачем миші і клавіатури.

72 Розробка програмного застосунку для логування дії миші.

73 Розробка програмного застосунку для запуску медіаданих з медіаносіїв.

74 Розробка програмного застосунку для перехоплення та аналізу натискання клавіш на клавіатурі.

75 Розробка програмного застосунку для логування роботи оператора на клавіатурі.

76 Розробка програмного застосунку для моніторингу роботи HDD і SSD.

77 Розробка програмного застосунку для керування і запису даних Web-камерою

78 Розробка програмного застосунку для планування управлінням Web-камерою

79 Розробка програмного застосунку для керування робочим столом за допомогою віртуальної панелі керуванн

80 Розробка програмного застосунку для створення звіту використання користувачем пристроїв.

81 Розробка програмного застосунку для управління і відображення пристроїв.

82 Розробка програмного застосунку для управління принтерами.

83 Розробка програмного застосунку для створення звіту використання користувачем мережевих ресурсів.

84 Розробка програмного застосунку для аналізу трафіка комп'ютерної мережі.

85 Розробка програмного застосунку для налаштування мережевого з'єднання.

86 Розробка програмного застосунку для конфігурування мережевих дисків.

87 Розробка програмного застосунку для встановлення та видалення програмних додатків.

88 Розробка програмного застосунку для аналізу завантаження системи додатками.

89 Розробка програмного застосунку для керування процесами Windows.

90 Розробка програмного застосунку для очищення системи від залишків видалених програм.

91 Розробка програмного застосунку для імпорту / експорту даних встановлених програмних додатків.

92 Розробка програмного застосунку для створення звіту використання користувачем додатків.

93 Розробка програмного застосунку для візуалізації багатопоясного годинника.

94 Розробка програмного застосунку для редагування даних (багатофункціональний текстовий редактор).

95 Розробка програмного застосунку для створення та редагування текстових файлів.

96 Розробка програмного застосунку для перевірки орфографії.

97 Розробка програмного застосунку для онлайн багатомовного перекладу.

98 Розробка програмного застосунку для синтезу мовлення текстових даних у TXT-форматі.

99 Розробка програмного застосунку для пошуку тексту за допомогою регулярних виразів.

100 Розробка програмного застосунку для заміни тексту за допомогою регулярних виразів.

101 Розробка програмного застосунку для перетворення числових даних у текстовий вигляд.

102 Розробка програмного застосунку для редагування даних у двійковій, десятковій, шістнадцятковій системах числення.

103 Розробка програмного застосунку для відображення тексту у різних системах кодування.

104 Розробка програмного застосунку симетричного шифрування текстових даних.

105 Розробка програмного застосунку асиметричного шифрування текстових даних.

106 Розробка програмного застосунку для перегляду *.doc, *.rtf файлів.

107 Розробка програмного застосунку для візуалізації HTML-файлів

108 Розробка програмного застосунку для редагування HTML-файлів.

109 Розробка програмного застосунку для відображення структури web-сторінок

110 Розробка програмного застосунку для роботи з Goggle-диском.

111 Розробка програмного застосунку для роботи з поштовими сервісами Hotmail, Outlook.com, Office 365.

112 Розробка програмного застосунку для роботи з поштовими сервісами Gmail.

113 Розробка програмного застосунку для перегляду web – сторінок (браузер)

114 Розробка програмного застосунку для статистичного аналізу даних.

115 Розробка програмного застосунку для обчислення користувацької функції.

116 Розробка програмного застосунку для побудови 2D-графіків функцій.

117 Розробка програмного застосунку для побудови 3D-графіків функцій.

118 Розробка програмного застосунку для групування даних за ознаками.

119 Розробка програмного застосунку для розрахунку спектра даних.

Електронне навчальне видання

Олександр Миколайович Рейда
Алла Василівна Денисюк

Розробка програмних застосунків ОС Windows до виконання курсової роботи з дисципліни «Операційні системи» для здобувачів спеціальності 121 «Інженерія програмного забезпечення»

Навчальний посібник

Рукопис оформив *О. Рейда*

Редактор *В. Дружиніна*

Оригінал-макет виготовила *Т. Старічек*

Підписано до видання 03.07.2025.

Гарнітура Times New Roman.
Зам. № Р2025-100.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.