**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

Кафедра Програмного забезпечення комп’ютерних систем

**Об’єктно-орієнтоване програмування**

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

**до виконання курсового проекту**

для студентів спеціальності 121

«Інженерія програмного забезпечення»

денної форми навчання

Чернівці, 2024

**ЗМІСТ**

Вступ

1. Мета й завдання курсового проєкту

2. Організація курсового проєкту

3. Тематика робіт курсових проєктів

4. Структура та обсяг курсового проєкту

5. Вимоги до розробки індивідуальних завдань

6. Вимоги до оформлення курсових проєктів

7. Підготовка до захисту

8. Порядок захисту

ДОДАТКИ

Додаток А. Зразок оформлення титульної сторінки курсового проєкту

Додаток Б. Зразок бланка завдання на курсовий проєкт

Додаток В. Приклад реферату

Додаток Г. Приклад змісту пояснювальної записки

Додаток Д. Приклад оформлення умовних позначень

Додаток Е. Опис структури програми

Додаток Ж. Опис методів програми

Додаток И. Опис інтерфейсу користувача

Додаток К. Приклад оформлення висновків до курсового проєкту

Додаток М. Приклад оформлення тексту програми

**ВСТУП**

Основоположною ідеєю об’єктно-орієнтованого підходу є об’єднання даних і дій, виконуваних над цими даними, в єдине ціле, яке називається об’єктом. Об’єктно-орієнтована методологія будується на принципах абстракції і класифікації. При об’єктно-орієнтованому підході необхідно спочатку ідентифікувати типи об’єктів, що беруть участь у системі. До об’єктів відносяться всі елементи системи, що міняють свій стан і відіграють певну роль при її функціонуванні.

Об’єктно-орієнтоване програмування (ООП) моделює об’єкти реального світу за допомогою програмних аналогів. ООП дає нам найбільш природний і інтуїтивний спосіб розгляду процесу програмування як моделювання реально існуючих об’єктів, їх атрибутів і поведінки. ООП моделює також зв’язки між об’єктами. Подібно тому, як люди посилають повідомлення, об’єкти теж зв’язуються один з одним за допомогою повідомлень.

У результаті вивчення дисципліни «Об'єктно-орієнтоване програмування» студент повинен знати принципи ООП (інкапсуляція, успадкування, поліморфізм), як оголошуються класи, створюються, ініціалізуються і руйнуються об’єкти класів, відмінність між абстрактними і конкретними класами, ієрархію класів потоків введення-виведення.

Після вивчення дисципліни студент повинен уміти створювати класи і об’єкти, управляти доступом до даних і функцій, ініціалізувати об’єкти за допомогою конструкторів, використовувати успадкування і віртуальні функції для поліморфізму, створювати, читати і записувати файли, форматувати дані, що виводяться на екран, а також використовувати свої знання й уміння в курсовому проєктуванні та при виконанні бакалаврської роботи.

**1. МЕТА Й ЗАВДАННЯ КУРСОВОГО ПРОЄКТУ**

Метою курсового проєкту є закріплення на практиці вміння використовувати основні концепції об’єктно-орієнтованого підходу при створенні складних програмних проєктів.

Робота над курсовим проєктом сприяє систематизації, поглибленню й закріпленню знань, отриманих студентами при вивченні дисципліни Об'єктно-орієнтоване програмування, в набутті навичок використання основ алгоритмізації та програмування на алгоритмічних мовах високого рівня (C++) з використанням принципів об’єктно-орієнтованого проєктування та програмування, методики розробки програмних модулів та програмного забезпечення комп’ютерних систем управління, набуття студентами теоретичних знань та практичних навичок в області використання сучасних систем візуального проєктування програмних засобів комп’ютерних систем управління та освоєння принципів та методів сучасних технологій програмування, набуття навичок професійних та практики наукових досліджень з оформленням відповідного текстового, програмного та ілюстративного матеріалу у формі наукового звіту.

У процесі курсового проєкту студенти розвивають навички практичного застосування отриманих знань при створенні комплексного додатку з використанням сучасних інструментальних засобів розробки. При цьому студент повинен показати вміння користуватися спеціальною літературою, державними стандартами, довідниками та іншими матеріалами з інформаційних технологій.

У розробленому курсовому проєкті студент повинен показати знання з:

* предметної області відповідно до постановки завдання;
* основних концепцій об’єктно-орієнтованого програмування;
* сучасних інструментальних засобів, призначених для розробки об’єктно-орієнтованих додатків.

Студент повинен показати вміння з:

* аналізу постановки завдання;
* декомпозиції програмної системи на підсистеми;
* розробки загальної архітектури програмної системи;
* деталізації загальної архітектури програмної системи у термінах об’єктно-орієнтованого програмування;
* програмної реалізації системи, що розробляється, мовою програмування C++;
* застосування стандартних бібліотек мови програмування;
* документування вихідного коду програми;
* використання засобів розробки програм та отримання довідкової інформації;
* розробки та оформлення програмної документації.

Робота над курсовим проєктом певною мірою визначає загальнотеоретичну та спеціальну підготовку студента і в остаточному підсумку готує його до майбутнього виконання більш складного й завершального етапу навчального процесу – дипломного проєктування.

**2. ОРГАНІЗАЦІЯ КУРСОВОГО ПРОЄКТУ**

Відповідно до навчального плану вивчення дисципліни «Об'єктно-орієнтоване програмування» включає лекційні та лабораторні заняття. Завершується вивчення дисципліни розробкою і захистом курсового проєкту.

Курсовий проєкт студент виконує самостійно. Якісне виконання курсового проєкту вимагає чіткої організації роботи студента з моменту вибору теми проєкту й до його захисту.

Керівництво курсовим проєктуванням здійснюється викладачами кафедри, які беруть участь у викладанні даної дисципліни.

Тематика курсових проєктів відповідає програмі навчальної дисципліни «Об'єктно-орієнтоване програмування».

Основними етапами виконання курсового проєкту є:

1. формулювання вимог до програми;
2. кодування програми;
3. тестування програми;
4. оформлення пояснювальної записки;
5. захист роботи.

***Формулювання вимог до програми***

На етапі формулювання вимог студент повинен:

* сформулювати призначення програми;
* визначити перелік функцій, які виконує програма;
* запропонувати інтерфейс користувача, який надасть можливість останньому застосувати дані функції.

Формулювання вимог є самостійним етапом роботи і завершується створенням розділу пояснювальної записки «Специфікація програми».

Специфікація програми складається з послідовного опису усіх виділених функцій, та пов’язаного з ними графічного інтерфейсу користувача. У специфікації програми треба обов’язково надати зображення усіх вікон програми.

***Кодування програми***

На етапі кодування створюється вихідний код програми. Він повинен бути прокоментований та увесь має бути наведений у додатку до пояснювальної записки. У тексті основних розділів пояснювальної записки можна наводити лише невеликі за розміром окремі фрагменти коду.

***Тестування програми***

Функціональне тестування перевіряє, як програма виконує свої функції в умовах коректних та некоректних дій користувача.

***Пояснювальна записка***

Оформлення пояснювальної записки є останнім етапом виконання курсового проєкту і має надати студентові навичок документування програмного продукту.

**3. ТЕМАТИКА РОБІТ КУРСОВИХ ПРОЄКТІВ**

Всі студенти виконують курсовий проєкт згідно теми, яку вони обирають з переліку індивідуальних завдань. Зміна варіанту завдання погоджується з керівником роботи у індивідуальному порядку.

Теми курсових проєктів розробляє та затверджує випускова кафедра з урахуванням специфіки спеціальності «Інженерія програмного забезпечення», власного досвіду керівників курсових проєктів; наукових досліджень та професійних інтересів професорсько-викладацького складу кафедри тощо.

За всі рішення, які приймаються під час виконання курсового проєкту, відповідальність несуть студенти – виконавці. Керівник курсового проєкту відповідає за науково-технічній рівень розробки та дає узагальнену оцінку роботі студента.

Для виконання курсових проєктів кафедра надає студентам машинний час у комп’ютерних класах та консультації викладачів.

**4. СТРУКТУРА ТА ОБСЯГ КУРСОВОГО ПРОЄКТУ**

Курсовий проект складається з пояснювальної записки та інших матеріалів, зокрема програмного додатку, що розробляється відповідно до завдання. У табл.1 наведена структура пояснювальної записки курсового проекту.

Таблиця 1 – Структурні елементи пояснювальної записки

|  |
| --- |
| Титульний аркуш |
| Завдання на курсове проєктування |
| Реферат |
| Зміст |
| Перелік умовних позначень |
| Пояснювальна записка |
| 1. Загальні вимоги до програми |
| 1. Призначення та область застосування |
| 1. Функціональні вимоги |
| 1. Структура програми (модулі проекту та алгоритми виконання) |
| 1. Опис методів програми |
| 1. Програмні засоби |
| 1. Опис інтерфейсу користувача |
| Висновки |
| Список використаних джерел |
| Додатки |
| Додаток А. Скролінг (текст) програми |

**Методичні рекомендації до розроблення структурних елементів пояснювальної записки курсового проєкту**

1. Титульний аркуш повинен містити інформацію про дисципліну, з якої виконувалося проектування, тему індивідуального завдання, виконавця курсового проекту. Титульний лист має бути підписаний студентом та керівником проекту.

Приклад оформлення титульного листа наведено у додатку A.

1. Завдання на курсове проектування є початковим документом для розробки проекту. Відповідно до теми студент разом із керівником складає завдання на курсовий проект.

Друкується завдання з обох сторін аркуша формату А4.

Приклад оформлення завдання на курсове проектування наведено у додатку Б.

Титульний лист і завдання на курсове проектування вважаються відповідно першим і другим листом пояснювальної записки, однак номера листів на них не проставляються.

1. Реферат – це короткий виклад змісту пояснювальної записки. Реферат має бути стислим, інформативним і містити відомості про обсяг звіту, кількість частин звіту, кількість ілюстрацій, таблиць, додатків, кількість використаних джерел (усі відомості наводять, включаючи дані додатків); текст реферату; перелік ключових слів.

У короткій лаконічній формі має бути представлений опис основних частин проекту. У формі, що стверджує, наголошується, що було виконано, розраховано, сконструйовано, запропоновано, модернізовано і так далі.

Перелік ключових слів повинен характеризувати вміст пояснювальної записки і включати від 5 до 15 слів (словосполучень), розміщених після тексту реферату, надрукованих великими літерами в називному відмінку в рядок через коми, курсивом.

Обсяг реферату не повинний перевищувати одного листа.

Реферат рекомендується складати після написання пояснювальної записки проекту.

Приклад реферату наведений у додатку В.

1. До змісту включають: вступ; послідовно перелічені назви всіх розділів, підрозділів, пунктів і підпунктів основної частини пояснювальної записки; висновки; список використаних джерел; назви додатків і номери сторінок. Справа наводяться номери сторінок, з яких починається зазначений матеріал.

Зміст поміщають після титульного листа і реферату та включають в загальну чисельність листів пояснювальної записки.

Приклад змісту пояснювальної записки наведений у додатку Г.

1. Перелік умовних позначень містить усі прийняті в пояснювальній записці мало поширені умовні позначення, символи, одиниці, скорочення та терміни.

Приклад оформлення переліку умовних посилань наведено у додатку Д.

1. Пояснювальна записка містить інформацію про загальні вимоги до програми, призначення та область застосування, функціональні вимоги та структуру програми, інтерфейс користувача.

*Приклад вимог до архітектури програми:*

1. Використання не менше одного стандартного контейнерного класу.
2. Використання файлового введення-виведення даних.
3. Використання механізму виключень для обробки помилок введення-виведення даних.

*Приклад вимог до функціональності додатка:*

1. Створення файлу бази даних (ім’я файлу бази та каталог файлової системи для його збереження вибираються користувачем з використанням відповідного діалогового вікна).
2. Читання всіх даних з файлу бази (каталог файлової системи та ім’я файлу бази вибираються користувачем з використанням відповідного діалогового вікна) та їх відображення.
3. Додавання елементу даних до файлу бази.
4. Оновлення будь-якого елемента даних у файлі бази.
5. Видалення будь-якого елемента даних з файлу.
6. Сортування інформації, що відображується в графічному інтерфейсі користувача, за різними реквізитами.
7. Фільтрація інформації, що відображується в графічному інтерфейсі користувача, за різними критеріями.
8. Отримання та відображення підсумкової інформації.
9. Забезпечення перевірки допустимості даних, що вводяться користувачем, з використанням компоненту ErrorProvider.
10. Видача користувачу попереджуючих та інформаційних повідомлень.

*Приклад вимог до вихідного коду додатка*

1. Вихідний код кожного з класів програми повинен міститися в окремому файлі.
2. Наявність коментарів (для класів – призначення класу; для методів – призначення методу, опис параметрів та значення, що повертається) з обов’язковим використанням відповідних документаційних XML-тегів.

У підрозділі «2. Призначення та область застосування» вказується функціональне (для чого можна застосовувати) та експлуатаційне (де можна застосовувати) призначення програми.

У підрозділі «3. Функціональні вимоги» наводиться перелік дій, які повинна виконувати програма.

У підрозділі «4. Структура програми» мають бути наведені описи модулів програми з алгоритмами виконання (додаток Е).

Алгоритм виконання представляють у вигляді блок-схем до кожного з модулів програми. Алгоритми створюються за функціональною схемою (ДСТУ 19.701-90) – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках та зазначенням використовуваних файлів і пристроїв. Для зображення функціональних схем використовують спеціальні позначення, встановлені стандартом (табл.2).

Таблиця 2 – Позначення елементів функціональних схем

|  |  |  |
| --- | --- | --- |
| Назва блоку | Позначення | Призначення блоку |
| Дані |  | Символ відображає дані, носій даних не визначений. |
| Запам’ятовуванні дані |  | Символ відображає збережені дані у вигляді, придатному для обробки, носій даних не визначений. |
| Оперативний пристрій |  | Символ відображає дані, що зберігаються в оперативному запам’ятовуючому пристрої. |
| Запам’ятовуючий пристрій з послідовним доступом |  | Символ відображає дані, що зберігаються в пристрої з послідовним доступом (магнітна стрічка, касета з магнітною стрічкою, магнітофонна касета). |
| Запам’ятовуючий пристрій з прямим доступом |  | Символ відображає дані, що зберігаються в пристрої з прямим доступом (магнітний диск, магнітний барабан, гнучкий магнітний диск). |
| Документ |  | Символ відображає дані, представлені на носії у зручній для зчитування формі (машинограма, документ для оптичного або магнітного зчитування, мікрофільм, рулон стрічки з підсумковими даними, бланки введення даних). |
| Ручне введення |  | Символ відображає дані, що вводяться вручну під час обробки з пристроїв будь-якого типу (клавіатура, перемикачі, кнопки, світлове перо, смужки з штриховим кодом). |
| Карта |  | Символ відображає дані, представлені на носії у вигляді карти (перфокарти, магнітні карти, карти зі зчитуваними мітками, карти з відривним ярликом, карти зі сканованими мітками). |
| Дисплей |  | Символ відображає дані, представлені в людино-читаючій формі на носії у вигляді відображуючого пристрою (екран для візуального спостереження, індикатори введення інформації). |
| Процес |  | Символ відображає функцію обробки даних будь-якого виду (виконання певної операції або групи операцій, що приводить до зміни значення, форми або розміщення інформації або до визначення, за яким з декількох напрямків потоку слід рухатися). |
| Умовний процес |  | Символ відображає умовний процес, що складається з однієї або декількох операцій або кроків програми, які визначені в іншому місці (в підпрограмі, модулі). |
| Ручна операція |  | Символ відображає будь-який процес, що виконується людиною. |
| Підготовка |  | Символ відображає модифікацію команди або групи команд з метою впливу на деяку подальшу функцію (встановлення перемикача, модифікація індексного регістра або ініціалізація програми). |
| Рішення |  | Символ відображає рішення або функцію перемикача типу, що має один вхід і ряд альтернативних виходів, один і тільки один з яких може бути активізований після обчислення умов, визначених всередині цього символу. Відповідні результати обчислення можуть бути записані по сусідству з лініями, що відображають ці шляхи. |

У підрозділі «5. Опис методів програми» приводяться назви всіх методів програми з описом операцій, що вони виконують (додаток Ж).

У підрозділі «6. Програмні засоби» коротко описується середовище розробки програми та інші програмні засоби необхідні для функціонування розробленої програми.

У підрозділі «7. Опис інтерфейсу користувача» представляють скріншоти програми по кожному з індивідуальних завдань (додаток И). На скріншотах необхідно подавати такі фрагменти програми, які найбільш повно показують результат виконання конкретного завдання. Опис користувацького інтерфейсу – дозволяє користувачу отримати навігаційну підтримку по роботі з програмою.

1. У висновках подають результати виконання роботи у вигляді коротко сформульованих і пронумерованих тез (додаток К). Кількість висновків залежить від обсягу отриманих результатів, складності програми та прикладних задач.
2. Список використаних джерел містить усі інформаційні джерела, які були використані у ході виконання курсового проекту (цитовано, згадано або розглянуто у роботі).

Приклад оформлення літературних джерел наведено у додатку Л.

1. У додатках вміщують матеріал, який є необхідним для повноти пояснювальної записки, але не може бути послідовно розміщений в її основній частині через великий обсяг або способи відтворення та з інших причин.

У тексті пояснювальної записки на всі додатки потрібно дати посилання, розміщувати додатки слід в порядку згадування в основному тексті.

Ілюстрації (діаграми бізнес-процесів, схеми алгоритмів, технологічних процесів, сценарії діалогів та ін.), таблиці, проміжні математичні докази, формули та розрахунки, текст допоміжного характеру та інші матеріали можуть бути оформлені у вигляді додатків.

У одному з додатків повинен знаходитися вихідний код програми, яка розроблена студентом при виконанні курсового проекту. Вихідний код, який згенерований візуальним середовищем (платформою) розробки, наводити не потрібно.

**5. ВИМОГИ ДО РОЗРОБКИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ**

### ****Загальні правила****

При розробці індивідуального завдання курсового проєкту використати об’єктно-орієнтований підхід, який завдяки принципам інкапсуляції, успадкування та поліморфізму дає можливість виділити всі спільні та відмінні риси окремих елементів поставленого завдання і записати їх у програмі в найлаконічнішій формі.

Розробка програми при об’єктно-орієнтованому підході розбивається на розробку класів та глобальних типів даних, реалізацію методів, глобальних функцій, створення об’єктів та організацію взаємодії між ними в головній програмі для досягнення поставленої мети.

### 

### Розробка системи класів

Після того, як виконана декомпозиція програми (розроблена її структура) і відомі алгоритми розв’язку задач, можна приступати до реалізації наміченого плану:

* Потрібно чітко уявити, з якими даними буде працювати програма (чи окремий модуль) та які дії над ними доведеться виконувати. Дані та дії над ними утворюють об’єкт. Також можна уявити програму як набір об’єктів (реально існуючих фізичних об’єктів або абстрактних понять, що існують лише в нашій уяві), які взаємодіють між собою.
* Наступним етапом буде поділ об’єктів на групи, що мають хоча б щось спільне (дані, властивості, функціональність). У більшості випадків об’єкти будуть, більш чи менш, подібні між собою. Деколи в групах можна виділити підгрупи і т.д., аж до складного ієрархічного дерева. У кожній групі виділяється найзагальніші властивості, притаманні всім без винятку об’єктам у групі та дії над ними. Отже, отримаємо базовий клас для цієї групи об’єктів, даними якого будуть загальні параметри об’єктів групи, а методами – дії, які можна проводити над даними будь-якого об’єкта групи.
* Далі для кожної з підгруп (у межах виділеного класу об’єктів) додаємо притаманні лише їй властивості, як поля нового, породженого від базового класу. Додаткові дії, які властиві кожній підгрупі, стають методами породжених класів, а параметри, яких не мали об’єкти базового класу, – даними породжених (похідних) класів. Породження класів продовжується до тих пір, поки не будуть описані всі параметри об’єктів та дії, які необхідно над ними здійснювати для розв’язку поставлених задач.
* Описана послідовність повторюється для кожної з Групи об’єктів у програмі.

Для прикладу розглянемо програму, яка повинна продемонструвати гру в шахи. У ній можна виділити такі дві Групи об’єктів: перша – реалізує графічне відображення на екрані шахової дошки та фігур на ній, друга – реалізує логіку роботи програми – генерує ходи, перевіряє їх правильність, слідкує за часом і т.д.

*Розглянемо першу групу:*

1. З чого вона складається?

Сюди входять різноманітні видимі елементи: дошка, клітинки на ній, різні фігури, позначення рядів на дошці.

1. Які в них параметри?

Це – координати, розміри, колір для фігур, для клітинки – ще тип фігури, яка на ній знаходиться, для дошки – ще масив клітинок на ній і т.д.

1. Які дії повинні виконувати об’єкти?

Вони повинні реагувати або не реагувати на натискання мишкою чи при настиканні відповідної клавіші клавіатури, відображатися, зникати, деякі ще повинні переміщуватись, причому по-різному.

1. Що ж є спільним для них усіх?

Серед параметрів – напевне, координати центра, ще колір і розмір по горизонталі та вертикалі, з дій – кожен із об’єктів повинен уміти створюватися, знищуватися та відображати себе на екрані. Таким чином, базовий клас буде містити такі поля даних: координати «х», «у», «ширина», «висота», «колір»; методи – конструктор, що створює об’єкт, деструктор, що його знищує, та метод «відобрази себе».

Оце й усе! Але описане лише один раз (!) для всіх видимих об’єктів.

1. Далі можна розділити видимі об’єкти на пасивні та активні.

Пасивними будуть написи на дошці, що позначають назви рядів, активними – всі решта, вони повинні мати метод, який змушує їх реагувати на натискання мишкою. Тепер виділимо ще підклас фігур, що породжується від класу «активні видимі елементи», який має додатковий метод «перемісти мене» та поле даних або метод, що визначає допустимі ходи для даної фігури. Так, дошка містить 64 клітинки, а кожна клітинка – може містити якусь фігуру. Тому до даних цього класу, також породженого від «активних видимих елементів», слід додати ще поле даних (напевне, вказівник), який містить інший об’єкт(-и) класу – «видимі елементи». Також необхідні методи, які ці об’єкти «вставляють» у клас «Група» або видаляють з класу «Група». Метод «відобрази себе» для «Група» завдяки поліморфізму можна доповнити відображенням об’єктів «Група» після того, як відобразилася сама «Група». Далі від «Група» можна успадкувати нові класи: «Шахова дошка», «Шахове поле», а від «Шахового поля» – «Поле останнього рядка», яке крім методів свого попередника може ще й перетворювати «пішака» у «королеву».

Цей приклад, далеко не повний, він дає можливість зрозуміти лише основний підхід до проєктування класів. Пофантазувавши трохи, можна отримати завершену ієрархію об’єктів, які будуть уміти робити все, що від них може вимагати головна програма для розв’язування поставлених задач. Основне – кожна спільна для кількох об’єктів дія (і спільний елемент даних) описана лише один раз у базовому класі і більше ніде її не потрібно описувати повторно.

Етап «Розробка системи класів» відображає основну роботу, яка виконується при об’єктно-орієнтованому програмуванні, в результаті чого, отримуються описи всіх класів з даними та методами, причому для методів уже задані їх параметри та тип результату, який вони повертають. Тобто, з точки зору інтерфейсу, робота над класами на цьому етапі вже завершена. Залишилося тільки «навчити» методи виконувати свою роботу. Глобальні функції, які не є методами ні одного з класів, також поміщаються в один із модулів, а їх інтерфейсна частина описується в цьому підрозділі.

### Розробка методів

На даному етапі слід описати принцип дії основних методів та відмінності поліморфних методів від методів базових класів. Це завдання є досить простим, якщо на попередньому етапі було правильно спроєктовано систему класів та детально пророблено інтерфейсні частини.

Реалізація будь-якої функції (чи методу) – запис тих дій, які має виконувати метод чи функція, не забуваючи користуватися вже готовими функціями та методами замість того, щоб увесь час змінювати дані за допомогою одних і тих же послідовностей операцій.

Так, метод «перемісти мене» (див. приклад у попередньому пункті) повинен лише змінити координати активного видимого елемента, а не намагатися перемалювати його самостійно – метод «відобрази себе» зробить це набагато краще. Аналогічно метод «відобрази себе» для «групи» (шахової дошки чи клітинки), який повинен відображати об’єкти «групи» після того, як відобразилася сама група, реалізується з використанням поліморфізму та успадкування. При цьому додається лише один-два оператори, оскільки і група, і видимі елементи вже самі вміють відображатися, необхідно лише «сказати» їм, щоб вони це зробили один за одним. У методі «відобрази себе» групи спочатку викликається метод «відобрази себе» базового класу («видимий елемент»), який відображає саму дошку або клітинку, а далі – методи «відобрази себе» для всіх видимих елементів, що входять до групи. Для клітинки – це буде виклик «відобрази себе» для фігури, що стоїть на клітинці, а для дошки – виклик «відобрази себе» для всіх позначок та клітинок, які, у свою чергу, відобразять фігури, що на них знаходяться.

### Створення об’єктів і розробка головної програми

На даному етапі слід описати процес створення самих об’єктів у програмі, функціональність та дані яких розроблені на попередніх етапах. Якщо кількість об’єктів, що будуть створені наперед невідома, або обсяг пам’яті, яку вони займають, є досить значним, тоді пам’ять для об’єктів слід виділяти «динамічно».

Стосовно розглянутого прикладу із шахами, даний етап буде описувати – як створюється шахова дошка з 32 фігурами на ній та як перетворюються вибрані партнерами ходи в повідомлення шаховим фігурам. Створивши об’єкти, можемо реалізувати певну послідовність дій або схему взаємодії між об’єктами, яка призводить до вирішення поставленого завдання. Опис цієї послідовності дій чи схеми взаємодії, можливо, з прикладами для реальних вхідних даних чи повідомлень і є основою даного етапу. Такий опис повинен бути достатнім для розуміння всіх деталей реалізації алгоритму та можливих випадків, що зустрічаються при роботі програми.

### Опис файлів даних та інтерфейсу програми

Якщо програма використовує файли як джерело вхідних даних або для зберігання проміжних чи кінцевих результатів роботи, то на даному етапі слід навести опис формату цих файлів.

Особливо важливий даний етап для програм, що працюють із базами даних, адже структура таблиць бази даних, перелік, типи полів, засоби взаємодії з базою даних, перетворення даних є основою таких програм. Для інтерактивної програми з розвинутою системою меню та діалогових вікон у цьому етапі слід описати призначення елементів меню, роботу з ними, параметри, що вибираються в діалогових вікнах тощо. Тут же слід описати інтерфейс програм, які працюють з параметрами командного рядка.

1. **Вимоги до програми**

Незалежно від теми програма, що розробляється, повинна задовольняти наступним загальним вимогам.

* ***Стійкість програми***. Програма не повинна втрачати працездатності ні при яких, навіть некоректних, діях користувача. Всякі дії, що загрожують втратою інформації, мають бути підтверджені користувачем. Інформація, що вводиться, скрізь, де це можливо, піддається логічному забезпеченню цілісності даних. При будь-яких діях користувача не повинні втрачатися дані або їх цілісність (некоректність індексів, втрата посилань в зв’язках після видалення-додавання записів і т. д.).
* ***Функціональна повнота***. Мають бути реалізовані усі функції (методи), вказані в специфікації програми.
* ***Інтерфейс користувача***. Використовуються тільки терміни, зрозумілі користувачеві, і не використовуються терміни розробника («запис», «індексація» і т.д.). Поява службових англомовних повідомлень неприпустима. У повідомленнях користувача слід дотримуватися норм ввічливості, колірна гамма повинна наслідувати загальноприйняті рекомендації.
* ***Використання клавіатури.*** На будь-якому етапі натиснення будь-якої клавіші повинне ігноруватися або викликати передбачені дії, описані в засобах допомоги. Прив’язка дій до клавіш має бути загальноприйнятою: F1 – допомога; Entег – згода, завершення введення; Еsс – відмова, повернення до попереднього вузла гілки алгоритму (з відновленням екранної форми); Таb – перехід до наступного поля, вікна і т.д.; Shift & Tab – повернення до попереднього поля, вікна і т.д.

**Деталізація вимог**

1. **Загальні умови:**

* мова програмування C++.
* програмний продукт – консольний додаток.
* вхідні дані подаються користувачем з клавіатури в консоль.
* результати відображаються користувачу у консолі режимі.
* сховище даних – це текстовий файл, тому передбачити збереження даних у файл та зчитування даних з файлу.
* передбачити навігаційну панель (меню), яка повинна забезпечувати перегляд, редагування, додавання, видалення даних.
* передбачити сортування, фільтрацію та пошук даних по самостійно вибраному ключу.
* обов’язкова перевірка коректності вводу даних та обробка виключень.
* наявність «посібника користувача» – інструкції з використання програми.

1. **Умови програмної реалізації**

Програмна реалізація має складатися не менше ніж з 4-5 класів, серед яких передбачити класи-нащадки, абстрактні класи та класи-інтерфейси.

Крім завдань, що вказані в індивідуальній умові, передбачити реалізацію не менше 5-ти «власно-придуманих» методів згідно предметної області.

Всі класи повинні містити:

* конструктор для ініціалізації об’єкта;
* конструктор перезавантаження із параметрами – значення створюваних об’єктів вводяться з клавіатури і передаються в конструктори об’єктів у вигляді параметрів;
* деструктор для звільнення пам’яті (з повідомленням про знищення об’єкта);
* методи обробки даних, зазначені в індивідуальному завданні.

Всі поля класу зробити закритими (модифікатор доступу *private*), а доступ, ініціалізацію та зміну кожного поля виконати через властивості C++: *get/set*.

В класах-нащадках передбачити поліморфізм методів класу.

**6. ВИМОГИ ДО ОФОРМЛЕННЯ КУРСОВИХ ПРОЕКТІВ**

**6.1 Загальні вимоги**

Пояснювальна записка оформляється згідно стандарту ДСТУ 3008-95 «Документація. Звіти в галузі науки і техніки за допомогою комп’ютера». Для підготовки і друкування роботи рекомендується використовувати текстовий редактор Microsoft Word.

Курсова робота (проект) повинна бути написана літературною українською мовою без орфографічних та синтаксичних помилок. Використання автоматизованого перекладача без подальшого редагування роботи, а також комп’ютерного та іншого сленгів забороняється. Усі терміни, скорочення та позначення повинні бути пояснені та розшифровані.

Роздруковується робота з однієї сторони стандартного аркушу формату А4 (210297 мм), але допускається використання аркушів формату A3 (297х420 мм), коли це необхідно.

**Для всіх аркушів роботи необхідно дотримуватись таких розмів полів сторінки: лівий – 3 см., правий – 1,5 см., верхній і нижній – 2 см.**

**Орієнтація сторінки – книжкова.**

Текст і таблиці друкуються чорним кольором, рисунки і схеми – чорно-білі або в градації сірого.

У тексті пояснювальної записки не рекомендується вживати звороти із займенниками першої особи, наприклад: «Я вважаю …», «Ми вважаємо …» тощо. Рекомендується вести виклад, не вживаючи займенників, наприклад: «вважаємо …», «… знаходимо …» тощо. Без пояснень дозволяється використовувати тільки загально-прийнятті скорочення, наприклад: ПК, ДСТУ, ООП тощо.

Матеріал пояснювальної записки повинен бути викладений грамотно, чітко та стисло. При цьому в тексті записки мають бути обов’язковими посилання на використані літературні та інші джерела.

При скороченні слів і словосполучень потрібно спочатку навести повну назву, а після цього в дужках – її скорочення.

Порядкові числівники, які йдуть один за одним, можуть бути написані цифрами з відмінковим закінченням, яке ставлять лише при останній цифрі, наприклад: 1-е; або 7, 8, 9-й; тощо.

Фрагменти коду, що ілюструють певні елементи програмної реалізації, оператори, назви класів, методів тощо, що подаються в тексті, доцільніше для зручності читання набирати шрифтом Arial.

Графічний ілюстративний матеріал у тексті пояснювальної записки (схеми, ескізи, графіки, рисунки) виконують у графічному редакторі. Кількість ілюстрацій повинна бути достатньою для пояснення тексту, що викладається.

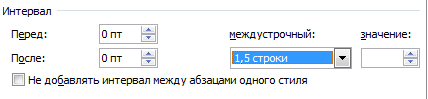
### 6.2 Вимоги до тексту

Для тексту, таблиць використовується:

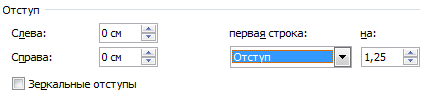
* вирівнювання тексту – по ширині аркуша;



* шрифт Times New Roman;
* розмір шрифту 14 пт;
* інтервал шрифту – «1,5 строки», «Интервал» після та перед рядком дорівнює 0 пт;



* масштаб – 100%;
* абзацний відступ дорівнює 1,25 см.



Забороняється застосовувати напівжирний шрифт. Курсивом можуть оформлюватися елементи формули в тексті, для однозначного трактування.

Наприклад: формула *С=А+В*. В тексті йде роз’яснення, що таке *А, В*, і *С*, тоді їх виділяють курсивом для наочності і розуміння відмінності мови подання.

### 6.3 Нумерація сторінок

Сторінки пояснювальної записки нумеруються арабськими цифрами **без знаку «№»** в правому верхньому куті дотримуючись наскрізної нумерації для всього тексту у форматі: 11 (без крапки в кінці).

Титульний аркуш включають у загальну нумерацію. Номер на титульному аркуші не ставлять. Титульний аркуш має номер 1, лист завдання, що має зворотній бік – має номер 2 тощо.

На титульному аркуші, листі завдання та анотаціях номери не ставляться, хоча самі аркуші враховуються у загальній кількості сторінок.

До загального обсягу роботи не входять додатки, список використаних джерел, таблиці та рисунки, які повністю займають площу сторінки. Але всі сторінки зазначених елементів роботи підлягають суцільній нумерації.

### 6.4 Оформлення назв, імен, власних імен

Прізвища, назви установ, організацій, фірм та інші імена власні в документі подають мовою оригіналу. Припускається транслітерувати власні імена і подавати назви організацій у перекладі мовою записки з додаванням (при першому нагадуванні) оригінальної назви.

### 6.5 Структурні елементи

Структурними елементами пояснювальної записки є «Лист завдання», «Реферат» (українською та англійською мовами), «Зміст», «Перелік умовних ПОЗНАЧЕНЬ» (за необхідністю), «вступ», «основна частина», «висновки», «СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ», «додатки».

Заголовки структурних частин роботи друкують великими літерами симетрично до набору.

Деякі елементи, а саме «РЕФЕРАТ» (українською та англійською мовами), «Зміст», «Перелік умовних ПОЗНАЧЕНЬ» (за необхідністю), «вступ», «основна частина», «висновки», «СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ», «додатки» створюють розділи пояснювальної записки і їхні найменування служать заголовками окремих розділів. Такі розділи не мають нумерації.

### 6.6 Оформлення заголовків

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

Заголовки структурних елементів і заголовки розділів слід розташовувати по середині рядка і друкувати великими літерами без крапки наприкінці, не підкреслюючи.

Заголовки підрозділів, пунктів і підпунктів варто починати з абзацного відступу і друкувати малими літерами, крім першої великої, не підкреслюючи, без крапки наприкінці. Якщо заголовок складається з двох або більше речень, їх розділяють крапкою. Переноси слів у заголовку розділу не припускаються.

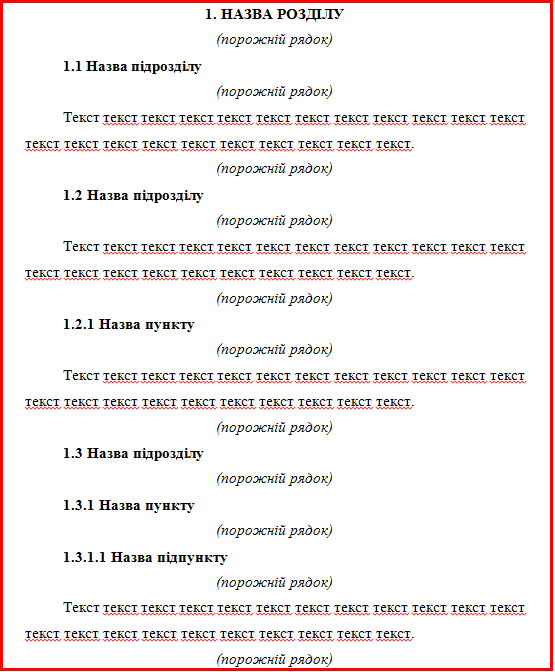
Відстань між заголовком і наступним або попереднім текстом повинна бути не менше одного порожнього рядка.

Кожний розділ починається з нового аркушу.

Не припускається розміщувати найменування розділу, підрозділу, а також пункту і підпункту в нижній частині сторінки, якщо після нього розташований тільки один рядок тексту.

Підрозділ, пункт, підпункт не може закінчуватися рисунком, таблицею, схемою, обов’язково після них має бути до двох рядків тексту.

*Наприклад:*



### 6.7 Нумерація заголовків

Нумерацію розділів, підрозділів, пунктів, підпунктів, подають арабськими цифрами без знака №. Після номеру розділу, підрозділу, пункту, підпункту перед їхньою назвою крапку не ставлять.

Такі структурні частини роботи, як ЗМІСТ, ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ВСТУП, ВИСНОВКИ, СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ, ДОДАТКИ не мають порядкового номера розділу. Звертаємо увагу на те, що всі аркуші, на яких розміщені згадані структурні частини роботи, нумерують звичайним чином. Не нумерують лише їх заголовки, тобто не можна друкувати: «1.ВСТУП» або «Розділ 6. ВИСНОВКИ».

Номер підрозділу складається з номеру розділу і порядкового номеру підрозділу, що розділяються крапкою.

*Наприклад:* 1.2 Назва другого підрозділу 1-го розділу

Номер пункту складається з номеру розділу, номеру підрозділу і порядкового номеру пункту, що розділяються крапкою.

*Наприклад:* 1.2.1 Назва першого пункту другого підрозділу 1-го розділу

Номер підпункту складається з номеру розділу, номеру підрозділу, номеру пункту і порядкового номеру підпункту що розділяються крапкою.

*Наприклад:* 1.2.1.5 Назва 5-го підпункту, 1-го пункту другого підрозділу, 1-го розділу

### 6.8 Складання переліку посилань

Перелік посилань складається згідно ДСТУ 7.1:2006. Бібліографічні описи в переліку посилань наводять у порядку першої зустрічі в тексті. Посилання на друковані джерела мають подаватися в квадратних дужках [ ]. Не допускається розривати ініціали і прізвище між двома рядками.

Якщо в посиланні на книгу вказують її загальний обсяг сторінок, то пишуть так: «... – 1098 с». Якщо в посиланні на книгу вказують її вибірковий обсяг сторінок, то пишуть так: «... – С. 5-15.».

Не допускається розташовувати на окремому рядку кількість сторінок, або номер ISBN.

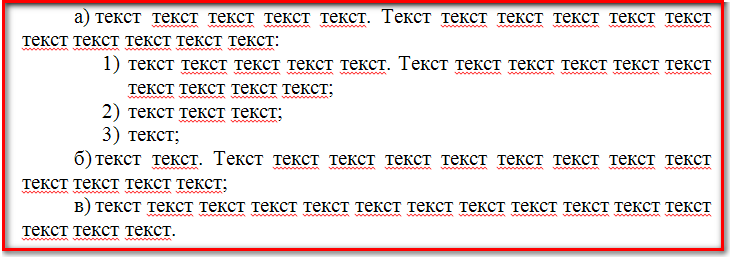
Приклад складання переліку посилань та бібліографічного опису наведено у додатку.

### 6.9 Перелічення

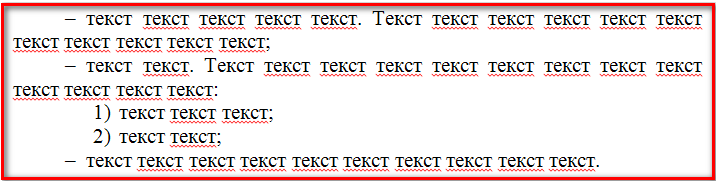
За необхідністю можуть бути використані перелічення. Перед переліченням ставлять двокрапку. Перед кожною позицією перелічення слід ставити малу літеру українського алфавіту з дужкою або, без нумерації – тире (перший рівень деталізації). Для подальшої деталізації перелічення слід використовувати арабські цифри з дужкою (другий рівень деталізації).

Перелічення першого рівня деталізації друкують маленькими літерами з абзацного відступу, другий рівень деталізації – з відступом відносно положення перелічення першого рівня (під текстом першого рівня деталізації).

*Наприклад:*



***Або***



У переліченнях, які розпочинаються із літер або цифр в кінці кожного перелічення можуть стояти знаки пунктуації, такі як: «.» (крапка) або «;» (крапка з комою). Якщо текст починається з малої літери, то в кінці кожного перелічення повинна стояти «;» (крапка з комою), крім останнього, де завжди крапка. Якщо ж текст починається з великої літери, то «.» (крапка) ставиться в кінці кожного перелічення.

У переліченнях, які розпочинаються із символу «–» завжди ставиться «;» (крапка з комою), крім останнього, де завжди крапка.

### 6.10 Оформлення таблиць

Таблицю слід розташовувати безпосередньо після тексту, в якому вона згадується вперше або на наступній сторінці по центру аркуша. На таблицю мають бути посилання в тексті.

Таблиці слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком таблиць, що наводять у додатках. Номер таблиці складають із номера розділу і порядкового номера таблиці, розділених крапкою.

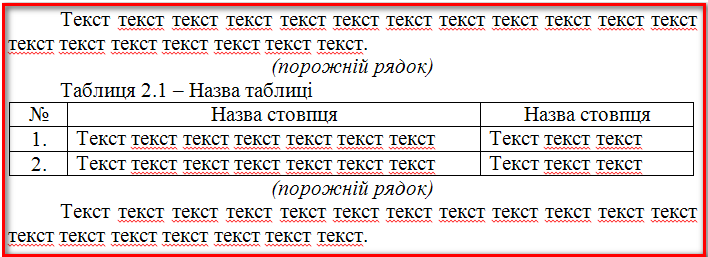
Таблиця може мати назву, яку поміщають над таблицею.

Назва стовпців таблиці розміщується по центру комірки таблиці. Якщо рядки або графи таблиці виходять за формат сторінки, таблицю поділяють на частини, розміщаючи одну частину під іншою, або поруч, або переносячи частину таблиці на наступну сторінку. При цьому в кожній частині таблиці повторюють її головку і боковик. При розподілі таблиці на частини припускається її головку і боковик заміняти відповідно номерами граф.

Слово «Таблиця <номер>» вказують один раз з абзацу ліворуч над першою частиною таблиці, над іншими частинами з абзацу ліворуч пишуть: «Продовження таблиці <номер>».

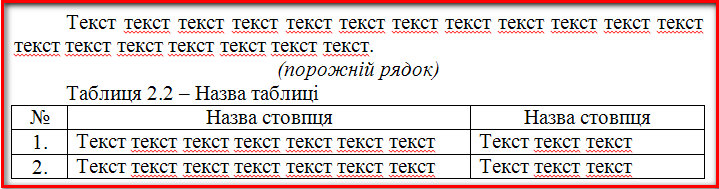
Відстань між таблицею та основним текстом становить один порожній рядок.

*Наприклад:*

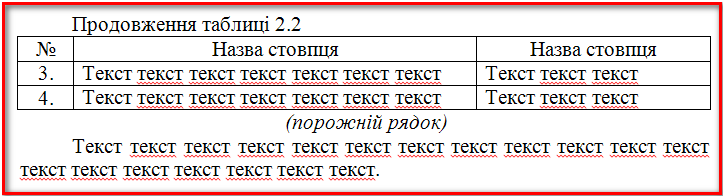


Якщо таблиця має розрив, то це оформляється так:

на 1-му аркуші



на 2-му аркуші



### 6.11 Оформлення рисунків

Ілюстрації (креслення, рисунки, графіки, схеми, діаграми, фотознімки) слід розташовувати в записці безпосередньо після тексту, в якому вони згадуються або на наступній сторінці. Якщо ілюстрація після посилання на неї не вміщується на сторінці, її розміщають на наступній сторінці, або у додатку якщо вона повністю займає один і більше аркушів формату А4.

На всі ілюстрації мають бути посилання в пояснювальній записці. Креслення, рисунки, графіки, схеми, діаграми мають відповідати вимогам стандартів ЕСКД і ЕСПД.

Ілюстрації можуть мати назву, яку розміщують під ілюстрацією по центру сторінки. Ілюстрація позначається словом «Рисунок <номер> – Назва рисунку», що разом із назвою ілюстрації розміщують після пояснень.

Ілюстрації слід нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком ілюстрацій, що наводяться в додатках. Номер ілюстрації містить номер розділу і порядковий номер ілюстрації, що розділені крапкою.

За необхідності під ілюстрацією розміщують роз’яснювальні дані (підрисунковий текст).

*Наприклад:*



Рисунок 3.1 – Схема розміщення

У додатках у номер рисунка входить літера додатка.

*Наприклад:*

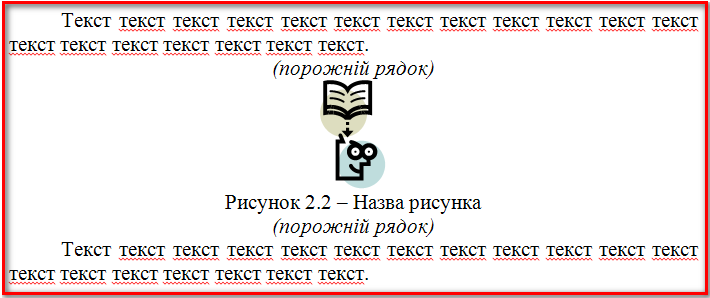


Рисунок А.1 – Графік продажу продукту

Якщо ілюстрація не вміщується на одній сторінці, можна переносити її на інші сторінки, при цьому назву ілюстрації розміщують на першій сторінці, а пояснення на кожній сторінці, і під ними вказують: «Рисунок \_\_, аркуш \_\_».

Відстань між ілюстрацією та основним текстом становить один порожній рядок.

*Наприклад.*



Якщо роз’яснення мають розповідний характер, то посилання на рисунок пишуть так: ... на рисунку 2.2 намальований круг.

Якщо йде роз’яснення і необхідне посилання на рисунок, то в кінці абзацу або в реченні ставиться посилання на рисунок у круглих дужках, наприклад, .... (див. рис.2.2).

В пояснювальній записці зазвичай дотримуються ідентичного оформлення назв рисунків (за повною або скороченою формою). Особливо необхідно приділити увагу чіткості ілюстрації, щоб читач без всякої напруги для очей зміг побачити окремі деталі. Особливо це стосується подання схем БД, всіляких діаграм, що розроблені в інших графічних редакторах і при вставці в текстовий редактор втрачають чіткість зображення.

### 6.12 Оформлення формул та частин програмного коду

Формули і рівняння розташовуються безпосередньо після тексту по центру сторінки. Відстань між основним текстом і формулою становить один рядок.

Формули і рівняння треба нумерувати арабськими цифрами порядковою нумерацією в межах розділу, за винятком формул і рівнянь, наведених у додатках.

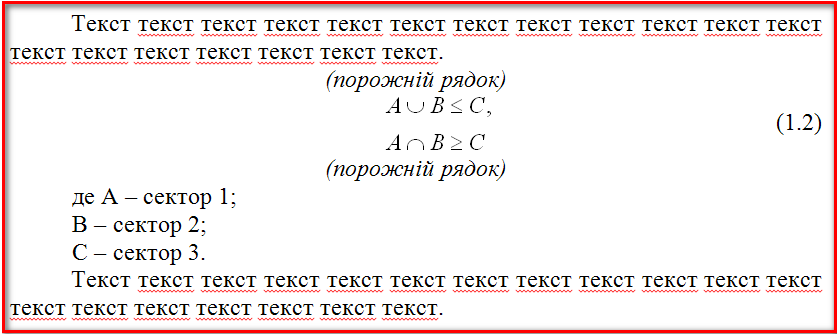
Номер формули або рівняння складається з номера розділу і порядкового номера формули або рівняння, розділених крапкою. Номер формули або рівняння вказують на рівні формули або рівняння в круглих дужках у крайньому правому положенні на рядку.

Формули набираються в редакторі формул. Формули які слідують одна за одною і які не розділені текстом розділять комою. Зручніше формулу розміщувати в невидиму таблицю.

Пояснення символів і числові коефіцієнти, що входять до формули або рівняння, варто робити безпосередньо після формули або рівняння, в тій послідовності, у якій вони надані у формулі або рівнянні. Пояснення значення кожного символу або числового коефіцієнта необхідно давати з нового рядка. Перший рядок пояснення починають з абзацу словом «де» без двокрапки.

Переносити формули або рівняння на інший рядок припускається тільки на знаках виконуваних операцій, причому знак операції на початку такого рядка повторюють. При переносі формули або рівняння на знаку операції множення застосовують знак «х».

*Наприклад:*



Частина програмного коду, за необхідністю може бути наведена в записці у вигляді тексту, якщо роз’яснення мають розповідний характер, або у вигляді рисунку, якщо є роз’яснення програмного механізму з посиланнями на частину коду.

*Наприклад:*

Наведемо програмну реалізацію функції динамічного розподілу пам’яті під двовимірний масив:

*int \*\*malloc2d (int r, int c)*

*{*

*int \*\*t = new int \*[r];*

*for (int i=0; i<r; i++)*

*{*

*t[i] = new int[c];*

*return t;*

*}*

*}*

***Або***

Покажемо програмну реалізацію цієї версії: Приклад наведено на рисунку 4.3.

*(порожній рядок)*

|  |
| --- |
| int gcd (int m, int n)  {  If (n==0) return m;  return gcd (n, m % n);  } |

Рисунок 4.3 – Рекурсивна версія алгоритму Евкліда

*(порожній рядок)*

## 6.13 Скорочення

У тексті звіту припускається робити тільки загальновживані скорочення (наприклад «і т.д., і т.п., та ін.») або поширені абревіатури.

Якщо в звіті прийнята специфічна термінологія, або вживаються малопоширені скорочення, нові символи, позначення і ін., їх перелік необхідно надати в окремому списку. Перелік повинен розташовуватися стовпчиком, у якому слова за абеткою наводяться скорочено, а справа – їхнє докладне тлумачення.

Якщо спеціальні символи, терміни, скорочення, і позначення наводяться менше трьох разів, перелік їх не складається, а їхнє тлумачення роблять у тексті при першому згадуванні.

**6.14 Оформлення додатків**

Додатки необхідно оформляти в окремій частині, розташовуючи додатки в порядку появи посилань на них у тексті роботи.

Кожний додаток повинен починатися з нової сторінки. Зверху по-центру друкується слово: **ДОДАТОК <ЛІТЕРА>. Назва додатку**

Додатки слід позначати послідовно прописними буквами українського алфавіту, за винятком букв Ґ, Є, З, І, Ї, Й, О, Ч, Ь. Один додаток позначається як одна літера.

Заголовок додатку друкують по центру, великими літерами, жирним шрифтом, не підкреслюючи, без крапки наприкінці.

Додатки повинні мати загальну з іншою частиною роботи наскрізну нумерацію. Текст кожного додатка, при необхідності, може бути розділений на підрозділи і пункти, що нумеруються арабськими цифрами в межах кожного додатка. У цьому разі перед кожним номером ставлять позначення додатка (літеру) і крапку, наприклад: А.2 – другий розділ додатка А; В.3.1 – перший підрозділ третього розділу додатка В.

Ілюстрації таблиці та формули, розміщені в додатках, нумерують у межах кожного додатка, наприклад: Рисунок Д.1.2 – Назва рисунку або Формула (А.1) – Назва формули.

**Машинний код програми, який поміщають у додаток, друкують з міжрядковим інтервалом – Одинарний, розмір – 8 пт.**

**7. ПІДГОТОВКА ДО ЗАХИСТУ**

Виконаний згідно зі стандартами відповідно до завдання і у повному обсязі **курсовий проєкт**, підписаний **виконавцем**, у не зброшурованому вигляді **треба подати** на перевірку **керівникові** проекту.

Студент подає наступні матеріали виконаного курсового проєкту: пояснювальну записку в друкованому та електронному вигляді, вихідний код програмного додатка, скомпільований виконуваний файл основної програми та допоміжних програм-утиліт (якщо такі є), об’єктні модулі (для випадку розроблення модулів або бібліотеки), презентацію доповіді для захисту в електронному вигляді, відео-ролик. Якщо програма вимагає для своєї роботи додаткових динамічних бібліотек (DLL), які не входять до складу операційної системи, то їх також необхідно додати або вказати назву та джерело з якого можна зчитати ці файли.

Виявлені при перевірці курсового проєкту неточності й помилки студент зобов’язаний виправити, а результати представити керівникові у визначені терміни. Якщо ж при огляді встановлено, що робота в будь-якій частині потребує суттєвого доопрацювання, то визначається обсяг доопрацювання і встановлюється термін подання виправленої роботи на повторну перевірку.

Роботи, що не відповідають затвердженій темі, без затвердженого завдання на курсовий проєкт, підписаного студентом і викладачем, а також ті, в яких виявлено запозичення з інших джерел, без посилання на джерело до захисту не допускаються.

Керівник перевіряє відповідність завдання до індивідуального варіанту, правильність виконання завдань, структуру та повноту пояснювальної записки.

Після чого робота повинна пройти етап нормоконтролю – перевірка документації на відповідність правилам оформлення.

Якщо робота відповідає поставленим умовам та поставленим правилам оформлення – керівник проставляє на титульному аркуші відмітку «До захисту», поточну дату та підпис.

Робота готова до представлення комісії.

**8. ПОРЯДОК ЗАХИСТУ**

**До захисту курсового проєкту допускаються студенти, роботи яких отримали допуск від керівника проєкту, з виставленими всіма підписами та датами допуску.**

На захист роботу слід представляти тільки в зброшурованому вигляді.

До пояснювальної записки додати електронний варіант на оптичному носії (прикріпленому до останньої сторінки обкладинки). На електронному носії необхідно вказати прізвище, ініціали студента, групу (вмістиме електронного носія наведено в додатку П).

Захист курсового проєкту відбувається при повному виконанні всіх розділів, і представляється для комісії у вигляді презентації розробленої програмної системи.

Захист курсового проєкту полягає в:

1. представленні звіту з роботи – звіт має відповідати всім вимогам до оформлення (п.6 даних методичних рекомендацій), містити всі необхідні розділи (п.4 даних методичних рекомендацій), та відповідати обраній тематиці та індивідуальному завданню (п.3 даних методичних рекомендацій);
2. демонстрації роботи – робота демонструється на великому екрані з використанням технічних засобів розробки та представлення демонстрацій, у презентацію повинні бути включені: тема, завдання, алгоритми, скрін-шоти програми, результати роботи, висновки. Інформація на слайдах повинна легко читатись, мати контрасний колір та чіткі графічні зображення, скрін-шоти програми – розташовані на повний екран, текст – містити лише основні положення (додаток Н);
3. представленні програми – давати пояснення по практичному використанню програми; виконання всіх вказаних дій згідно інструкції користувача; орієнтуватися в алгоритмі виконання функціональних дій згідно індивідуального завдання.
4. поясненні коду програми – давати пояснення по коду програми, орієнтуватися у коді для пошуку виконання необхідних функцій, вносити корективи в програмний код для отримання інших результатів, тощо.

Захист курсового проєкту проходить у такій послідовності:

* доповідь студента про основні результати виконаної роботи;
* відповіді студента на запитання присутніх;
* обговорення доповіді;
* відповіді на зауваження.

**Для доповіді** про результати виконаної роботи студенту **надається 2-3 хвилини**.

Доповідь повинна описувати, що саме студентом було зроблено.

Відповіді на запитання повинні бути короткими, за суттю й не виходити за межі поставленого запитання.

Оцінювання курсового проєкту проводиться за кредитно-модульною системою оцінювання, прийнятою для оцінювання знань студентів в університеті, та розподіляється по змістових модулях у відповідності до табл.3.

Курсовий проєкт оцінюється за критеріями, що наведені в табл.4. Та відповідає вимогам шкали оцінювання ECTS – табл.5.

При визначенні оцінки за курсовий проект береться до уваги рівень теоретичної й практичної підготовки студента.

Таблиця 3 – Розподіл балів за виконання курсового проєкту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ змісто-вого моду-ля** | **Елементи змістового модуля** | **Кількість балів** | | **Поточний контроль навчальної роботи студентів** |
| **мініма-льна** | **макси-мальна** | **методи контролю** |
| **1.** | Пункти розрахунково-пояснювальної записки: |  |  |  |
| «Структура програми» | 8 | 15 | Перевірка розділу |
| «Скролінг (текст) програми» | 6 | 10 | Перевірка розділу |
| «Опис методів програми» | 6 | 10 | Перевірка розділу |
| **ВСЬОГО** | **20** | **35** |  |
| **2.** | Графічна частина – форми та об’єкти програми | 8 | 15 | Перевірка графічної частини |
| Результати виконання | 10 | 20 | Перевірка роботи програми |
| **ВСЬОГО** | **18** | **35** |  |
| **3.** | Захист курсового проєкту: |  |  | Захист роботи у комісії |
| Презентація + відео-ролик | 6 | 15 |  |
| Доповідь | 6 | 15 |  |
| **ВСЬОГО** | **12** | **30** |  |
| **СУМА** | | **50** | **100** |  |

Таблиця 4 – Критерії оцінювання курсового проекту

|  |  |
| --- | --- |
| **Критерії** | **Бали** |
| Робота виконана повністю, представлена й вигляді цілісного проекту, оформлена у відповідності до вимог та вчасно захищена. | **90…100 балів** |
| Робота виконана повністю, є несуттєві відхилення від повнофункціонального представлення цілісного проекту, оформлена у відповідності до вимог з незначними неточностями та вчасно захищена. | **70…89 балів** |
| Робота виконана повністю, є відхилення від повнофункціонального представлення цілісного проекту, оформлена у відповідності до вимог з незначними неточностями та / або захищена з запізненням. | **50…69 балів** |
| Робота виконана частково, є суттєві недоліки у оформленні та невчасно захищена. | **0…49 балів** |

Таблиця 5 – Шкала оцінювання ECTS

|  |  |  |  |
| --- | --- | --- | --- |
| **Оцінка в балах** | **Оцінка за національною шкалою** | **Оцінка за шкалою ECTS** | |
| **Оцінка** | **Пояснення** |
| 90-100 | Відмінно | А | Відмінно (відмінне виконання лише з незначною кількістю помилок). |
| 80-89 | Добре | В | Дуже добре (вище середнього рівня з кількома помилками). |
| 70-79 | С | Добре (в цілому правильне виконання з певною кількістю суттєвих помилок). |
| 60-69 | Задовільно | D | Задовільно (непогано, але зі значною кількістю недоліків). |
| 50-59 | Е | Достатньо (виконання задовольняє мінімальним критеріям). |
| 35-49 | Незадовільно | FX | Незадовільно (з можливістю повторного складання). |
| 0-34 | F | Незадовільно (з обов’язковим повторним проходженням курсу). |