

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ім. Б. ХМЕЛЬНИЦЬКОГО**

**Факультет обчислювальної техніки, інтелектуальних
та управляючих систем**

**Кафедра програмного забезпечення
автоматизованих систем**

О. О. Супруненко, Ю. Є. Гребенович

МЕТОДИЧНІ ВКАЗІВКИ
ДО ВИКОНАННЯ ТА ОФОРМЛЕННЯ
КУРСОВОЇ РОБОТИ
з дисципліни
“Об’єктно-орієнтоване програмування”

*для студентів,
які навчаються за спеціальністю
121 Інженерія програмного забезпечення
галузі знань 12 Інформаційні технології
усіх форм навчання*

Черкаси 2021

УДК 004.42 + 004.021
ББК

Супруненко О.О., Гребенович Ю.Є. Методичні вказівки до виконання та оформлення курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» для студентів, які навчаються за спеціальністю 121 Інженерія програмного забезпечення галузі знань 12 Інформаційні технології усіх форм навчання. – Черкаси: Вид. від. ЧНУ імені Богдана Хмельницького, 202. – 38 с.

Рецензенти:

О.Б. Данченко, доктор технічних наук, професор, професор кафедри комп'ютерних наук та системного аналізу Черкаського державного технологічного університету,

Б.О. Онищенко, кандидат фізико-математичних наук, доцент кафедри програмного забезпечення автоматизованих систем Черкаського національного університету імені Богдана Хмельницького.

Профільна кафедра: кафедра обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут»;

Затверджено Науково-методичною радою факультету обчислювальної техніки, інтелектуальних та управляючих систем, як методичний посібник для організації роботи студентів над завданнями першої курсової роботи студентів денної, заочної та екстернатної форм навчання, які проходять підготовку за спеціальністю 121 Інженерія програмного забезпечення, галузі знань 12 Інформаційні технології, протокол № 5 від 17.12.2021.

.

УДК 004.42 + 004.021

ББК

Рекомендовано до друку Вченою радою Черкаського національного університету імені Богдана Хмельницького
(протокол № _ від «__» _____ 202 р.)

ЗМІСТ

Вступ	4
1. Індивідуальне завдання на курсову роботу	4
2. Етапи виконання курсової роботи.....	5
3. Структура та зміст пояснювальної записки до курсової роботи..	6
3.1. Загальні рекомендації до оформлення розділів	8
3.2. Рекомендації до змісту першого розділу курсової роботи...	9
3.3. Приклади оформлення інформаційних джерел.....	13
3.4. Формування блок-схем алгоритмів та структурної схеми програми (до другого розділу курсової роботи).....	15
3.5. Об'єктна модель системи. Класи: спадкування та інкапсу- ляція (до другого розділу курсової роботи)	18
3.6. Приклади оформлення третього розділу з реалізації програмного продукту	24
3.7. Порядок формування матеріалів у курсовій роботі.....	29
4. Написання доповіді та створення презентації до захисту курсвої роботи	30
5. Порядок рецензування курсової роботи.....	32
6. Перелік типових тем курсових робіт.....	33
Список рекомендованої літератури	34
Internet-посилання.....	35
Додаток А. Титульний лист курсової роботи	36

ВСТУП

Виконання програмних проектів у фаховій підготовці інженерів-програмістів займає центральне місце. **Курсова робота** є *видом самостійної роботи студента*, що спрямована на набуття практичних навичок по підбору алгоритмів реалізації поставленого завдання і створенню на їх основі програмних продуктів. Також робота передбачає розвиток творчого мислення та навичок кодування для реалізації практичних задач на основі теорії алгоритмів, технологій об'єктно-орієнтованого програмування. Курсова робота оформлюється у вигляді пояснювальної записки та оптичного диску з кодом програмного продукту.

Таким чином, **метою виконання курсової роботи** є набуття практичних навичок з *початкової обробки завдання* (1), *проектування* (2) та *реалізації* (3) програмного проекту, що передбачений темою курсової роботи, на мові високого рівня. За результатами проекту створюється технічна документація, якою в курсовій роботі є **пояснювальна записка**. Дані методичні вказівки призначені для допомоги студентам у виконанні всіх цих завдань.

1. Індивідуальне завдання на курсову роботу

Підставою для виконання курсової роботи є «**Індивідуальне завдання на курсову роботу**». Воно містить тему курсової роботи, вимоги до програмного компоненту (функціонал), вимоги до інтерфейсу, перелік розділів пояснювальної записки до курсової роботи, перелік наочних матеріалів та календарний план виконання задач курсового проектування. Індивідуальне завдання видається студенту керівником. *Перед початком роботи керівником і студентом обговорюються всі необхідні вимоги до проекту та зовнішні представлення компонентів програмного продукту.*

2. Етапи виконання курсової роботи

Робота виконується за чітко визначеними етапами:

- 1) студент *отримує тему та індивідуальне завдання* до курсової роботи і *обговорює завдання* з керівником курсової роботи;
- 2) проводиться огляд технічної інформації за темою роботи, це може бути інформація з Internet-сайтів, фахових журналів та книг (див. список літератури), аналогічні програмні продукти в демоверсії чи доступних версіях; у цих джерелах *знаходяться алгоритми для реалізації поставленого завдання та приклади вдалих і невдалих реалізацій* програм, подібних до отриманого завдання;
- 3) виконується *аналіз* оглянутої технічної *інформації*, *обираються алгоритми* та програмні рішення для реалізації отриманого завдання;
- 4) розробляється блок-схема алгоритму реалізації завдання;
- 5) розробляється структурна чи функціональна схема програмного продукту, описуються його елементи;
- 6) розробляється інтерфейс програмного продукту;
- 7) проводиться *кодування та тестування* розробленого програмного продукту;
- 8) по вище проробленим завданням *оформлюється пояснювальна записка*;
- 9) *представляється програма і пояснювальна записка керівнику* курсової роботи для написання відгуку;
- 10) одночасно з попереднім пунктом *пишеться доповідь та формується презентація* до захисту курсової роботи;
- 11) проводиться *корегування роботи* (доповіді і презентації) за зауваженнями керівника (*щоб зауважень було мінімум, потрібно всі незрозумілі питання по ходу виконання роботи вирішувати з керівником!*);
- 12) захист курсової роботи.

В ході виконання курсової роботи студенти мають навчитися

- 1) виконувати огляд алгоритмів та методів реалізації поставленого завдання,

2) обирати вдалі проектні рішення, проводити проектування невеликих програмних продуктів, 3) створювати за завданнями програмні продукти з використанням мов програмування високого рівня, 4) грамотно оформлювати та представляти отримані результати.

3. Структура та зміст пояснювальної записки до курсової роботи

Пояснювальна записка до курсової роботи *оформляється* на листах формату А4, формується у текстовому редакторі Word (14 кегль, полуторний інтервал, всі поля по 20 мм). Титульний лист оформлюється за зразком, наведеним у додатку А.

Обсяг пояснювальної записки обумовлюється її змістом і орієнтовно складає 12-18 сторінок друкованого тексту (без врахування додатків) з формулами, малюнками і таблицями, що займають менше 1-ї сторінки (всі інші малюнки та таблиці виносяться у додатки). Структура та зміст курсової роботи формується у наступній послідовності:

- 1) **вступ** (1 сторінка), в якому обґрунтовується актуальність теми, визначається мета та завдання роботи, наводяться базові поняття, що стосуються обраної теми;
- 2) **розділ 1*** – **огляд інформаційних джерел** (2-4 сторінок), в якому проводиться огляд алгоритмів та методів реалізації отриманого завдання, проводиться їх аналіз та вибір конкретних алгоритмів та методів реалізації програми, робляться висновки в яких формується ідея програмного проекту (див. п. 3.1);
- 3) **розділ 2*** – **проектування програмного продукту** (3-5 сторінок), в якому проводиться опис алгоритмів для реалізації задачі, формується

* При оформленні розділів **назва кожного розділу формується студентом у залежності від викладеного матеріалу** (!!! не допускається такі назви розділів, як теоретична частина, чи практична частина !!!).

* При оформленні розділів **назва кожного розділу формується студентом у залежності від викладеного матеріалу** (!!! не допускається такі назви розділів, як теоретична частина, чи практична частина !!!).

блок-схема основного та допоміжних алгоритмів; формується узагальнена структурна чи функціональна схема програмного продукту; робиться висновок, в якому ставиться задача на розробку програмного продукту по зробленому проекту;


- 4) **розділ 3*** – **реалізація програмного продукту** (3-5 сторінок): проводиться опис реалізації завдання у програмному коді, який може доповнюватися схемами, діаграмами та невеликими ділянками лістингу програми з необхідними коментарями (на всі ілюстративні матеріали мають бути посилання в основному тексті, наприклад: див. рис. 7, де відображено схему реалізації блоку обчислень координат руху фішки); опис результатів роботи програми та тестування доповнюється копіями екрану; у висновку описується реалізований функціонал та можливі результати застосування;
- 5) **висновки** (1 сторінка) – у висновках висвітлюється функціонал програмного продукту, який був створений на реалізацію поставленого завдання; коротко описується суть програмних рішень, що були застосовані; наводяться сфера застосування та шляхи розвитку програми;
- 6) **список використаних джерел** (1 сторінка) – список літературних та Internet-джерел (при посиланні на Internet-джерела вказується назва джерела, його адреса та дата перегляду матеріалу (див. пункт 3.3)). Список джерел складається у порядку їх згадування в роботі.
- 7) **додатки** – за необхідності громіздкі малюнки, таблиці, граф-схеми алгоритмів та інше, що *займають одну цілу сторінку формату А4 чи більше*, виносяться у додатки; кожен додаток починається з нової сторінки, нумерується великими українськими літерами (за винятком букв *Г, Є, З, І, Ї, Й, О, Ч, Ї*) і має назву в залежності від вмісту (див. додатки А-В).

Наприклад: Зміст курсової роботи:

Вступ.....	2
Розділ 1. Огляд алгоритмів роботи з циклічними списками та сортування великих обсягів вхідних даних.....	3
1.1. Алгоритми роботи з циклічними списками.....	4
1.2. Ефективні алгоритми сортування.....	6
1.3. Аналіз алгоритмів для реалізації задачі складання поліномів.....	7
1.4. Висновок до першого розділу.....	8
Розділ 2. Проектування програмного модуля для операцій над поліномами	9
2.1. Блок-схема алгоритму розщеплення циклічного списку.....	11
2.2. Блок-схема алгоритму множення поліномів	14
2.3. Структура програмного модуля для операцій над поліномами.....	16
2.4. Висновок до другого розділу.....	18
Розділ 3. Реалізація та тестування модуля для операцій над поліномами.....	19
3.1. Реалізація основних функцій програмного модуля.....	19
3.2. Реалізація користувацького інтерфейсу	21
3.3. Тестування програмного модуля для операцій над поліномами...	24
3.4. Висновок до третього розділу.....	26
Висновки.....	27
Список інформаційних джерел.....	28
Додатки	
Додаток А. Блок-схема модуля перетворення поліномів.....	29
Додаток Б. Узагальнена структурна схема програмного модуля для операцій над поліномами	30

3.1. Загальні рекомендації до оформлення розділів

Кожен **розділ** роботи починається з окремого листа. Опис завдань проводиться у текстовому редакторі Word, 14 кегль, полуторний інтервал, всі поля по 20 мм. Заголовки розділів набираються 16 кеглем напівжирним шрифтом, інші заголовки - 14 кеглем напівжирним шрифтом. Відступ перед і після заголовку - 12 пт.

Формули набираються у редакторі формул  і нумеруються – нумерація в пояснювальній записці наскрізна. Формули розміщуються по центру листа, мають відступ 3 інтервали (1 пустий рядок з полуторним інтервалом) до та після формули, номер формули ставиться у дужках по лівому краю листа (рис. 1).

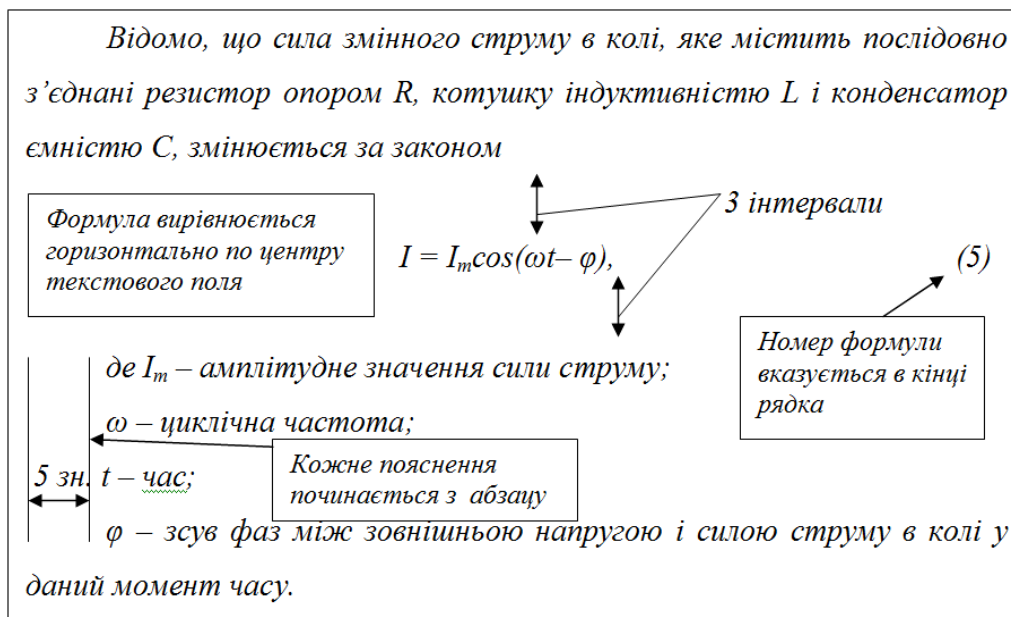


Рис.1. Правила оформлення формул у тексті курсової роботи [14].

Рисунки виконуються у окремій області (наприклад: «Рисунок Word» чи в окремій області малюнка), можуть виконуватися у інших додатках типу Visio чи CorelDraw і т.і. Рисунки нумеруються та підписуються (нумерація наскрізна у межах розділу, наприклад: рис. 2.3 – це 3-й малюнок 2-го розділу), у тексті документу мають бути посилання на кожний виконаний в роботі малюнок. Після підпису рисунка залишається простір (3 інтервали чи 1 пустий рядок з полуторним інтервалом) до наступного за ним тексту (рис. 2). Якщо рисунок не поміщається на тій сторінці, на якій на нього посилаються, його розміщують на початку наступної сторінки.

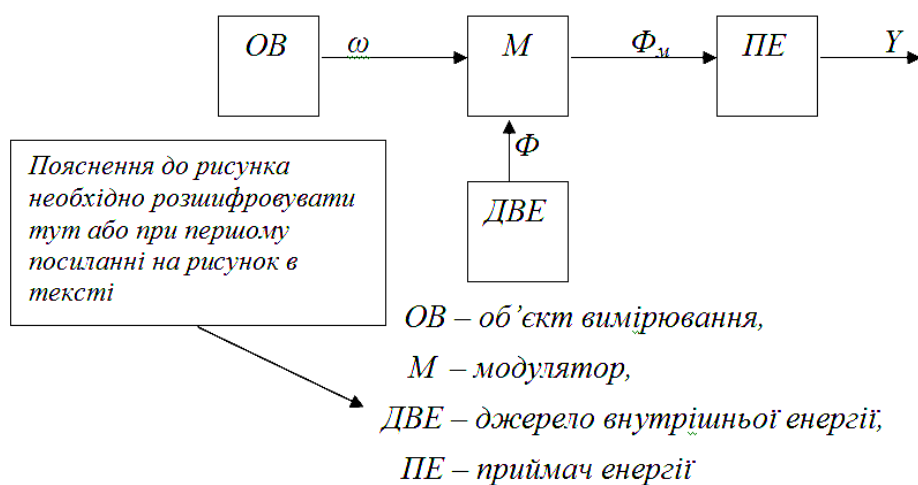


Рис 3. Приклад оформлення рисунка

3 інтервали

Рис.2. Правила оформлення рисунків у тексті курсової роботи [14].

Таблиці розміщуються по центру листа симетрично до тексту після першого посилання на даній сторінці або на наступній, якщо на даній вона не поміщається і таким чином, щоб зручно було її розглядати без повороту, або з поворотом на кут 90° за годинниковою стрілкою [14]. Назва таблиці пишеться над нею по центру, номер таблиці на наступному рядку з вирівнюванням по правому краю. Наприклад:

Елементи граф-схеми алгоритму та їх характеристики.

Таблиця 1.4

--	--	--	--	--

У верхній (чи лівій) частині таблиці розміщують заголовок таблиці, в яком вказують назви граф. Діагональний поділ комірок таблиці не допускається. Назви граф пишуть малими буквами, починаючи перше слово графи з великої, у підграфах назви повністю пишуть малими буквами, в кінці крапку не ставлять. Текст в таблицях пишуть з *одинарним інтервалом*.

Якщо всі параметри величин, які наведені в таблиці, мають одну й ту ж одиницю фізичної величини, то наприкінці назви таблиці після коми розміщують її скорочене позначення (*мм*). Якщо ж параметри мають різні одиниці фізичних величин, то позначення одиниць записують в заголовках граф після коми (*Довжина, мм*).

Дані, що наводяться в таблиці, можуть бути словесними і числовими. Слова записують в графах з однієї позиції. Якщо рядки таблиці не розділені лініями, то текст, який повторюється і складається з одного слова дозволяється замінювати лапками (,). Якщо текст складається з одного і більше слів, то при першому повторенні його замінюють словами “*те ж*”, а далі лапками. При розділенні таблиці горизонтальними лініями - ніякої заміни не виконують.

Числа записують посередині графи **так, щоб їх однакові розряди по всій графі були точно один під одним**, за винятком випадку, коли вказують інтервал. Інтервал вказують від меншого числа до більшого з тире між ними: *12 – 35 або 122 – 450*. Дробові числа наводять у вигляді десяткових дробів, з однаковою кількістю знаків після коми в одній графі. Ставити лапки замість цифр чи математичних символів, які повторюються, не можна. Якщо цифрові чи інші дані в таблиці не наводяться, то ставиться прочерк [14].

Якщо таблиця не поміщається на одній сторінці, її переносять на наступну, на якій повторюють заголовок таблиці, або під заголовком на початку таблиці нумерують рядки і цю цифрову нумерацію рядків розміщують в заголовку продовженої таблиці (рис. 3). Надписують продовження таблиці:

«Продовження таблиці 1.4»,
вирівнюють по правому краю
листа.

Назва таблиці.

Таблиця 1.4

Найменування елементу	Позначення елементу	Вхідні зв'язки	Вихідні зв'язки	Примітки
1	2	3	4	5

<на наступній сторінці>

Продовження таблиці 1.4

1	2	3	4	5

Рис. 3. Оформлення перенесення таблиць

У розділі роботи, який
стосується реалізації програм-
ного продукту, при опису
особливостей реалізації
можуть наводитися **ділянки**

лістингу, який доповнюється необхідними коментарями. Якщо частина лістингу перевищує розмір сторінки, такі ділянки коду розміщуються у додатках, на кожен з яких мають бути посилання в тексті роботи, де мова йде про елементи цих лістингів.

При виконанні курсової роботи студент користується індивідуальним завданням, інформаційними джерелами, рекомендованими керівником курсової роботи, конспектом лекцій з дисципліни «Об'єктно-орієнтоване програмування», рекомендованою літературою (див. п. «Список рекомендованої літератури») та Internet- посиланнями (див. п. «Internet-посилання»). При використанні літературних та Internet-ресурсів обов'язково в тексті роботи наводяться посилання, наприклад: [4, с.36-37]. У даному посиланні використовується 4-е джерело, в якому наведені факти розміщуються на сторінках 36-37. У списку використаних джерел книги та Internet-ресурси нумеруються в порядку їх використання в тексті курсової роботи.

Виконана робота роздруковується, підшивається і захищається студентом на засіданні комісії з захисту курсової роботи за розкладом, визначеним кафедрою.

Оцінка за курсову роботу приєднується до загального рейтингу оцінок студента, який враховується при видачі диплому бакалавра, також є складовою рейтингу при вступі на навчання до магістратури.

3.2. Рекомендації до змісту першого розділу курсової роботи

Перший розділ курсової роботи починається з формулювання *мети* та *задач** курсової роботи



Далі проводиться *огляд інформаційних джерел* чітко у відповідності з метою та сформульованими задачами. У висновках до розділу обираються алгоритми для реалізації завдання курсової роботи. Розберемо все послідовно.

При *формуванні мети* та *задач* слід керуватися визначеннями:

Мета роботи – це кінцевий результат, на досягнення якого вона спрямована.

Мета має адекватно відображатись у темі роботи, містити в узагальненому вигляді очікувані результати.



Задачі підпорядковуються меті роботи і спрямовані на послідовне (поетапне) її досягнення.

Приклад формування мети курсової роботи:

Метою даної курсової роботи є обрання алгоритмів та реалізація програмного модуля для операцій над поліномами.

Приклад формування задач курсової роботи:

1. Виконати аналіз алгоритмів для операцій над поліномами.
2. Сформулювати блок-схеми алгоритмів складання та множення поліномів.
3. Скласти структурну схему програмного модуля для операцій над поліномами.
4. Реалізувати у вигляді програмного продукту модуль для операцій над поліномами.
5. Зробити висновки щодо можливостей застосування розробленого програмного модуля для операцій над поліномами.

При огляді літературних та Internet-джерел студенту рекомендується:



1) переглянути обрані за списком рекомендованої літератури та за Internet-пошуком джерела, які містять інформацію про розв'язання поставлених задач або про певний алгоритм чи метод,

2) переглянути обраний матеріал, виділити джерела, у яких найбільш повно розкриті питання, що стосуються завдання курсової роботи,

3) за обраними джерелами провести опис (своїми словами) основного теоретичного матеріалу з посиланнями на оброблені джерела (виклад матеріалу студент має робити таким чином, щоб була зрозуміла його думка щодо викладеного матеріалу), можна використовувати схеми і малюнки з оглянутих інформаційних джерел, але обов'язково треба зробити посилання на ці джерела після назви рисунка чи схеми у формі: [5, с.304] (рисунок з 5-го джерела, зі сторінки 304);

4) по закінченні 1-го розділу потрібно зробити короткий висновок про доцільність обрання певних алгоритмів та методів для розв'язання поставленого завдання;

5) короткий висновок до розділу можна використовувати при формуванні загального висновку до курсової роботи.

3.3. Приклади оформлення інформаційних джерел

Список інформаційних джерел (літературних та Internet-джерел) складають за такими правилами:

1) джерело (книгу, журнал, електронний ресурс) наводять у списку мовою, на якій Ви його читали (українською, англійською чи ін.),

2) на початку опису джерела вказують основного автора, його ініціали, назву джерела,

3) в залежності від того, наводиться друковане видання чи електронне джерело, вказується його походження, відповідно [Текст]/[Електронний ресурс],

4) після слеш-риски вказують перелік інших авторів, якщо такі є, 5) вказують місто, видавництво, рік видання та загальну кількість сторінок (для книг); назву журналу (після //), номер, рік видання та початкову-кінцеву сторінки, на яких розміщено матеріал (для статей з журналів), дату, коли матеріал опрацьовувався (для електронних ресурсів).

При оформленні списку друкованих інформаційних джерел повну кількість сторінок вказують за зразком: 352 с. При вказівці вибраних сторінок журналу, на яких розміщується стаття вказівку оформлюють за зразком: С. 45-52.



Наприклад, 1) так оформлюється у списку літератури книга трьох авторів:

1. Ахо А. Структуры данных и алгоритмы [Текст] / А.Ахо, Дж. Хопкрофт, Дж. Ульман. – М.: Издательский дом «Вильямс», 2010. – 384 с.

2) так оформлюється у списку літератури стаття одного та двох авторів:

2. Драпак Л.С. Використання мови C# при проектуванні людино-машинного інтерфейсу [Текст] / Л.С. Драпак // Восточно-европейский журнал передовых технологий. – 2012. – № 4/2 (58). – С. 39-42.

3. Марковский А.П. Метод доопределения частично-заданных булевых функций для обеспечения их лавинных свойств. [Текст] / А.П. Марковский, Н.Н. Романец // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2009. – № 51. – С. 56-62.

При оформленні Internet-джерел важливо зафіксувати назву сторінки ресурсу, автора, якщо він вказаний, та електронну адресу сторінки ресурсу. Формувати у списку літератури посилання на Internet-джерел можна за прикладом:



4. Гавва А. «Адское» программирование. Ada-95. Компилятор GNAT. [Електронний документ]. Режим доступу: <http://ada-ru.org/V-0.4w/index.html>. Перевірено: 18.07.2012.

Дату перегляду Internet-джерела важливо вказувати, оскільки Internet-сторінки можуть оновлюватися, і потрібно знати, коли ви останній раз її переглядали. При посиланні на іноземні Internet-джерела можна керуватися одним з наступних прикладів:

5. Ada Reference Manual ISO/IEC 8652:2007(E) Ed. 3. [Електронний документ]. Режим доступу: <http://ocw.unican.es/ensenanzas-tecnicas/programacionen-lenguaje-ada/otros-recursos-2/RM-Final.pdf>. Перевірено: 18.07.2012.

6. Ada Reference Manual ISO/IEC 8652:2007(E) Ed. 3. Available at: <http://www.adapower.com/rm95/index.html> (accessed 18 July 2012).



3.4. Формування блок-схем алгоритмів та структурної схеми програми (до другого розділу курсової роботи)



Графічні засоби відображення структури та функцій програмного продукту служать для представлення і аналізу проектних рішень і є вдалим наочним засобом для командної роботи над програмним проектом.

Блок-схема (англ. block scheme) – графічне представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів [16]. У таблиці 1 представлені елементи блок-схеми алгоритмів, які відповідають окремим елементам алгоритму.

Таблиця 1.

Основні елементи блок-схеми алгоритму [16]

Назва	Позначення	Функція
1	2	3
Терміна-тор		Елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування - початок і кінець програми). Всередині фігури записується відповідна дія.
Процес		Виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

1	2	3
Рішення		Показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елемента. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елемента. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.
Зумовлений процес		Символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.
Дані		Перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).
Межа циклу		Символ складається з двох частин - відповідно, початок і кінець циклу - операції, що виконуються всередині циклу, розміщуються між ними. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу. Часто для зображення на блок-схемі циклу замість цього символу використовують символ рішення, вказуючи в ньому умову, а одну з ліній виходу замикають вище в блок-схемі (перед операціями циклу).
З'єднувач		Символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.
Коментар		Використовується для детальнішої інформації про кроки, процесу або групи процесів. Опис поміщається з боку квадратної дужки і охоплюється нею по всій висоті. Пунктирна лінія йде до описуваного елемента, або групи елементів (при цьому група виділяється замкнутою пунктирною лінією). Також символ коментаря слід використовувати в тих випадках, коли обсяг тексту в будь-якому іншому символі (наприклад, символ процесу, символ даних та ін) перевищує його обсяг.

Приклади представлення основних алгоритмічних конструкцій блок-схемами наведені на рис.4. Серед них:

- лінійна ділянка алгоритму (рис. 4, а),
- цикл з передумовою (цикл типу while) (рис. 4, б),
- цикл з постумовою (цикл типу do) (рис. 4, в),
- цикл з параметром (цикл типу for) (рис. 4, г),
- розгалуження у повній формі (розгалуження if) (рис. 4, д),

- розгалуження у неповній формі (розгалуження if-else) (рис. 4, *е*),
- множинне розгалуження (розгалуження if-elif-else) (рис. 4, *ж*) .

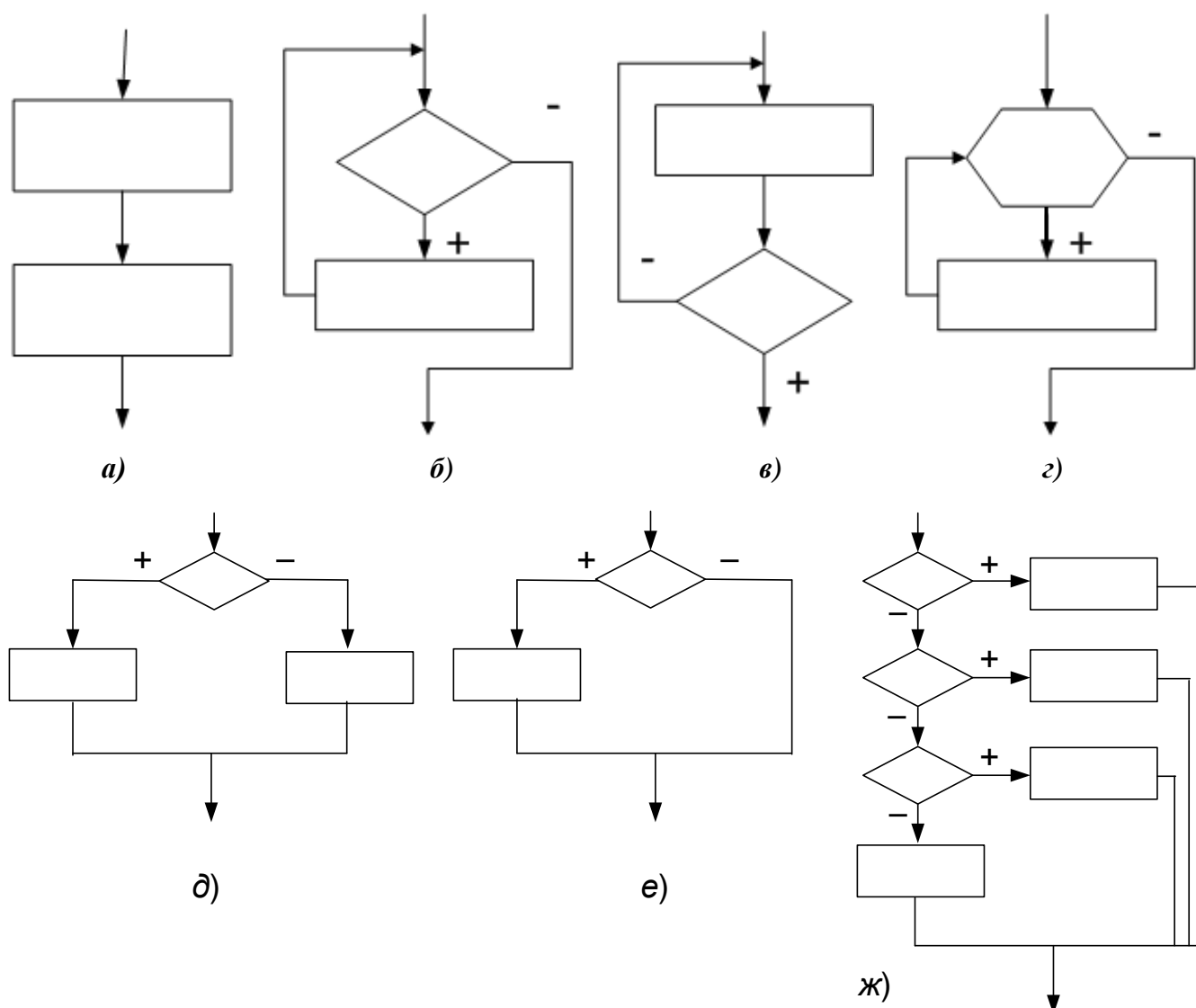


Рис. 4. Основні алгоритмічні конструкції, представлені блок-схемами: *а* – лінійна ділянка алгоритму, *б* – цикл з передумовою, *в* – цикл з посту мовою, *г* – цикл з параметром, *д* - розгалуження (повна форма), *е* - розгалуження (неповна форма) [17].

При виконанні курсової роботи студент має опрацювати алгоритми для реалізації завдання [2-12, Internet-джерела 1,4-5] та побудувати алгоритм (чи алгоритми) для їх подальшої практичної реалізації у вигляді програмного коду.

Проектування прикладної програми починається з **аналізу вимог**, яким вона повинна задовольняти. Такий аналіз проводиться з метою визначення призначення та умов експлуатації системи настільки, щоб скласти її початковий проект.

Під час використання об'єктно-орієнтованого підходу аналіз вимог до системи проводиться на моделях цієї системи, яка відображає її функції та якісні характеристики. Функції системи визначаються за переліком функціональних вимог, якісні характеристики та обмеження складають нефункціональні вимоги до програмної системи [20].

Моделлю програмної системи називають формальний опис системи, в якому виділені основні об'єкти, що є складовими системи і відношення між цими об'єктами. Побудова моделі базується на загальних, найбільш суттєвих елементах даної системи, зв'язках між ними.

Існує кілька технологій об'єктно-орієнтованої розробки прикладних програмних систем, в основі яких лежить побудова та інтерпретація моделей, з використанням комп'ютера. Одна з таких технологій – Object Modeling Techniques (OMT). Ця технологія передбачає представлення програмної системи у вигляді трьох взаємозв'язаних моделей:

- об'єктної моделі, яка зображує статичні, структурні аспекти системи, в основному зв'язані з даними;
- функціональної моделі, яка розглядає взаємодію окремих частин системи;
- динамічної моделі, яка описує роботу окремих частин системи.

Усі три види моделей дозволяють отримати три взаємно-ортогональні представлення системи в одній системі позначень. Сукупність моделей може бути інтерпретована на комп'ютері, що дозволяє продемонструвати функціонування майбутньої системи.

Моделі, розроблені і відлагоджені на першій фазі життєвого циклу системи продовжують використовуватися в усіх наступних фазах, полегшуючи програмування системи, її налагодження і модифікацію у майбутньому. Моделі системи не пов'язані з мовою програмування, засобами якої буде реалізована система.

3.5. Об'єктна модель системи. Класи: спадкування та інкапсуляція (до другого розділу курсової роботи)

Об'єктна модель системи. Об'єктна модель описує структуру об'єктів, що складають систему, їх атрибути, операції, взаємозв'язки з іншими

об'єктами. В об'єктній моделі повинні бути відображені ті поняття та об'єкти реального світу, які є важливими для системи, що розробляється.

Перерахуємо основні етапи побудови об'єктної моделі:

- визначення об'єктів та класів;
- підготовка словника даних;
- визначення відношень між об'єктами;
- визначення атрибутів об'єктів та зв'язків;
- організація та спрощення класів під час використання спадкування;
- дослідження та вдосконалення моделі надалі.

Визначення класів. Аналіз зовнішніх вимог до прикладної системи на етапі проектування дозволяє виділити об'єкти та класи об'єктів. Усі класи (узагальнені) мають бути сформульовані в термінах предметної області, класів, пов'язаних з комп'ютерною реалізацією (таких як стек, список і т.д.) на цьому етапі вводити не потрібно.

Визначення класів слід почати з письмової постановки прикладної задачі (технічного завдання та іншої документації). Це дуже важливий етап проектування, від нього залежить подальша доля проекту.

Під час визначення класів кожному іменнику, що зустрічається в попередній постановці задачі може відповідати клас. Тому найчастіше так і визначають класи на початковому етапі.

Далі список можливих класів аналізується на наявність непотрібних класів. Такими класами є:

- надлишкові класи: якщо два або декілька класів визначають одну інформацію, тоді треба зберігати тільки один з них;
- нерелевантні класи (що не мають прямого відношення до проблеми): для кожного імені можливого класу оцінити необхідність в майбутній системі;
- нечітко визначені (з точки зору проблемної задачі) класи;
- атрибути (деяким іменникам краще відповідають не класи, а атрибути, наприклад, вік, стать і т.і.);
- операції, деяким іменникам більше відповідають операції, а не класи (наприклад, телефонний виклик);
- ролі, деякі іменники визначають імена ролей в об'єктній моделі (наприклад, керівник, водій, службовець і т.д. – все це ролі одного класу - людина);

- конструкції реалізації (не потрібно вводити на даному етапі класи для позначення понять з алгоритмізації та програмування).

Після виключення усіх непотрібних (зайвих) класів ми отримаємо попередній список класів, що утворюють дану систему.

Підготовка словника даних. Окремі слова мають дуже багато інтерпретацій. Тому попередньо треба на початку проектування підготувати словник даних, що буде містити чіткі визначення усіх об'єктів (класів), атрибутів, операцій, ролей та інших сутностей, що використовуються у проекті.

Визначення відношень. Цей етап передбачає визначення відношень між класами. Перш за все, з класів виключаються атрибути, що є явними посиланнями на інші класи. Такі атрибути замінюються відношеннями. Суть такої заміни в тому, що відношення інколи є абстракцією такого ж рівня як і класи, тому не впливають на майбутню реалізацію (посилання на клас лише один із способів реалізації відношень).

Аналогічно тому, як імена можливих класів ми отримували з іменників у попередній постановці завдання, так і імена відношень можуть бути отримані з дієслів, що зустрічаються там же.

Потім потрібно прибрати непотрібні або неправильні відношення, використовуючи наступні критерії:

- відношення між виключеними класами мають бути знищені або переформульовані у термінах тих класів, що залишилися;
- нерелевантні відношення і відношення, пов'язані з реалізацією, повинні бути виключені;
- тернарні відношення – більшу частину відношень між трьома та більшим числом класів можна розкласти на декілька бінарних відношень;
- похідні відношення – потрібно виключити відношення, які можна виразити через інші відношення, оскільки вони надлишковими.

Уточнення атрибутів. Цей етап передбачає уточнення системи атрибутів: корегуються атрибути класів, вводяться за необхідності інші, нові атрибути. Атрибути виражають властивості об'єктів класу або визначають їх поточний стан. Атрибути зазвичай відповідають іменникам, наприклад, колір авто, позиція курсору і т.і. Атрибути зазвичай слабо впливають на структуру об'єктної моделі.

Разом з атрибутами об'єктів необхідно ввести і атрибути відношень між класами. Під час уточнення атрибутів використовують наступні критерії:

- заміна атрибутів на об'єкти, якщо наявність деяких сутностей важливіше, ніж їхнє значення, то це об'єкт, якщо важливіше значення – то це атрибут;
- кваліфікатори, якщо значення атрибута залежить від конкретного контексту, то його треба вважати кваліфікатором;
- ідентифікатори, ідентифікатори об'єктів пов'язані з їх реалізацією, на ранніх стадіях проектування їх не потрібно розглядати як атрибути;
- атрибути зв'язків, якщо деяка властивість характеризує не сам об'єкт, а його зв'язок з іншим об'єктом чи об'єктами, то це атрибут зв'язку, а не атрибут об'єкту;
- внутрішні значення, атрибути, що визначають лише внутрішній стан об'єкта треба виключити з розгляду;
- несуттєві деталі, атрибути, що не впливають на виконання більшої частини операцій, варто опустити.

Організація класів з використанням спадкування. На наступному етапі необхідно знайти суперкласи для введених класів, що допоможе уточнити структуру моделі і полегшить наступну її реалізацію.

Узагальнення – це відношення між більш загальною сутністю, яка називається суперкласом, та її конкретним втіленням, яке називають підкласом. Іноді узагальнення називають відношенням типу «являється, є» , маючи на увазі, що одні сутності (наприклад: коло, квадрат, трикутник) є втіленням більш загальної сутності (наприклад: геометрична фігура). При цьому, всі атрибути та операції суперкласу, незалежно від модифікаторів видимості, входять у склад підкласу.

Узагальнення часто називають спадкуванням. На діаграмах воно позначається не зафарбованою трикутною стрілкою (рис. 5). Для того, щоб ефективно виділити спадкування Греді Буч [20] рекомендує:

1) знайдіть атрибути, операції та обов'язки, загальні для двох і більше класів з даної сукупності – це дозволяє

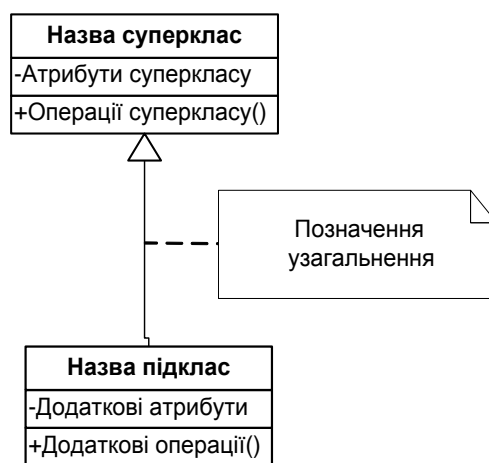


Рис. 5. Узагальнення.

позбавитися непотрібного дублювання структури та функціональності об'єктів,

2) внесіть ці елементи в деякий загальний суперклас, а якщо такого не існує, то створіть новий клас,

3) відмітьте у моделі, що підкласи спадкують від суперкласу, встановивши між ними відношення узагальнення.

Наведемо приклад застосування спадкування (рис. 6):

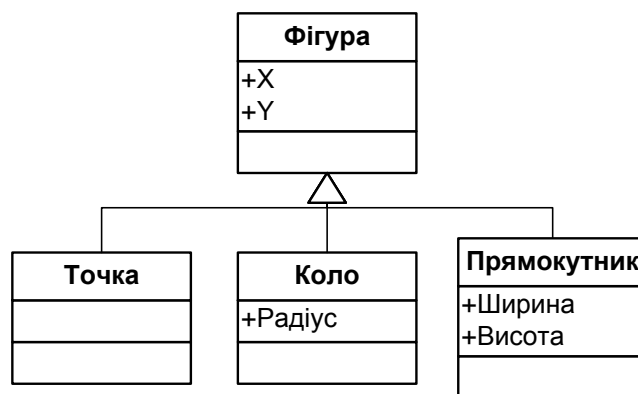


Рис. 6. Приклад спадкування.

На перший погляд дивно, що клас «Точка» не має ніяких атрибутів, а коло має тільки радіус. Прямокутник визначений шириною і висотою. Але пояснити це можна так: положення фігури можна визначити за допомогою пари чисел, для точки – це єдині характеристики, для кола і прямокутника – це точка їх центру, таким чином утворився суперклас «Фігура», що має загальні атрибути X та Y – координати центру фігури. Інші класи, що спадкуються від класу фігура потрібно довизначити: для кола це атрибут «радіус», для прямокутника – «ширина» та «висота». Операції для цих класів визначаються подібно атрибутам.

Інкапсуляція при визначенні атрибутів та операцій класів. Приховування від користувача (від дійсного користувача, чи від програміста, який використовує готовий клас) внутрішнього влаштування об'єктів називається інкапсуляцією. За офіційним визначенням [20], інкапсуляція – це захист окремих елементів об'єкту, що не зачіпають його суттєвих характеристик як цільного елементу. Інкапсуляція використовується не тільки для того, щоб створити ілюзію простоти для користувача, а й для захисту самого об'єкту. Розглянемо інкапсуляцію на прикладі телевізора, зовні нам цей прилад здається дуже простим, оскільки для роботи з ним ми використовуємо простий інтерфейс – пульт дистанційного керування. Для того, щоб збільшити гучність, потрібно натиснути відповідну кнопку з «+», а для перемикавання на

інший канал – натиснути кнопку з номером каналу. Як телевізор влаштований усередині, ми не знаємо, більш того, якби не було пульта керування, це було б незручним і небезпечним для нас, а також небезпечним і для телевізора. (Збільшувати гучність, наприклад, за допомогою паяльника – незручний і небезпечний спосіб). Тому, пульт телевізора захищає від нас внутрішню будову телевізора – так в реальному світі реалізується інкапсуляція.

У програмуванні інкапсуляція забезпечується дещо по-іншому – за допомогою так званих модифікаторів видимості. За їх допомогою можна обмежити доступ до атрибутів та операцій об'єкту з боку інших об'єктів. Якщо атрибут чи операція помічені модифікатором **private**, то доступ до них можливо отримати тільки з операції, визначеної у тому ж класі. Якщо ж атрибут чи операція помічені модифікатором видимості **public**, то до них можна отримати доступ з будь-якої частини програми. Модифікатор **protected** дозволяє доступ тільки з операцій цього ж класу та класів, що створюються на його основі (спадкуються). У мовах програмування можуть зустрічатися модифікатори видимості, які обмежують доступ на більш високому рівні, наприклад, до класів чи їх груп, однак суть інкапсуляції від цього не змінюється. У UML атрибути і операції з модифікаторами доступу позначаються спеціальними символами, що розміщуються зліва від їхніх імен:

Символ	Значення
+	public – відкритий доступ
–	private – доступ тільки з операцій того ж класу
#	protected – доступ тільки з операцій того ж класу і класів, що створені на його основі

На прикладі з телевізором можна побудувати узагальнений клас (на найвищому рівні абстракції), який буде виглядати, як зображено на рис. 7.

Телевізор
+Мова екранного меню
-Частота каналів
+Порядок та іменування каналів
-Самодіагностика()
+Увімкнути()
+Вимкнути()
+Пошук каналів()
-Декодування сигналу()
+...()

Рис. 7. Приклад застосування символів інкапсуляції.

3.6. Приклади оформлення третього розділу з реалізації програмного продукту

В пункті „ОПИС ІНТЕРФЕЙСУ” описується процес роботи з програмним продуктом. Наводяться всі реалізовані меню та підменю, та, в разі необхідності, пояснення до них. Цей розділ бажано доповнити роздруківкою відповідних екранних форм, що реалізовані в програмі.

Вимоги до графічного інтерфейсу користувача:

1. Назви елементів інтерфейсу повинні виконуватися українською або англійською мовою.
2. Головне вікно додатка – форма з такими елементами:
 - панель меню з підтримкою "акселераторів";
 - користувацька піктограма системного меню;
 - панель інструментів з підтримкою спливаючих "підказок" для кнопок;
 - рядок стану, в якому має відображатися інформація про основні режими роботи додатка.
3. Дані бази (якщо вона є) повинні відображатися у табличному вигляді.
4. Наявність модального діалогового вікна "Про програму" з інформацією про розроблювача програми, зокрема з його (її) фотографією.

Вимоги до архітектури програми:

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

Вимоги до функціональності додатка:

1. Створення файла бази даних (ім'я файла бази та каталог файлової системи для його збереження вибираються користувачем з використанням відповідного діалогового вікна).
2. Читання всіх даних з файла бази (каталог файлової системи та ім'я файла бази вибираються користувачем з використанням відповідного діалогового вікна) та їх відображення.
3. Додавання елементу даних до файла бази.
4. Оновлення будь-якого елемента даних у файлі бази.

5. Видалення будь-якого елемента даних з файла.
6. Сортуння 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.
10. Видача користувачу попереджуючих та інформаційних повідомлень.

Вимоги 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).
3. Виконання угод щодо запису тексту програм мовою програмування.

Якщо студент вибрав тему курсового проекту самостійно, то вимоги до програми обговорюються з керівником та можуть відрізнятися від наведених вище.

Приклад опису основних функцій програмного модуля інтелектуальної гри “Сапер”

Для того, щоб реалізувати спроектовані алгоритми, програма повинна мати чітку і прозору архітектуру. Щоб реалізувати таку архітектуру, в якій всі функції чітко розділені між класами, доцільно використати ідею про розділення логіки і інтерфейсу програми.

В першому наближенні ми повинні мати певну структуру для представлення мінного поля в пам'яті комп'ютера. Така структура повинна бути гнучкою і логічною, щоб з нею легко можна було зв'язати функції користувацького інтерфейсу.

Використовуючи переваги ООП підходу, програмну структуру можна описати в термінах предметної області.

Першим типом, що створений для даної програми є злічуваний тип `CellState`, що реалізує множину можливих станів комірки: закрита, відкрита і маркована комірка.

Для реалізації абстракції комірки використовується тип `Cell`, що містить інформацію про кількість замінованих сусідів, наявність міни а також поточний стан комірки у вигляді поля типу `CellState`.

Нарешті, для реалізації абстракції мінного поля, клас головної форми містить поле, яке являється двовимірним динамічним масивом елементів типу `Cell`.

Уся логіка програми базується на тому, що користувач, здійснюючи хід, змінює внутрішній стан мінного поля. Після кожного ходу відбувається перемальовування карти з новими даними про відкриті і закриті комірки. Увесь вивід мінного поля відбувається на компоненті малювання `GameView`, що використовує функціонал `OpenGL`.

Особливим є перший хід. Мінне поле з заданими розмірами створюється під час натиснення кнопки «Нова гра» (після чого викликається метод `NewGame` головної форми, який і ініціалізує мінне поле). Після здійснення першого кроку, коли стають відомі координати за якими був здійснений перший крок, визивається алгоритм розстановки мін, що реалізований в методі `FirstMove` головної форми. І лише після цього виконується обробка першого ходу згідно стандартних правил гри. Реалізація різноманітних ігрових функцій описана в обробнику подій `GameView_MouseClick` головної форми. При необхідності даний метод викликає методи `FirstMove`, `GameOver`, `BFS` (від англ. `breadth-firstsearch`– пошук у ширину).

Метод `BFS` реалізує алгоритм розкриття вільної області за допомогою алгоритму на графах пошуку в ширину. Метод `GameOver` реалізує функції підрахунку і збереження рекордів, підриву усіх мін на полі і блокування мінного поля, щоб неможливо було зробити хід після програшу або завершення гри.

Приклад опису користувацького інтерфейсу

Інтерфейс користувача в сучасних популярних реалізаціях гри передбачає використання маніпулятора «миша» для виконання ходів. Натиснення на праву кнопку означає, що гравець хоче відкрити клітинку, над якою знаходиться курсор, натиснення лівої кнопки призводить до маркування/розмаркування клітинки над якою в момент натиснення знаходився курсор.

Також в таких іграх присутнє головне меню яке дозволяє почати нову гру, вибрати налаштування мінного поля, такі як розміри і кількість мін а також можливості перегляду таблиці рекордів та імен найкращих гравців.

Алгоритми малювання мінного поля зазвичай досить складні, так як передбачають інтенсивне малювання великої кількості об'єктів – закритих, відкритих клітинок, маркування у вигляді прапорців, виведення кількості замінованих сусідніх мін і тому подібне. Один із можливих підходів – реалізувати мінне поле, як двовимірний масив стандартних Windows-кнопок, але подібна реалізація працює надзвичайно повільно. Як альтернативний варіант можна розглянути мінне поле як одну велику канву, на якій відбувається малювання. Такий підхід хоч і має коректну швидкість промальовування, але він вимагає великої кількості процесорного часу, що може бути небажаним для кінцевого користувача, що працює у багатозадачній операційній системі. В такому середовищі найефективнішим методом малювання мінного поля є використання потужності графічного прискорювача для формування зображення.

Саме тому для реалізації цієї задачі була вибрана відкрита бібліотека OpenTK. OpenTK є набором портованих на .NET відкритих бібліотек, що дозволяють повніше використовувати апаратні можливості обладнання, на якому працює користувач: бібліотеки OpenGL, що дозволяє напряму програмувати графічний автомат сучасних графічних процесорів, OpenAL – для програмування апаратного звуку і OpenCL – бібліотеки, що являє собою Тюрінг-повне розширення для програмування на графічних процесорах.

Малювання мінного поля здійснюватиметься з використанням бібліотеки OpenGL.

Користувачський інтерфейс реалізований у вигляді головної форми, яка має головне меню, у якому відображаються пункти меню «Нова гра», «Налаштування», «Список рекордів» та «Вихід». У нижній частині вікна розміщено рядок стану, на якому відображається кількість маркованих мін.

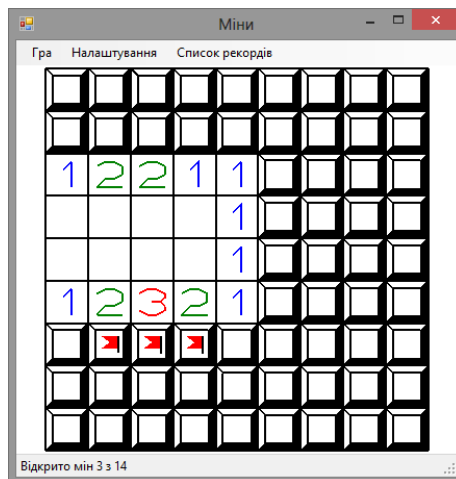


Рис. 8. Вигляд головного вікна програми

Всередині головного вікна розміщено палітру OpenGL, на якій і відбувається малювання мінного поля. Реалізація алгоритмів малювання елементів міститься в методах DrawClosedCell, DrawFlag, DrawNumber, DrawOpenedCell, DrawBomb. Координує малювання елементів обробник подій GameView_Paint. Відповідні елементи меню викликають меню налаштувань і таблиці рекордів.

Приклад опису керівництва користувача програмного продукту

Після першого ходу гравця повинна відкритись вільна від мін область.

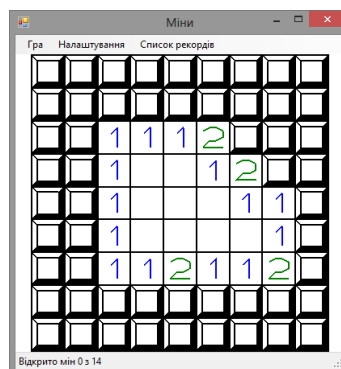


Рис. 9. Розкриття вільної області після першого ходу.

Рівні складності реалізуються зміною розмірів мінного поля і кількості мін. Гра має три рівні складності.

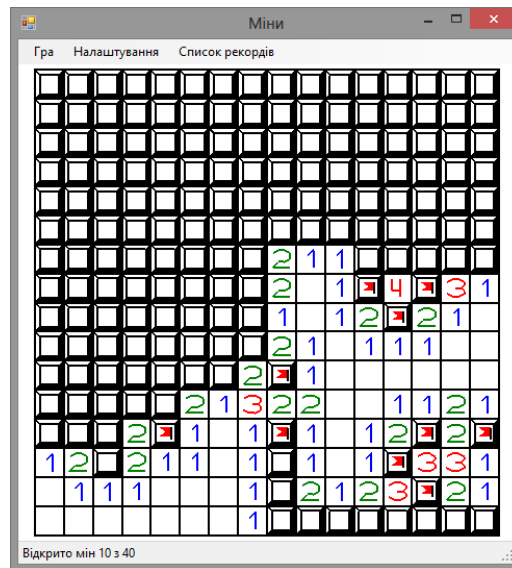


Рис. 10. Вигляд мінного поля при грі на важкому рівні складності

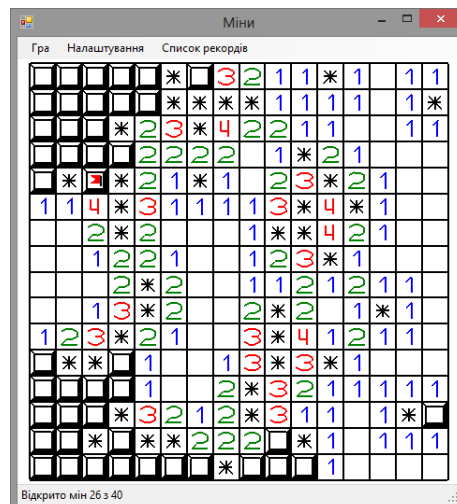


Рис. 11. Закінчення гри і відображення всіх мін.

Також, коли гравець програє, все мінне поле вибухає, також показуються неправильно зроблені ходи (маркування клітинок, що не містять мін).

3.7. Порядок формування матеріалів у курсовій роботі

При підготовці матеріалів курсової роботи до підшивання у папку потрібно скласти їх у такому порядку та пронумерувати сторінки (нумеруються не всі сторінки – див. список нижче по тексту):

- 1) титульний лист курсової роботи (не нумерується),

- 2) вступ (нумерація сторінок починається з 3 сторінки);
- 3) розділ 1 з відповідними пунктами, розміщеними у порядку нумерації (нумерація продовжується);
- 4) розділ 2 (і наступні ...) з відповідними пунктами, розміщеними у порядку нумерації (нумерація продовжується);
- 5) висновки (нумерація продовжується);
- 6) список використаних джерел (нумерація продовжується);
- 7) додатки у порядку, що визначений їх нумерацією: А, Б, В ... (сторінки додатків нумеруються окремо, починаючи з першої, але після номера вказується буквене позначення додатку, наприклад: 1А, 2А і т.д.);
- 8) посилання на репозиторій, у якому розміщується проект.

4. Написання доповіді та створення презентації до захисту курсової роботи

Приймаючи участь в наукових та практичних конференціях, ми можемо познайомитися з результатами інженерних розробок та наукових досліджень науковців з різних куточків нашої країни та зарубіжних колег і представити результати своїх наукових досліджень.

Для представлення наукових та практичних результатів потрібно підготувати *доповідь*. Зміст доповіді залежить від характеру роботи. Так, для представлення курсової роботи на захисті доповідь має таку структуру:

- *Шановні члени комісії та присутні, Вашій увазі пропонується курсова робота на тему: «...». Виконавець ..., керівник ...*
- короткий виклад мети та задач роботи,
- послідовний виклад результатів курсової роботи: обрані алгоритми та методи реалізації задач, граф-схеми алгоритмів з коментарями їх особливостей, структурна схема програмного продукту (всі положення ілюструються схемами, алгоритмами та ін. з пояснювальної записки),
- коротка характеристика створеного програмного продукту (результату курсової роботи), функціонал, інтерфейс користувача,

- практичне застосування результатів роботи, а також перспективи подальшого розвитку програми (якщо такі передбачаються).
- *Доповідь закінчена. Дякую за увагу!*

Доповідь розраховують на 4-5 хвилин. Доповідач має підготуватися до ймовірних запитань по курсовій роботі. Деякі з них він викликає сам, коротко говорячи про результати (але інакше в межах виділеного часу неможливо). Інші виникають по причині неякісного опрацювання теоретичного матеріалу чи неповного виконання завдання. Студент має підготуватися до відповіді на такі питання. Якщо не знаєте відповіді на поставлене запитання і цього не вимагалось індивідуальним завданням, можна сказати (там де це коректно), що така задача не ставилася.

Для викладення доповіді доповідач створює презентацію, яка містить ілюстративний матеріал до доповіді. Вона складається з послідовності слайдів, які можуть викладатися у наступній послідовності:

- ✓ **1 слайд.** Тема курсової роботи, виконавець та керівник.
- ✓ **2 слайд.** Мета та задачі, які були поставлені на курсову роботу.
- ✓ **3 ÷ n слайд.** Ілюстрування отриманих результатів (за розділами доповіді), це можуть бути таблиці, блок-схеми алгоритмів, структурні схеми та ін., на одному слайді не слід розміщувати більше 8-10 рядків тексту та 1-2 схем (блок-схем), кожен слайд повинен мати заголовок.
- ✓ **Останній слайд.** Висновки за отриманими результатами (коротко).
- ✓ **Додатковий слайд** (не обов'язковий). Перспективи і проблеми, які опрацьовує виконавець для подальшого розвитку теми роботи.

5. Порядок рецензування курсової роботи

Після повного виконання та оформлення пояснювальної записки до курсової роботи студент надає готовий матеріал на рецензування (для написання відгуку) керівнику курсової роботи.

Рецензія – (від лат. *recensio* – розгляд, обслідування) – це вид наукової, літературної і художньої критики, науково-критична стаття, що дає оцінку досліджуваному твору [14].

При рецензуванні курсової роботи керівник враховує ряд чинників: 1) відповідність змісту курсової роботи тому плану, який був зазначений у індивідуальному завданні, 2) ступінь виконання роботи, 3) якість ілюстрування роботи блок-схемами, схемами, рисунками, формулами, 4) якість оформлення пояснювальної записки (уміння підбирати і аналізувати фахову інформацію, керуватися нею при виконанні практичних завдань), 5) зазначає недоліки, якщо такі виявлені, 6) робить загальний висновок щодо допуску чи не допуску студента до захисту, виставляє оцінку керівника.

6. Перелік типових тем курсових робіт

Теми курсових робіт з дисципліни «Об'єктно орієнтоване програмування» мають відповідати ряду вимог. Дана курсова робота має на меті перевірити навички студентів у створенні простих програмних продуктів та оформленні супровідної документації. Тема має формуватися з врахуванням того, що студент має обрати та реалізувати не менше двох основних алгоритмів для реалізації завдання, а також побудувати простий інтерфейс програми (10-12 елементів). Нижче наведений перелік типових тем курсових робіт.

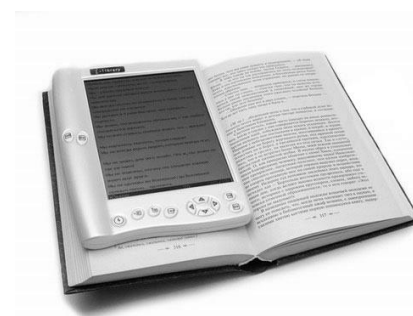
1. Українська мережа супермаркетів. (Створити програмний продукт, який при зменшенні кількості продуктів менше критичної маси автоматично відправляє замовлення (електронною поштою) постачальнику продукції).
2. Мережа зоопарків. (Створити програмний продукт, який веде облік тварин та при додаванні нової тварини у зоопарку, призначає її працівникові, у якого найменше тварин даного виду).
3. Їдальня. (Створити програмний продукт, який дозволяє формувати меню для їдальні на день. При чому не дозволяється змішувати м'ясні та рибні блюда в одному замовленні, молочні блюда та оселедець. Після формування меню відправляти його клієнтам на електронну пошту).
4. Фірма по розведенню акваріумних рибок. (Створити програмний продукт, який веде облік риб, необхідної кількості корму, утримання у різних акваріумах хижих та нехижих риб. Відправляти повідомлення клієнтам про появу нового виду риб).
5. Доставка піци. (Створити програмний продукт, який дозволяє приймати замовлення на виготовлення піци, враховує час (не дозволяється брати замовлення на уже зайнятий час), при замовленні 10 штук, 11-а безкоштовно. Відправляти повідомлення про оформлення замовлення).

Список рекомендованої літератури

1. Андерсон Д. Дискретная математика и комбинаторика: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 960 с.
2. Вирт Н. Алгоритмы и структуры данных.– М.: ДМК_Пресс, 2011. – 272 с.
3. Левитин А.В. Алгоритмы: введение в разработку и анализ. : Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 576 с.
4. Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К. Алгоритмы: построение и анализ, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1296 с.
5. Макконнелл Дж. Основы современных алгоритмов. – М.: Техносфера, 2004. – 368с.
6. Кнут Д. Искусство программирования, том 1. Основные алгоритмы, 3-е изд. : Пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 720 с.
7. Кнут Д. Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд. : Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 832 с.
8. Кнут Д. Искусство программирования, том 3. Сортировка и поиск. – М.: Издательский дом «Вильямс», 2011. – 824 с.
9. Ахо А., Дж. Хопкрофт, Дж. Ульман. Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2010. – 384 с.
10. Седжвик Р. Фундаментальные алгоритмы на C++. ч. 1-5. Анализ / Структуры данных / Сортировка / Поиск : Пер. с англ. / Роберт Седжвик. - К.: Издательство «ДиаСофт», 2003. – 688 с.
11. Окулов С. М. Программирование в алгоритмах. – М.: БИНОМ, Лаборатория знаний, 2002. – 341 с.
12. Гудман С., С. Хидетниemi. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981.
13. Романовский И.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ. 2006. – 112 с.



14. Методичні рекомендації щодо оформлення курсових і магістерських робіт для студентів факультету інформатики НАУКМА – К.: Вид. НАУКМА, 2010. – 38 с.
15. Шейко В.М., Кушнарєнко Н.М. Організація та методика науково-дослідницької діяльності: Підручник. – 2-е вид., перероб. і доп. – К.: Знання-Прес, 2002. – 295 с.
16. Блок-схема. [Електронний документ]. Режим доступу: <http://uk.wikipedia.org/wiki/Блок-схема>. Перевірено: 14.12.2012.
17. Старикова М.Е. Составление блок-схем. [Электронный документ]. Режим доступа: <http://festival.1september.ru/articles/588437/>. Проверено: 16.12.2012.
18. Алгоритмические структуры. [Электронный документ]. Режим доступа: <http://inf1.info/flowchart>. Проверено: 16.12.2012.
19. Мейер Б. Основы объектно-ориентированного программирования [Электронный документ]. Режим доступа: <http://www.intuit.ru/department/se/oopbases/>. Проверено 20.01.2013.
20. Греди Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – СПб: Невский Диалект, 1998. – 560 с.



Internet – посилання

1. <http://algolist.manual.ru/> – велика гарно структурована колекція алгоритмів
2. <http://lib.mexmat.ru/books/780> – електронна бібліотека опікунської ради механіко-математичного факультету Московського держуніверситету.
3. <http://www.iqlib.ru/support/about.visp> – електронна бібліотека освітніх і просвітницьких видань.
4. ips.ifmo.ru/courses/coursesinfo/index.html - курс С. Е. Столяра «Введение в алгоритмику»
5. <http://neerc.ifmo.ru/wiki/index.php> – Станкевич А.С. Дискретная математика, алгоритмы и структуры данных.
6. <http://www.diary.ru/~eek/p67723918.htm> – математика в технічному університеті – учбові матеріали МВТУ ім. Н.Е.Баумана.

7. <http://www.bookarchive.ru/computer/programming/page/1/> – безкоштовна електронна бібліотека книг і журналів з інформатики і програмування.
8. <http://www.knigonosha.net/computer/> – електронна бібліотека «Книгоноша».
9. <http://habrahabr.ru/search/> – ресурс, що має багато інформації для програмістів.
10. <http://jurnalprogman.narod.ru> – Internet-журнал «ProgMan» присвячений програмуванню, в ньому розміщено кількість статей по програмуванню під Windows та Web, є уроки програмування на Delphi, C++ та ін.
11. <http://www.mzero.by.ru> – зміст журналу «Zero» – це статті з програмування та web-дизайну, огляди ПЗ та різних версій Windows.



ДОДАТОК А. Титульний лист курсової роботи

Черкаський національний університет імені Богдана Хмельницького

(повне найменування вищого навчального закладу)

Кафедра програмного забезпечення автоматизованих систем

(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни “Об’єктно-орієнтоване програмування”

НА ТЕМУ “...”

Студента (ки) 2 курсу, групи КС-
спеціальності **121 Інженерія програмного
забезпечення**

(прізвище та ініціали)

Керівник _____

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

Національна шкала: _____

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

м. Черкаси – 2022 рік

Навчально-методичне видання

Супруненко Оксана Олександрівна

Гребенович Юлія Євгенівна

**Методичні вказівки
до виконання та оформлення
курсової роботи з дисципліни
“Об’єктно-орієнтоване програмування”
для студентів,
які навчаються за спеціальністю
121 Інженерія програмного забезпечення,
галузі знань 12 Інформаційні технології
усіх форм навчання**

*Комп’ютерна верстка
О.О. Супруненко, Ю.Є. Гребенович*

Підписано до друку . Формат 60×84/16. Гарнітура Times
Папір офсет. Ум. друк. арк. 1,55. Тираж прим. Зам. №идавець і виготовник
Черкаський національний університет імені Богдана Хмельницького.
Адреса: 18000, м. Черкаси, бул. Шевченка, 81, кімн. 117,
тел. (0472) 37-13-16, факс (0472) 37-22-33,
e-mail: vydav@cdu.edu.ua, <http://www.cdu.edu.ua>
Свідectво про внесення до державного реєстру
суб’єкт видавничої справи ДК № 3427 від 17.03.2009 р.