МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет
науки і технологій**

Кафедра економічної інформатики

В авторській редакції

ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Навчально-методичні рекомендації
з виконання лабораторних робіт

ДНІПРО
2024

УДК 681.518(076.5)
Т 38

Упорядники:
Л. І. Лозовська, Л. М. Савчук, Т. О. Климкович

Електронний аналог
друкованого видання

Схвалено Групою забезпечення якості освітньої програми
«Комп'ютерні технології в бізнесі»
Протокол № 2 від 18 вересня 2024 р.,
«Інформаційні технології та моделювання в економіці»
Протокол № 2 від 19 вересня 2024 р.

Т 38 Технологія проєктування інформаційних систем : навчально-методичні рекомендації з виконання лабораторних робіт / упоряд. Л. І. Лозовська, Л. М. Савчук, Т. О. Климкович ; Укр. держ. ун-т науки і технологій. – Дніпро : УДУНТ, 2024. – 72 с.

Навчально-методичні рекомендації призначені для використання студентами денної та заочної форм навчання спеціальностей 051 – Економіка, 126 – Інформаційні системи та технології (бакалаврський рівень) під час виконання лабораторних робіт з дисципліни «Технологія проєктування інформаційних систем».

Містить навчально-методичні матеріали для вивчення основ структурного та об'єктно-орієнтованого проєктування. Ознайомлення з можливостями технологій проєктування інформаційних систем дозволяє студентам з перших занять одержати уявлення про технологію розробки структурних та об'єктно-орієнтованих додатків.

ЗМІСТ

ПЕРЕДМОВА

Мета навчальної дисципліни «Технологія проєктування інформаційних систем» полягає в формуванні теоретичних знань про сучасні технології проєктування програмних систем та практичних навичок використання сучасних технологій у проєктуванні інформаційних систем.

Навчальна дисципліна забезпечує набуття таких передбачених освітньою програмою компетентностей:

здатність розв'язувати складні спеціалізовані задачі та практичні проблеми в області інформаційних систем та технологій, або в процесі навчання, що характеризуються комплексністю та невизначеністю умов, які потребують застосування теорій та методів інформаційних технологій;

здатність застосовувати стандарти в області інформаційних систем та технологій при розробці функціональних профілів, побудові та інтеграції систем, продуктів, сервісів і елементів інфраструктури організації;

здатність проектувати, розробляти та використовувати засоби реалізації інформаційних систем, технологій та інфокомунікацій (методичні, інформаційні, алгоритмічні, технічні, програмні та інші) ;

здатність вибору, проєктування, розгортання, інтегрування, управління, адміністрування та супроводжування інформаційних систем, технологій та інфокомунікацій, сервісів та інфраструктури організації;

для ОПП «Комп'ютерні технології в бізнесі» та :

здатність до абстрактного мислення, аналізу та синтезу;

здатність застосовувати знання у практичних ситуаціях;

здатність бути критичним і самокритичним;

здатність приймати обґрунтовані рішення;

здатність діяти соціально відповідально та свідомо;

здатність застосовувати прикладне програмне забезпечення для розв'язання професійних завдань та інформаційно-аналітичної підтримки процесів управління;

для ОПП «Інформаційні технології та моделювання в економіці».

Відповідно до освітньої програми дисципліна спільно з іншими освітніми компонентами має забезпечити досягнення таких програмних результатів навчання:

застосовувати знання фундаментальних і природничих наук, системного аналізу та технологій моделювання, стандартних алгоритмів та дискретного аналізу при розв’язанні задач проєктування і використання інформаційних систем та технологій;

проводити системний аналіз об’єктів проєктування та обґрунтовувати вибір структури, алгоритмів та способів передачі інформації в інформаційних системах та технологіях;

застосовувати правила оформлення проєктних матеріалів інформаційних систем та технологій, знати склад та послідовність виконання проєктних робіт з урахуванням вимог відповідних нормативно-правових документів для запровадження у професійній діяльності;

для ОПП «Комп’ютерні технології в бізнесі» та :

знати основні підходи до проєктування, уміти виконувати аналіз предметної області, розробляти вимоги до проєктованого об’єкту, виконувати проєктування з використанням сучасних технологій;

для ОПП «Інформаційні технології та моделювання в економіці».

Навчальна дисципліна є обов'язковою для вивчення студентами, які здобувають освітній ступінь бакалавра за освітніми програмами «Комп’ютерні технології в бізнесі» та «Інформаційні технології та моделювання в економіці».

Відповідно до робочої програми навчальної дисципліни «Технологія проєктування інформаційних систем» передбачено виконання лабораторних робіт. Дане методичне видання містить основні теоретичні положення для засвоєння матеріалу, інструкції до виконання завдань, запитання до захисту лабораторних робіт.

ЛАБОРАТОРНА РОБОТА №1

ТЕМА РОБОТИ: Застосування каскадної моделі для розробки програмного забезпечення систем.

МЕТА РОБОТИ: Навчитися застосовувати каскадну модель для розробки програмного забезпечення систем на прикладі реалізації алгоритмів розв'язання нелінійних рівнянь методами дихотомії та хорд.

ТРИВАЛІСТЬ: 2 години.

МЕТОДИЧНІ ВКАЗІВКИ

1.1 Постановка задачі

Нехай розглядається рівняння $f(x)=0$. ***Коренем*** рівняння називається значення $\bar{x}$, при якому $f(\bar{x})=0$. Корінь $\bar{x}$ називається ***простим***, якщо $f'(\bar{x})\neq 0$, у противному випадку корінь називається кратним. Ціле число m називається кратністю кореня $\bar{x}$, якщо $f^{(k)}(\bar{x})=0$ для k=1,2,3,…, m-1 і $f^{(m)}(\bar{x})\neq 0.$

Задача наближеного рішення рівняння з точністю ε полягає в тому, щоб знайти таке значення x, що $\left|x-\bar{x}\right|<\varepsilon$.

Рішення задачі розбивається на два етапи: на першому етапі здійснюють *локалізацію* коренів, на другому етапі роблять *ітераційне уточнення* коренів. На етапі локалізації коренів знаходять досить вузькі відрізки (або відрізок, якщо корінь єдиний), які містять один і тільки один корінь рівняння $f(x)=0$. На другому етапі обчислюють наближене значення кореня із заданою точністю. Часто замість відрізка локалізації досить указати початкове наближення до кореня.

1.2 Локалізація (відділення) коренів

Локалізацію коренів найчастіше роблять графічно. Тобто, будують графік функції і виділяють відрізки, на яких функція змінює знак (пересікає ось OX).

Приклад 1.1. Побудова графіків і локалізація коренів рівняння.

В MsEXEL побудуємо графік функції y*(x)= x^3 - cos(x) + 1 на відрізку [-0.6,0.6]*.

Результат наведений на рис. 1.1.

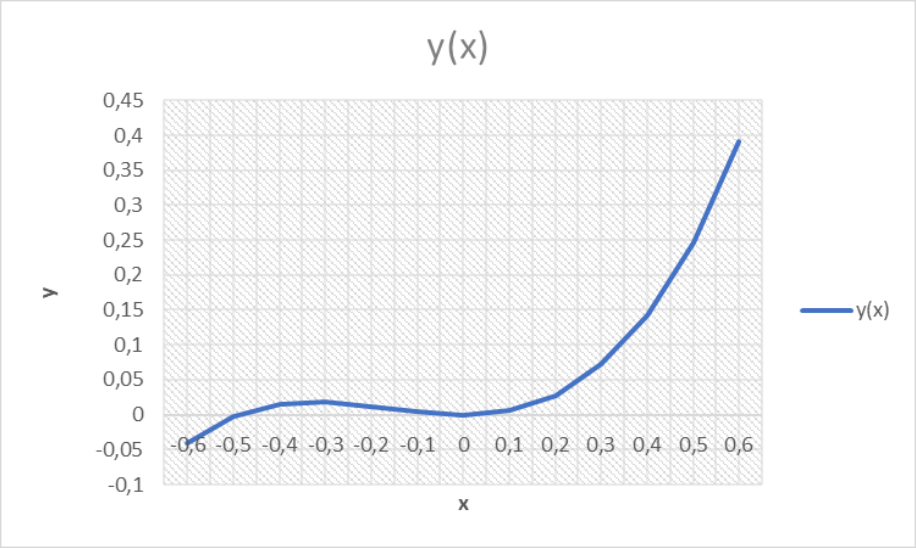


Рисунок 1.1 - Графік функції

З графіка видно, що один з коренів рівняння знаходиться на відрізку [-0.6,-0.4], а другий дорівнює нулю, що можна перевірити аналітично.

1.3 Методи ітераційного уточнення коренів

1.3.1 Метод дихотомії (поділу відрізку навпіл)

Нехай $[a,b]$ – відрізок локалізації. Припустимо, що функція *f(x)* безперервна на $[a,b]$ і на кінцях приймає значення різних знаків $f(a)\cdot f(b)<0$.

Алгоритм методу дихотомії полягає в побудові послідовності вкладених відрізків, на кінцях яких функція приймає значення різних знаків. Кожний наступний відрізок одержують діленням навпіл попереднього. Опишемо один крок ітерацій методу. Нехай на k-ому кроці знайдений відрізок $[a^{(k)},b^{(k)}]$ такий, що $f(a^{(k)})\cdot f(b^{(k)})<0$. Знайдемо середину відрізка $c^{(k)}=\dfrac{a^{(k)}+b^{(k)}}{2}$. Якщо $f(c^{(k)})=0$, то $c^{(k)}$ – корінь і задача вирішена. Якщо ні, то із двох половин відрізка вибираємо ту, на кінцях якої функція має протилежні знаки:

$a^{(k+1)}=a^{(k)}$, $b^{(k+1)}=c^{(k)}$, якщо $f(a^{(k)})\cdot f(c^{(k)})<0$,

$a^{(k+1)}=c^{(k)}$, $b^{(k+1)}=b^{(k)}$, якщо $f(a^{(k)})\cdot f(c^{(k)})>0$,

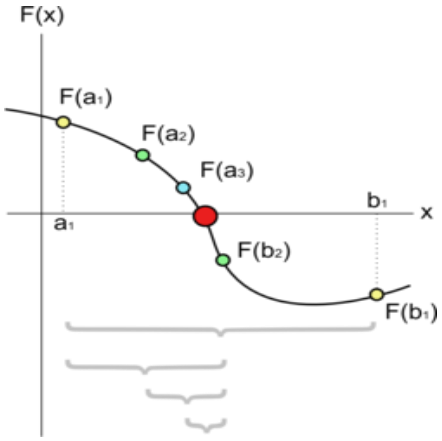


Рисунок 1.2 - Метод дихотомії

Критерій закінчення ітераційного процесу: якщо довжина відрізка локалізації менше 2ε, то ітерації припиняють і як значення кореня із заданою точністю приймають середину відрізка.

Теорема про збіжність методу дихотомії. Нехай функція *f(x)* безперервна на *[a,b]* і на кінцях приймає значення різних знаків $f(a)\cdot f(b)<0$. Тоді метод сходиться й справедлива оцінка похибки: $\left|x^{(k)}-\bar{x}\right|\le\left(b^{(k)}-a^{(k)}\right)/2=(b-a)/2^{k+1}$.

З останньої нерівності легко одержати апріорну оцінку кількості кроків, необхідних для досягнення заданої точності:

$\frac{b-a}{2^{k+1}}\le\varepsilon,$ звідки одержуємо $k\ge 1+\frac{\ln((b-a)/\varepsilon)}{\ln 2}$.

1.3.2 Метод хорд

У деяких випадках трохи більшу швидкість збіжності має метод хорд, у якого на другому етапі при виборі чергового наближення усередині відрізка, що містить корінь, враховується величина нев’язки на кінцях відрізка: точка вибирається ближче до того кінця, де нев'язка менше (але в деяких випадках це може сповільнити збіжність у порівнянні з методом дихотомії).

Геометричний зміст полягає в заміні кривої $y=f(x)$ хордою. Чергове наближення обчислюється як точка перетинання хорди з віссю абсцис.

Якщо $[a,b]$ – відрізок, що має корінь, то записуємо рівняння хорди і за нове наближення до кореня приймаємо точку перетинання хорди з віссю ОХ.

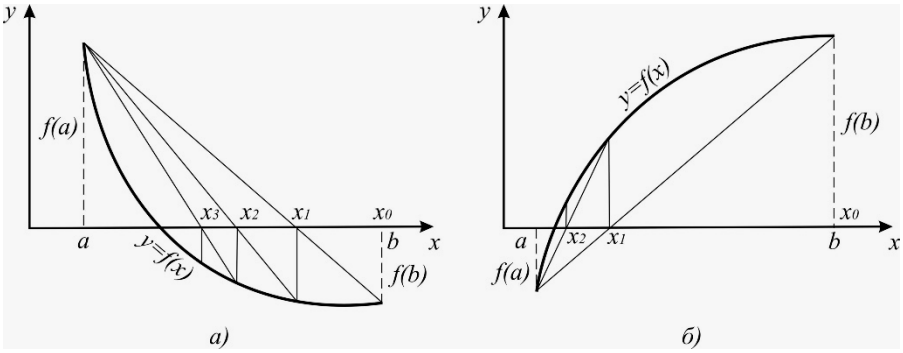


Рисунок 1.3 - Метод хорд

Рівняння хорди через кінцеві точки має вигляд: $\frac{x-a}{b-a}=\frac{y-f(a)}{f(b)-f(a)}$.

Для точки перетинання хорди з віссю абсцис $x=x_1,\quad y=0$ і тому маємо $x_1=a-\frac{b-a}{f(b)-f(a)}\cdot f(a),\quad x=x_1$ приймається за чергове наближення до кореня. Далі вибирається той із проміжків $[a,x_1],\quad [x_1,b]$, на кінцях якого функція має значення різних знаків, і т.д. При цьому, якщо $f(x)$ двічі безупинно диференційована функція і знак $f''(x)$ зберігається на розглянутому проміжку, то отримані наближення будуть сходитися до кореня монотонно. У цьому випадку у всіх одержуваних проміжків один кінець буде загальним, а саме той, на якому збігаються знаки функції й другої похідної. Цей кінець називають нерухомим, а протилежний кінець відрізка приймають за початкове наближення до кореня. Нехай c – той з кінців відрізка $[a,b]$ $f(c)\cdot f''(c)>0$, а x_0- початкове значення (протилежний кінець відрізка, для якого виконується умова), тоді ітерації проводимо за формулою:

$$x_{k+1}=x_k-\frac{c-x_k}{f(c)-f(x_k)}\cdot f(x_k),\ k=0,1,2,\ldots$$

Закінчують ітерації, коли різниця між двома наближеннями до кореня стає менш ніж необхідна точність обчислення кореня, тобто коли для якогось k $|x_{k+1}-x_k|\le\varepsilon$.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №1

Задача 1.1. Розробіть програму, що реалізує:

1 варіант (непарний номер в списку групи). Метод дихотомії.

2 варіант (парний номер в списку групи). Метод хорд.

Етапи розробки програми:

1) побудувати блок-схему алгоритму;
2) визначити типи змінних;
3) записати алгоритм мовою С;
4) розробити тести, протестувати програму.
5) супроводити текст програми коментарями.

Зверніть увагу на такі вимоги до програми:

- перед використанням алгоритму перевіряється правильність заданих початкових даних, що вкрай важливе в чисельних методах;

- програма повинна бути складена таким чином, щоб на кожній ітерації обчислювалось тільки одне значення функції.

Задача 1.2. За допомогою розробленої програми знайти корінь рівняння, обраного відповідно свого індивідуального завдання з точністю ε=0.3. Відрізок на якому потрібно знайти корінь рівняння визначити, побудувавши графік функції в EXEL.

Скільки потрібно зробити ітерацій для одержання точності ε=0.01.

Отриманий результат перевірте засобами EXEL, використовуючи пакет «пошук рішення».

Індивідуальні завдання до лабораторної роботи

№	Рівняння	№	Рівняння
1	$3x^4+4x^3-12x^2-5=0$	11	$2x^4-8x^3+8x^2-1=0$
2	$2x^3-9x^2-60x+1=0$	12	$2x^4+8x^3+8x^2-1=0$
3	$x^4-x-1=0$	13	$x^4-4x^3-8x^2+1=0$
4	$2x^4 - x^2-10=0$	14	$2x^4-9x^3-60x^2+1=0$
5	$3x^4+8x^3+6x^2-10=0$	15	$x^5 +x^2-5=0$
6	$x^4 -18x^2+5x-8=0$	16	$3x^4+4x^3-12x^2-7=0$
7	$x^4+4x^3-12x^2+1=0$	17	$3x^4+8x^3+6x^2-11=0$
8	$x^4 - x^3-2x^2+3x-3=0$	18	$x^4 -18x^3-10=0$
9	$3x^4+4x^3-12x^2+1=0$	19	$3x^4-8x^3-18x^2+2=0$
10	$3x^4-8x^3-18x^2+2=0$	20	$x^4 -18x -10=0$

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Сформулюйте постановку задачі наближеного рішення нелінійного рівняння й основні етапи її рішення.
2. Опишіть алгоритм методу дихотомії.
3. При яких умовах метод дихотомії дає збіжну до кореня рівняння послідовність?
4. Опишіть алгоритм методу хорд.
5. При яких умовах метод хорд дає збіжну до кореня рівняння монотонну послідовність?

ЛАБОРАТОРНА РОБОТА №2

ТЕМА РОБОТИ: Застосування ітераційної моделі для розробки програмного забезпечення систем.

МЕТА РОБОТИ: Розробити програмне забезпечення для розв'язання нелінійних рівнянь методами Ньютона та простих ітерацій на основі ітераційної моделі.

ТРИВАЛІСТЬ: 2 години.

МЕТОДИЧНІ ВКАЗІВКИ

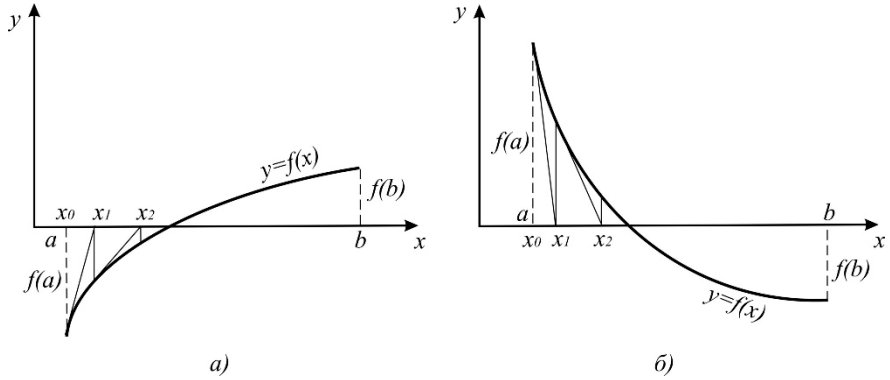
2.1 Метод Ньютона (метод дотичних)

Розрахункова формула методу Ньютона має вигляд: $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$.

Геометрично метод Ньютона означає, що наступне наближення до кореня $x^{(n+1)}$ є точка перетинання з віссю OX дотичної, проведеної до графіка функції $y=f(x)$ у точці $\left(x^{(n)}, f(x^{(n)})\right)$.

Теорема 1. Нехай на відрізку $[a,b]$ функція $f(x)$ має неперервні першу та другу похідні постійного знаку та нехай $f(a)\cdot f(b)<0$. Тоді, якщо точка x_0 обрана на $[a,b]$ так, що $f(x_0)\cdot f''(x_0)>0$, то послідовність, що починається з цієї

точки та визначається за методом Ньютона, збігається монотонно до кореня рівняння $\bar{x} \in [a,b], \quad f(\bar{x}) = 0$.



а) *б)*

Рисунок 2.1 - Метод Ньютона

Критерій закінчення ітераційного процесу. При заданій точності ε >0 обчислення варто вести доти, поки не буде виконана нерівність $\left|x^{(n)} - x^{(n-1)}\right| < \varepsilon$.

2.2 Метод простих ітерацій (послідовних повторень)

Для застосування методу простої ітерації треба рівняння $f(x) = 0$ *перетворити до виду* $x = \varphi(x)$. Це перетворення можна виконати різними способами. Функція $\varphi(x)$ називається ітераційною функцією. Розрахункова формула методу простої ітерації має вигляд: $x^{(n+1)} = \varphi(x^{(n)})$.

Критерій закінчення ітераційного процесу. При заданій точності ε >0 обчислення варто вести доти, поки не виконається нерівність $\left|x^{(n)} - x^{(n-1)}\right| < \frac{1-q}{q}\varepsilon$. Якщо величина $0 < q \le 0.5$, то можна використати більш простий критерій закінчення ітерацій: $\left|x^{(n)} - x^{(n-1)}\right| < \varepsilon$.

Ключовий момент у застосуванні методу простої ітерації складається в еквівалентному перетворенні рівняння. Спосіб, при якому виконана умова збіжності методу простої ітерації, полягає в наступному: вхідне рівняння приводиться до виду $x = x - \alpha \cdot f(x)$. Припустимо додатково, що похідна f' зберігає знак й $m \le f'(x) \le M$ на відрізку [a,b]. Тоді при виборі ітераційного параметра $\alpha = \frac{2}{M+m}$ метод збігається й значення $q = \left|\frac{M-m}{M+m}\right| < 1$.

2.3 Метод січних

Існує досить багато модифікацій методу Ньютона, ціль яких зменшити обчислювальні витрати, пов'язані з необхідністю обчислення похідної на кожному кроці ітерацій. Одна з найбільш вдалих модифікацій - метод січних - вимагає обчислення на кожному кроці тільки одного значення функції. Ітерації проводяться за формулою:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})(x^{(k-1)} - x^{(k)})}{f(x^{(k-1)}) - f(x^{(k)})}, \quad k = 1,2,3,...$$

$x^{(k+1)}$ є абсциса точки перетинання з віссю OX прямої, проведеної через точки $\left(x^{(k-1)}, f(x^{(k-1)})\right), \left(x^{(k)}, f(x^{(k)})\right)$, тобто січної.

Цей метод двокроковий, тому що на кожній ітерації використовує два попередні наближення до кореня, а для початку ітераційного процесу необхідно мати $x^{(0)}$ і $x^{(1)}$.

Метод має високий порядок збіжності, рівний ≈ 1.618. Закінчення обчислень за методом січних, з огляду на його швидку збіжність, можна контролювати за допомогою перевірок на малість модулів нев'язок $\left|f(x^{(k)})\right|$ або значень $\left|x^{(k-1)} - x^{(k)}\right|$. Однак, головне тут – це зуміти вчасно зупинити процес обчислень, не чекаючи моменту, коли похибки обчислень почнуть перевершувати похибки методу внаслідок обчислення різниці двох близьких значень $f(x^{(k-1)}), f(x^{(k)})$. У плані чисельної стійкості метод січних програє методу Ньютона.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №2

Завдання 2.1. Виконайте відділення коренів із використанням графічної оцінки

Завдання 2.2. Знайдіть один з коренів методами

Варіант 1. Методом Ньютона з декількома відносними погрішностями. Порівняйте об'єм обчислень для різних похибок ε.

Варіант 2. Методом простої ітерації з декількома відносними погрішностями. Порівняйте об'єм обчислень для різних похибок ε.

Варіант 3. Методом січних з декількома відносними погрішностями. Порівняйте об'єм обчислень при використанні зазначених методів і для різних похибок ε.

Етапи розробки програми:

1) побудувати блок-схему алгоритму;
2) визначити типи змінних;
3) записати алгоритм мовою С;
4) розробити тести, протестувати програму.
5) супроводити текст програми коментарями.

Зверніть увагу на такі вимоги до програми:

- перед використанням алгоритму перевіряється правильність заданих початкових даних, що вкрай важливе в чисельних методах;

- програма повинна бути складена таким чином, щоб на кожній ітерації обчислювалось тільки одне значення функції.

Завдання 2.3. Перевірте отримані результати засобами MSEXEL.

Індивідуальні завдання до лабораторної роботи

№	Рівняння	№	Рівняння
1	$\ln(x)+(x+1)^3=0$	11	$2\operatorname{arctg}(x)-x+3=0$
2	$x\cdot 2^x=1$	12	$(x-3)\cdot\cos(x)=1$
3	$x+\cos(x)=1$	13	$x^x= 20-9x$
4	$x+\lg(1+x)=1.5$	14	$x\cdot\lg(x)=1$
5	$\lg(2+x)+2x=3$	15	$tg^3x=x-1$
6	$2^x+5x-3=0$	16	$5^x =1+e^{-x}$
7	$5^x+3x =0$	17	$5^x =3-e^x$
8	$3e^x=5x+2$	18	$\operatorname{arctg}(x^2+1/x)=x$
9	$5^x=6x+3$	19	$tg(0.55x+0.1)=x^2$
10	$2e^x+5x-6=0$	20	$5^x-6x =7$

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Запишіть розрахункову формулу методу Ньютона й дайте геометричну інтерпретацію методу.
2. Виведіть критерій закінчення ітерацій для методу простої ітерації з оцінки похибки.
3. Опишіть алгоритм методу січних. Укажіть переваги й недоліки методу січних у порівнянні з методом Ньютона.

ЛАБОРАТОРНА РОБОТА №3

ТЕМА РОБОТИ: Застосування методології RAD для проєктування програмної системи.

МЕТА РОБОТИ: Навчитися застосовувати методологію RAD для проєктування програмної системи, призначеної для мінімізації функцій однієї змінної .

ТРИВАЛІСТЬ: 4 години.

МЕТОДИЧНІ ВКАЗІВКИ

3.1 Постановка задачі

Позначимо $R = \{u : -\infty < u < \infty\}$ – числова вісь, U - деяка множина з R, $J(u)$ – функція, визначена на множині U, яка має в кожній точці $u \in U$ кінечне значення.

Прикладами множини з R є:

відрізок $[a,b] = \{u \in R : a \le u \le b\}$,

інтервал $(a,b) = \{u \in R : a < u < b\}$,

де a, b – відомі задані числа.

В залежності від властивостей множини U і функції $J(u)$ множина U_* може містити одну, кілька або безкінечну кількість точок, а також можливі випадки, коли U_* є пустою.

Приклад. Нехай $J(u) = u^2$. На множині $U = \{u : -1 \le u \le 3\}$ мінімальне значення $J(u)$ дорівнює нулю, множина U_* складається з єдиної точки $u_* = 0$.

Приклад. Функція $J(u)=\ln(u)$, $U=\{u:0<u\le 1\}$. В цьому випадку $U_*=\varnothing$, так як в усіх точках з U функція приймає кінечні значення, а для послідовності $u_k=\frac{1}{k}$, $k=1,2,...$, маємо $\lim\limits_{k\to\infty} J(u_k)=-\infty$.

В подальшому будемо розглядати два типи задач. До першого типу будемо відносити задачі, в яких потрібно визначити величину $J_*=\inf\limits_{u\in U} J(u)$. Підкреслимо, що в цьому випадку неважливо, чи буде множина U_* точок мінімуму $J(u)$ на U непустою чи пустою. До другого типу задач відносимо ті задачі, у яких множина U_* є непустою і слід поряд з J_* знайти деяку точку $u_*\in U_*$.

Можлива і більш широка постановка задачі мінімізації другого типу – коли шукаються не тільки точки мінімуму в сенсі визначення 1, але і точки локального мінімуму.

Розглянемо функцію, графік якої зображено на рис. 3.1. Для неї точки u_0,u_2,u_4 є точками строю локального мінімуму, а в точках $u_5<u\le u_6$ та $u_8\le u\le u_9\,\mathrm{u}_8\le \mathrm{u}\le \mathrm{u}_9$ функція досягає нестрогого локального мінімуму.

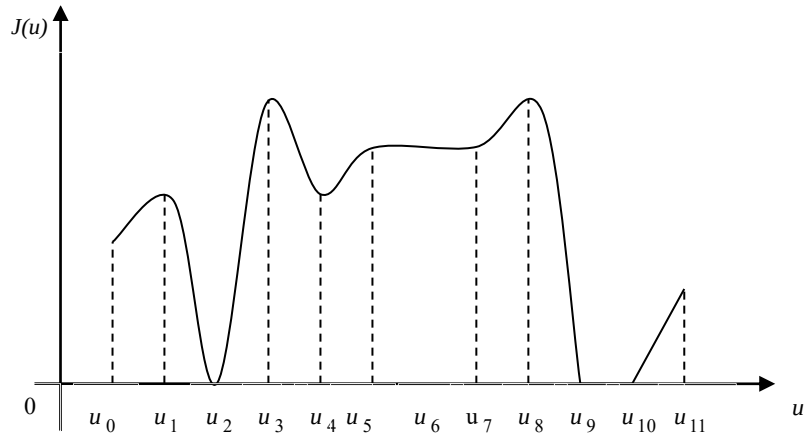


Рисунок 3.1−Графік функції з локальними мінімумами

Точки локального мінімуму, в яких мінімум досягається відповідно визначення 1, називають точками глобального або абсолютного мінімуму функції $J(u)$ на множині U.

Визначимо клас функції, для яких всі точки локального мінімуму є точками глобального мінімуму.

Визначення. Функція $J(u)$ називається унімодальною на відрізку $U=[a,b]$, якщо вона неперервна на $[a,b]$ та існують числа α,β $(a \le \alpha \le \beta \le b)$ такі, що:

1) $J(u)$ строго монотонно спадає при $a \le u \le \alpha$, якщо $a<\alpha$;

2) $J(u)$ строго монотонно зростає при $\beta \le u \le b$, якщо $\beta<b$;

3) $J(u)=J_*=\inf\limits_{u\in U} J(u)$ при $\alpha \le u \le \beta$, так що $U_*=[\alpha,\beta]$.

У випадку коли $\alpha = \beta$, функція $J(u)$ називається строго унімодальною на відрізку $[a,b]$.

Слід відзначити, що якщо функція $J(u)$ унімодальна на $[a,b]$, то вона буде унімодальною і на будь-якому відрізку $[c,d]\in[a,b]$.

3.2 Класичний метод

Класичний метод базується на диференціальному счисленні і докладно викладений в підручниках по математичному аналізу.

Нехай функція $J(u)$ кусково-неперервна та кусково-гладка на відрізку $[a,b]$. Це означає, що на $[a,b]$ може існувати лише кінечна кількість точок, в яких $J(u)$ або має розрив першого роду, або неперервна але не має похідної. В цьому випадку точками екстремуму функції $J(u)$ на $[a,b]$ можуть бути лише ті точки в яких виконується одна із наступних умов:

- $J(u)$ має розрив;
- $J(u)$ неперервна, але в цій точці не існує похідна $J'(u)$;
- похідна $J'(u)$ існує і дорівнює нулю;
- $u = a$ або $u = b$.

Такі точки називають точками, підозрілими на екстремум.

Пошук точок екстремуму функції починають зі знаходження всіх точок, підозрілих на екстремум. Потім проводять додаткові дослідження і відбирають із нас ті, які є точками локального мінімуму або максимуму. Для цього досліджують знак першої похідної $J'(u)$ в околі (або відповідному полуоколі граничних точок $u = a$, $u = b$) підозрілої точки. Для того щоб підозріла точка $u\in[a,b]$ була точкою локального мінімуму, достатньо, щоб $\lim\limits_{v\to u-0} J(v)\ge J(u)$, $\lim\limits_{v\to u+0} J(v)\ge J(u)$ та при деякому $\alpha>0$ на множинах

$[a,b]\bigcap(u,u+\alpha)=O_\alpha^+(u)$, $[a,b]\bigcap(u-\alpha,u)=O_\alpha^-(u)$ існувала похідна $J'(u)$, причому $J'(u)>0$ при $u\in O_\alpha^+(u)$ та $J'(u)<0$ при $u\in O_\alpha^-(u)$. Коли $\lim\limits_{v\to u-0}J(v)\le J(u)$, $\lim\limits_{v\to u+0}J(v)\le J(u)$ та $J'(u)<0$ при $u\in O_\alpha^+(u)$, $J'(u)>0$ при $u\in O_\alpha^-(u)$, то u - точка локального максимуму.

В тих випадках, коли можливо обчислити в підозрілій точці похідні другого та більш високих порядків, то їх також можна застосувати для дослідження поведінки функції в околі цієї точки. Нехай відомі похідні $J'(u),\dots,J^{(n)}(u)$, причому $J^{(i)}(u)=0, i=\overline{1,n-1}$, а $J^{(n)}(u)\neq 0$, $n\ge 1$. Якщо n – парне число, то у випадку $J^{(n)}(u)>0$ в точці u функція $J(u)$ має локальний мінімум. Якщо n – непарне, то при $a < u < b$ в точці u не може бути локального мінімуму або максимуму; при $u = a$ ($u = b$) у випадку $J^{(n)}(u)>0$ в точці u маємо локальний мінімум (максимум), а у випадку $J^{(n)}(u)<0$ – локальний максимум (мінімум).

Щоб знайти глобальний мінімум функції $J(u)$ на $[a,b]$ треба перебрати всі точки локального мінімуму і серед них вибрати точку з найменшим значенням функції, якщо таке існує.

Класичний метод дослідження функції на екстремум використовується в тих випадках, коли достатньо просто вдається з'ясувати всі підозрілі на екстремум точки та реалізувати описану вище схему відбору екстремальних точок. Нажаль класичний метод має дуже обмежене застосування. Справа в тому, що обчислення похідної $J'(u)$ в практичних задачах часто є непростою задачею. Наприклад, може статися, що значення функції $J(u)$ визначаються із спостережень або фізичних експериментів і одержати інформацію про її похідні вкрай важко. Але навіть в тих випадках, коли похідну все ж вдається обчислити, розв'язання рівняння $J'(u)=0$, та виявлення інших підозрілих точок, може бути пов'язане зі значними труднощами. Тому важливо мати також і інші методи пошуку екстремуму, які не потребують обчислення похідних, та більш зручні для реалізації на сучасних ЕОМ.

3.3 Метод половинного поділу відрізка.

Цей метод є найпростішим методом мінімізації функції однієї змінної, який не потребує обчислення похідної. Будемо припускати, що мінімізуєма функція $J(u)$ є унімодальною на відрізку $[a,b]$. Пошук мінімуму починається з вибору двох точок $u_1=(a+b-\delta)/2$ та $u_2=(a+b+\delta)/2$, де δ – константа, яка є параметром методу, $0<\delta<(b-a)$. Величина δ вибирається самостійно і може визначитися значимою кількістю вірних десяткових знаків при заданні аргументу u. Зокрема, зрозуміло, що δ не може бути менше машинної похибки ЕОМ, яка використовується для розв'язання задачі. Точки u_1, u_2 розміщені симетрично на відрізку $[a,b]$ відносимо його середини та при малих б ділять його майже пополам (це і пояснює назву методу).

Після вибору точок u_1, u_2 обчислюють значення $J(u_1)$, $J(u_2)$ і порівнюються між собою. Якщо $J(u_1)\le J(u_2)$, то $a_1=a$, $b_1=u_2$; якщо ж $J(u_1)>J(u_2)$, то $a_1=u_1$, $b_1=b$. Оскільки $J(u)$ унімодальна на $[a,b]$, то зрозуміло, що відрізок $[a_1,b_1]$ має спільну точку з множиною U_* точок мінімуму $J(u)$ на $[a,b]$ і його довжина дорівнює

$$b_1-a_1=(b-a-\delta)/2+\delta.$$

Нехай відрізок $[a_{k-1},b_{k-1}]$, який має непустий перетин з U_*, вже відомий, і

Тоді виберемо точки

$$u_{2k-1}=(a_{k-1}+b_{k-1}-\delta)/2,$$

$$u_{2k}=(a_{k-1}+b_{k-1}+\delta)/2,$$

які розміщені на відрізку $[a_{k-1},b_{k-1}]$ симетрично відносно його середини, і обчислюємо значення $J(u_{2k-1})$, $J(u_{2k})$. Якщо $J(u_{2k-1})\le J(u_{2k})$, то $a_k=a_{k-1}$, $b_k=u_{2k}$; якщо ж $J(u_{2k-1})>J(u_{2k})$, то $a_k=u_{2k-1}$, $b_k=b_{k-1}$. Довжина відрізка $[a_k,b_k]$, який одержали, дорівнює

$$b_k-a_k=(b-a-\delta)/2^k+\delta>\delta \text{ і } [a_k,b_k]\bigcap U_*\neq\varnothing .$$

Якщо кількість обчислень значень функції не обмежена, то процес поділу відрізка можна продовжувати доки не отримаємо відрізок $[a_k,b_k]$ довжини $b_k-a_k<\varepsilon$, де ε - задана точність, $\varepsilon>\delta$. Звідси маємо,

$$k>\log_2((b-a-\delta)/(\varepsilon-\delta)).$$

Оскільки, кожний поділ відрізка вимагає обчислення двох значень функції, то для досягнення точності $b_k - a_k < \varepsilon$ знадобиться всього $n = 2k > 2\log_2((b-a-\delta)/(\varepsilon-\delta))$ таких обчислень.

Після визначення відрізку $[a_k, b_k]$ в якості наближення до множини U_* можна взяти точку

$$\overline{u_n} = u_{2k-1} \text{ при } J(u_{2k-1}) \le J(u_{2k})$$

або

$$\overline{u_n} = u_{2k} \text{ при } J(u_{2k-1}) > J(u_{2k}),$$

а значення $J(\overline{u_n})$ може виступати в якості наближення для $J_* = \inf\limits_{u\in[a,b]} J(u)$.

При такому виборі наближення для U_* буде допущена похибка $\rho(\overline{u_n}, U_*) \le \max\left\{b_k - \overline{u_n};\ \overline{u_n} - a_k\right\} = (b-a-\delta)/2^k$.

Якщо не вимагати того, щоб значення функції, яке є наближенним до J_*, було обчислене неодмінно в тій же точці, яка служить наближенням до U_*, то замість $\overline{u_n}$ можна взяти точку $u_n = (a_{k-1} + b_{k-1})/2$ з меншою похибкою

$$\rho(\overline{u_n}, U_*) \le b_k - a_k = (b-a-\delta)/2^{k+1} + \delta/2,$$

тут k = n/2 і δ – достатньо мале.

Зрозуміло, що в цьому випадку можна провести ще одне додаткове обчислення значення функції в точці u_n і покласти $J(u_n) \approx J_*$.

Зі сказаного вище слідує, що методом половинного поділу за допомогою $n=2k$ обчислень значення функції можна визначити точку мінімуму унімодальної функції на відрізку $[a,b]$ в кращому випадку з точністю $\approx (b-a)/2^{(n/2+1)}$. Виникає питання про існування методів, які дозволяють за допомогою такої ж кількості обчислень значень функції розв'язати задачу мінімізації унімодальної функції точніше. Такі методи існують і один з них буде розглянуто в наступному пункті.

Відзначимо, що метод половинного поділу без змін можна застосовувати для мінімізації функції, які не є унімодальним. Але в цьому випадку не можливо гарантувати, що знайдений розв'язок буде достатньо точним наближенням до глобального мінімуму.

3.3 Метод золотого перетину

Як відомо, золотим перетином відрізку називається поділ відрізка на дві нерівні частини, щоб відношення довжини всього відрізку до довжини його більшої частини дорівнювало відношенню довжини більшої частини відрізка до довжини його меншої частини.

Неважко перевірити, що золотий перетин відрізку $[a,b]$ будується двома точками

$$u_1 = a + (3-\sqrt{5})(b-a)/2$$

$$u_2 = a + (\sqrt{5}-1)(b-a)/2,$$

які розміщені симетрично відносно середини відрізку, причому $a < u_1 < u_2 < b$.

Відзначимо також, що точка u_1 в свою чергу будує золотий перетин відрізку $[a,u_2]$, так як $\frac{u_2-a}{u_1-a} = \frac{u_1-a}{u_2-u_1}$.

Аналогічно точка u_2 будує золотий перетин відрізка $[u_1,b]$. Спираючись на цю властивість золотого перетину, опишемо відповідний метод.

Покладемо $a_1 = a$, $b_1 = b$. На відрізку $[a_1,b_1]$ візьмемо точки u_1, u_2, які будують золотий перетин, і обчислимо значення $J(u_1)$, $J(u_2)$. Далі, якщо $J(u_1) \le J(u_2)$, то покладемо $a_2 = a_1$, $b_2 = u_2$, $\overline{u_2} = u_1$; якщо ж $J(u_1) > J(u_2)$, то покладемо $a_2 = u_1$, $b_2 = b_1$, $\overline{u_2} = u_2$. Так як функція $J(u)$ унімодальна на $[a,b]$, то відрізок $[a_2,b_2]$ має хоча б одну спільну точку з множиною U_* точок мінімуму $J(u)$ на [a, b]. Крім того, $b_2 - a_2 = \frac{(\sqrt{5}-1)(b-a)}{2}$, а також $\overline{u_2}$ належить відрізку $[a_2,b_2]$. В цій точці $\overline{u_2}$ відоме значення

$$J(\overline{u_2}) = \min\{J(u_1), J(u_2)\}$$

і вона будує золотий перетин відрізка $[a_2,b_2]$.

Припустимо, що все відомі точки u_1, …, u_{n-1}, обчислені значення $J(u_1)$, …, $J(u_{n-1})$, знайдено відрізок $[a_{n-1},b_{n-1}]$ такий, що

$$[a_{n-1},b_{n-1}] \cap U_* \neq \varnothing ,$$

$$b_{n-1} - a_{n-1} = ((\sqrt{5}-1)/2)^{n-1}(b-a),$$

і відома точка $\overline{u_{n-1}}$, яка будує золотий перетин відрізка $[a_{n-1},b_{n-1}]$ і така що

$$J(\overline{u_{n-1}}) = \min_{1\le i\le n-1} J(u_i),\ n \ge 2.$$

Тоді в якості наступної точки візьмемо точку

$$u_n = b_{n-1} + a_{n-1} - \overline{u_{n-1}},$$

яка також будує золотий перетин відрізка $[a_{n-1}, b_{n-1}]$, обчислимо значення $J(u_n)$.

Нехай для визначенності $a_{n-1} < u_n < \overline{u_{n-1}} < b_{n-1}$ (випадок $\overline{u_{n-1}} < u_n$ $\bar{u}_{n-1} < u_n$ розглядається аналогічно). Якщо $J(u_n) \le J(\overline{u_{n-1}})$, то покладемо

$$a_n = a_{n-1},\ b_n = \overline{u_{n-1}},\ \overline{u_n} = u_n$$

якщо ж $J(u_n) > J(\overline{u_{n-1}})$, то покладемо

$$a_n = u_n,\ b_n = b_{n-1},\ \overline{u_n} = \overline{u_{n-1}}.$$

Новий відрізок $[a_n, b_n]$ такий, що

$$[a_n, b_n] \bigcap U_* \ne \varnothing\ ,$$

$$b_n - a_n = ((\sqrt{5}-1)/2)^n (b-a)$$

точка $\bar{u}_n$ будує золотий перетин відрізка $[a_n, b_n]$ і

$$J(\overline{u_n}) = \min\{J(u_n), J(\overline{u_{n-1}})\} = \min_{1\le i\le n} J(u_i).$$

Якщо кількість обчислень значень $J(u)$ наперед не обмежена, то цей процес можна продовжувати, наприклад, до тієї пори, поки не виконається нерівність $b_n - a_n < \varepsilon$, де ε - задана точність. Якщо ж кількість обчислень значень $J(u)$ наперед чітко задана і дорівнює n, по процес на цьому закінчується і в якості розв'язку задачі другого типу можна вважати пару $J(\overline{u_n}), \overline{u_n}$, де $J(\overline{u_n})$ є наближенням для $J_* = \inf_{u\in[a,b]} J(u)$, а також $\overline{u_n}$ служить наближенням для множини U_* з похибкою

$$\rho(\overline{u_n}, U_*) \le \max\left\{b_n - \overline{u_n};\ \overline{u_n} - a_n\right\} = \frac{1}{2}(\sqrt{5}-1)^n (b-a) = A_n\ .$$

Згадаємо, що за допомогою методу половинного поділу за $n=2k$ обчислень функції $J(u)$ в аналогічному випадку була одержана точка $\overline{u_n}$ з похибкою

$$\rho(\overline{u_n}, U_*) \le 2^{-n/2}(b-a-\delta) < 2^{-n/2}(b-a) = B_n.$$

Звідси маємо $\frac{A_n}{B_n} = \left(\frac{2\sqrt{2}}{\sqrt{5}+1}\right)^n$. Видно, що вже при невеликих n перевага методу золотого перетину перед методом половинного поділу є відчутною.

Для чисельної реалізації доречніше використовувати модифікацію методу золотого перетику, яка дозволяє уникнути швидкого зростання похибок при обчисленні точок $u_n (n \geq 2)$.

В цьому випадку, на кожному відрізку $[a_n, b_n]$, який містить точку $\bar{u}_n$ з попереднього кроку, уникати при обчисленні u_{n+1} використання формули $u_{n+1} = b_n + a_n - \bar{u}_n$ а в якості u_{n+1} вибрати ту з точок $a_n + (3-\sqrt{5})(b_n - a_n)/2$, $a_n + (\sqrt{5}-1)(b_n - a_n)/2$, яка більш віддалена від $\bar{u}_n$. Зрозуміло, що після такої модифікації метод золотого перетику, взагалі не буде мати властивості симетричності, і може бути не таким вже виточеним, але значно точнішим в реалізації.

Відзначимо також, що цей метод може бути застосований і для функцій, які є не унімодальними, але в цьому випадку одержаний розв'язок може досить відрізнятися від глобального.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №3

Завдання 3.1. Виберіть функцію з наведеної нижче таблиці, побудуйте її графік засобами MatLab'a (або Excel) в розумному діапазоні для пошуку точок її екстремумів (найбільшого й найменшого значень) із заданою точністю.

Завдання 3.2. Для обраної функції знайдіть оптимальне значення мінімуму, попередньо визначивши відрізок локалізації мінімуму.

Варіант 1. Методом половинного поділу.

Варіант 2. Методом золотого перетину.

Для кожного методу слід визначити: N_k - кількість виконаних ітерацій, N_f - кількість обчислень значень цільової функції, u_*-наближене значення точки оптимуму, $J(u_*)$ - наближене значення оптимуму цільової функції. Одержані результати подати у вигляді таблиці.

Завдання 3.3. Перевірте отримані результати засобами MSEXEL.

№	Назва способу визначення	$\varepsilon = 10^{-4}$				$\varepsilon = 10^{-8}$			
		N_k	N_f	u_*	$J(u_*)$	N_k	N_f	u_*	$J(u_*)$
1	Розроблена програма								
2	Засобами MSEXEL								
3	Пакет пошук рішення MSEXEL								

Індивідуальні завдання до лабораторної роботи

№	Функція $I(u)$	№	Функція $I(u)$
1	$\lvert u\rvert + \lvert u+1\rvert - 1$	11	$e^{-2u} + 0,5u^4$
2	$1 + u - 2.5u^2 + 0.25u^4$	12	$2\sqrt{1+u^2} + e^{-u}$
3	$u^2/3 + \ln(u) - 1$	13	$\dfrac{\cos 3u}{u}$
4	$4 + (u-2)^2$	14	$0.5\cos u - \sqrt{10u - u^2}$
5	$\lg u + \sin u$	15	$5u\ln(u+1) - 2u^3$
6	$2u\lg(u+2) + u^3 - 6$	16	$3u^2$-2(u+1)(ln(u+1)+2)
7	$\left\lvert \dfrac{u}{\ln u} - 2u^3 \right\rvert$	17	$u\sin 3u - \cos 2u$
8	$e^{2u} + \dfrac{3}{(u+2)}$	18	$3u^2 + 4u - \cos^2 u$
9	$\sin(0.2u-1) + (0.1u-5)^6$	19	$4(3-u)^{\frac{2}{3}} + 2x^3$
10	$u^2 - 3u + 4\cos u$	20	$u^2 - 3u + u\ln u$

ЛАБОРАТОРНА РОБОТА №4

ТЕМА РОБОТИ: Розробка програмного забезпечення на основі методології структурного аналізу.

МЕТА РОБОТИ: Навчися застосовувати методологію структурного аналізу для розробки програмного забезпечення на прикладі розв'язання задач багатовимірної оптимізації градієнтними методами.

ТРИВАЛІСТЬ: 4 години.

МЕТОДИЧНІ ВКАЗІВКИ

4.1 Методи мінімізації функції багатьох змінних

На теперішній час розроблена і досліджена велика кількість методів мінімізації функцій багатьох змінних. В посібнику будуть розглянуті лише найбільш відомі методи, та ті що застосовуються на практиці найчастіше. Надамо опис кожного методу, дослідимо сходимість.

4.1.1 Градієнтний метод

Будемо розглядати задачу

$$J(u) \to \inf; \quad u \in U \equiv E_n, \tag{4.1}$$

вважаючи, що функція $J(u)$ неперервно диференційована на E_n, тобто $J(u) \in C^1(E_n)$. Це означає, що

$$J(u+h) - J(u) = \langle J'(u), h \rangle + 0(h;u), \tag{4.2},$$

де $\lim\limits_{|h| \to 0} \dfrac{0(h;u)}{|h|} = 0.$

Справедлива нерівність Коші-Буняковського

$$-|J'(u)||h| \le \langle J'(u), h \rangle \le |J'(u)||h|,$$

причому, якщо $J'(u) \neq 0$, то права нерівність перетворюється в рівність тільки для $h = \alpha J'(u)$, а ліва нерівність – тільки при $J'(u) \neq 0,\ \alpha = const \ge 0.$ Звідси зрозуміло, що для $J'(u) \neq 0$ напрям найскорішого зростання функція $J(u)$ в

точці u співпадає з напрямом градієнта $J'(u)$, а напрям найскорішого спадання – з напрямом антиградієнта $(-J'(u))$.

Ця властивість градієнта лежить в основі багатьох ітераційних методів мінімізації функцій. Одним з таких методів є градієнтний метод. Цей метод, як і всі ітераційні методи, передбачає вибір початкового наближення – деякої точки u_0. Загальних правил вибору точки u_0 нажаль не існує. В тих випадках, коли з геометричних, фізичних чи будь-яких інших міркувань може бути одержана апріорна інформація про розміщення точки (точок) мінімума, то початкове наближення u_0 намагаються вибрати поближче до цієї області.

Будемо вважати, що деяка початкова u_0 вже вибрана. Тоді градієнтний метод заключається в побудові послідовності $\{u_k\}$ за правилом

$$u_{k+1} = u_k - \alpha_k \mathrm{J}'(u_k),\ \alpha_k > 0, \quad k = 0,1,\ldots \tag{4.3}$$

Число α_k називається довжиною кроку, кроком або крововим множником градієнтного методу.

Якщо $J'(u) \neq 0$, то крововий множник $\alpha_k > 0$ можна вибирати так, щоб $J(u_{k+1}) < J(u_k)$. Насправді з рівності (4.2) маємо

$$J(u_{k+1}) - J(u_k) = \alpha_k \left[-\left| \mathrm{J}'(u_k)^2 \right| + \frac{0(\alpha_k)}{\alpha_k} \right] < 0$$

для всіх достатньо малих $\alpha_k > 0$.

Якщо $J'(u_k) = 0$, то u_k - стаціонарна точка. В цьому випадку процес (4.3) завершується, і при необхідності проводяться додаткові дослідження поведінки функції в околі точки u_k.

В залежності від способу вибору α_k можна одержати різні варіанти градієнтного методу.

Перерахуємо кілька способів вибору α_k, які застосовуються найчастіше.

1. На промені $\{u \in E_n : u = u_k - \alpha J'(u_k),\ \ \alpha \geq 0\}$

Введемо функцію однієї змінної

$$f_k(\alpha) = J\left(u_k - \alpha J'(u_k)\right),\ \ \alpha \geq 0,$$

і визначимо α_k з умови

$$f_k(\alpha_k) = \inf_{\alpha \geq 0} f_k(\alpha) = f_{k_*},\ \ \alpha_k > 0. \tag{4.4}$$

Метод (3) - (4) називається методом найшвидшого спуску. Якщо $J'(u_k) \neq 0$, то

$$f_k'(0) = -\left|J'(u_k)\right|^2 < 0,$$

тому нижня грань може досягатися в (4) лише за умови $\alpha_k > 0$.

Наведемо приклад, коли значення α_k може бути визначена в явному вигляді.

Приклад . Нехай задана квадратична функція

$$J(u) = \frac{1}{2}\langle Au, u\rangle - \langle b, u\rangle, \tag{4.5}$$

де A – симетрична додатньо визначена матриця порядку $n \times n$,
b – вектор з E_n. Вище було показано, що функція сильно випукла і її похідні обчислюються за формулами

$$J'(u) = Au - b; \quad J''(u) = A.$$

Тому метод (4.3) в даному випадку виглядає так

$$u_{k+1} = u_k - \alpha_k(Au - b), k = 0,1,\ldots$$

Таким чином, градієнтний метод для функції (4.5) є добре відомим ітераційним методом розв'язання системи лінійних алгебраїчних рівнянь $Au=b$.

Визначимо α_k з умови (4.4). Користуючись формулою

$$J(u+h) - J(u) = \langle Au - b, h\rangle + \frac{1}{2}\langle Ah, h\rangle,$$

маємо

$$f_k(\alpha) = J(u_k) - \alpha\left|J'(u_k)\right|^2 + \left(\alpha^2/\alpha\right)\langle AJ'(u_k), J'(u_k)\rangle, \quad \alpha \geq 0.$$

Коли $J'(u_k) \neq 0$ умова

$$f'_k(\alpha) = -\left|J'(u_k)\right|^2 + \alpha\langle AJ'(u_k), J'(u_k)\rangle = 0$$

дає

$$\alpha_k = \frac{|J'(u_k)|^2}{< AJ'(u_k), J'(u_k) >} = \frac{|Au_k - b|^2}{< A(Au_k - b), Au_k - b >} > 0. \tag{4.6}$$

Оскільки $f_k(\alpha)$ випукла, то в знайденій точці α_k ця функція досягає своєї нижньої грані при $\alpha > 0$.

Але точне обчислення величини α_k з умови (4.4) не завжди можливе. Тому практично обмежуються знаходженням величини α_k, яка наближено задовольняє умовам (4.4).

2. На практиці нерідко задовольняються знаходженням деякого $\alpha_k > 0$, яке забезпечує виконання умови монотонності: $J(u_{k+1}) < J(u_k)$. Цією метою вибирають деяке значення $\alpha > 0$ і в методі (4.3) на кожній ітерації беруть $\alpha_k = \alpha$. При цьому для кожного $k \geq 0$ перевіряють умову монотонності, і в випадку, коли вона порушується, $\alpha_k = \alpha$ ділять до тих пір, доки монотонність не поновиться. Час від часу корисно пробувати збільшити α зі збереженням монотонності.

3. Якщо функція $J(u) \in C^1(E_n)$ і градієнт $J'(u)$ задовольняє умові Ліпшиця

$$|J'(u) - J'(\upsilon)| \leq L|u - \upsilon|, \quad u, \upsilon \in E_n, \tag{4.7}$$

причому константа L відома, то в (4.3) в якості α_k може бути взяте будь-яке число, яке задовольняє умовам

$$0 < \varepsilon_0 \leq \alpha_k \leq 2/(\alpha + 2\varepsilon). \tag{4.8}$$

де $\varepsilon_0, \varepsilon$ - додатні числа, які називаються параметрами методу. Зокрема, для $\varepsilon = L/2,\ \varepsilon_0 = 1/L$, маємо метод (4.3) з сталим кроком $\alpha_k = 1/L$. Звідси ясно, що якщо константа Ліпшиця L велика, або одержана за допомогою досить грубих оцінок, то кроковий множник α_k в (4.3) буде малим.

4. Можливий вибір α_k з умови

$$J(u_k) - J(u_k - \alpha_k J'(u_k)) \geq \varepsilon \alpha_k |J'(u_k)|^2, \quad \alpha > 0. \tag{4.9}$$

Для виконання умови (4.9) спочатку беруть деяке число $\alpha_k = \alpha > 0$ (одне і теж саме на всіх інтераціях; наприклад, $\alpha_k = 1$), а потім при необхідності дрібнять його, тобто змінюють за законом $\alpha_k = \lambda^i \alpha \quad (i = 0,1, 0 < \lambda < 1)$ до тих пір, поки вперше не виконається умова (4.9).

5. Можливе також і апріорне задання величин α_k з умови

$$\alpha_k > 0, k = 0,1,\ldots; \sum_{k=0}^{\infty} \alpha_k = \infty, \sum_{i=0}^{\infty} \alpha_k^2 < \infty. \tag{4.10}$$

Наприклад, в якості α_k можна взяти $\alpha_k = c(k+1)^{-\alpha}$, де $c = const > 0$, а число α таке, що $1/2 < \alpha \leq 1$. Зокрема, якщо $\alpha = 1,\ c = 1$, то одержимо $\alpha_k = (k+1)^{-1}, k = 0,1,\ldots$. Такий вибір $\{\alpha_k\}$ в (4.3) дуже простий для реалізації, але не гарантує виконання умови монотонності $J(u_{k+1}) < J(u_k)$, і взагалі, сходиться дуже повільно.

Коли деякий спосіб вибору α_k уже вибраний, то на практиці ітерації (4.3) продовжують до тих пір, поки не виконається деякий критерій закінчення обчислень.

Часто застосовуються наступні критерії:

$$|u_k - u_{k+1}| \le \varepsilon,$$

$$|J(u_k) - J(u_{k-1})| \le \delta,$$

$$|J'(u_k)| \le \gamma,$$

де $\varepsilon, \delta, \gamma$ - задані, достатньо малі додатні числа. Інколи наперед задають кількість ітерацій, можливі різноманітні комбінації цих та інших критеріїв. Нажаль, надійних критеріїв завершення обчислень, які б гарантували одержання розв'язання задачі (4.1) з заданою точністю, доки не існує, оскільки ці критерії можуть виконатися і далеко від точки мінімума.

Відзначимо, що коли досліджується сходимість метода, вважається, що процес (4.3) продовжується необмежено і приводить до побудови послідовності $\{u_k\}$. Виникає питання, чи буде послідовність $\{u_k\}$ мінімізуючою для задачі (4.1), і чи буде вона сходитися до множини точок мінімуму:

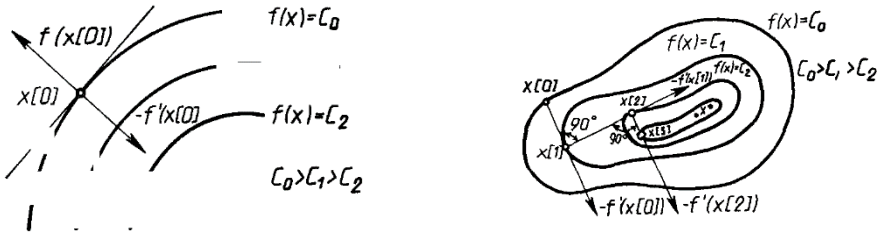
$$u_* = \left\{ u \in E_n : J(u) = J_* = \min_{E_n} J(u) \right\},$$

іншими словами, чи виконується співвідношення

$$\lim_{k\to\infty} J(u_k) = J_*, \quad \lim_{k\to\infty} \rho(u_k, U_*) = 0, \tag{4.11}$$

Для методу найшвидшого спуска відповідь на ці питання позитивна, якщо функція $J(u)$ – сильно випукла та $J(u) \in C^1(E_n)$.

Метод найшвидшого спуска має простий геометричний сенс: виявляється, точка u_{k+1}, яка визначається умовами (4.3), (4.4), лежить на промені $L_k = \{u : u = u_k - \alpha J'(u_k), \quad \alpha \ge 0\}$ в точці його дотику лінії рівня (при $n \ge 3$ – поверхні рівня) $\Gamma_{k+1} = \{u \in E_n : J(u) = J(u_{k+1})\}$, а сам промінь Γ_k перпендикулярний до лінії рівня $\Gamma_k = \{u \in E_n : J(u) = J(u_k)\}$ (рис. 4.1, 4.2).



а б

Рисунок 4.1− Лінії рівня функції двох змінних

З рис. 4.1 і 4.2 можна зрозуміти, що чим ближче лінії рівні $J(u) = const$ до кола, тим краще сходиться метод найшвидшого спуску.

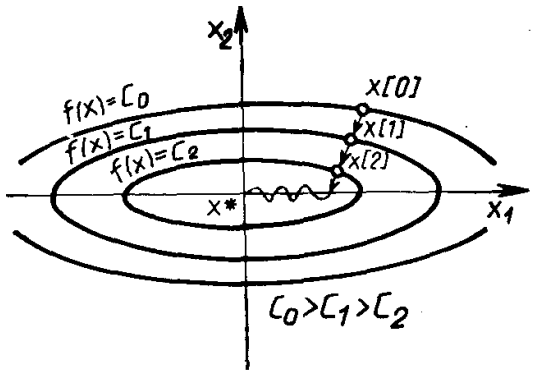


Рисунок 4.2−Геометрична інтерпретація методу найшвидшого спуску

Ці ж малюнки ілюструють, а теоретичні дослідження і чисельні експерименти підтверджують , що метод найшвидшого спуску і інші варіанти градієнтного методу повільно сходяться в тих випадках, коли поверхні рівня функції $J(u)$ дуже витягнуті і функція має „яружний" характер. Це означає, що невелика зміна деяких змінних призводить до різкої зміни значення функції – ця група змінних характеризує „схил яру", а по іншим змінним, які задають „дно яру", функція змінюється незначно (на рис. 4.2 зображені лінії рівня „яружної" функції двох змінних). Якщо точка лежить на „схилі яру", то напрям спуску з цієї точки буде майже перпендикулярним до напряму „дна яру", і в результаті наближення $\{u_k\}$ будуть по черзі знаходитися то на одному то на іншому „схилі яру". Якщо „схили яру" достатньо круті, то такі скачки „зі схилу на схил" точок u_k можуть дуже сповільнити сходимість градієнтного методу.

4.1.2 Метод Ньютона

Всі методи, розглянуті вище, відносяться до методів першого порядку, оскільки вони використовують лише перші похідні мінімізуємої функції. В усіх цих методах для визначення напрямку зменшення функції використовується лише лінійна частина розкладення функції в ряд Тейлора. Якщо мінімізуєма функція двічі неперервно диференційована і похідні $J'(u), J''(u)$ обчислюються достатньо просто, то можна застосовувати методи мінімізації другого порядку, які використовують квадратичну частину розкладення цієї функції в ряд Тейлора.

Розглянемо один з таких методів – метод Ньютона, який сходиться з квадратичною швидкістю.

Розглянемо метод Ньютона для задачі

$$J(u) \to \min; \quad u \in U \equiv E_n \tag{4.12}$$

де $J(u) \in C^2(E_n)$. Нехай $u_0 \in E_n$ - деяке початкове наближення. Якщо відоме наближення u_k, то прирощення функції $J(u) \in C^2(E_n)$ в точці u_k можна представити в вигляді

$$J(u) - J(u_k) = \langle J'(u_k), u - u_k \rangle + \frac{1}{2}\langle J''(u_k)(u - u_k), u - u_k \rangle + 0\left(|u - u_k|^2\right)$$

Розглянемо квадратичну частину цього прирощення

$$J_k(u) \equiv \langle J'(u_k), u - u_k \rangle + \frac{1}{2}\langle J''(u_k)(u - u_k), u - u_k \rangle \tag{4.13}$$

і відзначимо допоміжне наближення $\bar{u}_k$ з умови

$$J_k(\bar{u}_k) = \inf_{E_n} J_k(u). \tag{4.14}$$

Наступне $(k+1)-$е наближення будемо шукати у вигляді

$$u_{k+1} = u_k + \alpha_k(\bar{u}_k - u_k), \quad 0 \le \alpha_k \le 1. \tag{4.15}$$

В залежності від способу вибору величини α_k в (4.15) можна одержати різні варіанти методу (4.13) – (4.14). Найбільш часто використовують такі:

1) покладемо

$$\alpha_k = 1, k = 0,1,... \tag{4.16}$$

в цьому випадку $u_{k+1} = \bar{u}_k \ (k = 0,1,...,)$, тобто умова (4.14) одразу визначає (k+1)-е наближення. Іншими словами,

$$J_k(u_{k+1}) = \inf_{E_n} J_k(u), \quad k = 0,1,..., \tag{4.17}$$

Відомо, що то в точці мінімуму функції $J_k(u)$ її похідна дорівнює 0, тобто

$$J_k'(u_{k+1}) = J_k'(u_k) + J''(u_k)(u_{k+1} - u_k) = 0. \qquad (4.18)$$

Це означає, що на кожній ітерації методу (4.13) – (4.16) або (4.17) треба розв'язувати систему лінійних алгебраїчних рівнянь (4.18) відносно невідомої різниці $u_{k+1} - u_k$. Якщо матриця цієї системи $J''(u_k)$ - не вироджена, то з (4.18) маємо

$$u_{k+1} = u_k - \left(J''(u_k)\right)^{-1} J'(u_k), \quad k = 0,1,... \qquad (4.19)$$

2) в якості α_k в (4.15) можна прийняти $\alpha_k = \lambda^{i_0}$,

де i_0-мінімальний серед $i \geq 0$ номер, для них виконується нерівність

$$J(u_k) - J\left(u_k + \lambda^i(\bar{u}_k - u_k)\right) \geq \varepsilon\lambda^i \left|J_k(\bar{u}_k)\right|, \qquad (4.20)$$

де λ, ε - параметри метода, $\lambda > 0,\ \varepsilon < 1$.

3) можливий вибір α_k в (4.15) з умов

$$\begin{aligned} &0 \leq \alpha_k \leq 1, \quad f_k(\alpha_k) = \min_{0 \leq \alpha \leq 1} f_k(\alpha), \\ &f_k(\alpha) = J\left(u_k + \alpha(\bar{u}_k - u_k)\right). \end{aligned} \qquad (4.21)$$

Відзначимо, що метод Ньютона з вибором довжини кроку α_k за правилами (4.20), (4.21) являється аналогічним відповідним варіантам методу умовного градієнта.

Метод Ньютона для розв'язання задачі (4.1) звичайно застосовується в тих випадках, коли обчислення похідних $J'(u), J''(u)$ не викликає особливих труднощів і допоміжна задача (4.3) розв'язується достатньо просто. Достоїнством методу Ньютона являється висока швидкість сходимості. Тому хоч трудоємність кожної ітерації цього методу вище ніж в методах першого порядку, але загальний об'єм обчислювальної роботи, необхідної для розв'язання задачі (4.1) з заданою точністю, з застосуванням методу Ньютона може виявитися меншим ніж коли будуть використовуватися більш прості методи.

Сходимість методу Ньютона встановлюється наступною теоремою.

Теорема 1. Нехай функція $J(u)$ сильно випукла на E_n, $J(u) \in C^2(E_n)$ і, крім того,

$$\begin{aligned} &\left\|J''(u) - J''(v)\right\| \leq \alpha|u - v|, \quad u, v \in E_n, \\ &\alpha = const > 0. \end{aligned} \qquad (4.22)$$

нехай початкове наближення u_0 вибране таким, що

$$\alpha\left|J'(u_0)\right| \le 2\mu^2 q, \qquad (4.23)$$

де μ>0 – стала, така, що

$$\left\langle J'(u)u,u\right\rangle \ge \mu\|u\|^2, \forall u \in E_n,$$

q – деяка стала 0 < q <1. Тоді послідовність $\{u_k\}$, яка визначається умовами (4.8), існує, сходиться до точки u_* мінімуму $J(u)$ на E_n, причому, справедлива оцінка

$$\left|u_R - u_*\right| \le 2\mu L^{-1} q^{2^k}, k = 0,1,..., \qquad (4.24)$$

(без доведення).

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №4

Завдання 4.1. Виберіть функцію з наведеного нижче списку, побудуйте її графік засобами MatLab'a (або Excel) в розумному діапазоні для пошуку точок її екстремумів (найбільшого й найменшого значень) із заданою точністю.

Завдання 3.2. Для обраної функції знайдіть оптимальне значення мінімуму, попередньо визначивши множину локалізації мінімуму.

- Варіант 1. Градієнтним методом.
- Варіант 1. Методом Ньютона.

Спосіб вибору крокового множника обрати самостійно.

Завдання 3.3. Перевірте отримані результати засобами MSEXEL.

Розв′язати задачі безумовної оптимізації:

1. $J(u) = x^2 + y^2 + xy \to \min, \quad u \in E_2$
2. $J(u) = x^3 + y^3 - 5xy \to \min, \quad u \in E_2$
3. $J(u) = x^3 + y^3 - x^2 - 2xy - y^2 \to \min, \quad u \in E_2$
4. $J(u) = xy + \dfrac{1}{2(x^2 + y^2)} \to \min, \quad u \in E_2$
5. $J(u) = \sin(x+y) - \sin(x) - \sin(y) \to \min, \quad u \in E_2$
6. $J(u) = e^{\frac{x}{4}}(3x + 2y) \to \min(\max), \quad u \in E_2$

7. $J(u) = 10\ln(x) + xy^2 - y^3 \to \min(\max), \quad u \in E_2$

8. $J(u) = e^{-(x-xy+y)}(x+y) \to \min(\max), \quad u \in E_2$

9. $J(u) = 2x^2 + y^2 - xy - xz + 2z \to \min, \quad u \in E_3$

10. $J(u) = xy^2z^2(1-x-2y-3z) \to \min(\max), \quad u \in E_3$

ЛАБОРАТОРНА РОБОТА № 5

ТЕМА РОБОТИ: Розробка програмного забезпечення на основі функціональної моделі.

МЕТА РОБОТИ: Навчитися розробляти програмне забезпечення на основі функціональної моделі з застосуванням операцій введення вихідних даних та запис результатів в текстовий файл.

ТРИВАЛІСТЬ: 2 години.

МЕТОДИЧНІ ВКАЗІВКИ

5.1 Постановка задачі

Для програм однієї з лабораторних робіт організуйте введення вихідних даних з файлу та збереження результатів виконання в інший файл.

При цьому імена файлів вводять користувачем.

Програма повинна обчислити і записати результати обчислень для всіх варіантів вихідних даних, що містить файл вихідних даних.

5.2 Приклад реалізації

Припустимо необхідно реалізувати введення з файлу для розв’язання рівняння методом дихотомії.

Файл з вихідними даними “input_file.txt” може мати наступну структуру :

Початок_відрізку_ -0.6

Кінець_відрізку_ 0.6

Точність_ 0.001

Початок_відрізку_ -0.4
Кінець_відрізку_ 0.5
Точність_ 0.0001

Для зручності цей файл зручно розмістити в теку, в якій знаходиться файл з текстом програми, так як у цьому випадку не потрібно буде користувачу вводити весь шлях до файлу.

В даному випадку програма має забезпечити розв’язання рівняння за двома вихідних даних та збереження результатів.

Для забезпечення роботи з файловими потоками необхідно підключити заголовочний файл:

```
#include <fstream>
```

Та, для спрощення застосування потокових операцій, використати простір імен std

```
using namespace std;
```

Потім потрібно створити один потік для введення, один потік для виведення:

```
ifstream in;            // потік для введення з файлу
ofstream out;                  //  потік для виведення в файл
```

Запропонувати користувачу та ввести імена файлів. Наприклад

```
cout << "Введіть ім’я файлу з вихідними даними_";
cin >> input;
```

Змінна input повинна бути оголошена як символьний рядок:

```
char input[255];     // ім’я файлу з вихідними даними
```

Аналогічно вводимо і ім’я файлу для збереження результатів.

Після введення імен файлів відкриваємо потоки та пересвідчуємося, що потрібні нам файли успішно відкриті. Ця операція може виглядати так:

```
in.open(input);
if (!in.is_open())
        {
        cout << "Файл " << input<< " відкрити неможливо!\n"<<endl;
```

```
            cout << "Перевірте його існування і запустіть програму ще раз\n"
<< endl;
            cin >> yn;
            return 0;
        }
```

Процедури введення та обробки даних необхідно проводити в циклі до тих пір доки не вичерпаються дані в файлі введення:

```
while (!in.eof())
        {
        // тут розміщується програмний код введення, обробки даних та
        // виведення результатів
        …
        }
```

Для введення даних використовуємо потокову операцію >>

```
in >> s >> a;
in >> s >> b;
in >> s >> eps;
```

Зверніть увагу, що при введені даних з файлу нам необхідно пропускати символьну інформацію, що знаходиться в рядках файлу з вихідними даними. Для цього нам потрібен символьний рядок s.

Далі слідує обробка даних та виведення результатів.

Наприклад:

```
out << "Корінь рівняння  x*=" << c << endl;
out << "x^3-cos(x)+1=" << yc << endl;
out << "Кількість ітерацій к=" << k << endl << endl;
```

Аналогічно виводяться і дані на кожній ітерації.

Перед завершенням програми потрібно закрити задіяні файли:

```
in.close();
out.close();
```

ЛАБОРАТОРНА РОБОТА № 6

ТЕМА РОБОТИ: Аналіз предметної області та створення діаграми прецедентів.

МЕТА РОБОТИ: Отримати практичні навички побудови діаграм прецедентів.

ТРИВАЛІСТЬ: 8 години.

МЕТОДИЧНІ ВКАЗІВКИ

Технологія розробки діаграми прецедентів наступна. Для предметної області "*Підприємство по складанню й продажу комп'ютерів*" розробимо проект системи. У всіх лабораторних роботах ми будемо створювати діаграми для вказаної моделі.

1. **Створення головної діаграми прецедентів.**

Для нашої предметної області ми виділили наступних акторів:

Актор	Короткий опис
Менеджер по роботі із клієнтами	Співробітник, який спілкується із замовником і працює із замовленням
Менеджер по постачанню	Співробітник, який займається закупівлею необхідних комплектуючих
Інженер по складанню настільних комп'ютерів	Співробітник, який займається складанням настільних комп'ютерів
Інженер по складанню ноутбуків	Співробітник, який займається складанням ноутбуків
Інженер по тестуванню	Співробітник, який займається тестуванням зібраних комп'ютерів
Завскладом	Співробітник, який завідує складом комплектуючих

Розглянемо тепер, які можливості повинна надавати наша система:

- актор *Менеджер по роботі із клієнтами* використовує систему для оформлення, редагування замовлень і управління інформацією про клієнтів підприємства;

- актор *Менеджер по постачанню* використовує систему для перегляду переліку необхідних для закупівлі комплектуючих і ведення інформації про постачання;
- актор *Інженер по складанню настільних комп'ютерів* використовує систему для перегляду нарядів на складання персональних комп'ютерів, для замовлення комплектуючих зі складу й відмітки про хід виконання роботи;
- актор *Інженер по складанню ноутбуків* використовує систему для перегляду нарядів на складання ноутбуків, для замовлення комплектуючих зі складу й відмітки про хід виконання роботи;
- актор *Інженер по тестуванню* використовує систему для перегляду нарядів на тестування зібраної продукції й відмітки про хід виконання роботи;
- актор *Завскладом* використовує систему для обліку надходження й видачі комплектуючих.

На підставі вищевикладеного можна виділити наступні прецеденти:

Прецедент	**Короткий опис**
Робота із замовленням	Запускається менеджером. по роботі із клієнтами. Дозволяє вносити, змінювати, видаляти або переглядати замовлення.
Управління інформацією про клієнта	Запускається менеджером по роботі із клієнтами. Дозволяє додавати, змінювати або видаляти клієнтів, а також переглядати інформацію про клієнтів.
Управління інформацією про постачальників	Запускається менеджером по постачанню. Дозволяє додавати, змінювати або видаляти постачальників, а також переглядати інформацію про постачальників.
Управління інформацією про комплектуючі	Запускається менеджером по постачанню. Дозволяє переглядати інформацію про комплектуючі, робити аналіз їх витрати, прогнозувати необхідну кількість і робити замовлення.

Прецедент	Короткий опис
Вимога необхідних комплектуючих	Запускається інженером по складанню. Призначене для зажадання необхідних комплектуючі зі складу.
Тестування комп'ютерів	Запускається інженером по тестуванню. Дозволяє переглянути список комп'ютерів, що підлягають тестуванню й зробити відмітки про хід виконання робіт.
Облік постачання й видачі комплектуючих	Запускається завскладом. Дозволяє вести облік постачання й видачі запчастин і комплектуючих.

Створена головна діаграма прецедентів показана на рис. 6.1.

Розглянемо тепер відносини між акторами й прецедентами. У мові UML можливий тільки один тип відносин між актором і прецедентом - відношення комунікації. Тому всіх актором ми зв'язали із прецедентами відношенням *Unidirectional Association*. Оскільки інший тип відносин тут ми задати не може, то стереотип *communicate* можна не вказувати (він неявно мається на увазі). Для прецеденту *Складання комп'ютерів* не має значення який саме актор буде з ним взаємодіяти - *Інженер по складанню настільних комп'ютерів* або *Інженер по складанню ноутбуків.* Тому ми ввели ще одного актора - *Інженер по складанню*, з яким зв'язали перших двох акторів відношенням узагальнення (*Generalization*).

Відношення між прецедентами *Робота із замовленням* і *Управління інформацією про клієнта* - відношення розширення, оскільки коли актор *Менеджер по роботі із клієнтами* працює із замовленням (оформляє, змінює і т.д.), те не завжди при цьому він управляє інформацією про клієнтів.

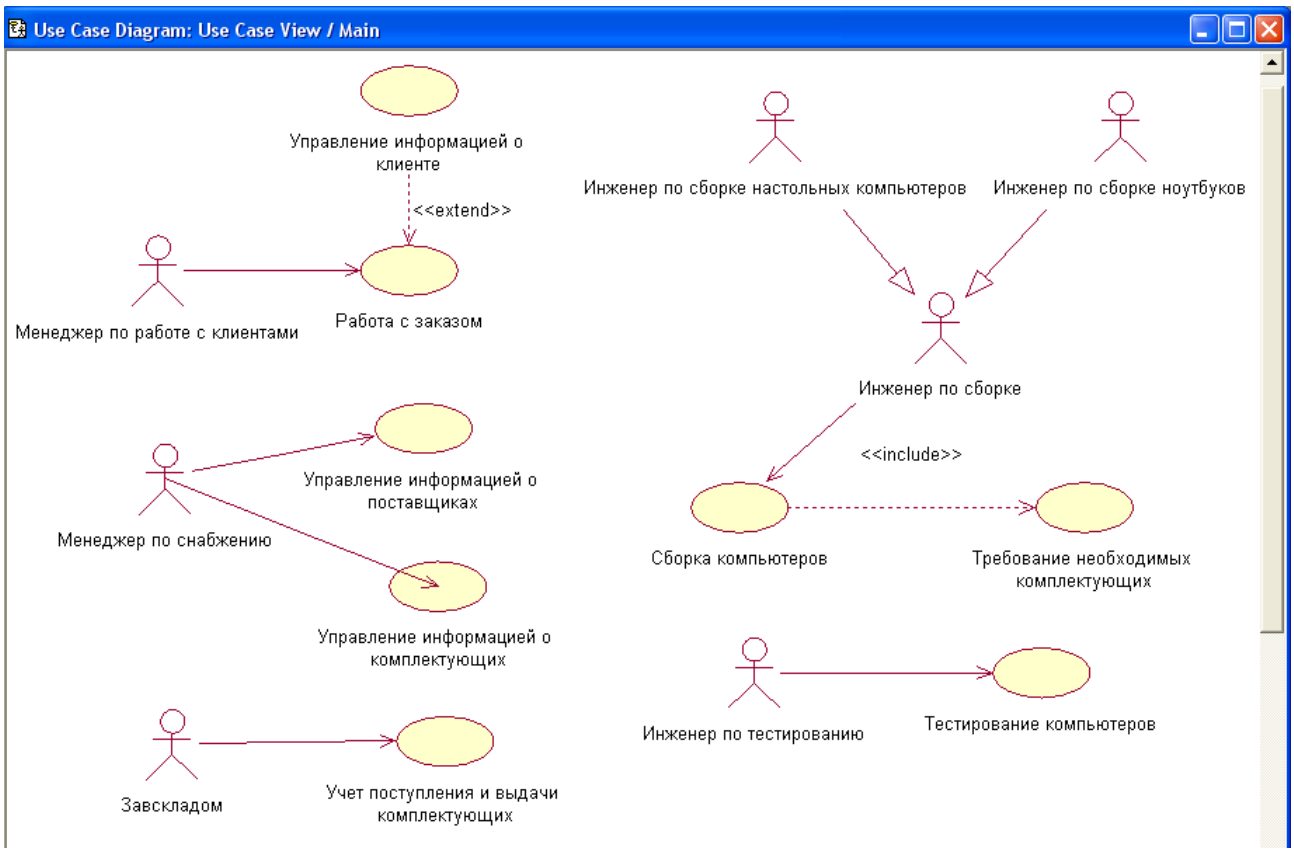


Рисунок 6.1 - Головна діаграма прецедентів

Відношення між прецедентами *Складання комп'ютерів* і *Вимога необхідних комплектуючих* - відношення включення, оскільки для складання комп'ютерів обов'язково потрібно замовляти необхідні комплектуючі зі складу.

2. Потік подій для прецедентів головної діаграми прецедентів

Потоки подій для прецедентів будемо описувати за наступним шаблоном:

- X.1 передумови;
- X.2 головний потік;
- X.3 під-потоки;
- X.4 альтернативні потоки;
- X.5 післяумови.

де X - число від одиниці до кількості прецедентів.

Потік подій для прецеденту «Робота із замовленням».

1.1. Передумови. Якщо замовлення оформляється для нового клієнта, то під-потік *додати нового клієнта (Add a New Client)* прецеденту *Управління інформацією про клієнта* повинен бути виконаний перед його початком.

1.2 Головний потік. Прецедент починає виконуватися, коли менеджер підключається до системи й уводить своє ім'я й пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій: *додати (Add)*, *змінити (Change), вилучити (Delete)*, *переглянути (View)* або *вийти (Exit).*

Якщо обрана операція *додати (Add)*, S-1: виконується потік *додати нове замовлення (Add a New Order)*.

Якщо обрана операція *змінити (Change)*, S-2: виконується потік *змінити замовлення (Change Order)*.

Якщо обрана операція *вилучити (Delete)*, S-3: виконується потік *вилучити замовлення (Delete Order).*

Якщо обрана операція *переглянути (View)*, S-4: виконується потік *переглянути замовлення (View Order).*

Якщо обрана операція *вийти (Exit)* прецедент завершується.

1.3 Під-Потоки.

S-1: *додати нове замовлення (Add a New Order).* Система відображає діалогове вікно, що містить поле, у якому менеджер повинен вибрати тип комп'ютера (настільний або ноутбук). Користувач вибирає необхідний тип. Система відображає поле для вибору клієнта й список можливих комплектуючих для обраного типу комп'ютера, у якому менеджер відзначає обрані клієнтом комплектуючі. Менеджер заповнює поля (E-2). Система запам'ятовує введені дані й роздруковує рахунок для оплати. Потім прецедент починається спочатку.

S-2: *змінити замовлення (Change Order).* Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідне замовлення зі списку або вводить номер замовлення в поле (E-3). Система відображає інформацію про дане замовлення. Менеджер робить необхідні зміни (E-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-3: *вилучити замовлення (Delete Order).* Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідне замовлення зі списку або вводить номер замовлення в поле (E-3). Система видаляє обране замовлення (E-4). Потім прецедент починається спочатку.

S-4: *переглянути замовлення (View Order).* Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідне замовлення зі списку або вводить номер замовлення в поле (E-3). Система відображає інформацію про обране замовлення. Коли менеджер перегляне інформацію, прецедент почнеться спочатку.

1.4 Альтернативні потоки.

E-1: уведене неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

E-2: обрані не всі комплектуючі, необхідні для складання комп'ютера або комплектуючих немає в наявності. Менеджер повинен змінити склад комп'ютера або завершити прецедент.

E-3: уведений неправильний номер замовлення. Менеджер повинен повторити введення або завершити прецедент.

E-4: система не може вилучити замовлення. Інформація зберігається, система вилучить замовлення пізніше. Виконання прецеденту триває.

Потік подій для прецеденту «Управління інформацією про клієнта».

2.1 Передумови.

2.2 Головний потік. Прецедент починає виконуватися, коли менеджер підключається до системи й уводить своє ім'я й пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій:
додати (Add), *змінити (Change), вилучити (Delete)*, *переглянути (View)* або *вийти (Exit).*

Якщо обрана операція *додати (Add)*, S-1: виконується потік *додати нового клієнта (Add a New Client)*.

Якщо обрана операція *змінити (Change)*, S-2: виконується потік *змінити дані про клієнта (Change Client Data)*.

Якщо обрана операція *вилучити (Delete)*, S-3: виконується потік *вилучити клієнта (Delete Client).*

Якщо обрана операція *переглянути (View)*, S-4: виконується потік *переглянути дані про клієнта (View Client Data).*

Якщо обрана операція *вийти (Exit)* прецедент завершується.

2.3 Під-Потоки.

S-1: *додати нового клієнта (Add a New Client).* Система відображає діалогове вікно, що містить поля для введення даних про нового клієнта. Користувач заповнює поля (E-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-2: *змінити дані про клієнта (Change Client Data).* Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер у поле (E-3). Система відображає інформацію про даного клієнта. Менеджер робить необхідні зміни (E-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-3: *вилучити клієнта (Delete Client).* Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер у поле (E-2). Система видаляє обраного клієнта (E-4). Потім прецедент починається спочатку.

S-4: *переглянути дані про клієнта (View Client Data).* Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер у поле (E-3). Система відображає інформацію про обраного клієнта. Коли менеджер перегляне інформацію, прецедент почнеться спочатку.

2.4 Альтернативні потоки.

E-1: уведене неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

E-2: заповнені не всі поля. Менеджер повинен заповнити незаповнені поля або завершити прецедент.

E-3: уведений неправильний номер клієнта. Менеджер повинен повторити введення або завершити прецедент.

E-4: система не може вилучити клієнта. Інформація зберігається, система вилучить клієнта пізніше. Виконання прецеденту триває.

Потік подій для прецеденту «Облік надходжння й видачі комплектуючих.

Передумови.

3.2 Головний потік. Прецедент починає виконуватися, коли завскладом підключається до системи й уводить своє ім'я й пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій: *додати (Add)*, *відзначити (Mark)* або *вийти (Exit)*.

Якщо обрана операція *додати (Add)*, S-1: виконується потік *внести, що зробили комплектуючі (Add a New Components)*.

Якщо обрана операція *відзначити (Mark)*, S-2: виконується потік *зробити оцінку про видачу комплектуючих (Mark Components)*.

Якщо обрана операція *вийти (Exit)* прецедент завершується.

3.3 Під-Потоки.

S-1: *внести, що зробили комплектуючі (Add a New Components).* Система відображає діалогове вікно, що містить поля для введення найменування комплектуючих, їх кількості, постачальника. Завскладом заповнює зазначені поля (E-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-2: *зробити оцінку про видачу комплектуючих (Change Order).* Система відображає список комплектуючих, що перебувають на складі. Завскладом напроти потрібних комплектуючих уводить кількість виданих (E-3). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

3.4 Альтернативні потоки.

E-1: уведене неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

E-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля або завершити прецедент.

E-3: зазначена кількість виданих комплектуючих, перевищує їх кількість на складі. Користувач повинен повторити введення або завершити прецедент.

Потік подій для прецеденту «Складання комп'ютерів».

4.1 Передумови.

4.2 Головний потік. Прецедент починає виконуватися, коли *Інженер по складанню* підключається до системи й уводить своє ім'я й пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій: *переглянути (View)*, *відзначити (Mark)* або *вийти (Exit)*

Якщо обрана операція *переглянути (View)*, S-1: виконується потік *Переглянути наряди на складання комп'ютера (View an Make Computer Order)*.

Якщо обрана операція *відзначити (Mark)*, S-2: виконується потік *зробити відмітку про статус комп'ютера, що збирається, по поряд (Mark Computer*).

Якщо обрана операція *вийти (Exit)* прецедент завершується.

4.3 Під-Потоки.

S-1: *Переглянути наряди на складання комп'ютера (View an Make Computer Order).* Система відображає діалогове вікно, що містить список нарядів і поле для введення номера наряду. Інженер вибирає необхідне наряди зі списку або вводить його номер у поле (E-2). Система відображає інформацію про обране наряди. Коли інженер перегляне інформацію, прецедент почнеться спочатку.

S-2: *зробити оцінку про статус комп'ютера, що збирається (Mark Computer*) Система відображає діалогове вікно, що містить список убрань. Біля необхідного наряди інженер робить оцінку про статус комп'ютера по даному вбранню. Інженер зберігає зміни. Потім прецедент починається спочатку.

4.4 Альтернативні потоки

E-1: уведене неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

E-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля або завершити прецедент.

E-3: уведений неправильний номер убрання. Інженер повинен повторити введення або завершити прецедент.

Потік подій для прецеденту «Вимога необхідних комплектуючих.

5.1 Передумови.

5.2 Головний потік. Прецедент починає виконуватися, коли інженер по складанню підключається до системи й уводить своє ім'я й пароль. Система перевіряє правильність пароля (E-1) і виводить можливі варіанти дій: *переглянути (View)*, *зажадати (Order)* або *вийти (Exit).*

Якщо обрана операція *переглянути (View)*, S-1: виконується потік *переглянути замовлені комплектуючі на складі (View Ordered Components on Warehouse).*

Якщо обрана операція *зажадати (Order)*, S-2: виконується потік *зажадати необхідні комплектуючі на складі (Order Required Components on Warehouse).*

Якщо обрана операція *вийти (Exit)* прецедент завершується.

5.3 Під-Потоки.

S-1: *Переглянути замовлені комплектуючі на складі (View Ordered Components on Warehouse).* Система відображає наступну інформацію про всі зроблені замовлення даним інженером по складанню: дата замовлення, найменування комплектуючих, їх кількість, замовлення виконане чи ні. Коли *інженер по складанню* переглянув список, він повідомляє систему. Прецедент починається спочатку.

S-2: *замовити необхідні комплектуючі на складі (Order Required Components on Warehouse).* Система відображає діалогове вікно, що містить поля для введення списку необхідних комплектуючих і їх кількості. *Інженер по складанню* заповнює його. Система запам'ятовує введені дані. Потім прецедент починається спочатку.

5.4 Альтернативні потоки.

E-1: уведене неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

Опис потоків подій для прецедентів *Управління інформацією про постачальників* і *Управління інформацією про комплектуючі* аналогічно опису для прецеденту *Управління інформацією про клієнта;* для прецеденту *Тестування комп'ютерів* - прецеденту *Складання комп'ютерів.*

4. Створення додаткової діаграми прецедентів. Як видно з опису потоку подій для всіх прецедентів кожний з них включає перевірку користувача. Перевірка здійснюється одноманітно для будь-якого прецеденту. Тому її можна представити у вигляді окремого прецеденту *Аутентифікація користувача*, пов'язаного відношенням включення з усіма іншими. Результат створення діаграми показаний на рис. 6.2:

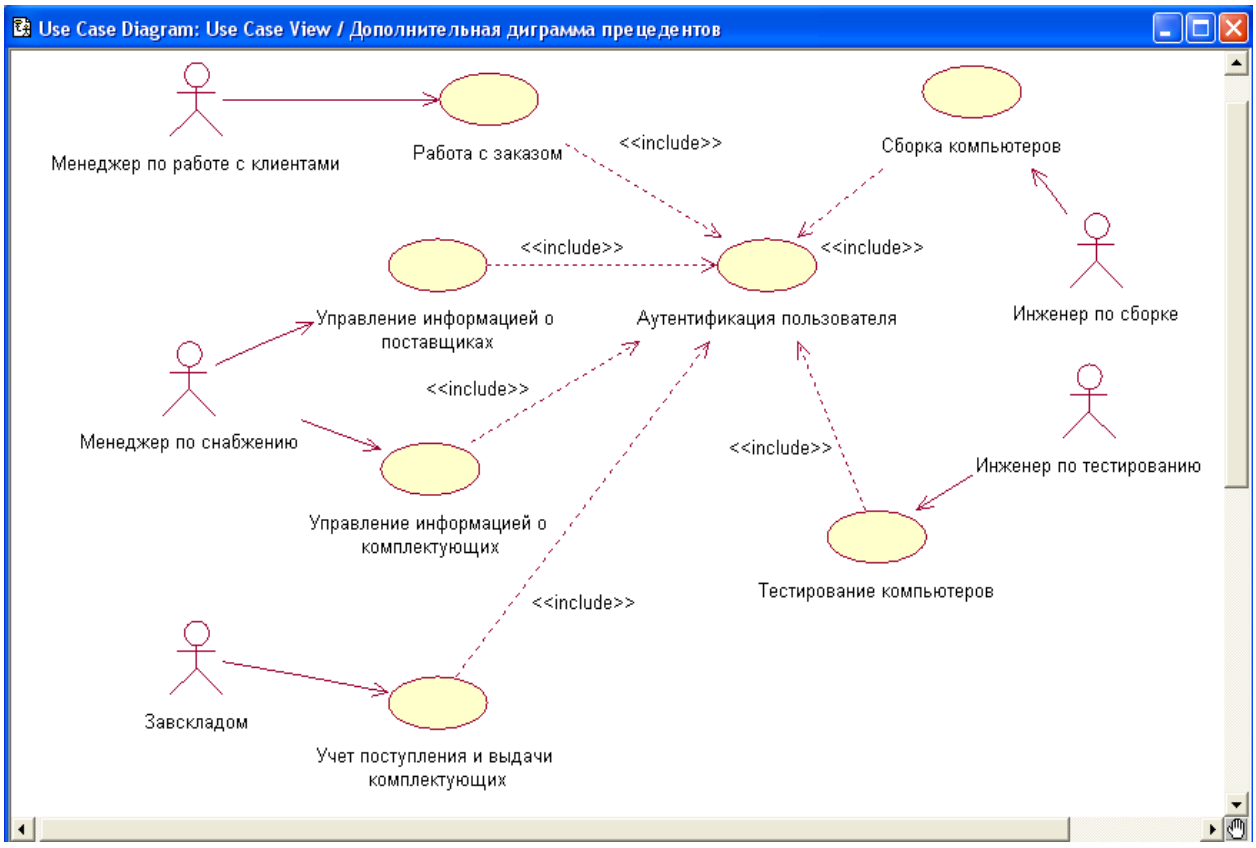


Рисунок 6.2 - Додаткова діаграма прецедентів

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №6

1. Відповідно до номеру навчальному журналі вибрати варіант предметної області (Список наведено нижче).
2. Створити головну діаграму прецедентів, задавши на ній варіанти використання й акторів;
3. Додати відносини між акторами й варіантами використання;
4. Створити додаткову діаграму прецедентів;
5. Додати опис до акторів і варіантів використання;

6. для кожного варіанта використання задати потік подій у вигляді окремого файлу й прикріпити його до варіанта використання.

ВАРІАНТИ ПРЕДМЕТНОЇ ОБЛАСТІ

№	Завдання
1	Робота лікарні
2	Робота супермаркету
3	Робота фірми з продажу комп’ютерів
4	Робота аптеки
5	Робота косметичного салону
6	Рух потягів
7	Функціонування аеропорту
8	Робота бухгалтерії
9	Робота кінотеатру
10	Робота бібліотеки
11	Випуск продукції на підприємстві
12	Робота хлібозаводу
13	Робота банку
14	Робота спортклубу
15	Робота центра зайнятості
16	Проведення олімпіади
17	Виробництво метало пластикових вікон
18	Транспортні послуги (вантажоперевезення)
19	Функціонування ресторану
20	Проведення міського культурно-масового заходу
21	Будівельна фірма
22	Робота складу
23	Робота посередницької фірми
24	Робота фермерського господарства
25	Робота ательє мод
26	Функціонування валютної біржі
27	Робота відділу кадрів
28	Робота міської телефонної станції

29	Робота комп’ютерного клубу
30	Робота таксопарку
31	Бізнес під ключ: пункт обміну
32	Робота туристичної фірми
33	Фірма з продажу побутової техніки
34	Робота шоколадної фабрики
35	Робота мобільних операторів
36	Організація зоопарку
37	Робота рекламного агентства

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як розшифрувати абревіатуру UML?
2. Яка версія UML є поточною?
3. Хто є авторами UML?
4. Які програмні засоби, що підтримують UML, ви знаєте?
5. Що таке нефункціональні вимоги? Як вони відображуються на діаграмах прецедентів?
6. Які способи зображення акторів ви знаєте?
7. В які відношення можуть вступати актори між собою?
8. В чому полягає сенс відношень включення і розширення?
9. Що таке точка розширення?
10. Перелічіть відомі Вам причини використання прецедентів.
11. Як прецеденти використовують в прямому та зворотному проектуванні?

ЛАБОРАТОРНА РОБОТА №7

ТЕМА РОБОТИ: Створення діаграми класів та об’єктів.

МЕТА РОБОТИ: Одержати навички побудови діаграм класів, створення пакетів і угруповання класів у пакети.

ТРИВАЛІСТЬ: 6 годин.

МЕТОДИЧНІ ВКАЗІВКИ

Діаграми класів (class diagram) використовуються для моделювання статичного виду системи з погляду проектування. *Діаграма класів* - діаграма, на якій показана множина класів, інтерфейсів, кооперацій і відносин між ними. Використовується в наступних цілях:

- для *моделювання словника* системи: припускає ухвалення рішення про те, які абстракції є частиною системи, а які - немає. За допомогою діаграм класів можна визначити ці абстракції і їх обов'язку;
- для *моделювання простих кооперацій*. Кооперація - це співтовариство класів, інтерфейсів і інших елементів, що працюють спільно для забезпечення деякої кооперативної поведінки;
- для *моделювання логічної схеми бази даних*.

Згідно з Мартіном Фаулером існують три різні точки зору на побудову діаграм класів або будь-якої інший моделі:

- *концептуальна точка зору* - діаграми класів служать для представлення понять досліджуваної предметної області. Ці поняття будуть відповідати класам, що їх реалізують, але пряма відповідність може бути відсутньою. Концептуальна модель може мати слабке відношення або взагалі не мати ніякого відношення до програмного забезпечення, що її реалізує, тому її можна розглядати без прив'язки до якоїсь мови програмування;
- *точка зору специфікації* - розглядається програмна система, при цьому розглядається тільки її інтерфейси, але не реалізація;
- *точка зору реалізації* - класи діаграми відповідають реальним класам програмної системи.

Для моделі з лабораторної роботи №6 розробимо діаграми класів.

1. Створення діаграми класів для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"

Діаграми класів будемо розглядати з концептуальної точки зору. Для спрощення завдання й щоб не захаращувати діаграми несуттєвими деталями методи setx, getx для кожного атрибута X класів задавати не будемо. Створимо в Логічному представленні браузера нову діаграму класів і назвемо її "*Add New Order*". У поле документації запишемо для неї наступний текст: "*Діаграма класів для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"*". Заповнення діаграми почнемо з визначення класів-сутностей. Розглянутий сценарій складається з:

- самого *замовлення*;
- *клієнта*, який робить замовлення;
- *комплектуючих виробів*, які входять у замовлення.

Створимо класи-сутності *Order (Замовлення)*, *Client (Клієнт)* і *Componentpart (Комплектуючий виріб)*. Оскільки в одне замовлення може входити багато різних комплектуючих виробів, і один комплектуючий виріб може входити в багато замовлень, то введемо ще один клас-сутність *Orderitem (Склад замовлення)*. Опишемо кожний клас.

Клас *Client:*

Параметр	**Значення**
Коментар	Клас, що представляє собою клієнта фірми
Атрибути	name : String - найменування клієнта address : String - адреса клієнта phone : String - телефон клієнта Всі атрибути мають модифікатор доступу - private
Операції	Addclient() - додавання нового клієнта Removeclient() - видалення існуючого клієнта Getinfo() - одержати інформацію про клієнта Всі операції мають модифікатор доступу - public

Клас *Order:*

Параметр	Значення
Коментар	Клас, що представляє собою замовлення, яке робить клієнт
Атрибути	ordernumber : Integer - номер замовлення orderdate : Date - дата оформлення замовлення ordercomplete : Date - дата виконання замовлення Всі атрибути мають модифікатор доступу - private
Операції	Create() - створення нового замовлення Setinfo() - занести інформацію про замовлення Getinfo() - одержати інформацію про замовлення Всі операції мають модифікатор доступу - public

Клас *Orderitem:*

Параметр	Значення
Коментар	Клас, що представляє собою пункт замовлення, яке робить клієнт
Атрибути	itemnumber : Integer - номер пункту замовлення quantity : Integer - кількість комплектуючих виробів price : Double - ціна за одиницю Всі атрибути мають модифікатор доступу - private
Операції	Create() - створення нового рядка замовлення Setinfo() - занести інформацію про рядок замовлення Getinfo() - одержати інформацію про рядок замовлення Всі операції мають модифікатор доступу - public

Клас *Componentpart:*

Параметр	**Значення**
Коментар	Клас, що представляє собою комплектуючі вироби
Атрибути	name : String – найменування manufacturer : String – виробник price : Double - ціна за одиницю description – опис Усі атрибути мають модифікатор доступу - private
Операції	Addcomponent() - додавання нового комплектуючого виробу Removecomponent() - видалення комплектуючого виробу Getinfo() - одержати інформацію про комплектуючий виріб Усі операції мають модифікатор доступу - public

Результат створення класів-сутностей показаний на рис. 7.1.

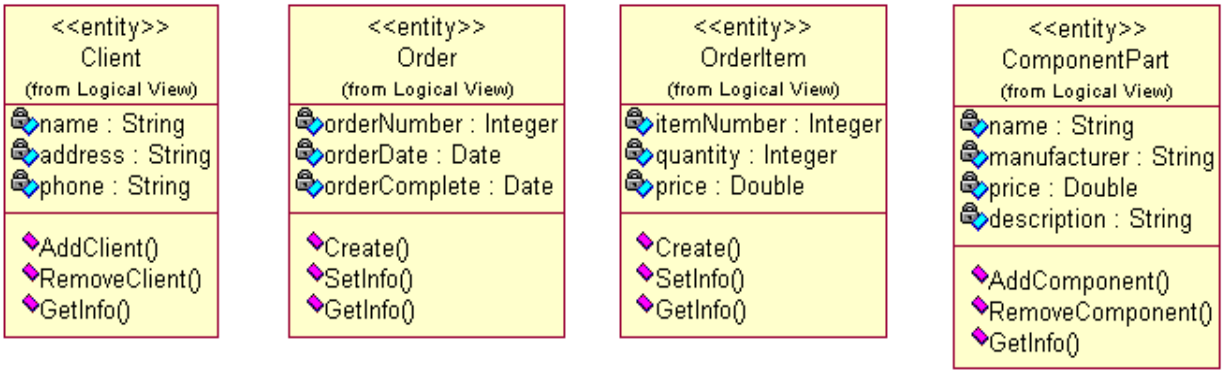


Рисунок 7.1– Створені класи-сутності

Додамо відносини між класами (рис.7.2):

- клас *Client* і *Order* - відношення асоціації, оскільки дані два класи просто зв'язано один з одним і ніякі інші типи зв'язків тут застосувати не можна. Один клієнт може зробити кілька замовлень, кожне замовлення надходить тільки від одного клієнта, тому кратність зв'язки з боку класу *Client* - 1, з боку *Order* - 1..n;

- клас *Order* і *Orderitem* - відношення композиції, оскільки рядок замовлення є частиною замовлення, і без нього існувати не може. В одне замовлення може входити кілька рядків замовлення, рядок замовлення ставиться тільки до одному замовленню, тому кратність зв'язки з боку *Order* - 1, з боку *Orderitem* - 1..n;
- клас *Orderitem* і *Componentpart* - відношення агрегації, оскільки комплектуючі вироби є частинами рядка замовлення, але й ті, і інші, являються самостійними класами. Один комплектуючий виріб може входити в багато рядків замовлення, в один рядок замовлення входить тільки один комплектуючий виріб, тому кратність зв'язки з боку *Componentpart* - 1, з боку *Orderitem* - 1..n.

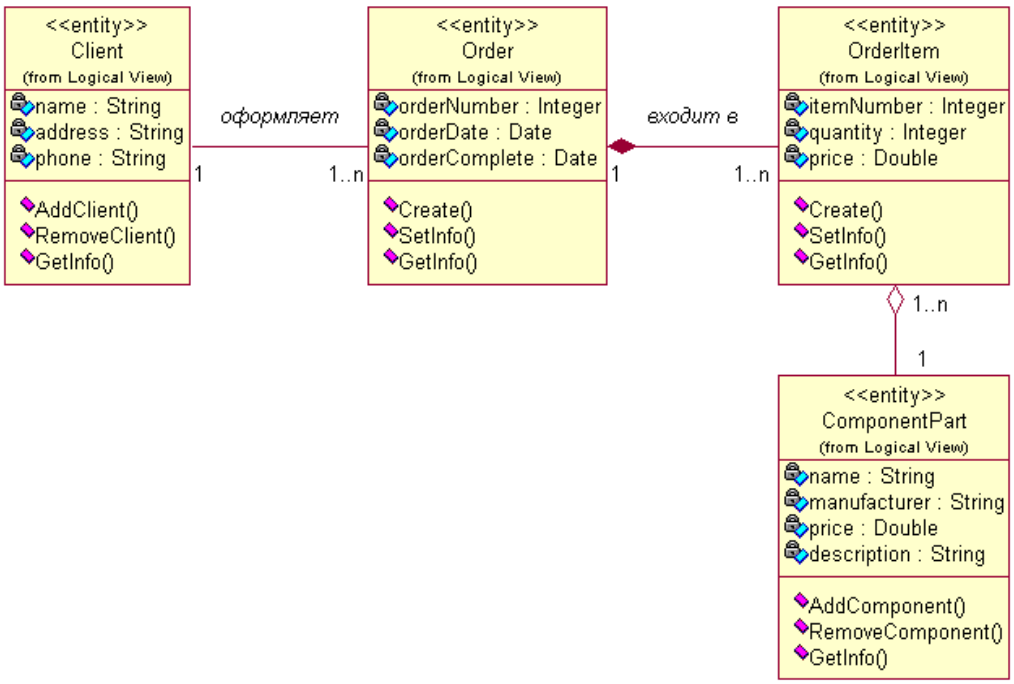


Рисунок 7.2 −Класи-Сутності й відносини між ними

Додамо тепер на діаграму граничні й управляючі класи (рис. 7.3). Розглянутий сценарій - це тільки одна з дій, які забезпечує прецедент "Робота із замовленням". Прецедент також дозволяє переглянути, відредагувати або вилучити замовлення. Це означає, що необхідно передбачити механізм, який дозволяє вибирати необхідну дію. Створимо для цього граничний клас *Orderoptions* (*Параметри роботи із замовленням*) з коментарем "*Клас, що забезпечує механізм роботи із замовленнями*". Також створимо граничний

клас *Addneworder (Додавання нового замовлення)*, який буде служити для додавання нових замовлень (коментар - "*Клас служить для додавання нових замовлень*". Відношення між цими класами - агрегація, оскільки в цьому випадку клас *Addneworder* розглядається як частина класу *Orderoptions*, частинами якого також будуть класи для перегляду, редагування й видалення замовлень. Кратність зв'язку 1 до 1, оскільки до складу класу *Orderoptions* входить тільки один клас *Addneworder*.

Перейдемо тепер до управляючих класів. Додамо управляючий клас *Ordermanager* (*Менеджер по роботі із замовленнями*) з коментарем "*Управляючий клас для обробки потоку подій прецеденту "Робота із замовленнями"*", який буде забезпечувати обробку потоку подій для розглянутого прецеденту. Даний клас буде пов'язаний із класами *Addneworder* і *Order*. Відношення між класами *Addneworder* і *Ordermanager* - односпрямована асоціація із кратністю зв'язку 1 до 1, оскільки один екземпляр класу *Addneworder* взаємодіє тільки з одним екземпляром класу *Ordermanager*. Відношення між класами *Ordermanager* і *Order* - односпрямована асоціація із кратністю зв'язку 1 до 1..n, оскільки один клас *Ordermanager* може взаємодіяти з декількома класами *Order*. Остаточний варіант діаграми класів показаний на рис. 7.3:

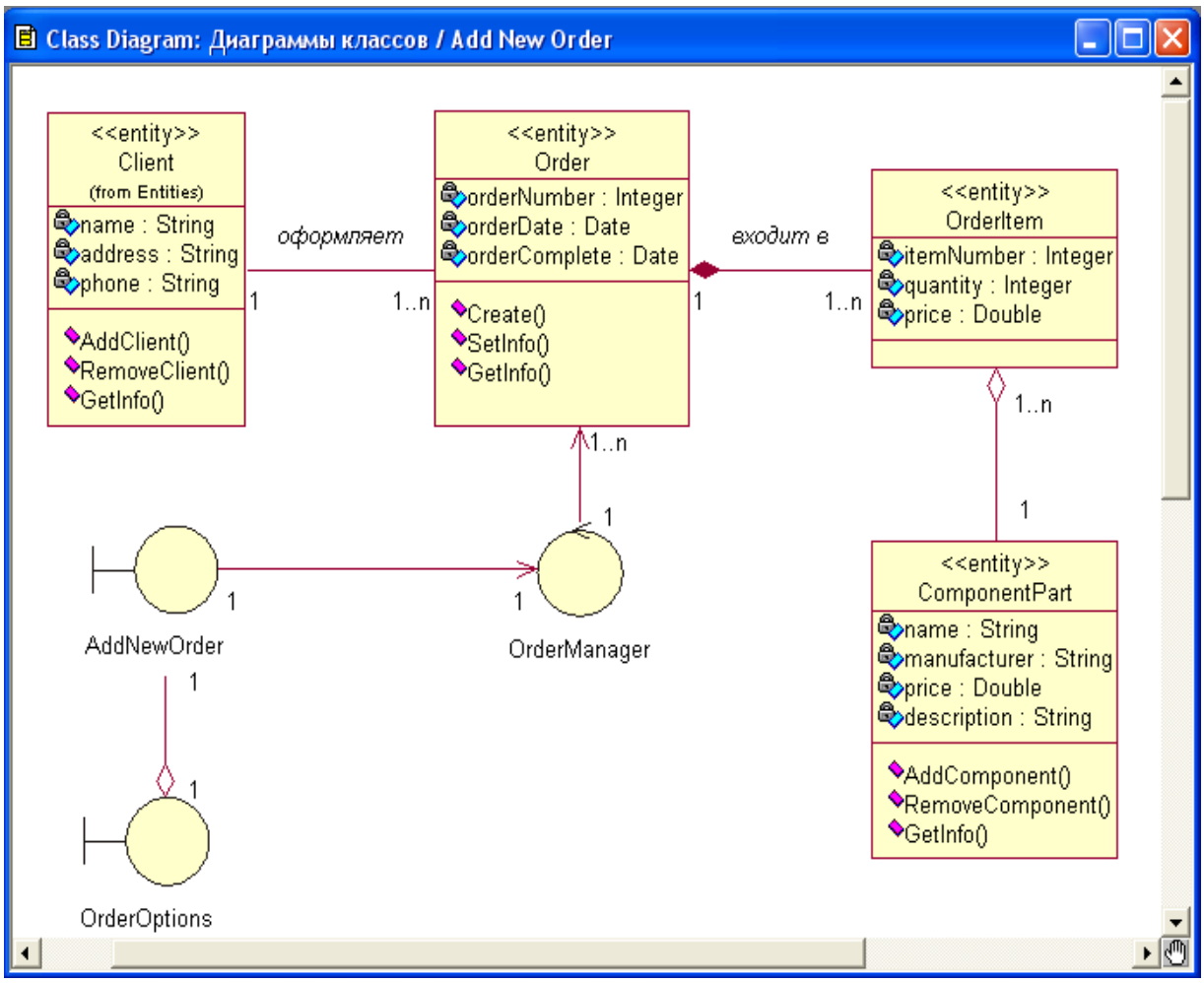


Рисунок 7.3−Підсумкова діаграма класів

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №7

1. Створити діаграму класів для одного зі сценаріїв діаграми прецедентів, створеної в попередній лабораторній роботі. Для кожного класу необхідно задати атрибути й операції. Кожний клас повинен бути докладно задокументований - необхідно задати текстовий опис самого класу, опису його атрибутів і операцій;
2. Надати
   a) створені діаграми прецедентів;
   b) короткий опис кожного актора й прецеденту;
   c) опис потоку подій для кожного варіанта використання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що таке діаграми класів?
2. Коли необхідно створювати діаграми класів?
3. Яка стратегія побудови діаграми класів?
4. Яким чином вибираються імена методів?
5. Яким чином зображуються асоціації на діаграмі класів?

ЛАБОРАТОРНА РОБОТА №8

ТЕМА РОБОТИ: Створення діаграми станів.
МЕТА РОБОТИ: Одержати навички побудови діаграм станів.
ТРИВАЛІСТЬ: 2 години.

МЕТОДИЧНІ ВКАЗІВКИ

Діаграми станів застосовуються, як правило, для моделювання поведінки класів, прецедентів або системи в цілому. Складемо діаграму станів для класу *Order (Замовлення)*, оскільки в нашій моделі він найбільше часто буде змінювати свій стан. Замовлення може перебуває в декількох станах:

- при створенні замовлення він переходить у стан *Ініціалізація,* у якому виконуються деякі попередні дії;
- після завершення ініціалізації замовлення переходить у стан *Відкрите*, у якому до замовлення додаються нові пункти. Вихід із цього стану можливий або у випадку скасування замовлення, або у випадку заповнення всіх необхідних пунктів замовлення;
- якщо заповнені всі необхідні пункти замовлення, то він переходить у стан *Закрите*, у якому відбувається виписка рахунку. Вихід із цього стану відбудеться тільки після того, як рахунок буде виписаний;

- якщо замовлення відмінне, то зі стану *Відкрите* він переходить у стан *Відмінене*. При виході із цього стану відбувається видалення всіх пунктів замовлення.

Діаграма станів для класу *Order* представлена на рис. 8.1.

Першим станом на діаграмі станів є початковий стан. При виконанні події "замовлення створене" замовлення переходить у стан *Ініціалізація.* При вході в цей стан виконується вхідна дія "Зберегти дату замовлення". Основна дія, яка буде виконуватися в плині всього часу, поки замовлення буде перебуває в цьому стані, це "Внести інформацію про клієнта". Перехід із цього стану в стан *Відкритий* відбудеться тільки при виконанні сторожової умови "ініціалізація завершена".

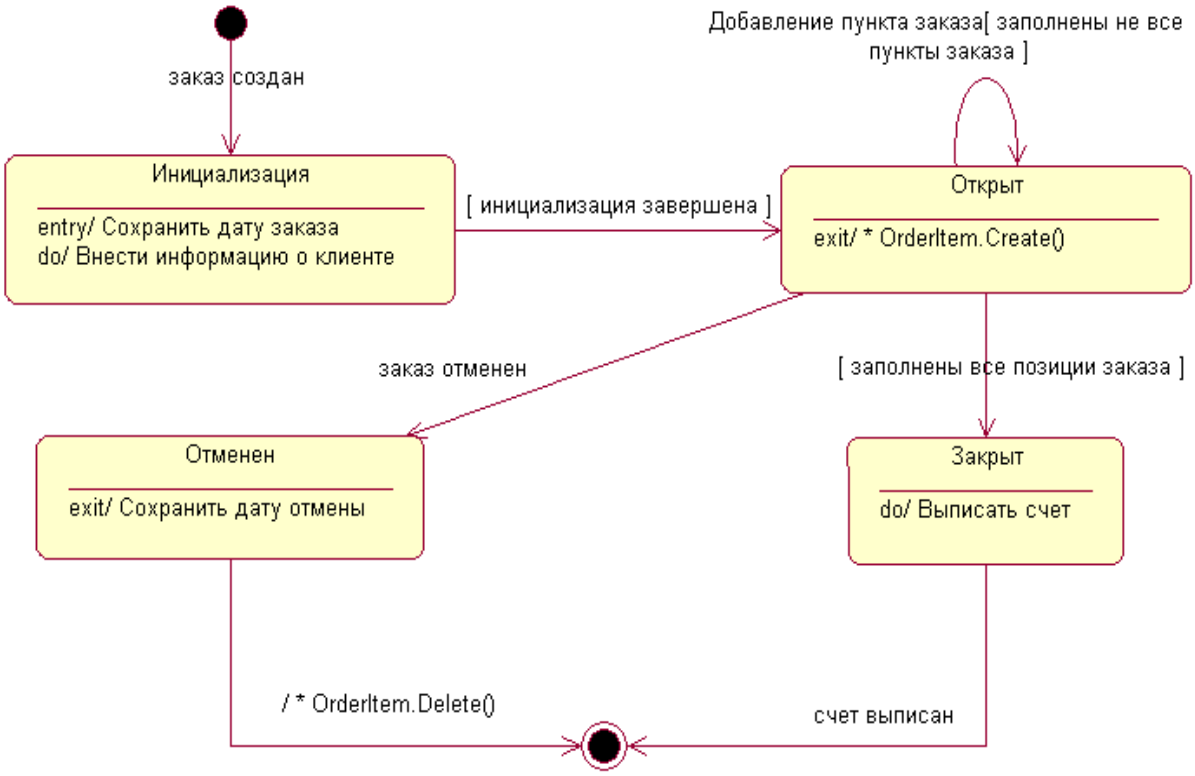


Рисунок 8.1−Діаграма станів для класу Order

У стані *Відкритий* є вихідна дія й перехід у себе. Перехід у себе означає, що подія ініціює перехід, відбувається вихід з поточного стану, виконується деяка дія, після чого відбувається повернення у вихідний стан. Оскільки при переході в себе відбувається вихід зі стану й повторний вхід у нього ж, те виконується дія, асоційоване з переходом, і, крім того, дія при

вході в стан. У стані *Відкритий* до замовлення додаються нові пункти, причому їх можна додати тільки в тому випадку, якщо є незаповнені пункти. Для показу цього ми використовували перехід у себе "Додавання пункту замовлення" зі сторожовою умовою "заповнені не всі пункти замовлення". Вихід із цього стану відбудеться у двох випадках - або коли виконається сторожова умова "заповнені всі позиції замовлення" (при цьому замовлення перейде в стан *Закрите*), або коли настане подія "замовлення відмінене" (при цьому замовлення перейде в стан *Відмінене*). При виході зі стану виконається дія виходу "* Orderitem.Create()" (створення пункту замовлення). Символ "*" указує на те, що ця дія виконається багато раз (по числу доданих пунктів у замовлення).

У стані *Закритий* є присутнім тільки внутрішня дія - "Виписати рахунок". У цей стан замовлення переходить зі стану *Відкрите* тільки при виконанні сторожової умови "заповнені всі позиції замовлення". Вихід із цього стану й перехід у кінцеве відбудеться при настанні події "рахунок виписаний".

У стан *Відмінене* замовлення переходить зі стану *Відкрите* при настанні події "замовлення відмінене". При виході з нього виконується дія виходу "Зберегти дату скасування". При переході із цього стану в кінцеве виконується дія "* Oderitem.Delete()" (видалення пункту замовлення). Тут також стоїть "*", оскільки ця дія буде виконуватися багато раз.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №8

1. Розробити діаграму станів та опис станів для одного з раніше розроблених класів або прецедентів.
2. Опис станів представити у вигляді таблиці:

Стан	Опис стану

КОНТРОЛЬНІ ПИТАННЯ

1. Чому потрібно будувати різні діаграми при моделюванні системи?
2. Які діаграми відповідають статичному уявленню про систему?

3. Ви розроблюєте комп'ютерну програму для гри в шахмати. Яка діаграма UML була б корисною у цьому випадку? Чому?
4. Складіть список питань потенційного користувача такої програми. Поясніть, чому Ви хотіли б задати саме їх.
5. Які три принципи лежать в основі ООП?
6. Що таке інтерфейс? На якому з базових принципів ООП заснований механізм інтерфейсів?
7. Що таке n-арна асоціація?
8. В чому полягає різниця між агрегацією та композицією?
9. Що таке клас асоціації?

ЛАБОРАТОРНА РОБОТА №9

ТЕМА РОБОТИ: Створення діаграм діяльності.
МЕТА РОБОТИ: Одержати навички побудови діаграм діяльності.
ТРИВАЛІСТЬ: 4 години.

МЕТОДИЧНІ ВКАЗІВКИ

1. Створення діаграми діяльності для бізнес-процесу підприємства.

Розглянемо в цілому, що відбувається на підприємстві від моменту оформлення замовлення на складання комп'ютера до видачі готового комп'ютера. Після оформлення замовлення менеджер по роботі із клієнтами передає його менеджерові по складанню, який перш ніж почати складання замовляє необхідні комплектуючі зі складу. На складі завідувач підбирає необхідні комплектуючі (у випадку їх відсутності замовляє їх у менеджера по постачанню) і передає їх інженерові по складанню. Після одержання комплектуючих менеджер по складанню здійснює складання комп'ютера й передає його інженерові по тестуванню. Якщо комп'ютер не пройшов тестування, він повертається для повторного складання. При успішному завершенні тестування комп'ютер передається на склад на зберігання. Зі складу комп'ютер на вимогу передається інженерові по роботі із клієнтами, який оформляє на нього документи й видає клієнтові.

Для створення діаграми дій необхідно клацнути правою кнопкою миші за Поданням Варіантів Використання і у меню, що з'явилося, вибрати пункт New > Activity Diagram, увести її ім'я, після чого двічі клацнути по ній у браузері, щоб відкрити її.

Результат побудови діаграми показаний на рис. 9.1:

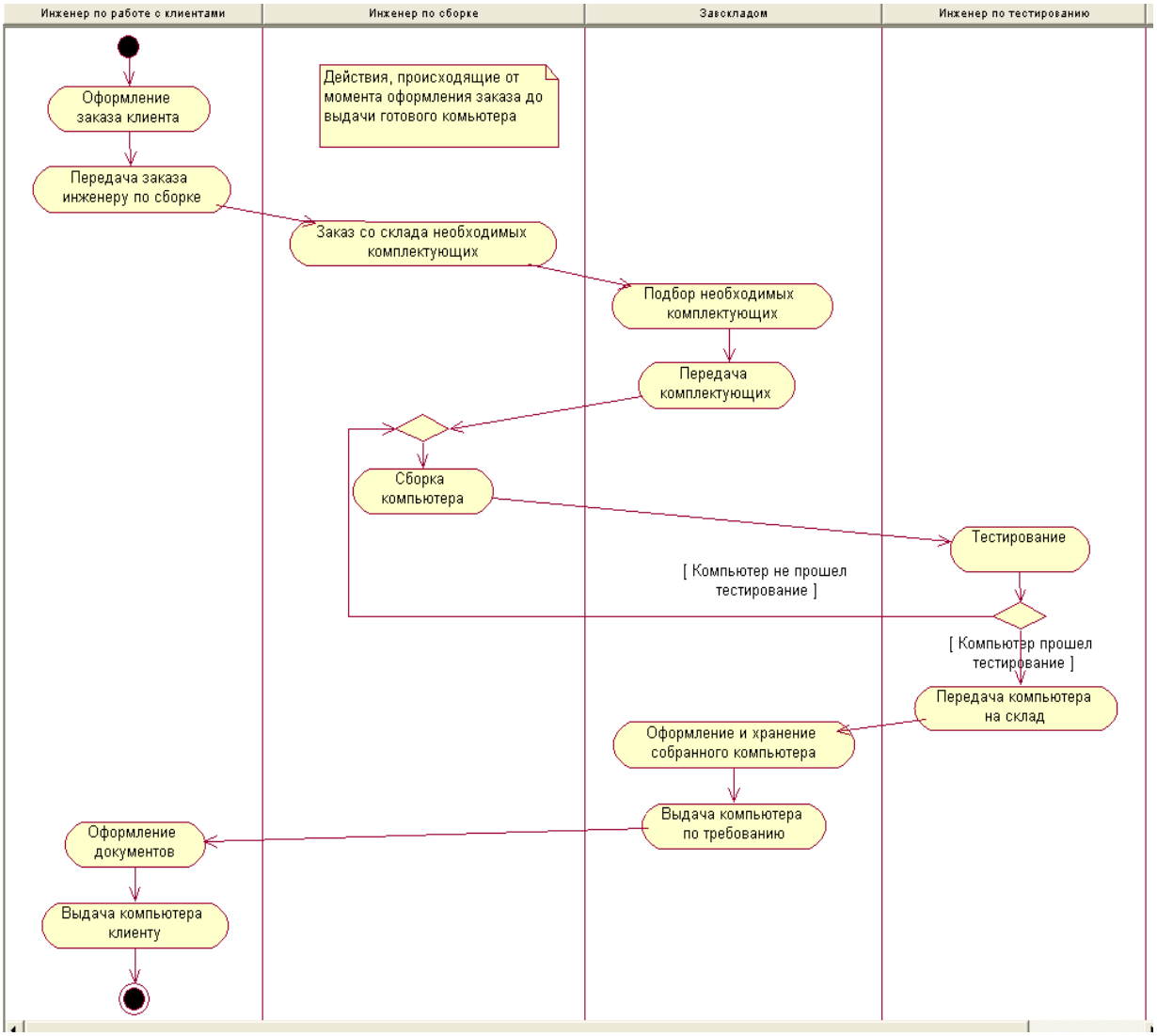


Рисунок 9.1–Діаграма діяльності бізнес-процесу

2. Створення діаграми діяльності потоку події варіанта використання "Робота із замовленням"

Потік подій варіанта використання "Робота із замовленням" складається з головне потоку, під-потоків і альтернативних потоків. Щоб не захаращувати діаграму покажемо потік подій на декількох діаграмах діяльності. На першій з них (умовно назвемо її головною) покажемо дії для основного потоку й пов'язаний з ним альтернативний потік

(рис. 9.2). Під-Потоки можна буде показати шляхом декомпозиції відповідного дії головної діаграми.

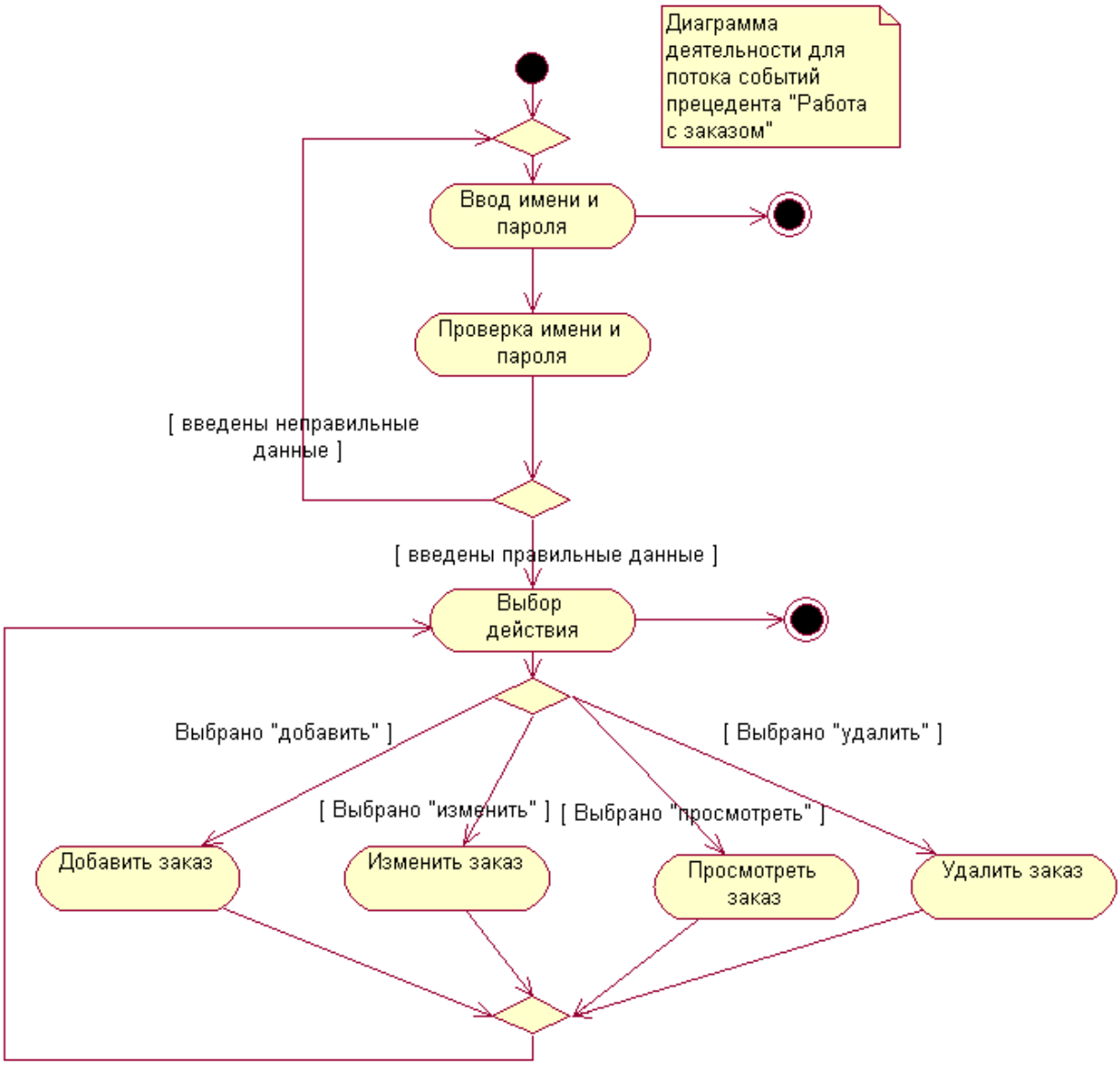


Рисунок 9.2−Діаграма діяльності для потоку подій прецеденту "Робота із замовленням"

Для декомпозиції дії діаграми діяльності слід клацнути по ній правою кнопкою миші й у меню, що з'явився, вибрати пункт Sub Diagrams > New Activity Diagram.

Приклад декомпозиції дії. На рис. 9.3 показана діаграма діяльності для під-потоку "Додати замовлення", яка є декомпозицією дії "Додати замовлення" головної діаграми діяльності.

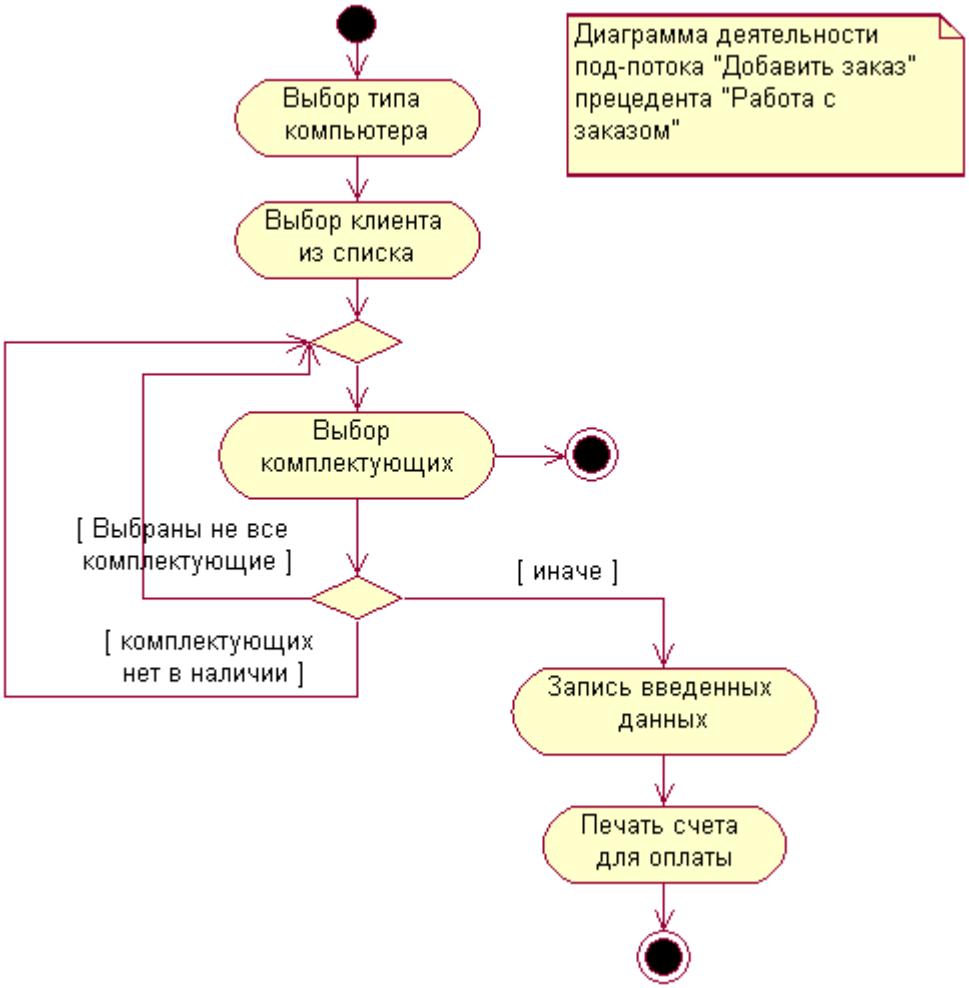


Рисунок 9.3 −Діаграма діяльності для дії "Додати замовлення"

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №9

1. Створити діаграму діяльності, що описує один з бізнес-процесів обраної предметної області;

2. Створити діаграму діяльності, що описує потік подій одного з варіантів використання, створеного в лабораторній роботі № 6.

КОНТРОЛЬНІ ПИТАННЯ

1. Які види діаграм (крім діаграм діяльності) можна ще використовувати для моделювання динаміки системи?
2. Чим діаграми діяльності відрізняються від блок-схем? Які переваги це надає розробникам?
3. Що таке траєкторія об'єкта?

4. Чим кінечний стан потоку відрізняється від кінечного стану діяльності?
5. Чим моделювання процесів відрізняються від моделювання операцій?
6. Чи можливе використання діаграм діяльності безвідносно до ООП?

ЛАБОРАТОРНА РОБОТА №10

ТЕМА РОБОТИ: Створення діаграм послідовності й кооперації.

МЕТА РОБОТИ: Одержати навички побудови діаграм послідовності й кооперації.

ТРИВАЛІСТЬ: 4 години.

МЕТОДИЧНІ ВКАЗІВКИ

Створювати діаграми взаємодії будемо для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням". У цьому сценарії крім основного потоку існують ще й альтернативні потоки. Хоча стандарт мови UML допускає розгалуження на діаграмах послідовності й кооперації, ми, щоб не захаращувати наші діаграми, обмежимося розглядом тільки випадку, коли користувач правильно вводить свій пароль, правильно заповнює необхідні поля й уведені дані без помилок зберігаються в базі даних. Якщо буде потреба альтернативні потоки можна показати на додаткових діаграмах послідовності й кооперації.

Діаграми взаємодії будемо створювати в Логічному представлені браузера. Для того, щоб відокремити ці діаграми від інших (які ми вже створили або створимо надалі), створимо спочатку новий пакет у Логічному представлені - *Діаграми взаємодії*, у якому будуть розташовуватися створені далі діаграми.

Побудова будь-якої діаграми взаємодії починається з визначення переліку об'єктів, які будуть брати участь у взаємодії. Для обраного сценарію в лабораторній роботі № 7 була розроблена діаграма класів. Екземпляри класів цієї діаграми й будуть учасниками діаграм взаємодії.

У даній роботі ми обоє типу діаграм взаємодії будемо будувати з нуля.

Створення діаграми послідовності для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"

Побудова діаграми послідовності починається з розміщення на ній об'єктів, які будуть обмінюватися повідомленнями. Спочатку необхідно розмістити об'єкти, які посилають повідомлення, а потім об'єкти, що одержують їх. Ініціатором взаємодії виступає актор *Менеджер по роботі із клієнтами.* Тому на діаграмі він буде перебуває в лівому куті. Далі розміщаємо (рис. 10.1):

- об'єкт класу *Orderoptions* (*Параметри роботи із замовленням*), відповідальний за вибір можливої дії із замовленням у розглянутому прецеденті;
- об'єкт класу *Addneworder* (*Додавання нового замовлення*), відповідальний за додавання замовлення;
- об'єкт класу *Ordermanager* (*Менеджер по роботі із замовленнями*), відповідальний за обробку потоку подій розглянутого прецеденту;
- об'єкт класу *Order* (*Замовлення*);
- об'єкт класу *Client* (*Клієнт*);
- об'єкт класу *Componentpart* (*Комплектуючий виріб*).

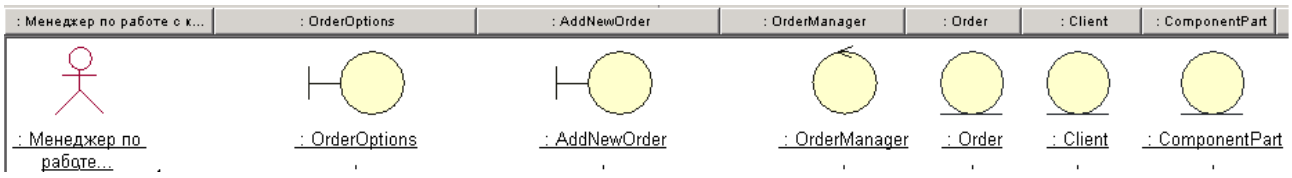


Рисунок 10.1–Розташування об'єктів на діаграмі послідовності

Тепер на діаграмі слід розмістити повідомлення, якими будуть обмінюватися об'єкти (табл. 10.1, рис. 10.2).

Таблиця 10.1 - Повідомлення, якими будуть обмінюватися об'єкти

Номер повідомлення	Об'єкт - відправник повідомлення	Об'єкт - одержувач повідомлення	Назва
1	Менеджер по роботі із клієнтами	Orderoptions	уведення пароля
2	Orderoptions	Orderoptions	перевірка пароля
3	Менеджер по роботі із клієнтами	Orderoptions	вибір операції "додати"
4	Orderoptions	Addneworder	відображення полів уведення
5	Менеджер по роботі із клієнтами	Addneworder	вибір типу комп'ютера
6	Addneworder	Ordermanager	одержання списку клієнтів
7	Ordermanager	Client	одержання списку клієнтів
8	Client	Addneworder	список клієнтів
9	Addneworder	Addneworder	відображення списку клієнтів
10	Менеджер по роботі із клієнтами	Addneworder	вибір клієнта
11	Addneworder	Ordermanager	одержання списку комплектуючих
12	Ordermanager	Componentpart	одержання списку комплектуючих
13	Componentpart	Addneworder	список комплектуючих
14	Addneworder	Addneworder	відображення списку комплектуючих
15	Менеджер по роботі із клієнтами	Addneworder	* вибір необхідних комплектуючих
16	Менеджер по роботі із клієнтами	Addneworder	зберегти замовлення
17	Addneworder	Ordermanager	передача управління
18	Ordermanager	Order	Зберегти

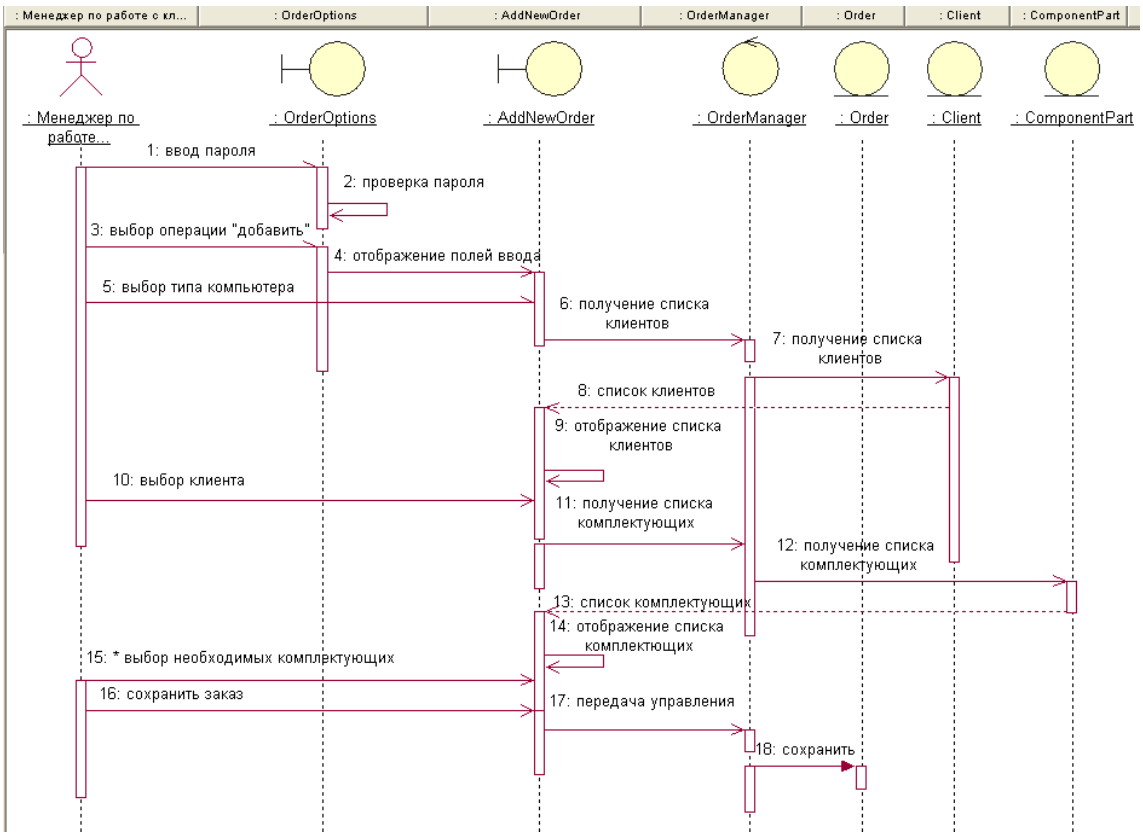


Рисунок 10.2– Підсумкова діаграма послідовності

Створення діаграми кооперації для сценарію "Додати нове замовлення" прецеденту "Робота із замовленням"

Побудова діаграми кооперації починається з розміщення на ній об'єктів, які будуть обмінюватися повідомленнями. Перелік об'єктів на даній діаграмі такої ж, як і на попередній. Далі необхідно додати на діаграму зв'язку між об'єктами, які обмінюються повідомленнями (рис. 10.3): «*Менеджер по роботі із клієнтами* й *Addneworder», «Менеджер по роботі із клієнтами* й *Orderoptions», «Addneworder* і *Orderoptions», «Addneworder* і *Ordermanager», «Addneworder* і *Client», «Addneworder* і *Componentpart», «Ordermanager* і *Client», «Ordermanager* і *Componentpart», «Ordermanager* і *Order».*

Останніми на діаграму кооперації додаються повідомлення між об'єктами. Перелік повідомлень між об'єктами такої ж, як і на діаграмі послідовності (рис. 10.4).

Останніми на діаграму кооперації додаються повідомлення між об'єктами. Перелік повідомлень між об'єктами такої ж, як і на діаграмі послідовності (рис. 10.4).

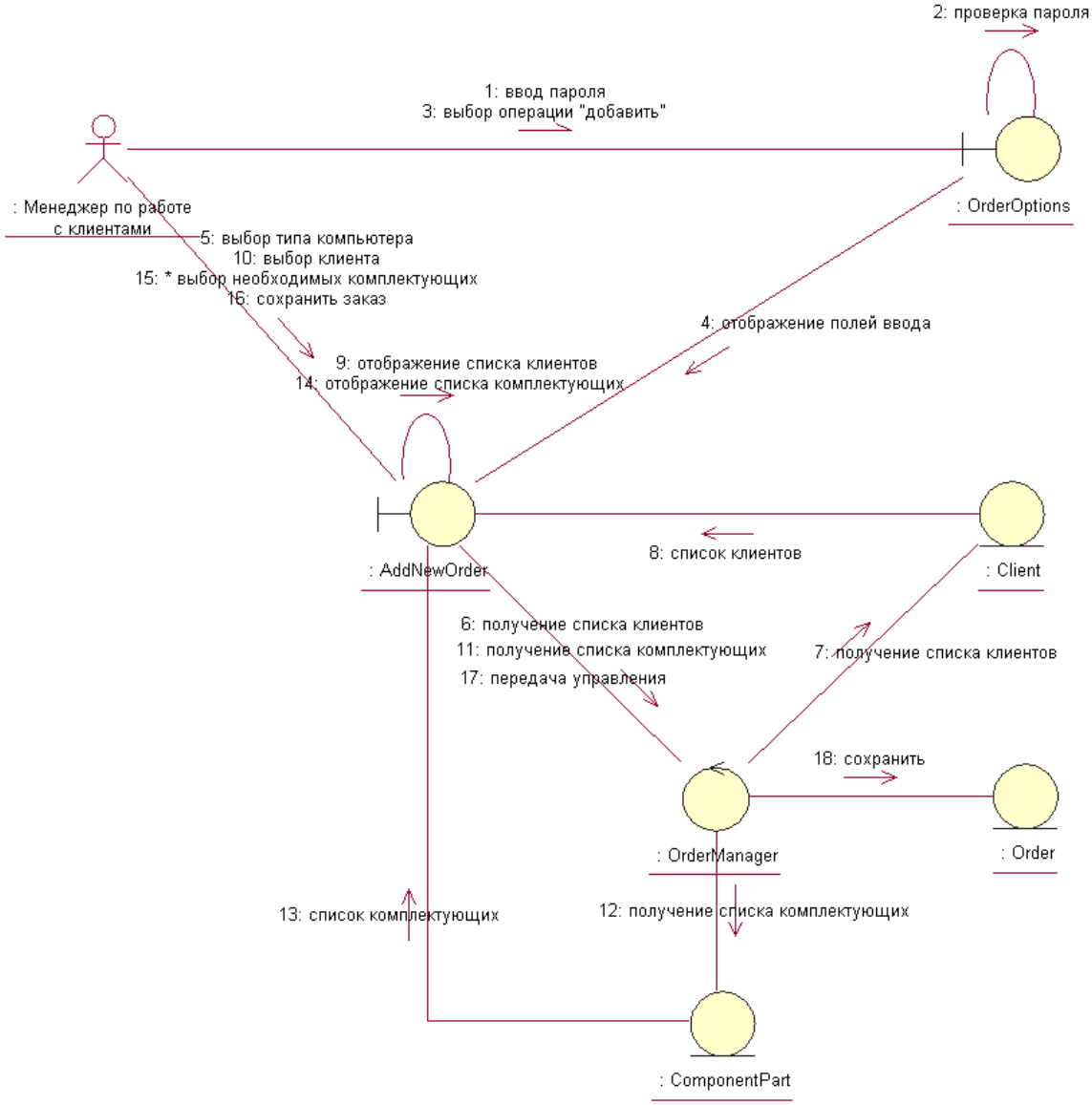


Рисунок 10.4 –Підсумкова діаграма кооперації

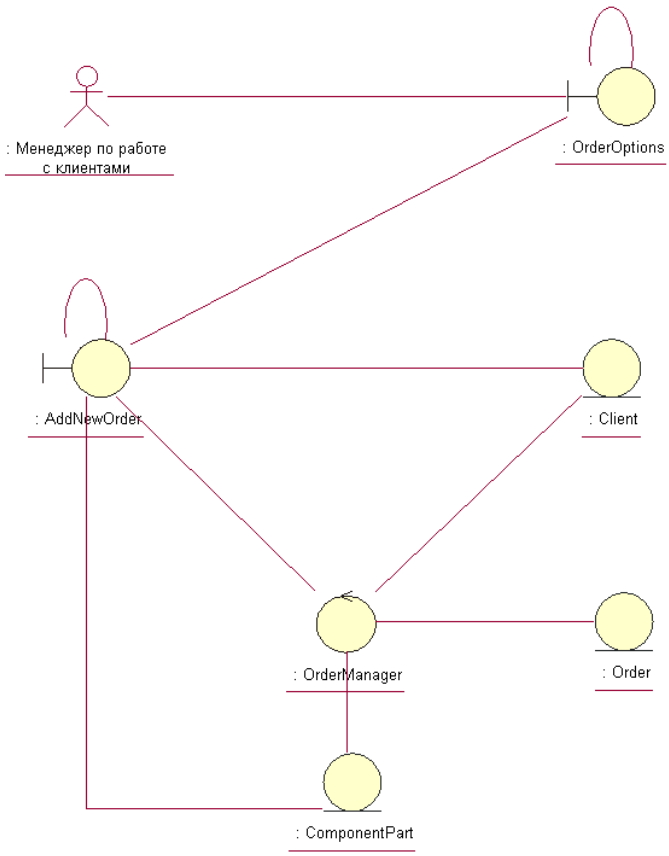


Рисунок 10.3–Об'єкти й зв'язку на діаграмі кооперації

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ № 10

1. Створити діаграму послідовності й кооперації для одного зі сценаріїв будь-якого прецеденту, створеного в лабораторній роботі № 6.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Чи може діаграма послідовностей містити об'єкт з лінією життя, але без фокусу управління?
2. Чим відрізняються представлення кооперації на рівні специфікації і на рівні прикладів?
3. У чому різниця між активними та пасивними об'єктами?
4. Чим асинхронне повідомлення відрізняється від синхронного?
5. Що таке мультиоб'єкт?
6. Що таке композитний об'єкт і як він пов'язаний з поняттям кооперації?

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Зінов’єва О. Г., Шаров С. В. Проєктування інформаційних систем : конспект лекцій. Запоріжжя : ТДАТУ ім. Дмитра Моторного, 2024. 148 с.
2. Зелінська О. В., Потапова Н. А., Волонтир Л. О. Інформаційні системи та технології в галузі : навч. посіб. Вінниця : ВНАУ, 2020. 263 с.
3. Ізмайлова О. В. Проектування інформаційних систем : навч. посіб. Київ : КНУБА, 2022. 88 с.
4. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) : навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського ; уклад.: О. С. Коваленко, Л. М. Добровська. Київ : КПІ ім. Ігоря Сікорського, 2020. 192 с.
5. Литвин В. В., Пасічник В. В., Шаховська Н. Б. Проектування інформаційних систем : навч. посіб. (затв. МОН України). Львів : Магнолія 2006, 2021. 380 с.
6. Величко О. М., Гордієнко Т. Б., Інтелектуальні інформаційні системи; структура і застосування : підручник. Одеса : Олді+, 2022. 728 с.
7. Hoffer J. A., George J. F., Valacich J. S. Modern Systems Analysis and Design. 9th ed. London : Pearson, 2020. 528 p.
8. Gallaugher J. Information Systems: A Manager's Guide to Harnessing Technology. Minnesota : University of Minnesota Libraries Publishing, 2019. 664 p.
9. Зінов’єва О. Г. Використання CASE-засобів для проектування інформаційних систем. *Українські студії в європейському контексті*. 2023. № 7. С. 220–227.
10. Кузнєцов М. С, Климкович Т. О., Савчук Л. М. Об’єктно-орієнтована технологія створення програмних систем : навч. посіб. Дніпропетровськ : НМетАУ, 2008. 80 с.
11. Кузнєцов М. С. Об’єктно-орієнтоване програмування з використанням

UML та мови C++ : навч. посіб. Дніпропетровськ : НМетАУ, 2003. 90 с.

12. Кузнєцов М. С., Климович Т. О. Процедурне програмування з використанням мови С : навч. посіб. Дніпропетровськ : НМетАУ, 2005. 84 с.
13. Тарасова К. І. Еволюція інформаційних систем в економіці. *Бізнес Інформ*. 2020. №4 . С. 289–295
14. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем : навч. посіб. Ірпінь : Ун-т держ. фіск. служби України, 2019. 321 с.
15. Бевз О. М., Папінов В. М., Скидан Ю. А. Проектування програмних засобів систем управління : навч. посіб. Вінниця : ВНТУ, 2010. Ч. 1 : Основи об`єктно-орієнтованого проектування. 125 с.
16. Основні принципи ООП: інкапсуляція, наслідування, поліморфізм. *Це госи! Wiki*. URL: https://gos-it.fandom.com/wiki (дата звернення: 05.08.2024).
17. Основи об'єктно-орієнтованого програмування. *IT-Academy*. URL: https://www.it-academy.by/course/osnovy-programmirovaniya/osnovy-oop/ (дата звернення: 15.08.2024).

Навчально-методичне видання

Лозовська Людмила Іванівна,
Савчук Лариса Миколаївна,
Климкович Тетяна Олександрівна

ТЕХНОЛОГІЯ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Навчально-методичні рекомендації
з виконання лабораторних робіт

В авторській редакції

Комп’ютерна верстка Л. І. Лозовська

Експертний висновок склала канд. екон. наук, доц. К. О. Удачина

Зареєстровано НМВ УДУНТ (№ 787 від 07.11.2024)

Формат 60х84 1/16. Ум. друк. арк. 4,21. Обл.-вид. арк. 2,12.

Зам. № 103

Видавець: Український державний університет науки і технологій
вул. Лазаряна, 2, ауд. 2216, м. Дніпро, 49010.
Свідоцтво суб’єкта видавничої справи ДК № 7709 від 14.12.2022

Адреса видавця та оперативної поліграфії:
вул. Лазаряна, 2, Дніпро, 49010