

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ**

**ЗАТВЕРДЖУЮ**

**Ректор** \_\_\_\_\_ С.В. Іванов  
(підпис)

« \_\_\_\_\_ » \_\_\_\_\_ 2014 р.

**ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ**

**ЛАБОРАТОРНИЙ ПРАКТИКУМ**

для студентів напряму підготовки 6.050101

«Комп'ютерні науки»

денної та заочної форм навчання

Всі цитати, цифровий та фактичний матеріал, бібліографічні відомості перевірені. Написання одиниць відповідає стандартам

Підпис авторів \_\_\_\_\_/Литвинов В.А./

\_\_\_\_\_ /Гладка М.В./

\_\_\_\_\_ /Хлобистова О.А./

«12» лютого 2014 р.

Реєстраційний номер електронного лабораторного практикуму у НМВ

51.17-26.06.2014

**СХВАЛЕНО**

на засіданні кафедри  
Інформаційних систем  
Протокол № 6  
від 13.02.2014 р.

**Технологія створення програмних продуктів** [Електронний ресурс]: лабораторний практикум для студентів напряму підготовки 6.050101 «Комп'ютерні науки» денної та заочної форм навч. / уклад. В.А. Литвинов, М.В. Гладка, О.А. Хлобистова: НУХТ, 2014.— 86 с.

Рецензент О.М. М'якшило, к.т.н., доц. \_\_\_\_\_

Укладачі: **В.А. Литвинов**, д-р техн. наук, проф  
**М.В. Гладка**,  
**О.А. Хлобистова**, канд. техн. наук, доц.

Відповідальний за випуск **В.В. Самсонов**, канд. техн. наук, проф.

**Подано в авторській редакції**

## Зміст

ВСТУП .....	4
ЛАБОРАТОРНА РОБОТА №1 Методи структурування програм. Метод дублювання кодів.....	6
ЛАБОРАТОРНА РОБОТА №2 Методи структурування програм. Метод введення змінної стану програми .....	10
ЛАБОРАТОРНА РОБОТА №3, 4 Розробка технічного завдання .....	13
Контрольні питання .....	18
ЛАБОРАТОРНА РОБОТА №№ 5,6 Структурний підхід до програмування. Стадія «Ескізний проект».....	20
ЛАБОРАТОРНА РОБОТА №№ 7, 8 Структурний підхід до програмування. Стадія «Технічний проект» .....	23
ЛАБОРАТОРНА РОБОТА №№ 9, 10 Структурний підхід до програмування. Стадія «Реалізація. Розробка програмного засобу» .....	35
ЛАБОРАТОРНА РОБОТА №№ 11, 12 Структурний підхід до програмування. Стадія «Налагодження програмного засобу та розробка програмної документації».....	39
ЛАБОРАТОРНА РОБОТА №№ 13, 14 Структурний підхід до програмування. Стадія «Тестування програмного засобу». ....	47
ЛАБОРАТОРНА РОБОТА №№ 15, 16 Моделювання проекту з допомогою пакету Rational Rose (Технологія RUP) .....	51
ЛАБОРАТОРНА РОБОТА №№ 17, 18 Моделювання проекту з допомогою пакету Rational Rose. Побудова діаграм прецедентів і класів. Побудова діаграм послідовностей і діяльності.....	67
ДОДАТОК 1. Варіанти завдань до лабораторної роботи №1 .....	76
ДОДАТОК 2. Варіанти завдань до лабораторної роботи №2 .....	79
ДОДАТОК 3. Індивідуальні завдання для проектування програмних продуктів (роботи 3-18) .....	83

## ВСТУП

Предметом дисципліни "Технологія створення програмних продуктів" є методи, призначені для створення проектних основ, моделей та алгоритмів, описового представлення майбутнього проекту, документації призначеної для розробки програмних комплексів.

Використання автоматизованих систем і систем автоматизованого виробництва постійно розширюється. Успіх цих систем безпосередньо залежить від нашої здатності випередити їх розробку і впровадження описом всього комплексу проблем, які необхідно вирішити, зазначенням того, які функції системи повинні бути автоматизовані, визначенням точок інтерфейсу людина-машина і того, як взаємодіє система зі своїм оточенням. Іншими словами, етап проектування системи є критичним для створення високоякісних систем. Даний лабораторний практикум дозволить студентові навчитись виявляти задачі, що потребують автоматизації, оформлювати документації, необхідну для створення автоматизованих проектів, застосовувати методи тестування комп'ютерних проектів, зрозуміти поняття що стосуються методів та моделей управління проектами, ознайомитись з системою моделювання Rational Rouse.

При виконанні лабораторних робіт закріплюються знання, отримані на лекціях, де студенти вивчають основні поняття, структуру, вимоги до створення і моделювання комп'ютерних проектів. Студенти вивчають методи відслідковування помилок на етапі створення проекту, та методи тестування готових програмних комплексів. "Технологія створення програмних продуктів" – це дисципліна, на якій будуть визначені підсистеми, компоненти і способи їх з'єднання, що задає обмеження, при яких система має функціонувати, що вибирає найбільш ефективне поєднання людей, машин і програмного забезпечення для реалізації системи, та документальне оформлення усіх цих визначень.

На лабораторних роботах студенти самостійно виконують індивідуальні завдання на ПК у одному із вивчених раніше програмних засобів. Індивідуальні варіанти обираються відповідно до номеру за списком у журналі. Виконання кожної лабораторної роботи передбачає ознайомлення студента з лабораторним практикумом та його теоретичну підготовку з відповідних розділів дисципліни. До кожного розділу наведено перелік питань для контролю. Об'єкт автоматизації для дослідження визначається викладачами індивідуально. На захист виконаної студентом лабораторної роботи оформлюється окремий звіт, за потреби демонструється електронний документ із виконаним завданням. Підсумкові оцінки, отримані студентом за виконання лабораторних робіт і відповіді на контрольні запитання, враховуються при обчисленні модульних оцінок та семестрової підсумкової оцінки з даної навчальної дисципліни.

Послідовне виконання лабораторних робіт другого розділу формує цілісне документальне представлення, що є необхідним для створення автоматизованої системи управління, починаючи від визначення самого проекту, опису робіт, представлення моделі проекту, алгоритм виконання програми, розробка програмного модуля, методи відслідковування помилок.

При виконанні лабораторних робіт студент отримує знання що розвинути аналітичні здібності, нададуть змогу правильно оформлювати проектну документацію, виступати керівником проекту з впровадження автоматизованих систем. Отримані знання забезпечать високу кваліфікацію студента, як спеціаліста в сфері аналітики так і керівника проектів.

# ЛАБОРАТОРНА РОБОТА №1

## Методи структурування програм. Метод дублювання кодів

**Мета роботи** полягає в вивченні та практичній реалізації деяких методів структурування (в сенсі концепції структурного програмування) неструктурованих програм, що представлені блок-схемою програмних блоків (модулів) мережевого вигляду.

### Завдання

1. Одержати від викладача номери варіантів неструктурованих блок-схем (додаток 1)
2. Перетворити заданий варіант неструктурованої блок – схеми (додаток 1) в структуровану, використовуючи метод дублювання кодів.
3. Виконати обчислення трудомісткості виконання тестування для кожного блоку в структурованій та неструктурованій програмі. Співставити орієнтовну сумарну трудомісткість тестування первинної та результатної схем.
4. Зробити висновки за отриманими результатами.

### Підготовка до роботи

Проробити відповідні розділи [1 - 5] та наступні теоретичні відомості.

### Теоретичні відомості

Структурне програмування — методологія програмування (модель конструювання програмного забезпечення) запропонована в 1970-х роках голландським науковцем Дейкстрою (Edsger Wybe Dijkstra), була розроблена та доповнена Ніклаусом Віртом. Ґрунтується на теоремі Бьома-Якопіні (Corrado Bohm, Giuseppe Jacopini), яка була опублікована у 1966 р.

Згідно з цією методологією будь-яка програма може бути створена використовуючи три конструкції:

- *послідовне виконання* - одноразове виконання операції в порядку запису їх (операцій) в тексті програми;
- *розгалуження* – виконання певної операції або декількох операцій в залежності від стану певної, наперед заданої умови;
- *цикл* – багаторазове виконання операції або групи операцій за умови виконання деякої наперед заданої умови. Така умова називається - умова продовження циклу.

Кожна конструкція являє собою блок із одним входом і одним виходом.

Блок **Слідування** передбачає послідовне (лінійне) виконання операторів програми.

Блок **Вибір** являє собою точку прийняття рішення про подальший перебіг виконання операторів програми. Вибір здійснюється однією із трьох структур:

- *if* (єдиний вибір)
- *if...else* (подвійний вибір)
- *switch* або *case* (множинний вибір)

Усі три структури можна звести до одного типу *if*.

Блок **Повторення** реалізується одним із трьох способів:

- *структура while*
- *структура do/while*
- *структура for*

Усі три структури можна звести до структури *while*.

Структурована програма складається із вищеназваних блоків за двома правилами: **пакетування** (вихід одного блоку з'єднується із входом наступного) і **вкладення** (будь-який блок може бути замінений на структуру керування вибору або повторення).

Таким чином, структуровані програми містять всього сім типів структур керування, які з'єднуються всього двома способами.

Переваги:

- такі програми легко створювати та тестувати;
- скорочується термін розробки програми;
- полегшується супровід програми
- скорочення варіантів побудови програми, що знижує складність програмного забезпечення;
- логічно зв'язані операції знаходяться ближче один до одного, що полегшує аналіз алгоритму дозволяючи обходитись без блок-схем алгоритму (хоча наявність такої блок-схеми полегшує розуміння роботи алгоритму).

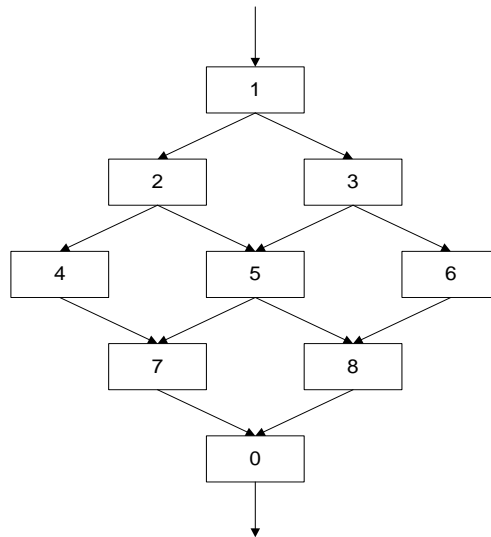
Недоліки:

- непропорційне зростання складності програми при збільшенні об'єму коду програми;
- складність створення паралельних програм.

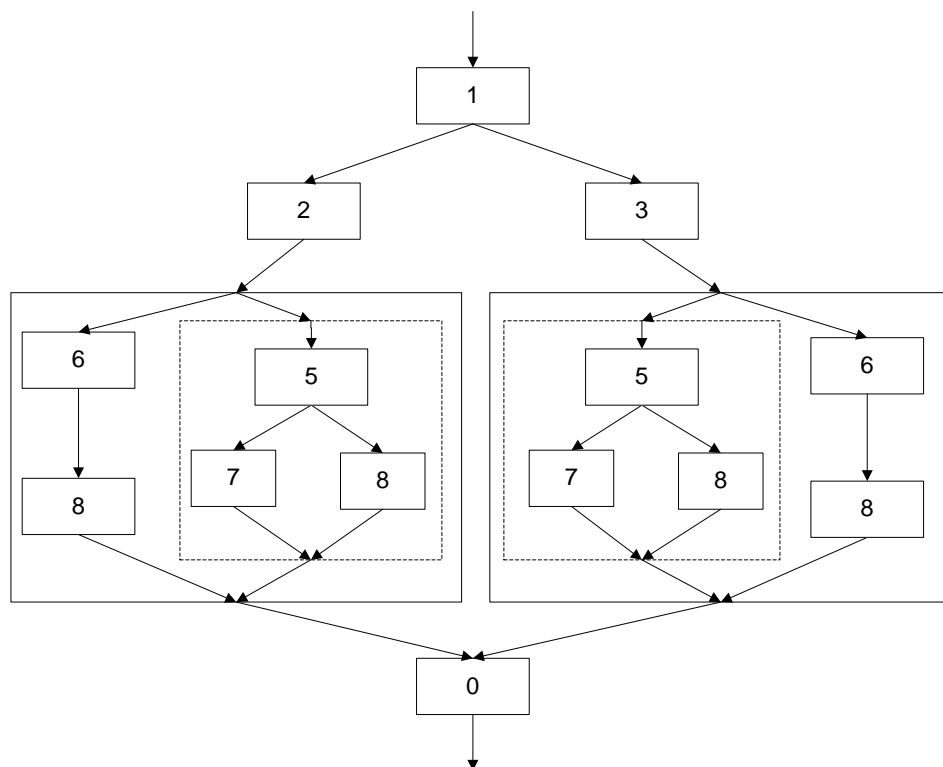
### Метод дублювання кодів

Розглянемо програму, гіпотетична блок-схема якої наведено на рис.1.1 . Блоки цієї блок – схеми не задовольняють вимозі «один вхід – один вихід» . Метод структурування програми полягає в дублюванні (або виділенні відповідних формальних підпрограм з формальними параметрами) тих модулів, які мають кілька входів,- тобто перетворення початкової мережевої структури в деревоподібну. Результати такого перетворення наведені на рис. 1.2. Виділені групи модулів являють собою деякі «чорні скриньки –

віртуальні блоки, що мають один вхід і один вихід. Завдяки показаному перетворенню «читабельність» програми стає кращою, бо вона має меншу логічну складність. Поряд з цим, знижується і очікувана трудомісткість тестування. Так, зокрема, тестування модуля 7 схеми рис.1.1 потребує спільного тестування модулів 1,2,3,4,5,7. Для схеми рис.1.2 модуль 7 є підпрограмою з формальними параметрами, завдяки чому його правильність може бути встановлена незалежно від контексту, в якому він використовується.



*Рис 1.1. Неструктурований модуль виконання програми.*



*Рис. 1.2. Структурований модуль виконання програми*



*Таблиця 1.1. Виконання обчислень складності тестування.*

Неструктурована модель			Структурована модель		
№ блоку	Шлях	Сума	№ блоку	Шлях	Сума
1	1	1	1	1	1
2	1,2	2	2	1,2	2
3	1,3	2	3	1,3	2
4	1,2,4	3	4	1,2,4	3
5	1,2,3,5	4	5	1,2,5	4
6	1,3,6	3	6	1,3,6	3
7	1,2,3,4,5,7	6	7	1,2,5,7	4
8	1,2,3,5,6,8	6	8	1,2,6,8	4
0	1,2,3,4,5,6,7,8	8	0	1,2,3,4,5,6,7,8	8
Сума кількості тестів		35	Сума кількості тестів		31

### **Зміст звіту**

Заданий варіант первинної блок – схеми.

Перетворена блок – схема з виділенням «чорних скриньок» (структурованих фрагментів «один вхід – один вихід»).

Результати співставлення очікуваної трудомісткості тестування.

### **Контрольні питання**

1. Криза 60-х років у програмуванні.
2. Основні етапи розвитку технологій програмування.
3. Принципи етапу структурного програмування.
4. Призначення і сутність процесу структурування програм
5. Особливості програм обчислюваного типу і задача їхньої структуризації.
6. Структуризація програм методом дублювання.

### **Рекомендована література**

1. Городняя Л.В. Основы функционального программирования. М.: Интернет-Университет Информационных технологий [Текст] / Л.В. Городняя, 2004. - 272 с
2. Жоголев Е.А. Технология программирования [Текст] / Е.А. Жоголев. — М.: Научный мир, 2004. — 216 с.
3. Лавров С.С. Программирование. Математические основы, средства, теория [Текст] / С.С. Лавров - СПб: БХВ-Петербург, 2001. - 184 с
4. Одинцов И.О. Профессиональное программирование. Системный подход [Текст] / И.О. Одинцов. — СПб: ВНУ-СПб., 2002. — 512 с.
5. Електронний ресурс: Структурне програмування. Режим доступу: <http://uk.wikipedia.org/>

## ЛАБОРАТОРНА РОБОТА №2

### Методи структурування програм. Метод введення змінної стану програми

**Мета роботи** полягає в вивченні та практичній реалізації деяких методів структурування програм логічного типу, що представлені блок-схемою програмних блоків (модулів) з великою кількістю умовних переходів.

#### Завдання

1. Одержати від викладача номери варіантів неструктурованих блок-схем (додаток 2)
2. Перетворити заданий варіант неструктурованої блок – схеми (додаток 2) в структуровану, використовуючи метод Ашкрофта – Манни.
3. Дослідити можливість спрощення результатної схеми.

#### Підготовка до роботи

Проробити відповідні розділи [1 - 5] та наступні теоретичні відомості.

#### Теоретичні відомості

Поняття «структурного програмування» полягає у сукупності деяких принципів написання програм у відповідності з набором жорстких правил і має ціллю полегшення процесів тестування та поліпшення «читабельності» результатних програм. Наслідком дотримання цих принципів є підвищення продуктивності роботи програмістів. В жорстко структуровані мови (наприклад, ПАСКАЛЬ) ці принципи заздалегідь закладені в синтаксис і семантику мови).

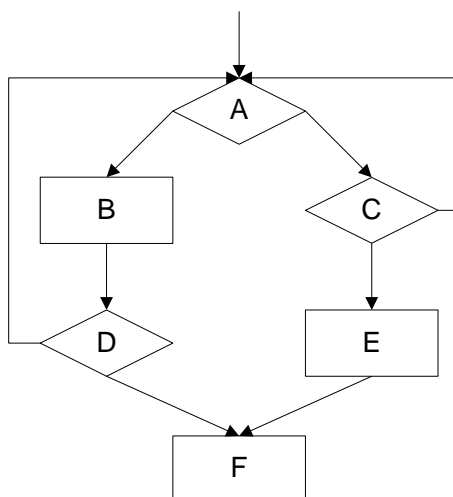


Рис 2.1.

Правила структурного програмування прості : всі операції в програмі повинні виконуватися у вигляді функціональних блоків з одним входом і

одним виходом, управляючої структури IF – THEN – ELSE (додатково можливо CASE) та циклічної структури DO – WHILE (додатково можливо REPEAT (PERFORM) – UNTIL, FOR – DO). Найбільш важливими є вимоги виключення (в максимально можливій мірі) «шкідливих» операторів GO TO та блоків з кількома входами.

Розглянемо метод перетворення неструктурованої блок-схеми деякої довільної програми в структуровану за допомогою введення додаткової змінної.

### Метод введення змінної стану програми (метод Ашкрофта-Манни)

На відміну від першого методу, орієнтованого на «решітчасту», мережеву структуру, цей метод може бути застосований до будь-яких програм (в тому числі програм з циклами, складними логічними конструкціями тощо) та припускає автоматичне застосування.

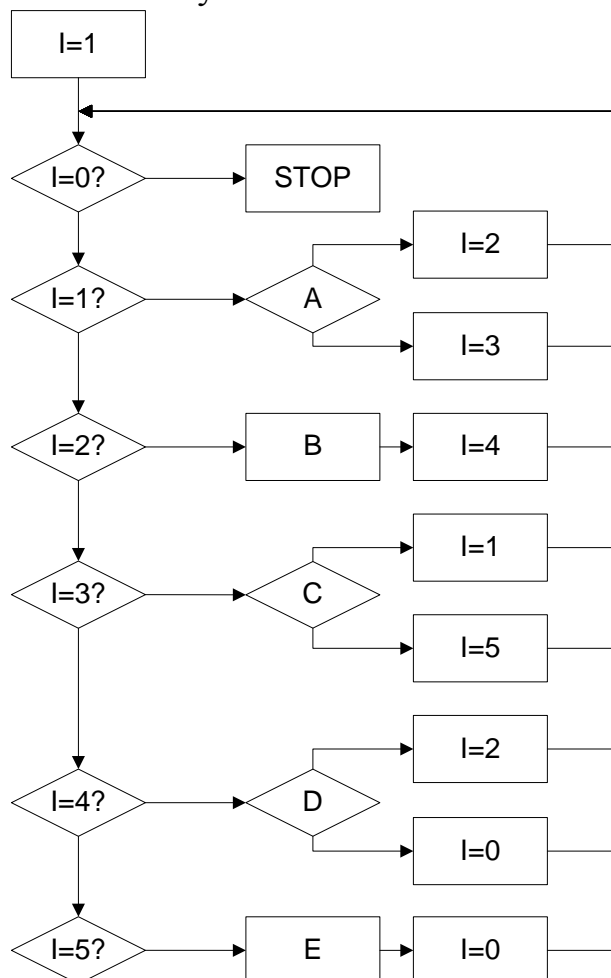


Рис 2.2.

Приклад первинної неструктурованої програми наведено на рис. 2.1. Процес перетворення складається з наступних кроків.

Кожному блоку неструктурованої програми надається довільний унікальний номер (звичайно №1 – це перший блок, №0 – останній).

В програму вводиться нова змінна з довільним ім'ям (нехай у нашому випадку буде *i*).

Функціональні блоки неструктурованої схеми замінюються блоками, які виконують ті ж самі обчислення і додатково присвоюють змінній *i* номер наступного блоку у відповідності з первинною схемою рис. 2.1.

В програму закладається логіка покрокового виконання з опитуванням стану змінної і запуском відповідного функціонального блоку.

Відповідним чином структурована схема програми наведена на рис 2.2.

Відзначимо, що схема рис 2.2 може бути трохи спрощена з урахуванням того, що після блоку *B* завжди виконується блок *D*.

### **Зміст звіту**

Заданий варіант первинної блок – схеми.

Перетворена блок – схема.

Спрощена перетворена блок – схема з коментарями.

### **Контрольні питання**

1. Чому структуровані програми повинні бути простішими в налаштуванні ?
2. Чому структуровані програми мають кращу читабельність?
3. Особливості програм логічного типу і задача їхньої структуризації
4. Сутність і особливості методу Ашкрофта – Манни
5. Можливості вдосконалення структурованої програми

### **Рекомендована література**

1. Городняя Л.В. Основы функционального программирования. М.: Интернет-Университет Информационных технологий [Текст] / Л.В. Городняя, 2004. - 272 с
2. Жоголев Е.А. Технология программирования [Текст] / Е.А. Жоголев. — М.: Научный мир, 2004. — 216 с.
3. Лавров С.С. Программирование. Математические основы, средства, теория [Текст] / С.С. Лавров - СПб: БХВ-Петербург, 2001. - 184 с
4. Одинцов И.О. Профессиональное программирование. Системный подход [Текст] / И.О. Одинцов. — СПб: ВНУ-СПб., 2002. — 512 с.
5. Електронний ресурс: Структурне програмування. Режим доступу: <http://uk.wikipedia.org/>

## **ЛАБОРАТОРНА РОБОТА №3, 4**

### **Розробка технічного завдання**

**Мета:** Навчитися розробляти технічне завдання на створення інформаційної системи для підприємства (відділу, підрозділу).

**Завдання:**

1. Розробити технічне завдання на систему, що включає наступні пункти:
  - Загальні положення про розроблювану систему.
  - Призначення і цілі створення системи.
  - Характеристика об'єкту автоматизації.
  - Вимоги до системи.
  - Технічні вимоги, які включають.
  - Економічні показники.
  - Порядок контролю і приймання об'єкта.
2. Структурну послідовність оформлення ТЗ наведено у додатку 4.

**Підготовка до роботи**

Проробити відповідні розділи джерел [1 - 6] щодо тем «Етапи розробки програмного забезпечення. Постановка задачі», «Інженерія вимог до програмного забезпечення».

Ознайомитися з матеріалами розділу 2 навчального посібника [4]

### **Теоретичні відомості**

#### **3.1. Зміст та структура технічного завдання**

Технічне завдання (ТЗ) — вихідний документ для проектування споруди чи промислового комплексу, конструювання технічного пристрою (приладу, машини, системи керування тощо), розробки автоматизованої системи, створення програмного продукту або проведення науково-дослідних робіт (НДР), відповідно до якого проводиться виготовлення, приймання при введенні в дію та експлуатація відповідного об'єкта.

Згідно з ГОСТ 34.602-89 ТЗ є основним документом, що визначає вимоги і порядок створення (розвитку або модернізації) інформаційної системи, відповідно до якого проводиться її розробка і приймання при введенні в дію.

Технічне завдання на розробку АСУ (ТЗ) повинно містити такі розділи:

1. Загальні положення, де вказують повну назву системи та її умовне позначення, а також перелік документів, на підставі яких створюється система, термін розробки системи і порядок подання замовнику результатів роботи. Бажано, щоб ТЗ було узгоджено представником підприємства, де студент проходив технологічну практику.

2. Призначення і цілі створення системи. В підрозділі «Призначення системи» вказують від діяльності, що автоматизується, і перелік об'єктів автоматизації, де буде впроваджена система. В підрозділі «Цілі створення системи» наводять назви і необхідні значення технічних, технологічних, техніко-економічних та інших показників, які повинні бути досягнуті в результаті впровадження АСУ.

3. Характеристика об'єкту автоматизації містить стислі відомості про об'єкт автоматизації та відомості про умови експлуатації та навколишнє середовище.

4. Вимоги до системи складаються з таких підрозділів: вимоги до системи в цілому; вимоги до функцій (задач), які виконує система; вимоги до видів забезпечення.

### **3.2. Вимоги до розроблюваної інформаційної системи**

***Вимоги до розроблюваної інформаційної системи в цілому містять:***

- вимоги до структури і функціонування системи;
- вимоги до чисельності і кваліфікації персоналу системи та умов його роботи;
- показники призначення;
- вимоги до надійності;
- вимоги безпеки;
- вимоги до ергономіки та технічної естетики;
- вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи;
- вимоги до захисту інформації від несанкціонованого доступу;
- вимоги до збереження інформації в разі аварії;
- вимоги до захисту від впливу зовнішніх чинників;
- додаткові вимоги.

***Вимоги до структури і функціонуванню системи містять:***

- перелік підсистем, їх призначення і основні характеристики, вимоги до числа рівнів ієрархії та ступеню централізації системи;
- вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами системи;
- вимоги і характеристики взаємозв'язків системи, що створюється із суміжними системами, а також засобів обміну інформацією;
- вимоги до режимів функціонування системи;
- вимоги до діагностування системи;
- перспективи розвитку та модернізації системи.

***Вимоги до чисельності і кваліфікації персоналу системи містять:***

- вимоги до чисельності персоналу (або користувачів) системи, вимоги до кваліфікації персоналу, порядку підготовки та контролю його знань та навичок, режим роботи персоналу.

***У вимогах до показників призначення наводять***

- значення параметрів, що характеризують ступінь відповідності системи її призначенню, а саме: ступінь адаптації системи до зміни процесів та методів управління, допустимі межі модернізації системи, імовірно-часові характеристики системи, за яких зберігається цільове призначення системи.

***Вимоги до надійності системи містять:***

- склад та значення кількісних показників надійності для системи в цілому або її підсистем;
- перелік аварійних ситуацій, відносно яких повинні бути регламентовані вимоги до надійності, та значення відповідних показників;
- вимоги до надійності технічних засобів та програмного забезпечення;
- вимоги до методів оцінки та контролю показників надійності на різних стадіях створення системи у відповідності з діючими нормативними документами.

***Вимоги безпеки містять***

- вимоги до забезпечення безпеки під час монтажу, налагодження, експлуатації, обслуговування та ремонту технічних засобів системи ( захист від дії електричного струму, електромагнітних полів, шуму, тощо), вимоги до рівня освітлення, вібраційних та шумових навантажень.

***У вимогах до ергономіки та технічної естетики наводять***

- показники АС, що задають необхідну якість взаємодії людини з ПК і комфортність умов роботи персоналу.

***Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи складаються з:***

- умов і регламенту експлуатації, які повинні забезпечити використання технічних засобів системи з обумовленими технічними показниками, в тому числі види і періодичність обслуговування технічних засобів або допустимість роботи без обслуговування;
- вимог до кількості і кваліфікації обслуговуючого персоналу та режимів його роботи і регламенту обслуговування;
- вимог до складу, розміщення та умов зберігання комплекту запасних виробів та приладів.

***У вимогах до захисту інформації від несанкціонованого доступу містяться***

вимоги, які записані в нормативних документах, що діють в галузі (відомстві) замовника.

***У вимогах до збереження інформації в разі аварії наводять***

перелік подій, які кваліфікуються як аварія, в разі яких повинна бути збережена інформація.

***Вимоги до захисту від впливу зовнішніх чинників складаються***

з вимог до радіоелектронного захисту засобів АС та вимог щодо стійкості, сталості та міцності відносно негативних зовнішніх чинників.

***У вимогах до функцій (задач) наводять:***

- по кожній підсистемі перелік функцій, задач або їх компонентів;
- часовий регламент реалізації кожної задачі або функції;
- вимоги до якості реалізації кожної функції (або задачі), до форми подання вихідної інформації, характеристики необхідної точності і часу виконання, вимоги одночасного виконання групи функцій, достовірності видачі результатів;
- перелік і критерії відмов для кожної функції, для якої задаються вимоги до надійності.

***У вимогах до видів забезпечення в залежності від виду системи наводять вимоги до математичного, інформаційного, лінгвістичного, програмного, технічного, метрологічного, організаційного та інших видів забезпечення системи.***

Для *математичного забезпечення* наводять вимоги до складу, галузі і способів використання у системі математичних методів і моделей, типових алгоритмів і алгоритмів, що необхідно розробити.

Для *інформаційного забезпечення* наводять вимоги:

- до складу, структури і засобів організації даних в системі;
- до інформаційного обміну між системами;
- до інформаційної сумісності із суміжними системами;
- до використання систем управління баз даних;
- до збору, обробки, передачі даних в системі і подання даних;
- до захисту даних від пошкоджень в разі аварії або збою в електроживленні системи;
- до контролю, зберігання, оновлення та відновлення даних.

Для *лінгвістичного забезпечення* наводять вимоги до використання в системі мов програмування високого рівня, мов взаємодії користувачів і технічних засобів системи, а також вимоги до кодування і декодування даних, до мов введення-виводу даних, до способів організації діалогу.



Для *програмного забезпечення* системи наводять перелік необхідних програмних засобів, а також вимоги:

- до незалежності програмних засобів від засобів обчислювальної техніки та операційного середовища;
- до якості програмних засобів, а також до способів його забезпечення і контролю.

Для *технічного забезпечення* системи наводять вимоги:

- до видів технічних засобів, в тому числі видів комплектуючих технічних засобів, програмно-технічних комплексів та інших комплектуючих виробів, які можуть бути використані в системі;
- до функціональних, конструктивних і експлуатаційних характеристик засобів технічного забезпечення системи.

У вимогах до *метрологічного забезпечення* вказують вимоги до метрологічного забезпечення технічних і програмних засобів, які входять до складу вимірювальних каналів системи, і засобів вимірювань, що використовуються під час налагодження і випробувань системи.

Для *організаційного забезпечення* наводять вимоги:

- до структури і функцій підрозділів, що беруть участь в експлуатації системи або забезпечують її експлуатацію;
- до організації функціонування системи;
- до захисту від помилкових дій персоналу системи.
- 

### **3.3. Склад і зміст робіт зі створення системи**

Розділ технічного завдання склад і зміст робіт зі створення інформаційної системи містить:

- перелік стадій та етапів робіт зі створення системи і терміни їх виконання;
- перелік документів, що подаються після виконання кожного етапу;
- вид і порядок проведення експертизи технічної документації;
- програму робіт із забезпечення надійності системи.

**Порядок контролю та приймання системи складається з:**

- опису видів, складу, об'єму та методів випробувань системи та її складових частин;
- загальних вимог до прийому робіт по стадіях.

**Вимоги до складу та змісту робіт з підготовки об'єкта автоматизації до введення системи в дію** містять перелік основних заходів, які слід виконати під час підготовки об'єкта автоматизації до введення АС в дію, а саме:

- доведення інформації, що надходить в систему до виду, придатного до введення в ПК;

- створення умов функціонування об'єкту автоматизації, за яких гарантується відповідність системи, що створюється, вимогам ТЗ;
- терміни і порядок комплектації штату і навчання персоналу.

**Вимоги до документування** містять узгоджений із замовником перелік документів, які належить розробити, а також вимоги до складу і змісту цих документів.

**Джерела розробки** складаються з переліку документів та інформаційних матеріалів (нормативних документів), на підставі яких було розроблено ТЗ і які повинні використовуватися при створенні системи.

**В додатках до ТЗ наводять:**

- розрахунок очікуваної ефективності системи;
- оцінку науково-технічного рівня системи.

### **Контрольні питання**

1. Що таке технічне завдання?
2. Наведіть приклади вимог до надійності системи.
3. Перелічіть основні джерела розробки технічного завдання.
4. З яких розділів складається технічне завдання на розробку системи?
5. Згідно до яких стандартів, виконується технічне завдання на розробку автоматизованої системи?
6. Що міститься у розділі «програмне забезпечення» технічного завдання?
7. Що характеризують значення параметрів, які наводять у вимогах до показників призначення?
8. Що входить в розділ «Склад і зміст робіт зі створення системи» технічного завдання?
9. Які вимоги ставлять до математичного забезпечення?
- 10.3 чого складається порядок контролю та приймання системи?

### **Рекомендована література**

1. Андон Ф.И. Основы инженерии качества программных систем [Текст] / Ф.И. Андон и др. – Издательский Дом «Академперіодика», 2007. – 673 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст] / А.М. Вендров - М.: Финансы и статистика, 2005.- 544 с.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем [Текст] / А.М. Вендров -, М.: Финансы и статистика, 2003, - 352 с.

4. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
5. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.
6. Плескач В.Л. Інформаційні технології та системи [Текст] / В.Л.Плескач, Ю.В.Рогущина, Н.П.Кустова – К.: «КНИГА», 2004.- 520 с

## ЛАБОРАТОРНА РОБОТА №№ 5,6

### Структурний підхід до програмування. Стадія «Ескізний проект»

#### Мета роботи:

Ознайомитися з методами моделювання і аналізу вимог, правилами розробки, підготовки та оформлення технічного завдання.

#### Завдання

1. На основі технічного завдання з лабораторної роботи № 3,4 виконати аналіз функціональних і експлуатаційних вимог до програмного продукту.
2. Визначити основні технічні рішення (вибір мови програмування, структура програмного продукту, склад функцій ПП, режими функціонування)
3. Визначити діаграми потоків даних для вирішуваного завдання.
4. Визначити діаграми «сутність-зв'язок», якщо програмний продукт містить базу даних.
5. Визначити функціональні діаграми.
6. Визначити специфікації процесів (задач).
7. Додати словник термінів.
8. Оформити результати, використовуючи MS Office або MS Visio, у вигляді ескізного проекту.
9. Здати і захистити роботу.

#### Підготовка до лабораторної роботи

Проробити відповідні розділи джерел [1- 6] щодо тем «Етапи розробки програмного забезпечення. Аналіз вимог і визначення специфікацій програмного забезпечення», «Інженерія вимог до програмного забезпечення».

Ознайомитися з матеріалами розд. 3.5 Учбового посібника [4]

#### Теоретичні відомості

Розділ "Ескізний проект" складається з 3-х підрозділів:

- документи ескізного проекту;
- документи робочої документації;
- пояснювальна записка до ескізного проекту.

**Документи ескізного проекту** складаються з переліків вхідних сигналів та даних і переліку вихідних сигналів та даних.

В переліку вхідних сигналів та даних вказують:

- для аналогового сигналу – назву вимірюваної величини, одиниці виміру, діапазон змін, вимоги точності і періодичності змін, тип сигналу;

- для дискретного сигналу – назву, розрядність і періодичність, тип сигналу;

- для сигналу типу «так-ні» – джерело формування та змістовне значення сигналу;

- назву, кодове позначення і значення реквізитів вхідних даних;

- назву і кодове позначення документів, які містять ці повідомлення.

Крім того, слід обов'язково вказати, чи будуть вводитися вхідні дані з документів, чи треба буде заповнювати деякі проміжні форми, і, в такому разі, надати вид цих форм. Якщо вхідні дані вибираються з інших документів, то треба обов'язково вказати, які саме дані.

Перелік вихідних сигналів і документів містить:

- перелік вихідних сигналів з позначенням їх назв, призначення, одиниць виміру і діапазонів значень, способів подання, користувачів інформації;

- перелік вихідних документів з позначенням їх назв, кодових позначень і значення реквізитів, користувачів інформації.

***До документів робочої документації належать:***

- загальний опис інформаційної системи;

- інструкція користувача системою;

- інструкція з формування і ведення баз даних;

- інструкція з експлуатації комплексу технічних засобів;

- опис технологічного процесу обробки даних.

***Пояснювальна записка до ескізного проекту складається з таких документів:***

- опис функцій, які автоматизуються, і алгоритму функціонування у вигляді функціональної моделі (в будь-якій нотації);

- опис комплексу задач;

- опис інформаційного забезпечення АСУ;

- опис програмного забезпечення;

- опис математичного забезпечення.

***а. Опис інформаційного забезпечення складається з:***

- складу інформаційного забезпечення із зазначенням всіх баз даних і наборів даних;

- організації інформаційного забезпечення;

- організації збору і передачі інформації;

- побудови системи класифікації і кодування;

- організації внутрішньої інформаційної бази;

- організації зовнішньої інформаційної бази.

***б. Опис програмного забезпечення складається з:***

- структури програмного забезпечення;

- функцій частин програмного забезпечення;
- методів і засобів розробки програмного забезпечення;
- операційної системи;
- засобів, що розширюють можливості операційної системи.

**с.** *Опис математичного забезпечення системи наводиться в документі «Опис алгоритму (проектної процедури)», який складається з таких розділів:*

- призначення і характеристика;
- використана інформація;
- результаті рішення;
- математичний опис;
- алгоритм рішення.

Всі документи виконуються у відповідності з РД 50-34.698-90.  
Додаткові відомості.

### **Зміст звіту**

Постановка завдання (мета лабораторної роботи).

Документ «Ескізний проект», що містить:

- основні технічні рішення;
- специфікації процесів;
- всі отримані діаграми;
- словник термінів.

### **Висновки**

Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навиків і відповідях на питання.

### **Контрольні питання**

1. Назначение спецификаций программного продукта.
2. Назначение и суть функциональных диаграмм при анализе требований к программному продукту.
3. Методология и модели SADT. Приведите пример.
4. Назначение и суть диаграмм потоков данных (DFD).
5. Что такое «контекстная диаграмма» DFD. Приведите пример.
6. Назначение и суть диаграмм «сущность – связь».
7. Модели ERM. Приведите пример.
8. Назначение и суть диаграмм переходов состояний.
9. Назначение словаря терминов.

### **Рекомендована література**

1. Андон Ф.И. Основы инженерии качества программных систем [Текст] / Ф.И. Андон и др. – Издательский Дом «Академперіодика», 2007. – 673 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст] / А.М. Вендров - М.: Финансы и статистика, 2005.- 544 с.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем [Текст] / А.М. Вендров -, М.: Финансы и статистика, 2003, - 352 с.
4. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
5. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.
6. Плєскач В.Л. Інформаційні технології та системи [Текст] / В.Л.Плєскач, Ю.В.Рогущина, Н.П.Кустова – К.: «КНИГА», 2004.- 520 с

**ЛАБОРАТОРНА РОБОТА №№ 7, 8**  
**Структурний підхід до програмування.**  
**Стадія «Технічний проект»**

**Мета:** Ознайомитись із змістом, методами и правилами розробки, підготовки та оформлення технічного проекту.

**Підготовка до лабораторної роботи:** Проробити відповідні розділи джерел [9 - 14] щодо тем «Розробка програмного забезпечення. Структурний підхід ». Ознайомитись з матеріалами лекції.

**Завдання:**

1. На Основі ескізного проекту з лабораторної роботи 5,6 Розробити уточнені рішення щодо функцій (завдань) розроблюваного програмного продукту, структур даних, технічних засобів, алгоритмів вирішення завдань (див. Примітка в теоретичній частині). При розробці алгоритмів рекомендується використати метод покрокової деталізації.
2. Розробити структурну схему програмного продукту і функціональну схему
3. Представити структурну схему у вигляді структурних карт Константайна і структурних карт Джексона
4. Оформити результати, використовуючи MS Office або MS Visio.
5. Здати і захистити роботу.

**Зміст звіту**

1. Постановка Завдання (мета лабораторної роботи).
2. Пояснювальна записка з викладенням сформульованих джерел (див. п.1 Завдання).
3. Структурна і функціональна схеми програмного продукту.
4. Структурні карти Константайна, Джексона (за рішенням Виконавця)
5. Висновки.
6. Захист Звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманий результатів, Демонстрації отриманий навиків и відповідях на питання.

**Теоретичні відомості**

**Технічний проект (загальні відомості)**

Мета технічного проекту – деталізація рішень Ескізного проекту, що використовуються при створенні інформаційної системи, і остаточне визначення її кошторисної вартості.



Технічне проектування підсистем здійснюється відповідно до затвердженого технічного завдання. Склад документації Технічного проекту автоматизованих систем регламентує документ РД 50-34.698-90.

Технічний проект програмної системи повинен, в основному, описувати:

- функції що виконуються (розв'язувані завдання);
- уточнені структури (форми) вхідних та вихідних даних;
- структуру оброблюваних баз даних;
- структуру розроблюваного програмного забезпечення (програмного засобу);
- алгоритми вирішення завдань;
- склад програмного забезпечення, що використовується (ОС, СУБД та ін.);
- конфігурацію та характеристики технічних засобів що використовуються.

При розробці технічного проекту автоматизованої системи оформлюються:

- відомість технічного проекту. Загальна інформація по проекту;
- пояснювальна записка до технічного проекту. Вступна інформація, що дозволяє користувачеві швидко освоїти дані по конкретному проекту;
- опис систем класифікації та кодування;
- перелік вхідних даних (документів). Перелік інформації, яка використовується як вхідний потік і служить джерелом накопичення даних;
- перелік вихідних даних (документів). Перелік інформації, яка використовується для аналізу накопичених даних;
- опис програмного забезпечення що використовується. Перелік програмного забезпечення і СУБД, які плануються використовувати для створення інформаційної системи;
- опис технічних засобів що використовується. Перелік апаратних засобів, на яких планується робота проектного програмного продукту;
- проектна оцінка надійності системи. Експертна оцінка надійності з виявленням найбільш надійних ділянок програмної системи та її вузьких місць;
- відомість устаткування і матеріалів. Перелік обладнання та матеріалів, які потрібні в ході реалізації проекту.

### **Структурна схема**

Структурною називають схему, що відображає склад і взаємодію частин розроблюваного програмного забезпечення. Структурна схема визначається архітектурою розроблюваного ПЗ.

## Функціональна схема

Функціональна схема – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках та зазначенням використовуваних файлів і пристроїв.

### ***Розробка алгоритмів***

Метод покрокової деталізації реалізує спадний підхід до програмування і передбачає покрокову розробку алгоритму

### ***Структурні карти***

Методика структурних карт використовується на етапі проектування ПЗ для того, щоб продемонструвати, яким образом програмний продукт виконує системні вимоги. Структурні карти Константайна призначені для опису відносин між модулями

Техніка структурних карт Джексона заснована на методі структурного програмування Джексона, який виявляє відповідність між структурою потоків даних і структурою програми. Основна увага в методі сконцентровано на відповідності вхідних і вихідних потоків даних.

## Проектування програмного забезпечення

### ***Проектування програмного забезпечення при структурному підході***

При проектуванні складного програмного забезпечення перш за все необхідно визначити структурні компоненти і зв'язки між ними. Отримана в результаті структура ПЗ повинна бути представлена вигляді структурної або функціональної схем і специфікацій її компонентів.

### ***Структурна схема розроблюваного програмного забезпечення***

Структурної називають схему, що відображає склад і взаємодію з управління частин розроблюваного програмного забезпечення.

Структурна схема визначається архітектурою розроблюваного ПЗ.

Розробку структурної схеми програми зазвичай виконують методом покрокової деталізації.

Структурні схеми *пакетів програм* розробляють для кожної програми пакета окремо, оскільки організація програм в пакети не передбачає передачі управління між ними.

Компонентами структурної схеми програмної системи або програмного комплексу можуть служити програми, підсистеми, бази даних, бібліотеки ресурсів і т. п.

Як правило, для програмних систем розробляється функціональна схема, яка дає більш повне уявлення про проєктований програмному забезпеченні з точки зору взаємодії його компонентів між собою і з зовнішнім середовищем. На рисунку 7.1. представлено структурну схему програми для обчислення розрахунку оплати за електроенергію. Метод покрокової деталізації передбачає що деякі функції повинні бути деталізовані для

виконання певних алгоритмічних дій. Тому необхідно виконати деталізацію пункту, що виділено сірим кольором.

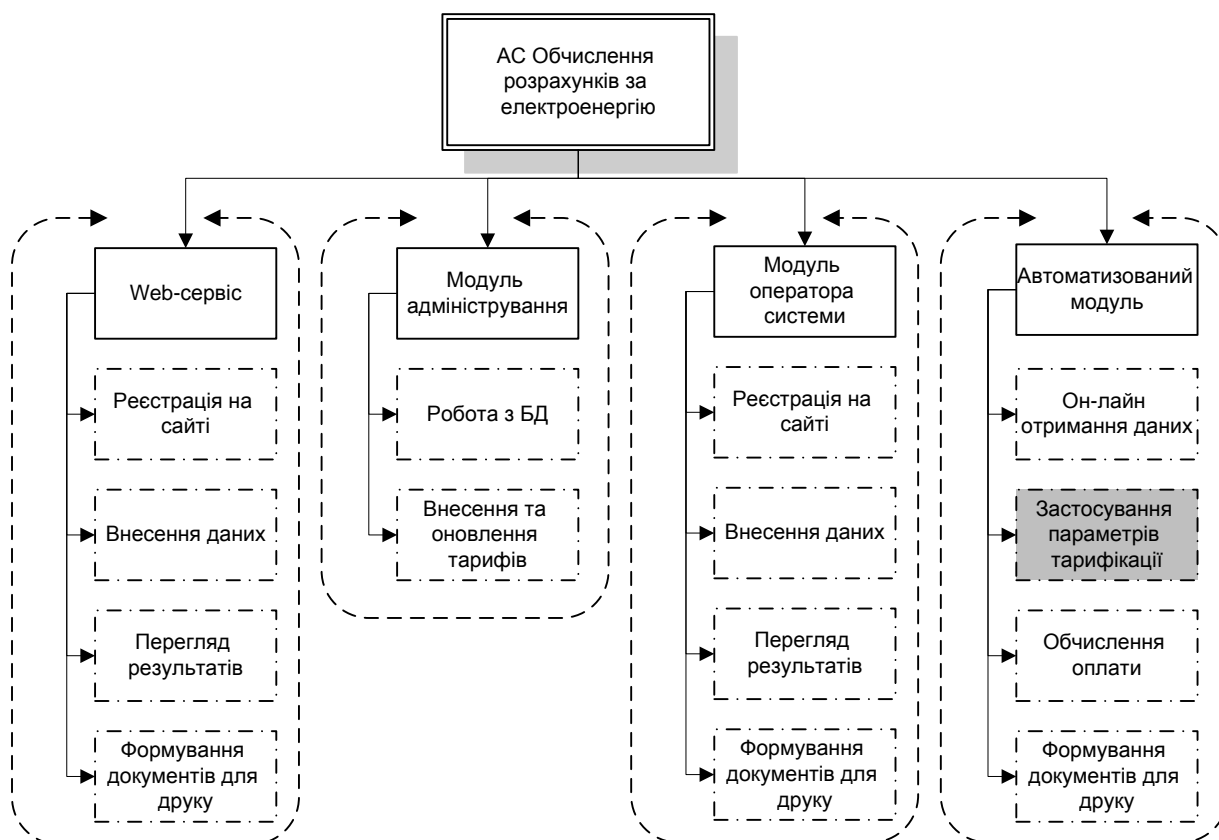


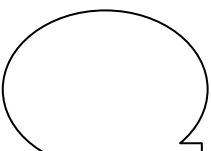
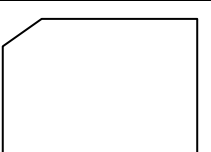
Рис. 7.1. Структурна схема програми для обчислення плати за електроенергію.

### Функціональна схема

Функціональна схема (ГОСТ 19.701-90 ) – це схема взаємодії компонентів програмного забезпечення з описом інформаційних потоків, складу даних в потоках та зазначенням використовуваних файлів і пристроїв [1]. Для зображення функціональних схем використовують спеціальні позначення, встановлені стандартом (табл. 7.1) .

Таблиця 7.1 . Позначення елементів функціональних схем

Назва блоку	Позначення	Призначення блоку
Дані		Символ відображає дані, носій даних не визначений

Запам'ятовування дани		Символ відображає збережені дані у вигляді, придатному для обробки, носій даних не визначений
Оперативний пристрій		Символ відображає дані, що зберігаються в оперативному запам'ятовуючому пристрої
Запам'ятовуючий пристрій з послідовним доступом		Символ відображає дані, що зберігаються в пристрої з послідовним доступом (магнітна стрічка, касета з магнітною стрічкою, магнітофонна касета).
Запам'ятовуючий пристрій з прямим доступом		Символ відображає дані, що зберігаються в пристрої з прямим доступом (магнітний диск, магнітний барабан, гнучкий магнітний диск).
Документ		Символ відображає дані, представлені на носії у зручній для зчитування формі (машинограма, документ для оптичного або магнітного зчитування, мікрофільм, рулон стрічки з підсумковими даними, бланки введення даних )
Ручне введення		Символ відображає дані, що вводяться вручну під час обробки з пристроїв будь-якого типу (клавіатура, перемикачі, кнопки, світлове перо, смужки з штриховим кодом)
Карта		Символ відображає дані, представлені на носії у вигляді карти (перфокарти, магнітні карти, карти зі зчитуваними мітками, карти з відривним ярликом, карти зі сканованими мітками).
Дисплей		Символ відображає дані, представлені в людино-читаючій формі на носії у вигляді відображуючого пристрою (екран для візуального спостереження, індикатори введення інформації).
Процес		Символ відображає функцію обробки даних будь-якого виду (виконання певної операції або групи операцій, що приводить до зміни


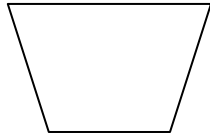
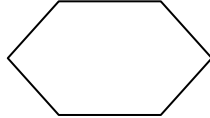
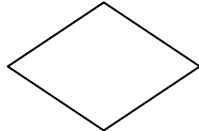
		значення , форми або розміщення інформації або до визначення , за яким з декількох напрямків потоку слід рухатися ).
Умовний процес		Символ відображає умовний процес, що складається з однієї або декількох операцій або кроків програми, які визначені в іншому місці (в підпрограмі, модулі)
Ручна операція		Символ відображає будь-який процес, що виконується людиною.
Підготовка		Символ відображає модифікацію команди або групи команд з метою впливу на деяку подальшу функцію (встановлення перемикача, модифікація індексного регістра або ініціалізація програми)
рішення		Символ відображає рішення або функцію перемикача типу, що має один вхід і ряд альтернативних виходів, один і тільки один з яких може бути активізований після обчислення умов, визначених всередині цього символу. Відповідні результати обчислення можуть бути записані по сусідству з лініями , що відображають ці шляхи



Рис. 7.2 . Приклад функціональної схеми програмного комплексу

Функціональні схеми більш інформативні, ніж структурні. На рис. 7.2 наведена функціональна схема програмного комплексу, що реалізує різні методи сортування масивів.

### **Метод покрокової деталізації при складанні алгоритмів**

Метод покрокової деталізації реалізує спадний підхід до програмування і передбачає покрокову розробку алгоритму. Можна виділити наступні етапи:

1. Створюється опис програми в цілому. Визначаються основні логічні кроки, необхідні для вирішення завдання, навіть якщо поки невідомо, як їх виконати. Ці логічні кроки можуть відображати різні фізичні способи рішення або можуть бути зручними груповими іменами для тих дій, виконання яких представляється досить нечітко. Послідовності кроків, необхідних для вирішення задачі, записуються на звичайній мові або на псевдокоді.
2. У загальних термінах деталізується опис кроків, введених на етапі 1. У деталізований опис може входити позначення циклічних структур, у той час як дії всередині циклів можуть як і раніше залишатися неясними. Таким чином, виконуються тільки загальні ескізи складних дій.

3. На цьому і наступних рівнях у вигляді послідовних ітерацій виконуються ті ж дії, що описані на етапі 2. При кожній новій ітерації уточнюються деталі, що залишилися неясними після попередніх ітерацій, і створюються більш точні описи. По мірі виконання ітерацій невизначені деталі стають все простіше і простіше, так що на якомусь етапі можуть бути повністю описані.
4. Розробка завершена: у модульному вигляді отримано опис необхідної програми. Переклад цього опису в програму на конкретній мові програмування повинна бути досить простим завданням.

### **Структурні карти Константайна**

Методика структурних карт використовується на етапі проектування ПЗ для того, щоб продемонструвати, яким чином програмний продукт виконує системні вимоги.

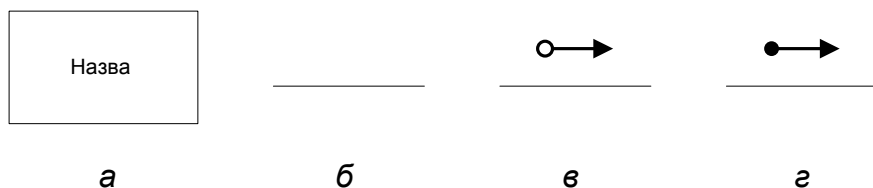
При цьому найбільш часто застосовуються дві техніки: структурні карти Константайна (Constantine), призначені для опису відносин між модулями, і структурні карти Джексона (Jackson), призначені для опису внутрішньої структури модулів.

Структуру програмної системи складають модулі, які в будь-якій мові програмування мають такі загальні характеристики:

- модуль має ім'я, за яким до нього можна звертатися як до єдиного фрагменту;
- модуль складається з безлічі операторів мови програмування, записаних послідовно;
- модуль може приймати і/або передавати дані як параметри в послідовності, що викликається, або пов'язувати дані через фіксовані клітинки, або загальні області.

Структурні карти Константайна являють собою модель відносин між модулями програми. Вузли структурних карт відповідають модулям і областям даних, потоки зображують міжмодульні зв'язки. На діаграмі спеціальними вузлами зображуються циклічні та умовні виклики модулів, а потоки проходять через ці спеціальні вузли. Потоки, що зображують міжмодульні зв'язки за даними та управління, також зображуються на діаграмі спеціальними вузлами, а стрілками вказуються напрямки потоків.

На рис. 7.3 наведені основні компоненти структурних карт Константайна.



*Рис . 7.3 . Елементи структурних карт:*

а - модуль, б - виклик модуля; в - зв'язок за даними; г - зв'язок з управлінням

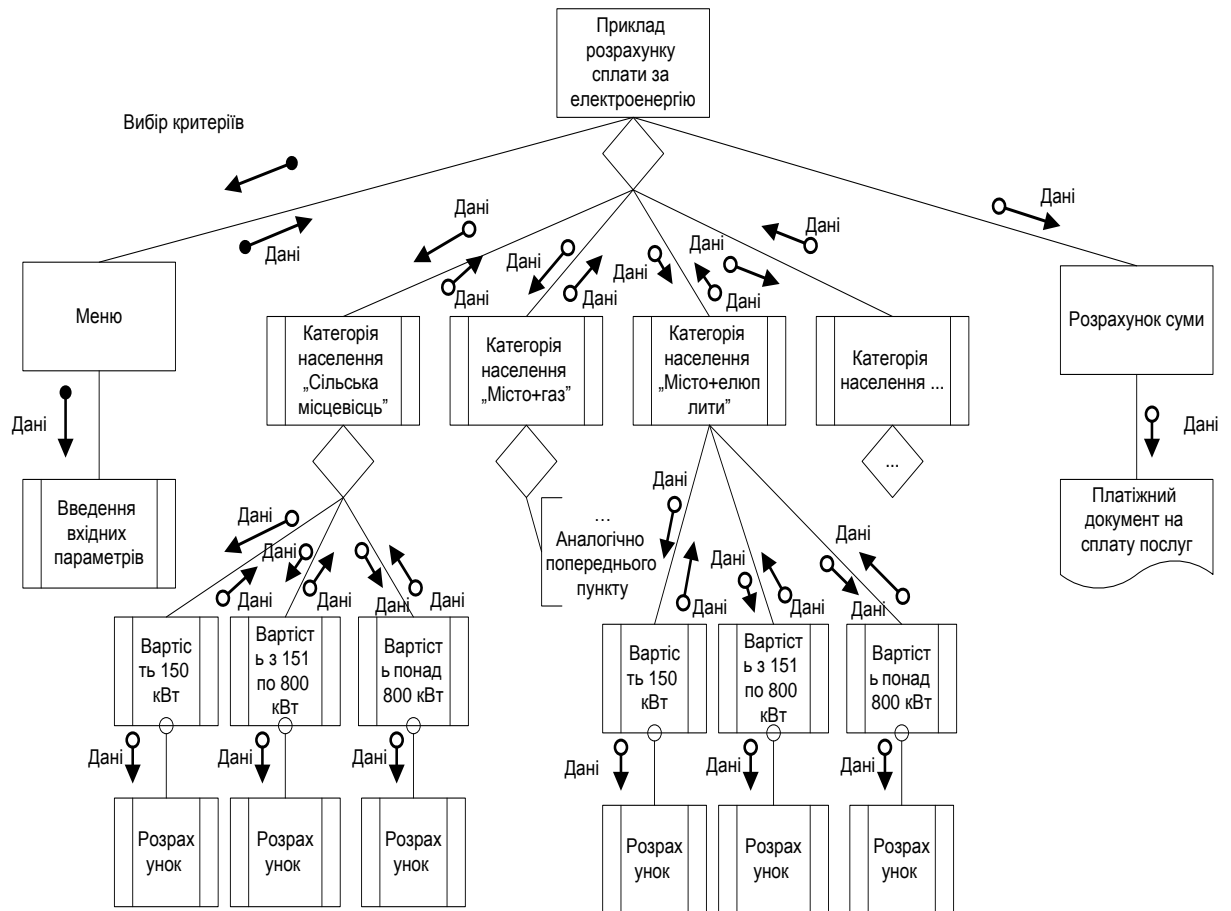


Рис . 7.4 . Приклад структурної карти Константайна для опису процесу обчислення плати за електроенергію

### Структурні карти Джексона

Техніка структурних карт Джексона заснована на методі структурного програмування Джексона , який виявляє відповідність між структурою потоків даних і структурою програми. Основна увага в методі сконцентровано на відповідності вхідних і вихідних потоків даних.

Структури на діаграмах Джексона будуються з чотирьох основних компонентів:

- операція – блок-кодів , що має один вхід і один вихід;
- слідування – послідовне виконання операцій зліва направо;
- вибір – виконання однієї з операцій залежно від виконання умови;
- ітерація – багаторазове виконання блоку.





Рис . 7. . Структурна карта Джексона

### Контрольні питання

1. Який документ регламентує склад технічного проекту автоматизованої системи?
2. Що повинен описувати технічний проект програмної системи?
3. Призначення і суть структурної схеми ПЗ. Приведіть приклад.
4. Призначення і суть функціональної схеми ПЗ. Приведіть приклад.
5. Призначення і суть структурних карт Константайна. Приведіть приклад.
6. Призначення і суть структурних карт Джексона.

### Рекомендована література

1. ДСТУ 2226-93 Автоматизовані системи. Терміни та визначення.
2. ДСТУ 3330-96 Інформаційні технології. Система стандартів з баз даних. Еталонна модель керування даними.
3. ДСТУ 3626-97 Базові програмно-технічні комплекси локального рівня для розосереджених автоматизованих систем керування технологічними процесами. Загальні вимоги.
4. ДСТУ ISO 11442-2:2004 Документація на технічну продукцію. Автоматизоване оброблення технічної інформації. Частина 2. Документація оригіналів.
5. ДСТУ ISO 11442-3:2004 Документація на технічну продукцію. Автоматизоване оброблення технічної інформації. Частина 3. Стадії процесу проектування продукції.
6. ДСТУ ISO 11442-4:2004 Документація на технічну продукцію. Автоматизоване оброблення технічної інформації. Частина 4. Системи керування та пошуку документів
7. РД 50-680-88 Керівний документ по стандартизації. Методичні вказівки. Автоматизовані системи. Основні положення.
8. РД 50-34.698-90 Автоматизовані системи. Вимоги до змісту документів
9. Андон Ф.И. Основы инженерии качества программных систем [Текст] / Ф.И. Андон и др. – Издательский Дом «Академперіодика», 2007. – 673 с.

10. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст] / А.М. Вендров - М.: Финансы и статистика, 2005.- 544 с.
11. Вендров А.М. Проектирование программного обеспечения экономических информационных систем [Текст] / А.М. Вендров -, М.: Финансы и статистика, 2003, - 352 с.
12. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
13. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.
14. Плескач В.Л. Інформаційні технології та системи [Текст] / В.Л.Плескач, Ю.В.Рогущина, Н.П.Кустова – К.: «КНИГА», 2004.- 520 с

**ЛАБОРАТОРНА РОБОТА №№ 9, 10**  
**Структурний підхід до програмування.**  
**Стадія «Реалізація. Розробка програмного**  
**засобу»**

**Мета:** Розробити діючу версію програми та оформити текст програми у відповідності із рекомендаціями щодо стилю програмування та оформлення програм.

**Підготовка до лабораторної роботи:** Проробити відповідні розділи джерел [1, 2] щодо тем «Розробка програмного забезпечення. Технологічність програмних продуктів. Стель оформлення програми.»

**Завдання:**

1. На основі технічного проекту з лабораторної роботи №№ 7,8 розробити діючу версію програми.
2. Оформити текст програми у відповідності з рекомендаціями щодо стилю оформлення.
3. Представити до захисту програмний файл.
4. Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації виконаного завдання та отриманих навиків і відповідях на питання

**Зміст звіту**

1. Постановка завдання (мета лабораторної роботи), структурна і функціональна схеми програмного продукту – з Технічного проекту.
2. Оформлений текст програми.
3. Скріншоти виконання програми.
4. Виконання програми на ПК.
5. Висновки.

**Теоретичні відомості**

**Поняття технологічності програмного забезпечення**

Під технологічністю розуміють якість проекту програмного продукту, від якого залежать трудові та матеріальні витрати на його реалізацію і наступні модифікації.

Хороший проект порівняно швидко і легко кодується, тестується, налагоджували і модифікується.

З досвіду кількох поколінь розробників програмного забезпечення відомо, що технологічність програмного забезпечення визначається пропрацьованістю його моделей, рівнем незалежності модулів, стилем програмування і ступенем повторного використання кодів.

Чим краще опрацьована модель розроблюваного програмного забезпечення, тим чіткіше визначені підзадачі і структури даних, що зберігають вхідні, проміжну і вихідну інформацію, тим простіше їх проектування і реалізація і менше ймовірність помилок, для виправлення яких знадобиться істотно змінювати програму.

Чим вище незалежність модулів, тим їх легше зрозуміти, реалізовувати, модифікувати, а також знаходити в них помилки і виправляти їх.

Стиль програмування, під яким розуміють стиль оформлення програм і їх «структурність», також істотно впливає на читаність програмного коду і кількість помилок програмування. Криза 60-х років XX в. був викликаний у тому числі і стилем програмування, при якому програма нагадувала клубок поплутаних ниток або блюдо спагетті, і відсутністю мовних конструкцій підтримки «структурного» стилю.

Збільшення ступеня повторного використання кодів передбачає як використання раніше розроблених бібліотек підпрограм або класів, так і уніфікацію кодів ті Кучок розробки. Причому для даного критерію ситуація не така однозначна, як у попередніх випадках: якщо ступінь повторного використання кодів підвищується штучно (наприклад, шляхом розробки «суперуніверсальних» процедур), то технологічність проекту може істотно знизитися.

### Стиль оформлення програми

З точки зору технологічності хорошим вважають стиль оформлення програми, який полегшує її сприйняття як самим автором, так і іншими програмістами, яким, можливо, доведеться її перевіряти або модифікувати. «Пам'ятайте, програми читаються людьми», закликав Д. Ван Тассел, автор однієї з відомих монографій, присвяченій проблемам програмування [1].

Саме виходячи з того, що будь-яку програму неодноразово доведеться переглядати, слід дотримуватися гарного стилю написання програм.

Стиль оформлення програми включає:

- правила іменування об'єктів програми (змінних, функцій, типів, даних тощо);
- правила оформлення модулів;
- стиль оформлення текстів модулів.

Правила іменування об'єктів програми. При виборі імен програмних об'єктів слід дотримуватися наступних правил:

- ім'я об'єкта має відповідати його змісту, наприклад:

*MaxItem - максимальный элемент;*

*NextItem - следующий элемент;*

- якщо дозволяє мова програмування, можна використовувати символ «\_» для візуального розділення імен, що складаються з декількох слів, наприклад:

*Max\_Item, Next\_Item;*

- необхідно уникати близьких за написанням імен, наприклад:  
*Index* и *InDec*.

### **Правила оформлення модулів.**

Кожен модуль має передувати заголовком, який, як мінімум, містить:

- назва модуля;
- короткий опис його призначення;
- короткий опис вхідних і вихідних параметрів із зазначенням одиниць виміру;
- список використовуваних (що викликаються) модулів;
- короткий опис алгоритму (методу) та / або обмежень;
- ПІБ автора програми;
- ідентифікаційну інформацію (номер версії і / або дату останньої коригування).

### **Стиль оформлення текстів модулів.**

Стиль оформлення текстів модулів визначає використання відступів, пропусків рядків і коментарів, що полегшують розуміння програми.

Як правило, пропуски рядків і коментарі використовують для візуального розділення частин модуля, наприклад:

```
{перевірка кількості відрізків і вихід, якщо відрізки не задані}
If n<0 then
begin
  Writeln (' Кількість відрізків не задано ');
exit;
end;
{цикл суми довжин відрізків}
S:= 0;
for i:= 0 to n-1 do S:= S + Len [i];
```

Для таких мов, як Pascal, C++ і Java, використання відступів дозволяє прояснити структуру програми: зазвичай додатковий відступ позначає вкладення операторів мови, наприклад:

```
amax:= a[1,1];
for i:= 1 to n do
  for j:= 1 to m do
    If a[i,j]>amax then amax:= a [i,j];
```

Дещо складніше справа йде з коментарями. Досвід показує, що перекладати з англійської мови кожен оператор програми не потрібно: будь-який програміст, що знає мову програмування, на якому написана програма, без праці прочитає той чи інший оператор.

Коментувати слід мети виконання тих чи інших дій, а також групи операторів, пов'язані спільним дією, тобто коментарі повинні містити деяку додаткову (неочевидну) інформацію, наприклад:

```
{перевірка умови і вихід, якщо умова не виконується}  
If n<0 then  
begin  
    Writeln (' Кількість відрізків не задано ');  
    exit;  
end;
```

Для мов низького рівня, наприклад, Асемблера, стиль, який полегшує розуміння, запропонувати важче. У цьому випадку може виявитися доцільним коментувати і блоки операторів, і кожен оператор, наприклад:

```
; цикл суми елементів масиву  
;установки циклу  
mov    AX, 0      ; обнуляємо суму  
mov    CX, n      ; завантажуюмо лічильник циклу  
mov    BX, 0      ; зміщення першого елементу масиву  
;тіло циклу  
cycle: add    AX, a [BX]    ; додаємо елемент  
        add    BX, 2      ; визначаємо адресу наступного  
        loop   cycle      ; цикл на n повторень  
;          вихід з циклу при обнуленні лічильника
```

### **Контрольні питання по тексту програми**

#### **Рекомендована література**

1. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
2. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана - 2002. – 243 с.

**ЛАБОРАТОРНА РОБОТА №№ 11, 12**  
**Структурний підхід до програмування.**  
**Стадія «Налагодження програмного засобу та**  
**розробка програмної документації»**

**Мета:** Ознайомитися із вимогами до складу і змісту документації, необхідної для супроводження та експлуатації програмного засобу.

**Підготовка до лабораторної роботи:** Проробити відповідні розділи джерел [2, 3] щодо тем «Налагодження програмного забезпечення. Розробка і оформлення програмної документації».

**Завдання:**

1. Налагодження програмного засобу, розробленого в лабораторній роботі №№9,10.
2. Розробка і оформлення програмної документації.
3. Представити до захисту програмний файл.
4. Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації виконаного завдання та отриманих навиків і відповідях на питання

**Зміст звіту**

1. Оформлена документація на налагоджений програмний засіб у складі:
  - пояснювальна записка;
  - керівництво системного програміста;
  - керівництво користувача
2. Оформлений текст програми.
3. Скріншоти виконання програми.
4. Виконання програми на ПК.
5. Висновки.

**Теоретичні відомості**

**Комплексне налагодження програмного засобу.**

При комплексному налагодженні тестується ПЗ в цілому, причому тести готуються щодо кожного з документів ПЗ. Тестування цих документів здійснюється, як правило, в порядку, зворотному їх розробці. Виняток становить лише тестування документації по застосуванню, яка розробляється за зовнішнім описом паралельно з розробкою текстів програм – це тестування краще проводити після завершення тестування зовнішнього опису. Тестування при комплексному налагодженні являє собою застосування ПЗ до конкретних даних, які в принципі можуть виникнути у користувача (зокрема, всі тести готуються у формі, розрахованої на користувача), але можуть бути і у вигляді моделей. Наприклад, деякі

недоступні при комплексному налагодженні пристрої введення і виведення можуть бути замінені їх програмними імітаторами.

*Тестування архітектури ПЗ.* Метою тестування є пошук невідповідності між описом архітектури та сукупністю програм ПЗ. До моменту початку тестування архітектури ПЗ повинно бути закінчено автономне налагодження кожної підсистеми. Помилки реалізації архітектури можуть бути пов'язані, насамперед, з взаємодією цих підсистем, зокрема, з реалізацією архітектурних функцій (якщо вони є). Тому необхідно перевірити всі шляхи взаємодії між підсистемами ПЗ. При цьому необхідно протестувати всі ланцюжки виконання підсистем без повторного входження останніх. Якщо задана архітектура ПЗ являє собою сукупність малих системи з виділених підсистем, то число таких ланцюжків буде цілком визначеним.

*Тестування зовнішніх функцій.* Метою тестування є пошук розбіжностей між функціональною специфікацією і сукупністю програм ПЗ. Незважаючи на те, що всі ці програми автономно вже налагоджені, зазначені розбіжності можуть бути, наприклад, через невідповідність внутрішніх специфікацій програм і їх модулів (на підставі яких проводилося автономне тестування) функціональної специфікації ПЗ. Як правило, тестування зовнішніх функцій проводиться так само, як і тестування модулів, аналогічно методу чорного ящика.

*Тестування якості ПЗ.* Метою тестування є пошук порушень вимог якості, сформульованих у специфікації якості ПЗ. Це найбільш важкий і найменш вивчений вид тестування. Ясно лише, що далеко не кожен параметр якості ПЗ може бути випробуваний тестуванням. Завершеність ПЗ перевіряється вже при тестуванні зовнішніх функцій. На даному етапі тестування цього параметру якості може бути продовжено, якщо потрібно отримати якусь вірогідну оцінку ступеня надійності ПЗ. Однак, методика такого тестування ще вимагає своєї розробки. Можуть тестуватися такі параметри якості, як точність, стійкість, захищеність, тимчасова ефективність, в якійсь мірі – ефективність по пам'яті, ефективність по пристроях, розширюваність і, частково, незалежність від пристроїв. Кожен з цих видів тестування має свою специфіку і заслуговує окремого розгляду. Легкість застосування ПЗ оцінюється при тестуванні документації по застосуванню ПЗ.

*Тестування документації по застосуванню ПЗ.* Метою тестування є пошук неузгодженості документації по застосуванню і сукупністю програм ПЗ, а також виявлення незручностей, що виникають при застосуванні ПЗ. Цей етап безпосередньо передуює підключенню користувача до завершення розробки ПЗ (тестуванню визначення вимог до ПЗ та атестації ПЗ), тому досить важливо розробникам спочатку самим скористатися ПЗ так, як це буде робити користувач. Всі тести на цьому етапі готуються виключно на



підставі тільки документації по застосуванню ПЗ. Перш за все, повинні тестуватися можливості ПЗ, як це робилося при тестуванні зовнішніх функцій, але тільки на підставі документації по застосуванню. Повинні бути протестовані всі неясні місця в документації, а також всі приклади, використані в документації. Далі тестуються найбільш важкі випадки застосування ПЗ з метою виявити порушення вимог відносності легкості застосування ПЗ.

*Тестування визначення вимог до ПЗ.* Метою тестування є з'ясування, якою мірою ПЗ не відповідає пред'явленим визначенням вимог до нього. Особливість цього виду тестування полягає в тому, що його здійснює організація-покупець або організація-користувач ПЗ, як один із шляхів подолання бар'єру між розробником і користувачем. Зазвичай це тестування проводиться за допомогою контрольних завдань – типових завдань, для яких відомий результат рішення. У тих випадках, коли розроблюване ПЗ має прийти на зміну іншій версії ПЗ, яка вирішує хоча б частину завдань ПЗ, що розробляється. Тестування проводиться шляхом вирішення загальних завдань за допомогою як старого, так і нового ПЗ (з наступним зіставленням отриманих результатів). Іноді, як форми такого тестування, використовують *дослідну експлуатацію* ПЗ – обмежене застосування нового ПЗ з аналізом використання результатів у практичній діяльності. По суті, цей вид тестування багато в чому перетинається з випробуванням ПЗ при його атестації, але виконується до атестації, а іноді і замість атестації.

### **Документування програмних засобів.**

#### **Документація, що створюється і використовується в процесі розробки програмних засобів.**

При розробці ПЗ створюється і використовується великий обсяг різноманітної документації. Вона необхідна як:

- Засіб передачі інформації між розробниками ПЗ
- Засіб управління розробкою ПЗ
- Засіб передачі користувачам інформації, необхідної для застосування і супроводу ПЗ. На створення цієї документації припадає велика частка вартості ПЗ.

Цю документацію можна розбити на дві групи:

- Документи управління розробкою ПЗ.
- Документи, що входять до складу ПЗ.

*Документи управління розробкою ПЗ (software process documentation)* керують і протоколюють процеси розробки і супроводу ПЗ, забезпечуючи зв'язки всередині колективу розробників ПЗ і між колективом розроблювачів і менеджерами ПЗ (*software managers*) – особами, які керують розробкою ПЗ. Ці документи можуть бути наступних типів:

- *Плани, оцінки, розклади.* Ці документи створюються менеджерами для прогнозування і управління процесами розробки і супроводу ПЗ.
- *Звіти* про використання ресурсів у процесі розробки. Створюються менеджерами.
- *Стандарти.* Ці документи наказують розробникам, якими принципами, правилами, угодами вони повинні слідувати в процесі розробки ПЗ. Ці стандарти можуть бути як міжнародними або національними, так і спеціально створеними для організації, в якій ведеться розробка ПЗ.
- *Робочі документи.* Це основні технічні документи, що забезпечують зв'язок між розробниками. Вони містять фіксацію ідей і проблем, що виникають у процесі розробки, опис використовуваних стратегій і підходів, а також робочі (тимчасові) версії документів, які мають увійти до ПЗ.
- *Замітки і листування.* Ці документи фіксують різні деталі взаємодії між менеджерами та розробниками.

**Документи, що входять до складу ПЗ** (*software product documentation*), описують програми ПЗ як з точки зору їх застосування користувачами, так і з точки зору їх розробників та супровідників (відповідно до призначення ПЗ). Тут слід зазначити, що ці документи будуть використовуватися не тільки на стадії експлуатації ПЗ (в її фазах застосування і супроводу), але і на стадії розробки для керування процесом розробки (разом з робочими документами) – у всякому разі, вони повинні бути перевірені (протестовані) на відповідність програмам ПЗ. Ці документи утворюють два комплекти з різним призначенням:

- Користувацька документація ПЗ (К-документація).
- Документація по супроводженню ПЗ (С-документація).

### **Користувацька документація програмних засобів.**

*Користувацька документація ПЗ* (*user documentation*) пояснює користувачам, як вони повинні діяти, щоб застосувати розроблювальне ПЗ. Вона необхідна, якщо ПЗ припускає яку-небудь взаємодію з користувачами. До такої документації належать документи, якими повинен керуватися користувач при *інсталяції* ПЗ (при установці ПЗ з відповідним налагодженням на середовище застосування ПЗ), при *застосуванні* ПЗ для вирішення своїх завдань і при *управлінні* ПЗ (наприклад, коли розробляється ПЗ буде взаємодіяти з іншими системами). Ці документи частково торкаються питань супроводу ПЗ, але не стосуються питань, пов'язаних з модифікацією програм.

При створенні користувача документації слід розрізняти дві категорії користувачів ПЗ: ординарних користувачів ПЗ та адміністраторів ПЗ.

*Ординарний користувач ПЗ* (*end - user*) використовує ПЗ для вирішення своїх завдань (в своїй предметній області). Це може бути інженер, який

проектує технічний пристрій, або касир, який продає залізничні квитки за допомогою ПЗ. Він може і не знати багатьох деталей роботи комп'ютера або принципів програмування.

*Адміністратор ПЗ (system administrator)* управляє використанням ПЗ ординарними користувачами і здійснює супровід ПЗ, не пов'язане з модифікацією програм. Наприклад, він може регулювати права доступу до ПЗ між ординарними користувачами, підтримувати зв'язок з постачальниками ПЗ або виконувати певні дії, щоб підтримувати ПЗ в робочому стані, якщо воно включено як частину в іншу систему.

Склад документації користувача залежить від аудиторій користувачів, на які орієнтовано розроблювальне ПЗ, і від режиму використання документів. Під *аудиторією* тут розуміється контингент користувачів ПЗ, у якого є необхідність у певній документації користувача ПЗ. Вдалий користувацький документ суттєво залежить від точного визначення аудиторії, для якої він призначений. Користувацька документація повинна містити інформацію, необхідну для кожної аудиторії. Під *режимом використання* документації розуміється спосіб, що визначає, яким чином використовується цей документ. Звичайно користувачеві досить великих програмних систем потрібні або документи для вивчення ПЗ (використання у вигляді *інструкції*), або для уточнення деякої інформації (використання у вигляді *довідника*).

У відповідності зі сказаним можна вважати типовим наступний склад користувацької документації для достатньо великих ПЗ:

- *Загальний функціональний опис ПЗ.* Дає коротку характеристику функціональних можливостей ПЗ. Призначено для користувачів, які повинні вирішити, наскільки необхідно їм дане ПЗ. (ГОСТ 19.401-78. Єдина система програмної документації. Текст програми, вимоги до змісту та оформлення).
- *Керівництво по інсталяції ПЗ.* Призначено для адміністраторів ПЗ. Воно повинно детально наказувати, як встановлювати системи в конкретному середовищі, зокрема, має містити опис комп'ютерно-зчитуваного носія, на якому поставляється ПЗ, файли, що представляють ПЗ, і вимоги до мінімальної конфігурації апаратури.
- *Інструкція по застосуванню ПЗ.* Призначена для ординарних користувачів. Містить необхідну інформацію щодо застосування ПЗ, організовану у формі зручній для її вивчення.
- *Довідник по застосуванню ПЗ.* Призначений для ординарних користувачів. Містить необхідну інформацію щодо застосування ПЗ, організовану у формі зручній для виборчого пошуку окремих деталей.
- *Керівництво з управління ПЗ.* Призначено для адміністраторів ПЗ. Воно повинно описувати повідомлення, що генеруються, коли ПЗ взаємодіє з іншими системами, і як повинен реагувати адміністратор на ці

повідомлення. Крім того, якщо ПЗ використовує системну апаратуру, цей документ може пояснювати, як супроводжувати цю апаратуру.

Розробка документації користувача починається відразу після створення зовнішнього опису. (ГОСТ 2.119-73. Єдина система конструкторської документації. Ескізний проект. ГОСТ 2.120-73. Єдина система конструкторської документації. Технічний проект).

Якість цієї документації може істотно визначати успіх ПЗ. Вона повинна бути досить проста і зручна для користувача (в іншому випадку це ПЗ, взагалі, не варто було створювати). Тому, хоча чорнові варіанти (начерки) користувача документів створюються основними розробниками ПЗ, до створення їх остаточних варіантів часто залучаються професійні технічні письменники. Крім того, для забезпечення якості документації користувача розроблено ряд стандартів, в яких пропонується порядок розробки цієї документації, формулюються вимоги до кожного виду користувача документів і визначаються їх структура та зміст.

### **Документація по супроводженню програмних засобів.**

*Документація по супроводженню ПЗ (system documentation)* описує ПЗ з точки зору її розробки. Ця документація необхідна, якщо ПЗ припускає вивчення того, як воно влаштоване (сконструйована), і модернізацію його програм. Як уже зазначалося, супровід – це триваюча розробка. Тому в разі потреби модернізації ПЗ до цієї роботи залучається спеціальна команда розробників-супровідників. Цій команді доведеться мати справу з такою ж документацією, яка визначала діяльність команди первісних (основних) розробників ПЗ, – з тією лише різницею, що ця документація для команди розробників-супровідників буде, як правило, чужою (вона створювалася іншою командою).

Щоб зрозуміти будову і процес розробки модернізованого ПЗ, команда розробників-супровідників повинна вивчити цю документацію, а потім внести в неї необхідні зміни, повторюючи в значній мірі технологічні процеси, за допомогою яких створювалося первинне ПЗ.

Документацію по супроводженню ПЗ можна розбити на дві групи:

- документація, яка визначає будову програм і структур даних ПЗ і технологію їх розробки;
- документацію, що допомагає вносити зміни в ПЗ.

Документація першої групи містить підсумкові документи кожного технологічного етапу розробки ПЗ. Вона включає наступні документи:

- Зовнішній опис ПЗ (Requirements document) ГОСТ 19.202-78. Єдина система програмної документації. Специфікація, вимоги до змісту та оформлення.
- Опис архітектури ПЗ (description of the system architecture), включаючи зовнішню специфікацію кожної її програми

(підсистеми). ГОСТ 19.003-80 Схеми алгоритмів і програм. Позначення умовні графічні

- Для кожної програми ПЗ – опис її модульної структури, включаючи зовнішню специфікацію кожного включеного в неї модуля.
- Для кожного модуля – його специфікація і опис його будови (design description).
- Тексти модулів вибраною мовою програмування (program source code listings) ГОСТ 19.401-78. Єдина система програмної документації. Текст програми, вимоги до змісту та оформлення.
- Документи встановлення достовірності ПЗ (validation documents), які описують, як встановлювалася достовірність кожної програми ПЗ і як інформація про встановлення достовірності пов'язувалася з вимогами до ПЗ.

Документи встановлення достовірності ПЗ включають, перш за все, документацію по тестуванню (схема тестування і опис комплекту тестів), але можуть включати і результати інших видів перевірки ПЗ, наприклад, докази властивостей програм. Для забезпечення прийнятної якості цієї документації корисно слідувати загальноприйнятим рекомендаціям і стандартам.

Документація другої групи містить

- *Керівництво по супроводженню ПЗ* (system maintenance guide), яке описує особливості реалізації ПЗ (зокрема, труднощі, які довелося долати) і як враховані можливості розвитку ПЗ в його будові (конструкції). У ньому також фіксуються, які частини ПЗ є апаратно і програмно залежними.

Загальна проблема супроводу ПЗ – забезпечити, щоб всі його операції йшли послідовно (залишалися узгодженими), коли ПЗ змінюється. Щоб цьому зарадити, зв'язки і залежності між документами і їх частинами повинні бути відображені в керівництві по супроводженню, і зафіксовані в базі даних управління конфігурацією.

### **Контрольні питання**

1. Для чого потрібна документування програмного засобу?
2. Вимоги до документації на програмний засіб
3. Як вимоги реалізовані в представленій документації?
4. Призначення і зміст Керівництва системного програміста.
5. Призначення і зміст керівництва користувача.
6. Види інформативної документації.

### **Рекомендована література**

1. ДСТУ 4302 – 2004. Інформаційні технології. Настанови щодо документування комп'ютерних програм

2. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
3. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.

**ЛАБОРАТОРНА РОБОТА №№ 13, 14**  
**Структурний підхід до програмування.**  
**Стадія «Тестування програмного засобу».**

**Мета:** Ознайомитися із методами тестування програмного засобу та практичною реалізацією тестів.

**Підготовка до лабораторної роботи:** Проробити відповідні розділи джерел [1, 2] щодо теми «Тестування програмного засобу».

**Завдання:**

1. Спроекувати тести обраними методами (за принципом «білої скриньки») для тестування функції вводу даних і заповнення БД у програмному засобі, розробленому у Лаб. Роб. 11,12 (контроль помилок користувача, коректність заповнення таблиць БД і т.ін).
2. Спроекувати тести для перевірки іншої функції (за вибором Виконавця).
3. Протестувати програмний засіб.
4. Оформити результати тестування
5. Представити до захисту програмний файл.
6. Здати і захистити роботу.

**Зміст звіту**

1. Опис спроектованих тестів по п.1,2 Завдання.
2. Опис результатів тестування
3. Висновки.

Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навиків і відповідях на питання.

**Теоретичні відомості**

**Види тестування**

Тестування програмного забезпечення включає в себе цілий комплекс дій, аналогічних послідовності процесів розробки програмного забезпечення. У нього входять:

- постановка задачі для тесту;
- проектування тесту ;
- написання тестів;
- тестування тестів;
- виконання тестів;
- вивчення результатів тестування.

Найбільш важливим є проектування тестів. Існують різні підходи до проектування тестів.

Перший полягає в тому, що тести проектуються на основі зовнішніх специфікацій програм і модулів або специфікацій сполучення модуля з іншими модулями, програма при цьому розглядається як «чорний ящик».

Суть тесту заключається в тому, щоб перевірити, чи відповідає програма зовнішнім специфікаціям. При цьому зміст модуля не має значення. Такий підхід отримав назву – стратегія «**чорного ящика**».

Другий підхід – стратегія «**білого ящика**», заснований на аналізі логіки програми. При такому підході тестування полягає у перевірці кожного шляху, кожної гілки алгоритму. При цьому зовнішня специфікація до уваги не береться.

Жоден з цих підходів не є оптимальним. Реалізація тестування методом «чорного ящика» зводиться до перевірки всіх можливих комбінацій вхідних даних. Неможливо протестувати програму, подаючи на вхід нескінченну безліч значень, тому обмежуються певним набором даних. При цьому виходять з максимальної віддачі тесту в порівнянні з витратами на його створення. Вона вимірюється ймовірністю того, що тест виявить помилки, якщо вони є в програмі. Витрати вимірюються часом і вартістю підготовки, виконання та перевірки результатів тесту.

Тестування методом «білого ящика» також не дає 100 % - кової гарантії того, що модуль не містить помилок. Навіть якщо припустити, що виконані тести для всіх гілок алгоритму, не можна з повною впевненістю стверджувати, що програма відповідає її специфікаціям. Наприклад, якщо було потрібно написати програму для обчислення кубічного кореня, а програма фактично обчислює корінь квадратний, то реалізація буде абсолютно неправильною, навіть якщо перевірити всі шляхи. Друга проблема – відсутність шляху. Якщо програма реалізує специфікації не повністю (наприклад, відсутня така спеціалізована функція, як перевірка на від'ємне значення вхідних даних програми обчислення квадратного кореня), ніяке тестування існуючих шляхів не виявить такої помилки. І, нарешті, проблема залежності результатів тестування від вхідних даних. Одні дані будуть давати правильний результат, а інші ні. Наприклад, якщо для визначення рівності трьох чисел програмується вираз виду:

$$\text{IF } (A + B + C)/3 = A, \quad (13.1)$$

то воно буде вірним не для всіх значень А, В і С ( помилка виникла в тому випадку, коли з двох значень В або С одне більше, а інше на стільки ж менше А). Якщо концентрувати увагу тільки на тестуванні шляхів, немає гарантії, що ця помилка буде виявлена.

Таким чином, повне тестування програми неможливе, тобто ніяке тестування не гарантує повну відсутність помилок в програмі. Тому необхідно проектувати тести таким чином, щоб збільшити ймовірність виявлення помилки в програмі.

### **Стратегія « білого ящика »**

Існують наступні методи тестування за принципом "білого ящика" :

- покриття операторів;



- покриття рішень;
- покриття умов;
- покриття рішень / умов;
- комбінаторне покриття умов.

### ***Метод покриття операторів***

Метою цього методу тестування є виконання кожного оператора програми хоча б один раз.

### ***Метод покриття рішень ( покриття переходів )***

Згідно з методом покриття рішень кожний напрям переходу має бути реалізовано, принаймні, один раз. Цей метод включає в себе критерій покриття операторів, так як при виконанні всіх напрямків переходів виконуються всі оператори, що знаходяться на цих напрямках.

### ***Метод покриття умов***

Цей метод може дати кращі результати в порівнянні з попередніми. Відповідно до методу покриття умов записується число тестів, достатнє для того, щоб все можливі результати кожної умови в рішенні виконувалися, принаймні, один раз.

### ***Метод покриття рішень / умов***

Критерій покриття рішень/ умов вимагає такого достатнього набору тестів, щоб всі можливі результати кожної умови виконувалися принаймні один раз, всі результати кожного рішення виконувалися принаймні один раз і, крім того, кожній точці входу передавалося управління принаймні один раз.

Недоліки методу:

- не завжди можна перевірити всі умови;
- неможливо перевірити умови, які приховані іншими умовами;
- недостатня чутливість до помилок в логічних виразах.

### ***Метод комбінаторного покриття умов***

Критерій комбінаторного покриття умов задовольняє також і критеріям покриття рішень, покриття умов і покриття рішень / умов.

Цей метод вимагає створення такого числа тестів, щоб всі можливі комбінації результатів умови в кожному рішенні виконувалися принаймні один раз.

### **Контрольні питання**

1. Охарактеризуйте етап реалізації та тестування програмного продукту.
2. Які існують види тестування ?
3. Назвіть критерії вибору тестів.
4. Перелічіть властивості тестів.
5. Наведіть критерії надійності програм.
6. У чому полягає оцінка надійності програм ?

### **Рекомендована література**

1. Гагарина Л.Г. Технология разработки программного обеспечения : учебное пособие [Текст] / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул – М.: ИД «ФОРУМ» -2008 .-400с.
2. Иванова Г.С. Технология программирования: Ученик [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.

## **ЛАБОРАТОРНА РОБОТА №№ 15, 16**

### **Моделювання проекту з допомогою пакету Rational Rose (Технологія RUP)**

**Мета:** Ознайомитися із основними відомостями і навичками щодо роботи в об'єктно-орієнтованому середовищі пакету Rational Rose (технологія RUP). Виконати учбовий проект (див. завдання)

**Підготовка до лабораторної роботи:** Проробити матеріали лекцій за темою UML, відповідні розділи джерел [1, 2, 3, 4] щодо теми «Об'єктно-орієнтований підхід до проектування ПЗ. Мова UML», теоретичні відомості (п.1), Виконання завдання (п.2.).

#### **Завдання:**

1. Ознайомитися з основними елементами інтерфейсу Rational Rose (п.1 Теоретичних відомостей)
2. Виконати проект (п.16 вправи 1 – 6 Теоретичних відомостей). На діаграмах прецедентів і класів відобразити модель відповідно до індивідуального завдання (на основі робіт 3-14).
3. Опис виконаних дій по п.1 Завдання (у довільній формі).
4. Скріншоти головного вікна Rational Rose (по одному на кожну вправу, за вибором виконавця).
5. Скріншоти діаграм прецедентів і класів.
6. Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навиків і відповідях на питання

### **Теоретичні відомості**

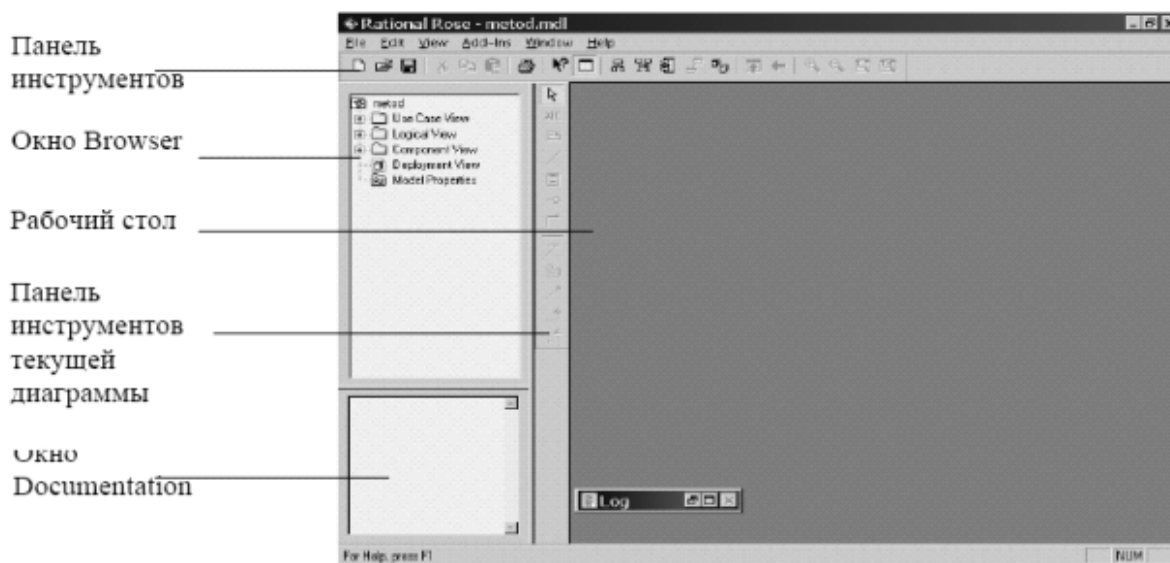
## **15. Основні відомості про роботу в середовищі Rational Rose**

### **15.1 Елементи екрану**

П'ять основних елементів інтерфейсу Rose - це браузер, вікно документації, панелі інструментів, вікно діаграми і журнал (log). Їх призначення полягає в наступному:

- Браузер (browser) – використовується для швидкої навігації по моделі
- Вікно документації (documentation window) – застосовується для роботи з текстовим описом елементів моделі
- Панелі інструментів (toolbars) – застосовуються для швидкого доступу до найбільш поширеним командам
- Вікно діаграми (diagram window) – використовується для перегляду і редагування однієї або декількох діаграм UML
- Журнал (log) – застосовується для перегляду помилок і звітів про результати виконання різних команд

На рис. 15.1 та 15.2. показані різні частини інтерфейсу R-Rose.



*Рис. 15.1. Головне вікно «Rational Rose»*

У верхній частині екрану знаходиться панель інструментів – Tool Bar.

Зліва - вікно Browser для швидкого доступу до діаграм. Це вікно дозволяє швидко переміщатись по дереву діаграм, перетягувати діаграми «мишою» та змінювати структуру моделі.

Під вікном Browser знаходиться вікно Documentation. У цьому вікні з'являється опис, введений розробником для активного на даний момент елемента.

В правій частині екрану знаходяться ті діаграми, які відкриті в даний момент. Ця частина називається Робочим столом Rational Rose.

Між вікном Browser та Diagram знаходиться панель інструментів поточної діаграми, яка змінюється в залежності від обраної діаграми.

Знизу робочого столу знаходиться згорнуте вікно Log (протокол). В ньому фіксуються всі дії виконані над діаграмою.

## **Браузер**

Браузер – це ієрархічна структура, що дозволяє здійснювати навігацію по моделі.

Все, що додається до неї – дійові особи, варіанти використання, класи, компоненти – буде показано у вікні браузера.

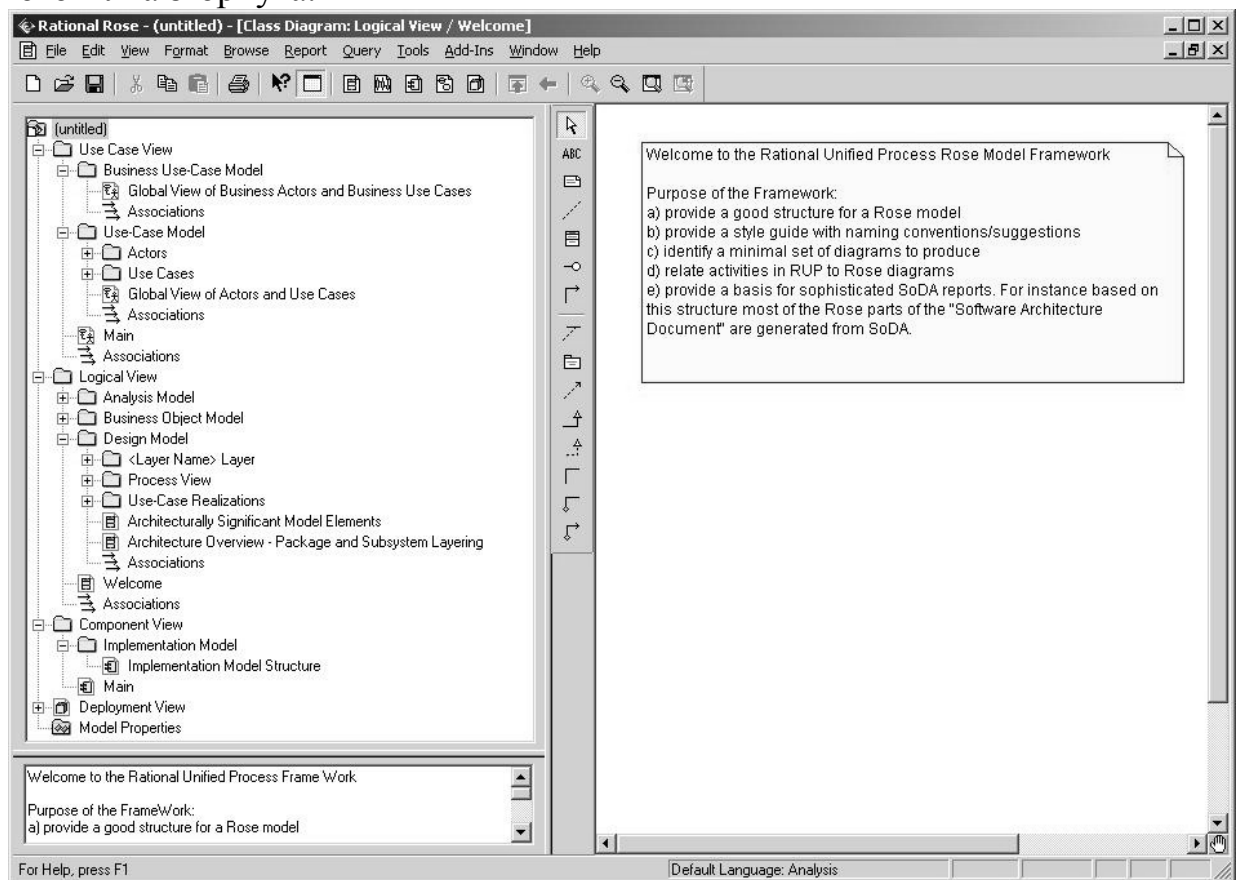
За допомогою браузера можна:

- Додавати до моделі елементи (діючі особи, варіанти використання, класи, компоненти, діаграми і т.д.)
- Переглядати існуючі елементи моделі
- Переглядати існуючі зв'язки між елементами моделі
- Переміщати елементи моделі
- Перейменовувати ці елементи

- Додавати елементи моделі до діаграми
- Зв'язувати елемент з файлом або адресою Інтернет
- Групувати елементи в пакети
- Працювати з деталізованою специфікацією елемента
- Відкривати діаграму

Браузер підтримує чотири вистави (view): подання варіантів використання, компонентів, розміщення і логічне уявлення.

Браузер організований у деревовидному стилі. Кожен елемент моделі може містити інші елементи, що знаходяться нижче його в ієрархії. Знак "-" близько елемента означає, що його гілка повністю розкрита. Знак "+" - що його гілка згорнута.



*Рис. 15.2 Інтерфейс Rational Rose.*

### Вікно документації

З його допомогою можна документувати елементи моделі Rose. Наприклад, можна зробити короткий опис кожної дійової особи. При документуванні класу все, що буде написано у вікні документації, з'явиться потім як коментар в сгенерованном коді, що позбавляє від необхідності згодом вносити ці коментарі вручну. Документація буде виводитися також у звітах, що створюються в середовищі Rose.

## Панелі інструментів

Панелі інструментів Rose забезпечують швидкий доступ до найбільш поширених команд. У цьому середовищі існує два типи панелей інструментів: стандартна панель і панель діаграми. Стандартна панель видна завжди, її кнопки відповідають командам, які можуть використовуватися для роботи з будь-якою діаграмою. Панель діаграми своя для кожного типу діаграм UML.

Всі панелі інструментів можуть бути змінені і налаштовані користувачем. Для цього виберіть пункт меню Tools> Options, потім виберіть вкладку Toolbars.

Щоб показати або приховати стандартну панель інструментів (або панель інструментів діаграми):

1. Виберіть пункт Tools> Options.
2. Виберіть вкладку Toolbars.
3. Щоб зробити видимою або невидимою стандартну панель інструментів, позначте (Або зніміть позначку) контрольний перемикач Show Standard ToolBar (або Show Diagram ToolBar)

Щоб збільшити розмір кнопок на панелі інструментів:

1. Клацніть правою кнопкою миші на необхідній панелі.
2. Виберіть у спливаючому меню пункт Use Large Buttons (Використовувати великі кнопки)

Щоб налаштувати панель інструментів:

1. Клацніть правою кнопкою миші на необхідній панелі.
2. Виберіть пункт Customize (налаштувати)
3. Щоб додати або видалити кнопки, виберіть відповідну кнопку і потім клацніть на кнопці Add (додати) або Remove (видалити), як показано на рис. 15.3.

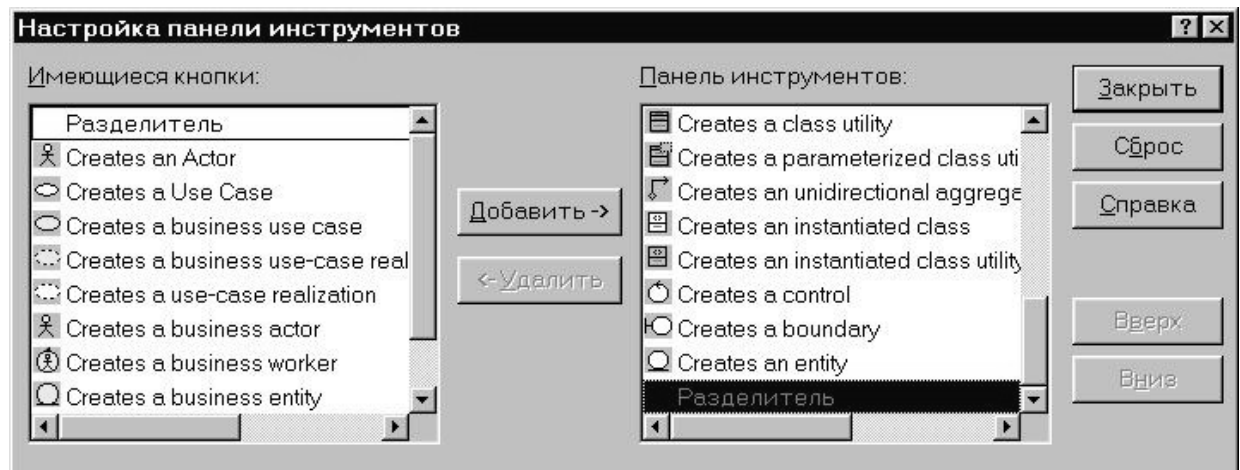


Рис. 15.3. Налаштування стандартної панелі інструментів.

Існує два способи видалити елемент моделі - з однієї діаграми або з усієї моделі.

Щоб видалити елемент моделі з діаграми:

1. Виділіть елемент на діаграмі.
2. Натисніть на клавішу Delete.
3. Зверніть уваги, що, хоча елемент і видалений з діаграми, він залишився в браузері і на інших діаграмах системи.

Щоб видалити елемент з моделі:

1. Виділіть елемент на діаграмі.
2. Виберіть пункт меню Edit > Delete from Model або натисніть клавіші CTRL + D.

## **Вікно діаграми**

У вікні діаграми видно, як виглядає одна або кілька діаграм UML моделі. При внесення в елементи діаграми змін Rose автоматично оновить браузер. Аналогічно, при внесенні змін до елемент за допомогою браузера Rose автоматично оновить відповідні діаграми. Це допомагає підтримувати модель в несутеречливою стані.

## **Журнал**

Принаймні роботи над вашою моделлю певна інформація буде спрямовуватися у вікно журналу. Наприклад, туди поміщаються повідомлення про помилки, що виникають при генерації коду.

Не існує способу закрити журнал зовсім, але його вікно може бути мінімізовано.

## **15.2 Чотири представлення моделі Rose**

У моделі Rose підтримується чотири вистави (views) – представлення варіантів використання, логічне подання, подання компонентів і уявлення розміщення. Кожне з них призначено для своїх цілей.

**Представлення варіантів використання**

Це подання містить модель бізнес-процесів і модель варіантів використання.

На рис. 15.2 показано, як виглядає уявлений варіант використання в браузері Rose.

**Логічне уявлення**

Логічне уявлення концентрується на тому, як система буде реалізовувати поведінка, описане у варіантах використання. Воно дає докладну картину складових частин системи і описує взаємодію цих частин. Логічне уявлення включає в основному класи та діаграми класів. З їх допомогою конструюється детальний проект створюваної системи.

**Логічне подання містить:**

- Класи.
- Діаграми класів. Як правило, для опису системи використовується кілька діаграм класів, кожна з яких відображає деяку підмножину всіх класів системи.

- Діаграми взаємодії, застосовувані для відображення об'єктів, що беруть участь у одному потоці подій варіанту використання.

- Діаграми станів.

- Пакети, які є групами взаємопов'язаних класів.

Представлення компонентів

Представлення компонентів містить:

- Компоненти, які є фізичними модулями коду.

- Діаграми компонентів.

- Пакети, які є групами пов'язаних компонентів.

### **Представлення розміщення**

Останнє подання Rose – це подання розміщення. Воно відповідає фізичного розміщення системи, яке може відрізнятися від її логічної архітектури.

**У уявлення розміщення входять:**

- Процеси, що є потоками (threads), виконуваними у відведеній для них області пам'яті.

- Процесори, що включають будь-які комп'ютери, здатні обробляти дані. Будь процес виконується на одному або декількох процесорах.

- Пристрої, тобто будь-яка апаратура, не здатна обробляти дані. До числа таких пристроїв належать, наприклад, термінали введення-виведення та принтери.

- Діаграма розміщення.

## **15.3 Параметри налаштування відображення**

У Rose є можливість налаштувати діаграми класів так, щоб:

- Показувати всі атрибути і операції

- Сховати операції

- Сховати атрибути

- Показувати тільки деякі атрибути або операції

- Показувати операції разом з їх повними сигнатурами або тільки їх імена.

- Показувати чи не показувати видимість атрибутів і операцій.

- Показувати чи не показувати стереотипи атрибутів і операцій.

Значення кожного параметра за замовчуванням можна задати за допомогою вікна, що відкривається при виборі пункту меню Tools> Options.

У даного класу на діаграмі можна:

- Показати всі атрибути.

- Сховати всі атрибути.

- Показати тільки вибрані вами атрибути.

- Придушити висновок атрибутів.

Придушення виведення атрибутів призведе не тільки до зникнення атрибутів з діаграми, але і до видалення лінії, що показує місце розташування атрибутів в класі.

Існує два способи зміни параметрів подання атрибутів на діаграмі.



Можна встановити потрібні значення у кожного класу індивідуально. Можна також змінити значення потрібних параметрів за замовчуванням до початку створення діаграми класів. Внесені таким чином зміни вплинуть тільки на новостворювані діаграми.

Щоб показати всі атрибути класу:

1. Виділіть на діаграмі потрібний клас.
2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.
3. У ньому виберіть Options> Show All Attributes.

Щоб показати у класу лише обрані атрибути:

1. Виділіть на діаграмі потрібний вам клас.
2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.
3. У ньому виберіть Options> Select Compartment Items.

4. Вкажіть потрібні вам атрибути у вікні Edit Compartment.

Щоб придушити висновок всіх атрибутів класу діаграми:

1. Виділіть на діаграмі потрібний вам клас.
2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.
3. У ньому виберіть Options> Suppress Attributes.

Щоб змінити прийнятий за замовчуванням вид атрибута:

1. У меню моделі виберіть пункт Tools> Options.
2. Перейдіть на вкладку Diagram.
3. Для установки значень параметрів відображення атрибутів за замовчуванням

скористайтесь контрольними перемикачами Suppress Attributes і Show All Attributes.

Зміна цих значень за замовчуванням вплине тільки на нові діаграми.

Вид існуючих діаграм класів не зміниться.

Як і у випадку атрибутів, є кілька варіантів представлення операцій на діаграмах.

- Показати всі операції.
- Показати тільки деякі операції.
- Сховати всі операції.
- Придушити висновок операцій.

Крім того, можна:

- Показати тільки ім'я операції. Це означає, що на діаграмі буде представлено тільки ім'я операції, але не аргументи або тип значення.

- Показати повну сигнатуру операції. На діаграмі буде представлено не тільки ім'я операції, але і всі її параметри, типи даних параметрів і тип значення операції.

Щоб показати всі операції класу:

1. Виділіть на діаграмі потрібний вам клас.

2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.

3. У ньому виберіть Options> Show All Operations.

Щоб показати тільки обрані операції класу:

1. Виділіть на діаграмі потрібний вам клас.

2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.

3. У ньому виберіть Options> Select Compartment Items.

4. Вкажіть потрібні вам операції у вікні Edit Compartment.

Щоб придушити висновок всіх операцій класу діаграми:

1. Виділіть на діаграмі потрібний вам клас.

2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.

3. У ньому виберіть Options> Suppress Operations.

Щоб показати на діаграмі класів сигнатуру операції:

1. Виділіть на діаграмі потрібний вам клас.

2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.

3. У ньому виберіть Options> Show Operation Signature.

Щоб змінити прийнятий за замовчуванням вид операції:

1. У меню моделі виберіть пункт Tools> Options.

2. Перейдіть на вкладку Diagram.

3. Для установки значень параметрів відображення операцій за замовчуванням скористайтесь контрольними перемикачами Suppress Operations, Show All Operations і Show Operation Signatures.

Щоб показати видимість атрибута або операції класу:

1. Виділіть на діаграмі потрібний вам клас.

2. Клацніть на ньому правою кнопкою миші, щоб відкрити контекстно-залежне меню.

3. У ньому виберіть Options> Show Visibility.

Щоб змінити прийняте за замовчуванням значення параметра показу видимості:

1. У меню моделі виберіть пункт Tools> Options.

2. Перейдіть на вкладку Diagram.

3. Для установки параметрів відображення видимості за замовчуванням скористайтесь контрольним перемикачем Show Visibility.

Для перемикання між нотаціями видимості Rose та UML:

1. У меню моделі виберіть пункт Tools> Options.

2. Перейдіть на вкладку Notation.

3. Для перемикання між нотаціями скористайтесь перемикачем Visibility as Icons.

Якщо цей перемикач позначений, буде використовуватися нотація Rose. Якщо ні, то нотація UML. Зміна цього параметра вплине тільки на нові діаграми.

Існуючі діаграми класів залишаться колишніми.

## **16. Виконання проекту моделі**

### **16.1 Моделювання бізнес-процесів**

Як приклад створення моделі виконаємо постановку завдання:

*Перед керівником інформаційної служби університету ставиться завдання розробки автоматизованої системи реєстрації студентів на додаткові платні курси. Система повинна дозволяти студентам реєструватися на курси і переглядати свої таблиці успішності з персональних комп'ютерів, підключених до локальної мережі університету.*

*Професора повинні мати доступ до системи, щоб вказати курси, які вони будуть читати, і проставити оцінки за курси.*

*В даний час в університеті функціонує база даних, що містить всю інформацію про курси (каталог курсів). Реєстрація на курси відбувається наступним чином: у початку кожного семестру студенти можуть запросити у реєстратора каталог курсів, що містить список курсів, пропонує в даному семестрі. Інформація про кожному курсі повинна включати ім'я професора, найменування кафедри та вимоги до попереднього рівня підготовки (Прослуханих курсів).*

*Студент може вибрати 4 курсу в майбутньому семестрі. На додаток до цього кожен студент може вказати 2 альтернативних курсу на той випадок, якщо який-небудь з обраних їм курсів виявиться вже заповненим або скасованим. На кожен курс може записатися не більше 10 і не менше 3 студентів (якщо менше 3, то курс буде відмінений). У кожному семестрі існує період часу, коли студенти можуть змінити свої плани (додати або відмовитися від вибраних курсів). Після того, як процес реєстрації деякого студента завершений, реєстратор направляє інформацію в розрахункову систему, щоб студент міг внести плату за семестр. Якщо курс виявиться заповненим в процесі реєстрації, студент повинен бути повідомлений про це до остаточного формування його особистого навчального плану. У кінці семестру студенти можуть переглянути свої таблиці успішності.*

Студент за індивідуальним завданням виконає постановку задачі та розробку моделі.

### **16.2 Створення моделі варіантів використання**

Дійові особи (business actors):

- Студент - записується на курси і переглядає свій таблиць успішності.
- Професор - вибирає курси для викладання і ставить оцінки за курси.
- Розрахункова система - отримує інформацію з оплати за курси.
- Каталог курсів - база даних, що містить інформацію про курси.

## Вправа 1. Створення дійових осіб в середовищі Rational Rose

При запуску Rational Rose у вікні Create New Model виберіть варіант Rational Unified Process (Рис. 15.4). В результаті екран Rose прийме вигляд, показаний на рис. 15.2.

Щоб помістити дійову особу в браузер:

1. Клацніть правою кнопкою миші на пакеті Business Use Case Model подання Use Case View в браузері.
2. Виберіть в меню пункт New> Actor
3. У браузері з'явиться нова дійова особа під назвою NewClass. Зліва від його імені ви побачите піктограму діючої особи UML.
4. Виділивши нова дійова особа, введіть його ім'я.
5. Клацніть правою кнопкою миші на діючому обличчі.
6. У меню, виберіть пункт Open Specification.
7. У полі стереотипу виберіть Business Actor і натисніть на кнопку OK.
8. Після створення дійових осіб збережіть модель під ім'ям coursereg (analysis) за допомогою пункту меню File> Save.

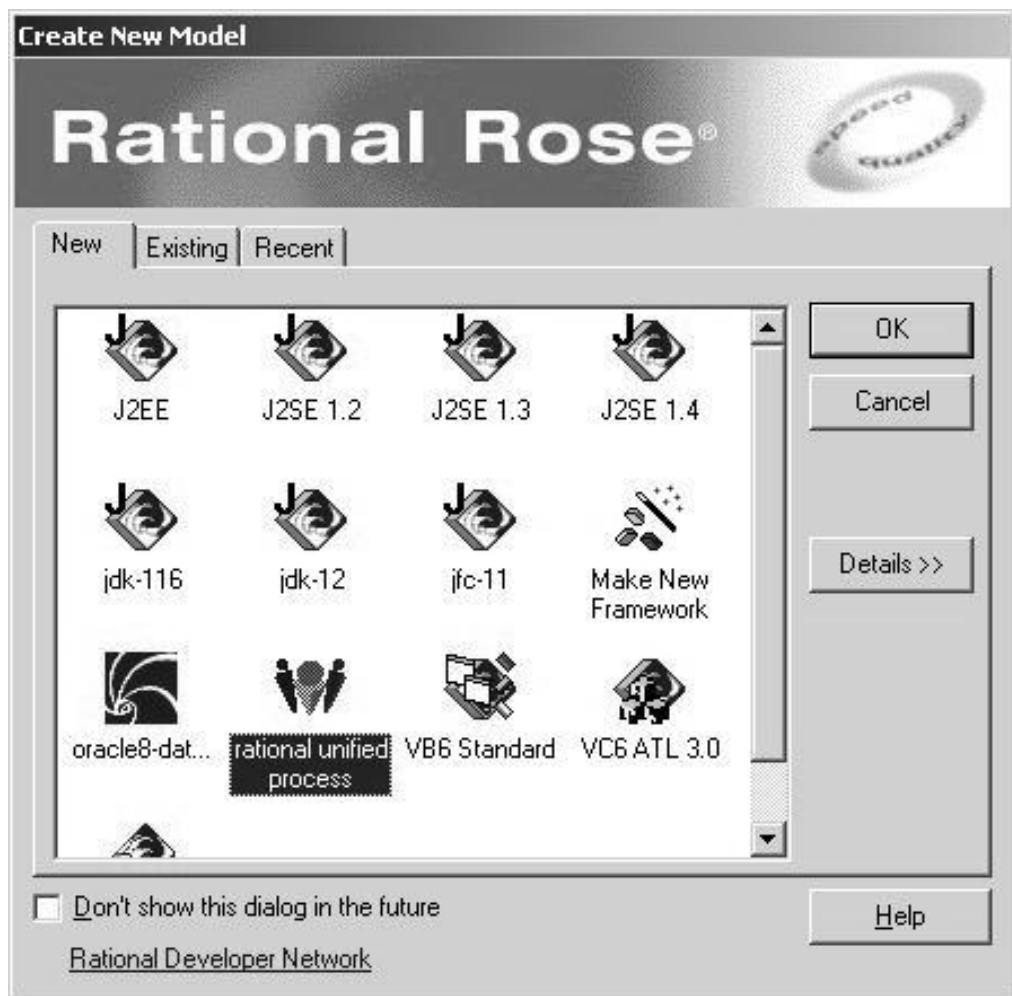


Рис. 15.4. Вікно вибору шаблону моделі

Варіанти використання:

Виходячи з потреб дійових осіб, виділяються наступні варіанти використання (Business Use Case):

- Зареєструватися на курси;
- Переглянути таблиць успішності;
- Вибрати курси для викладання;
- Проставити оцінки.

## **Вправа 2. Створення варіантів використання в середовищі Rational Rose**

Щоб помістити варіант використання в браузер:

1. Клацніть правою кнопкою миші на пакеті Business Use Case Model подання Use Case View в браузері.
2. Виберіть в меню пункт New> Use Case
3. Новий варіант використання під назвою NewUseCase з'явиться в браузері. Зліва від нього буде видно піктограма варіанту використання UML.
4. Виділивши новий варіант використання, введіть його назву.
5. Клацніть правою кнопкою миші на варіанті використання.
6. У меню, виберіть пункт Open Specification.
7. У полі стереотипу виберіть Business Use Case і натисніть на кнопку ОК.

### **Діаграма варіантів використання:**

Створіть діаграму варіантів використання для бізнес-моделі системи реєстрації.

Необхідні для цього дії детально перераховані далі. Готова діаграма варіантів використання повинна виглядати як на рис. 15.5.

## **Вправа 3. Побудова діаграми варіантів використання**

Для створення нової діаграми варіантів використання:

1. Клацніть правою кнопкою миші на пакеті Business Use Case Model подання Use Case View в браузері.
2. З спливаючого меню виберіть пункт New> Use Case Diagram.
3. Виділивши нову діаграму, введіть її ім'я (Business Use Case Diagram).
4. Двічі клацніть на назві цієї діаграми в браузері, щоб відкрити її.
5. Щоб помістити дійова особа або варіант використання на діаграму, перетягніть його мишею з браузера на діаграму варіантів використання.
6. За допомогою кнопки Unidirectional Association (Односпрямована асоціація) панелі інструментів намалюйте асоціацію між дійовими особами і варіантами використання.



Рис. 15.5. Діаграма варіантів використання для системи реєстрації.

#### Вправа 4. Додавання описів до варіантів використання

1. Виділіть у браузері варіант використання «Зареєструватися на курси».
2. У вікні документації введіть наступний опис до цього варіанту використання: «Даний Business Use Case дозволяє студенту зареєструватися на конкретні курси в поточному семестрі. Студент може змінити свій вибір, якщо зміна виконується в встановлений час на початку семестру».
3. Створіть за допомогою MS Word текстовий файл із описом варіанту використання «Зареєструватися на курси».

#### **Специфікація Business Use Case «Зареєструватися на курси»:**

Назва:

*Зареєструватися на курси.*

Короткий опис:

*Даний Business Use Case дозволяє студенту зареєструватися на пропоновані курси в поточному семестрі. Студент може змінити свій вибір, якщо зміна виконується в встановлений час на початку семестру.*

*Основний сценарій:*

- 1. Студент приходить до реєстратора і просить зареєструвати його на пропоновані курси або змінити свій графік курсів.*
- 2. Залежно від запиту студента, виконується один з підлеглих сценаріїв (Створити графік або змінити графік).*

*Підлеглий сценарій «Створити графік»:*

- 1. Реєстратор виконує пошук в каталозі курсів доступних на даний момент курсів і видає студенту їх список.*
- 2. Студент обирає зі списку 4 основних курсу і 2 альтернативних курсу.*
- 3. Реєстратор формує графік студента.*
- 4. Виконується підлеглий сценарій «Прийняти графік».*

*Підлеглий сценарій «Змінити графік»:*

- 1. Реєстратор знаходить поточний графік студента.*
- 2. Реєстратор виконує пошук в каталозі курсів доступних на даний момент курсів і видає студенту їх список.*
- 3. Студент може змінити свій вибір курсів, видаляючи або додаючи запропоновані курси.*
- 4. Після вибору реєстратор оновлює графік.*
- 5. Виконується підлеглий сценарій «Прийняти графік».*

*Підлеглий сценарій «Прийняти графік»:*

- 1. Для кожного обраного студентом курсу реєстратор підтверджує виконання студентом попередніх вимог (проходження певних курсів), факт відкриття пропонованого курсу та відсутність конфліктів графіка.*
- 2. Реєстратор вносить студента в список кожного обраного пропонованого курсу. Курс фіксується у графіку.*

*Альтернативні сценарії:*

*Не виконані попередні вимоги, курс заповнений або мають місце конфлікти графіка:*

*Якщо під час виконання підлеглого сценарію «Прийняти графік» реєстратор виявить, що студент не виконав необхідні попередні вимоги, або вибраний їм пропонований курс заповнений (вже записалося 10 студентів), або мають місце конфлікти графіка (два або більше курсів з збігається розкладом), то він пропонує студенту змінити свій вибір курсів, або скасувати формування графіка і повернутися до нього пізніше.*

*Система каталогу курсів недоступна:*

*Якщо під час пошуку в каталозі курсів виявиться, що неможливо встановити зв'язок з системою каталогу курсів, то реєстрацію доведеться перервати і дочекатися відновлення зв'язку.*

*Реєстрація на курси закінчена:*

*Якщо на самому початку виконання реєстрації виявиться, що реєстрація на поточний семестр вже закінчена, то процес завершиться.*

### **Вправа 5. Прикріплення файлу до варіанту використання**

1. Клацніть правою кнопкою миші на варіанті використання.
2. У меню, виберіть пункт Open Specification
3. Перейдіть на вкладку Files.
4. Клацніть правою кнопкою миші на білому полі і з меню, виберіть пункт Insert File.
5. Вкажіть створений раніше файл і натисніть на кнопку Open, щоб прикріпити файл до варіанту використання.

2.3 Створення моделі бізнес-аналізу варіанту використання «Зареєструватися на курси» і відповідної діаграми класів

Виконавці:

1. Реєстратор - формує навчальний план та каталог курсів, записує студентів на курси, веде всі дані про курси, професорів, успішності і студентах.

Сутності:

2. Студент.
3. Професор.
4. Графік студента (список курсів)
5. Курс (в програмі навчання).
6. Пропонований курс (курс в розкладі).

### **Вправа 6. Створення класів, що беруть участь у реалізації бізнес-процесу, і кооперації, яка описує реалізацію бізнес-процесу**

1. Клацніть правою кнопкою миші на пакеті Business Object Model подання Logical View в браузері.
2. Виберіть в меню пункт New> Class. Новий клас під назвою NewClass з'явиться в браузері.
3. Виділіть його і введіть ім'я «Реєстратор».
4. Клацніть правою кнопкою миші на даному класі.
5. У меню, виберіть пункт Open Specification.
6. У полі стереотипу виберіть Business Worker і натисніть на кнопку OK.
7. Створіть аналогічним чином класи-сутності зі стереотипом Business Entity.
8. Клацніть правою кнопкою миші на пакеті Object Model подання Logical View в браузері.
9. У меню, виберіть пункт New> Package
10. Назвіть новий пакет Business Use-Case Realizations.
11. У пакеті Business Use-Case Realizations створіть кооперацію «Зареєструватися на курси» (кооперація являє собою варіант використання зі



стереотипом «business use-case realization», який задається в специфікації варіанту використання).

12. Клацніть правою кнопкою миші на створеній кооперації.
13. У меню, виберіть пункт New> Class Diagram.
14. Назвіть нову діаграму класів VOPC.
15. Відкрийте її і перетягніть класи на відкриту діаграму відповідно до рис. 15.6

Діаграма класів для моделі бізнес-аналізу, яка описує Business Use Case «Зареєструватися на курси», наведена на рис. 15.6. (для даних класів використано зображення стереотипу у вигляді мітки - label. Налаштування зображення стереотипу може бути виконана наступними способами:

1. Для всієї моделі - в меню Tools> Options> Diagram> Stereotype Display. зображення стереотипу у вигляді мітки - label.
2. Для окремого елемента - в його контекстному меню Options> Stereotype Display.

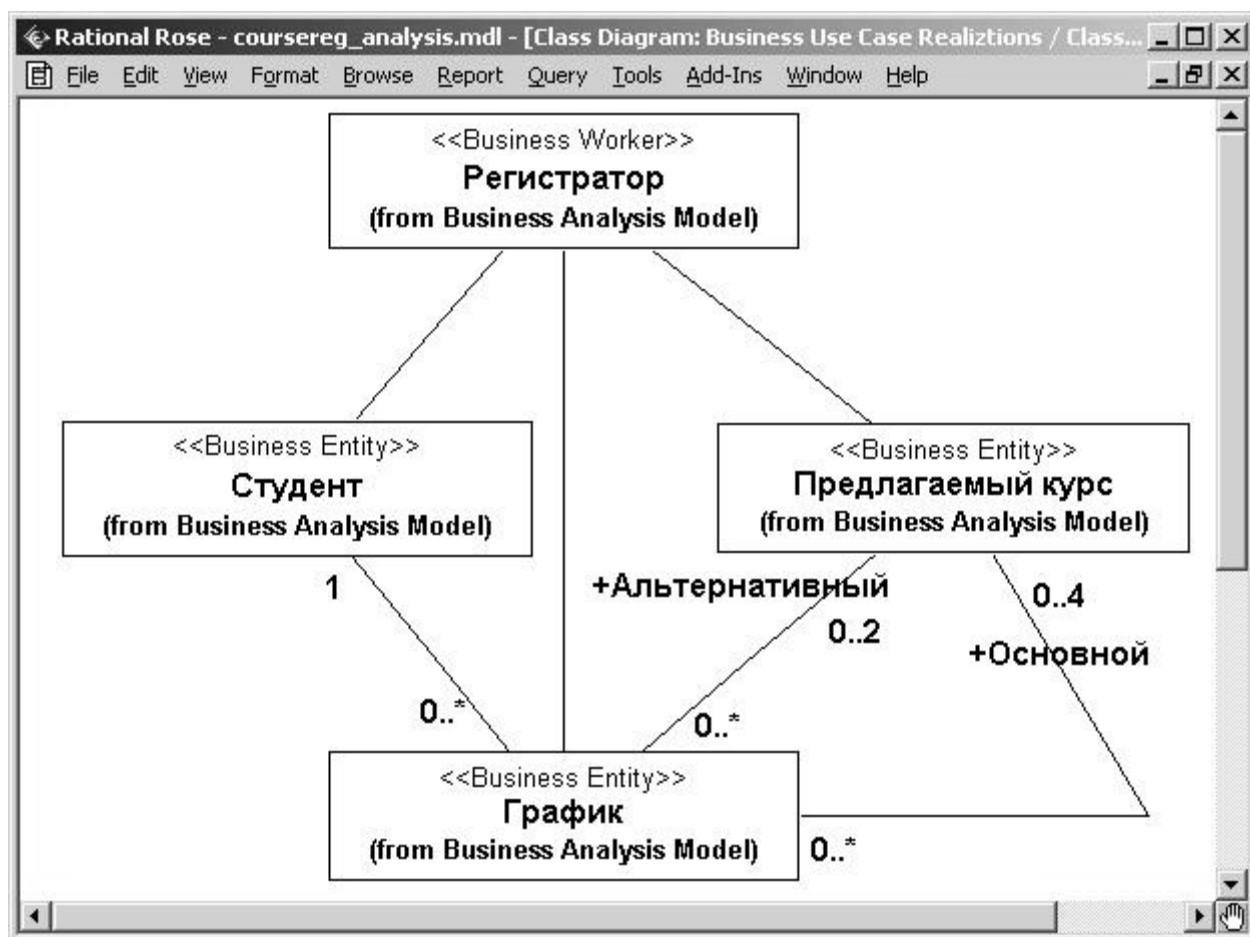


Рис. 15.6. Діаграма класів моделі бізнес-аналізу

### Контрольні питання

1. Дайте визначення UML.
2. Перелічіть головні властивості UML.

3. Перелічіть структурні сутності UML.
4. Для чого використовують представлення прецедентів?
5. Для чого використовують браузер Rational Rose?
6. Для чого використовують вікно документації Rational Rose?
7. Для чого використовують вікно діаграми Rational Rose?
8. Для чого використовують журнал Rational Rose?
9. Як розмістити нового актора на діаграмі прецедентів?
10. Як розмістити новий прецедент на діаграмі прецедентів?

### **Рекомендована література**

1. Вендров А.М.. Объектно-ориентированный анализ и проектирование с использованием языка UML и Rational Rose [Текст] / А.М. Вендров – Практикум, М.: Финансы и статистика, 2004. – 365 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст] / А.М. Вендров - М.: Финансы и статистика, 2006. – 545 с.
3. Иванова Г.С. Технология программирования: Учебник [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.
4. Мацяшек Л. Анализ и проектирование информационных систем с помощью UML 2.0 [Текст] / Лешек Мацяшек К.: Вильямс, 2008, - 816 с.

**ЛАБОРАТОРНА РОБОТА №№ 17, 18**  
**Моделювання проекту з допомогою пакету Rational Rose.**  
**Побудова діаграм прецедентів і класів.**  
**Побудова діаграм послідовностей і діяльності**

**Мета роботи:** Засвоїти методи і правила побудови основних діаграм UML для програмного продукту, розробленого за індивідуальним завданням.

**Підготовка до лабораторної роботи:** Проробити матеріали лекцій за темою UML, відповідні розділи джерел [1, 2, 3] щодо теми «Об'єктно-орієнтований підхід до проектування ПЗ. Мова UML», теоретичні відомості, звіти попередніх лабораторних робіт.

**Завдання:**

1. Побудувати діаграми прецедентів, класів, послідовностей і діяльності для програмного продукту за своїм індивідуальним завданням..
2. Описати сценарій (flow of events) для обраного прецеденту – за вибором виконавця.

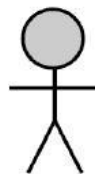
**Зміст звіту**

1. Відзначені діаграми.
2. Опис сценарію обраного прецеденту.
3. Захист звіту по лабораторній роботі полягає в пред'явленні викладачеві отриманих результатів, демонстрації отриманих навиків і відповідях на питання.

**Теоретичні відомості**

**17.1. Діаграма прецедентів (use case diagram)**

Будь-які (в тому числі і програмні) системи проектуються з урахуванням того, що в процесі своєї роботи вони будуть використовуватися людьми та / або взаємодіяти з іншими системами. **Сутності, з якими взаємодіє система в процесі своєї роботи, називаються акторами.** Графічно актор зображується або «чоловічком» (рис. 17.1.), або символом класу з відповідним стереотипом. Обидві форми подання мають один і той же зміст і можуть використовуватися в діаграмах. "Стереотипована" форма частіше застосовується для представлення системних акторів, або у випадках, коли актор має властивості і їх потрібно відобразити (рис. 17.2.).



*Рис. 17.1. Зображення сутності в системі.*



Рис. 17.2. Зображення стереотипованої форми актора.

Прецедент (use case) – опис множини послідовних подій (включаючи варіанти), виконуваних системою, які призводять до результату, що спостерігає Актор. Прецедент являє поведінку сутності, описуючи взаємодію між Актором і системою. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Прецеденти позначаються дуже простим чином – у вигляді еліпса, всередині якого зазначено його назву (рис. 17.3.). Прецеденти і Актори з'єднуються за допомогою ліній. Часто на одному з кінців лінії зображують стрілку, причому спрямована вона до того, у кого запитують сервіс, іншими словами, чийми послугами користуються. Це просте пояснення ілюструє розуміння прецедентів, як сервісів.



Рис. 17.3. Зображення прецеденту

Діаграми прецедентів відносяться до тієї групи діаграм, які представляють динамічні або поведінкові аспекти системи. Це відмінний засіб для досягнення взаєморозуміння між розробниками, експертами та кінцевими користувачами продукту. Як ми вже могли переконатися, такі діаграми дуже прості для розуміння і можуть сприйматися і, що важливо, обговорюватися людьми, які не є фахівцями в галузі розробки ПЗ.



Рис. 17.4. Зображення прецеденту з Акторами

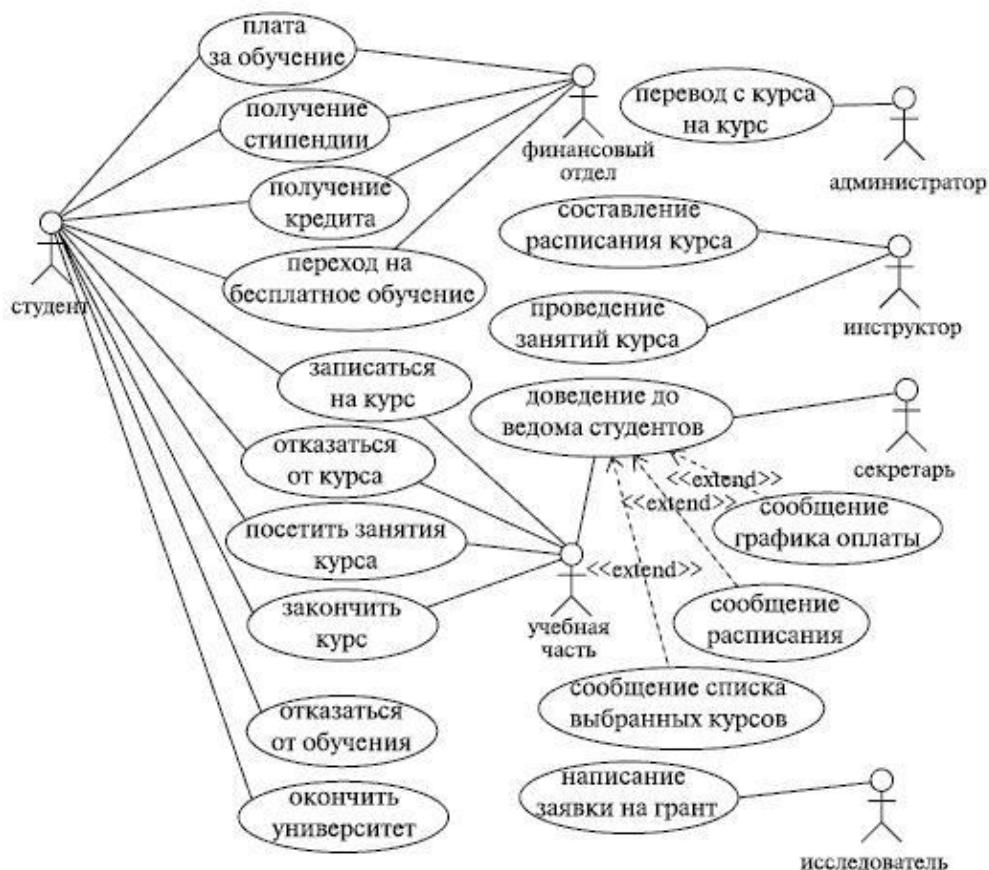


Рис. 17.5. Схема зв'язків Акторів та прецедентів.

Підводячи підсумки, можна виділити такі цілі створення діаграм прецедентів:

- визначення кордону і контексту модельованої предметної області на ранніх етапах проектування;
- формування загальних вимог до поведінки проектованої системи;
- розробка концептуальної моделі системи для її подальшої деталізації;
- підготовка документації для взаємодії із замовниками та користувачами системи.

## 17.2. Діаграма класів (class diagram)

**Клас (class)** – категорія речей, які мають спільні атрибути і операції.

Продовжуючи тему, скажімо, що класи – це будівельні блоки будь-якої об'єктно-орієнтованої системи. Вони являють собою опис сукупності об'єктів із загальними атрибутами, операціями, відносинами і семантикою. При проектуванні об'єктно-орієнтованих систем діаграми класів обов'язкові.

Класи використовуються в процесі аналізу предметної області для складання словника предметної області, що розробляється. Це можуть бути як абстрактні поняття предметної області, так і класи, на які спирається Розробка та які описують програмні або апаратні сутності.

**Діаграма класів** – це набір статичних, декларативних елементів моделі. Діаграми класів можуть застосовуватися і при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається у вихідний код додатка - у більшості існуючих інструментів UML-моделювання можлива кодогенерація для певної мови програмування (зазвичай Java або C ++). Таким чином, діаграма класів – кінцевий результат проектування та відправна точка процесу розробки.

Клас на діаграмі зображується у вигляді прямокутника, розділеного горизонтальними лініями на три частини. У першій частині вказується назва класу. Як правило, ім'я класу складається з одного, максимум двох слів. Друга частина містить перелік атрибутів класу, які характеризують той чи інший об'єкт цього класу в моделі предметної області. Третя частина містить перелік операцій, що відображають його поведінку в моделі предметної області (рис. 17.6.).

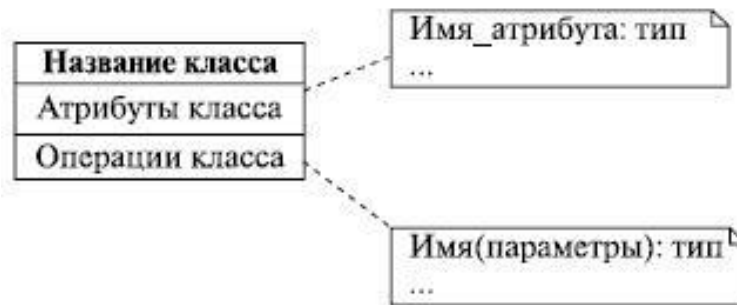


Рис. 17.6. Вигляд діаграми класів.



Рис. 17.7. Діаграма класів поведінки моделі вибору побутової техніки.

Перший приклад (рис. 17.7.) досить простий. Як бачимо, він, хоч і трохи однобоко, ілюструє за допомогою операції спадкування або генералізації "генеалогічне древо" побутової техніки.

Думаємо, ми б зрозуміли сенс цієї діаграми, навіть якщо б нічого не знали про класи і не займалися програмуванням взагалі.

Розглянемо ще приклад (рис. 17.8.): І знову-таки сенс цієї діаграми ясний без особливих пояснень. Навіть побіжно розглянувши її, можна легко здогадатися, що вона описує предметну область завдання про автоматизацію роботи якогось вузу або навчального центру. Зверніть увагу на позначення кратності на кінцях зв'язків.

Як бачимо, тут вже все більш серйозно – крім кратності позначені властивості (і їх типи) та операції, і взагалі, ця діаграма справляє враження набору класів для реалізації, а не просто опису предметної області, як попередні.



Рис. 17.8. Діаграма класів роботи вузу

### 17.3. Діаграма об'єктів (object diagram)

#### Об'єкт (object) -

- конкретна матеріалізація абстракції;
- сутність з добре визначеними межами, в якій вміщені стан і поведінку;
- екземпляр класу (вірніше, класифікатора - актор, клас або інтерфейс). Об'єкт унікально ідентифікується значеннями атрибутів, що визначають його стан в даний момент часу.

Об'єкт, як і клас, позначається прямокутником, але його ім'я підкреслюється. Під словом ім'я тут ми розуміємо назву об'єкта та назву його класу, розділені двокрапкою. Для вказівки значень атрибутів об'єкта в його позначенні може бути передбачена спеціальна секція. Ще один нюанс полягає в тому, що об'єкт може бути анонімним: це потрібно в тому випадку,

якщо в даний момент не важливо, який саме об'єкт даного класу бере участь у взаємодії.

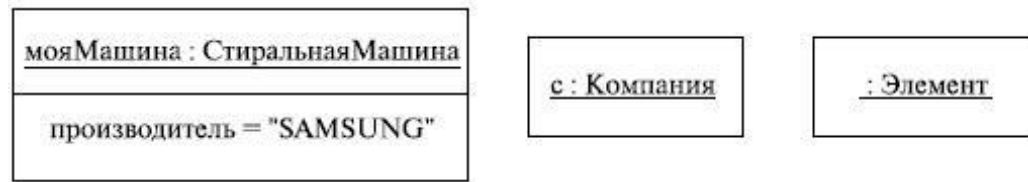


Рис. 17.9. Діаграма об'єктів.

Для чого потрібні діаграми об'єктів? Вони показують безліч об'єктів – екземплярів класів (зображених на діаграмі класів) і відносин між ними в деякий момент часу. Тобто діаграма об'єктів – це свого роду знімок системи у певний момент часу, що показує безліч об'єктів, їх стан і відносини між ними в даний момент.

Таким чином, діаграми об'єктів представляють статичний вид системи з точки зору проектування і процесів, будучи основою для сценаріїв, що описуються діаграмами взаємодії.

Кажучи іншими словами, діаграма об'єктів використовується для пояснення і деталізації діаграм взаємодії, наприклад, діаграм послідовностей.

Наведемо найпростіший приклад такої діаграми (рис. 17.10.).

Про що тут йде мова, в принципі, зрозуміло: деяка фірма "розкручує" новий товар або послугу. У цьому процесі беруть участь віце-президент з маркетингу, віце-президент з продажу, менеджер з продажу, торговий агент, спеціаліст з реклами, якесь друковане видання і покупець.

Причому навіть без вказівки повідомлень, якими обмінюються ці об'єкти, добре видно, хто з ким взаємодіє. До речі, зверніть увагу, що на цій діаграмі всі об'єкти анонімні!



Рис. 17.10. Діаграма об'єктів маркетингових дій компанії.



## 17.4. Діаграма послідовностей (sequence diagram)

**Діаграма послідовностей** відображає взаємодію об'єктів в динаміці.

У UML взаємодія об'єктів розуміється, як обмін інформацією між ними. При цьому інформація приймає вид повідомлень. Крім того, що повідомлення несе якусь інформацію, воно певним чином також впливає на одержувача. Як бачимо, в цьому плані UML повністю відповідає основним принципам ООП, відповідно до яких інформаційна взаємодія між об'єктами зводиться до відправки і прийому повідомлень.

Діаграма послідовностей відноситься до діаграм взаємодії UML, що описує поведінкові аспекти системи, але розглядає взаємодію об'єктів в часі. Іншими словами, діаграма послідовностей відображає часові особливості передачі і прийому повідомлень об'єктами.

Можна сказати, що щось подібне робить і діаграма прецедентів.

Так, дійсно, діаграми послідовностей можна (і потрібно) використовувати для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це гарний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертають результати, що пов'язані з повідомленнями. Втім, часто, результати що повертаються, позначають лише в тому випадку, якщо це не очевидно з контексту.

Які ж позначення використовуються на діаграмі послідовностей

- об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів),
- повідомлення (виклики методів) – лініями зі стрілками,
- результати, що повертаються – пунктирними лініями зі стрілками.

Прямокутники на вертикальних лініях під кожним з об'єктів показують "час життя" (фокус) об'єктів. Втім, досить часто їх не зображують на діаграмі, все це залежить від індивідуального стилю проектування.

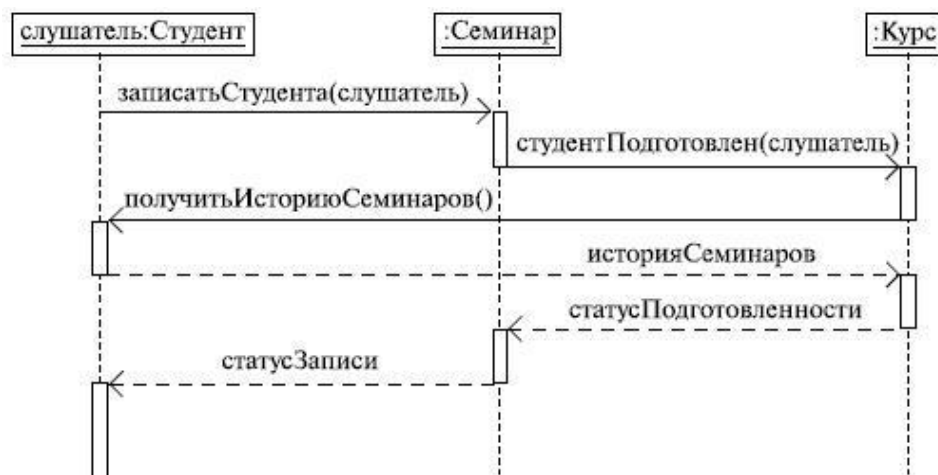


Рис. 17.11. Діаграма послідовностей вибору семінару.

Сенс діаграми (рис.17.11.) цілком зрозумілий: студент хоче записатися на якийсь семінар, пропонований в рамках деякого навчального курсу. З цією метою проводиться перевірка підготовленості студента, для чого запитується список (історія) семінарів курсу, вже пройдених студентом (перейти до наступного семінару можна, лише пропрацювавши матеріал попередніх занять - знайома картина, чи не так?). Після отримання історії семінарів об'єкт класу "Семінар" отримує статус підготовленості, на основі якої студенту повідомляється результат (статус) його спроби запису на семінар.

Ще один приклад діаграми послідовностей представлено на рисунку 17.12., де змодельовано роботу деякої інтернет сторінки на якій відбувається обчислення певних полів.

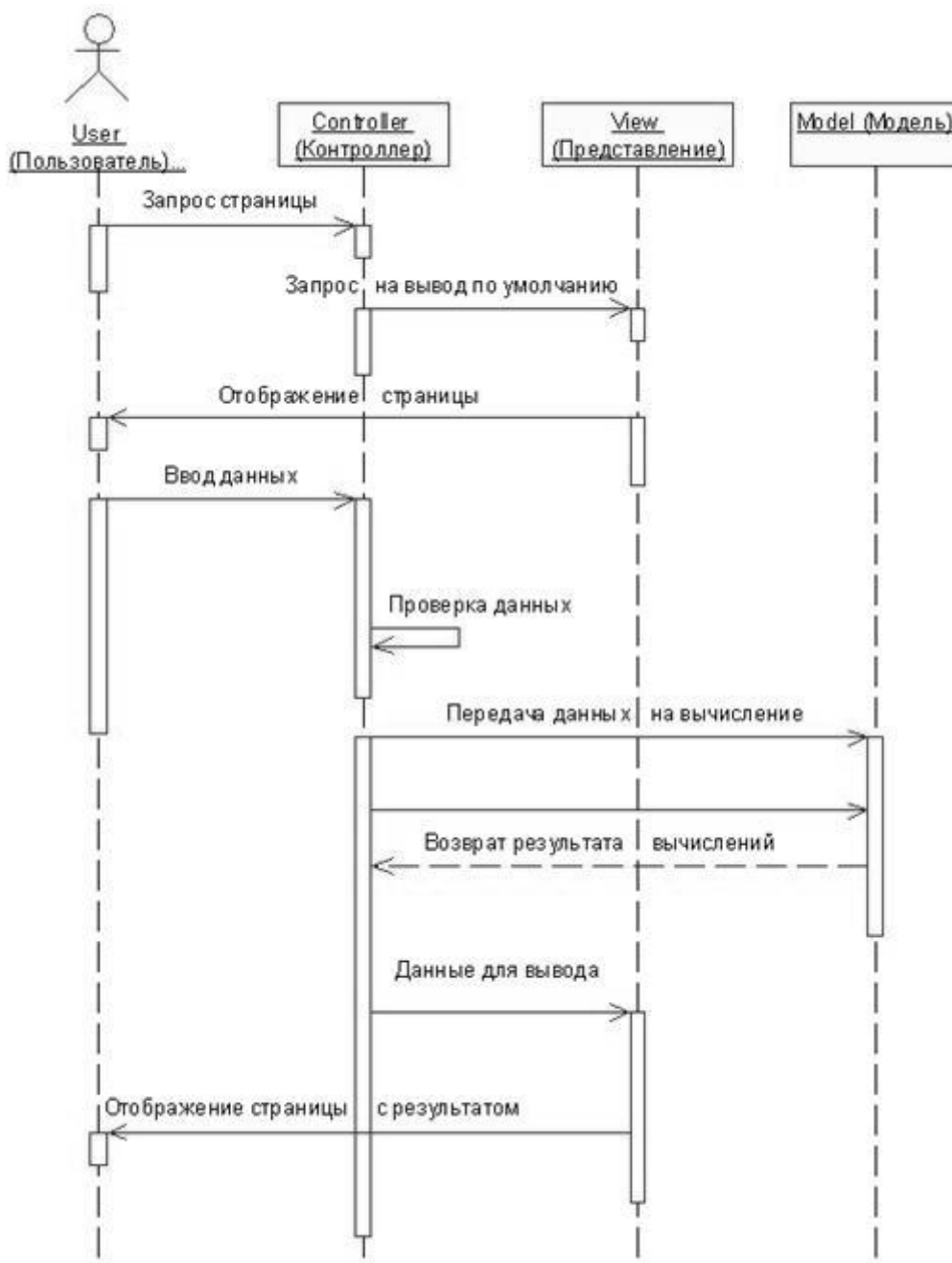


Рис. 17.12. Діаграма послідовностей роботи сторінки з обчислювальним полем.

### **Контрольні питання**

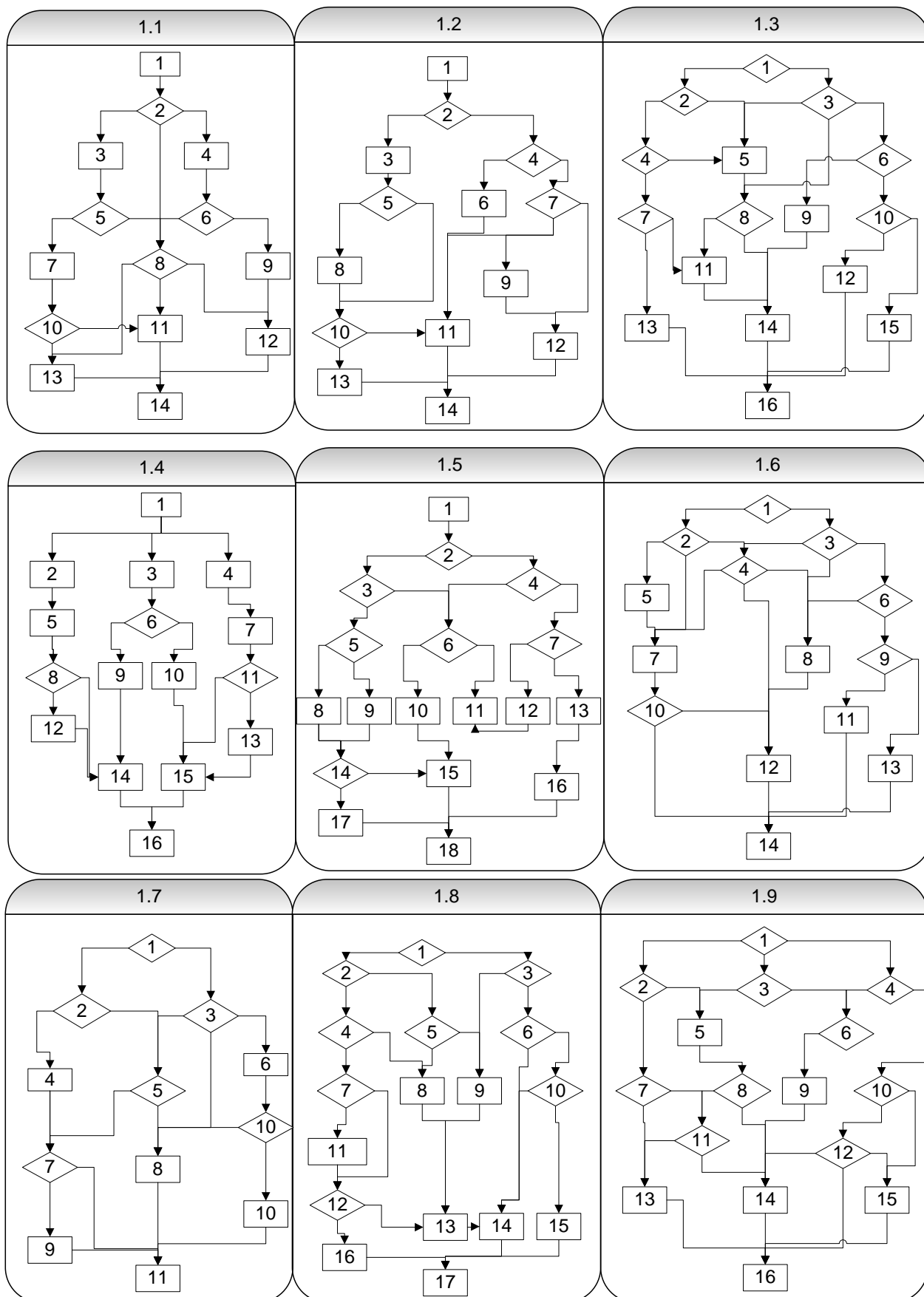
- 1 Для чого використовують діаграму прецедентів?
- 2 Дайте визначення актора.
- 3 Дайте визначення прецеденту.
- 4 Що таке сценарій прецеденту?
- 5 Опишіть шаблон сценарію прецеденту.
- 5 Дайте визначення класу й об'єкта.
- 6 Для чого використовують діаграми класів і об'єктів?

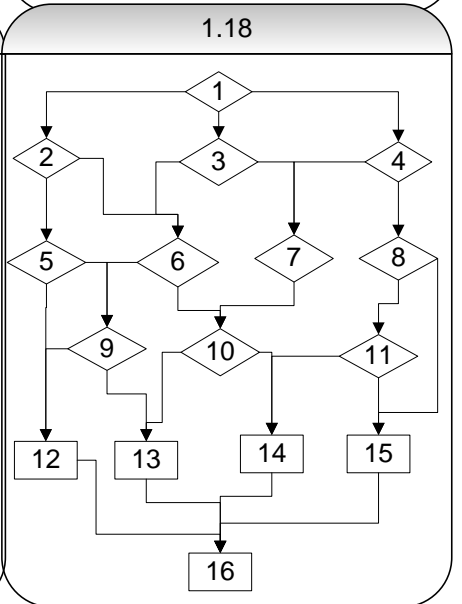
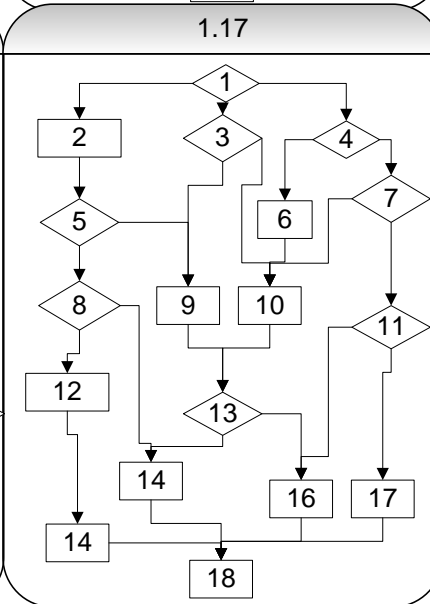
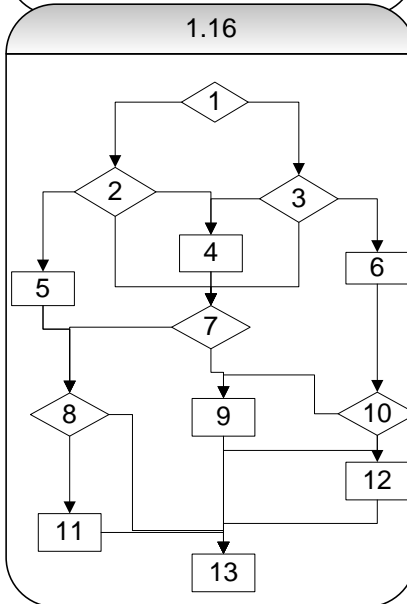
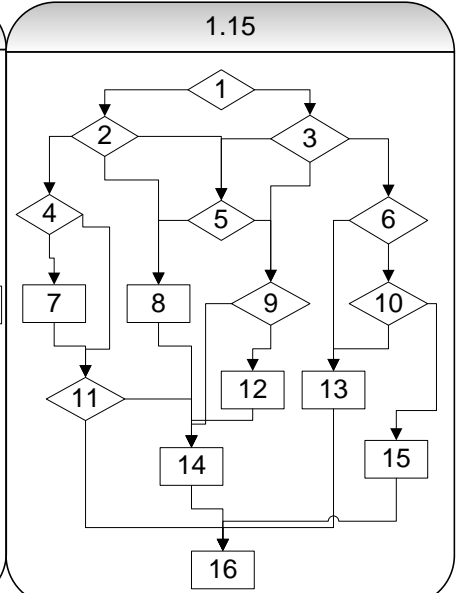
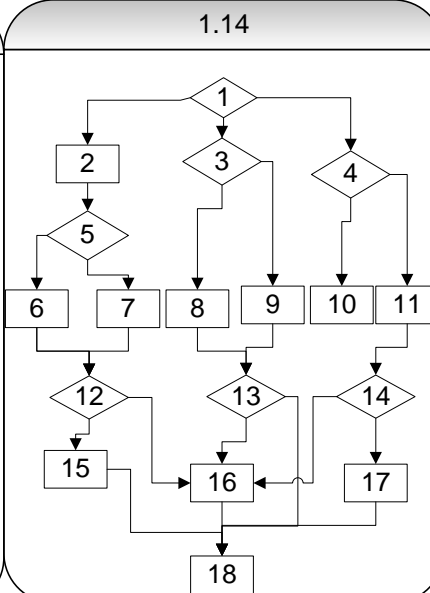
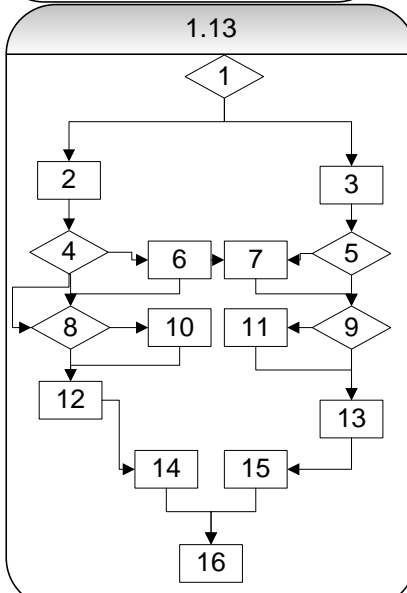
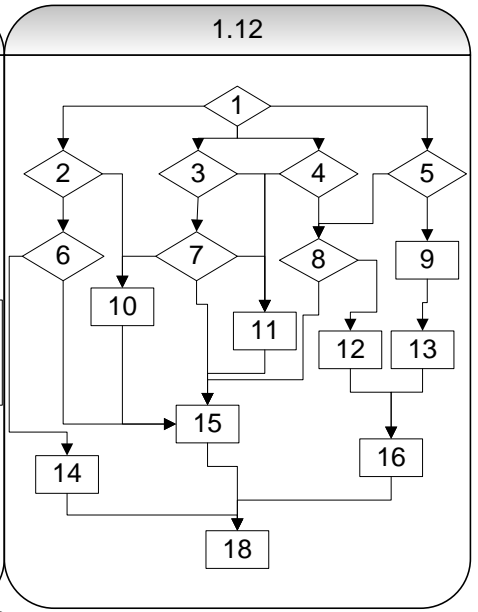
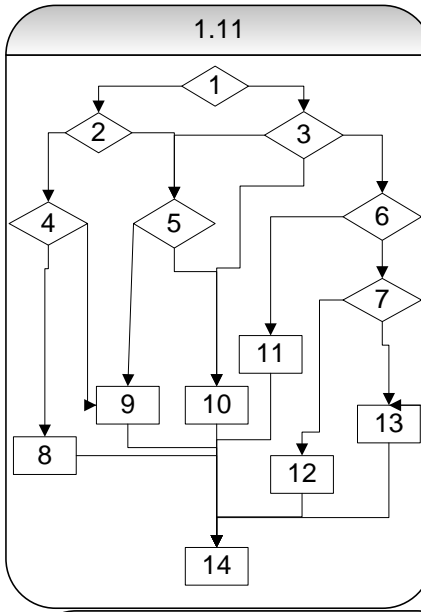
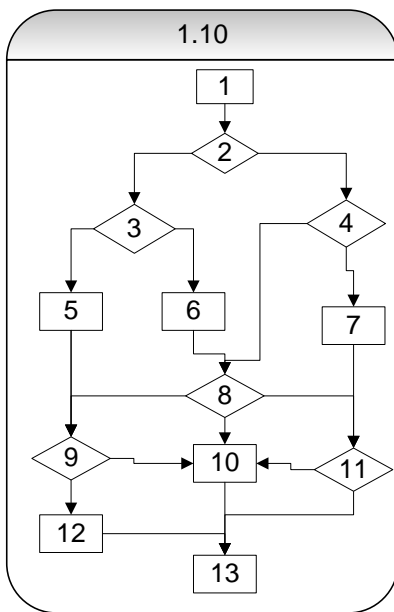
### **Рекомендована література**

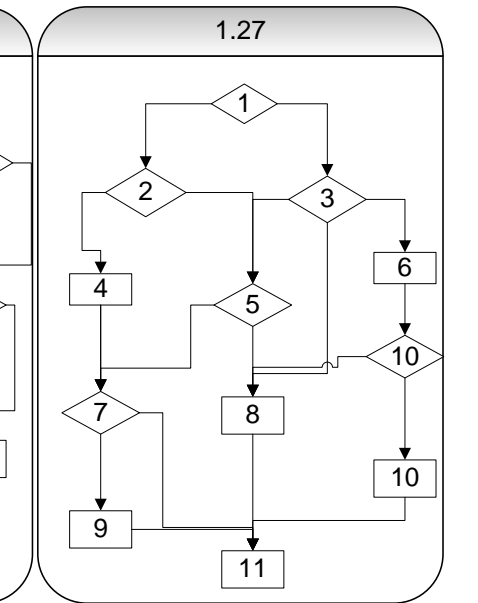
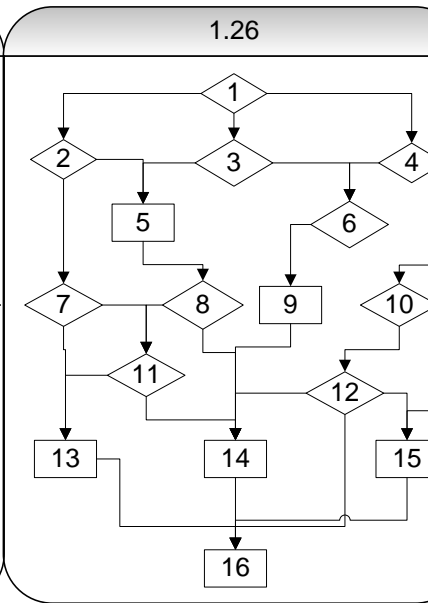
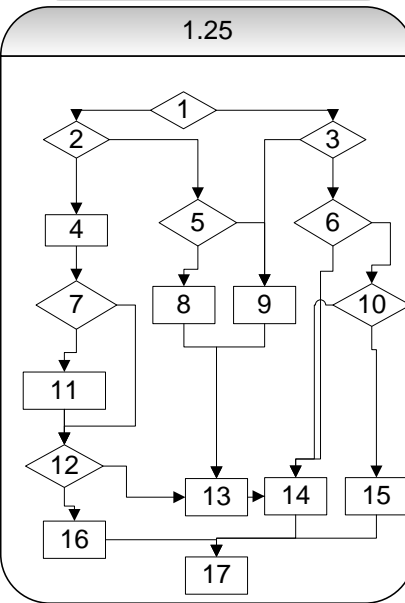
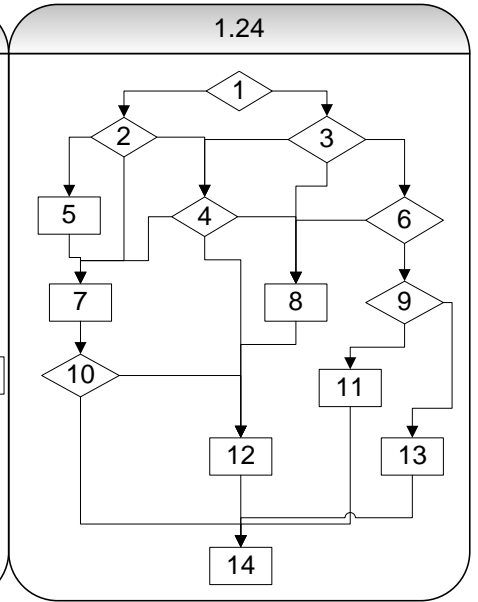
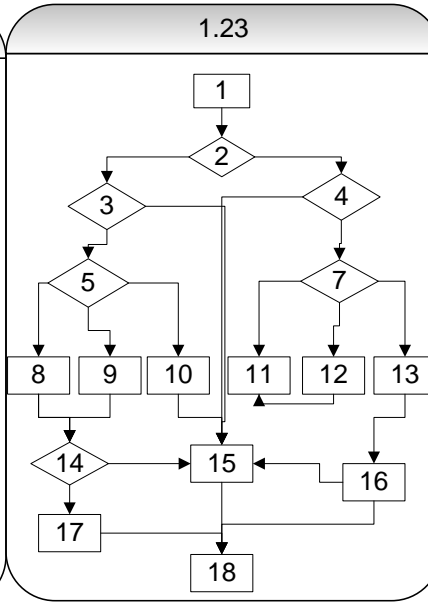
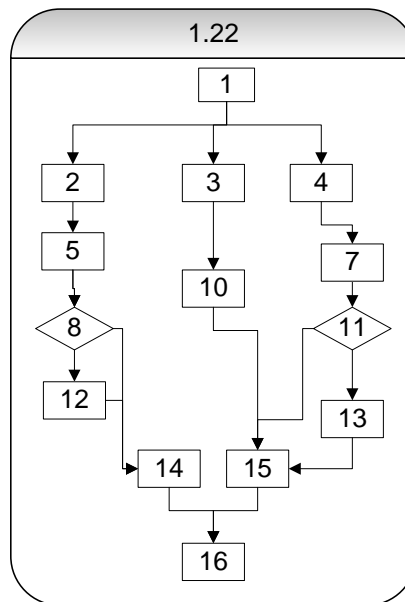
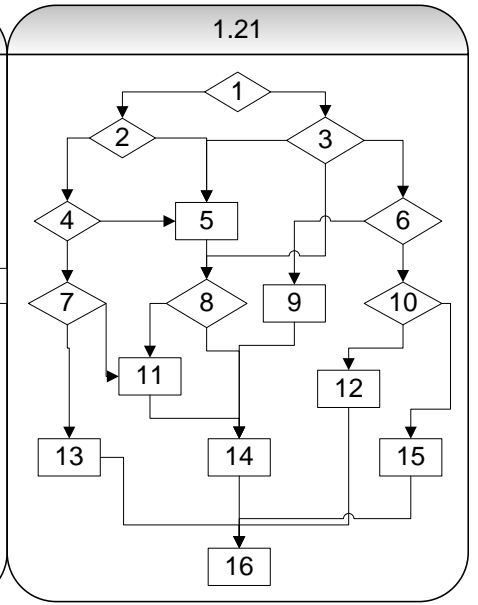
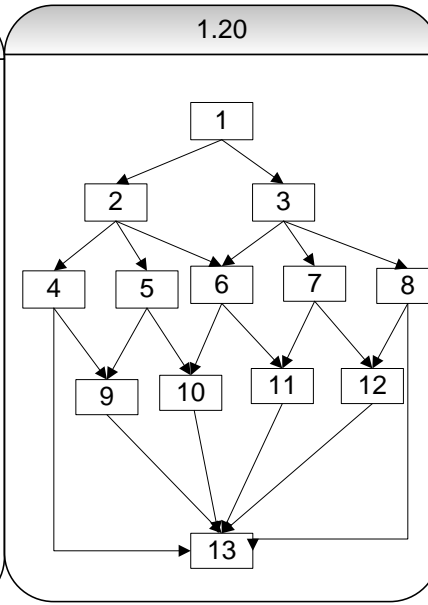
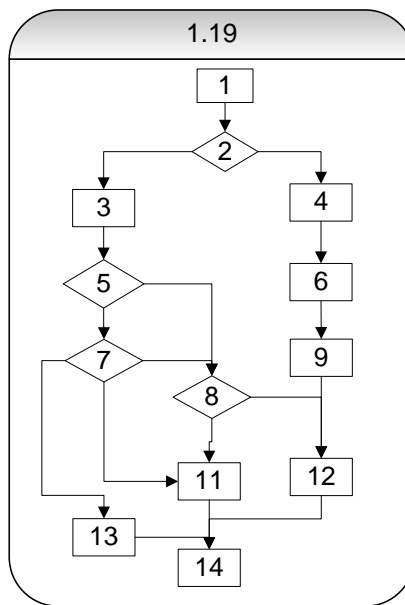
1. Вендров А.М.. Объектно-ориентированный анализ и проектирование с использованием языка UML и Rational Rose [Текст] / А.М. Вендров – Практикум, М.: Финансы и статистика, 2004. – 365 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст] / А.М. Вендров - М.: Финансы и статистика, 2006. – 545 с.
3. Иванова Г.С. Технология программирования: Учебник [Текст] / Г.С. Иванова – М.: Издательство МГТУ им. Н.Э. Баумана, 2002. – 243 с.
4. Мацяшек Л. Анализ и проектирование информационных систем с помощью UML 2.0 [Текст] / Лешек Мацяшек К.: Вильямс, 2008, - 816 с.

# ДОДАТКИ

## ДОДАТОК 1. Варіанти завдань до лабораторної роботи №1

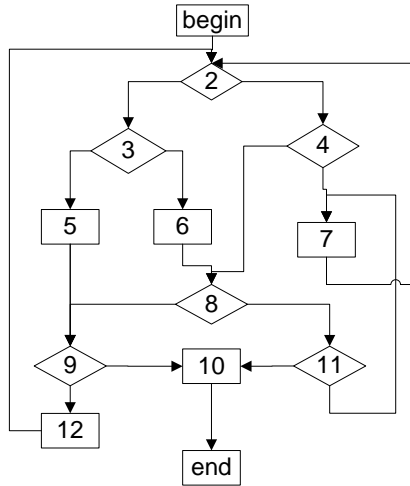




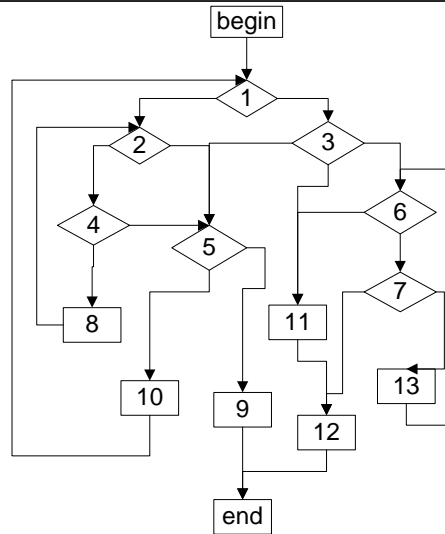


## ДОДАТОК 2. Варіанти завдань до лабораторної роботи №2

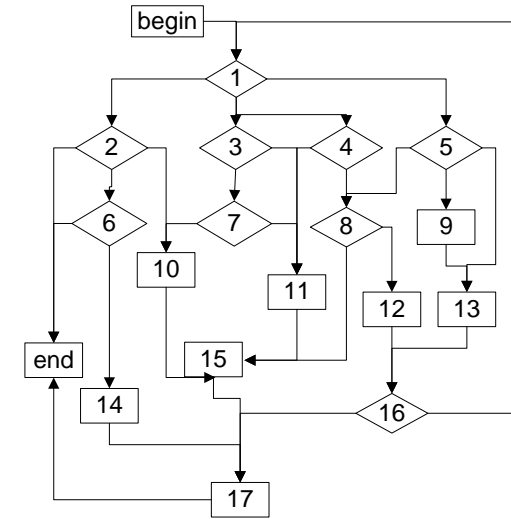
2.1



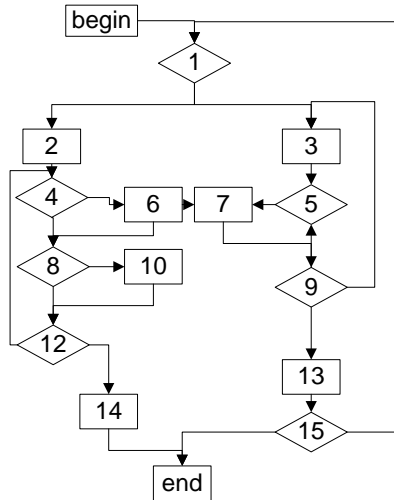
2.2



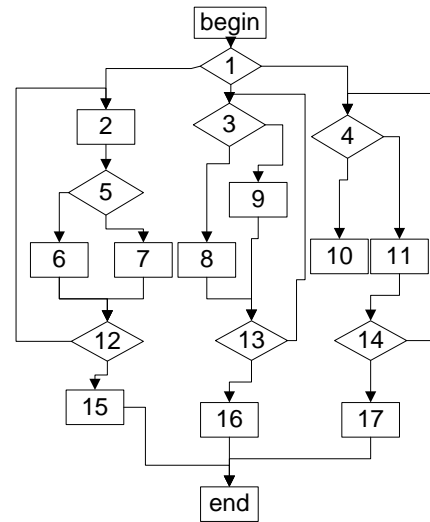
2.3



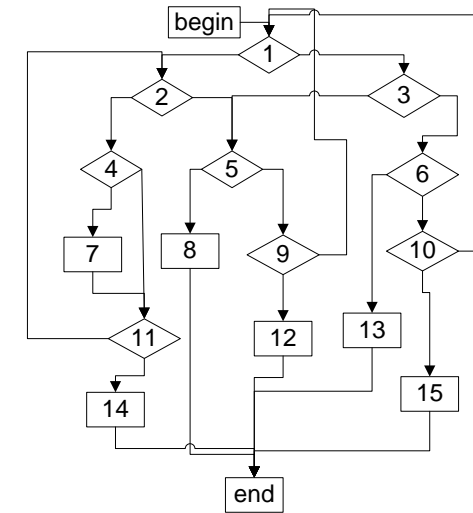
2.4

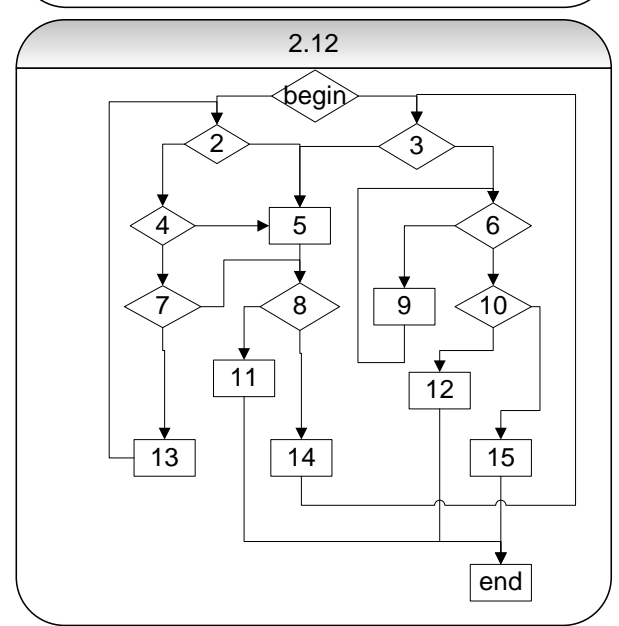
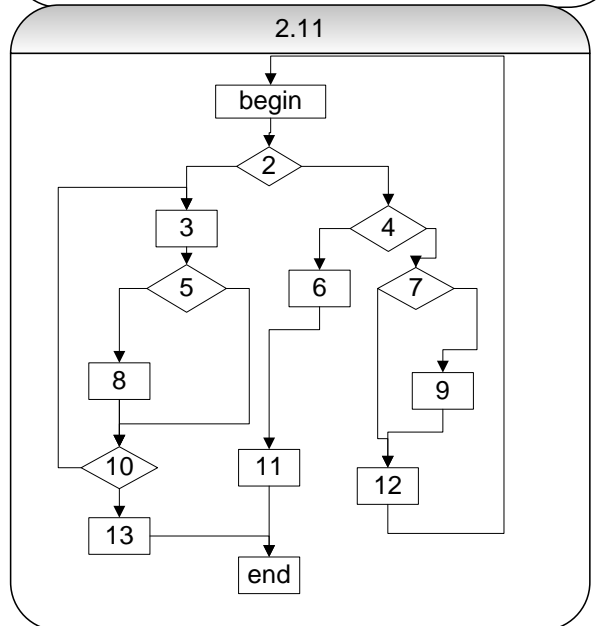
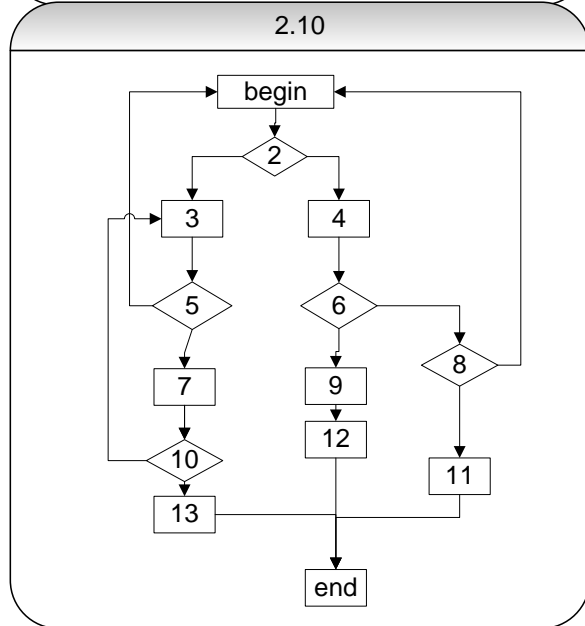
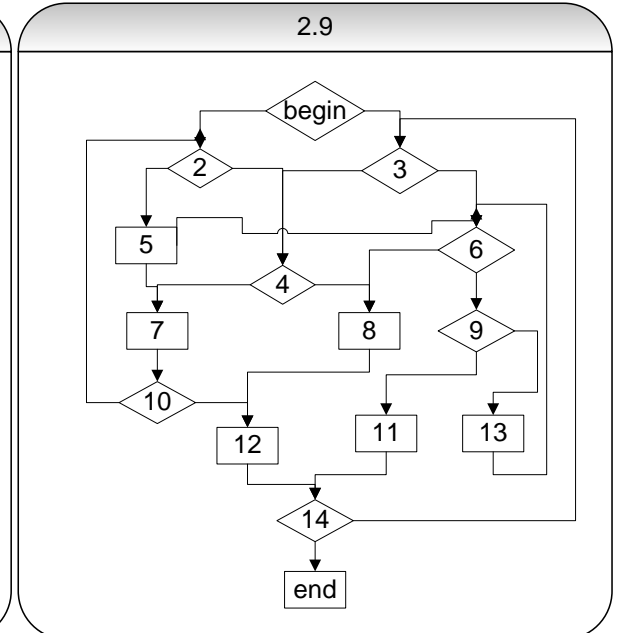
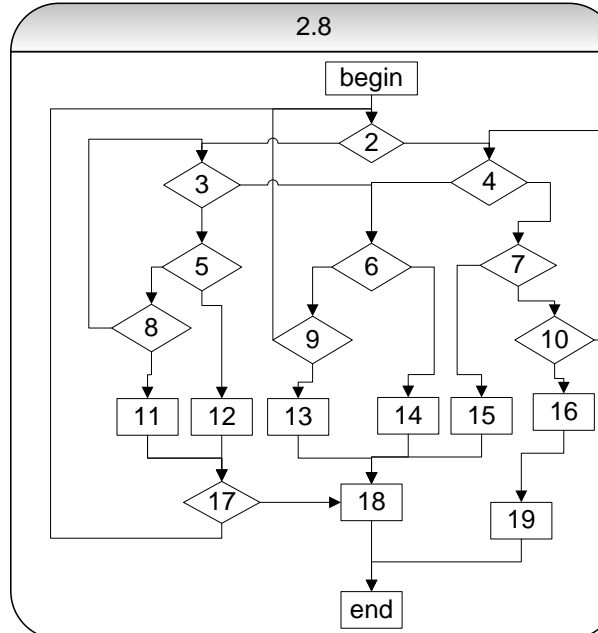
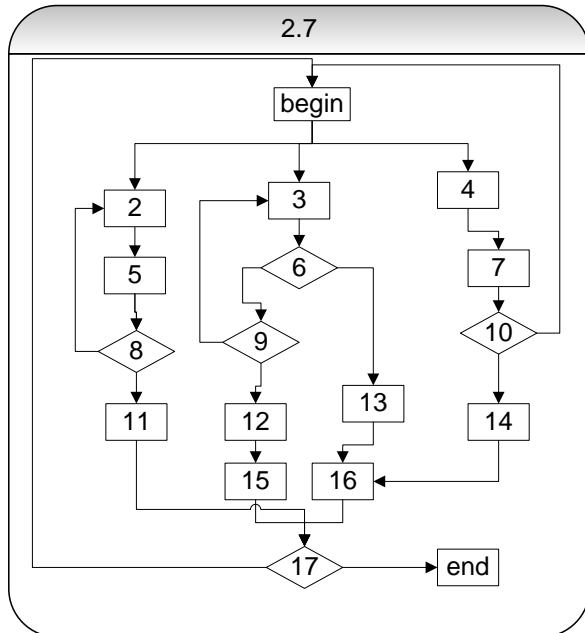


2.5

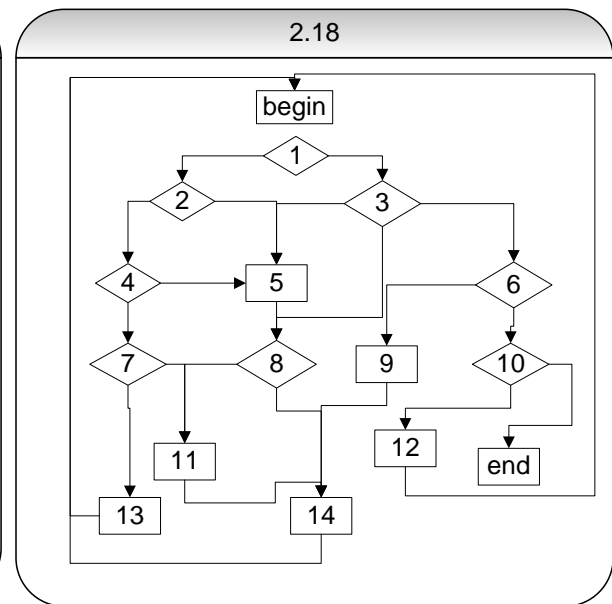
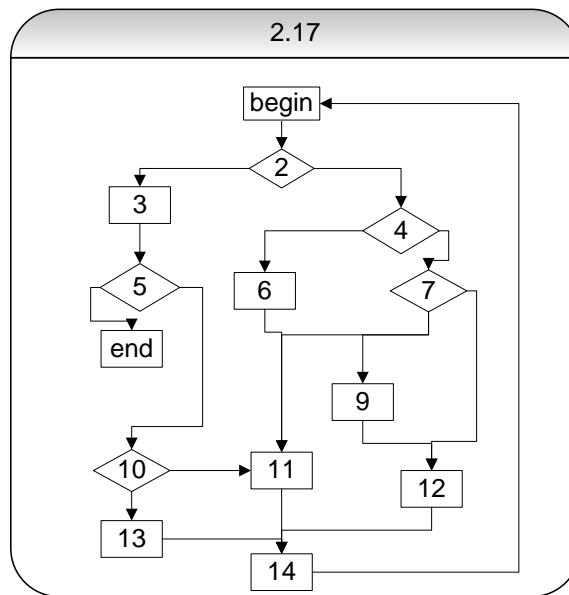
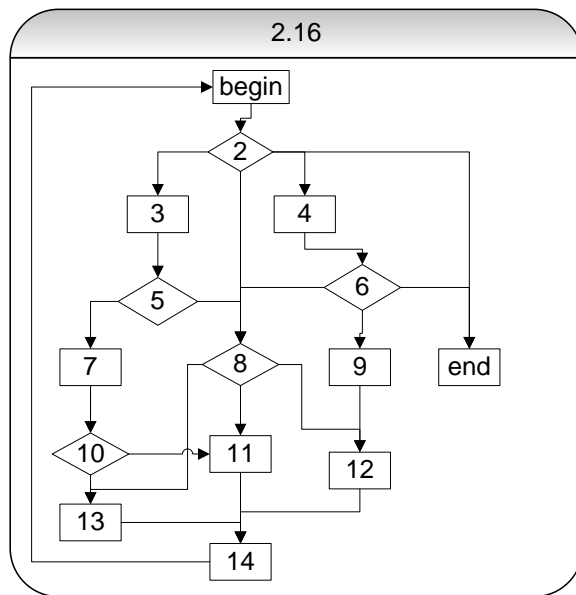
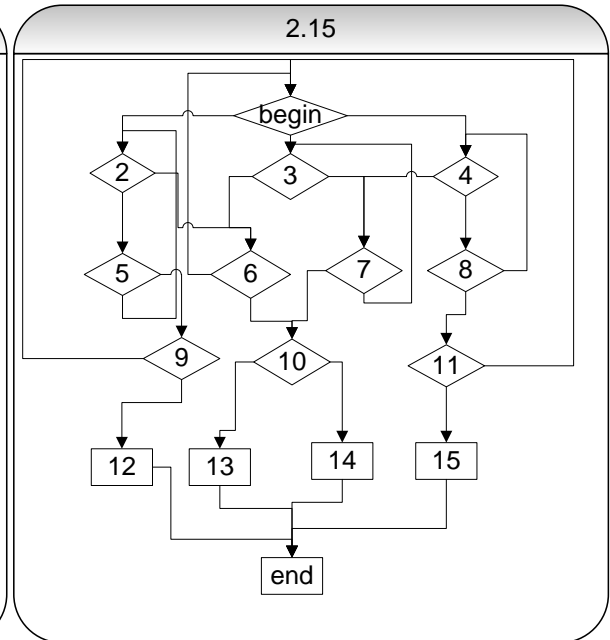
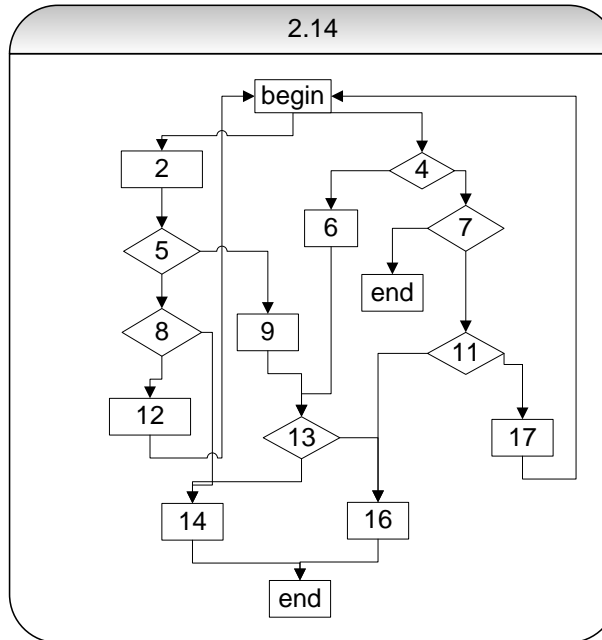
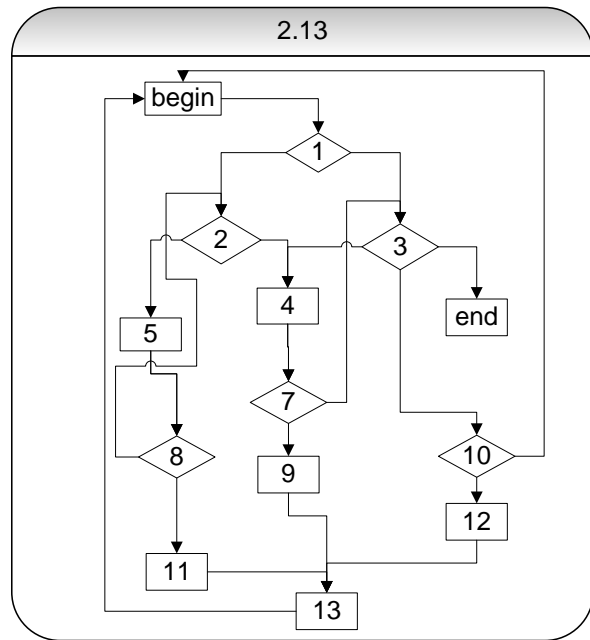


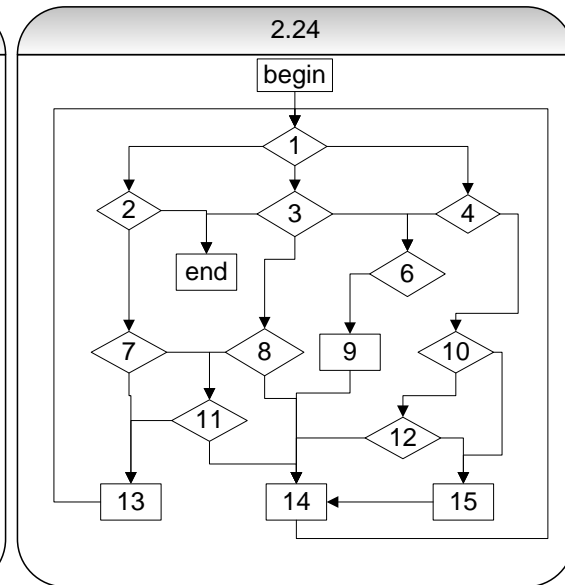
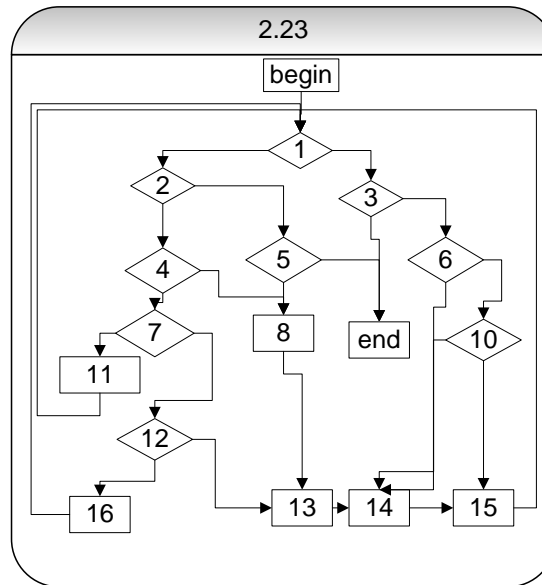
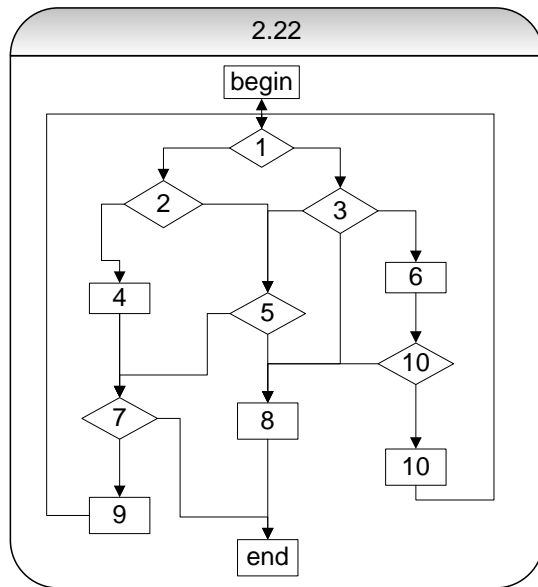
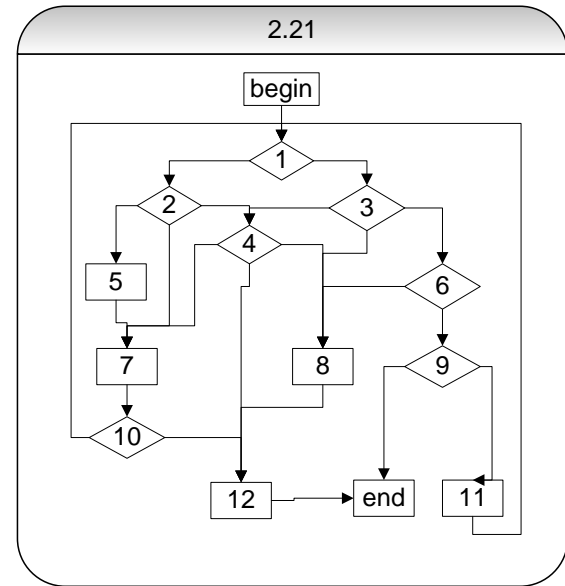
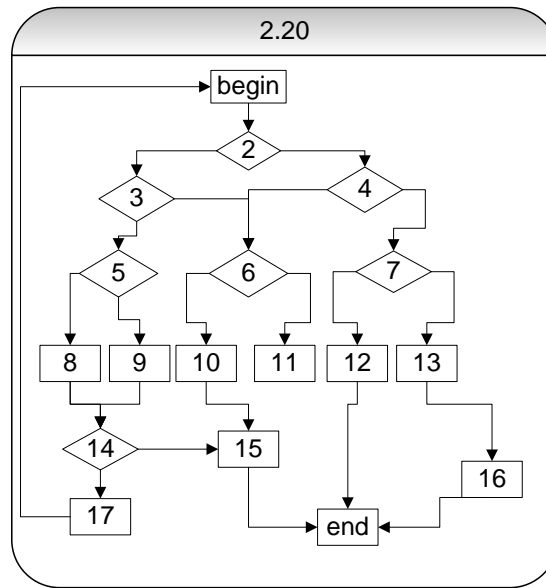
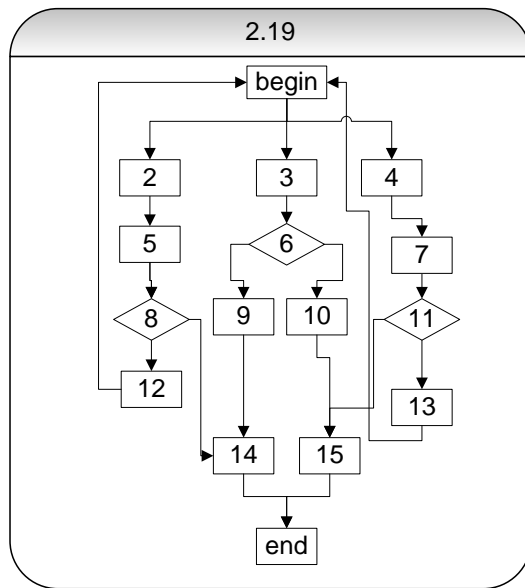
2.6











### **ДОДАТОК 3. Індивідуальні завдання для проектування програмних продуктів (роботи 3-18)**

1. Проектування автоматизованої системи роботи лабораторії молокозаводу.
2. Проектування автоматизованої системи виробничого підрозділу молокозаводу.
3. Проектування автоматизованої системи виробничого підрозділу спиртзаводу.
4. Проектування автоматизованої системи виробничого підрозділу хлібзаводу.
5. Проектування автоматизованої системи виробничого підрозділу пивзаводу.
6. Проектування автоматизованої системи виробничого підрозділу мясокомбінату.
7. Проектування автоматизованої системи виробничого підрозділу макаронного підприємства.
8. Проектування автоматизованої системи контролю якості продукції на молокозаводі.
9. Проектування автоматизованої системи логістики транспортного підрозділу підприємства.
10. Проектування автоматизованої системи відділу збуту готової продукції хлібзаводу.
11. Проектування автоматизованої системи контролю якості продукції на харчовому підприємстві.
12. Проектування автоматизованої системи адміністрування системи дистанційного надання освітніх послуг.
13. Проектування автоматизованої системи алгоритмічної моделі для системи масового обслуговування підприємства.
14. Проектування автоматизованої системи управління запасами сировини підприємства.
15. Проектування автоматизованої системи роботи залізничного вузла.

- 16.Проектування автоматизованої системи електромонтажного підприємства.
- 17.Проектування автоматизованої системи туристичної фірми.
- 18.Проектування автоматизованої системи роботи підприємства по вирощуванню квітів.
- 19.Проектування автоматизованої системи аеропорту.
- 20.Проектування автоматизованої системи косметичного салону.
- 21.Проектування автоматизованої системи меблевого салону.
- 22.Проектування автоматизованої системи морських вантажних перевезень.
- 23.Проектування автоматизованої системи з працевлаштування.
- 24.Проектування автоматизованої системи надання кредитів.
- 25.Проектування автоматизованої системи обліку кадрів підприємства.
- 26.Проектування автоматизованої системи роботи фотостудії.
- 27.Проектування автоматизованої системи роботи букмекерської контори.
- 28.Проектування автоматизованої системи інформаційної системи аптеки.
- 29.Проектування автоматизованої системи організації руху таксі.
- 30.Проектування автоматизованої системи продажу квитків залізничного вокзалу.
- 31.Проектування автоматизованої системи аптекоуправління.
- 32.Проектування автоматизованої системи роботи автозаправного комплексу.
- 33.Проектування автоматизованої системи роботи майстерні з ремонту побутової техніки.
- 34.Проектування автоматизованої системи організації екскурсій.
- 35.Проектування автоматизованої системи по обліку обладнання організації та проведення інвентаризації.
- 36.Проектування автоматизованої системи шлюбної агенції.
- 37.Проектування автоматизованої системи обліку вузів України.
- 38.Проектування автоматизованої системи садового комплексу.

- 39.Проектування автоматизованої системи управління складом готової продукції підприємства.
- 40.Проектування автоматизованої системи роботи авіакаси.
- 41.Проектування автоматизованої системи електронного магазину з продажу побутової техніки.
- 42.Проектування автоматизованої системи фірми з надання кертерингових послуг.
- 43.Проектування автоматизованої системи житлово-комунального господарства (надання послуг тепло-, водо-, газо-, електропостачання, прибирання, тощо).
- 44.Проектування автоматизованої системи міського транспорту.
- 45.Проектування автоматизованої системи магазину іграшок.
- 46.Проектування автоматизованої системи ветеринарної клініки.
- 47.Проектування автоматизованої системи сільськогосподарських угідь.
- 48.Проектування автоматизованої системи тваринницької ферми.
- 49.Проектування автоматизованої системи з продажу програмного забезпечення.
- 50.Проектування автоматизованої системи лізингових послуг.
- 51.Проектування автоматизованої системи роботи охоронної фірми.
- 52.Проектування автоматизованої системи роботи журналу «Спорт».
- 53.Проектування автоматизованої системи телефонного вузла.
- 54.Проектування автоматизованої системи «Вступ».
- 55.Проектування автоматизованої системи роботи деканату.
- 56.Проектування автоматизованої системи станції швидкої допомоги.
- 57.Проектування автоматизованої системи роботи готельного комплексу.
- 58.Проектування автоматизованої системи роботи майстерні з ремонту взуття та одягу.
- 59.Проектування автоматизованої системи туристичної компанії.
- 60.Проектування автоматизованої системи «Картотека Інтерполу».
- 61.Проектування автоматизованої системи біржі праці.

- 62.Проектування автоматизованої системи «Записна книжка».
- 63.Проектування автоматизованої системи роботи ломбарду.
- 64.Проектування автоматизованої системи «Довідник меломана».
- 65.Проектування автоматизованої системи для командира військової частини.
- 66.Проектування автоматизованої системи «Довідник комерційних банків».
- 67.Проектування автоматизованої системи магазину запчастин до автомобіля.
- 68.Проектування автоматизованої системи «Довідник нумізмата».
- 69.Проектування автоматизованої системи розробки тесту знань студентів.
- 70.Проектування автоматизованої системи телеканалу.