МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
Кафедра кібербезпеки та DATA-технологій, факультет №6

МЕТОДИЧНІ МАТЕРІАЛИ

до практичних занять із
навчальної дисципліни

" Об'єктно-орієнтоване програмування "

обов'язкових компонент освітньої програми першого (бакалаврського) рівня вищої освіти

Спеціальність: 125 "Кібербезпека"
(«Безпека інформаційних та комунікаційних систем»)

Харків 2023

ЗАТВЕРДЖЕНО
Науково-методичною радою
Харківського національного
університету внутрішніх справ
Протокол від 30.01.2023 № 1

СХВАЛЕНО
Вченою радою факультету № 6
Протокол від 18.01.2023 № 1

ПОГОДЖЕНО
Секцією Науково-методичної ради
ХНУВС з технічних дисциплін
Протокол від 27.01.2023 № 1

Розглянуто на засіданні кафедри кібербезпеки та DATA-технологій факультету № 6 *(протокол від 13.01.2023 № 1)*

Розробники:

Професор кафедри, к.т.н., доцент; Струков В. М.;

Старший викладач, Цуранов М.В.;

Рецензенти:

1. *Пєвнєв В.Я., д.т.н., доцент, професор кафедри комп'ютерних систем, мереж та кібербезпеки факультету радіоелектроніки, комп'ютерних систем та інфокомунікацій НАУ «ХАІ» ім. М.Є. Жуковського;*
2. *Світличний В.А., к.т.н., доцент, доцент кафедри протидії кіберзлочинності факультету №4 ХНУВС.*

1. Розподіл часу навчальної дисципліни за темами (денна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 4							
Тема № 1. Основні поняття об'єктно-орієнтованого програмування.	**14**	**2**		**2**		**10**	
Тема № 2. Основи уніфікованої мови моделювання UML.	**14**	**2**		**2**		**10**	
Тема № 3. Основи моделювання поведінки системи в UML.	**14**	**2**		**2**		**10**	
Тема № 4. Моделювання класів в UML.	**16**	**2**		**2**	**2**	**10**	
Тема № 5. Архітектура платформи .NET та програмування для .NET Framework	**22**	**4**		**2**	**6**	**10**	
Тема № 6. Синтаксис мови C#. Основні операції, оператори та типи даних C#	**25**	**4**		**2**	**6**	**13**	
Тема № 7. Класи та об'єкти. Перетворення типів даних	**20**	**4**			**6**	**10**	
Тема № 8. Спадкування. Інтерфейси та їх використання.	**25**	**4**			**8**	**13**	
Всього за семестр № 2:	**150**	**24**		**12**	**28**	**86**	**залік**

Розподіл часу навчальної дисципліни за темами
(заочна форма навчання)

Номер та найменування теми	Кількість годин відведених на вивчення навчальної дисципліни						Вид контролю
	Всього	з них:					
		лекції	Семінарські заняття	Практичні заняття	Лабораторні заняття	Самостійна робота	
Семестр 4							
Тема № 1. Основні поняття об'єктно-орієнтованого програмування.	**17**	**2**				**15**	
Тема № 2. Основи уніфікованої мови моделювання UML.	**17**			**2**		**15**	
Тема № 3. Основи моделювання поведінки системи в UML.	**15**					**15**	
Тема № 4. Моделювання класів в UML.	**15**					**15**	
Тема № 5. Архітектура платформи .NET та програмування для .NET Framework	**26**	**2**			**4**	**20**	
Тема № 6. Синтаксис мови C#. Основні операції, оператори та типи даних C#	**20**					**20**	
Тема № 7. Класи та об'єкти. Перетворення типів даних	**20**					**20**	
Тема № 8. Спадкування. Інтерфейси та їх використання.	**20**					**20**	
Всього за семестр № 2:	**150**	**6**	**4**			**140**	**залік**

2. МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИЧНИХ ЗАНЯТЬ

ПРАКТИЧНА РОБОТА №1 ВВЕДЕННЯ В RATIONAL ROSE

Мета роботи: вивчити принципи роботи з Case-пакетом Rational Rose. **Час проведення**: 2 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Призначення і використання елементів екрана інтерфейсу Rose.
2. Призначення і використання видів подання моделі системи в Rational Rose.

Література:

Основна література.

1. Дудзяний І.М. Об'єктно-орієнтоване моделювання програмних систем: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. - 108 с.
2. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл. **Допоміжна література.** 1. Інструментальні програмні засоби розробки ІУС. Методичні вказівки до виконання лабораторних робіт. / уклад.: К.І. Київська–Київ: КНУБА, 2018. – 40 с.

Інформаційні ресурси в Інтернеті

1. Об'єктно-орієнтоване програмування. Частина 1. Основи об'єктно-орієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. [Електронний ресурс]. – Режим доступу

План проведення заняття: I.

Вступ до заняття.

Rational Rose - сімейство об'єктно-орієнтованих Case-засобів, призначених для автоматизації процесів аналізу й проектування ПЗ, для генерації кодів на різних мовах програмування й випуску проектної документації.

Розглянемо призначення елементів екрана інтерфейсу Rose. Запустимо програму командою Пуск/Все программы/IBM Rational/IBM Rational Rose Enterprise Edition.

II. Порядок проведення основної частини заняття.

1. Призначення і використання елементів екрана інтерфейсу Rose.

Браузер (browser) - використається для швидкої навігації по моделі. С допомогою браузера можна додавати до моделі елементи, переглядати існуючі елементи моделі й зв'язку між ними, переміщати й перейменовувати елементи моделі, додавати елементи моделі до діаграми, групувати елементи в пакети, зв'язувати елемент із файлом або адресою Інтернету, працювати з деталізованою специфікацією елемента, відкривати діаграму. Браузер підтримує чотири подання (view): подання варіантів використання, компонентів, розміщення й логічне подання.

Вікно документації (documentation window) - застосовується для роботи з текстовим описом елементів моделі. С його допомогою можна документувати елементи моделі Rose. Наприклад, можна зробити короткий опис кожного діючої особи. При документуванні класу все, що буде написано у вікні документації, з'явиться потім як коментар у генерованому коді. Документація буде виводитися також у звітах, створюваних у середовищі Rose.

Панелі інструментів (toolbars) - застосовуються для швидкого доступу до найпоширеніших команд. Панелі інструментів Rose забезпечують швидкий доступ до найпоширеніших команд. В цьому середовищу існують два типи панелей інструментів: стандартна панель і панель діаграми. Стандартна панель видна завжди, її кнопки відповідають командам, які можуть використатися для роботи з будь-якою діаграмою. Панель діаграми своя для кожного типу діаграм UML. Усі панелі інструментів можуть бути змінено й настроєні користувачем. Для цього використається пункт меню Tools > Options, потім вкладку Toolbars.

Вікно діаграми (diagram window) - використається для перегляду й редагування однієї або декількох діаграм UML. В ньому показано, як виглядає діаграми UML-моделі. При внесенні в елементи діаграми змін Rose автоматично обновить браузер. Аналогічно при внесенні змін в елемент за допомогою браузера Rose автоматично обновить відповідні діаграми. Це допомагає підтримувати модель у несуперечливому стані.

Журнал (log) - застосовується для перегляду помилок і звітів про виконання різних команд. У міру роботи над моделлю певна інформація буде направлятися у вікно журналу. Наприклад, туди містяться повідомлення про помилки, що виникають при генерації коду. Не існує способу закрити журнал зовсім, але його вікно може бути мінімізоване.

На рисунку 1.1 показані різні частини інтерфейсу Rose.

Для навігації по браузеру використовують лінійки прокрутки, а також позначки «+» та «-» праворуч елементів моделі.

Для створення елементу моделі:

1. Клацніть правою кнопкою миші по пакету Use Case View в браузері.

2. Виберіть пункт New > Class у меню, що відкрився.

3. У браузері з'явиться новий елемент з назвою NewClass. Виділивши його, уведіть його ім'я – Class1.

4. Для розміщення елемента моделі на вікні діаграми перетягніть лівою кнопкою миші Class1 з браузера на вікно.

5. Для створення текстового опису елемента моделі Class1 клацніть лівою кнопкою миші по ньому і введіть у вікно документації текст «Учбовий клас».

Контрольні запитання і завдання

1.1 Для чого застосовується вікно документації?

1.2 Для чого застосовується вікно журналу?

1.3 Створити в браузері новий елемент моделі і подати його опис у вікні документації.

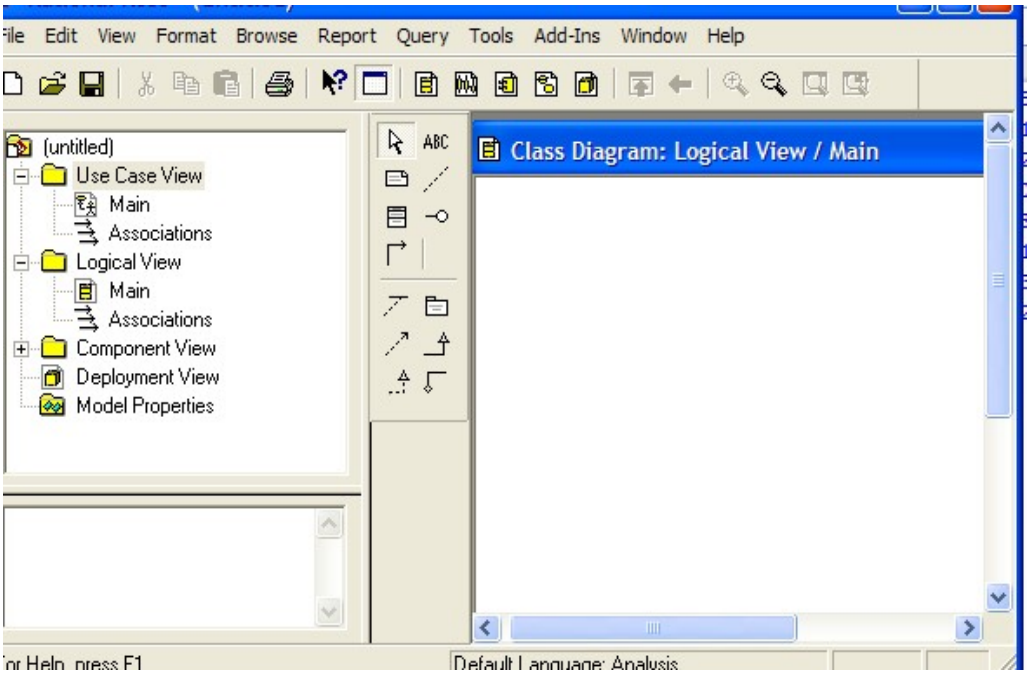

Рисунок 1.1 - Інтерфейс Rose

2. Призначення і використання видів подання моделі системи в Rational Rose.

У моделі Rose підтримуються чотири подання моделі - це подання варіантів використання, логічне подання, подання компонентів і подання розміщення. Кожне з них призначене для своїх цілей.

Подання варіантів використання (Use Case View) містить всіх діючих осіб, всі варіанти використання і їхньої діаграми для конкретної системи. Воно може

також містити деякі діаграми послідовності й кооперативні діаграми. На рисунку 1.2 зображене подання варіантів використання в браузері Rose.

Подання варіантів використання містить:

- діючих осіб;
- варіанти використання;
- документацію по варіантах використання, що описує процеси, що відбуваються в них (потоки подій), включаючи обробку помилок. Ця піктограма відповідає зовнішньому файлу, прикріпленому до моделі Rose;
- діаграми варіантів використання. Звичайно в системи буває кілька таких діаграм, кожна з яких показує підмножину діючих осіб й/або варіантів використання;
- пакети, що є групами варіантів використання й/або діючих осіб.

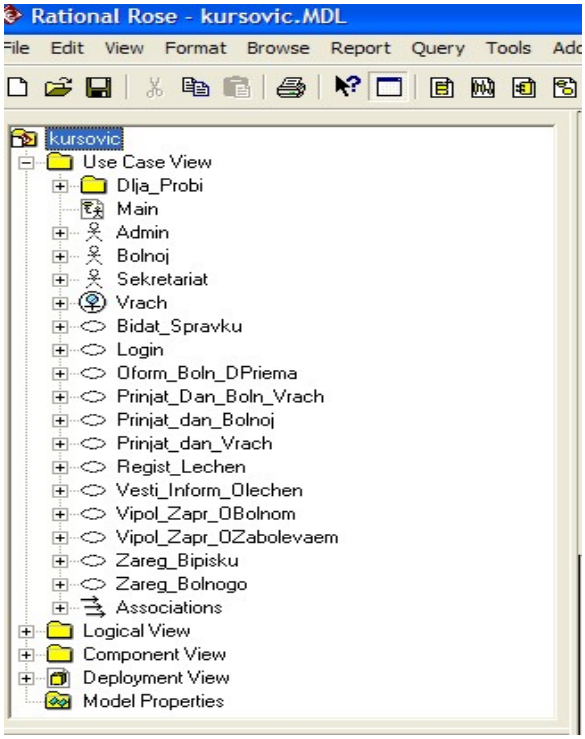

Рисунок 1.2 - Подання варіантів використання

Логічне подання (Logical View) (рисунок 1. 3) показує, як система буде реалізовувати поводження, описане у варіантах використання. Воно дає докладну картину складових частин системи й описує взаємодію цих частин. Логічне подання включає конкретні класи, діаграми класів і діаграми станів. З їхньою допомогою конструюється детальний проект створюваної системи. Логічне подання містить:

- класи;

 діаграми класів. Як правило, для опису системи використається кілька діаграм класів, кожна з яких відображає деяку підмножину всіх класів системи;

 діаграми взаємодії, застосовувані для відображення об'єктів, що беруть участь в одному потоці подій варіанта використання;

 діаграми станів;

 пакети, що є групами взаємозалежних класів.

Подання компонентів (Component View) містить :

 компоненти, що є фізичними модулями коду;

 діаграми компонентів;

 пакети, що є групами зв'язаних компонентів.

Подання розміщення (Deloyment View)- це останнє подання Rose. Воно відповідає фізичному розміщенню системи, що може відрізнятися від її логічної архітектури.

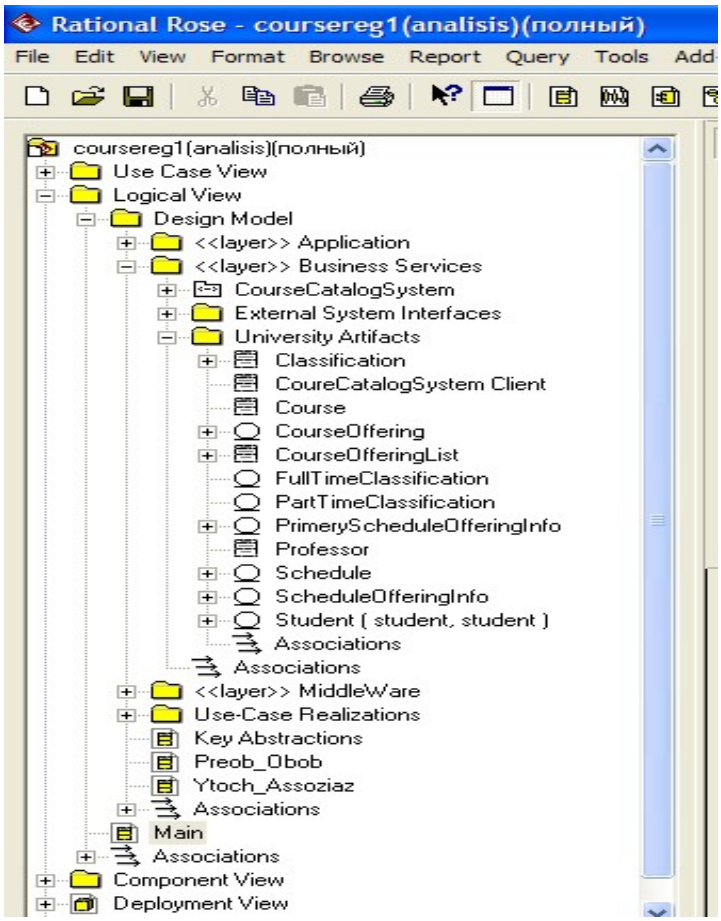

Рисунок 1.3 - Логічне подання системи У

подання розміщення входять:

 процеси, що є потоками (threads), що виконують у відведеній для них області пам'яті;

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

 пристрої, тобто будь-яка апаратури, не здатна обробляти дані (наприклад, термінали вводу-висновку й принтери).

- діаграма розміщення.

В Rose є можливість настроїти діаграми класів так, щоб:

 показувати всі атрибути й операції;

 сховати операції/ сховати атрибути;

 показувати тільки деякі атрибути або операції;

 показувати операції разом з їхніми повними сигнатурами або тільки їхні імена;

 показувати або не показувати видимість атрибутів й опе-рацій;

показувати або не показувати стереотипи атрибутів й операцій.

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

Існують два способи зміни параметрів подання атрибутів на діаграмі. Можна встановити потрібні значення в кожного класу індивідуально. Можна також змінити значення потрібних параметрів за замовчуванням до початку створення діаграми класів. Внесені в такий спосіб зміни вплинуть тільки на знову створювані діаграми.

Для перемикання між нотаціями видимості Rose й UML:

1. У меню моделі виберіть пункт Tools > Options.
2. Перейдіть на вкладку Notation.
3. Для перемикання між нотаціями скористайтеся перемикачем Visibility as Icons. Якщо цей перемикач позначений, буде використатися нотація Rose, у противному випадку - нотація UML. Зміна цього параметра вплине тільки на нові діаграми. Існуючі діаграми класів залишаться колишніми.

Контрольні запитання і завдання

2.1 Що містить подання компонентів?

2.2 Для чого застосовується вікно журналу?

2.3 По черзі розкрити в браузері всі подання моделі й знову їх згорнути

Завдання

1. Виберіть пункт Tools > Options і відкрийте вкладку Toolbars.

Щоб зробити видиму або невидимої стандартну панель інструментів, позначте (або зніміть позначку) контрольний перемикач show standard Too1Bar (або show Diagram ToolBar).

2. Збільште розмір кнопок на панелі інструментів.

Клацніть правою кнопкою миші по необхідній панелі.

3. Виберіть у спливаючому меню пункт Use Large Buttons (Використати більші кнопки), поверніться до нормального розміру кнопок.

4. Настройте панель інструментів.

Клацніть правою кнопкою миші по панелі діаграми Main пакета Use Case View. Виберіть пункт Customize (настроїти) і додайте кілька кнопок.

Щоб додати або видалити кнопки, виберіть відповідну кнопку й потім клацніть мишею по кнопці Add (додати) або Remove (видалити).

III **Заключна частини заняття**: відповіді на контрольні запитання і виконання контрольних завдань.

ПРАКТИЧНА РОБОТА №2 СПЕЦІФІКАЦІЯ ВАРІАНТІВ ВИКОРИСТАННЯ

Мета роботи: вивчити принципи розробки специфікації варіантів використання в Rational Rose. **Час проведення**: 2 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Освоєння шаблону специфікації варіантів використання за стандартом Rational Unified Process.
2. Розробка специфікації варіантів використання.

Література:

Основна література.

1. Дудзяний І.М. Об'єктно-орієнтоване моделювання програмних систем: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. - 108 с.
2. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.

Інформаційні ресурси в Інтернеті

2. Об'єктно-орієнтоване програмування. Частина 1. Основи об'єктно-орієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. [Електронний

ресурс]. – Режим доступу https://ela.kpi.ua/bitstream/123456789/16671/1/OOP_manual.pdf

План проведення заняття: I.

Вступ до заняття.

Специфікації варіантів використання (прецедентів) записують окремо від графічних позначень і доповнюють діаграми прецедентів. UML не визначає точний формат специфікацій прецедентів. Специфікації представляють собою стислий та інформативний опис поведінки системи, що відповідає прецеденту.

Специфікації прецедентів можуть налічувати передумови та постумови. Передумови визначають стан, в якому перебуватиме система до виконання прецеденту. Постумови специфікують стан, в який перейде система після виконання прецеденту. Перед та постумови відображають лише ті аспекти стану, які є важливими для певного прецеденту.

II. Порядок проведення основної частини заняття.

1. Освоєння шаблону спеціфікації варіантів використання за стандартом Rational Unified Process.

Описуючи специфікації прецедентів, використовуватимемо шаблон, запозичений з регламенту Rational Unified Process:

Заголовок прецеденту
1. Короткий опис
2. Потоки подій
2.1. Головний потік подій
2.2. Підлеглі потоки подій
2.3. Альтернативні потоки подій
3. Спеціальні вимоги
4. Передумови
5. Постумови
6. Додаткові зауваження

Приклад документу з описом специфікацій прецедентів “Зареєструватися на курси” для системи реєстрації студентів на навчальні курси.

Прецедент “Зареєструватися на курси”.

1. Короткий опис. Прецедент дає змогу студентові зареєструватися на доступні курси у поточному семестрі. Студент може змінити свій вибір (обновити чи видалити курси), якщо зміна виконується у встановлений час на

початку семестру. Система каталогу курсів надає список усіх доступних курсів у поточному семестрі.

2. Потоки подій.

2.1. Головний потік подій.

Цей прецедент розпочне виконуватися, коли студент захоче зареєструватися на конкретні курси чи змінити свій графік курсів:

1) система запитує необхідну дію (створити графік, обновити графік, видалити графік);

2) коли студент зазначає дію, виконується один з підлеглих потоків (створити, обновити, видалити чи прийняти графік).

Якщо реєстрацію на поточний семестр уже завершено, то виконуватиметься потік 2.3.5.

2.2. Підлеглі потоки подій.

2.2.1. Створити графік:

1) система каталогу курсів виконує пошук доступних курсів і виводить їхній список; якщо система каталогу курсів недоступна, то виконується потік 2.3.4;

2) студент обирає зі списку 4 базові і 2 альтернативні курси; 3) після вибору система створює графік студента;

4) виконується підлеглий потік “Прийняти графік”.

2.2.2. Обновити графік:

1) система виводить поточний графік студента; якщо система не може знайти графіка студента, то виконується потік 2.3.3;

2) система каталогу курсів виконує пошук доступних курсів і виводить їхній список; якщо система каталогу курсів недоступна, то виконується потік 2. 3.4;

3) студент може обновити свій вибір курсів, видаляючи чи додаючи запропоновані курси;

4) після вибору система обновляє графік; 5) виконується підлеглий потік “Прийняти графік”.

2.2.3. Видалити графік:

1) система виводить поточний графік студента; якщо система не може знайти графіка студента, то виконується потік 2.3.3;

2) система запитує у студента підтвердження видалення графіка.

3) студент підтверджує видалення; якщо студент скасує видалення, то виконується потік 2.3.6:

4) система видаляє графік; якщо графік містить курси, на які записався студент, то студент вилучається зі списків цих курсів.

2.2.4. Прийняти графік. Для кожного обраного, проте ще не зафіксованого курсу в графіку система перевіряє виконання таких вимог: проходження студентом попередніх курсів, факт відкриття запропонованого курсу і відсутність конфліктів графіка. Якщо усі вимоги виконано, то система додає студента у список курсу. Інакше виконується потік 2.3.2. Курс фіксується у графіку студента.

2.3. Альтернативні потоки.

2.3.1. Зберегти графік. У будь-який момент студент може замість прийняття графіка зберегти його. Тоді потік “Прийняти графік” заміняється на такий:

1) “незафіксовані” конкретні курси позначаються у графіку як “обрані”; 2) графік зберігається в системі.

2.3.2. Не виконані попередні вимоги, курс заповнений чи простежуються конфлікти графіка. Видається повідомлення про помилку. Студент може або вибрати інший пропонований курс і продовжити виконання прецеденту, або зберегти графік, або скасувати операцію (після чого головний потік почнеться з початку).

2.3.3. Графік не знайдений. Видається повідомлення про помилку. Після того, як студент підтвердить це повідомлення, головний потік почнеться з початку.

2.3.4. Система каталогу курсів недоступна. Видається повідомлення про помилку. Після того, як студент підтвердить це повідомлення, прецедент завершиться.

2.3.5. Реєстрація на курси завершена. Видається відповідне повідомлення і прецедент завершується.

2.3.6. Видалення скасоване. Головний потік почнеться з початку.

3. Спеціальні вимоги. Відсутні.

4. Передумова. Перед виконанням прецеденту студент має увійти в систему.

5. Постумова. Якщо прецедент завершиться успішно, графік студента буде створений, обновлений чи вилучений. Інакше стан системи не зміниться.

Контрольні запитання:

1. Скількі розділів містить шабон специфікації?
2. Що таке предумова? 3. Що таке основний потік подій?

2. Розробка специфікації варіантів використання.

Специфікація варіанту використання може бути створена в окремому файлі (наприклад в текстовому редакторі) і потім прикреплена до варіанту використання.

Для цього необхідно виконати наступні дії (рисунок 2.1)

1. Активізувати правою кнопкою миші потрібний варіант використання і в контекстному меню вибрати команду New.
2. Вибрати команду File
3. Вказати шлях до файлу з описом специфікації варіанту використання.

Специфікація може бути введена безпосередньо в діалог властивостей варіанту використання.

Для цього необхідно виконати наступні дії (рисунок 2.1)

4. Активізувати правою кнопкою миші потрібний варіант використання і в контекстному меню вибрати команду Open Specification.
5. Перейти на закладку General.
6. В полі Documentation ввести текст специфікації варіантів використання

Контрольні запитання і завдання:

1. Як прикрипити файл специфікації до варіанту використання?
2. Скількі існує способів завдання специфікації варіанту використання?

Завдання:

Створіть специфікацію одного з варіантів використання для системи згідно таблиці 1.1, номер варіанта співпадає з номером за списком.

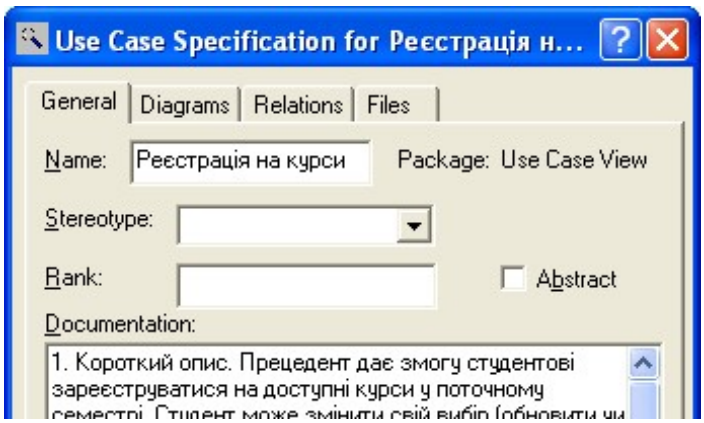

Рисунок 2.1 - Введення тексту специфікації варіанту використання

Таблиця 1.1

Вариант	Завдання
1.	Система керування банкоматом
2.	Система бронювання квитків на потяги
3.	Система замовлень страв в ресторані
4.	Система обліку відвідування занять студентами
5.	Система обліку особистих справ співробітників фірми
6.	Система обліку безробітних в центрі зайнятості
7.	Система обліку книг в бібліотеці
8.	Система бронювання квитків на автобуси
9.	Система тестування знань студентів
10.	Система управління тренувальним центром
11.	Система обліку товарів на складі
12.	Система управління комп’ютерними курсами
13.	Система управління internet - магазином.
14.	Система бронювання квитків на літаки.
15.	Система управління туристичною фірмою.
16.	Система бронювання номерів в готелі
17.	Система автоматизації продаж в магазині
18.	Інформаційна система деканату
19.	Система ведення електронного каталогу товарів
20.	Інформаційна система пункту обміну валюти
21.	Система управління басейном
22.	Система управління автомагазином
23.	Система обліку студентів в гуртожитку
24.	Система бронювання квитків в кінотеатрі
25.	Система замовлень ліків в аптеці

III Заключна частини заняття: відповіді на контрольні запитання і виконання контрольних завдань.

ПРАКТИЧНА РОБОТА №3
ДІАГРАМА СТАНІВ

Мета роботи: побудова діаграми станів в середовище Rational Rose.

Час проведення: 2 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Розробка діаграми станів.
2. Специфікації стану об'єкта та переходу.

Література:

Основна література.

1. Дудзяний І.М. Об'єктно-орієнтоване моделювання програмних систем: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. - 108 с.
2. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.

Інформаційні ресурси в Інтернеті

1. Дацун Н.М. Об'єктно-орієнтоване програмування: навчальний посібник для [Електронний ресурс]. – Режим доступу http://ea.donntu.edu.ua/bitstream/123456789/27021/1/%d0%9e%d0%9e%d0%9f%202014.pdf

План проведення заняття: І.

Вступ до заняття.

Діаграма станів (Statechart diagram) визначає усі можливі стани (state), у яких може знаходитися конкретний об'єкт під час свого існування, а також процес зміни станів цього об'єкта у результаті настання деяких подій (event). Немає потреби створювати діаграми станів для кожного класу, їх використовують тільки у складних випадках. Однак, якщо об'єкт деякого класу може існувати у декількох станах і у кожному з них по-різному поводиться, то для нього така діаграма може бути дуже корисною.

ІІ. Порядок проведення основної частини заняття.

1. Розробка діаграми станів .

На діаграмі виокремлюють два спеціальних стани – початковий (start) і кінцевий (stop). Початковий стан виокремлюють чорною крапкою, він відповідає тільки що створеному стану об’єкта.

Кінцевий стан виокремлюють чорною крапкою в білому крузі, він відповідає стану об’єкта безпосередньо перед його знищенням. На діаграмі станів може бути тільки один початковий стан і декілька кінцевих станів (або кінцевих станів може не бути взагалі).

Усі інші стани зображають прямокутником із заокругленими краями. Кожен стан має дві частини. У верхній частині відображають назву стану. Назва стану може бути порожньою (анонімний стан). Нижню частину стану призначено для відображення внутрішніх дій (actions), які виконуються у відповідь на визначені події, що виникають у цьому стані. Наприклад, діаграма станів об’єкта класу Account (Рахунок) може набувати вигляду як на рис. 3.1.

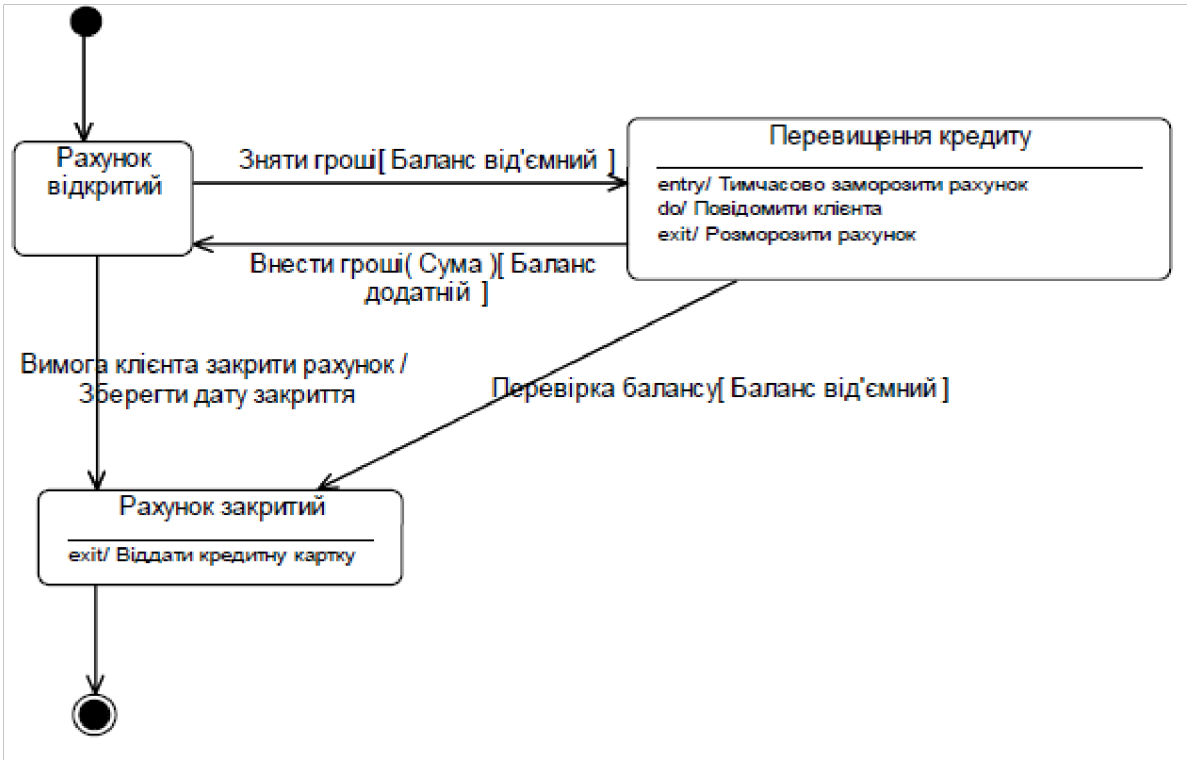

Рис. 3.1. Діаграма станів об’єкта класу Account (рахунок).

Для створення діаграми необхідно:

1. Клацнути правою кнопкою миші на класі в браузері, для об’єктів якого створюється діаграма в логічному поданні.
2. Вибрати в контекстному меню команду New\ Statechart Diagram.
3. Ввести назву діаграми.

4. Двічі клацнути лівою кнопкою миші на елементі діаграми і активізувати вікно діаграми Statechart Diagram.

5. Використовуючи панель інструментів створити на діаграмі стани (state) об’єктів класу, початковий (start) і кінцевий (stop) стани, поєднати їх переходами (State Transition).

Контрольні запитання і завдання:

1. Які основні елементи діаграми станів?
2. Як створити діаграму станів?

Завдання: визначити один з об’єктів з динамічною поведінкою для свого варіанта інформаційної системи з таблиці 1.1 і створити діаграму станів для цього об’єкта.

2. Специфікації стану об’єкта та переходу.

Специфікації стану об’єкта відображаються всередені прямокутника, що позначає на діаграмі цей об’єкт.

Для стану об’єкта можна задати вхідну подію (On Entry), вихідну подію (On Exit), довільну подію (On Event НазваПодії) і діяльність (Do). У відповідь на виникнення події чи під час реалізації діяльності можна задати просту дію (Action) чи надіслати повідомлення/активізувати подію (Send Event) іншому об’єкту.

Опис простих дій має такий формат:

ТИП / дія, де ТИП має одне із

значень:

- entry;
- exit; - do;
- event НазваПодії [(СписокПараметрів)] [умова].

Опис активізації події має такий формат:

ТИП / ^Об'єкт.НазваПодії[(СписокПараметрів)]

Для реалізації реакції на подію стану об’єкта чи реалізації діяльності стану об’єкта необхідно виконати наступні кроки:

1. Активізувати правою кнопкою миші на стані об’єкта контекстне меню і вибрати пункт Open Specification.

2. На вкладці Actions діалогу Open Specification активізувати правою кнопкою миші контекстне меню на області розміщення дій і вибрати пункт Insert.

3. Виділити новий рядок, що зявится в діалозі, активізувати правою кнопкою миші контекстне меню і вибрати пункт Specification.

4. В діалозі Actions Specification for в полі When уточнити подію (On Entry, On Exit, On Event) чи обрати діяльність (Do).Для On Event відкриваються поля введення: Event (назви події); Arguments (списку параметрів); Condition (умови) 5. В полі Type треба обрати Action (чи Send Event). Для Send Event(повідомлення) відкриваються поля введення: Name (назви події); Send Arguments (списку параметрів); Send target (назва об’єкта). Треба описати дію чи сформувати повідомлення і підтвердити через ОК.

Специфікації переходу.

Переходом (Transition) називають переміщення з одного стану об’єкта в інший внаслідок виникнення певної події (Event). На діаграмі перехід зображають стрілкою, що починається на первісному стані і закінчується наступним станом. Переходи можуть бути рефлексивними (Transition To Self). Найпростіший перехід має таку специфікацію (відображається поруч зі стрілкою):

НазваПодії [(СписокПараметрів)] [[умова]]

Найпростіший перехід зі стану 1 у стан 2 ілюструє, що об’єкт, який перебуває у стані 1, перейде у стан 2 тоді, коли відбудеться визначена для переходу подія і справдиться задана умова (на діаграмі обмежується квадратними дужками).

Подія, що визначається НазвоюПодії, може задаватися назвою операції або звичайною фразою (як на рис. 3.1). Наприклад, замість фрази “Вимога клієнта закрити рахунок” можна задати назву операції RequestClosure.

У подій можуть бути параметри. Наприклад, подія “Внести гроші” має параметр Сума.

Під час виконання переходу можна задати просту дію (Action) чи надіслати повідомлення або активізувати подію (Send Event) іншому об’єктові. Опис переходу матиме такий формат:

НазваПодії [(СписокПараметрів)] [[умова]] / дія

НазваПодії [(СписокПараметрів)] [[умова]] / ^Об'єкт.НазваПодії[(Список Параметрів)]

У будь-якому випадку для опису специфікації переходу необхідно виконати такі дії:

1. Активізувати правою кнопкою миші на переході контекстне меню і вибрати пункт Open Specification.

2. На вкладці General діалогу Open Specification ввести в поле Event НазвуПодії; в поле Arguments ввести СписокПараметрів.

3. Перейти на вкладку Detail і ввести в поле Guard Condition умову переходу.

4. За потреби описати дію у полі Action чи сформувати повідомлення у полях Send Event (НазваПодії), Send Arguments (СписокПараметрів); Send target (Об'єкт) і пітвердити дані кнопкою ОК.

Контрольні запитання і завдання:

1. Який формат має опис специфікації стану об’єкта для простих дій? 2. Як задати в середовище Rational Rose специфікацію переходу?

Завдання: для визначеного раніше об’єкта з динамічною поведінкою свого варіанта інформаційної системи з таблиці 1.1 задати специфікації станів об’єкта та переходів.

III Заключна частини заняття: відповіді на контрольні запитання і виконання контрольних завдань.

ПРАКТИЧНА РОБОТА №4

Створення класів в мові C#

Мета роботи: знайомство з середовищем розробки Visual Studio на прикладі побудови простої форми програми C# - консольного застосування . **Час проведення**: 4 години. **Місце проведення:** комп’ютерний клас.

Навчальні питання:

1. Вивчення створення проектів в середовищі Visual Studio .
2. Освоєння методів роботи з консоллю.

Література:

Основна література.

1. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.
2. H. Schildt, “C# 4.0 The Complete Reference McGraw Hill Education; 1st edition” – 984 p., 2017.
3. J. Sharp, “Microsoft Visual C# Step by Step”, Pearson Education – 878p., 2018.

Допоміжна література.

1. A. Troelsen, “Pro C# 7 With .NET and .NET Core” - Apress, — 1372 p., 2017.

Інформаційні ресурси в Інтернеті

1. C# Reference, сайт розробників MSDN. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/languagereference/index 2. Об'єктно-орієнтоване програмування. Частина 1. Основи об'єктно-орієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. [Електронний ресурс]. – Режим доступу https://ela.kpi.ua/bitstream/123456789/16671/1/OOP_manual.pdf

План проведення заняття: I.

Вступ до заняття.

Середовище розробки *Visual Studio 2010* - інтегроване середовище розробки програмного забезпечення, продукт компанії Майкрософт.

.NET Framework (вимовляється Dot Net) - програмна платформа, випущена компанією Microsoft в 2002 році.

У *.Net Framework* можна виділити два основні компоненти:

- статичний - *FCL* (Framework Class Library) - бібліотеку класів;
- динамічний - *CLR* (Common Language Runtime) – загальномовне виконавче середовище.

Common Language Runtime (CLR) - виртуальная машина, яка інтерпретує і виконує код програм, написаних на *C#, VisualBasic.NET, VisualJ#* і т. п., компонент пакету *Microsoft .NET Framework.*

FCL (Framework Class Library) - бібліотека класів, важливою частиною якої стали класи, задаючі примітивні типи - ті типи, які вважаються вбудованими в мову програмування.

Простори імен є способом організації різних класів, присутніх в програмах C#, які визначають для класів унікальні повні імена. Основним простором імен бібліотеки *FCL* є простір *System*, що містить як класи, так і інші вкладені простори імен. Наприклад, примітивний тип *Int32* безпосередньо вкладений в простір імен *System* і його повне ім'я, що включає ім'я простору - *System.Int32*. *Visual Studio .Net* для мови *C#*, *Visual Basic* і *J#* пропонує 12 можливих видів проектів. На початковому етапі, розглянемо консольні застосування.

II. Порядок проведення основної частини заняття.

1. Вивчення створення проектів в середовищі Visual Studio .

Розберемося, як створюються проекти і що вони собою представляють. Розглянемо поняття:

- рішення (*solution*);

• проект (project);
• простір імен (*namespace*).
Створемо рішення, яке містить, єдиний проект — *ConsoleApplication1:*

```
using System;
namespace ConsoleApplication1          //простір імен
{
   class Program                       //клас
   {
      static void Main (string[] args) //точка входу
      {
      }
   }
}
```

Клас проекту, занурений в простір імен, що має за умовчанням те ж ім'я, що і рішення, і проект. Файл із стандартним ім'ям *Program* є побудованим за умовчанням класом, який задає точку входу - процедуру *Main*.

Програма на C# складається з одного або декількох файлів. Кожен файл може містити один або декілька просторів імен. Кожний простір імен може містити вкладені простори імен і типи, такі як класи, структури, інтерфейси, перерахування і делегати- функціональні типи. При створенні нового проекту C# у середовищі *Visual Studio* вибирається один з можливих типів проектів. На підставі зробленого вибору автоматично створюється каркас проекту.

При створенні нового проекту автоматично створюється складна вкладена структура - рішення, що містить проект, який містить простір імен, що містить клас, який містить точку входу.

Простору імен може передувати одна або декілька пропозицій *using*, де після ключового слова слідує назва простору імен - з бібліотеки *FCL* або з проектів, пов'язаних з поточним проектом. В даному випадку задається простір імен *System* - основний простір імен бібліотеки *FCL*.

Підключення просторів імен полегшує запис при використанні класів, що входять в простір, оскільки в цьому випадку не потрібно кожного разу задавати повне ім'я класу з вказанням імені простору, що містить цей клас. Наприклад, замість *System.Console.ReadLine ()*, вказуючи в *using* простір імен *System* можна записати:

```
Console.ReadLine();
```

В прикладі демонструється виклик методів *WriteLine* класу *Consol* для виведення текстового рядка:

```
using System;
  namespace ConsoleApplication1
{
class Class1
 {
  static void Main()       {
Console.WriteLine("Введите Ваше имя"); string
name;
name = Console.ReadLine(); if
(name=="")
System.Console.WriteLine ("Здравствуй, мир!");  else
Console.WriteLine("Здравствуй, " + name + "!");
    }
  }
}
```

Виконайте запропонований приклад в програмному середовищі.

Контрольні питання

1. Що є простір імен?
2. Який простір імен використовується в цьому застосуванні?
3. Який простір імен створюється?
4. Які класи належать використовуваному простору імен?
5. Які класи належать створеному простору імен?
6. Що є точка входу проекту?

2. Освоєння методів роботи з консоллю.

Для роботи з консоллю в .NET використовується клас ***Console***. Переваги цього класу полягають в двох аспектах: усі його методи є статичними, так що не треба створювати для використання його екземпляр. Він об'єднує в собі введення, виведення і виведення помилок. За умовчанням введення/виведення робиться на стандартну консоль.

Для роботи з консоллю зазвичай використовуються чотири методи: ***Read, ReadLine, Write і WriteLine***, з них перших два – для введення даних, останні – для виведення даних.

Метод Read

Метод *Read* читає символ з потоку введення. Він повертає значення типу int, що дорівнює коду прочитаного символу, або – 1 (мінус один), якщо нічого прочитано не було. Наведемо приклад програми :

```
do
 {
int i = Console.Read();
if (i != -1)
Console.WriteLine("{0} {1}", (char)i, i) ;
else break;
 } while (true);
```

Ця програма показує на екрані введені символи і їх коди.

Метод ReadLine

Метод *ReadLine* читає з потоку введення рядок тексту (вона завершується символом перекладу рядка або повернення каретки). Метод повертає об'єкт типу *string* або *null*, якщо введення здійснити не вдалося:

```
do
  {
string s = Console.ReadLine(); if
(s != null)
Console.WriteLine("Введенная строка: " + s) ;
else break;
  } while (true);
```

Методы Write и WriteLine

Метод *Write* виводить на екран значення переданої йому змінної.

Він визначений для усіх базових типів і підтримує форматовані рядки. Таким чином, можна або викликати *Write* з вказаним значенням в якості параметра:

```
Console.Write (I);
Console.Write("Hello!");
```

або передати рядок форматування і список значень. У рядку форматування застосовується безліч модифікаторів. Замість {n} підставляється *n*-й вхідний параметр (нумерация починається з нуля) :

```
Console.Write("Привіт, {0}", Name);
```

Метод *WriteLine* відрізняється від *Write* тільки тим, що виводить символ закінчення рядка у кінці.

Структура програми складається з двох частин : розділу «*using*» та «простору імен». Код програми представлений нижче:

```
using System;

namespace TestConsole
{
class Classl
{
static void Main(string[] args)
 {

}
}

class Class2 {
//method1
//method2
...
}

class Class3 {
//method1
//method2
...
}
}
```

(Слід зауважити, що файли C# мають розширення «.cs»)

Ця програма нічого доки не робить, але вона робоча і готова до запуску. Пояснимо деякі моменти цієї програми:

```
using System; using
System.Collections;
using System.ComponentModei;
```

Цей код визначає, які простори імен використовуватимуться в цьому файлі. Кожен представлений рядок складається з двох частин : ключового слова *Using* і визначеного простору імен. Далі слідує оголошення власного простору імен:

namespace TestConsole

який має основний клас *Class1*, що містить точку входу проекту або метод *Main( )*,а також може містити і інші допоміжні класи *Class2,Class3*.
Створення допоміжних класів дозволяє зробити код зрозумілішим.

При виконанні основного завдання лабораторної роботи необхідно створити клас, окрім класу *Program*, який містить метод *Main( )*. Новий клас міститиме метод, що виконує обчислення факторіалу числа. Метод приймає як параметр число, факторіал якого необхідно обчислити, і повертає результат в точку виклику методу.

Необхідно також створити допоміжний клас, який містить метод обчислення відстані між двома точками на площині. Цей метод, який викликається з допоміжного классу, не має бути прив'язаний до об'єкту, тому метод має бути статичним і відкритим, що дає можливість викликати його поза класом, до якого він належить. Приклад опису такого методу:

```
public static int MyMethod()
{ … }
```

Тут *public* означає відкритий метод, а *static* – статичний метод, тобто не прив'язаний до об'єкту.

Контрольні питання і завдання

1. Що таке статичні методи?
2. Як викликаються статичні методи?
3. Які методи використовуються для прочитування даних, що вводяться з клавіатури?
4. Які методи використовуються в програмі для виведення даних? До якої категорії методів вони належать?
5. Що означає поняття «відкритий метод»?
6. Поняття значення, що повертається, для методу.
7. Який оператор повертає значення?

Завдання 1.

Створити допоміжний клас *Function*. Класс повинен містити метод *Factorial*, в якому здійснюється перевірка діапазону введення (число має бути в

межах 1 до 9) і алгоритм обчислення факторіалу числа. Цей метод викликається в точці входу проекту, отримує початкові дані, повертає значення в точку виклику. Вивести отримані результати.

Завдання 2

Створити допоміжний класс, окрім классу *Program*. Цей клас повинен містити метод, в якому знаходиться алгоритм обчислення відстані між двома точками на площині. Координати точок ввести з командного рядка. Метод викликається в точці входу проекту, отримує початкові дані, повертає значення в точку виклику. Вивести отримані результати.

III Заключна частини заняття: відповіді на контрольні запитання і виконання контрольних завдань.

ПРАКТИЧНА РОБОТА №5 ОБРОБКА МАСИВІВ В МОВІ C#

Мета роботи: отримати знання і уміння щодо оголошення, ініціалізація і обробка масивів

Час проведення: 2 години. **Місце проведення:** комп'ютерний клас.

Навчальні питання:

1. Освоєння методів роботи з одновимірнми масивами.

Література:

Основна література.

1. H. Schildt, "C# 4.0 The Complete Reference McGraw Hill Education; 1st edition" – 984 p., 2017.
2. J. Sharp, "Microsoft Visual C# Step by Step", Pearson Education – 878p., 2018.

Допоміжна література.

1. A. Troelsen, "Pro C# 7 With .NET and .NET Core" - Apress, — 1372 p., 2017.
2. J. Albahari, B. Albahari, "C# 7.0 in a Nutshell", O'Reilly Media— 1090 p., 2017.

Інформаційні ресурси в Інтернеті

1. C# Reference, сайт розробників MSDN. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/languagereference/index 2. Об'єктно-орієнтоване програмування. Частина 1. Основи об'єктно-орієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. [Електронний ресурс]. – Режим доступу https://ela.kpi.ua/bitstream/123456789/16671/1/OOP_manual.pdf

План проведення заняття: I.

Вступ до заняття.

Поняття масиву застосовується при вирішенні багатьох практично важливих завдань обробки даних. Їх використання дозволяє компактно і доцільно організувати програмний модуль і ефективно вирішити поставлену задачу.

II. Порядок проведення основної частини заняття.

1. Освоєння методів роботи з одновимірнми масивами.

Одновимірний масив об'єктів оголошується таким чином:

```
type[] arrayName;
```

Елементи в масиві ініціалізуються, як показано нижче:

```
int[] array = new int[5];
```

Значення за умовчанням числових елементів масиву задане рівним нулю, але значення можна ініціалізувати при створенні масиву таким чином:

```
int[] array1 = new int[] { 1, 3, 5, 7, 9 }; або
так

int[] array2 ={1, 3, 5, 7, 9};
```

Індексація масивів розпочинається з нуля, тому номер першого елементу масиву дорівнює нулю:

```
string[] days ={"Sun", "Mon", "Tue", "Wed", "Thr", "Fri", "Sat"};

System.Console.WriteLine (days[0]);

//  Результат: "Sun"
```

Зазвичай, при роботі з масивами, використовується оператор циклу з параметром ***for (....)***, для обробки усіх елементів масиву, наприклад,

```
for (int i=0; i<7; i++)

System.Console.WriteLine (days[i]);
```

При роботі з масивами можливе використання оператора ***foreach.***
Оператор *foreach* часто використовується для доступу до кожного елементу, що зберігається в масиві:

```
int[] numbers ={ 4, 5, 6, 1, 2, 3, - 2, - 1, 0 };

foreach (int i in numbers)
{
   System.Console.Write ("{0} ", i);
}
```

Методи, які викликаються з допоміжного класу для обробки масивів мають бути відкритими і статичними.

Контрольні питання і завдання 1.

Що таке массив?

2. Які існують способи створення масиву в програмі?
3. Який тип даних массиву у створеній лабораторній роботі?
4. Які методи виконують обробку массиву в програмі? До якої категорії методів вони належать?
5. Які стандартні методи мови C# використовуються в програмі? До якого класу вони відносяться?

Завдання

Вибрати завдання згідно варианту (див.нижче). У застосуванні створити допоміжний клас. Клас повинен містити методи, які виконують обробку масиву згідно із завданням. Методи викликаються в точці входу проекту. Вивести отримані результати. Звіт повинен містити постановку задачі, надруковану програму, результат виконання та висновки.

Варіанти завдань

Варіант 1

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- суму негативних елементів масиву;
- добуток елементів масиву, розташованих між максимальним і мінімальним елементами;
- упорядкувати елементи масиву за збільшенням.

Варіант 2

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- суму позитивних елементів масиву;
- добуток елементів масиву, розташованих між максимальним по модулю і мінімальним по модулю елементами;
- упорядкувати елементи масиву по убуванню.

Варіант 3

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- добуток елементів масиву з парними номерами;

• суму елементів масиву, розташованих між першим і останнім нульовими елементами;

• перетворити масив так, щоб спочатку розташовувалися усі позитивні елементи, а потім - усі негативні (елементи, рівні 0, вважати позитивними).

Варіант 4

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- суму елементів масиву з непарними номерами;
- суму елементів масиву, розташованих між першим і останнім негативними елементами;
- упорядкувати елементи масиву за збільшенням модулів елементів.

Варіант 5

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• максимальний елемент масиву;

• суму елементів масиву, розташованих до останнього позитивного елементу;

• упорядкувати за збільшенням позитивні елементи масиву і помістити їх в початок масиву, зрушивши управо усі інші елементи.

Варіант 6

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• мінімальний елемент масиву;

• суму елементів масиву, розташованих між першим і останнім позитивними елементами;

• перетворити масив так, щоб спочатку розташовувалися усі елементи, рівні нулю, а потім - усі інші.

Варіант 7

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• номер максимального елементу масиву;

• добуток елементів масиву, розташованих між першим і другим нульовими елементами;

• перетворити масив так, щоб в першій його половині розташовувалися елементи, що стояли в непарних позиціях, а в другій половині - елементи, що стояли в парних позиціях.

Варіант 8

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• номер мінімального елементу масиву;

• суму елементів масиву, розташованих між першим і другим негативними елементами;

• перетворити масив так, щоб спочатку розташовувалися усі елементи, модуль яких не перевищує 1, а потім - усі інші.

Варіант 9

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- максимальний по модулю елемент масиву;
- суму елементів масиву, розташованих між першим і другим позитивними елементами; перетворити масив так, щоб елементи, рівні нулю, розташовувалися після усіх інших.

Варіант 10

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- мінімальний по модулю елемент масиву;
- суму модулів елементів масиву, розташованих після першого елементу, рівного нулю;
- перетворити масив так, щоб в першій його половині розташовувалися елементи, що стояли в парних позиціях, а в другій половині - елементи, що стояли в непарних позиціях.

Варіант 11

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- номер мінімального по модулю елементу масиву;
- суму модулів елементів масиву, розташованих після першого негативного елементу;
- упорядкувати за збільшенням негативні елементи масиву і помістити їх в початок масиву, зрушивши управо усі інші елементи.

Варіант 12

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- номер максимального по модулю елементу масиву;
- суму елементів масиву, розташованих після першого позитивного елементу;
- перетворити масив так, щоб спочатку розташовувалися усі елементи, ціла частина яких лежить в інтервалі [a, b], а потім - усі інші.

Варіант 13

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- кількість елементів масиву, що лежать в діапазоні від A до B;
- суму елементів масиву, розташованих після максимального елементу;
- упорядкувати елементи масиву по убуванню модулів елементів.

Варіант 14

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

- кількість елементів масиву більших, чим 3;
- добуток елементів масиву, розташованих після максимального по модулю елементу;

• перетворити масив так, щоб спочатку розташовувалися усі негативні елементи, а потім - усі позитивні (елементи, рівні 0, вважати позитивними).

Варіант 15

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• кількість негативних елементів масиву;

• суму модулів елементів масиву, розташованих після мінімального по модулю елементу;

• замінити усі негативні елементи масиву їх квадратами і упорядкувати елементи масиву за збільшенням.

Варіант 16

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• кількість позитивних елементів масиву;

• суму елементів масиву, розташованих після останнього елементу, рівного нулю;

• перетворити масив так, щоб спочатку розташовувалися усі елементи, ціла частина яких не перевищує 1, а потім - усі інші.

Варіант 17

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• кількість елементів масиву, більших 5;

• суму цілих частин елементів масиву, розташованих після останнього негативного елементу;

• перетворити масив так, щоб спочатку розташовувалися усі елементи, розташовані після мінімального, а потім - усі інші.

Варіант 18

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• добуток негативних елементів масиву;

• суму позитивних елементів масиву, розташованих до максимального елементу;

• змінити порядок дотримання елементів в масиві на зворотний.

Варіант 19

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• добуток позитивних елементів масиву;

• суму елементів масиву, розташованих до мінімального елементу;

• упорядкувати за збільшенням окремо елементи, що стоять на парних місцях, і елементи, що стоять на непарних місцях.

Варіант 20

У одновимірному масиві, що складається з n дійсних елементів, вичислити:

• добуток елементів масиву з парними номерами;

• суму елементів масиву, розташованих після останнього позитивного елементу;

• замінити усі негативні елементи масиву їх квадратами і упорядкувати елементи масиву по убуванню.

III Заключна частини заняття: відповіді на контрольні запитання і виконання контрольних завдань.

3. Рекомендована література (основна, допоміжна), інформаційні ресурси в Інтернеті

Основна література.

1. Дудзяний І.М. Об'єктно-орієнтоване моделювання програмних систем: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. - 108 с.
2. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник / В.С. Авраменко, А.С. Авраменко. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.
3. H. Schildt, "C# 4.0 The Complete Reference McGraw Hill Education; 1st edition" – 984 p., 2017.
4. J. Sharp, "Microsoft Visual C# Step by Step", Pearson Education – 878p., 2018.

Допоміжна література.

1. A. Troelsen, "Pro C# 7 With .NET and .NET Core" - Apress, — 1372 p., 2017.
2. J. Albahari, B. Albahari, "C# 7.0 in a Nutshell", O'Reilly Media— 1090 p., 2017.
3. Інструментальні програмні засоби розробки ІУС. Методичні вказівки до виконання лабораторних робіт. / уклад.: К.І. Київська–Київ: КНУБА, 2018. – 40 с.

Інформаційні ресурси в Інтернеті

1. C# Reference, сайт розробників MSDN. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/csharp/languagereference/index 4.
2. Об'єктно-орієнтоване програмування. Частина 1. Основи об'єктно-орієнтованого програмування на мові C#.: Навчальний посібник. / Д.В. Настенко, А. Б. Нестерко. – К.: НТУУ «КПІ», 2016. - 76с. [Електронний ресурс]. – Режим доступу https://ela.kpi.ua/bitstream/123456789/16671/1/OOP_manual.pdf
3. Дацун Н.М. Об'єктно-орієнтоване програмування: навчальний посібник для [Електронний ресурс]. – Режим доступу

http://ea.donntu.edu.ua/bitstream/123456789/27021/1/%d0%9e%d0%9e%d0%9f%202014.pdf