МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
“ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ”

МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ ДО ВИКОНАННЯ КУРСОВОЇ РОБОТИ ІЗ ДИСЦИПЛІНИ *“ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ“*

(для студентів напряму підготовки
6.050103 “Програмна інженерія”)

Донецьк ДонНТУ 2011

ЗМІСТ

1 ОСНОВНІ ПОЛОЖЕННЯ

Курсову роботу студенти виконують з метою закріплення практичних навичок самостійної роботи за методикою об'єктно-орієнтованого аналізу і проектування програм і технологічними прийомами розробки об'єктно-орієнтованих програм мовою Сі + +.

Умовами успішного виконання курсової роботи є:

- знання лекційного матеріалу;
- уміння користатися технічною і нормативною літературою (у тому числі програмною документацією і ДСТ [1]);
- практичні навички роботи на ПЕОМ, отримані при виконанні лабораторних робіт.

Під час виконання курсової роботи студент повинний оволодіти методикою проведення проектної роботи з формалізації і розв’язання поставленої задачі, показати уміння використовувати технічну і нормативну літературу, обґрунтовано вибирати методи для розв’язання задачі на ЕОМ, використовувати методику об'єктно-орієнтованого аналізу, проектування [2-4] і розробки об’єктно-орієнтованих програм [3-8], аналізувати результати.

2 МЕТОДИКА ВИКОНАННЯ КУРСОВОЇ РОБОТИ

Завдання на курсову роботу видається студентові на першому тижні після початку семестру і фіксується на спеціальному стандартному бланку (див. додат. А), **який підписується викладачем - керівником роботи і студентом**.

Типова тема курсової роботи - "Об'єктно-орієнтована система" (ООС). Тематика курсової роботи охоплює всі розділи дисципліни “Об'єктно-орієнтоване програмування”.

Технічне завдання на курсову роботу генерується кожному студентові індивідуально за допомогою Windows-програми “Генератор ТЗ” на першому тижні семестра. Студент повинен відформатувати (див. додаток Б) відповідний Word-документ (**.rtf* - файл) у відповідності з вимогами оформлення звітів у сфері науки і техніки (див. додаток А [9]) та роздрукувати його для затвердження керівником курсової роботи та завідувачем кафедри ПМІ.

Курсова робота виконується студентом самостійно. На виконання курсової роботи приділяється 14 тижнів, протягом яких рекомендується регулярна робота з виконання завдання. Час на виконання наступного етапу завдання регулюється студентом самостійно в рамках термінів, приведених у завданні. Курсова робота повинна бути виконана в термін, зазначений у завданні, і захищена не пізніше, ніж за 2 тижні до кінця семестру.

Проміжними формами звітності студента при виконанні курсової роботи є:

- технічне завдання і його затвердження керівником, постановка задачі (2-й тиждень);
- результати об'єктно-орієнтованого аналізу та проектування (8-й тиждень);
- результати технічного проектування (9-й тиждень);
- результати робочого проектування (10-й тиждень);
- текст програми (12-й тиждень);
- пояснювальна записка (14-й тиждень).

Проміжною формою контролю є атестація студента із виконання курсової роботи на 9-10 тижні.

Курсову роботу студент оформляє у вигляді пояснювальної записки, виконаної на ЕОМ чи рукописним способом, загальним обсягом 30-40 сторінок. Студент повинний представити пояснювальну записку керівникові для перевірки за **2-3 робочі дні** до терміну захисту, зазначеного в завданні.

Курсова робота захищається у формі доповіді студента комісії з 2-3 викладачів та відповідей на питання членів комісії. Доповідь за завданням повинна складатися з:

- формулювання завдання;
- постановка задачі.
- об'єктно-орієнтований аналіз і проектування;
- технічне проектування:
- робоче проектування;
- обґрунтування обраних відношень між класами і структури програми;

- обґрунтування ефективності обраних для розв’язання задачі структур даних і алгоритмів;
- пояснення контрольного приклада;
- демонстрації роботи ООС на ПЕОМ.

3 ВИМОГИ ДО КУРСОВОЇ РОБОТИ

3.1 Середовище і технологія програмування

Необхідно розробити об'єктно-орієнтовану систему мовою C++ [3-8, 11-12] у середовищі Borland C++ або MS Visual Studio як консольну програму в операційній системі Windows згідно з варіантом завдання.

Індивідуальне завдання варто виконувати, використовуючи характерні прийоми об’єктно-орієнтованого програмування.

Курсова робота являє собою великий програмний комплекс, тому при програмуванні завдання необхідно використовувати технологію розробки програм з використанням проекту, роздільної компіляції файлів та поділу інтерфейсу класу і його реалізації [3, 4, 8].

3.2 Організація даних

Початкові дані варто організувати ***у вигляді файлів*** (у тому числі тексти для користувальницького інтерфейсу, тексти допомоги і т.п), звівши до мінімуму введення даних безпосередньо користувачем.

Результати роботи (стан об'єктів системи) необхідно представити ***у вигляді файлів***, відповідно до вимог, зафіксованим у “Технічному завданні”.

Необхідно забезпечити двомовність мови спілкування з користувачем. Рекомендовані пари мов: одна, що використовує кирилицю, інша – що використовує латиницю. ***В разі необхідності в ООС виконати транслітерацію фраз мови кирилиці на латиницю.***

3.3 Режим роботи програми

Введення даних користувачем організувати в режимі діалогу з перевіркою коректності інформації, що вводиться, (наприклад, виключити введення тексту в числові поля і т.п.).

Команди для вибору режиму роботи ООС і виконання сценарію роботи з програмою сгрупувати в меню, скоротивши обсяг введення інформації користувачем.

ООС повинна працювати в режимах:

а) опис предметної області (ПрО);

б) режим демонстрації роботи об'єктів;

а також мати систему допомоги.

Перераховані вище режими роботи інструменту створення БЗ ЕС обов’язкові для будь-якої оцінки з курсової роботи.

Режим демонстрації роботи об'єктів повинний виконувати дії:

а) створення об'єктів базових класів;

б) показ стану об'єктів базових класів;

в) створення об'єктів похідних класів;

г) показ стану об'єктів похідних класів;

д) перевантаження операторів базових класів;

е) перевантаження операторів похідних класів.

В режимі демонстрації роботи об'єктів дії а)-е) обов’язкові для будь-якої оцінки з курсової роботи.

3.4 Структури даних, методи та алгоритми

При виборі структур даних для виконання індивідуального завдання із курсової роботи варто скористатися стандартними структурами даних (масивами і файлами).

Методи та алгоритми розв’язання задач індивідуального завдання курсової роботи повинні бути ефективними для обраних структур даних.

3.5 Технології розробки ООС

При об'єктно-орієнтованому аналізі та проектуванні ООС необхідно використати знання технологічних етапів створення проекту моделі ПрО: виявлення і ідентифікація класів, виявлення і ідентифікація відношень між класами; формалізація класів і відношень між ними засобами мови UML; опис поведінки і стану об'єктів ПрО засобами мови UML.

При технічному проектуванні ООС необхідно використати знання технологічних етапів створення програм на ЕОМ: постановка задачі, створення методу розв’язання задачі, створення алгоритму розв’язання задачі.

При робочому проектуванні ООС необхідно використати знання технологій створення програм з використанням проекту, роздільної компіляції файлів та поділу інтерфейсу класу і його реалізації.

При розробці документації до проекту інформатизації необхідно ознайомитися із стандартом “Единая система программной документации” [1] та засвоти технологію створення програмних документів:

а) технічне завдання;

б) опис програми;

в) керівництво оператора;

г) текст програми.

Ці програмні документи необхідно розмістити в додатках пояснювальної записки до курсової роботи. “Опис програми” і “Керівництво оператора” в електронному вигляді необхідно представити в системі **допомога/довідка.** Також ця система повинна містити інформацію про автора.

3.6 Інтерфейс користувача

Порція інформації, надана користувачу на екрані в окремий момент часу, називається ***кадром.***

Робота демонстраційного приклада повинна починатися з **форми заставки**. *Текст заставки* повинний містити:

- назви ВНЗ, факультету, кафедри;

- назву теми індивідуального завдання;
- прізвище і ініціали студента;
- назву групи;
- прізвище і ініціали керівника роботи;
- місце і рік створення.

Також на заставці необхідно передбачити можливість зміни мови спілкування з користувачем.

Наступний кадр повинний містити головне меню з режимами роботи ООС.

Кадр опису ПрО повинний містити **опис функціонального призначення ООС і опис предметної області.**

Кадр режиму демонстрації роботи об'єктів повинний містити меню з можливами діями над об'єктами.

Дії по створенню об'єктів потребують використання введення-виведення інформації користувачем. **Введення даних безпосередньо користувачем необхідно звести до мінімуму та контролювати правильність введених даних.**

Кадр допомоги/довідки може бути створений з використанням текстових файлів.

Припустимо дані результатів роботи програми представляти у вигляді текстових файлів.

3.7 Пояснювальна записка

Пояснювальна записка повинна бути оформлена відповідно до Державного стандарту України ДСТУ 3008-95 "Документація. Звіти в сфері науки і техніки. Структура і правила оформлення" (див. додаток А [9]).

Зміст пояснювальної записки (з нумерацією складових частин) приведений в додатку Б).

3.8 Графічний матеріал

*У пояснювальній записці для **розробленої ООС** повинні бути представлені у вигляді схем:*

а) статична модель ПрО, створена засобами UML:
   1) діаграми класів;
   2) діаграми об'єктів;

б) динамічна модель ПрО, створена засобами UML:
   1) діаграми прецедентів;
   2) діаграма послідовностей;
   3) діаграма кооперації;
   4) діаграми станів об'єктів;

в) структура ООС;

г) основний алгоритм функціонування ООС;

д) опис структур даних ООС.

Приклади діаграм UML а)-б) розлянуті в лекційному курсі [1, 4, 11], а також представлені в дистанційному курсі [12].

Опис структур даних, використаних в ООС включає:

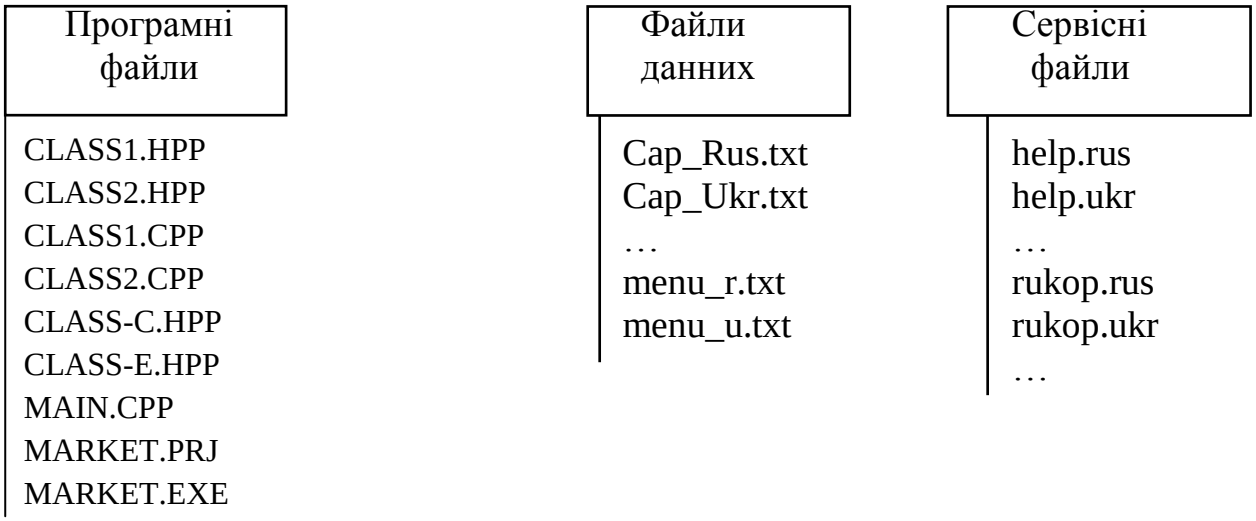
- структуру даних програми (вхідних і вихідних даних, файлів);
- структуру файлової системи програми (програмних файлів, файлів даних і т.д.).

На мал. 3.1 приведений приклад структури файлу вхідних даних “Написи для форми заставки” (*Cap_Rus.txt*), що містить написи російською мовою для інтерфейсу ООС:

№ рядку файлу	Вміст рядку	№ рядку файлу	Вміст рядку
	Для першого кадру		***Опції головного меню:***
1	назва ВНЗ	15	опис ПрО
2	назва факультету	16	режим демонстрації роботи об'єктів
3	назва кафедри	17	допомога/довідка
4	назва теми індивідуального завдання	18	вихід з системи
5	назва групи	19	***Опції головного меню режим демонстрації роботи об'єктів:***
6	прізвище і ініціали студента	20	створення об'єктів базових класів
7	прізвище і ініціали керівника роботи	21	показ стану об'єктів базових класів
8	місце і рік створення	22	створення об'єктів похідних класів
9	продовжити (перехід на наступну форму)	23	показ стану об'єктів похідних класів
10	завершити роботу	24	перевантаження операторів базових класів
11	назва основної мови інтерфейсу	25	перевантаження операторів похідних класів
12	назва альтернативної мови інтерфейсу	26	***Повідомлення про помилки:***
	Повідомлення про:	27	створення об'єктів
13	продовження роботи	28	показ стану об'єктів
14	створення об'єктів	29	перевантаження операторів
		30	вибору опцій меню

Малюнок 3.1 - Приклад структури файлу вхідних даних

На мал. 3.2 приведений приклад структури файлової системи ООС “Торговий центр та магазин самообслуговування”. Тут представлені 3 групи файлів:



Малюнок 3.2 - Структура файлової системи ООС “Торговий центр та магазин самообслуговування”

а) програмні файли (файли інтерфейсів класів *class1.hpp* і *class2.hpp*, файли реалізації класів *class*.cpp*, файл використання *main.cpp*, файли для зовнішніх змінних, які використуються при роздільної компіляції файлів *class-*.hpp*, файл *market.prj* проекту для трансляції модулів, здійсненний файл *market.exe*);

б) файли вхідних даних на двох мовах для інтерфейсу користувача:

1) файли *Cap*.txt* написів інтерфейсу ООС;

2) файли *menu*.txt* написів опцій меню;

в) сервісні файли на двох мовах (*help.** допомоги, кервництва оператора *rukop.** і т.п.).

“Специфікацію модулів” необхідно оформити у вигляді таблиці. Термін “модуль” використовуюється для позначення в мові С++:

а) функції-члена класа;

б) дружньої функції;

в) статичної функції;

г) віртуальної функції;

д) звичайної функції мови С++;

Про кожен модуль повідомляється така інформація:

- найменування;

- семантика (короткий опис виконуваних дій);

- інтерфейс за даними (опис типів і структури вхідних і вихідних даних, у ролі яких можуть бути використані глобальні змінні);

- інтерфейс по керуванню (найменування модулів, що стосовно даного є тими, що викликають, і тими, шо викликаються).

У табл. 3.1 приведений приклад специфікації модулів різних типів, які використовуються в ООС “Торговий центр та магазин самообслуговування”.

Таблиця 3.1 - Специфікація модулів ООС "Торговий центр та магазин самообслуговування"

Найменування	*Семантика*	*Вхідні дані*	*Вихідні дані*	*Викликається*	*Викликає*
CMarket - конструктор за замовчуванням базового класу ***CMarket*** (торговий центр)	Створення об'єкту класу ***CMarket*** з характеристиками за замовчуванням: назва магазину "new", торгова площа 20.0, кількість торгових залів 1 (пам'ять для рядка з назвою виділяється в купі)	-	Об'єкт класу ***CMarket*** з даними-членами класу: - *mName;* - *mSq;* - *mCZals.*	Один раз перед початком роботи системи для створення покажчика market на об'єкт класу ***CMarket***	-
SetCost - функція-член похідного класу ***CSelfM*** (магазин самообслуговування)	Встановлення значення для даного-члена класу *mCost* - вартість камери зберігання (виконується перевірка правильності цього значення, яке повинне бути більше 0).	*ACost* – Цілочисельний, аргумент функції; початкове значення для даного-члена класу *mCost*	а) цілочисельний результат повертається через ім'я функції; це ознака помилки встановлення значення для даного-члена класу *mCost*; - 0, якщо значення не встановлено; - 1, якщо значення встановлено і дорівнює значенню аргумента *ACost*; б) рядок повідомлення про помилку в системному потоці виведення: "ПОМИЛКА: вартість камери зберігання"	а) функцією-членом класу ***SetSelfM***; б) дружньою функцією ***operator!***	Оператор виведення в системний потік помилок
operator! – дружня функція класу ***CSelfM***	Збільшує вдвічі вартість камери зберігання у об'єкта класу ***CSelfM***	*m* – об'єкт класу ***CSelfM***; аргумент функції	*m* - модифікований об'єкт класу ***CSelfM***	Функцією *ov_un*	Функції-члени класу ***SetSelfM***: - GetCost; - SetCost

Продовження табл. 3.1

Найменування	*Семантика*	*Вхідні дані*	*Вихідні дані*	*Викликається*	*Викликає*
main_menu – функція	Робота головного меню ООС: а) виведення тексту меню; б) отримання відповіді користувача; в) розгалуження процесу роботи ООС; г) виведення повідомлення при помилковлму введені номеру опції меню та повторення виведення тексту меню	*reply* – Цілочисельне, номер опції меню, ввведений користувачем через системний потік введення	Рядок повідомлення про помилку в системному потоці виведення: "Ви помилилися. Натисніть будь-яку клавішу і введіть номер опції"	Функцією *zastavka*	а) Функції: 1) *uslovie;* 2) *menu_reply;* 3) *help ;* 4) *exit ;* 5) *zastavka;* 6) *getch;* б) оператор виведення в системний потік виведення; в) оператор виведення в системний потік помилок; г) оператор введення з системного потоку введення; д) метод *width* і маніпулятор *endl* системного потоку виведення
main - головна функція	Очищення екрану монітору і виведення першого екрану інтерфейсу ООС		Цілочисельний результат повертається через ім'я функції до операційної системи; це ознака наявності/відсутності помилки в роботі програми	Операційною системою	Функції: а) clrscr з бібліотеки <conio.h>; б) zastavka.

3.9 Перелік посилань

“Перелік посилань” повинний містити такі види літератури:

- *наукова чи науково-популярна,* у якій містяться опис предметної області, методи розв’язання задачі і т.д.;
- *технічна*, у якій міститься опис структур даних, алгоритмів, програмних і апаратних засобів, використаних при розробці програмного проекту;
- *методична* (методичні посібники, вказівки і т.д.);
- *нормативна* (закони, стандарти і т.д.).

На всі літературні джерела обов'язково повинні бути посилання в тексті пояснювальної записки. Правила оформлення джерел див. в [9].

3.10 Додатки

Додатки містять програмні документи, тексти файлів вхідних даних, тексти файлів вихідних даних, екранні форми.

Розмір шрифту тексту всіх додатків – 8, міжрядковий інтервал - 1.

Усі програмні документи необхідно виконати відповідно до стандарту [1]:

- *“Технічне завдання”* (див. додат. Б [9]);
- *“Опис програми”* (див. додат. А [10]);
- *“Керівництво оператора”* (див. додат. В [9]).

Тексти програм повинні бути:

- прокоментовані (кожна процедура і кожний предикат повинні містити коментарі в описах змінних і в частині, що виконується);
- структуровані (відступи для вкладених конструкцій і складових частин);
- **текст програм мовою С++ необхідно розмістити в дві колонки.**

Розмір окремої процедури на С++ повинний бути таким, щоб його можна було прочитати в межах одного екрана без листання.

Екранні форми повинні містити вигляд **всіх режимів роботи ООС.** Екранні форми можуть бути підготовлені рукописним чи машинним способом.

Додаткові вимоги на оцінки “відмінно”, “добре” і “задовільно” приведені в тексті прикладу “Технічного завдання” з критеріями оцінювання (див. додаток Б).

4 ПЕРЕЛІК ВИКОРИСТОВУВАНОГО ПРОГРАМНОГО ТА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Перелік використовуваного програмного забезпечення

Програмне забезпечення, використовуване при виконанні курсової роботи:

а) операційна система – MS Windows (версія не нижче XP);

б) транслятори мови C++ (ф.Borland Inc., Microsoft і т.д.);

в) текстові редактори або процесори.

4.2 Склад використовуваних технічних засобів

Для виконання курсової роботи використовуються лабораторії обчислювальної техніки кафедри прикладної математики і інформатики на базі ПЕОМ. При необхідності можуть використовуватися індивідуальні ПК студентів.

Індивідуальні завдання по курсовій роботі повинні бути виконані ***тільки з використанням стандартних периферійних пристроїв ПЕОМ.***

5 МЕТОДИЧНІ РЕКОМЕНДАЦІЇ З ПРОГРАМУВАННЯ

5.3 Реалізація двомовності інтерфейсу

Для реалізації двомовності інтерфейсу використовуємо текстові файли, вміст яких структуруємо відповідно послідовності розміщення написів на екрані. На мал. 5.1 представлений приклад вмісту файлів *Cap_Rus.txt* і *Cap_Ukr.txt*. **Кожна запис цих файлів містить один рядок написи на екрані, тому запис з номером *i* файлу *Cap_Ukr.txt* містить переклад на українську мову тексту запису з номером *i* файлу *Cap_Rus.***

Министерство образования и науки Украины ДонНТУ Кафедра ПМИ … Нажмите любую клавишу Создан объект базового класса … 1. создание объекта базового класса 2. показ состояния объекта базового класса ... ОШИБКА: нельзя перегрузить унарный оператор. Выберите опцию 3	Міністерство освіти і науки України ДонНТУ Кафедра ПМІ ... Натисніть будь-яку клавішу Створений об'єкт базового класу … 1. створення об'єкта базового класу 2. показ стану об'єкта базового класу … ПОМИЛКА: не можна перевантажити унарний оператор. Виберіть опцію 3 …
Вміст файлу *Cap_Rus.txt*	Вміст файлу *Cap_Ukr.txt*

Малюнок 5.1 - Приклад файлів даних для організації двомовності інтерфейсу

На мал. 5.1 показані записи файлів у випадку, коли написи всіх кадрів інтерфейсу розміщені в одному файлі.

Альтернативними рішеннями є:

а) розміщення текстів написів різних кадрів у різних файлах;

б) розміщення текстів написів у файлах за семантикою.

Наприклад, на мал. 3.2 видно, що в проекті тексти написів кадрів розміщені у файлах *Cap_*. *txt*, тексти написів елементів меню - у файлах *menu_*.txt*, тексти системи допомоги - у файлах *help.**.

Файли даних з текстами написів елементів інтерфейсу можуть бути розміщені:

а) в одній папці з виконуваним файлом ООС. Тоді необхідно:

1) отримати exe-файл програми за допомогою IDE мови С++;

2) у цій ж папці (поточній) розмістити файли даних;

3) у програмі мовою С++ імена таких файлів можуть бути задані константами без вказівки шляху доступу до папки проекту, наприклад:

```
char namef[20]="help.txt";
char namef[20]="uslovie.txt";
```

б) у зазначеній папці, яка не збігається з папкою з виконуваним файлом ООС. Тоді необхідно:

1) отримати exe-файл програми за допомогою IDE мови C++;

2) у зазначену папку помістити файли даних (**можливо, файли структуровані по папках відповідно до мови інтерфейсу**);

3) у програмі мовою C++ імена таких файлів можуть бути задані константами із зазначенням шляху доступу до папок цих файлів, наприклад.

char namef[20]="D:\STUDENTS\PS\help.txt";
char namef[20]="D:\STUDENTS\PS\uslovie.txt";

Рішення а) і б) відповідають вимогам на оцінку "задовільно".

в) у будь-якому місці файлової системи і шлях доступу до файлів визначається користувачем на етапі виконання програми. Тоді необхідно передбачити в головному меню ООС опцію налаштування, в якій користувач вказує шлях доступу до файлів даних і/або їх імена;

Рішення в) відповідає вимогам на оцінку "добре".

г) у будь-якому місці файлової системи, але користувачу не доводиться вводити шлях доступу до файлів. Тоді необхідно передбачити у файловій системі ООС файл конфігурації з даними про шляхи доступу до файлів даних і/або їхніх іменах;

Рішення г) відповідає вимогам на оцінку "відмінно".

Розглянемо реалізацію зміни мови інтерфейсу в програмі на C++ для випадку: файли даних розміщені в одній папці з виконуваним файлом ООС. Виконуємо такі дії:

а) в головному меню ООС передбачаємо опцію для зміни мови:

1. Опис ПрО
2. Демонстрація роботи об'єктів
3. Зміна мови інтерфейсу
4. Допомога
5. Вихід

б) визначаємо глобальну змінну ***lang*** цілочисельного типу для зберігання значення ознаки мови. Наприклад, її значення ***1 (True)*** відповідає російській мові, значення ***0 (False)*** - українській. Такий вибір типу даних для ознаки мови дозволяє збільшити множину можливих мов інтерфейсу ООС. За замовчуванням після завантаження програми встановлено українську мову:

int lang = 0;

в) у функції *main_menu(),* яка реалізує роботу головного меню ООС, обробляємо результат вибору користувачем опції "Зміна мови інтерфейсу " (введення значення 3):

```
cout<<"Введіть номер опції: 1, 2, 3, 4, 5 "<<endl;
cin>>reply;
switch(reply)
    {…
    case 3: if(lang)lang=0; else lang=1;   // Зміна мови
            load_cap(); break;  …
    }
```

г) визначаємо глобальні змінні для зберігання імен файлів з написами кадрів інтерфейсу:

```
const char nfile_r[12]="Cap_Rus.txt";
const char nfile_u[12]="Cap_Ukr.txt";
```

д) визначаємо глобальну змінну для зберігання імені файлу для поточної мови інтерфейсу:

```
char nfile[12];
```

е) визначаємо глобальний масив рядків для зберігання написів кадрів інтерфейсу:

```
const CCAPT=35;
const LCAPT =50;
char mas_cap[CCAPT][LCAPT];
```

У ньому можна зберігати 35 написів-рядків, кожна з яких не більше 49 символів.

ж) створюємо функцію *load_cap*, яка буде змінювати вміст написів-рядків в масиві *mas_cap*:

```
void LoadCaption()
 {
  if (lang)
    strcpy(nfile, nfile_r);
  else
    strcpy(nfile, nfile_u);

  ifstream fcapt;
  fcapt.open(nfile);
  if(fcapt.fail())
   {cerr<<"Файл "<<nfile<<
          " не відкритий."  <<endl;
    getch();
    exit(-1);
   }
  int i;
  for(i=0; i< CCAPT; i++)
      fcapt >> mas_cap[i];
}
```

з) елементи масиву *mas_cap* необхідно використовувати для виведення рядків інтерфейсу в системні потоки виведення і помилок (з урахуванням нумерації написів у цьому масиві), Наприклад:

```
void zastavka()
 {
  clrscr();
  cout <<endl;
  cout.width(45); cout.setf(ios::right);
  cout<< mas_cap[0]  <<endl;
  cout.width(42); cout.setf(ios::right);
  cout<< mas_cap[1]  <<endl;
  cout.width(45); cout.setf(ios::right);
  cout<< mas_cap[2]  <<endl;
  cout<<endl<<endl<<endl<<endl;
  …
 }
```

```
void main_menu()
 {int reply;
  cout.width(10);
  cout<<"1.  " << mas_cap[14];
  cout.width(25);
  cout<<"2. " << mas_cap[15];
  cout.width(10);
  cout<<"3. " << mas_cap[16];
  cout.width(10);
  cout<<"4. "<< mas_cap[17]<<endl;

 }
```

6 ТЕМАТИКА СПЕЦІАЛЬНОЇ ЧАСТИНИ КУРСОВОЇ РОБОТИ

В цьому розділі представлені варіанти індивідуальних завдань з курсової роботи у вигляді опису базового і похідного класів.

Варіант 1

Створити клас *Point*, що містить в собі такі елементи:

- поле «координата X» *float X*;
- поле «координата Y» *float Y*;
- метод встановлення координат *void SetCoordinate (float X, float Y)*;
- метод отримання координати X *float GetX()*;
- метод отримання координати Y *float GetY();*
- конструктор без параметрів *Point()*;
- конструктор з параметрами *Point(float X, float Y)*.

Успадкувати від класу *Point* клас *ColorPoint*, що містить в собі елементи:

- поле «колір» *int Color*;
- метод встановлення кольору *void SetColor(int Color)*;
- метод отримання кольору *int GetColor()*;
- конструктор без параметрів *ColorPoint();*
- конструктор з параметрами *ColorPoint(float X, float Y, int Color)*.

В класі *ColorPoint* перевантажити оператор ++ (унарний, оператор класу), нова дія – збільшити колір на одиницю (колір змінюється по колу в діапазоні 0..16).

Варіант 2

Створити клас *Animal*, що містить в собі такі елементи:

- поле «вага» *float Mass*;
- поле «стать» *char* Sex*;
- поле «колір» *char* Color*;
- поле «вік» *int Age*;
- метод отримання ваги *float GetMass()*;
- метод отримання статі *char* GetSex()*;
- метод отримання віку *int GetAge()*;
- метод отримання кольору *char* GetColor()*;
- конструктор з параметрами *Animal(float Mass, char* Sex, ,char* Color, int Age)*.

Успадкувати від класу *Animal* клас *Dog*, що містить в собі елементи:

- поле «окличка» *char* Name*;
- поле «порода» *char* Race*;
- поле «вид» (мисливська, декоративна, бійцівська і т.д.) *char* Type*;
- конструктор з параметрами *Dog(float Mass, char* Sex, ,char* Color, int Age, char* Name, char* Race, char* Type)*.

В класі *Animal* перевантажити оператор -- (унарний, оператор класу), нова дія – зменшити вагу на 0.1 кг (вага повинна бути не менше 0.1 кг).

Варіант 3

Створити клас *Vegetable*, що містить в собі такі елементи:

- поле «вага» *float Mass*;
- поле «стиглість» *int Ripeness*;
- метод отримання ваги *float GetMass()*;
- метод отримання стиглості *int GetRipeness()* ;
- метод здобрити овоч *void Fertilize()*; (збільшує масу на 0.1 кг)
- конструктор з параметрами *Vegetable(float Mass, int Ripeness)*.

Успадкувати від класу *Vegetable* клас *Tomato*, що містить в собі елементи:

- поле «сорт» *char* Type*;
- поле «колір» *char* Color*;
- поле «розмір» *char* Size*;
- конструктор з параметрами *Tomato(float Mass, int Ripeness, char* Type, char* Color, char* Size)*.

В класі *Tomato* перевантажити оператор - (бінарний, дружній), нова дія – з двох помідорів вернути той, у якого вага більше.

Варіант 4

Створити клас *Vehicle*, що містить в собі такі елементи:

- поле «швидкість» *int Speed*;
- поле «вага» *int Mass*;
- метод отримання ваги *int GetMass()*;
- метод зупинення *void Stop()*; (швидкість=0);
- конструктор з параметрами *Vehicle(int Speed, int Mass)*.

Успадкувати від класу *Vehicle* клас *Truck*, що містить в собі елементи:

- поле «вантажність» *int Capacity*;
- поле «вага вантажу» *int Cargo*;
- конструктор з параметрами *Truck(int Speed, int Mass, int Capacity)*;
- метод «завантажити вантаж» *int Load(int Mass)*; (якщо успішно, то повертає 0, інакше -1);
- метод «розвантажити вантаж» *int UnLoad()*; (повертає вагу вантаж і встановлює її в 0).

В класі *Vehicle* перевантаж оператор ++ (унарний, оператор класу), нова дія – збільшити швидкість на 10 км/год.

Варіант 5

Створити клас *Veapon*, що містить в собі такі елементи:

- поле «об’єм магазину» *int Capacity*;
- поле «калібр» *int Calibre*;
- поле «дальність стрільби» *int Distance*;
- метод «зарядити зброю» *void Charge()*;
- метод «розрядити зброю» *void UnCharge()*;
- конструктор з параметрами *Veapon(int Capacity, int Calibre, int Distance)*.

Успадкувати від класу *Veapon* клас *Gun*, що містить в собі елементи:

- поле «табельний номер» *int Number*;
- поле «тип дула» *char* BarrelType*; (гладкодульне, нарізне)
- поле «стан» *int State*; (на запобіжнику або ні)
- конструктор з параметрами *Gun(int Capacity, int Calibre, int Distance, int Number, char* BarrelType)*;
- метод «одиночний постріл» *void Shot()*.

В класі *Gun* перевантажити оператор ! (унарний, оператор класу), нова дія – поставити/зняти з запобіжника.

Варіант 6

Створити клас *Media*, що містить в собі такі елементи:

- поле «об'єм» *long Size*;
- поле «Швидкість читання» *int ReadSpeed;*
- поле «Швидкість запису» *int WriteSpeed;*
- поле «Виробник» *char* Vendor;*
- поле «Дані» *char* Data;*
- метод зчитування даних *char* Read();*
- метод запису даних *void Write(char* buffer);*
- конструктор з параметрами *Media(long Size, int ReadSpeed, int WriteSpeed)*.

Успадкувати від класу *Media* клас *HDD*, що містить в собі елементи:

- поле «інтерфейс» *char* Interface*; (SATA, IDE, SCSI)
- поле «кількість розділів на диску» *int PartitionNum*;
- конструктор з параметрами *HDD(long Size, int ReadSpeed, int WriteSpeed, char* Interface , int PartitionNum)*;
- метод «форматувати диск» *void Format()*;
- метод «форматувати розділ» *int Format(int PartitionNum)*; (форматує розділ з вказаним номером, якщо такого розділу не має, то повертає -1, інакше -0).

В класі *Media* перевантажити оператор ~ (унарний, оператор класу), нова дія – повертає кількість розділів.

Варіант 7

Створити клас *File*, що містить в собі такі елементи:

- поле «розмір» *long Size*;
- поле «Дата створення» *char* Date*;
- поле «Власник» *char* Owner*;
- поле «зміст файлу» *char* Content*;
- метод зчитування даних *char* Read()*;
- метод запису даних *void Write(char* buffer)*;
- конструктор з параметрами *File(long Size, char* Date, char* Owner*).

Успадкувати від класу *File* клас *Document*, що містить в собі елементи:

- поле «розмір шрифту» *int FontSize*;
- поле «колір шрифту» *int FontColor*;
- конструктор з параметрами *Document(long Size, char* Date, char* Owner, int FontSize, char* Color)*;
- метод «Встановити колір» *void SetColor(char* Color)*.

В класі *Document* перевантажити оператор + (бінарний, дружній оператор), нова дія – зливання змісту документів.

Варіант 8

Створити клас *Employee*, що містить в собі такі елементи:

- поле «ПІБ» *char* FIO*;
- поле «Табельний номер» *int Number*;
- поле «Вік» *int Age*;
- поле «Стаж» *int Stage*;
- метод отримання ПІБ *char* GetFIO()*;
- метод отримання таб. номера *int GetNumber()*;
- метод отримання стажу *int GetStage()*;
- метод отримання віку *int GetAge()*;
- конструктор з параметрами *Employee(char* FIO, int Number, int Stage, int Age).*

Успадкувати від класу *Employee* клас *Turner* (Токарь), що містить в собі елементи:

- поле «Розряд» *int Experience*;
- поле «Номер цеху» *int Department*;
- конструктор з параметрами *Turner(char* FIO, int Number, int Stage, int Age, int Department, int Experience)*;
- метод «Зміна цеху» *void ChangeDepartment(int NewDepartment)*.

В класі *Turner* перевантажити оператор ++ (унарний, оператор класу), нова дія – підвищити розряд.

Варіант 9

Створити клас *LightDevice*, що містить в собі такі елементи:

- поле «Яскравість» *int Light*;
- поле «Потужність» *int Power*;
- поле «Напруга» *int Voltage*;
- метод отримання яскравості *int GetLight()*;
- метод отримання потужності *int GetPower()*;
- метод отримання напруги *int GetVoltage()*;
- конструктор з параметрами *LightDevice(int Light, int Power, int Voltage)*.

Успадкувати від класу *LightDevice* клас *Lamp*, що містить в собі елементи:

- поле «стан» (вкл/викл) *int State*;
- поле «колір» *int Color*;
- конструктор з параметрами *Lamp(int Light, int Power, int Voltage,*
- *int Color);*

- метод «Включити» *void On()*;
- метод «Виключити» *void Off()*.

В класі *Lamp* перевантажити оператор ++ (унарний, оператор класу), нова дія – збільшити яскравість.

Варіант 10

Створити клас *Confection*, що містить в собі такі елементи:

- поле «Назва» *char* Name;*
- поле «Дата виготовлення» *struct Date {int year; int month; int day} ProductDate*;
- поле «Строк придатності» *struct Date {int year; int month; int day} BestBefore*;
- включити до класу поле класу *Filling* (начинка) *Filling* filling*;
- метод отримання дати виготовлення *int GetProductDate()*;
- метод отримання назви *int GetName();*
- метод додавання начинки *void AddFilling(Filling filling)*;
- конструктор з параметрами *Confection(char* Name, Date ProductDate, Date BestBefore, Filling* filling).*

Створити клас *Filling*, що містить в собі елементи:

- поле «кількість інгредієнтів» *int Number;*
- поле «Назва» *char* Name;*
- конструктор з параметрами *Filling(int Number,char* Name);*
- метод «Видалити інгридієнт» *void AddIngredient();*
- метод «Додати інгридієнт» *void DelIngredient()*.

В класі *Confection* перевантажити оператор ~ (унарний, оператор класу), нова дія – видалити начинку.

Варіант 11

Створити клас *Ellipse*, що містить в собі такі елементи:

- поле «центр» *Point* Center*;
- поле «малий радіус» *float a;*
- поле «великий радіус» *float b;*
- метод отримання координат центра *Point GetCenter();*
- метод отримання малого радіуса *float Get_a();*
- метод отримання великого радіуса *float Get_b();*
- метод перевірки, чи є еліпс окружностью *int isCircle();*
- конструктор с параметрами *Ellipse(Point* Center, float a, float b)*.

Створити клас *Point*, що містить в собі елементи:

- поле «координата X» *float X*;
- поле «координата Y» *float Y*;
- метод встановлення координат *void SetCoordinate(float X, float Y)*;
- метод отримання координати X *float GetX()*;
- метод отримання координати Y *float GetY()*;
- конструктор з параметрами *Point(float X, float Y)*.

В класі *Ellipse* перевантажити оператор - (бінарний, дружній оператор), нова дія – повернути відстань між центрами еліпсів.

Варіант 12

Створити клас *Function*, що містить в собі такі елементи:

- поле «запис функції» *char* FunctionString*;
- поле «кількість точок розриву» *int NumOfPointDiscontinuity*;
- поле «кількість змінних» *int NumOfVariables*;
- метод отримання запису функції *char* GetFunctionString()*;
- метод отримання кількості точок розриву *int GetNumOfPointDiscountinuity()*;
- метод отримання кількості змінних *int GetNumOfVariables()*;
- конструктор с параметрами *Function (char* FunctionString, int NumOfPointDiscontinuity, int NumOfVariables)*.

Створити клас *Sin*, що містить в собі елементи:

- поле «період» *float Period*;
- поле «амплітуда» *float Amplitude*;
- метод встановлення амплітуди *void SetAmplitude(float A)*;
- метод встановлення періоду *void SetPeriod(float T)*;
- метод отримання значення функції у вказаній точці *float GetValue (float x)*;
- конструктор без параметрів *Sin()*.

В класі *Sin* перевантажити оператор -- (унарний, оператор класу), нова дія – зменшити амплітуду на 0.1.

Варіант 13

Створити клас *Fly*, що містить в собі такі елементи:

- поле «швидкість» *int Speed*;
- поле «висота польоту» *int Height*;
- поле «дальність польоту» *int Distance*;
- поле «кількість пасажирів» *int NumOfPass*;
- метод отримання швидкості *int GetSpeed()*;
- метод отримання дальності польоту *int GetDistance()*;
- метод отримання висоти польоту *int GetHeight()*;
- метод отримання кількості пасажирів *int GetNumOfPass()*;
- метод перевірки, чи може літати на вказаній висоті *int IsFlying(int Height)*;
- конструктор з параметрами *Fly(int Speed, int Distance, int NumOfPass, int Height).*

Створити клас *Helicopter*, що містить в собі елементи:

- поле «кількість гвинтів» *int NumOfScrew*;
- поле «об’єм паливного баку» *int Capacity*;
- метод заправки баку *int AddFuel(int Fuel)*; (якщо бак вже повний, то повертає -1, інакше 0)
- конструктор з параметрами *Helicopter(int Speed, int Distance, int NumOfPass, int Height, int NumOfScrew, int Capacity).*

В класі *Helicopter* перевантажити оператор ~ (унарний, оператор класу), нова дія – зброс палива.

Варіант 14

Створити клас *CelestialBody*, що містить в собі такі елементи:

- поле «вага» *float Mass*;
- поле «радіус» *float Radius*;
- метод отримання ваги *int GetMass()*;
- метод отримання радіуса *int GetRadius()*;
- метод розрахунку щільності речовини небесного тіла *float GetDensity()*;
- конструктор з параметрами *CelestialBody(float Mass, float Radius)*.

Створити клас *Planet*, що містить в собі елементи:

- поле «період оберту навколо осі» *float RotatePeriod*;
- поле «період оберту навколо центра системи» *float RotationPeriod*;
- відстань до центру системи *float Distance*;
- метод отримання швидкості оберту навколо осі *float GetRotatePeriod()*;
- метод отримання швидкості оберту навколо центра системи *float GetRotationPeriod();*
- метод отримання відстані від центра системи *float GetDistance();*
- конструктор с параметрами *Planet(float Mass, float Radius, float RotatePeriod, float RotationPeriod, float Distance).*

В класі *CelestialBody* перевантажити оператор + (бінарний, дружній оператор), нова дія – зливання двох тіл (отримання нового з сумарною вагою й радіусом).

Варіант 15

Створити клас *MobilePhone*, що містить в собі такі елементи:

- поле «виробник» *char* Vendor*;
- поле «модель» *char* Model*;
- клас включає об'єкт класу SIM-карта *SIM* card*;
- метод отримання назви телефону (виробник+модель) *char* GetName()*;
- метод здійснення дзвінка *void Call()*;
- метод відправки SMS *void SendSMS();*
- метод «вставити SIM» *void InsertSIM(SIM card)*;
- конструктор з параметрами *MobilePhone(char* Vendor, char* Model)*.

Створити клас *SIM*, що містить в собі елементи:

- поле «номер» *char* Number*;
- поле «оператор» *char* Operator*;
- конструктор з параметрами *SIM(char* Operator, char* Number)*.

В класі *MobilePhone* перевантажити оператор -- (унарний, оператор класу), нова дія – вийняти SIM-карту.

Варіант 16

Створити клас *Money*, що містить в собі такі елементи:

- поле «валюта» *char* Currency*;
- поле «сума» *long Sum*;

- метод отримання валюти *char* GetCurrency()*;
- метод отримання суми *long GetSum();*
- конструктор з параметрами *Money(char* Currency, long Sum)*.

Створити клас *Account*, що містить в собі елементи:
- поле «ПІБ» *char* FIO*;
- поле «ідентифікаційний код» *long ident*;
- метод «положить гроші на рахунок» *void AddMoney(long Sum)*;
- метод «зняти гроші з рахунку» *long SubMoney(long Sum)*; (-1 якщо не має вказаної суми)
- конструктор з параметрами *Account(char* FIO, long Ident, Money m)*.

В класі *Account* перевантажити оператор ! (унарний, оператор класу), нова дія – обнулити рахунок.

Варіант 17

Створити клас *Print*, що містить в собі такі елементи:
- поле «назва» *char* Title*;
- поле «кількість сторінок» *int Number*;
- метод отримання назви *char* GetTitle()*;
- метод отримання кількості сторінок *int GetNumberPage()*;
- конструктор з параметрами *Print(char* Title, int Number)*.

Успадкувати від класу *Print* клас *Book*, що містить в собі елементи:
- поле «Автор» *char* Author*;
- поле «Видавництво» *char* Publicator*;
- поле «Тираж» *int Count*;
- конструктор з параметрами *Book(char* Publicator,int Count, char*)*.

В класі *Book* перевантажити оператор ++ (унарний, оператор класу) нова дія – збільшити тираж на 10 екземплярів.

Варіант 18

Створити клас *Matrix*, що містить в собі такі елементи:
- поле «розмір» *Size s*;
- поле «дані» *int Data[s.Raw,s.Column]*;
- метод отримання кількість рядків *int GetRaw()*;
- метод отримання кількість стовпців *int GetColumn()*;
- метод отримання значення комірки *int GetValue(int i, int j);*
- конструктор з параметрами *Matrix(Size s, int InitialValue)*.

Створити клас *Size*, що містить в собі елементи:
- поле «кількість рядків» *int Raw*;
- поле «кількість стовпців» *int Column*;
- конструктор з параметрами *Size(int Raw, int Column)*.

В класі *Matrix* перевантажити оператор ++ (бінарний, дружній оператор), нова дія – перемноження матриць.

Варіант 19

Створити клас *Wood*, що містить в собі такі елементи:

- поле «сорт» *char* Type*;
- поле «вік» *int Age*;
- поле «вологість» *int humidity*;
- метод отримання сорту *int GetType()*;
- метод отримання віку *int GetAge()*;
- метод отримання вологості *int GetHumidity()*;
- конструктор з параметрами *Wood(char* Type, int Age, int humidity)*.

Створити клас *Timber* (брус), що містить в собі елементи:

- поле «довжина» *int Length*;
- поле «площа поперечного перетину» *float Square*;
- метод відпилювання куска *void Truncate(int length)*;
- конструктор з параметрами *Timber(char* Type, int Age, int humidity, int Length, float Square)*.

В класі *Wood* перевантажити оператор – – (унарний, оператор класу), нова дія – зменшення вологості на 5%.

Варіант 20

Створити клас *Mebel*, що містить в собі такі елементи:

- поле «виробник» *char* Vendor*;
- поле «матеріал» *char* Material*;
- метод отримання виробника *char* GetVendor()*;
- метод отримання матеріалу *char* GetMaterial()*;
- конструктор з параметрами *Mebel(char* Vendor, char* Matrial)*.

Успадкувати від класу *Mebel* клас *Table*, що містить в собі елементи:

- поле «кількість ніжок» *int Num*;
- поле «висота» *int Height*;
- поле «ширина» *int Width*;
- поле «довжина» *int Length*;
- метод розрахунку периметра стола *float Perimeter()*;
- конструктор з параметрами *Table(char* Vendor, char* Material, int Num, int Height, int Width, int Length)*.

В класі *Table* перевантажити оператор ++ (унарний, оператор класу), нова дія – повернути площу кришки стола.

7 ПРИКЛАД РОЗРОБКИ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ СИСТЕМИ

В цьому прикладі розглядається проект, який задовольняє мінімальним вимогам **(на оцінку "задовільно").** Вимоги для інших оцінок сформульовані в прикладі "Технічного завдання" (див. додаток Є).

7.1 Постановка задачі

Цей розділ починають із словесного опису предметної області (див. розділ 6), в якому описані базовий і похідний класи.
Створити клас *CelestialBody*, що містить в собі такі елементи:

- поле «вага» *float Mass*;
- поле «радіус» *float Radius*;
- метод отримання ваги *int GetMass()*;
- метод отримання радіуса *int GetRadius()*;
- метод розрахунку щільності речовини небесного тіла *float GetDensity()*;
- конструктор з параметрами *CelestialBody(float Mass, float Radius)*;

Створити клас *Planet*, що містить в собі елементи:

- поле «період оберту навколо осі» *float RotatePeriod*;
- поле «період оберту навколо центра системи» *float RotationPeriod*;
- відстань до центру системи *float Distance*;
- метод отримання швидкості оберту навколо осі *float GetRotatePeriod()*;
- метод отримання швидкості оберту навколо центра системи *float GetRotationPeriod();*
- метод отримання відстані від центра системи *float GetDistance();*
- конструктор с параметрами *Planet(float Mass, float Radius, float RotatePeriod, float RotationPeriod, float Distance);*

В класі *CelestialBody* перевантажити оператор + (бінарний, дружній оператор), нова дія – зливання двох тіл (отримання нового з сумарною вагою й радіусом).

Крім того, тут описуються вхідні дані, обмеження на них, результати і зв'язок між ними. Тут враховуються вимоги до відповідної оцінки з курсової роботи (див. додаток В).

7.2 Об'єктно-орієнтований аналіз і проектування

7.2.1 Об'єктно-орієнтований аналіз

У даному підрозділі описуються, класи які можна виділити з заданої предметної області. Кількість класів визначається вимогами до відповідної оцінки з курсової роботи.

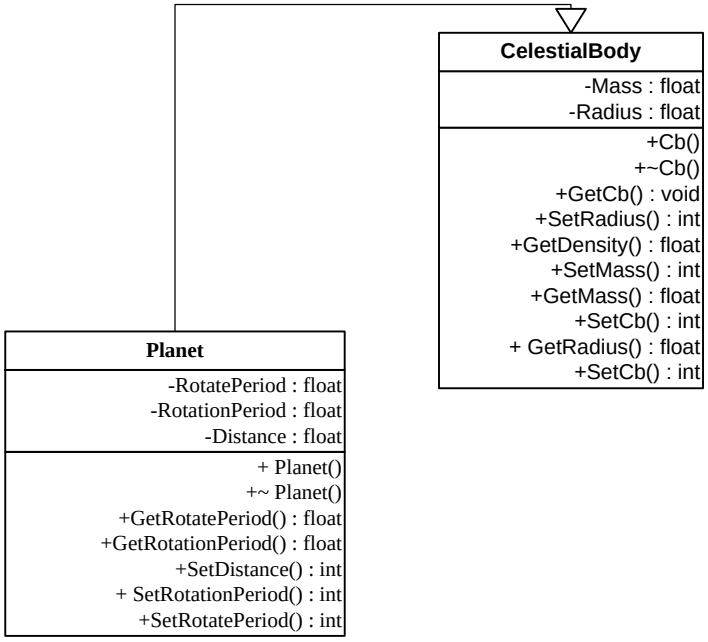
У прикладі для виконання визначеної задачі потребуються класи: базовий *CelestialBody*, похідний від нього *Planet.*

7.2.2 Об’єктно-орієнтоване проектування

7.2.2.1 Діаграми класів

У даному пункті необхідно описати, що таке діаграми класів, їх структуру і призначення.

Діаграми класів відображають класи та відношення між ними, тобто структуру системи, що проектується. На малюнку 7.1 зображено діаграму класів для обраної предметної області. Вона описує класи об’єктно-орієнтованої системи, їх поля та основні методи. Також на цій діаграмі показана взаємодія класів (одиночне спадкування). Кожний клас має Set-методи для встановлення значень полів та Get-методи для для отримання доступу до значень полів.

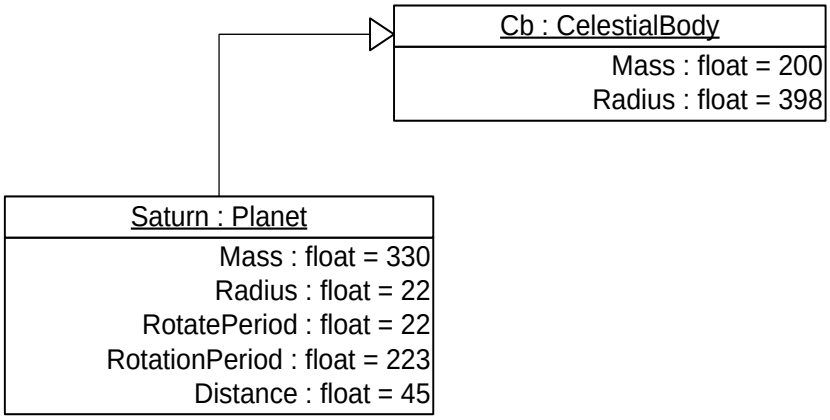


Малюнок 7.1 – Діаграма класів

7.2.2.2 Діаграми об’єктів

У даному пункті необхідно описати, що таке діаграми об’єктів, їх структуру і призначення. Описуються розроблені діаграми об’єктів для описаної у попередньому підрозділі системи класів.

На малюнку 7.2 наведена діаграма об’єктів, яка відображає характерні значення атрибутів об’єктів класів заданої об’єктно-орієнтованої системи. У даному випадку діаграма об’єктів виступає як приклад використання об’єктно-орієнтованої системи та показує які значення можуть приймати поля об’єктів заданих класів.



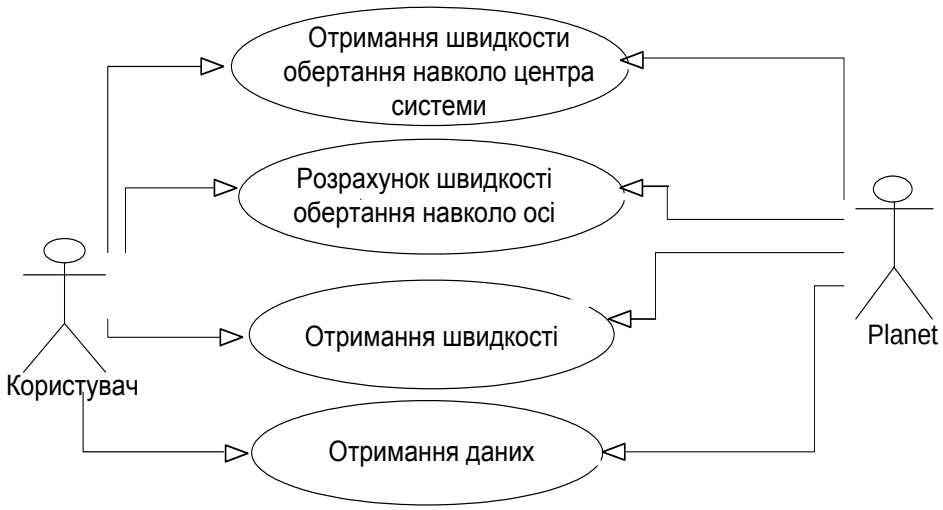
Малюнок 7.2 – Діаграма об’єктів

7.2.2.3 Діаграми прецедентів

У даному пункті необхідно описати, що таке діаграми прецедентів, їх структуру і призначення.

Діаграми прецедентів відображають набір дій, які можуть бути здійснені користувачем у системі за для визначеної мети. Тому діаграми прецедентів для класів *CelestialBody* та *Planet* можуть бути такими як зображено на мал. 7.3 та 7.4. На них зображені основні дії, що може виконувати користувач відносно системи, а також відношення між виконавцями. Ці діаграми чітко ілюструють взаємодію користувача з системою. Стрілки зв’язку вказують, які виконавці зайняті у яких прецедентах.

Малюнок 7.3 – Діаграма прецедентів 1



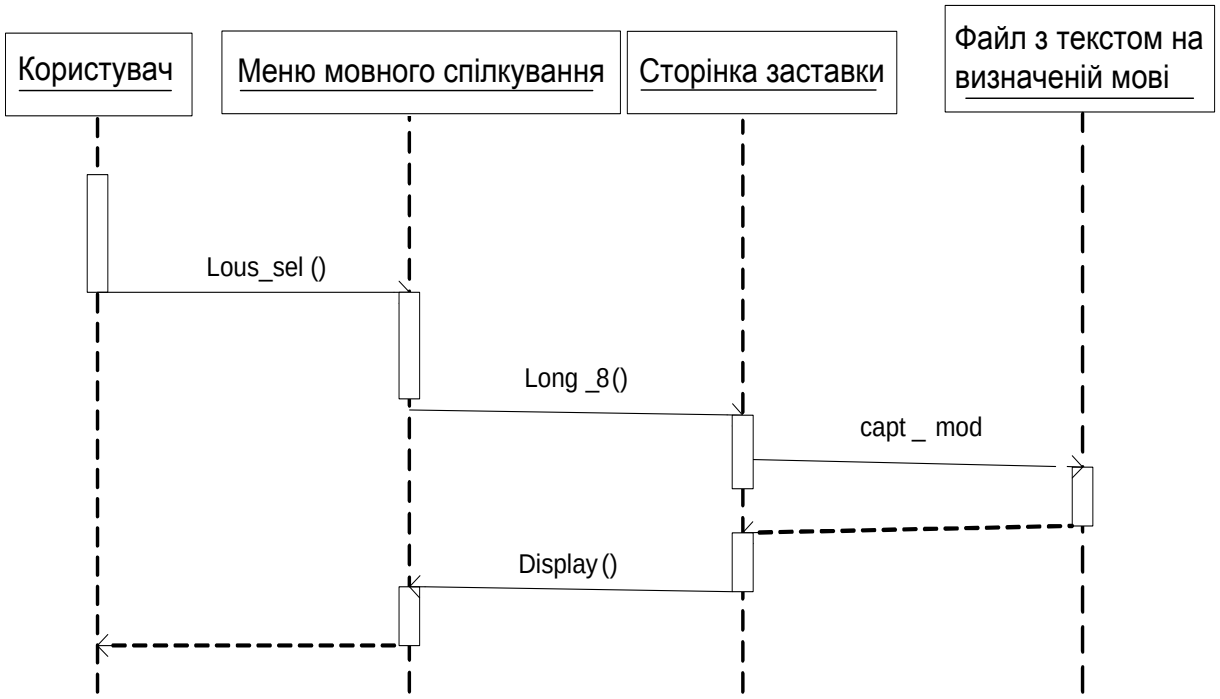
Малюнок 7.4 – Діаграма прецедентів 2

7.2.2.4 Діаграми послідовностей

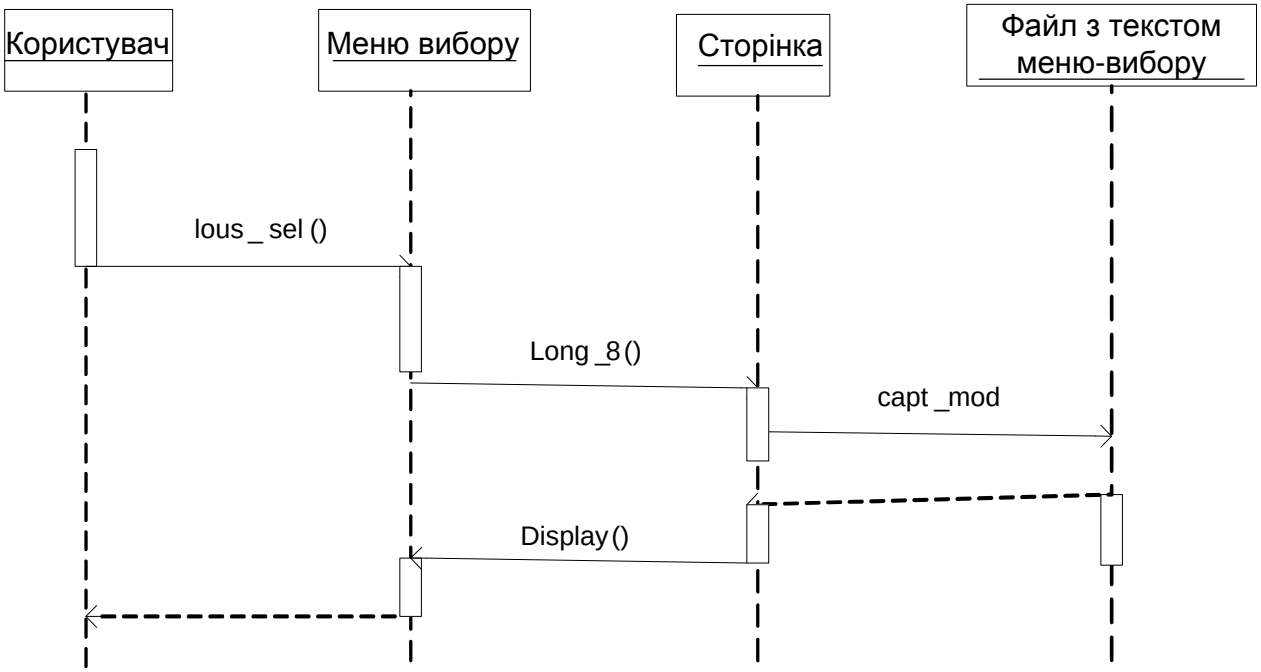
У даному пункті необхідно описати, що таке діаграми послідовностей, їх структуру і призначення.

Діаграми послідовностей відображають порядок повідомлень, що передаються між об’єктами. Вони використовуються для розподілення операцій між класами.

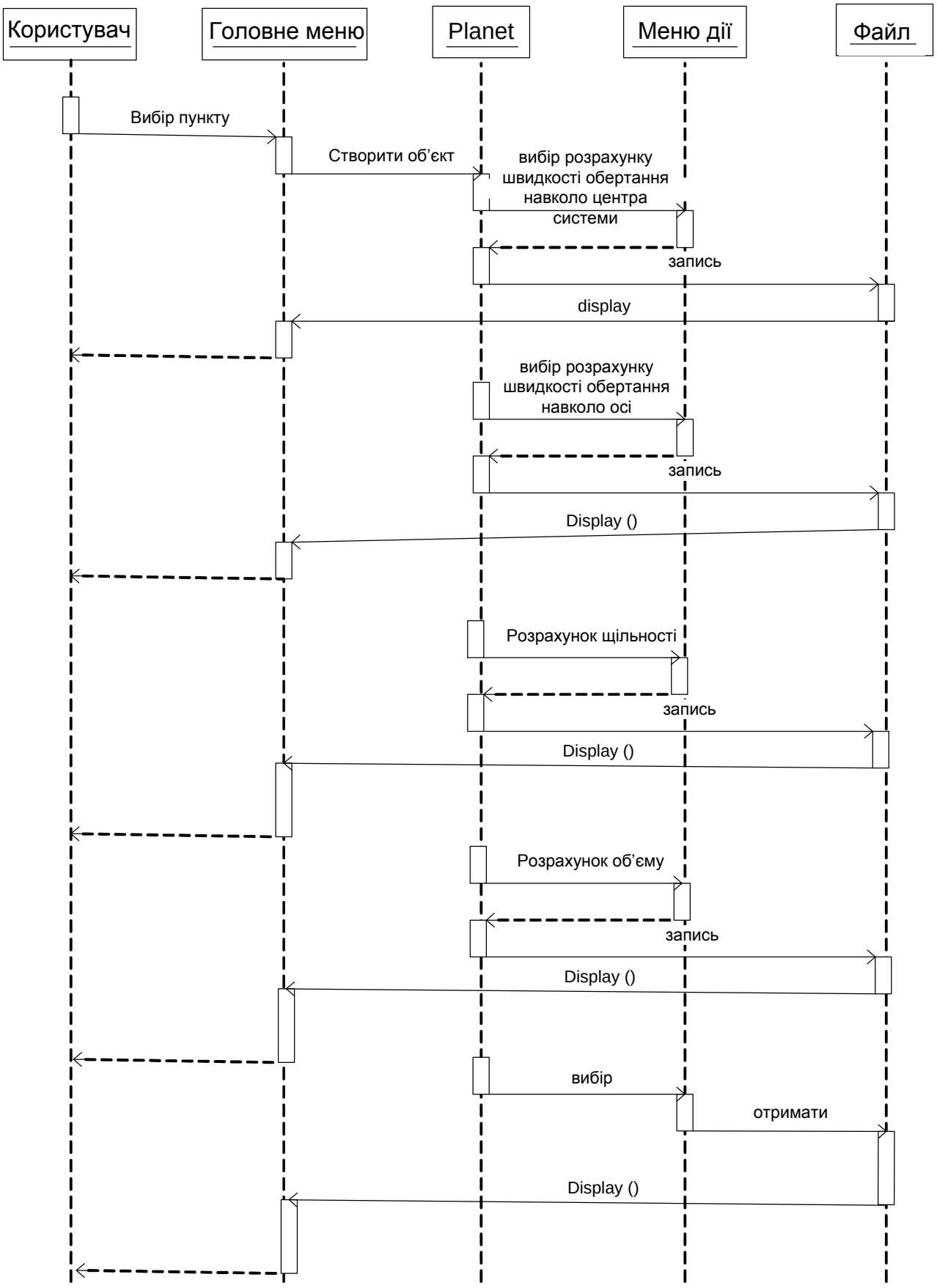
На малюнку 7.5 зображена діаграма послідовностей для роботи з мовної панеллю. На малюнку 7.6 зображена діаграма послідовностей, що відображає «основний потік» роботи користувача з інтерфейсом програми. На малюнку 7.7 зображена діаграма послідовностей, що відображає роботу користувача з одним з класів.



Малюнок 7.5– Діаграма послідовностей 1



Малюнок 7.6 - Діаграма послідовностей 2



Малюнок 7.7 – Діаграма послідовностей 3

7.2.2.5 Діаграми станів

У даному пункті необхідно описати, що таке діаграми станів, їх структуру і призначення.

Діаграми станів показують стан, в якому знаходиться об’єкт у процесі життя, події, на які він реагує, реакції та переходи між станами.

На малюнку 7.8 зображена діаграма станів для об’єктів класу *CelestialBody*, на малюнку 7.9. - для об’єктів класу *Planet.*

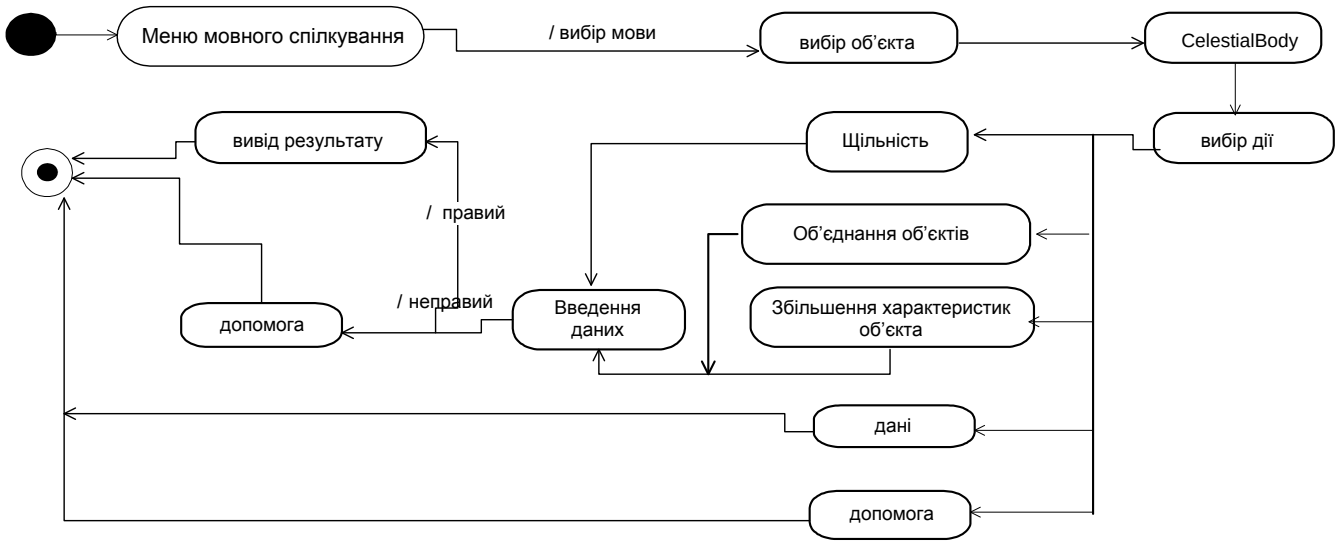


Рисунок 7.8– Діаграма станів для об’єктів класу CelestialBody

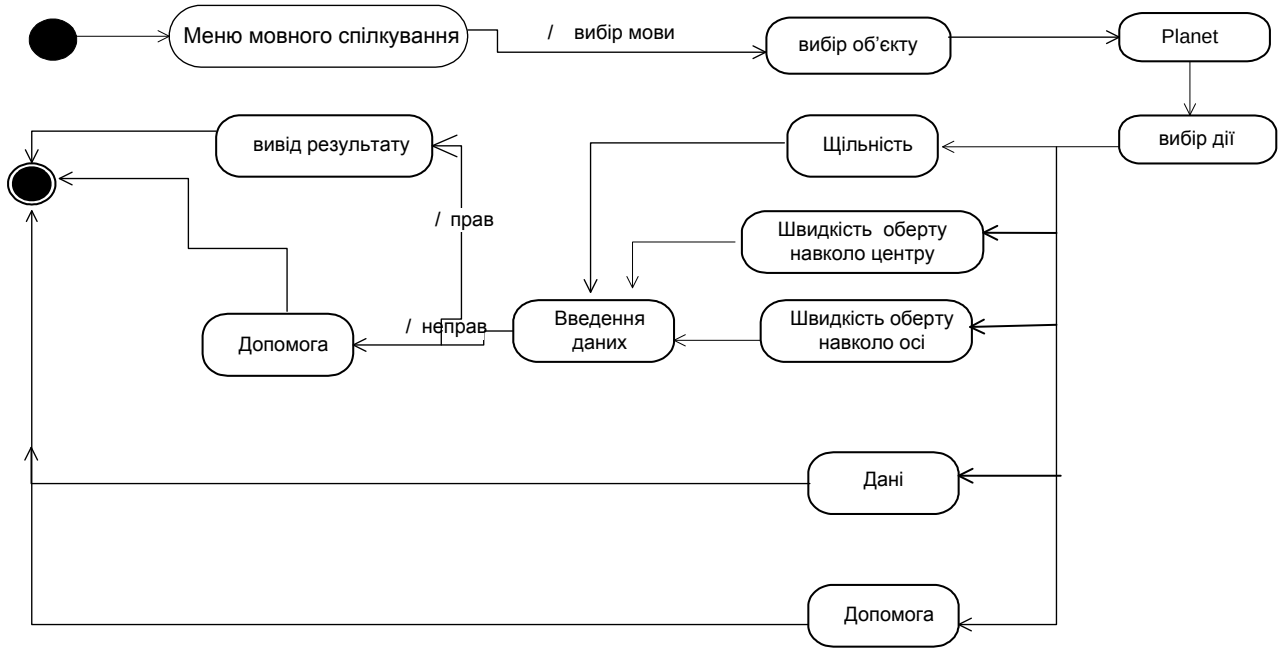


Рисунок 7.9– Діаграма станів для об’єктів класу Planet

7.3 Робоче проектування

7.3.1 Структури даних програми

У цьому підрозділі необхідно описати структури даних які будуть використовуватися у програмі: основні змінні, файли, структури.

Структури даних програми можуть бути описані за допомогою таблиць наведених на малюнках 7.10 та 7.11.

Назва	*Тип*	*Використання*
....	....	....

Малюнок 7.10 – Приклад таблиці „Список змінних”

Назва файлу	*Застосування*	*Мова*
menu_u.txt	Файл головного меню	Українська
menu_r.txt	Файл головного меню	Російська
...	...	...

Малюнок 7.11 – Приклад таблиці „Список ресурсних файлів”

7.3.2 Розробка класів

У цьому підрозділі необхідно описати всі класи в залежності від вимог до оцінки, на яку виконується робота. У розглянутому прикладі розроблено тільки два класи *CelestialBody* і *Planet* (базовий і похідний).

7.3.2.1 Реалізація інкапсуляції

У цьому пункті необхідно описати основні положення інкапсуляції та її у конкретній системі класів відповідно варіанту.

На малюнку 7.12 зображено заголовочний файл класу *Planet.*

```
class Planet : public Cb {
 private: float Distance;
          float RotationPeriod;
          float RotatePeriod;

 public:  Planet();
          Planet(float iMass, float iRadius,     float iRotatePeriod,float iRotationPeriod, float iDistance);
          ~Planet();

          float GetRotatePeriod(Cb *Cb);

           float GetRotationPeriod();
           float operator !();
           float GetRotate();
           float GetRotation();
           float GetDistance();
          int SetDistance(float ADistance);
          int SetRotationPeriod(float ARotationPeriod);
          int SetRotatePeriod(float ARotatePeriod);

};
```

Малюнок 7.12 – Заголовочний файл класу Planet

Елементом інкапсуляції служить те, що його члени розташовані у закритій (*private*) частині класу, таким чином, не має прямого доступу до даних ззовні. Методи *GetRotate(), GetRotation()* та *GetDistance()* дозволяють отримувати доступ до даних ззовні.

Дані скриваються від користувача, але йому доступний інтерфейс, через який він може взаємодіяти з ними.

7.3.2.2 Реалізація спадкування

У цьому пункті необхідно описати основні положення спадкування та його реалізацію у конкретній системі класів відповідно варіанту.

У розглянутій об'єктно-орієнтованій системі реалізовано лише одиночне спадкування (для оцінки „задовільно") (див. мал. 7.13).

```
class Cb
 { float Mass;
  float Radius;
  public:
        Cb();
    ~Cb();
    Cb(float iMass, float iRadius);
    void GetCb( char *buf);
    friend Cb operator + (Cb &cb1, Cb &cb2);
    float GetDensity(float Mass, float Radius);
  int SetRadius(float ARadius);
int SetMass(float AMass);
float GetMass();
int SetCb( char *buf);
float GetRadius();
int SetCb(float AMass, float ARadius);
```

Малюнок 7.13– Заголовочний файл класу CelestialBody

При одиночному спадкуванні у похідного класу всього один предок. Прикладом одиночного спадкування є public-спадкування класу Planet від класу CelestialBody. На малюнку 7.12 зображено заголовочный файл похідного класу Planet, а на малюнку 7.13 зображено реалізацію базового класу CelestialBody. Клас Planet успадковує від класу CelestialBody всі поля (Mass, Radius) і методи. Під час виклику конструктора класу Planet викликається конструктор базового класу CelestialBody, при виклику деструктора також викликається деструктор базового класу. Також клас Planet має свої власні поля й методи, окрім тих, які спадкуються від класу CelestialBody.

7.3.2.3 Реалізація поліморфізму

У цьому пункті необхідно описати основні положення поліморфізму та його реалізацію у конкретній системі класів відповідно варіанту.

Поліморфізм в ООС реалізовано за допомогою перевантаження операторів. В данной системі класів перевантаження операторів представлено у класі *CelestialBody* (див. мал. 7.13).

На малюнку 7.14 приведено реалізацію перевантаження бінарного оператора +, який є функцією-членом класу та виконує злиття двох небесних тіл в одно з сумарними характеристиками.

```
Cb operator + (Cb &cb1, Cb &cb2)
{ float m1,  m2, r1,m,r, r2;
  m1=cb1.GetMass(); r1=cb1.GetRadius();
  m2=cb2.Mass; r2=cb2.Radius;
          m=r=0;
  m=m1+m2;
  r=r1+r2;
  return Cb(m,r);}
```

Рисунок 7.14– Реалізація оператора +

У цьому розділі необхідно привести приклад використання перевантаженого оператора, а також пояснити необхідність та сенс перевантаження. **Для робіт на оцінки “добре” та “відмінно”** необхідно пояснити різницю між перевантаженням унарних та бінарних операторів, використанням дружніх функцій та операторів класу, переваги застосування перевантажених методів, можливості використання абстрактних класів і віртуальних методів.

7.3.3 Тестування працездатності системи класів

У цьому пункті необхідно описати всі можливі виключні ситуації, які може опрацьовувати програмна система, а також навести тести, що відображають роботу програми з різними наборами вхідних даних з прикладами скріншотів роботи програми.

Наприклад, можна розглянути такі виключні ситуації:

– відсутність ресурсного файлу з мовними даними або помилка його відкриття;

– відсутність ресурсного файлу з головним меню або помилка його відкриття;

– відсутність ресурсного файлу з даними про об’єкт або помилка його відкриття;

– неправильний вибір користувачем пункту меню програми і таке інше.

Приклад поведінки програми у даному випадку зображено на малюнку 7.15.

Малюнок 7.15 – Помилка відкриття файлу

Також повинні опрацьовуватися помилки виходу за діапазон правильних числових значень. Поведінка програми у цьому випадку може бути такою, як зображено на малюнку 7.11.

Малюнок 7.16 – Невірні дані

Крім того, треба описати чи можлива робота програми після обробки нею виключних ситуацій.

7.3.4 Обґрунтування вибору структур даних

У цьому підрозділі необхідно пояснити чому використовувались ті, чи інші структури даних у програмі, тобто, наприклад, чому використовувались файли прямого доступу, а не послідовного; чому не були використані структури даних (чи навпаки були використані), динамічні структури і таке інше.

7.3.5 Обґрунтування вибору алгоритмів

Відомо, що одна задача може вирішуватися різними способами, тому у цьому підрозділі необхідно навести основні алгоритми розробленої системи, та пояснити чому вони були використані, які мають переваги.

На малюнку 7.17 зображено основний алгоритм роботи програми для системи класів *CelestialBody* та *Planet*. Нацьому малюнку:

а) блок 1 – читання з файлів мовних даних, вибір мови;

б) блок 2 – вивід файлу головного меню;

в) блоки 3, 4 – вибір об’єкта, вивід меню об’єкта та вибір дії.

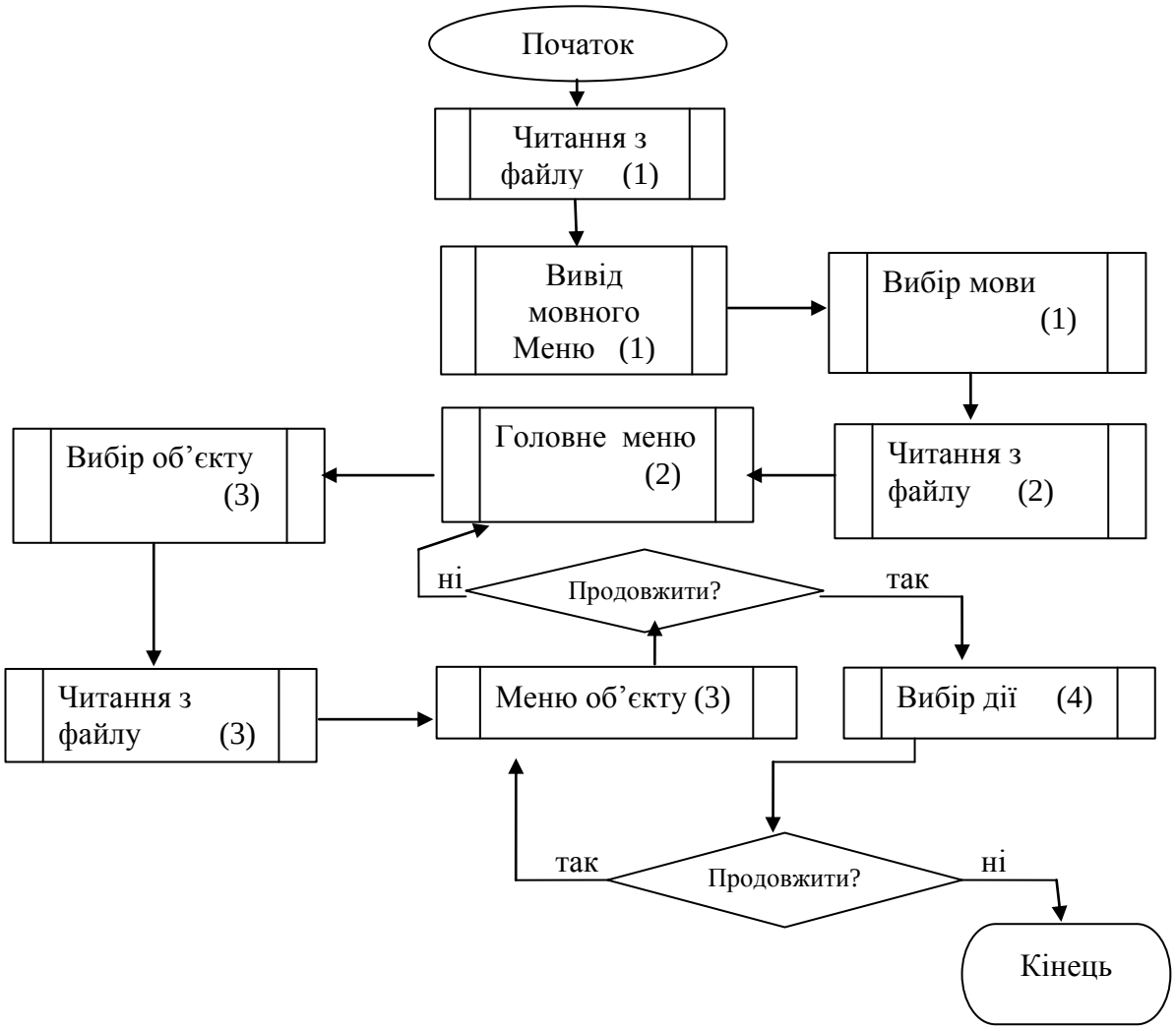
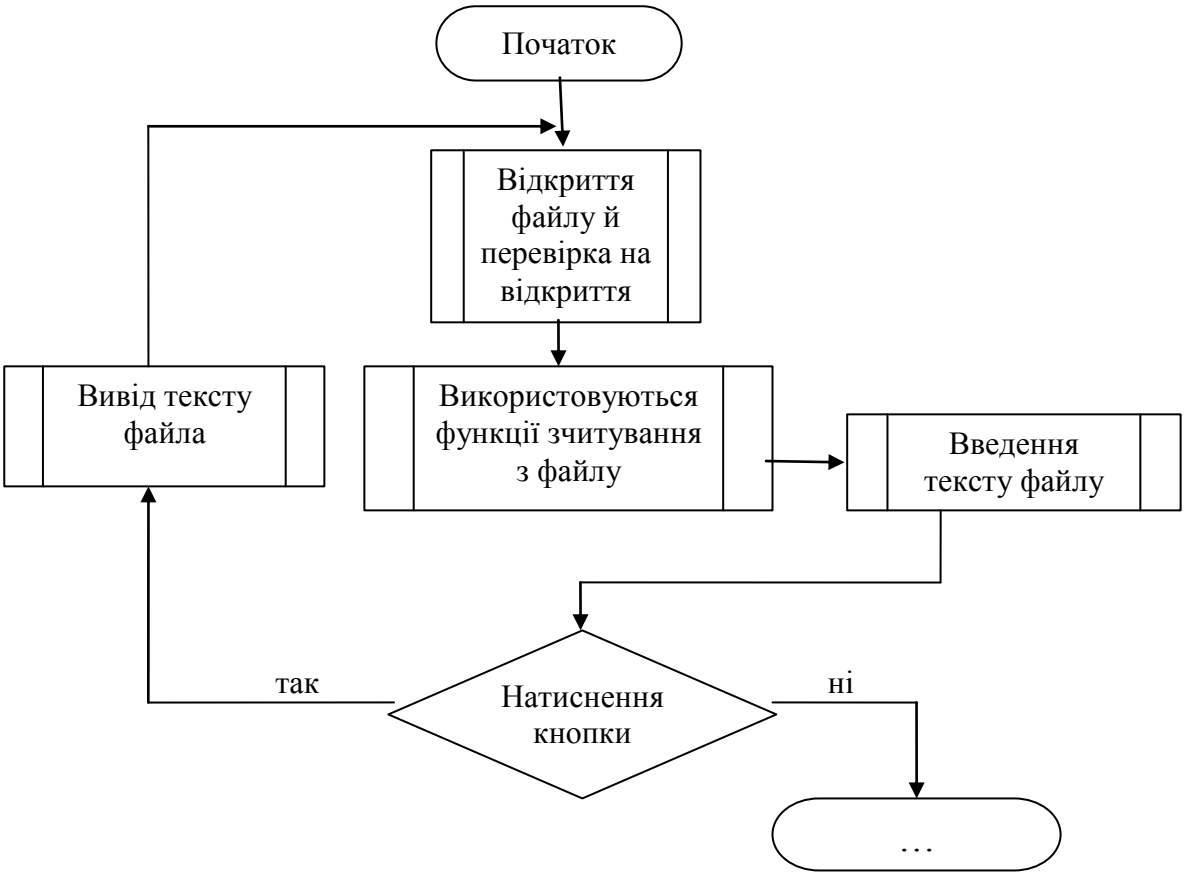


Рисунок 7.17 – Основний алгоритм

На малюнку 7.18 приведений алгоритм читання з файлу та вибір мови. На першому етапі відкривається файл на читання, перевіряться на відкриття та зчитується інформація з файлу.



Малюнок 7.18 – Алгоритм роботи мовного меню

ПЕРЕЛІК ПОСИЛАНЬ

1. Единая система программной документации. - М.: Государственный комитет СССР по стандартам, 1982. - 128 с.
2. Скотт К. UML. Основные концепции. – М.: Издательский дом “Вильямс”, 2002. – 144с.
3. Страуструп Б. Язык программирования С++. - М.: Бином, Невский Диалект, 2004. - 1104с.
4. Прата С. Язык программирования С++. Лекции и упражнения. Учебник. - М.: Диасофт, 2005. - 1102с.
5. Павловская Т.А. С/С++. Программирование на языке высокого уровня. - СПб.: Питер, 2006. - 461с.
6. Подбельский В.В. Язык Си ++. Учебное пособие. - М.: Финансы и статистика, 2008. – 560с.
7. Франка П. С++: учебный курс. - СПб.: Питер, 2006. - 528с.
8. Дейтел Х.М., Дейтел П.Дж. Как программировать на С++. - М.: Бином. Лаборатория знаний, 2003. - 1152 с.
9. Методические указания и задания к выполнению курсовой работы по дисциплине "Основы программирования и алгоритмические языки (для студентов специальности "Программное обеспечение автоматизированных систем") / Сост .Н.Н. Дацун. - Донецк, ДонГТУ, 2001. - 56с.
10. Методичні вказівки і завдання до лабораторних робіт із курсу «Основи програмування» (для студентів напряму підготовки 6.050103 «Програмна інженерія» денної форми навчання і очно-заочної форми навчання з наданням денних послуг). Част. 1./ Укладачі: Н.М. Дацун, І.О. Коломойцева. - Донецьк, ДонНТУ, 2010. - 128 с.
11. Конспект лекцій з дисципліни «Об’єктно-орієнтоване програмування» (для студентів напряму підготовки 6.050103 «Програмна інженерія»/ Укладач: Н.М. Дацун. - Донецьк, ДонНТУ, 2010. - 120 с.
12. http://dist.donntu.edu.ua/course/view.php?id=13 - Дистанційний навчальний курс “Об'єктно-орієнтоване програмування”
13. Методические указания и задания к выполнению лабораторных работ по дисциплине "Объектно-ориентированное программирование" (для студентов направления подготовки 6.050103 «Программная инженерия» всех форм обучения)./ Сост. Дацун Н.Н., Попов Ю.В., Ковальский С.В., Грищенко Д.А.. - Донецк, ДонНТУ, 2010 - 84с.

ДОДАТОК А

Завдання на курсову роботу
студента_____(прізвище, ім'я, по батькові)_групи_____

1 Тема роботи _____
_____
2 Термін здачі студентом закінченої роботи _____ “__.__.____”
(тиждень) (дата)
3. Вхідні дані для роботи: _____
_____
4. Зміст пояснювальної записки:_____
_____
_____
5 Перелік графічного матеріалу: _____
_____
_____
_____
6. Дата видачі завдання _____.02. 20___

ГРАФІК ВИКОНАННЯ КУРСОВОЇ РОБОТИ

№	Найменування етапу	Термін виконання	
		тиждень	дата
1	Видача завдання на курсову роботу. З'ясування завдання	1	
2	Постановка задачі. Визначення вимог до програми. а) запис постановки завдання; б) складання технічного завдання та його затвердження.	1-2: 2 2	
3	Об'єктно-орієнтований аналіз та проектування: а) об'єктно-орієнтований аналіз; б) проектування структури класів; в) проектування відносин між класами; г) проектування структури об'єктів та їх поведінки.	3-8: 3-4 4-5 5-6 6-8	
4	Технічне проектування: а) модульний аналіз: визначення структури програми, її модулів та їх взаємозв'язків; б) розробка основного алгоритму функціонування програми; в) створення специфікації модулів.	5-9: 5-6 7 8-9	
5	Робоче проектування: а) визначення структур даних і розробка алгоритмів роботи модулів; б) розробка класів: 1) реалізація інкапсуляції; 2) реалізація спадкування; 3) реалізація поліморфізму.	5-10: 5-6 6-7 7-9 9-10	
6	Написання програми	9-12	
7	Налагодження програми	11-13	
8	Комплексне налагодження і тестування	13	
9	Написання пояснювальної записки	14	
10	Захист курсової роботи	15	

Студент _____(підпис) Керівник _____(підпис)
“___” лютого 20___ р.
(дата)

ДОДАТОК Б
ПРИКЛАД “ТЕХНІЧНОГО ЗАВДАННЯ” НА КУРСОВУ РОБОТУ
З КРИТЕРІЯМИ ОЦІНЮВАННЯ

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ВДНЗ “ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ”
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

Затверджую
Башков Є.О.
_____
__.02.20__ р.

ТЕХНІЧНЕ ЗАВДАННЯ
до курсової роботи з дисципліни
“Об'єктно-орієнтоване програмування”
на тему: “Об'єктно-орієнтована система
‘Торговий центр та магазин самообслуговування’ “

Керівник:
_____ каф. ПМІ
_____
__.02.20__ р.

Виконав:
студент(ка) гр.ПС-__
_____
__.02.20__ р.

Донецьк 20__

ВВЕДЕНИЕ

На современном этапе развития программного обеспечения все большая роль отводится объектно-ориентированному стилю проектирования и программирования (ООП). Это позволяет разрабатывать большие программные комплексы быстрее, используя повторно имеющие модули, обеспечивать инкапсуляцию данных, полиморфизм операций, наследование свойств и методов. Ведущие фирмы по разработке прикладного и системного ПО выполняют свои разработки средствами ООП. Современные операционные системы, например, платформа .NET имеет внутренний объектно-ориентированный язык и аппаратную поддержку отдельных его компонент. Поэтому изучение и практическое освоение специалистами по программной инженерии этого стиля программирования является актуальной задачей.

Профессиональное обучение программированию по направлению подготовки “Программная инженерия” предусматривает освоение языка Си в качестве базового. Дальнейшее развитие навыков программирования на языке Си и изучение основ ООП выполняется в дисциплине “Объектно-ориентированное программирование”. Для закрепления навыков проектирования и разработки больших программ в объектно-ориентированном стиле учебным планом направления подготовки “Программная инженерия” предусмотрено выполнение курсовой работы по этой дисциплине.

Целью курсовой работы является закрепление практических навыков самостоятельной постановки и решения задачи обработки данных с помощью ЭВМ средствами ООП. Во время выполнения курсовой работы студент должен овладеть методикой объектно-ориентированного проектирования программ по формализации и решению поставленной задачи, технологическими приемами разработки объектно-ориентированных программ на языке Си++.

Условием успешного выполнения курсовой работы являются практические навыки ООП, полученные при выполнении лабораторных работ по дисциплине “ООП”.

Разработанная программа market.exe является объектно-ориентированной системой для предметной области “Торговый центр и магазин самообслуживания”. Она предназначена для использования в высших учебных заведениях с целью демонстрации знаний, приобретенных обучаемым в области технологии разработки объектно-ориентированных программ средствами языка Си++.

1 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Курсовая работа выполняется на основании “Задания на курсовую работу” по дисциплине “Объектно-ориентированное программирование” для студентов специальности “Программная инженерия”, выданное кафедрой “Прикладной математики и информатики” ДонНТУ.

2 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Программное изделие market.exe является информационной системой типа “Объектно-ориентированная система”.

Должно быть разработано программное, которое может работать в следующих режимах:

а) демонстрация описания классов Market и SelfM и работоспособности функций-членов класса:

1) конструкторов;
2) функций-членов Getxx (Get-методов);
3) функций-членов Setxx (Set-методов);
4) деструктора;
5) функций-членов Actionxx;

б) демонстрация работоспособности класса Market при перегрузке бинарного оператора -=;

в) демонстрация работоспособности класса SelfM при перегрузке унарного оператора !;

г) демонстрация работоспособности классов Market и SelfM при одиночном наследовании;

д) система помощи:

1) о программе (программный документ ”Описание программы”);

2) руководство оператора (программный документ “Руководство оператора”);

3) об авторе.

Программное изделие должно функционировать в среде MS Windows версии не ниже XP.

3 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

3.1 Требования к функциональным характеристикам

Программное изделие должно удовлетворять следующим общим требованиям:

а) использование технологии раздельной компиляции файлов;

б) режим работы монитора для программы – текстовый (консольное приложение);

в) использование файлов для хранения всех данных;

г) обеспечение двуязычия в общении с пользователем.

Критерии оценивания программного изделия по степени выполнения требований технологии раздельной компиляции:

а) для оценки «удовлетворительно»: отсутствие файла проекта, объединение исходных файлов с помощью директив препроцессора;

б) для оценок «хорошо» и «отлично»: использование файла проекта и технологии раздельной компиляции файлов;

Критерии оценивания программного изделия по степени выполнения требований для инкапсуляции:

а) для оценки «удовлетворительно»: реализованы только два класса, описание которых представлено в словесном описании предметной области;

б) для оценки «хорошо»:

1) реализованы классы, описание которых представлено в словесном описании предметной области;

2) реализованы дополнительные классы, расширяющие словесное описание предметной области.

в) для оценки «отлично»:

1) реализованы два класса, описание которых представлено в словесном описании предметной области;

2) реализованы дополнительные классы, расширяющие словесное описание предметной области и возможности интерфейса программы;

3) для каждого из классов предусмотрена возможность подсчета количества объектов, существующих в текущий момент в программе.

Критерии оценивания программного изделия по степени выполнения требований для наследования:

а) для оценки «удовлетворительно»: реализованы только два класса, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

б) для оценки «хорошо»:

1) реализованы классы, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

2) реализованы дополнительные классы - иерархия классов и/или контейнерные классы (только одиночное наследование);

в) для оценки «отлично»:

1) реализованы классы, описание которых представлено в словесном описании предметной области, и указанные отношения между ними;

2) реализованы дополнительные классы - иерархия классов и/или контейнерные классы (одиночное наследование);

3) реализовано множественное наследование.

Критерии оценивания программного изделия по степени выполнения требований для полиморфизма:

а) для оценки «удовлетворительно»: реализована перегрузка только тех операторов, описание которых представлено в словесном описании предметной области;

б) для оценки «хорошо»:

1) реализована перегрузка операторов, описание которых представлено в словесном описании предметной области;

2) реализована перегрузка операторов для некоторых дополнительных классов;

3) перегрузка операторов выполнена как внутри, так и вне класса.

в) для оценки «отлично»:

1) реализована перегрузка только тех операторов, описание которых представлено в словесном описании предметной области;

2) реализована перегрузка операторов для всех дополнительных классов;

3) перегрузка операторов выполнена как внутри, так и вне класса.

4) в реализации полиморфизма использованы абстрактные классы и виртуальные функции.

Критерии оценивания программного изделия по степени выполнения требований к демонстрации работы системы:

а) для оценки «удовлетворительно»: все результаты работы выводятся в системный поток вывода;

б) для оценок «хорошо» и «отлично»:

1) результаты работы выводятся в системный поток вывода;

2) состояние объектов системы выводится в файловый поток вывода.

Критерии оценивания программного изделия по степени выполнения требований в режиме помощи:

а) для оценки «удовлетворительно»: использование файла с условием задачи и файла помощи;

б) для оценок «хорошо» и «отлично» - использование файлов:

1) о программе (программный документ “Описание программы”);

2) руководство оператора (программный документ);

3) об авторе.

Критерии оценивания программного изделия по степени выполнения требований по организации работы с файлами:

а) для оценки «удовлетворительно»:

1) путь доступа файлам данных установлен программным путем (является константой в программе);

б) для оценки «хорошо»:

1) путь доступа к файлам определяется пользователем на этапе выполнения программы (наличие в главном меню проекта опции настройки, в которой пользователь указывает путь доступа к файлам данных и/или их имена);

в) для оценки «отлично»:

1) путь доступа к файлам определяется пользователем на этапе выполнения программы (наличие файла конфигурации с данными о пути доступа к файлам данных и/или их именах);

2) предусмотрена обработка исключительных ситуаций при работе с файлами.

Критерии оценивания программного изделия по степени выполнения требований по обеспечению двуязычия интерфейса пользователя:

а) для оценки «удовлетворительно»:

1) приложение имеет фиксированную пару языков общения (тексты надписей являются константами в программе);

2) переключение языков реализовано только в одном кадре интерфейса;

3) двуязычие реализовано не для всех режимов работы;

б) для оценок «хорошо» и «отлично»:

1) приложение имеет произвольную пару языков общения (тексты надписей читаются из файлов);

2) переключение языков реализовано на каждом кадре интерфейса;

3) двуязычие реализовано для всех режимов работы;

3.2 Требование к надежности

Программное изделие для обеспечения надежности функционирования должно:

а) проверять наличие всех файлов данных;

б) обеспечить минимизацию количества информации, вводимой пользователем;

в) контролировать корректность ввода данных пользователем;

г) обрабатывать исключительные ситуации, вызванные не корректностью ввода данных пользователем с целью предотвращения прерывания выполнения программы.

3.3 Условия эксплуатации

Пользователь должен иметь квалификацию не ниже “оператор ПЭВМ”.

Для эксплуатации программного изделия необходимо наличие программиста, в функции которого входит внесение изменений в информационную часть системы.

3.4 Требования к составу и параметрам технических средств

Для функционирования программного изделия необходим персональный компьютер со стандартным набором периферийных устройств (монитор, клавиатура, мышь).

3.5 Требования к информационной и программной совместимости

Курсовая работа должна быть выполнена на языке Си++ в операционной системе MS Windows версии не ниже XP.

4 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Пояснительная записка по курсовой работе должна сопровождаться следующими программными документами:

а) описание программы;

б) руководство оператора;

в) текст программы.

5 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Перечень этапов и сроки приведен в табл.5.1

Таблица 5.1 - График выполнения курсовой работы

№	Наименование этапа	Срок выполнения	
		неделя	дата
1	Выдача задания на курсовую работу. Уяснение задания	1	
2	Постановка задачи. Определение требований к программе. а) запись постановки задачи; б) составление технического задания и его утверждение.	1-2: 2 2	
3	Объектно-ориентированный анализ и проектирование: а) Объектно-ориентированный анализ б) проектирование структуры классов в) проектирование отношений между классами г) проектирование структуры и их поведения объектов	3-8: 3-4 4-5 5-6 6-8	
4	Техническое проектирование: а) модульный анализ: определение структуры программы, её модулей их взаимосвязей; б) разработка основного алгоритма функционирования программы; в) составление спецификации модулей.	5-9: 5-6 7 8-9	
5	Рабочее проектирование: а) определение структур данных и разработка алгоритмов работы модулей, б) разработка классов: 1) реализация инкапсуляции; 2) реализация наследования; 3) реализация полиморфизма.	5-10: 5-6 6-7 7-9 9-10	
6	Написание программы	9-12	
7	Отладка программы	11-13	
8	Комплексная отладка и тестирование	13	
9	Написание пояснительной записки	14	
10	Защита курсовой работы	15	

6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМА

Курсовая работа выполняется 14 недель. Пояснительная записка по курсовой работе предоставляется на проверку преподавателю не менее чем за 3 рабочих дня до даты защиты.

Защита происходит в присутствии комиссии в составе 2-3 человек и включает:

а) доклад (до 3 минут), отражающий все этапы выполнения курсовой работы;

б) презентацию программы (с демонстрацией всех ее функций);

в) ответы на вопросы комиссии.

ПРИЛОЖЕНИЕ А
СЛОВЕСНОЕ ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

ПРИЛОЖЕНИЕ Б
СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

ДОДАТОК В
ТЕКСТ КОДУ ПРОГРАМИ ООС “ТОРГОВИЙ ЦЕНТР ТА МАГАЗИН САМООБСЛУГОВУВАННЯ” З ВИКОНАННЯМ МІНІМАЛЬНИХ ВИМОГ

```
// Файл CLASS1.HPP
// опис базового класу
class CMarket
 {
  protected:
    char *mName;       // назва
    float mSq;         // торгова площа
    int mCZals;        // кількість залів
  public:
    CMarket();
    CMarket(char * AName, float ASq,
                int ACZals);
    char * GetName();
    float GetSq();
    int GetCZals();
    void GetMarket(char *buf);
    int SetMarket(char * AName, float ASq,
                    int ACZals);
    int SetName( char * AName);
    int SetSq(float ASq);
    int SetCZals(int ACZals);
    ~CMarket();
    void print();
    void operator-=(CMarket&);
 };

//++++++++++++++++++++++++++++
// Файл CLASS2.HPP
// опис похідного класу
#include "class1.hpp"
class CSelfM : public CMarket
 {
  private:
    int mSt;     // кількість автостоянок
    // вартість зберігання в камері схову:
    int mCost;
  public:
    CSelfM(char * AName, float ASq,
              int ACZals, int ASt, int ACost);
    int GetSt();
    int GetCost();
    void GetSelfM(char * buf);
    int SetSt(int ASt);
    int SetCost(int ACost);
    int SetSelfM(int ASt, int ACost);
    ~CSelfM();
    void print();
    friend operator!(CSelfM& sm);
 };
```

```
// Файл CLASS1.CPP
// реалізація базового класу
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
#include <iomanip.h>
#include "class1.hpp"
#include "class-c.hpp"

// конструктори
CMarket:: CMarket()
 {
  mName= new char [4];
  strcpy(mName, "new");
  mSq=20.0;
  mCZals=1;
 }
CMarket:: CMarket(char * AName,
                    float ASq, int ACZals)
 {
  mName= new char [strlen(AName)+1];
  strcpy(mName, AName);
  mSq=ASq;
  mCZals=ACZals;
 }

// GET-методи
char * CMarket::GetName()
 {
  return mName;
 }
float CMarket::GetSq()
 {
  return mSq;
 }
int CMarket::GetCZals()
 {
  return mCZals;
 }
void CMarket::GetMarket(char *buf)
 {
  sprintf(buf, "|%-15s|%10f.2|%5d|",
           mName, mSq, mCZals);
 }
```

```
// SET-методи
int CMarket::SetName( char * AName)
 {
  int lenA = strlen(AName);
  if(lenA==0 || lenA >=len_name)
    {cerr<< "ERROR Name !!!"<< endl;
     return 0;
    }
  mName =  new char [lenA+1];
  strcpy(mName, AName);
  return 1;
 }
int CMarket::SetSq(float ASq)
 {
  if(ASq >0)
    {mSq=ASq;
     return 1;
    }
  else
   {cerr<<"ERROR Square" <<endl;
    return 0;
   }
 }
int CMarket::SetCZals(int ACZals)
 {
  if(ACZals >0)
    {mCZals=ACZals;
     return 1;
    }
  else
   {cerr<<"ERROR Counter Showroom" <<endl;
    return 0;
   }
 }
int CMarket::SetMarket(char * AName,
                       float ASq, int ACZals)
 {
  return SetName(AName) && SetSq(ASq)
         && SetCZals(ACZals);
 }

// метод візуалізації стану об'єкта
void CMarket::print()
 {
  cout.setf(ios::left);
  cout.width(10); cout<<"CMarket: ";
  cout.width(20); cout<< mName;
  cout.width(10); cout.precision(2);
  cout<<mSq;
  cout.width(10);
  cout <<mCZals<<endl;
 }

// деструктор
CMarket::~CMarket()
 {
  delete mName;
 }

// перевантаження бінарного оператора -=
void CMarket::operator-=(CMarket& m)
 {
  int z;
  z=m.GetCZals();
  int newz= mCZals-z;
  if(newz>0)
  mCZals=newz;
 }
//+++++++++++++++++++++++++++
// Файл CLASS2.CPP
// реалізація похідного класу
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include <iomanip.h>
#include "class2.hpp"

// конструктори
CSelfM::CSelfM(char * AName, float ASq,
               int ACZals, int ASt, int ACost)
          :CMarket(AName, ASq, ACZals)
 {
  if(ASt<=0 || ACost<=0)
    {cerr << "The number of parking and"<<
      " the cost of lockers should be greater"<<
      " than 0"  << endl;
     exit(-1);
    }
  mSt=ASt;
  mCost=ACost;
 }

// GET-методи
int CSelfM :: GetSt()
 {
  return mSt;
 }
int CSelfM ::GetCost()
 {
  return mCost;
 }
void CSelfM ::GetSelfM(char * buf)
 {
  sprintf(buf, "|%10d|%10d|", mSt, mCost);
 }
```

```
// SET-методи
int CSelfM ::SetSt(int ASt)
 {
  if(ASt<=0 )
    {cerr<<"ERROR number of parking"<<endl;
     return 0;
    }
  else
   {mSt=ASt;
    return 1;
   }
 }
int CSelfM ::SetCost(int ACost)
 {if(ACost<=0 )
    {cerr<<"ERROR cost of lockers" << endl;
     return 0;
    }
  else
    {mCost=ACost;
     return 1;
    }
 }
int CSelfM ::SetSelfM(int ASt, int ACost)
 {
  return SetSt(ASt) && SetCost(ACost);
 }
// деструктор
CSelfM::~CSelfM()
 { }
// метод візуалізації стану об'єкта
void CSelfM::print()
 {CMarket::print();
  cout.setf(ios::left);
  cout.width(10); cout << "CSelfM: ";
  cout.width(10); cout<<mSt;
  cout.width(10); cout<< mCost<< endl;
 }
// перевантаження унарного оператора !
operator! (CSelfM& m)
 {int oldCost;
  oldCost=m.GetCost();
  int newCost=oldCost *2;
  m.SetCost(newCost);
 }
//++++++++++++++++++++++++++++
//  Файл CLASS-C.HPP
// визначення зовнішньої змінної
int len_name=30;
//++++++++++++++++++++++++++++
//  Файл CLASS-E.HPP
// декларація зовнішньої змінної
extern int len_name;
```

```
// Файл MAIN.CPP
// використання класів
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include "class2.hpp"
#include "class-e.hpp"
// прототипи функцій
void cr_base();
void pr_base(int n);
void cr_child();
void pr_child();
void ov_un();
void ov_bin();
void main_menu();
void uslovie();
void help();
void zastavka();

// оголошення об'єктів
CMarket *market;
CMarket *market2;
CSelfM * self;
// рядок повідомлення
char any_key[]="Press any key to continue...";
// запрошення в меню об'єктів
void menu_prompt()
 {
  clrscr();
  cout<<"1. Create an object base class:"<<endl;
  cout<<"2. Show the object base class:"<<endl;
  cout<<"3. Create the object derived class:"<<endl;
  cout<<"4. Show the object derived class:"<<endl;
  cout<<"5. Overload unary operator:"<<endl;
  cout<<"6. Overload binary operator:"<<endl;
  cout<<"7. End of working with objects"<<endl;
  cout<<"Enter the number of modes: "<<
          "1, 2, 3, 4, 5, 6, 7."<<endl;
 }
// робота меню об'єктів
void menu_reply()
 {
  int reply;
  int pr=1, prbase=0, prchild=0;
  while(1)
    {
     menu_prompt();
     cin>>reply;
```

```
switch(reply)
 {// створення об'єкта базового класу
  case 1:
    if(!prbase)
     {cr_base(); pr=2; prbase=1;break;}
    else
     break;
  // показ стану об'єкта базового класу
  case 2:
    if(prbase)
     {pr_base(1); pr=3; break;}
    else
     {cerr<<"ERROR: Can not print a "<<
            "status object base class. "<<
            "Select option 1. "<<endl;
      pr=1;
      cerr<< any_key <<endl; getch();
      break;
     }
  // створення об'єкта похідного класу
  case 3:
    if(pr>=2 || pr==1)
     {cr_child(); pr=4; prchild=1;
      break;}
  // показ стану об'єкта похідного класу
  case 4:
    if(prchild)
     {pr_child(); pr=5; break;}
    else
     {cerr<<"ERROR: Can not print a "<<
            "status object derived class. "<<
            "Select option 3. "<<endl;
      pr=3;
      cerr<< any_key <<endl; getch();
      break;
     }
  // перевантаження унарного оператора
  case 5:
    if(prchild)
     {ov_un(); pr=6; break;}
    else
     {cerr<<"ERROR: can not overload "<<
            "a unary operator. "<<
            " Select option 3." <<endl;
      pr=3;
      cerr << any_key << endl; getch();
      break;
     }
  //  перевантаження бінарного оператора
  case 6:
    if(prbase)
     {ov_bin(); pr=7; break;}
    else
     {cerr<<"ERROR: can not overload "<<
            "a binary operator. "<<
            " Select option 1." <<endl;
      pr=1;
      cerr << any_key << endl; getch();
      break;
     }
    // повернення до меню
   case 7: zastavka();
    // помилка
   default: cerr<<"ERROR in the index "<<
            "number of modes!!!"<<endl;
    cout << any_key << endl; getch();
  } // switch
 } // while
}
// створення об'єкта базового класу
void cr_base()
{cout <<"Created the object base class "<<
       " CMarket"  <<endl;
 market= new CMarket("Donetsk-City",
                     200.0, 50);
 cout << any_key << endl;  getch();
}
// показ стану об'єктів базового класу
void pr_base(int n)
{if(n==1)
   {cout<<"The first object base class:" <<endl;
    market->print();
   }
 else
  {cout <<"Second object base class:"  <<endl;
   market2->print();
  }
 cout << any_key << endl;  getch();
}
// створення об'єкта похідного класу
void cr_child()
{cout <<"Created the object derived class "<<
       " CSelfM"  <<endl;
 self= new CSelfM("Metro", 500.0, 2, 1, 10);
 cout << any_key << endl;  getch();
}
// показ стану об'єктів похідного класу
void pr_child()
{cout <<" Object derived class:"  <<endl;
 self->print();
 cout << any_key << endl;  getch();
}
```

```
// перевантаження бінарного оператора
void ov_bin()
 {
  // створеня другого об'єкту
  market2 = new CMarket("Gold Ring",
                        150.0, 30);
  pr_base(2);
  *market-= *market2;
  cout<<" Overload binary operator "<<endl;
  pr_base(1);
 }
// перевантаження унарного оператора
void ov_un()
 {
  !(*self);
  cout<<" Overload unary operator "<<endl;
  self->print();
  cout << any_key << endl;  getch();
 }
// головний кадр інтерфейсу
void zastavka()
 {
  clrscr();
  cout <<endl;
  cout.width(45); cout.setf(ios::right);
  cout<<"MES of Ukraine"<<endl;
  cout.width(42); cout.setf(ios::right);
  cout<<"DonNTU"<<endl;
  cout.width(45); cout.setf(ios::right);
  cout<<" AMI"<<endl;
  cout<<endl<<endl<<endl<<endl;

  cout.width(47); cout.setf(ios::right);
  cout<<"Coursework"<<endl;
  cout.width(60); cout.setf(ios::right);
  cout<<"in the discipline 'OOP' "<<
        " variant №30"<<endl;
  cout.width(67); cout.setf(ios::right);
  cout<<"Topic: Shopping mall and "<<
        " a supermarket"<<endl;
  cout<<endl<<endl<<endl<<endl;

  cout.width(70); cout.setf(ios::right);
  cout<<"work checked: "<<
     "                work performed:"<<endl;
  cout.width(70); cout.setf(ios::right);
  cout<<"AP. kaf. AMI"<<
     "              Student gr. PS-__"<<endl;
  cout.width(70); cout.setf(ios::right);
  cout<<"Novikov S.P. "<<
     "                    Snegiryov E."<<endl;
  cout<<endl<<endl<<endl<<endl;
  cout.width(50); cout.setf(ios::right);
  cout<<"Donetsk – 20__ "<<endl;
  main_menu();
 }
// головне меню
void main_menu()
 {int reply;
  cout.width(10);
  cout<<"1.  Description of the problem";
  cout.width(25);
  cout<<"2. Demonstration of the objects";
  cout.width(10); cout<<"3. Help";
  cout.width(10); cout<<"4. Exit"<<endl;
  cout<<"Enter the number of modes: "<<
        "1, 2, 3, 4 "<<endl;
  cin>>reply;
  switch(reply)
        {
         case 1:  uslovie(); break;
         case 2:  menu_reply(); break;
         case 3:  help(); break;
         case 4: exit(-1); break;
         default:
         cerr<<"You are mistaken. Press any "<<
               " key and then type the option "<<
               " number." <<endl;  getch();
         zastavka();
        }
  cout << any_key << endl;  getch();
}
// читання і показ файлу допомоги
void help()
 {const KSTR=21;
  const LS=80;
  FILE * fhelp;
  char namef[20]="help.txt";
  char s[61];
  int k=0;
  if((fhelp=fopen(namef, "r"))==NULL)
    {printf("File %s  is not open.\n", namef);
     cerr << any_key << endl; getch();
     exit(-1);
    }
  clrscr();
  int pr=1;
  int c;
  int l;
  int nend=1;
  strcpy(s, "");
```

```
  while(nend)
    {while(pr)
       {c=fgetc(fhelp);
        if(c!=EOF)
          {ungetc(c, fhelp);
           fgets(s, LS, fhelp);
           if(!nend)
             {cout<<endl<<endl <<
                    any_key<< endl; getch();
              clrscr();
              k=0;  nend=1;
             }
           printf("%s",s);
           k++;
           if(k==KSTR)  nend=0;
          }
        else
          {pr=0;   nend=0;
          }
       } // while(pr)
    } // while(nend)
  fclose(fhelp);
  cerr << endl << any_key << endl;  getch();
  zastavka();
 }
// читання і показ файлу умови завдання
void uslovie()
 {const KS=22;
  int ss[KS]={5, 8, 7, 8, 13, 9, 7, 8, 6, 10, 8, 7,
                  9, 11, 7, 9, 5, 8, 11, 12, 11, 12};
  ifstream fhelp;
  char namef[20]="uslovie.txt";
  fhelp.open(namef);
  if(fhelp.fail())
    {cerr<<"File "<< namef<<" is not open."
         <<endl;
     cout << any_key <<endl; getch();
     exit(-1);
    }
  clrscr();
  char s[80];
  int i;
  int j;
  for(i=0; i<KS; i++)
     {
      for(j=0; j<ss[i]; j++)
         {
          fhelp>>s;
          cout<<s<< " ";
         }
      cout<<endl;
     }
  fhelp.close();
  cout <<endl<<endl;
  cout << any_key << endl;  getch();
  zastavka();
 }
// головна функція
int main()
 {
  clrscr();
  zastavka();
   return 0;
 }
//++++++++++++++++++++++++++++
// Файл MARKET.PRJ
// вміст файлу проекту
CLASS1.CPP
CLASS2.CPP
MAIN.CPP
```