МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана ПУЛЮЯ

Кафедра програмної інженерії

# Методичні вказівки

до виконання курсових робіт з дисципліни

# «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

(CS-102)

Для студентів напряму підготовки 6.050103 “Програмна інженерія"

Тернопіль 2015

Розроблено у відповідності з навчальним планом напряму підготовки 6.050103 “Програмна інженерія".

Розглянуто мету й основні завдання курсової роботи з курсу "Об'єктно-орієнтоване програмування", вимоги до структури, змісту й оформлення курсової роботи, порядок її виконання та захисту. У додатках наведено зразки титульної сторінки, приклад змісту пояснювальної записки та примірну тематику курсових робіт.

Методичні рекомендації призначені для студентів 2-го року навчання напряму підготовки 6.050103 “Програмна інженерія", зокрема для студентів факультету комп’ютерно-інформаційних систем і програмної інженерії Тернопільського державного технічного університету імені Івана Пулюя.

Укладачі: Петрик М.Р., Петрик О.Ю.

Розглянуто на засіданні кафедри програмної інженерії, протокол № 2 від 14.09.2015 р.

Рекомендовано до друку засіданням методичної комісії факультету комп’ютерно-інформаційних систем і програмної інженерії, протокол № 2 від 22.09.2015 р.

ЗМІСТ

ВСТУП ..... 4

1 ЗАГАЛЬНІ ВКАЗІВКИ ..... 6

2 ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ ..... 7

2.1 Вимоги до оформлення текстових документів ..... 7

3 ЗМІСТ КУРСОВОЇ РОБОТИ ..... 10

3.1 Завдання на курсову роботу ..... 10

3.2 Вступ ..... 11

3.3 Аналіз специфікації вимог технічного завдання ..... 11

3.4. Об’єктно - орієнтований аналіз досліджуваної проблеми та методів моделювання й проектування ..... 12

3.5 Процес розроблення програми ..... 14

3.6 Тестування програми і результати її виконання ..... 20

3.7 Висновки ..... 20

3.8 Перелік використаних джерел ..... 20

3.9 Додатки ..... 21

ПІДГОТОВКА ДО ЗАХИСТУ ..... 22

ПОРЯДОК ЗАХИСТУ ..... 23

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 26

# ВСТУП

Курсова робота — це перша самостійна наукова праця майбутнього програмного інженера. Виконуючи її, студент поглиблює знання з фундаментальних і професійно-орієнтованих дисциплін, засвоює методику системного аналізу, програмної інженерії, оволодіває практичними навиками проектування програмних систем, прийняття рішень, наукового узагальнення і літературного оформлення одержаних результатів дослідження, веде науковий пошук, що розвиває в молодого програмного інженера творчий підхід до роботи. Курсова робота повинна продемонструвати вміння використовувати набуті теоретичні знання для вирішення конкретних прикладних задач.

Основна мета виконання курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» — закріплення на практиці вміння використовувати основні концепції об'єктно-орієнтованого підходу (ООП) — класи, інкапсуляцію, успадкування, поліморфізм, перевантаження методів і операцій, шаблони методів і класів та один із найпотужніших інструментаріїв ООП і програмної інженерії -– STL- бібліотеку (Стандартну бібліотеку шаблонів С++) засоби UML- моделювання при створенні складних програмних проектів.

Студент повинен уміти використовувати та практично застосовувати переваги об'єктно-орієнтованої парадигми при вирішенні поставленого завдання на етапах:

- об'єктно-орієнтованого аналізу прикладної проблеми (*object oriented analysis* (*OOA*)),

- об'єктно-орієнтованого проектування (*object oriented design* (*OOD*)),

- об'єктно-орієнтованого програмування (*object oriented programming* (*OOP*)).

При цьому недостатня й недопустима в курсовій роботі лише формальна заміна функції звичайної структурованої програми на методи якогось одного класу, наприклад, MainClass і потім здійснення їх виклику один за одним у головній функції (main() – функції).

Основним критерієм при проектуванні програми має бути практична доцільність. До реалізації проекту та створення програмного забезпечення слід підходити творчо і виважено, провівши аналіз специфікації вимог, здійснити об’єктно-

орієнтований аналіз досліджуваної проблеми як системи, виділивши в ній на основі абстраґування основні класи об'єктів, що взаємодіють між собою інформаційними потоками (обмінюються повідомленнями), виявити їх основні властивості й на основі цього розробляти систему класів, що є основою для системної архітектури програми.

Не слід також вдаватися до іншої крайності — використовувати класи та методи там, де можна обійтися звичайними операторами чи функціями, які не прив'язані до конкретних об'єктів з даними. Засоби і технології об'єктно-орієнтованого програмування повинні покращувати якість програмного коду: робити його доступнішим і зрозумілішим, зменшувати його обсяг, скорочувати час розроблення та відлагоджування програми, підвищувати її надійність і стійкість до внесення нових помилок, значною мірою полегшувати подальше її вдосконалення та модифікацію.

# 1 ЗАГАЛЬНІ ВКАЗІВКИ

Курсова робота — це складова навчального процесу. Її написання є обов'язковим елементом підготовки фахівця з розроблення та тестування програмного забезпечення за напрямом підготовки 6.050103 «Програмна інженерія».

Теми курсових робіт визначає викладач за особистими письмовими заявами студентів з урахуванням їх бажань, нахилів і практичного досвіду. Як правило, завданням курсової роботи є розроблення нової або удосконалення існуючої програми, бібліотеки програм, програмного пакета з використанням об'єктно-орієнтованої технології програмування мовою C++ (рекомендовано Visual Mirosoft C++).

При виконанні курсової роботи необхідно, виходячи з поставленого завдання, скласти план роботи, оцінити обсяг роботи й передбачити очікувані результати; бажано забезпечити достатню кількість якісних та змістовних ілюстрацій (оригінальних схем, UML-діаграм тощо).

Про результати виконання курсової роботи студент обов'язково повинен періодично звітувати перед керівником.

Витрати навчального часу студентів на виконання курсової роботи визначаються робочим навчальним планом.

Відбір, вивчення і реферування літературних джерел з теми курсової роботи займає важливе місце в дослідженнях студента. При виконанні курсових робіт недостатньо користуватися лише підручниками і навчальними посібниками, оскільки вони здебільшого розкривають лише основи тієї чи іншої мови або технології програмування, а не описують прикладних задач і проблем. Літературні джерела краще вивчати, переходячи від простих до складніших. При вивченні наукової літератури потрібно навчитися чітко відрізняти головне від другорядного, яке безпосередньо не пов'язане з темою дослідження.

# 2 ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ

## 2.1 Вимоги до оформлення текстових документів

Пояснювальна записка (записка) повинна розкривати зміст курсової роботи, містити обґрунтування класифікації основних абстракцій та вибору архітектури розроблюваної системи, сценарії реалізації, методів, алґоритмів, засобів стандартної бібліотеки параметризованих (шаблонних) класів тощо для виконання поставленого завдання, аналіз отриманих результатів та інші матеріали.

Матеріал пояснювальної записки повинен бути викладений грамотно, чітко та стисло. При цьому в тексті мають бути обов'язковими посилання на використані літературні та інші джерела.

У тексті пояснювальної записки не рекомендується вживати звороти із займенниками першої особи, наприклад: "Я вважаю ...", "Ми вважаємо ..." тощо. Рекомендується вести виклад, не вживаючи займенників, наприклад: "вважаємо ...","... знаходимо ..." тощо. Без пояснень дозволяється використовувати тільки загальноприйняті скорочення, наприклад: ПК, ДСТУ, ООП тощо.

Записку до курсової роботи виконують на аркушах білого паперу (з одного боку) формату А4 (210 х 297 мм) за формами 5 і 5а (ГОСТ 2.106-68), згідно з вимогами ГОСТу 2.105-79 та ДСТУ 3008-95, українською мовою із застосуванням друкуючих і графічних пристроїв виведення ПК (ГОСТ 2.004-88).

Обсяг записки повинен складати 30-40 сторінок машинописного тексту (без урахування додатків), надрукованого через 1,5 комп'ютерного інтервалу (до 30 рядків на аркуші А4). Розмір шрифту слід вибирати рівним 14 пунктів, гарнітуру – Times New Roman. Параметри сторінки: поля верхнє і нижнє – по 2 см, праве – 1,5 см, а ліве – 3 см. Абзацний відступ для першого рядка –1,27..1,7 см.

Пояснювальна записка повинна починатися з титульного аркуша. Виконують його згідно з ГОСТом 2.105-95 на аркуші формату А4 за формою, поданою в додатку А. Далі розміщують завдання на КР, анотацію та список скорочень (за необхідністю), зміст, основний текст, перелік використаної літератури та додатки.

Нумерацію сторінок записки починають із титульного аркуша, на якому номер не проставляють. Аркуш, розміщений після завдання на КР, нумерують цифрою 3.

Пояснювальну записку поділяють на розділи і підрозділи, пункти і підпункти. Розділи в межах усієї пояснювальної записки повинні мати порядкові номери, позначені арабськими цифрами без крапки. Вступ не нумерують. Підрозділи повинні мати нумерацію в межах розділу: номер підрозділу складається з номера розділу і підрозділу, розділених крапкою, наприклад, 2.3 означає: третій підрозділ другого розділу. У кінці порядкового номера розділу, підрозділу тощо крапку не ставлять. Номер пункту містить номер розділу, підрозділу і пункту, які розділені крапками, наприклад, 3.2.1 – перший пункт другого підрозділу третього розділу. Для забезпечення стислості викладу та економії паперу назви пунктів / підпунктів можна не нумерувати, а вставляти їх в основний текст на початку абзацу, виділивши жирним шрифтом.

Назви розділів повинні бути короткими і записувати їх слід у вигляді заголовків прописними буквами посередині рядка. Назви підрозділів записують у вигляді заголовків рядковими буквами (перша прописна). Переноси слів у заголовках не допускаються. Крапку в кінці заголовка не ставлять. Між назвами розділів, підрозділів і основним текстом повинен бути пропущений рядок.

Заголовки розділів відділяють від тексту зверху й знизу трьома інтервалами (30..36 пунктів на комп'ютері). Абзаци в тексті починають відступом, що дорівнює п'яти пропускам. (13-17 мм).

Фраґменти коду, що ілюструють певні елементи програмної реалізації, оператори, назви класів, методів тощо, що подаються в тексті, доцільніше для зручності читання набирати шрифтом Arial.

Графічний ілюстративний матеріал у тексті ПЗ (схеми, ескізи, графіки, рисунки) виконують у графічному редакторі. Кількість ілюстрацій повинна бути достатньою для пояснення тексту, що викладається. Ілюстрації розміщують одразу після посилання на них за текстом ПЗ. Усі розміщені в ПЗ ілюстрації нумерують арабськими цифрами в межах одного розділу, наприклад, Рис. 2.3 - розділ 2, рисунок 3. Посилання на ілюстрації подають за типом: на рис. 2.3, повторно – див. рис. 1.3.

Помилки, описки і графічні неточності, виявлені в процесі виконання документа, допускається зафарбовувати коректором і нанесенням на тому ж місці виправленого тексту (графіки) машинописним способом чи чорною пастою

рукописним способом.

Пошкодження аркушів пояснювальної записки, помарки і сліди неповністю видаленого попереднього тексту (графіки) не допускаються.

Порядкові числівники, які йдуть один за одним, можуть бути написані цифрами з відмінковим закінченням, яке ставлять лише при останній цифрі, наприклад: 1-е; 7, 8, 9-й тощо.

Перелік використаної літератури повинен містити лише ті літературні джерела, які використані при виконанні курсової роботи і на які є посилання в тексті документа.

Робота повинна бути написана державною мовою.

До пояснювальної записки курсової роботи обов'язково в якості її електронного додатка додавати у файлі формату A5  CD/DVD-диск з файлами, що містять відформатовану електронну версію пояснювальної записки  та вихідний C++ – код розробленої програмної системи. CD/DVD-диск є невід'ємною частиною додатка курсової роботи, що подається до захисту.

Перед поданням до захисту оформлена згідно з викладеними вимогами  курсова робота повинна бути переплетена разом із вказаним CD/DVD-диском у файлі A5 у додатку.

# 3 ЗМІСТ КУРСОВОЇ РОБОТИ

Рекомендована наступна структура курсової роботи:

- титульний аркуш – 1 с.;
- завдання на курсове проектування – 1 аркуш;
- зміст – 1-2 с.;
- вступ – 2-3 с.;
- суть – 15-30 с.;
- висновки – 1 с.;
- перелік посилань – 1 с.;
- додатки.

Титульний аркуш повинен містити інформацію про дисципліну, з якої виконувалося проектування, тему індивідуального завдання, упорядника пояснювальної записки.

## 3.1 Завдання на курсову роботу

Теми курсових робіт затверджують на засіданні кафедри на початку семестру. Відповідно до теми студент разом із керівником складає завдання на курсову роботу. У завданні вказується: тема курсової роботи; термін здавання завершеної роботи на кафедру; початкові дані до роботи; зміст роботи (перелік питань, які слід розробити). Підписують завдання керівник курсової роботи і студент. Допускається виконувати завдання з обох боків аркуша білого паперу формату А4 . Форма завдання наведена в додатку В.

У п. 1 завдання вказують тему курсової роботи, яка має бути сформульована чітко й лаконічно. Тема є індивідуальною для кожного студента, її визначають на кафедрі з урахуванням тематики наукових досліджень і договорів.

У п. 2 вказують термін, до якого студент повинен здати керівникові закінчену роботу на перевірку.

У п. 3 формулюють мету розробки, дають короткий опис прикладної проблеми і завдань, які потрібно вирішити в курсовій роботі. Вказують вид розробки (програмний проект, бібліотеку класів тощо), задають формат вхідних і вихідних даних програми.

У п. 4 зазначають основний перелік питань (розділів і підрозділів пояснювальної записки), які повинні бути розроблені при виконанні курсової роботи.

У п. 5 вказують конкретні технічні вимоги, параметри програми, формат вхідних і вихідних файлів, основні пункти меню, тип та версію програмного середовища, у якому розробляють та виконують програму.

У п. 6 вказують дату затвердження завдання на курсову роботу.

### 3.2 Вступ

У вступі студент обґрунтовує вибір теми, коротко викладає її актуальність, призначення розробки, для вирішення яких конкретних практичних завдань вона може бути використана. Тут потрібно сформулювати мету і завдання роботи, визначити основні підходи та ідеї, вибрати спосіб розв'язання задач. За наявності аналоґів програмних продуктів, які розв'язують подібні задачі, потрібно коротко охарактеризувати основні обмеження та недоліки й запропонувати шляхи їх усунення.

У вступі слід акцентувати увагу на прикладній проблемі, яку треба вирішити в курсовій роботі. У цьому розділі недоцільно наводити означення відомих термінів об'єктно-орієнтованого програмування, давати надто детальні характеристики й описи використаного програмного забезпечення та іншу інформацію, що мало стосується теми курсової роботи.

Обсяг вступу — 1 ..1,5 сторінки.

### 3.3 Аналіз специфікації вимог технічного завдання

Основною метою даного розділу пояснювальної записки є аналіз специфікації вимог технічного завдання на курсову роботу й формулювання додаткових вимог, які безпосередньо випливають з нього та мети дослідження.

Результатом аналізу є конкретизація постановки завдання, структури вхідних і вихідних даних, основних операцій і методів, які для них використовуються, основних параметрів та функціональних можливостей програмної системи, що в контексті основних вимог визначає загальну концепцію розроблення, основні засоби реалізації, інтерфейс із користувачем, робоче середовище.

Якщо у попередньому розділі формулюють тільки загальні підходи до розв'язування проблеми, то тут визначають конкретний план роботи. Просте переписування вимог завдання на курсову роботу не допускається, потрібно творчо осмислити його та визначити й сформулювати конкретні шляхи їх досягнення.

Обсяг розділу — 1..3 сторінки.

### 3.4. Об’єктно - орієнтований аналіз досліджуваної проблеми та методів моделювання й проектування

#### 3.4.1. Вибір моделі розроблення програмної системи

Основною метою даного підрозділу є дослідження поставленої проблеми з позиції об’єктно-орієнтованого аналізу, методології моделювання та проекту-вання створюваної програмної системи. На цьому етапі визначають основні ідеї щодо формалізації проблеми, підходи до моделі розроблення програмної системи (каскадний, уніфікований, спіральний тощо) з нових підходів позиції програмної інженерії. Результатом цього етапу з урахуванням складності розроблення має бути обґрунтований вибір прийнятної моделі розроблення програмної системи.

#### 3.4.2. Обґрунтування архітектури програми, сценаріїв та варіантів використання

Виходячи прийнятої моделі розроблення програмного забезпечення з позиції об’єктно–орієнтованого аналізу, важливим етапом є моделювання і проектування системної архітектури, яка визначає основні компоненти системи, специфікації вимог до них та механізми взаємодій між ними. Поряд з розробленням архітектури створюваної програмної системи виникають задачі розроблення основних та альтернативних сценаріїв і варіантів її використання. Як вказувалось раніше, ефективним інструментом проектування системної архітектури об’єктно-орієнтованого ПЗ і майбутніх сценаріїв та варіантів використання програми все ж є універсальна мова моделювання UML. Сценарії використання (алґоритми реалізації) як і розроблювана системна архітектура на базі ієрархії класів можуть бути відображені у вигляді UML-діаґрам (за необхідності блок-схем, оформлених згідно з вимогами стандартів, або словесних описів, що містять послідовність узагальнених операцій та переходів між ними). Такі описи можуть мати ієрархічну структуру, де спочатку записують загальніші операції (повідомлення між об’єктами), які потім розшифровують на нижчому ієрархічному рівні.

Описи сценаріїв використання (основного та одного чи кількох альтернативних) при об’єктно-орієнтованому підході до проектування програмного забезпечення зручніше проводити в термінах об'єктів, що взаємодіють між собою за допомогою UML-діаґрам варіантів використання, відношень, повідомлень і подій. До таких програм можна віднести різноманітні редактори (текстові, графічні), програми керування інформаційними сховищами (базами даних), інформаційно-довідкові системи, системи опрацювання переривань тощо. Щодо останніх, то у таких випадках алґоритм може розпадатися на кілька незалежних одна від одної послідовних операцій, що реагують на події та керують опрацюванням повідомлень.

Для систем, що працюють у реальному часі, важливою є часова послідовність опрацювання повідомлень і подій. Для них, крім опису алґоритму, застосовують ще й часові UML-діаґрами, що характеризують послідовність виконання різних процесів їх взаємодію з апаратними засобами, динаміку поведінки об’єктів.

У загальному спосіб описування сценаріїв чи варіантів використання, у тому числі й окремих алґоритмів, вибирають, виходячи з конкретного завдання, однак він повинен бути повним, несуперечливим і достатнім для розуміння суті та ілюстрації запропонованого способу вирішення та принципу роботи спроектованої програми.

Аналізуючи отримані системну архітектуру програми та сценарії використання, потрібно визначити основні структурні частини проекту в цілому, які дозволять здійснити його декомпозицію (розбиття) з точки зору реалізації на завершені програмні одиниці, що містять бібліотеки класів, методів тощо. Це суттєво спрощує як процес розроблення проекту в цілому, так і налагодження, тестування та верифікацію програмного коду.

Найчастіше окремі класи та змінні ґрупують за функціональним призначенням та розміщують в окремих програмних одиницях (файлах чи бібліотеках), які можна окремо компілювати. Для програмування на мові C++ бібліотеку розбивають на два типи файлів: заголовний файлів розширенням *.h , що містить інтерфейсну частину бібліотеки з описом класів, та вихідні файли реалізації з розширенням *.cpp, що містять детальні описи методів, сценаріїв (основного та альтернативних) і варіантів використання.

Крім вказаних файлів потрібна головна main()-програма, яка їх викликає. Для випадку, якщо завданням передбачено розроблення повноцінної програми, то ця main()-

програма може реалізувати й основний сценарій роботи.

Для деяких прикладних застосувань може виникнути необхідність створення допоміжних програм (утиліт), що виконують певні сервісні функції (конвертування файлів, формування тестових даних, виведення графіків тощо). Згідно з вказаною вище схемою потрібно провести системну декомпозицію проекту на основі розробленої архітектурної моделі та відобразити її в даному розділі. Обсяг розділу — 3..7 сторінок.

### 3.5 Процес розроблення програми

#### 3.5.1 Загальні правила

Цей розділ є основним у курсовій роботі. Його обсяг — 15..20 сторінок.

Залежно від обраного стилю програмування, згідно з концепцією архітектури та моделями сценаріїв використання можливі різні підходи до розроблення програми. Об'єктно-орієнтований підхід як ключовий метод розроблення складних програмних проектів, завдяки принципам інкапсуляції, ієрархії класів, успадкуванню та поліморфізму дає можливість виділити всі спільні та відмінні риси окремих елементів поставленого завдання і записати їх у програмі в найлаконічнішій формі.

Розроблення програми при об'єктно-орієнтованому підході розбивається на розроблення системи класів та глобальних типів даних, реалізацію сценаріїв використання, методів, дружних та глобальних функцій, створення об'єктів та організацію взаємодії між ними в головній програмі для досягнення поставленої мети.

#### 3.5.2 Проектування системи класів

Після того, як виконана декомпозиція програми (спроектована її системна архітектура) і розроблені основні та альтернативні сценарії використання, визначені основні об’єкти та методи і алґоритми їх взаємодії, можна приступати до реалізації наміченого плану. Потрібно чітко уявити, з якими даними буде працювати програма та які дії над ними доведеться виконувати. Дані та дії над ними утворюють об'єкт. Також можна уявити програму як набір об'єктів (реально існуючих фізичних об'єктів або абстрактних понять, що існують лише в нашій уяві), які взаємодіють між собою.

Наступним етапом є поділ об'єктів на групи, що мають щось спільне (дані, властивості, функціональність). У більшості випадків об'єкти будуть більше чи менше

подібні. Деколи в групах можна буде виділити підгрупи тощо, аж до складного ієрархічного дерева. У кожній групі виділяються найзагальніші властивості, притаманні всім без винятку об'єктам у групі та дії (операції) над ними. Отже, отримаємо базовий клас (суперклас) для певної групи об'єктів, даними якого будуть загальні параметри об'єктів групи, а методами — дії, які можна проводити над даними будь-якого об'єкта групи.

Далі для кожної з підгруп (у межах виділеного класу об'єктів) додаємо притаманні лише їй властивості, як поля нового породженого від базового класу. Додаткові дії, властиві кожній підгрупі, стають методами породжених класів, а параметри, яких не мали об'єкти базового класу, — даними породжених (похідних) класів. Породження класів продовжується до тих пір, поки не будуть описані всі параметри об'єктів та дії, які необхідно над ними здійснювати для розв'язання поставлених задач. Описана послідовність повторюється для кожної групи об'єктів у програмі.

Для прикладу розглянемо програму, яка організовує облік різних видів матеріального і нематеріального майна — нерухомості, банківських рахунків, акцій та облігацій. Банківські рахунки бувають поточними та ощадними. Акції та облігації можна віднести до цінних паперів, керування якими зовсім відрізняється від банківських рахунків, але й рахунки, й цінні папери є різновидностями майна. Можна використовувати інші класифікації подібних видів майна. Одним із аспектів, який потрібно враховувати, є те, що майно можна застрахувати (нерухомість і ощадні вкладення до деякої міри). Інший аспект — це здатність майна приносити дивіденди, що є загальною властивістю банківських рахунків та цінних паперів.

Для створення системи класів у даному випадку найприйнятнішим є використання множинного успадкування. Отримана структура ієрархії класів зображена на рис. 3.5.1. Тут клас Security (*цінні папери*) успадковує одночасно від класів InterestBearingltem (*джерело дивідендів*) і Asset (*майно*). Аналогічним чином клас BankAccount (*банківський рахунок*) успадковує відразу три класи: insurableltem (*страховане*) і відомі вже класи Asset та IntereetBearingltc.

Ось як це буде відображено у фрагменті C++ -коду:

```
// базові класи (суперкласи)
class Asset …
```

```
class  InsurableItem  ...
class InterestBearingItem ...
// проміжні класи; кожен успадковує  кілька суперкласів
class BankAccount : public Asset,
                    public InsurableItem,
                    public InterestBearingItem …
class RealEstate :   public Asset,
                    public InsurableItem …
class Security    :   public Asset,
                    public InterestBearingItem …
class BankAccount : public Asset,
class BankAccount : public Asset,
// класи- листя
class SavingsAccount   : public BankAccount ...
class CheckingAccount : public BankAccount ...
class Stock : public Security ...
class Bond  :  public Security  ...
```

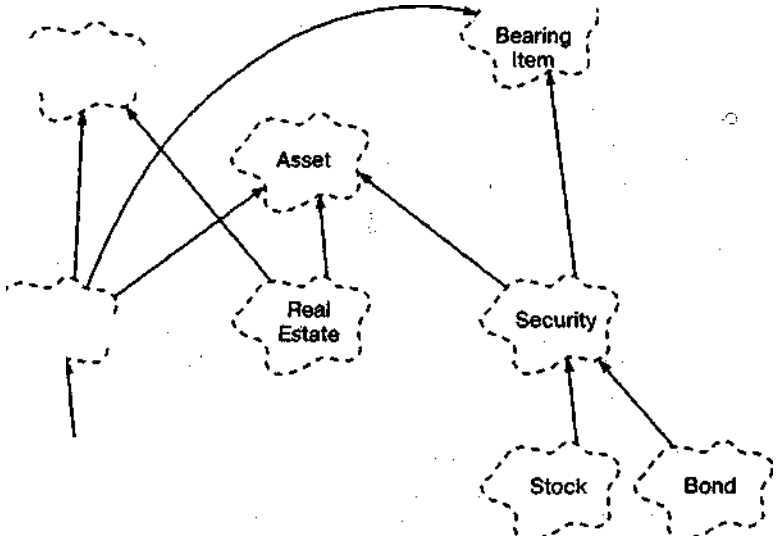


Рисунок 1 - Ієрархія класів для системи організації обліку майна

Цей далеко не повний приклад дає можливість зрозуміти основний підхід до проектування класів. Трохи пофантазувавши, можна отримати досить елеґантну й, головне, завершену ієрархію класів об'єктів, які будуть уміти робити все, що від них може вимагати головна програма для розв'язування поставлених задач. Основне — кожна спільна для кількох об'єктів дія (метод, повідомлення, операція) та спільні

елементи даних описані лише один раз у базовому  чи похідному класі, більше ніде їх не потрібно повторно описувати.

Якщо деякі дії передбачають виконання операцій над кількома об'єктами (обмін повідомленнями між кількома об'єктами) і при цьому необхідно мати безпосередній доступ до закритих даних класу, то такі дії робимо дружними функціями (методами) для того класу, над даними якого вони працюють. Методи, визначені в базових класах і які будуть використовуватися для перезавантаження в похідних класах, доцільно визначати як віртуальні.

Деякі типи даних, які недоцільно робити класом, оскільки вони не мають власної функціональності, але які неодноразово використовуються в програмі (наприклад, масиви фіксованої довжини, списки чи структури) можемо описати, використавши typedef, і помістити в заголовок одного з файлів.

Підрозділ "Розроблення системи класів" відображає основну роботу, яка виконується при об'єктно-орієнтованому підході, в результаті чого отримуються описи всіх класів з даними та методами, причому для методів уже задані їх параметри й тип результату, який вони повертають. Тобто, з точки зору інтерфейсу, робота над класами на цьому етапі вже завершена. Залишилося тільки "навчити" методи виконувати свою роботу.

Глобальні методи, які не є методами ні одного з класів (їх можна назвати алґоритмами), також поміщаються в один із *.cpp - файлів, а їх інтерфейсна частина описується разом з описами класів у заголовному файлі (*.h - файлі).

### 3.5.3 Реалізація методів

Після того, як з інтерфейсною частиною "закінчено", залишилося лише розробити реалізацію методів, дружних та глобальних функцій (методів) для забезпечення виконання сценаріїв використання. Це завдання є досить простим, якщо на попередньому етапі було правильно спроектовано систему класів і детально пророблено інтерфейсні частини. Реалізація будь-якого методу розміщується в *.cpp - частині бібліотеки, на відміну від інтерфейсної, розміщеної у *.h-файлі. Потрібно лише акуратно записати ті дії, які має виконувати метод, не забуваючи користуватися вже готовими бібліотечними функціями, шаблонними класами та методами, зокрема STL-засобами замість того, щоб увесь час змінювати дані за допомогою одних і тих же послідовностей операцій.

Аналізуючи різні варіанти використання та їх текстові описи для конкретно розробленої архітектури, можемо з'ясувати, якими повідомленнями обмінюватимуться класи. Оскільки повідомлення — це, по суті справи, виклик методу в об'єкті, то визначення повідомлень зводиться до визначення методів класу, що приймає те або інше повідомлення. У складених описах варіантів використання в якості таких повідомлень можуть використовуватись дієслова (іменники використовуються для найменувань у системі класів та їх подальшу трансформацію у їх відповідні англомовні еквіваленти). Звичайно, що не кожне дієслово стає кандидатом у повідомлення. Деякі з них, замість приймання даних від користувачів, зв'язуються з такими операціями, як виведення інформації на екран чи ще якими-небудь діями.

Для прикладу розглянемо опис варіанта використання, що застосовується в проекті однієї з інформаційно-довідкових систем Вивести список абонентів. У текстовому описі даного варіанта використання курсивом виділені дієслова.

Програма *виводить* на екран список абонентів, кожен рядок списку *складається* з трьох полів: адреса, ідентифікаційний код та ім'я абонента.

Під словом «програма» ми насправді маємо на увазі екран інтерфейсу користувача, отже, дієслово «виводить» означає, що об'єкт класу Екран інтерфейсу користувача посилає повідомлення об'єкту класу Список абонентів (тобто викликає його метод). У повідомленні міститься вказівка вивести самого себе на екран. Нескладно здогадатися, що метод може називатися, наприклад, display().

Аналогічний підхід можна застосувати для реалізації інших методів. Окремі методи можуть бути реалізованими з використанням поліморфізму та успадкування, використовуючи уже реалізовані методи базових класів. У даному підрозділі слід описати принцип дії основних методів та відмінності поліморфних (віртуальних) методів від методів базових класів.

### 3.5.4 Створення об'єктів і розроблення головної програми

У цьому підрозділі слід описати процес створення самих об'єктів у програмі, функціональність і дані яких розроблені в попередніх підрозділах. Якщо кількість об'єктів, що будуть створені наперед, невідома, або обсяг пам'яті, яку вони займають, є досить значним, тоді пам'ять для об'єктів слід виділяти "динамічно" з використанням оператора new. У цьому випадку доброю практикою є постійний контроль обсягу вільної пам'яті та використання певних застережних дій (наприклад, видача

попереджувальних повідомлень), про що слід обумовити в альтернативних сценаріях.

Створивши об'єкти, можемо реалізувати певну послідовність дій або схему взаємодії між об'єктами, яка призводить до вирішення поставленого завдання.

Опис цієї послідовності дій чи схеми взаємодії, можливо, з прикладами для реальних вхідних даних чи повідомлень і є основою даного підрозділу. Такий опис повинен бути достатнім для розуміння всіх деталей реалізації алґоритму та можливих випадків, що зустрічаються в роботі програми.

Для роботи з великою кількістю об’єктів, що займають значний обсяг пам’яті, де передбачені операції копіювання, сортування, переміщення, доцільно з метою економії пам’яті вводити об’єкти вказівників і допоміжні функціональні об’єкти (шаблонні класи порівняння) та інші STL-засоби (контейнери, алґоритми, методи та ітератори).

Для забезпечення ефективності потокового вводу/виводу слід використовувати потокові класи й об’єкти (у тому числі файлові).

3.5.5 Опис файлів даних та інтерфейсу програми

Якщо програма використовує файлові об’єкти як джерело вхідних даних або для зберігання проміжних чи кінцевих результатів роботи, то в даному підрозділі слід навести опис формату цих файлів. Він може бути виконаний у текстовому, табличному чи графічному вигляді. Можна додавати до нього приклади реальних файлів із даними програми.

Особливо важливий даний підрозділ для програм, що працюють із базами даних, адже структура таблиць бази даних, перелік, типи полів, засоби взаємодії з базою даних, перетворення даних є основою таких програм. Тут важливим також є використання та обґрунтований вибір оптимального стандартного STL-класу сховища даних (list, vector, deque, set, map, multyset, multymap) та відповідних їм STL-методів.

Для інтерактивної програми з розвиненою системою меню та діалогових вікон у цьому підрозділі слід описати призначення елементів меню, роботу з ними, параметри, що вибираються в діалогових вікнах, тощо. У цьому ж розділі слід описати інтерфейс програм, які працюють з параметрами командного рядка.

Якщо темою роботи є розроблення модулів або бібліотеки, то слід детально

описати порядок їх використання, навести в алфавітному порядку всі доступні для користувача глобальні типи даних і змінні, функції та класи.

Слід зауважити, що п. 3.4 - 3.5 відображають зміст й обсяги роботи, які повинен виконати студент для розроблення програмної системи. Щодо назв цих пунктів, назв їх підпунктів та кількості, то все це може бути змінено залежно від поставлених завдань, і методів реалізації та способу викладу матеріалу.

3.6 Тестування програми і результати її виконання

У даному розділі треба описати методику тестування програми, тестові дані та навести результати роботи програми. Якщо програма працює в графічному режимі, то слід роздрукувати копію графічного вікна програми. Якщо результатом роботи програми є текстовий файл, то необхідно вивести вміст цього файлу. Для програм з розвиненою системою діалогових вікон і меню слід обмежитися видруком лише найсуттєвіших результатів, які демонструють правильну роботу програми, а не передруковувати весь екран для кожного відкритого пункту меню. Перелік усіх пунктів меню в такому випадку та вміст неосновних діалогових вікон можна подати в текстовому вигляді. Якщо для відображення роботи програми необхідна значна кількість роздруків, то їх можна навести в додатках.

Обсяг розділу — 1..5 сторінок. У розділі потрібно зробити висновок, який підтверджує (або заперечує) працездатність програми.

3.7 Висновки

На підставі власних досліджень автор повинен написати висновки, у вигляді коротко сформульованих і пронумерованих тез. Кількість висновків залежить від обсягу отриманих результатів, складності програми та прикладних задач.

Обсяг висновків — до 1 сторінки.

3.8 Перелік використаних джерел

Перелік використаної літератури оформляти згідно з чинними стандартами та відповідними вимогами ВАК України до оформлення дисертацій.

## 3.9 Додатки

Матеріал, що доповнює текст пояснювальної записки, можна розміщувати в розділі "Додатки". У них можуть бути лістинґи програмного коду, графіки й таблиці, діаґрами тощо. У тексті записки на всі додатки потрібно дати посилання, розміщувати додатки слід у порядку посилань на них у тексті.

Кожний додаток треба розпочинати з нової сторінки із зазначенням зверху посередині сторінки слова "Додаток" і його позначення. Згідно з ДСТУ 3008 - 95, додатки слід позначати послідовно великими літерами української абетки, за винятком літер Ґ, Є, З, І, Ї, О, Ч, Ь, наприклад: додаток А, додаток Б і т. д. Оформляти додатки можна на аркушах формату А4, А3, А2, А1 згідно з ГОСТом 2.301 -68.

Першими мають бути додатки з лістинґами програмного коду, розробленими в курсовій роботі (А1, А1, …). Якщо при розробленні використовувалися технології візуального програмування та при значному обсязі коду програми, допускається наводити лише тексти основних модулів програми.

Ілюстрації кожного додатка позначають окремою нумерацією з додаванням перед арабською цифрою буквеного позначення додатка, наприклад: Рис. А.2, Рис. В.3 тощо. Додатки повинні мати загальну з курсовою роботою наскрізну нумерацію сторінок і бути перераховані в змісті роботи із зазначенням їх позначень і заголовків.

# ПІДГОТОВКА ДО ЗАХИСТУ

Виконану згідно зі стандартами відповідно до завдання і у повному обсязі курсову роботу, підписану виконавцем, у незброшурованому вигляді треба подати на перевірку керівникові.

До текстової частини записки необхідно додати повний електронний варіант усіх файлів з текстами програм і модулів, тестові файли, скомпільований виконуваний файл основної програми та допоміжних програм-утиліт (якщо такі є), об'єктні модулі (для випадку розроблення модулів або бібліотеки). Якщо програма вимагає для своєї роботи додаткових динамічних бібліотек (DLL), які не входять до складу операційної системи, то їх також необхідно додати або вказати назву та джерело, з якого можна зчитати ці файли.

Для забезпечення надійності зчитування, при записуванні файлів на носій, додають додаткову копію в підкаталозі сору або інший носій з копією інформації. На електронному носії необхідно вказати прізвище, ім'я студента, ґрупу, дату захисту та тему курсової роботи.

Роботу подати на перевірку не пізніше, як за три робочих дні до захисту. Виявлені при перевірці курсової роботи неточності й помилки студент зобов'язаний виправити, а результати представити керівникові у визначені терміни. Якщо ж при огляді встановлено, що робота в будь-якій частині потребує суттєвого доопрацювання, то визначається обсяг доопрацювання і встановлюється термін подання виправленої роботи на повторну перевірку.

Роботи, що не відповідають затвердженій темі, без затвердженого завдання на курсову роботу, підписаного студентом і викладачем, а також ті, в яких виявлено запозичення з інших джерел, без посилання на джерело до захисту не допускаються.

# ПОРЯДОК ЗАХИСТУ

До захисту курсової роботи допускаються студенти, які виконали всі вимоги навчальної програми та календарного плану, своєчасно представили роботу й усі необхідні матеріали.

Захист курсових робіт проводиться відкрито у відповідності і встановленим графіком. На захист роботу слід представляти тільки в зброшурованому вигляді, до пояснювальної записки додати електронний варіант на оптичному носії (прикріпленому до останньої сторінки обкладинки або іншим способом), оформлений згідно з п. 4.2.

Захист курсової роботи проходить у такій послідовності:

- доповідь студента про основні результати виконаної роботи;
- відповіді студента на запитання присутніх;
- обговорення доповіді;
- відповіді на зауваження.

Для доповіді про результати виконаної роботи студенту надається 7-10 хвилин. Доповідь повинна складатися з трьох частин (вступна та основна частини й висновки).

У вступній частині доповіді необхідно відзначити актуальність теми, в загальному проаналізувати стан питання, сформулювати основні задачі, з розв'язуванням яких пов'язано виконання роботи.

В основній частині доповіді необхідно навести короткі відомості про зміст виконаних досліджень, відзначити основні підходи та показати ефективність прийнятих рішень, надати короткі відомості про отримані результати. Основну частину доповіді можна супроводжувати посиланням на графічні матеріали та демонструвати роботу програми.

У висновках необхідно чітко сформулювати основні результати курсової роботи, наголосивши на повноті вирішення поставленого завдання.

Відповіді на запитання повинні бути короткими, за суттю й не виходити за межі поставленого запитання.

При захисті курсової роботи студент може використовувати електронну презентацію.

При визначенні оцінки курсової роботи береться до уваги рівень теоретичної й практичної підготовки студента.

Оцінювання курсової роботи проводиться за кредитно-модульною системою оцінювання, прийнятою для оцінювання знань студентів в університеті.

Оцінка "відмінно" (90-100 балів ВНЗ, A по ECTS) ставиться у випадку, коли задовольняються всі перераховані нижче вимоги:

1) якщо в роботі немає суттєвих недоліків;

2) програма повністю виконує поставлене завдання;

3) при розробленні програми вміло використано переваги технології об'єктно-орієнтованого програмування, а саме: абстраґування, інкапсуляцію, успадкування та поліморфізм;

4) при захисті роботи студент аргументовано виклав основні технічні рішення, прийняті в процесі розроблення та відповів на поставлені запитання;

5) робота виконана самостійно.

Робота оцінюється оцінкою "добре" (75-89 балів, C, B по ECTS), якщо:

1) у роботі немає суттєвих недоліків (B);

2) програма повністю виконує поставлене завдання, але містить деякі незначні помилки(B);

3) при розробленні програми не використано всі переваги технології об'єктно-орієнтованого програмування, програма реалізована не оптимальним чином (C);

4) при захисті роботи студент допустив неточності або не було аргументованих відповідей на деякі з поставлених запитань (C).

Робота оцінюється оцінкою "задовільно" (50-74 балів; E, D за ECTS), якщо:

1) у роботі є суттєві недоліки(D);

2) програма не повністю виконує поставлене завдання, містить значні помилки, які не дозволяють використовувати її для деяких комбінацій вхідних параметрів (D);;

3) при розробленні програми не використано технологію об'єктно-орієнтованого програмування (E);

4) робота оформлена зі значними відхиленнями від стандартів і вимог або в процесі проектування були відхилення від затвердженого календарного плану чи

завдання (E);

5) при захисті роботи студент допустив суттєві неточності або не було аргументованих відповідей на поставлені запитання(E).

Якщо програма:

1) не виконала поставленого завдання (FX);

2) не оформлена належними чином (FX);;

3) студент систематично порушував календарний план, не виконав більшу частину завдання (FX);

4) програма містить значні недоліки, а наявні помилки не дають можливості встановити її працездатність для реальних вхідних даних (FX);

5) роботу виконав студент не самостійно, при її захисті не було обґрунтовано прийняті рішення, а запитання, які задавались, залишились без відповіді (F),

то роботу оцінюють оцінкою "незадовільно" (35-59 балів, FX; 1-34 балів; F ) на "2", а подальшу процедуру її захисту визначають за чинними правилами університету.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. *Страуструп Б.* Язык программирования С++. Специальное издание.-М., СПб.:"Издательство БИНОМ" – „Невский Диалект", 2001г.-1099 с.

2. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на С++. Пер. с англ. – М.: СПб.: "Издательство БИНОМ"-"Невский Диалект", 2001.-560 с.

3. *Глушаков С.В., Коваль А.В., Смирнов С.В.* Язык программирования С++: Учебный курс.- Харьков: Фолио; М.: «Видавництво АСТ», 2001.-500 с.

4. Р. Лафоре. Объектно-ориентированное программирование в С++. 4-е издание. Издательство: Питер. Серия: Классика computer science, 2005.- 928 с.

5. *Шилдт Г.* Самоучитель С++, 3-е издание: пер. с англ.-СПб.:BHV-Санкт-Петербург,1999.-688 с.

6. *Петрик О., Петрик М.* Технологія програмування в середовищі С++.– Тернопіль: ТДТУ ім. Ів. Пулюя, 2005. − 110 с.

7. *Шилдт Г.* Полный справочник по С++. 4-е издание. - М.: Изд. дом «Вильямс», 2003г.–800с.

8. *Седжвик Р.* Фундаментальные алгоритмы на С++. –К.: Изд-во «ДиаСофт», 2001. -688с

9. *Шилдт Г.* Теория и практика С++: пер. с англ.-СПб.: BHV-Санкт-Петербург,1999.-416 с.

10. *Подбельский В.В.* Язык С++.-М.: Финансы и статистика,1966.-558 с.

11. *Дейтл Н., Уимз Ч., Хедингтон М.* Программирование на С++. Пер. С англ. - М.:ДМК,2000. - 672 с.

12. *Айра П.* Объектно-ориентированное программирование на С++. Пер. с англ. – М.: СПб.: "Издательство БИНОМ"-"Невский Диалект", 1999.-462с.

13. *Бегун А.В.* Технологія програмування: об`єктно-орієнтований підхід: Навч.-метод. Посібник для самост. вивч. дисц.-К.:КНЕУ,2000.-200 с.

ДОДАТОК А

ЗРАЗОК ОФОРМЛЕННЯ ТИТУЛЬНОГО АРКУША КУРСОВОЇ РОБОТИ

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана ПУЛЮЯ
Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

Напрям підготовки 6.050103 «Програмна інженерія»

Курсова робота
з навчальної дисципліни
«Об'єктно-орієнтоване програмування» (CS-102)

на тему
«ІНФОРМАЦІЙНА СИСТЕМА РЕГУЛЮВАННЯ ДОРОЖНОГО РУХУ У м. ТЕРНОПОЛІ»

Виконав:
студент ґрупи СП-21
*Снігур Ігор*

Курсова робота захищена
з оцінкою _____
«__» _____ 2015 р.

Керівник роботи:
*Петрик М.Р.*

Тернопіль 2015

ДОДАТОК Б

РЕКОМЕНДОВАНА СТРУКТУРА КУРСОВОЇ РОБОТИ

Завдання на курсову роботу..... 2
Вступ.....3
1 Аналіз технічного завдання.....4
2 Обґрунтування алґоритму й структури програми.....6
3 Розроблення програми.....8
3.1 Розроблення системи класів.....10
3.2 Розроблення методів.....13
3.2 Створення об'єктів і розроблення головної програми.....16
3.3 Опис файлів даних та інтерфейсу програми.....18
4 Тестування програми і результати її виконання.....21
Висновки.....23
Перелік використаної літератури.....24
Додаток А1 Заголовний файл програми…….....27
Додаток А2 Вихідний файл програми з описами методів класів .....32
Додаток Б1 Вхідний тестовий файл.....35
Додаток Б1 Вихідний тестовий файл.....38

ДОДАТОК В

Тематика курсових робіт

**1. ПС “Відділ кадрів" (інституту).**

Завдання – інформаційна підтримка діяльності відділу кадрів.

Розрізняють три групи співробітників: а) адміністрація; б) викладацький і інженерно-технічний склад (по кафедрах); у) технічний персонал. ІС повинна містити штатний розклад по відділах (кафедрам) з вказівкою кількості ставок по посадах, включати архів співробітників і враховувати співробітників, що знаходяться у відпустці по догляду за дитиною.

ІС повинна надавати можливість складання посадових (штатних) розкладів по кафедрах і відділах і наступних списків:

- вакансій (з урахуванням співробітників, що знаходяться у відпустці по догляду за дитиною, тобто з вказівкою дати, до якої ставка вільна);
- пенсіонерів;
- людей передпенсійного віку (не більш 2-х років до пенсії);
- бездітних співробітників;
- ювілярів поточного року;
- багатодітних співробітників (троє і більш за дітей);
- ветеранів (що працюють в інституті не менші тридцять років);
- співробітників, що працюють більш ніж на одній ставці.

**2. «Бібліотечні засоби для розв'язування задач лінійної алгебри»**

Реалізувати програмну систему для роботи з матрицями. Передбачити - додавання та віднімання матриць; додавання та віднімання векторів; множення матриці на вектор; знаходження оберненої матриці; розв'язування системи лінійних рівнянь; скалярне множення векторів.

Розробити демонстраційно-тестуючу програму. Виконати тестування розроблених програмних засобів.

**3. ПС “Плановий відділ".**

Завдання – інформаційна підтримка діяльності планового відділу (вибрати конкретне виробництво).

ІС повинна здійснювати:

- ведення планової документації по основному і допоміжному виробництвах (план і факт);
- складання замовлень на постачання сировини і що комплектують (відповідно до плану випуску продукції);
- складання планів роботи допоміжних виробництв для забезпечення потреб основного виробництва;
- підрахунок енерговитрат;
- визначення відповідності результатів роботи плану (у відсотках).

**4. ПС “Кафедра".**

Завдання – інформаційна підтримка учбового процесу і організаційної діяльності на кафедрі вузу. ІС повинна містити учбовий план, розклад занять, списки груп, що випускаються кафедрою, і списки аспірантів (з керівниками і темами досліджень). ІС повинна забезпечувати складання:

- розклади занять на семестр (по групах);
- учбового плану (по семестрах) для кожного курсу;

- розклади занять для викладачів;
- списку телефонів співробітників;
- навантаження по годиннику для викладачів;
- списку наукових кадрів по наукових напрямах;
- списків студентів-дипломників (по групах і по викладачах).

**5. ПС “Поліклініка".**

Завдання – інформаційна підтримка діяльності поліклініки. ІС повинна здійснювати:

- ведення медичних карт пацієнтів;
- облік рецептів, напрямів на аналізи, процедури;
- облік платних послуг з видачею рахунку на оплату;
- ведення черг на прийом до фахівців з напрямів лікарів, що лікують.

Необхідно передбачити:

- визначення відвідуваності окремих кабінетів (навантаження лікарів);
- підрахунок кількості хворих за день для визначення настання епідемії.

**6. ПС “Готель".**

Завдання – інформаційна підтримка діяльності готелю.

ІС повинна здійснювати:

- ведення списку постояльців;
- облік заброньованих місць;
- ведення архіву вибулих постояльців за останній рік.

Необхідно передбачити:

- отримання списку вільних номерів (по кількості місць і класу);
- отримання списку номерів (місць), що звільняються сьогодні і завтра;
- видачу інформації по конкретному номеру;
- автоматизацію видачі рахунків на оплату номера і послуг;
- отримання списку заброньованих номерів;
- перевірку наявності броні по імені клієнта і/або назві організації.

**7. ПС “Продаж квитків".**

Завдання – інформаційна підтримка діяльності транспортних кас (вибрати вид транспорту). ІС повинна здійснювати:

- ведення списку рейсів і квитків на них з вказівкою класу;
- облік заброньованих місць;
- ведення архіву пасажирів за останній місяць.

Необхідно передбачити:

- продаж квитків в обидва кінці;
- пошук місця на рейс відповідно до вимог замовника;
- отримання списку вільних місць на рейс;
- видачу інформації по конкретному рейсу;
- отримання списку проданих місць;
- перевірку наявності броні по імені клієнта і/або назві організації.

**8. ПС “Спортивний клуб".**

Завдання – інформаційна підтримка діяльності спортивного клубу. ІС повинна здійснювати:

- ведення списків спортсменів і тренерів;
- облік змагань, що проводяться (з веденням їх архіву);
- облік травм, отриманих спортсменами.

Необхідно передбачити:

- можливість переходу спортсмена від одного тренера до іншого;
- складання рейтингів спортсменів;
- складання рейтингів тренерів;
- видачу інформації по змаганнях;
- видачу інформації по конкретному спортсменові;
- підбір можливих кандидатур на участь в змаганнях (відповідного рівня майстерності, віку і без травм).

**9. ПС “Кінотеатр".**

Завдання – інформаційна підтримка діяльності кас кінотеатру. ІС повинна здійснювати:

- облік заброньованих місць;

Необхідно передбачити:

- продаж квитків ;
- пошук місця відповідно до вимог замовника;
- отримання списку вільних місць ;
- видачу інформації по конкретному сеансу;
- отримання списку проданих місць;

**10. ПС “Відеопрокат".**

Завдання – інформаційна підтримка діяльності системи відеопрокату
ІС повинна забезпечувати:

- ведення автоматизованого обліку видачі/прийому касет;
- облік рейтингу фільмів (кількість користувачів і дата останньої видачі);
- пошук фільму по темі, режисерові, акторові, ключовому слову (із завданням періоду, що цікавить);
- складання списків боржників по роках.

**11. ПС “Фітнес-клуб".**

Завдання – інформаційна підтримка діяльності клубу. ІС повинна здійснювати:

- ведення списків тих, що займаються і тренерів;

Необхідно передбачити:

- можливість переходу що займається від одного тренера до іншого;
- складання рейтингів тренерів;
- видачу інформації по тому, що конкретному займається;

**12. ПС «Ремонт комп'ютерів»**

Розробити додаток “Ремонт комп'ютерів”.
Завдання – інформаційна підтримка діяльності фірми по ремонту комп'ютерів. ІС повинна здійснювати:

- введення і коректування інформації про співробітників, замовників, комп'ютери
- підсумкові дані по ремонтах (“Ремонт був виконаний” “Невиконані замовлення” “Сумарна вартість замовлень співробітника” “Число замовлень співробітника”)

**13. ПС “Магазин з продажу стільникових телефонів".**

Завдання – інформаційна підтримка діяльності магазина вибраного профілю. ІС повинна здійснювати:

- облік постачальників і постачань;
- облік продажів ;
- підрахунок залишків товарів;
- оформлення замовлень на товари;
- підбиття фінансових підсумків дня (по відділах і в цілому по магазину);

- аналіз результативності роботи продавців (для преміювання);
- аналіз об'ємів продажів по днях тижня і по місяцях.

**14. ПС “Магазин побутової техніки".**

Завдання – інформаційна підтримка діяльності магазина вибраного профілю. ІС повинна здійснювати:

- облік постачальників і постачань;
- облік продажів ;
- підрахунок залишків товарів;
- оформлення замовлень на товари;
- підбиття фінансових підсумків дня (по відділах і в цілому по магазину);
- аналіз результативності роботи продавців (для преміювання);
- аналіз об'ємів продажів по днях тижня і по місяцях.
- пошук техніки по необхідній марці, ціні.

**15. ПС “Відділ кадрів фірми" .**

Завдання – інформаційна підтримка діяльності відділу кадрів.

Розрізняють 2 групи співробітників: а) адміністрація; б) менеджери; у) технічний персонал. ІС повинна містити штатний розклад по відділах з вказівкою кількості ставок по посадах, включати архів співробітників і враховувати співробітників, що знаходяться у відпустці по догляду за дитиною.

ІС повинна надавати можливість складання посадових (штатних) розкладів по відділах і наступних списків:

- вакансій (з урахуванням співробітників, що знаходяться у відпустці по догляду за дитиною, тобто з вказівкою дати, до якої ставка вільна);
- пенсіонерів;
- ювілярів поточного року;
- багатодітних співробітників (троє і більш за дітей);

**16. ПС “Санаторій".**

Завдання – інформаційна підтримка діяльності санаторію. ІС повинна здійснювати:

- облік надходження пацієнтів (по відділеннях);
- облік проведеного лікування;
- облік платних послуг з видачею рахунків на оплату;
- ведення архіву пацієнтів, що виїхали.

Необхідно передбачити визначення (по відділеннях):

- пропускній спроможності санаторію;
- наявність вільних місць в палатах (окремо для чоловіків і для жінок);