ТЕХНІЧНИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ

Циклова комісія програмних систем і комплексів

**КУРСОВА РОБОТА**

з дисципліни:

**«Об’єктно-орієнтоване програмування»**

на тему: **«Програма автоматизації роботи готельного комплексу»**

Студента  3  курсу групи КН-321 спеціальності 122 «Комп’ютерні науки»

Ярмош І.О.

(прізвище та ініціали)

Керівник: викладач   Слободян  Р.О.         
Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_ Р.О. Слободян

(підпис)

\_\_\_\_\_\_\_\_\_ Г.Я. Марціяш

(підпис)

м. Тернопіль – 2019

ЗМІСТ

[ВСТУП 4](#_Toc26434668)

[1 ТЕХНІЧНЕ ЗАВДАННЯ 5](#_Toc26434669)

[1.1 Найменування та область застосування 5](#_Toc26434670)

[1.2 Підстави для розробки 5](#_Toc26434671)

[1.3 Призначення розробки 5](#_Toc26434672)

[1.4 Вимоги до програми чи програмного виробу 5](#_Toc26434673)

[1.5 Вимоги до програмної документації 6](#_Toc26434674)

[1.6 Техніко-економічні показники 6](#_Toc26434675)

[1.7 Стадії та етапи розробки 6](#_Toc26434676)

[1.8 Порядок контролю та прийому 6](#_Toc26434677)

[2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ 7](#_Toc26434678)

[2.1 Розробка загальної структури і варіантів використання програми 7](#_Toc26434679)

[2.2 Розробка системи класів 7](#_Toc26434680)

[2.3 Розробка методів 7](#_Toc26434681)

[2.4 Проєктування і опис інтерфейсу користувача 7](#_Toc26434682)

[2.5 Опис файлової структури програми 7](#_Toc26434683)

[2.6 Опис структури бази даних програми 7](#_Toc26434684)

[3 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 8](#_Toc26434685)

[ВИСНОВКИ 9](#_Toc26434686)

[ПЕРЕЛІК ПОСИЛАНЬ 10](#_Toc26434687)

[Додаток А Діаграма варіантів використання 11](#_Toc26434688)

[Додаток Б Лістинг файлу «program.h» 12](#_Toc26434689)

[Додаток В Лістинг файлу «program.cpp» 13](#_Toc26434690)

[Додаток Г Компакт-диск із програмним продуктом 14](#_Toc26434691)

ВСТУП

Протягом останніх декількох десятиліть комп'ютерні технології розвивались дуже швидко. Мови програмування також потерпіли значної еволюції. Поява більш потужних комп'ютерів дала життя більш об'ємним і складним програмам, які, водночас, висвітлювали нові проблеми в області керування програмами, а також їх подальшому супроводу.

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування в 1960-тих роках, одночасно з посиленням дискусій про [кризу програмного забезпечення](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%B8%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&redlink=1). Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), або як перелік [інструкцій](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F) комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати [повідомлення](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Сучасна економіка призвела до перегляду об’ємів витрат на комп’ютерну техніку та на спеціалізоване програмне забезпечення. Набагато краще і вигідніше стало розробляти прості програмні засоби, орієнтовані на розв’язання вузько спеціалізованих завдань. Тому, перед навчальними закладами постала мета підготувати спеціалістів, які спроможні створювати гнучкі програмні засоби, не витрачаючи на їх розробку та впровадження багато фінансів та часу.

Як приклад, в даній курсовій роботі представлено розробку спеціалізованого програмного рішення мовою С++ із використанням бази даних та принципів об’єктно-орієнтованого програмування. Ціль його розробки – автоматизація кас залізничного вокзалу, що значно полегшує їхню роботу.

Об'єктно-орієнтоване програмування - це парадигма програмування, заснована на поданні програми у вигляді сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування. Спочатку розробляється клас, а на його основі при виконанні програми створюються конкретні об'єкти (екземпляри класів). На основі класів можна створювати нові, які розширюють базовий клас і таким чином створюється ієрархія класів. Це дозволяє швидко конструювати достатньо функціональний програмний засіб.

# ТЕХНІЧНЕ ЗАВДАННЯ

## Найменування та область застосування

Найменування програми – «Програма автоматизації роботи готельного комплексу».

Стисла назва – «HotelH».

Область застосування програми – обслуговування клієнтів за допомогою автоматизованої програми для роботи з інформацією у готелі .

## Підстави для розробки

Підставами для проведення розробки являється індивідуальне завдання на курсову роботу з дисципліни «Об’єктно-орієнтоване програмування».

Найменування теми курсової роботи –«Програма автоматизації роботи готельного комплексу».

Замовник – керівник курсової роботи, викладач Слободян Руслан Олесійович.

Виконавець – студент групи КН-321 Ярмош Ігор Олегович.

## Призначення розробки

Підвищення ефективності і полегшення роботи працівників готельного комплексу.

Відповідно функціональне призначення – спрощення і автоматизація роботи готельного комплексу.

## Вимоги до програми чи програмного виробу

### Вимоги до функціональних характеристик

Програма автоматизації роботи готельного комплексу повинна забезпечити оптимізацію і спрощення таких форм обліку:

--

### 1.4.2Вимоги до надійності

Надійне функціонування програми повинне забезпечуватись наявністю блоку перевірки правильності введених даних. Це збереже від помилок, що виникають при виконанні рутинної роботи користувачем.

Також необхідно забезпечити достатню стійкість програми до некваліфікованих дій користувача, що досягається супроводом будь-яких операцій в програмі чіткими і зрозумілими повідомленнями.

### 1.4.3 Умови до експлуатації

Розроблений програмний продукт не потребуватиме яких-небудь видів обслуговування, для його використання достатньо буде навиків користувача ПК.

### 1.4.4 Вимоги до інформаційної і програмної сумісності

В склад технічних засобів повинні входити: персональний комп’ютер із такими мінімальними характеристиками:

− Процесор Pentium III;

− ОЗУ: 128Mb, або вище;

− обсяг дискової пам’яті – 80 ГБ.

− монітор;

− клавіатура;

− мишка.

### 1.4.5 Вимоги до інформаційної і програмної сумісності

Вихідні коди програми повинні бути реалізовані мовою С++. В якості середовища розробки програми повинне використовуватись середовище Qt Creator 4.8.0.

Системні програмні засоби і утиліти, які буде використовувати програма, повинні забезпечуватись операційною системою сімейства Windows версії не нижче Windows 7, Linux або MacOS оскільки середовище Qt є кросс-платформне.

Вимоги до захисту інформації і програми не пред’являються.

## Вимоги до програмної документації

По закінченню розробки програмного забезпечення потрібно підготувати таку документацію:

- інструкція інсталяції програми;

- загальні відомості про можливості програми;

- інструкція з експлуатації.

## Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться.

Приблизне число використань розробленої програми в рік – 500-1500 раз

## Стадії та етапи розробки

В загальному випадку, життєвий цикл визначається моделлю й описується у формі методології (методу). Модель або парадигма життєвого циклу визначає загальну організацію і, як правило, основні його фази та принципи переходу між ними. Методологія (метод) визначає комплекс робіт, їх детальний зміст і рольову відповідальність спеціалістів на всіх етапах вибраної моделі.

Життєвий цикл програмного забезпечення супроводжується розробленням, обігом та використанням програмної документації.

Програмна документація — сукупність документів, що містять відомості, необхідні для розробки, виготовлення, супроводу та експлуатації програм. Програмна документація є одним з видів технічної документації.

Комплекс державних стандартів, що встановлюють взаємопов'язані правила розробки, оформлення та обігу програм і програмної документації називається «Єдина система програмної документації» .

При розробці даного програмного забезпечення буде використовуватись спіральна модель життєвого циклу.

Розробка ітераціями відображає об'єктивно існуючий спіральний цикл створення системи. Неповне завершення робіт на кожному етапі дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки відсутню роботу можна буде виконати на наступній ітерації. Головне ж завдання - щонайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Виходячи з можливості внесення змін, як в процес, так і в проміжний продукт було створено спіральну модель ЖЦ .

Внесення змін орієнтоване на задоволення потреби користувачів одразу, як тільки буде встановлено, що створені артефакти або елементи документації не відповідають дійсному стану розробки.

Дана модель ЖЦ допускає аналіз продукту на витку розробки, його перевірку, оцінку правильності та прийняття рішення про перехід на наступний виток або повернення на попередній виток для доопрацювання на ньому проміжного продукту.

Відмінність цієї моделі від каскадної полягає в можливості багато разів повертатися до процесу формулювання вимог і до повторної розробки версії системи з будь-якого процесу моделі.

Для програмного продукту така модель не дуже підходить з декількох причин. По-перше, висловлення вимог замовником носить суб'єктивний характер, вимоги можуть багаторазово уточнюватися протягом розробки ПС і навіть після завершення та випробовування, і часом може з'ясуватися, що замовник «хотів зовсім інше». По-друге, змінюються обставини та умови використання системи, тому загальновизнаним законом програмної інженерії є закон еволюції, який сформулюємо так: кожна діюча ПС з часом потребує внесення змін або виводиться з експлуатації.

При необхідності внесення змін до системи на кожному витку з метою отримання нової версії системи обов'язково вносяться зміни в заздалегідь зафіксовані вимоги, після чого повертаються на попередній виток спіралі для продовження реалізації нової версії системи з урахуванням усіх змін.

## Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

− діючу програму, яка повністю відповідає даному технічному завданню;

− вихідний програмний код, записаний разом із програмою на оптичний носій інформації.

Прийом програмного забезпечення повинен відбуватися перед комісією з двох чоловік (один з яких – Замовник) у такій послідовності:

− доповідь Виконавця про виконану роботу;

− демонстрація Виконавцем роботи програми;

− контрольні випробовування роботи програми;

− відповіді на запитання і зауваження комісії.

# РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

## Розробка загальної структури і варіантів використання програми

Під час проходження етапу аналізу спочатку було визначено такі питання:

* Хто буде діючими суб'єктами?
* Хто вводитиме інформацію?
* Хто запрошуватиме?
* Чи буде хто-небудь ще взаємодіяти з програмою?
* Чи буде сама програма взаємодіяти з іншими програмами?

В даному випадку з програмою HotelH має працювати тільки один користувач – адміністратор готельного комплексу. Таким чином, один і той самий користувач вводить інформацію, і переглядає її у різних видах.

Наступне, що потрібно було з'ясувати, це групу дій, які ініціюватиме діючий суб'єкт. Виходячи із технічного завдання, діючий суб'єкт – адміністратор буде виконувати такі дії:

– почати роботу з програмою;

* авторизуватися;

– додати запис ;

– видалити запис;

– переглянути базу даних;

– відкрити базу даних;

– пошук даних;

Таким чином, діаграму варіантів використання, яка отримується внаслідок наведеного переліку дій, можна подати так, як це показано на рисунку 2.1.



Рисунок 2.1 – Діаграма варіантів використання для програми HotelH.

**Запустити програму**. Ця дія, здавалося б, є дуже очевидною для того, щоб про неї зовсім не згадувати, але все ж таки... Коли запускається програма, на екран повинно виводитися меню, з якого користувач може вибрати потрібну дію. Це може називатися екраном інтерфейсу користувача.

**Авторизація.** На екрані має відобразитися форма авторизації в якій адміністратор повинен підтвердити свою особу,після чого зможе продовжити роботу з самою програмою.

**Додати запис.** На екрані має відобразитися форма,у якій програма добавляє запис в вибрану таблицю (форма заповнення залежить від вибраної таблиці) та дві кнопки :

* добавити;
* закрити вікно;

**Видалити запис.** На екрані має відобразитися форма,у якій програма видаляє запис з вибраної таблиці та поле у якому вказується номер запису за 2 кнопки :

* видалити;
* закрити;

**Переглянути базу даних.** На екрані має відобразитися форма,у якій програма виводить дані з вибраної таблиці та кнопка:

* закрити;

**Відкрити базу даних.** На екрані має відобразитися вікно програми Провідник у якій потрібно вибрати файл з розширенням “.db” для перегляду інформації з обраної бази даних,а саме з вибраних стовбців і таблиць.

**Пошук даних.** На екрані є форма,у якій ми можемо записувати які дані ми хочемо побачити,але пошук виконується з вибраної таблиці і її стовбця.

## Розробка системи класів

Наступний етап розробки програмного забезпечення – етап проектування. Даний етап поділяється на логічне та фізичне проектування. Логічне полягає у розробці структури класів, коли визначаються поля для збереження складових об'єктів, алгоритми методів, що описують поведінку об'єктів. Насамперед необхідно передбачити класи, з яких буде складатися програма. Для цього спочатку проаналізуємо перелік іменників з опису варіантів використання, потім внесемо деякі їх уточнення, визначимо їх атрибути, перейдемо від дієслів до повідомлень, і, на завершення, розробимо діаграми класів і послідовностей.

**Аналіз переліку іменників з опису варіантів використання**. Розглянемо перелік усіх іменників, які візьмемо з опису варіантів використання:

1. База даних.
2. Головне вікно.
3. Вікно оформлення білету.
4. Вікно білети.
5. Вікно маршрути.
6. Вікно автобуси.
7. Вікно водії.
8. Вікно диспетчери.

**Уточнення переліку іменників.** З різних причин багато іменників не зможуть стати класами. Давайте проведемо відбір тільки тих іменників, які можуть претендувати на те, щоб бути класом.

Одним із таких претендентів може бути клас “База даних”, який буде містити підключення до сервера на якому буде зберігатись вся інформація в таблицях: таблиця білети, таблиця маршрути, таблиця автобуси, таблиця водії, таблиця диспетчери, таблиця перевезення. Наступні 6 класів це класи вікон, які містять інформацію про усі функції вікон і їх можливості. Отже, складемо перелік класів, які було тільки, що уточнено:

1. База даних.
2. Головне вікно.
3. Вікно оформлення білету.
4. Вікно білети.
5. Вікно маршрути.
6. Вікно автобуси.
7. Вікно водії.
8. Вікно диспетчери.

**Визначення** **атрибутів**. Багато іменників, яким відмовлено в реєстрації як кандидатами у класи, будуть потенційними кандидатами в атрибути (компонентні дані) класів.

**Перехід від дієслів до повідомлень.** Тепер подивимося, що нам дають варіанти використання для з'ясування того, якими повідомленнями обмінюватимуться класи. Оскільки повідомлення – це, по суті справи, є викликом методу в об'єкті, то визначення повідомлень зводиться до визначення методів класу, що приймає те або інше повідомлення. Як і у випадку з іменниками, далеко не кожне дієслово стає кандидатом у повідомлення.

## Розробка методів

## Проєктування і опис інтерфейсу користувача

## Опис файлової структури програми

## Опис структури бази даних програми

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

ВИСНОВКИ

Текст

ПЕРЕЛІК ПОСИЛАНЬ

1. Грицюк Ю.І. Рак Т.Є. Об’єктно-орієнтоване програмування мовою С++ : навч. посіб. Львів : Вид-во Львівського ДУ БЖД, 2011. 404 с.
2. Вайсфельд М. Объектно-ориентированное мышление. СПб. : Питер, 2014. 304 с.
3. Build with Qt. URL: https://www1.qt.io/built-with-qt (дата звернення: 15.12.2019).
4. EVILEG : вебсайт. URL: https://evileg.com/uk/knowledge/qt (дата звернення: 15.12.2019).

Додаток А  
Діаграма варіантів використання

Додаток Б  
Лістинг файлу «program.h»

/////////////// Лістинги програм оформляти цим шрифтом!!!! ///////////////

/////////////////////підключення бібліотек//////////////////////

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <graphics.h>

#include <dos.h>

#include <math.h>

#include <dir.h>

/////////////// Лістинги програм оформляти цим шрифтом!!!! ///////////////

. . .

/////////////////////Оголошення класу ХХХ/////////////////////

. . .

Додаток В  
Лістинг файлу «program.cpp»

Додаток Г  
Компакт-диск із програмним продуктом