Міністерство освіти і науки України Національний університет «Львівська політехніка» Інститут підприємництва і перспективних технологій

Кафедра інформаційних систем і технологій

# КУРСОВА РОБОТА

з дисципліни «ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему «Програма обліку виплат заробітної плати співробітникам підприємства»

Студента групи КНМС-23 спеціальності «122 Комп’ютерні науки»

Леськів Дмитро Романович

Керівник кандидат педагогічних наук,

доцент Кобильник Т.П.

Національна шкала:

Кількість балів:

Члени комісії

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

Дрогобич-2025

# Зміст

[Вступ 3](#_bookmark0)

1. [Аналіз і проектування 4](#_bookmark1)
   1. [Постановка задачі 4](#_bookmark2)
   2. [Алгоритм розв’язування 4](#_bookmark3)
   3. [Структурна модель 5](#_bookmark4)
   4. [Об’єктна модель 5](#_bookmark5)
2. [Розробка програми 7](#_bookmark8)
   1. [Призначення програми 7](#_bookmark9)
   2. [Вибір середовища програмування 7](#_bookmark10)
   3. [Структура бази даних 8](#_bookmark11)
   4. [Проектування інтерфейсу 8](#_bookmark13)
   5. [Структура програми 9](#_bookmark14)
   6. [Контрольний приклад 12](#_bookmark17)

[Висновок 14](#_bookmark18)

[Список використаних джерел 15](#_bookmark19)

[Додаток 16](#_bookmark24)

# Вступ

Сучасні підприємства потребують ефективних і надійних систем для обліку виплат заробітної плати своїм співробітникам. Комп'ютеризація даного процесу дозволяє значно підвищити ефективність роботи бухгалтерії, зменшити кількість помилок, а також забезпечити прозорість фінансових операцій.

Об'єктно-орієнтоване програмування пропонує широкі можливості для розробки подібних систем завдяки моделюванню реальних об'єктів через класи, що дозволяє створювати гнучкі, масштабовані та надійні програмні рішення. Використання принципів інкапсуляції, успадкування та поліморфізму дозволяє створити програмний продукт, який легко підтримувати, оновлювати та розширювати.

**Предмет роботи**: інформаційна система обліку виплат заробітної плати співробітникам підприємства.

**Мета роботи**: створити об'єктно-орієнтовану інформаційну систему обліку виплат заробітної плати, яка дозволить автоматизувати процес нарахування та обліку заробітної плати.

Відповідно до предмету та мети роботи було поставлено такі завдання:

* здійснити аналіз задачі та проектування системи, що передбачає постановку задачі, розроблення алгоритму, об'єктної та структурної моделі;
* розробити програму, що передбачає вибір середовища програмування, визначення структури даних, проектування інтерфейсу, визначення структури програми та її тестування.
  1. **Аналіз і проектування**
  2. **Постановка задачі**

Необхідно створити програму «Телефонний довідник». У програмі необхідно Необхідно створити програму обліку виплат заробітної плати співробітникам підприємства. У програмі необхідно передбачити диференціацію прав доступу для адміністраторів і звичайних користувачів (співробітників).

Основні функціональні вимоги:

1. Реєстрація та авторизація користувачів
2. Керування обліковими записами користувачів (додавання, видалення)
3. Ведення обліку нарахувань заробітної плати
4. Перегляд історії виплат для кожного співробітника

Система має забезпечувати:

* Захист інформації через механізми автентифікації
* Зручний інтерфейс для взаємодії з користувачем
  1. **Алгоритм розв'язування**

Програма базується на об'єктно-орієнтованому підході, де основними класами є User, PayrollRecord, Authentication, FileManager та UserInterface. Дані зберігаються у текстових файлах.

**Основний алгоритм роботи програми:**

* 1. Ініціалізація файлової системи та перевірка наявності необхідних файлів
  2. Відображення привітального екрану
  3. Відображення меню автентифікації
  4. Логіка авторизації:
* Якщо користувач увійшов як адміністратор – показати адміністративне меню
* Якщо користувач увійшов як звичайний співробітник – показати меню співробітника
  1. Обробка вибору користувача залежно від контексту меню
  2. Виконання відповідних операцій із даними
  3. Завершення роботи програми

**Алгоритм авторизації:**

1. Запит логіна та пароля
2. Перевірка наявності користувача в базі
3. Валідація облікових даних
4. Надання відповідних прав доступу

**Алгоритм виплати заробітної плати:**

1. Отримання списку всіх користувачів
2. Для кожного користувача:

* Перевірка базової заробітної плати
* Запит додаткової інформації (бонуси, коментарі)
* Створення запису про виплату
* Збереження запису в базі даних
  1. **Структурна модель**

Структуру програмного комплексу можна представити у вигляді схеми взаємодії основних модулів, як показано на Рис. 1.



**Блок керування**

Додавання співробітника/адміна

Авторизація

Видалення користувача

Перегляд користувачів

Виплата зарплати



Співробітник



Адмін

Перегляд всіх виплат

Вихід з акаунту

Додавання співробітника/адміна

Перегляд користувачів

Вихід з акаунту

# Рис. 1. Структурна модель програми

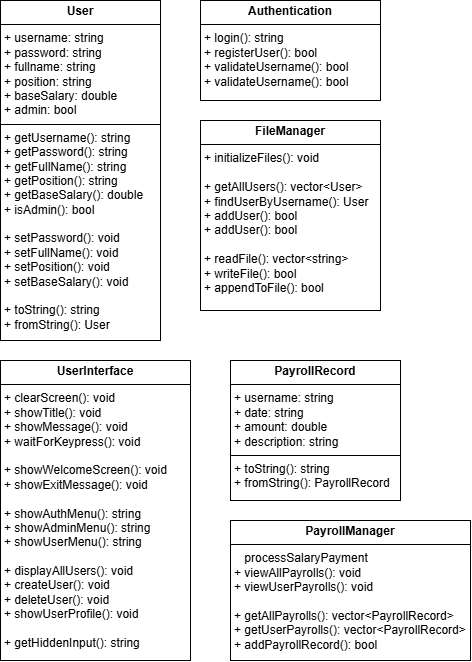
Основні модулі програми:

* **Main** - головний модуль, що координує роботу всіх інших компонентів
* **Authentication** - відповідає за реєстрацію та авторизацію користувачів
* **UserInterface** - відповідає за взаємодію з користувачем
* **FileManager** - відповідає за операції з файлами
* **PayrollManager** - відповідає за обробку виплат заробітної плати
  1. **Об'єктна модель**

Об'єктна модель системи представлена такими основними класами:

1. **User** - клас для представлення користувача системи:
   * Атрибути: username, password, fullName, position, baseSalary, admin
   * Методи: геттери, сеттери, toString, fromString
2. **Authentication** - клас для керування автентифікацією:
   * Методи: login, registerUser, validateUsername, validatePassword
3. **FileManager** - клас для роботи з файлами:
   * Методи: initializeFiles, getAllUsers, findUserByUsername, addUser, deleteUser, readFile, writeFile, appendToFile
4. **PayrollRecord** - структура для зберігання інформації про виплату:
   * Атрибути: username, date, amount, description
   * Методи: toString, fromString
5. **PayrollManager** - клас для керування виплатами:
   * Методи: processSalaryPayment, viewAllPayrolls, viewUserPayrolls, getAllPayrolls, getUserPayrolls, addPayrollRecord
6. **UserInterface** - клас для роботи з інтерфейсом користувача:
   * Методи: clearScreen, showTitle, showMessage, waitForKeypress, showWelcomeScreen, showExitMessage, showAuthMenu, showAdminMenu, showUserMenu, displayAllUsers, createUser, deleteUser, showUserProfile, getHiddenInput

Діаграма класів показана на Рис. 2.



# Рис. 2. Діаграма класів

# Розробка програми

* 1. **Призначення програми**

Програма "Облік виплат заробітної плати співробітникам підприємства" призначена для автоматизації процесів нарахування та обліку заробітної плати. Система надає можливість:

* Адміністраторам:
  + Керувати обліковими записами користувачів
  + Нараховувати зарплату співробітникам
  + Переглядати звіти про всі виплати
  + Керувати системою в цілому
* Співробітникам (звичайним користувачам):
  + Переглядати свій профіль
  + Отримувати інформацію про нараховану заробітну плату
  + Переглядати історію власних виплат

# Вибір середовища програмування

# Для розробки програми було обрано мову програмування C++ з компілятором Microsoft Visual C++, що є частиною Visual Studio. Вибір зумовлений такими факторами:

1. **Об'єктно-орієнтований підхід** - C++ повністю підтримує принципи ООП, що дозволяє створити гнучку і розширювану архітектуру програми.
2. **Швидкодія** - завдяки компіляції в машинний код, програма виконується швидко і з мінімальним використанням ресурсів.
3. **Відокремлене компілювання** - модульна структура програми з розділенням на .h та .cpp файли дозволяє ефективно організувати проект.
4. **Надійність** - строга типізація та контроль помилок на етапі компіляції.
5. **Кросплатформеність** - можливість компіляції програми під різні операційні системи.

Середовище розробки (IDE) Microsoft Visual Studio надає зручні інструменти для написання коду, налагодження, тестування та розгортання програми.

* 1. **Структура бази даних**

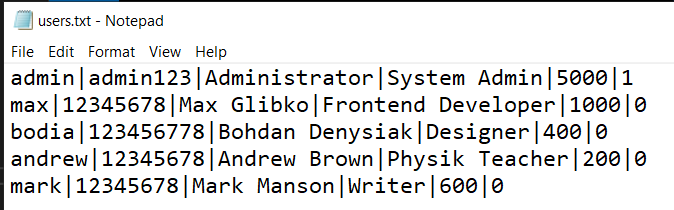
У програмі для зберігання даних використовуються текстові файли, які відіграють роль простої бази даних:

1. **users.txt** - файл для зберігання інформації про користувачів системи:

* Формат запису: username|password|fullName|position|baseSalary|isAdmin
* Приклад: admin|admin123|Administrator|System Admin|5000|1

1. **payroll.txt** - файл для зберігання інформації про виплати:

* Формат запису: username|date|amount|description
* Приклад: dmytro|2025-05-08|600|Monthly salary with additional 100$ for better team work.

Структура бази даних проілюстрована на Рис. 3.

# 

# Рис. 3. Структура бази даних

* 1. **Проектування інтерфейсу**

Програма має консольний інтерфейс користувача, організований у вигляді системи меню:

1. **Меню автентифікації (AuthMenu)** - дозволяє користувачу увійти в систему або зареєструватися.
2. **Меню адміністратора (AdminMenu)** - надає доступ до адміністративних функцій.
3. **Меню користувача (UserMenu)** - надає доступ до функцій звичайного користувача.

Приклад меню автентифікації показано на Рис. 4.

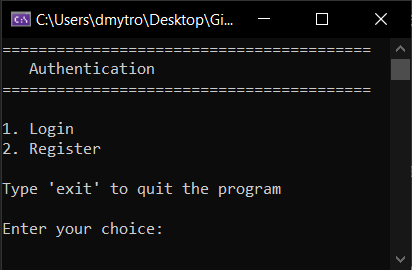


Рис. 4. Меню автентифікації

Приклад меню адміністратора показано на Рис. 5.



Рис. 5. Меню адміністратора

Приклад меню користувача показано на Рис. 6.



Рис. 6. Меню користувача

* 1. **Структура програми**

Структура проекту складається з таких файлів:

1. **main.cpp** - головний файл програми, містить функцію main() і керуючу логіку програми.
2. **authentication.h / authentication.cpp** - класи для реалізації функцій автентифікації користувачів.
3. **user.h / user.cpp** - клас для представлення користувача системи.
4. **fileManager.h / fileManager.cpp** - класи для роботи з файлами даних.
5. **payroll.h / payroll.cpp** - класи для роботи з виплатами зарплати.
6. **userInterface.h / userInterface.cpp** - класи для реалізації інтерфейсу користувача.
7. **users.txt** - файл для зберігання даних про користувачів.
8. **payroll.txt** - файл для зберігання даних про виплати зарплати.

Основні класи:

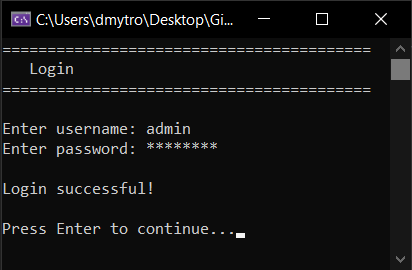
1. **User** - представляє користувача системи з атрибутами:
   * username - логін
   * password - пароль
   * fullName - повне ім'я
   * position - посада
   * baseSalary - базова заробітна плата
   * admin - статус адміністратора
2. **PayrollRecord** - представляє запис про виплату з атрибутами:
   * username - логін користувача
   * date - дата виплати
   * amount - сума
   * description - опис виплати
3. **Authentication** - відповідає за автентифікацію користувачів і містить методи:
   * login() - авторизація користувача
   * registerUser() - реєстрація нового користувача
   * validateUsername() і validatePassword() - валідація вхідних даних
4. **FileManager** - відповідає за роботу з файлами і містить методи:
   * initializeFiles() - ініціалізація файлової системи
   * getAllUsers() - отримання списку всіх користувачів
   * findUserByUsername() - пошук користувача за логіном
   * addUser(), updateUser(), deleteUser() - операції з користувачами
   * readFile(), writeFile(), appendToFile() - низькорівневі операції з файлами
5. **PayrollManager** - відповідає за обробку виплат зарплати і містить методи:
   * processSalaryPayment() - обробка нарахування зарплати
   * viewAllPayrolls() - перегляд всіх виплат
   * viewUserPayrolls() - перегляд виплат конкретного користувача
   * getAllPayrolls(), getUserPayrolls(), addPayrollRecord() - допоміжні методи
6. **UserInterface** - відповідає за взаємодію з користувачем і містить методи:
   * showAuthMenu(), showAdminMenu(), showUserMenu() - відображення меню
   * displayAllUsers(), createUser(), deleteUser() - операції з користувачами
   * clearScreen(), showTitle(), showMessage() - допоміжні методи виведення

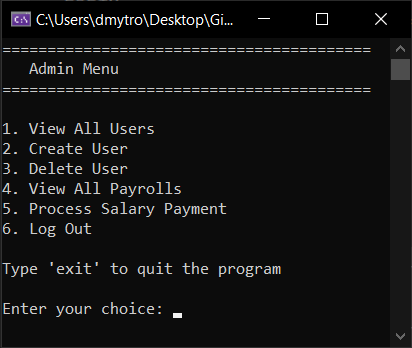
# Контрольний приклад

Розглянемо декілька типових сценаріїв роботи з програмою для демонстрації її функціональності

Сценарій 1: Вхід в систему як адміністратор

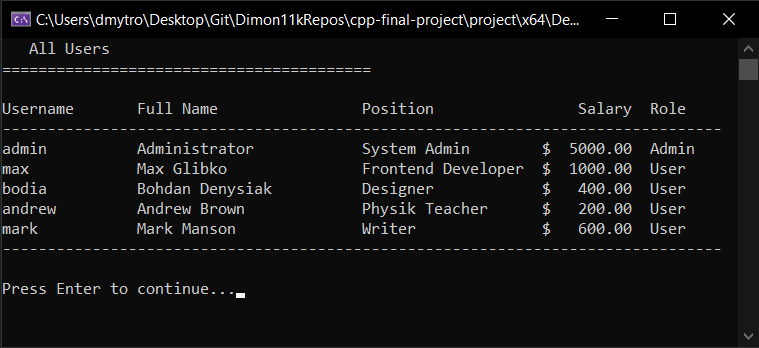
1. При запуску програми відображається привітальне вікно та меню автентифікації.
2. Користувач вибирає опцію "1. Увійти в систему".
3. Вводить логін "admin" та пароль "admin123".
4. Система перевіряє дані та надає доступ до адміністративного меню.





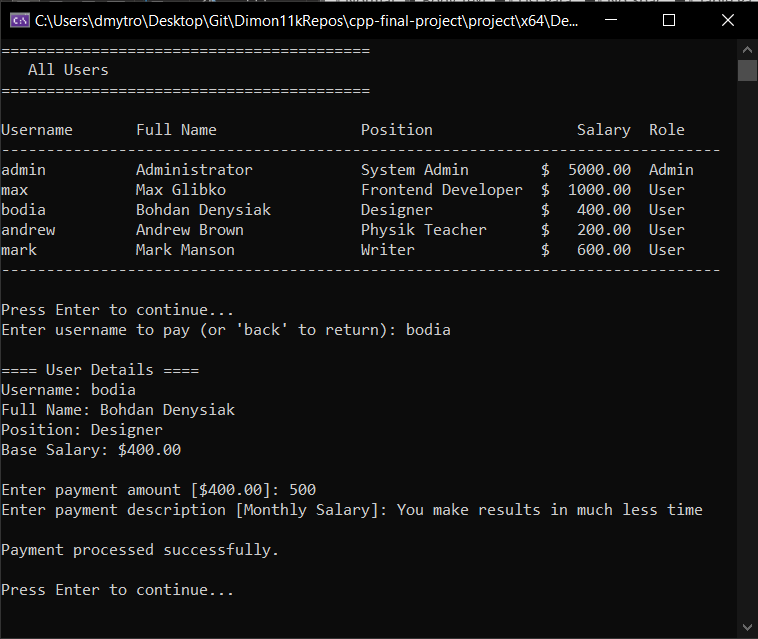
Сценарій 2: Перегляд списку користувачів

1. Адміністратор у своєму меню вибирає опцію "1. Переглянути всіх користувачів".
2. Система відображає таблицю з даними всіх користувачів.



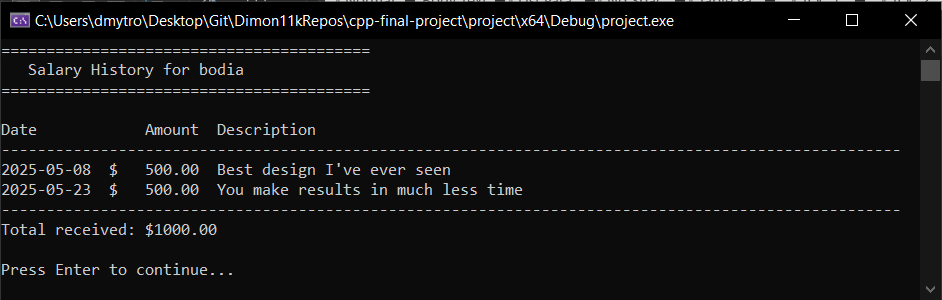
Сценарій 3: Нарахування зарплати

1. Адміністратор вибирає опцію "5. Нарахувати зарплату".
2. Система показує список користувачів і пропонує нарахувати зарплату для кожного.
3. Адміністратор вводить суму та опис для користувача.
4. Після завершення нарахування система повідомляє про успішну операцію.



Сценарій 4: Перегляд виплат користувачем

1. Користувач входить у систему, використовуючи свій логін і пароль.
2. У меню користувача вибирає опцію "2. Переглянути мої виплати".
3. Система відображає історію виплат для даного користувача.



# Висновок

# У ході виконання курсової роботи було розроблено програму для обліку виплат заробітної плати співробітникам підприємства з використанням об'єктно-орієнтованого підходу на мові програмування C++.

# Програма дозволяє автоматизувати процес нарахування та обліку заробітної плати, диференціюючи рівні доступу між адміністраторами та звичайними користувачами. Система забезпечує зручну роботу з даними про користувачів та виплати, зберігаючи їх у текстових файлах.

# В ході роботи над проектом були застосовані такі принципи об'єктно-орієнтованого програмування:

# Інкапсуляція - кожен клас об'єднує дані та методи їх обробки

# Абстракція - кожен клас представляє абстрактне поняття предметної області

# Модульність - програма розділена на логічні модулі, що спрощує розробку та підтримку

# Розроблена система є надійною, зручною у використанні та може бути легко розширена додатковими функціями у майбутньому. Всі поставлені завдання були успішно виконані, а проект повністю відповідає визначеним вимогам.

# Практичні результати тестування підтвердили коректність роботи програми та її відповідність поставленим задачам.

# Додаток

*#include* <iostream>

*#include* <string>

*#include* "user.h"

*#include* "fileManager.h"

*#include* "payroll.h"

*#include* "userInterface.h"

*#include* "authentication.h"

*using* *namespace* std;

int main() {

*// Ініціалізація*

  FileManager::initializeFiles();

  UserInterface::showWelcomeScreen();

  bool isRunning = *true*;

  User\* currentUser = *nullptr*;

*while* (isRunning) {

*if* (!currentUser) {

*// Якщо користувач НЕ увійшов в систему - показати меню авторизації*

      string authChoice = UserInterface::showAuthMenu();

*if* (authChoice == "1") {

*// Увійти*

        currentUser = Authentication::login();

      }

*else* *if* (authChoice == "2") {

*// Зареєстуватися*

        Authentication::registerUser();

      }

*else* *if* (authChoice == "exit") {

        isRunning = *false*;

      }

    }

*else* {

*// Користувач увійшов в систему - показати відповідне меню*

*if* (currentUser->isAdmin()) {

        string adminChoice = UserInterface::showAdminMenu();

*if* (adminChoice == "1") {

          UserInterface::displayAllUsers();

        }

*else* *if* (adminChoice == "2") {

          UserInterface::createUser();

        }

*else* *if* (adminChoice == "3") {

          UserInterface::deleteUser();

        }

*else* *if* (adminChoice == "4") {

          PayrollManager::viewAllPayrolls();

        }

*else* *if* (adminChoice == "5") {

          PayrollManager::processSalaryPayment();

        }

*else* *if* (adminChoice == "6") {

          delete currentUser;

          currentUser = *nullptr*;

          UserInterface::showMessage("Logged out successfully.");

        }

*else* *if* (adminChoice == "exit") {

          isRunning = *false*;

        }

      }

*else* {

*// Меню користувача (співробітника)*

        string userChoice = UserInterface::showUserMenu();

*if* (userChoice == "1") {

*// Подивитися профіль*

          UserInterface::showUserProfile(currentUser);

        }

*else* *if* (userChoice == "2") {

*// Записи зарплат*

          PayrollManager::viewUserPayrolls(currentUser->getUsername());

        }

*else* *if* (userChoice == "3") {

*// Вихід*

          delete currentUser;

          currentUser = *nullptr*;

          UserInterface::showMessage("Logged out successfully.");

        }

*else* *if* (userChoice == "exit") {

          isRunning = *false*;

        }

      }

    }

  }

*// Очищення*

*if* (currentUser) {

    delete currentUser;

  }

  UserInterface::showExitMessage();

*return* *0*;

}