Plant

Polina Puchek

December 2024

Зміст

1 Вступ **1**

2 Реалізація на Сі **1**
2.1 Структури (Plant.h) 1
2.2 Опис функцій (Plant.c) 2

3 Реалізація на C++ **3**
3.1 Класии(Plant.h) 3
3.2 Опис методів (Plant.cpp) 4

4 Запуск програм **5**

5 Висновок **5**

1 Вступ

Метою цієї роботи є розробка програмного забезпечення, яке моделює систему обліку на заводі. Завдання передбачає створення двох версій програми: на основі структур у мові C та за допомогою об'єктно-орієнтованого підходу в мові C++. Обидві версії мають забезпечувати основну функціональність: облік персоналу, обчислення заробітної плати, а також розрахунок вартості обладнання, закріпленого за працівниками.

Важливість цієї задачі полягає у формуванні базових навичок розробки систем для управління виробничими процесами. У межах роботи передбачено створення моделей працівників, їхніх професій, а також засобів обліку обладнання, з використанням сучасних методів програмування.

2 Реалізація на Сі

2.1 Структури (Plant.h)

Мій код демонструє використання концепцій наслідування та агрегації, хоча й не у формі об'єктно-орієнтованого програмування (ООП). Замість класів і підкласів використано структури, які дозволяють організувати дані у вигляді ієрархії. Структури:

1. **Базова інформація про працівників (Person):** Базова структура, яка описує основну інформацію про працівників і керівників, включаючи ім'я та прізвище.
2. **Працівники (Worker):** Використовує базову структуру Person для зберігання даних про працівника. Додатково включає інформацію про професію, стаж роботи, обсяг виробітку, використане обладнання та ім'я керівника.

3. **Керівники (Head):** Успадковує структуру Person і дає інформацію про професію та стаж.
4. **Працівники неповного дня (PartTimer):** Є похідною від структури Worker, зберігаючи основні характеристики працівника, але відображаючи специфіку неповного робочого дня.
5. **Облік обладнання (Equipment):** Містить дані про назву, вартість і кількість одиниць обладнання, що використовується підлеглими керівників.
6. **Професії (Profession):** Описує назву професії та базову ставку заробітної плати, яка є основою для розрахунку заробітку.

   Також зробила додаткову структуру для полегшення роботи:

7. **Облік обладнання за керівником (ManagerEquipment):** Систематизує обладнання, яке використовується працівниками під керівництвом конкретного керівника.

Наслідування реалізовано через повторне використання базових структур, які слугують основою для більш складних. Наприклад: структура Person зберігає базові характеристики, такі як ім'я та прізвище, і використовується як поле в інших структурах, зокрема Worker, Head і PartTimer. Це дозволяє уникнути дублювання коду, адже спільні властивості визначені один раз у базовій структурі. Аналогічно, структура Profession використовується для опису професійних характеристик як працівників, так і керівників.
Агрегація проявляється у включенні однієї структури в іншу як поля. Цей підхід дозволяє створювати складні об'єкти шляхом поєднання простіших. Структури Worker, Head, і PartTimer агрегують структури Person для опису базової інформації про людину, Profession для зберігання професійних характеристик і Equipment для деталізації обладнання, яким користується працівник. ManagerEquipment агрегує масиви об'єктів Equipment для обліку ресурсів, закріплених за працівниками певного керівника. Цей підхід робить структури гнучкими у використанні, адже вони можуть бути використані повторно у різних контекстах.

2.2 Опис функцій (Plant.c)

На початку хочу зауважити, що вся потрібна інформація для роботи поділена на два файла формату CSV:

1. **Plant_test.dat(p).txt**. Вміст файлу: Тип, Ім'я, Прізвище, Професія, Базова ставка, Досвід, Кількість продукції.
2. **Plant_test.dat(e).txt**. Вміст файлу: Ім'я, Прізвище, Керівник, Тип обладнання, Вартість обладнання

Рішення зробити два файла прийнятте для полегшення обробки інформації. Також у файлі містить інформація виключно англійською мовою.

Функції:

- **read_eq_from_file**(використовує файл **Plant_test.dat(e).txt**)

  Зчитує дані про працівників та їхнє обладнання з файлу. Файл відкривається у режимі читання, і кожен рядок, окрім заголовка, обробляється за допомогою sscanf. Дані розбиваються на поля: ім'я, прізвище, ім'я керівника, тип обладнання та його вартість. Якщо вартість обладнання більша за 0, інформація про нього додається до відповідної структури працівника. Інакше поле обладнання очищується. Функція завершується, коли прочитані всі рядки або досягнуто максимального дозволеного числа працівників.

- **calculate_total_equipment_cost** Підраховує сумарну вартість обладнання всіх працівників. Проходить по масиву працівників і додає вартість їхнього обладнання до змінної **total_cost**. Повертає суму як результат.

- **group_equipment_by_head** Групує обладнання за керівниками. Для кожного працівника перевіряє, чи є у нього обладнання. Якщо є, то додає інформацію про це обладнання до списку обладнання керівника. Якщо такого керівника ще немає у списку, створює нового запису. Якщо обладнання вже прив'язане до цього керівника, оновлює його кількість і вартість. Функція також підраховує загальну вартість обладнання для кожного керівника та виводить результати.

- **calculate_employee_salary** Розраховує зарплату працівника за формулою:

  $$\texttt{Зарплата} = \texttt{Базова ставка} + \texttt{Кількість років досвіду} \times 1000 + \texttt{Одна вироблена продукція} \times 10$$

  Використовує дані про працівника із відповідної структури.

- **calculate_part_timer_salary** Розраховує зарплату сумісника за формулою:

  $$\texttt{Зарплата} = \texttt{Базова ставка} + \texttt{Кількість років досвіду} \times 1000 + \texttt{Одна вироблена продукція} \times 5$$

- **calculate_head_salary** Розраховує зарплату керівника за формулою:

  $$\texttt{Зарплата} = \texttt{Базова ставка} \times 1.2 + 1000 \times \texttt{Кількість років досвіду}$$

- **read_workers_from_file**(використовує файл **Plant_test.dat(p).txt**)

  Зчитує працівників, сумісників і керівників із файлу. У кожному рядку визначає тип запису (Worker, PartTimer або Head), розбиває дані на поля та створює відповідну структуру. Записи розподіляються по масивах, відповідно до типу працівника.

- **print_salaries** Виводить зарплати всіх працівників на екран. Для кожного типу працівника (керівники, працівники, сумісники) обчислює зарплату за допомогою відповідних функцій розрахунку та виводить результат у форматованому вигляді.

- **print_full_equipment_inventory** Формує загальний список обладнання, використаного на фабриці. Для кожного працівника перевіряє, чи є у нього обладнання. Якщо тип обладнання вже є в інвентарі, збільшує його кількість. Інакше додає новий запис. Підраховує загальну вартість кожного типу обладнання, враховуючи кількість, та виводить цей список на консоль.

- **write_salary_to_file** Записує дані про зарплати у файл. Для кожної групи працівників (керівники, працівники, сумісники) розраховує зарплату і зберігає її у відповідному форматі. Функція створює або перезаписує файл, забезпечуючи збереження результатів.

- **write_equipment_to_file** Зберігає інформацію про інвентар фабрики у файл. Формує список усіх типів обладнання та їхню загальну вартість і кількість. Також групує обладнання за керівниками, зберігаючи для кожного загальну вартість закріпленого обладнання. Функція забезпечує детальну документацію інвентарю у форматі, зручному для подальшого аналізу.

3 Реалізація на C++

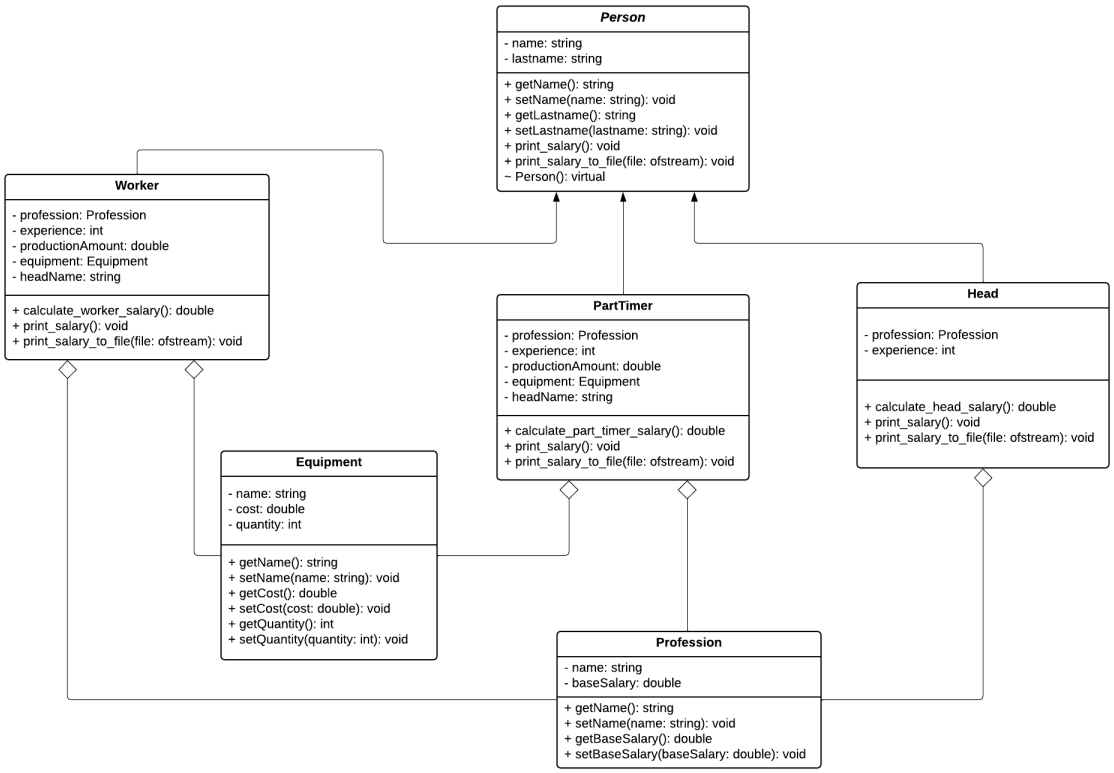
3.1 Класии(Plant.h)

Моя програма використовує кілька класів для представлення різних типів осіб та їх атрибутів. Нижче наведено огляд класів, що описані в моєму коді, та прикріплюю відповідну UML-діаграму для більшої ясності взаємозв'язків.

- **Клас Equipment** відповідає за зберігання інформації про обладнання, яке використовується працівниками.

- **Клас Person** є базовим абстрактним класом, від якого успадковуються інші класи. Він зберігає основну інформацію про особу. Цей клас містить чисто віртуальні методи `print_salary()` та `print_salary_to_file()`, які реалізовуються в похідних класах, що робить його абстрактним і дозволяє створювати загальну структуру для всіх типів осіб.
- **Клас Worker** є похідним від класу Person і зберігає інформацію про звичайного працівника.
- **Клас Head** також є похідним від класу Person і зберігає інформацію про керівника.
- **Клас PartTimer** зберігає інформацію про працівників з неповним робочим днем. Він є похідним від класу Person та містить ті ж атрибути, що й клас Worker.
- **Клас Profession** зберігає інформацію про професію працівника, включаючи назву професії та базову зарплату.

Також у моїй діаграмі класів можна побачити агрегацію між класами. У класах Worker, PartTimer та Head атрибути Profession та Equipment показують агрегацію, оскільки ці об'єкти можуть існувати незалежно від об'єктів класів, які їх містять.



UML-діаграма класів

3.2 Опис методів (Plant.cpp)

Також хочу додати, що як і на Сі, вхідних файлів два. Вони ідентичні до файлів на Сі.
Методи:

- **read_equipment_from_file**: Зчитує дані про працівників та їхнє обладнання з текстового файлу. Кожен запис містить ім'я, прізвище працівника, ім'я керівника, тип обладнання, його вартість і кількість. Метод повертає вектор об'єктів Worker.

- **read_workers_from_file**: Зчитує загальну інформацію про працівників із файлу. Підтримуються три типи працівників: Worker, PartTimer, і Head. Для кожного запису створюється відповідний об'єкт, який додається до вектора вказівників на об'єкти Person.
- **write_salary_and_equipment_to_file**: Записує до файлу інформацію про зарплати працівників і облік обладнання. Включає секції про зарплати кожного працівника, інвентаризацію обладнання та підрахунок загальної вартості обладнання для кожного керівника.
- **print_equipment_in_consol**: Виводить облік обладнання в консоль. Для кожного типу обладнання відображаються загальна кількість і загальна вартість.
- **print_salaries**: Виводить зарплати працівників у консоль. Для кожного працівника викликається відповідний метод розрахунку зарплати залежно від його типу (Worker, PartTimer, Head).
- **calculate_equipment_cost_for_head**: Розраховує загальну вартість обладнання для кожного керівника. Повертає вектор пар, де кожна пара містить ім'я керівника та відповідну вартість.
- **print_salary (реалізований для кожного класу)**: Виводить розраховану зарплату в консоль. Реалізація залежить від типу працівника.
- **print_salary_to_file (реалізований для кожного класу)**: Аналогічний метод print_salary, але записує зарплату до файлу.
- Методи **calculate_worker_salary**, **calculate_head_salary**та **calculate_part_timer_salary**: Розраховують зарплату працівників Worker, PartTimer та Head відповідно за такими самими формулами, що представлені у розділі Сі.

4 Запуск програм

На Сі та C++ після запуску програми користувачеві пропонується вибрати спосіб виведення результатів. Програма може відобразити інформацію на консолі або записати її у файл. При виведенні на консоль користувач бачить список працівників із розрахованими зарплатами, загальну вартість обладнання, яке закріплене за кожним керівником, а також повну інвентаризацію обладнання за його типами. Якщо обрано запис до файлу, програма створює текстовий звіт, який містить усі перелічені дані.

Хочу зазначити, що є деяка відмінність на Сі та C++. Якщо користувач обирає вивід інформації у файл на Сі, то програма створює два файли: перший - облік зарплат, другий - облік обладнання. Якщо на C++, то всю інформацію виводить в один файл. Це було зроблено для того щоб показати, що дані можна виводити в один файл або декілька, це може бути використано для зручності в аналізі даних.

5 Висновок

У ході виконання роботи було створено дві версії програми для моделювання системи обліку на заводі: на основі структур у мові С та з використанням класів у мові C++. У програмі реалізовано базову функціональність обліку заробітної плати працівників залежно від їхньої професії, стажу, а також виробітку. Також передбачено систему обліку обладнання, закріпленого за працівниками.

У версії на С було застосовано структури для побудови моделі, а у версії на C++ реалізовано об'єктно-орієнтований підхід із використанням абстрактного класу "Людина"та його нащадків

"Працівник "Керівник"і "Сумісник". Це забезпечило гнучкість і розширюваність системи.

Під час виконання роботи вдалося закріпити знання з використання структур та класів, а також опанувати роботу з файлами в текстовому форматі CSV. Програма має практичне значення, оскільки може бути основою для більш складних систем управління виробничими процесами.