**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**



**Курсова робота**

з дисципліни «Програмування, частина 2 (Об'єктно-орієнтоване програмування)»

на тему:

**«Базові принципи об’єктно-орієнтованого програмування»**

Індивідуальне завдання: «Залізничний вокзал»

**Виконав:**

ст.гр. КІ-15

Ковальський П.А.

**Прийняв:**

викл.

Козак Н.Б

Львів -2020

***Анотація***

Метою курсової роботи є створення спрощеної моделі функціонування залізничного вокзалу. Написання програмного забезпечення було виконано мовою програмування C++ із використанням об’єктно-орієнтованого підходу для формування програми у вигляді набору сутностей та об’єктів, що взаємодіють. Програма включає у собі можливість роботи із текстовими файлами, а саме: запис та зчитування даних, пошук необхідної інформації та її видалення, а також сортування за певною характеристикою для зручнішого використання та керування інформацією. Програмне забезпечення передбачає кілька рівнів доступу, що дозволяє встановити роль користувача та надати йому лише ті можливості керування інформацією, що необхідні конкретно даній особі. Інтерфейс програми написано українською мовою.

**Зміст**

**Завдання на курсову роботу ……………………………………………………… 3**

**Вступ …………………………………………………………………………………. 4**

**1. Визначення поняття ООП, його основні принципи та обґрунтування вибору його технологій …………………………………………………………………....... 5**

1.1 ООП як основа сучасного програмування …………………………............... 5

1.2 Визначення ООП ………………………………………………………………. 6

1.3 Фундаментальні поняття ООП ……………………………………………..… 7

1.4 Переваги вибраного методу ………………………………………………...... 8

**2. Аналіз та розробка алгоритму згідно індивідуального завдання …………… 9**

2.1 Загальний опис поставленої задачі …………………………………………… 9

2.2 Пасажир ………………………………………………………………………… 9

2.3 Персонал ………………………………………………………………….......... 10

2.4 Потяги ………………………………………………………………………….. 11

2.5 Визначення основних задач покладених в основу програми ……………… 12

2.6 Алгоритм виконання роботи програми ………………………………............ 14

**3. Реалізація алгоритму у програмному коді ……………………………………. 15**

**4. Тестування програми ……………………………………………………….......... 21**

**5. Аналіз продуктивності роботи програми ………………………………………. 23**

**Висновок ……………………………………………………………………………… 25**

**Література ……………………………………………………………………….......... 26**

**Додаток А ……………………………………………………………………….......... 30**

***Завдання на курсову роботу***

Згідно із отриманим індивідуальним завданням метою курсової роботи є реалізація мовою C++ із застосуванням об’єктно-орієнтованого підходу предметної галузі “Залізничний вокзал”. Програма повинна підтримувати можливості роботи з файлами : запис, зчитування, пошук, видалення та оновлення даних, а також їх сортування за певним критерієм. Для типу користувачів “Клієнт” передбачена система пунктів для кращої орієнтації у програмі. Для користувачів, що мають права адміністратора повинні бути реалізовані методи, що дозволяють керувати даними.

***Вступ***

**Об’єктно-орієнтоване програмування** (ООП) — одна з парадигм програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою. В основі ООП покладено три основні положення: інкапсуляцію, успадкування та поліморфізм. Однією з переваг ООП є краща модульність програмного забезпечення, що дозволяє групам розробників працювати незалежно один від одного, розробляючи окремі функціональні компоненти програми . Попри те, що парадигма ООП з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм та покращити їх модульність. Однією з таких мов програмування стала мова С++. У 1990-х р. вона стала однією із найпопулярніших мов програмування загального призначення. Розробником цієї мови став професор Техаського університету Б’єрн Страуструп. На початку свого існування ця мова дістала назву «Сі з класами». Пізніше у 1983 році Страуструп перейменував мову на C++. У 1998 році було ратифіковано міжнародний стандарт мови С++: ISO/IEC 14882 «Standard for the C++ Programming Language». Сьогодні цю мову використовують для розробки програмного забезпечення, написання драйверів, системного програмування, потужних серверних та клієнтських програм. Мова C++ мала значний вплив на інші мови програмування такі як Java, C# та ін.

**Метою виконання** курсової роботи є засвоєння основних методів та алгоритмів об’єктно-орієнтованого програмування мови С++ та їх практичне застосування у описі сутностей конкретної сфери.

**1. Визначення поняття ООП, його основні принципи та обгрунтування його технологій.**

**1.1 ООП як основа сучасного програмування**

Раніше програми розроблялися із застосуванням процедурного підходу, що полягав у поділі проекту на окремі більш дрібні складові – функції. Із розростанням проектів та ускладненням завдань з’явилась необхідність у більш раціональному підході у написанні програмного забезпечення та ефективному поділі поставленого завдання таким чином, щоб не виникало проблем при розробці окремих складових проекту. Для подолання цієї проблеми було розроблено структурне програмування. Структурне програмування, на відміну від свого попередника — стихійного програмування, має певні правила та засоби, які суттєво вдосконалили процес написання програм. Однак через обмеженість структурного програмування дуже важко писати програми ще більшої складності, яких вимагає сучасний ринок праці. Принципово новим підходом стало впровадження об’єктно-орієнтованого програмування. Першою мовою програмування, в якій були запропоновані базові поняття, які згодом об’єдналися у парадигму, була Симула, але термін “об'єктна орієнтованість” не можна було застосувати в контексті використання цієї мови. У 1967 році були запропоновані революційні ідеї: об'єкти, класи, віртуальні методи і ін., проте це не було сприйнято сучасниками як щось революційне та здатне змінити підхід до розробки програмного забезпечення. Погляд на програмування «під іншим кутом» (відмінним від процедурного) запропонували Алан Кей і Ден Інгаллс в мові Smalltalk. Тут поняття класу стало основотвірною ідеєю для всіх інших конструкцій мови (тобто клас в Смолток є примітивом, за допомогою якого описані більш складні конструкції). Саме ця мова стала першою широко поширеною об'єктно-орієнтованою мовою програмування. Станом на сьогодні кількість прикладних мов програмування (список мов), що реалізують об'єктно-орієнтовану парадигму, є найбільшою порівняно з іншими парадигмами.

Отже, впровадження ООП стало вагомим внеском у розвитку сучасного програмування раціональним шляхом мінімізації коду та покращенням ефективності.

**1.2 Визначення ООП**

Об'єктно-орієнтоване програмування (ООП) - методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи можуть утворюють ієрархію спадкування. Лука Карделла і Мартін Абаді побудували теоретичне обгрунтування ООП і класифікацію на основі цього обгрунтування. Вони відзначають, що виділені ними поняття і категорії разом зустрічаються далеко не у всіх об’єктно-орієнтованих мовах, більшість мов підтримують лише підмножини теорії, а часом і своєрідні відхилення від неї.

Ключові поняття:

* квантифікація змінних типу (універсальна, екзистенційна, обмежена);
* підтипізація (англ. subtyping - відносини «супертип-підтип»);
* включення (англ. subsumption) - окремий випадок підтипізації;
* об'єкт;
* об'єктний тип (фактично, спеціальна форма «запису», в число полів якої входять функції першого класу);
* клас.

Клас – це сутність, що описує абстрактну поведінку. Об'єктні типи будуються на основі класу за допомогою додавання різних приватних полів і методів. Об'єкт (тобто значення об'єктного типу, традиційно називається «екземпляром класу») породжується конструктором на основі початкових параметрів.

**1.3 Фундаментальні поняття ООП**

До фундаментальних понять (принципів), використаних у переважній більшості визначень об'єктно-орієнтованого програмування належать:

* *Абстракція даних* – виділення лише значущої інформації і виключення її з розгляду незначною. В ООП розглядають лише абстракцію даних (нерідко називаючи її просто «абстракцією»), маючи на увазі набір значущих характеристик об'єкта, доступних решті частини програми.
* *Інкапсуляція –* властивість системи, що дозволяє об'єднати дані і методи, що працюють з ними, в класі. Одні мови (наприклад, С ++, Java або Ruby) ототожнюють інкапсуляцію з приховуванням, але інші (Smalltalk, Eiffel, OCaml) розрізняють ці поняття.
* *Спадкування –* властивість, що дозволяє описати новий клас на основі вже існуючого, що передбачає часткове або абсолютне отримання властивостей чи методів батьківського класу. Клас, від якого відбувається спадкування, називається базовим, батьківським або суперкласом. Новий клас – нащадком, спадкоємцем, дочірнім або похідним класом.
* *Поліморфізм підтипів –* властивість, що дозволяє використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта. Інший вид поліморфізму – параметричний.
* *Клас –* абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для осіб, що не мають відношення до програмування, проте знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються членами даного класу.
* *Об’єкт –* сутність в адресному просторі обчислювальної системи, що з'являється при створенні екземпляра класу (наприклад, після запуску результатів компіляції і зв'язування вихідного коду на виконання).

**1.4 Переваги обраного методу.**

Оптимальність та раціональність вибору того чи іншого методу є однією із фундаментальних основ при розробленні програмного забезпечення для розв’язання конкретної задачі. Від вибору підходу при створенні програми залежить час виконання програми, ресурсозатратність і т.д. Тому грамотне та доцільне рішення може суттєво спростити процес розроблення та вдосконалення програмного забезпечення в майбутньому. Чим більшою стає програма, тим гостріше постає проблема про важливість чітко окресленого поділу на компоненти, абстрагуючись від незначних деталей і в той же час проектуючи складові так, щоб вони могли описати сутності, на базі яких буде передбачено створення об’єктів та описано правила їх взаємодії. Однією із основних переваг використання саме об’єктно-орієнтованого підходу є можливість створювати розширювані та модульні системи, що й дає перевагу над традиційними методами програмування. Розширюваність передбачає, що існуючу систему можна доповнити новими компонентами, причому без внесення до неї будь-яких змін. Компоненти можуть бути додані на етапі виконання.

**2. Аналіз та розробка алгоритму згідно індивідуального завдання**

**2.1 Загальний опис поставленого завдання**

Запропонованим завданням для моєї курсової роботи є створення сутності «Залізничний вокзал». Перш за все, дану сутність слід розглядати як взаємодію кількох важливих його складових, зокрема потягів, пасажирів та працівників самого залізничного вокзалу. Усі ці частини є невід’ємними для мінімального опису його роботи. Розглянемо кожен компонент залізничного вокзалу зокрема.

**2.2 Пасажири**

Поняття «пасажир» має під собою декілька визначень, а саме:

* особа, яка користується транспортним засобом, але не причетна до керування ним;
* особа, яка користується транспортним засобом і знаходиться в ньому, але не причетна до керування ним;
* особа, яка здійснює поїздку у дорожньому транспортному засобі (за винятком водія та іншого персоналу по обслуговуванню пасажирів);
* особа, за винятком членів екіпажу, яка перевозиться на повітряному судні відповідно до договору повітряного перевезення.

Я користуватимусь першим визначенням, тобто пасажир – це особа, яка користується транспортним засобом, але не причетна до керування ним.

Пасажир має широкий діапазон можливостей, які характеризують його стан, але для зручності використання я використовуватиму спрощену модель пасажира, яка в себе включає: можливість купівлі та бронювання квитка, перегляд розкладу та пошук потрібного поїзда, перегляд списку пільг на проїзд та список цін на додаткові послуги у поїзді. Я вдався до абстрагування моделі залізничного вокзалу, описавши фундаментальні та необхідні для його функціонування можливості.

**2.3 Персонал**

Персонал залізничного вокзалу несе відповідальність за правильне функціонування усіх його частин. За допомогою персоналу відбувається взаємодія між пасажирами та самим рухомим складом потягів. Термін «персонал» має багато визначень та поділів, зокрема такі:

* *Адміністративно-технічний* — керівники суб'єктів господарювання, головний енергетик (механік), начальники відділів, дільниць, інженери та техніки, майстри та інші особи, на яких покладено адміністративно-технічні функцій;
* *Оперативний* — персонал, який перебуває на чергуванні в зміні і допущений до оперативного керування і оперативних перемикань: диспетчери, чергові інженери та техніки, начальники змін, чергові на щитах керування, члени оперативно-виїзних бригад, які обслуговують теплові установки та мережі;
* *Оперативно-ремонтний* — ремонтний персонал, спеціально навчений і підготований для оперативного обслуговування в затвердженому обсязі закріпленого за ним устатковання;
* *Ремонтний* — персонал, навчений і допущений до ремонту теплового устатковання і мереж, засобів вимірювальної техніки і автоматики теплового устатковання;
* *Електротехнічний* — спеціально підготовлений адміністративно-технічний, оперативний, оперативно-ремонтний, ремонтний персонал, який здійснює монтаж, наладку, технічне обслуговування, ремонт і керування режимом роботи електроустановок.

Для опису такої великої кількості типів працівників та обслуговуючого персоналу потрібно програмно врахувати чималу кількість властивостей, методів та функцій. Тому я демонструватиму можливості адміністрації вокзалу, оскільки, на мою думку, саме персонал адміністрації вокзалу виконує більшість ключові завдання управління вокзалом. Представники адміністрації можуть додавати нову інформацію про поїзди чи працівників, виплачувати заробітну плату, розраховуватися з постачальниками за паливо та виконувати інші важливі з точки зору їх функціональності у залізничному вокзалі функції.

**2.4 Потяги**

Програмно сутність «потяг» повинна бути максимально наближено передати її роль у реальному світі.

Сьогодні відомо безліч типів поїздів, а саме:

* Поїзд пасажирський — поїзд для перевезення пасажирів, багажу й пошти, сформований з пасажирських вагонів. За видами сполучення пасажирські поїзди розподіляються на далекі, що рухаються на відстань більше 700 км, місцеві до 700 км та приміські до 150 км.
* вантажно-пасажирський — формується на малодіяльних ділянках із вантажних та пасажирських вагонів, які призначені для перевезення вантажів та пасажирів.
* вантажний підвищеної ваги — вантажний поїзд вагою більше 6 тисяч тонн з одним або декількома діючими локомотивами в голові рухомого складу, в голові та хвості, в голові та останній третині рухомого складу.
* поштово-багажний — формується з пасажирських і вантажних вагонів, які призначені для перевезення пошти, багажу й вантажобагажу, а також окремих пасажирських вагонів для перевезення пасажирів, що причіплюються тільки на дільницях, де не обертаються пасажирські поїзди.
* Електропоїзд — моторвагонний рухомий склад, який переміщують бортові електродвигуни, що отримують енергію із контактної мережі або від акумуляторних батарей. Потяги формуються із моторних і причіпних вагонів. Головний і хвостовий вагони електропоїзда мають кабіни керування.
* Рейковий автобус, рідше скорочено рельсобус — синонім або найменування автомотрис та дизель-поїздів малої складової (один-чотири вагони).
* Інтерсіті — денний міжрегіональний швидкісний поїзд, тобто пасажирський поїзд, який рухається з середньою швидкістю 80 км/год та максимальною швидкістю 160 км/год у денний час, вагони з місцями для сидіння 1-го та 2-го класу з можливим включенням вагонів 3-го класу.
* TGV — пасажирські потяги з електричним приводом, найшвидші потяги у світі.
* Поїзд людський — вантажний поїзд, у якому знаходиться 10 і більше вагонів, зайнятих людьми.
* Поїзд пасажирський довгоскладний — пасажирський поїзд, довжина якого перевищує довжину, встановлену схемою формування даного поїзда.
* Поїзд пасажирський довгоскладний — пасажирський поїзд, довжина якого перевищує довжину, встановлену схемою формування даного поїзда.

Оскільки існує ще багато видів поїздів, для опису їх функцій потрібно програмно передбачити незліченну кількість властивостей та методів. У своїй роботі я використав лише деякі з них. А саме: пасажирський, вантажний, електричний, рейковий автобус, TGV та intercity. Я вважаю, що обраних поїздів буде достатньо, щоб описати всі можливості та їхні функції.

**2.5** **Визначення основних завдань покладених в основу програми**

Дане програмне забезпечення створене для вирішення певного роду завдань, які тісно пов’язані з такими об’єктами залізничного вокзалу як пасажири, персонал та потяги. Основною задачею, яку повинна виконувати ця програма є операції над розкладом руху потягів, які докорінно можуть його змінити, забезпечення співробітникам швидкого доступу до інформації про їхню заробітну плату, надання послуг пасажирам, зокрема, можливість переглядати розклад руху потягів, шукати відповідний номер потягу у базі даних та купувати чи бронювати квиток на потрібний їм потяг. У підсумку я отримав набір функцій, які мені потрібно реалізувати для виконання поставленого завдання.

* Запис і збереження даних у файлах;
* Зчитування даних з файлу;
* Додавання або видалення даних;
* Сортування даних для виведення їх екран;
* Пошук потрібних даних по структурованих текстових файлах.

Проаналізувавши мою умову мого завдання опису сутності «Залізничний вокзал», можна виокремити основні об’єкти, відносно яких можна сформувати властивості, які однозначно їх відрізняють:

* Потяги;
* Персонал;
* Квиток;

Оскільки усі потяги мають відмінності між собою, то я виокремив такі, за якими можна однозначно охарактеризувати кожен:

1. Тип поїзда;
2. Номер поїзда;
3. Назва поїзда;
4. Час прибуття;
5. Час відправлення;
6. Номер платформи;
7. Кількість вільних місць;
8. Тип двигуна;
9. Потужність двигуна;
10. Максимальна та мінімальна швидкість;
11. Споживання палива на 10 кілометрів;
12. Ціна за паливо , яку потрібно заплатити.

Працівники також можуть бути охарактеризовані певною кількістю ознак:

1. Тип професії;
2. Ім’я працівника;
3. Прізвище працівника;
4. Вік;
5. Стаж;
6. Дата народження;
7. Премія за додаткові робочі дні;
8. Доплата за стаж;
9. Заробітна плата.

Для квитка також можна виділити набір ознак:

1. Тип поїзда;
2. Назва поїзда;
3. Номер вагона;
4. Місце;

**2.6 Алгоритм виконання роботи програми**

Оскільки, я виокремив чотири основні об’єкти, з якими буде працювати моя програма, то необхідно сформувати перелік операцій, які будуть доступні для виконання певним типом користувачів над цими об’єктами.

Тип користувача – Administrator – має наcтупний список можливостей:

* Додавання нового поїзда до бази даних;
* Перегляд списку співробітників;
* Додавання нового співробітника;
* Пошук інформації про співробітника;
* Друкування квитанції для оплати за паливо;

Тип користувача – User – власне, це тип користувача для якого і розроблялася дана програма. Дані користувачі матимуть наступний список можливостей:

* Перегляд розкладу;
* Пошук потрібного поїзда;
* Придбання чи бронювання квитка з можливістю зниження його вартості для пільгових категорій громадян;
* Перегляд цін на додаткові послуги;
* Перегляд цін на пільговий проїзд;

UML – схема роботи усіх вище вказаних процесів є у додатку В.

**3.** **Реалізація алгоритму у програмному коді**

Після складання алгоритму роботи моєї програми, я розпочав реалізовувати її програмним кодом. У своїй програмі я використовую 23 класи, а саме: TrainProcessing (клас файлових операцій для потягів), Train (клас де прописано ознаки поїзда), WorkerProcessing (клас файлових операцій для працівників), Worker (клас де прописано ознаки працівників), TicketProcessing (клас файлових операцій для придбання квитка), Ticket (клас де прописано ознаки квитка), DistanceProcessing (клас файлових операцій для розрахунку ціни за паливо), Distance (клас де прописано ознаки відстаней для різних маршутів), AdditionalServices (клас файлових операцій для цін на додаткові послуги), Benefits (клас файлових операцій для даних про пільги), Schedule (проміжний клас, за допомогою якого клієнт здійснює купівлю квитка). В класах: CargoTrain, ElectricityTrain, IntercityTrain, RailbusTrain, TGV, PassangerTrain я описав характеристики кожного типу поїзда зокрема. У класах Keeper, Maid, Driver, Conductor я описав характеристики кожного працівника зокрема. Класи User і Administrator слугують для здійснення входу з різними правами доступу до інформації.

Користувач – адміністратор, об’єкт – потяги.

Виконання запису нових даних у файл відбувається таким чином: відкривається файл для запису, поелементно записуються дані про поїзд, файл закривається, а на екран виводиться повідомлення, що дані додано успішно.

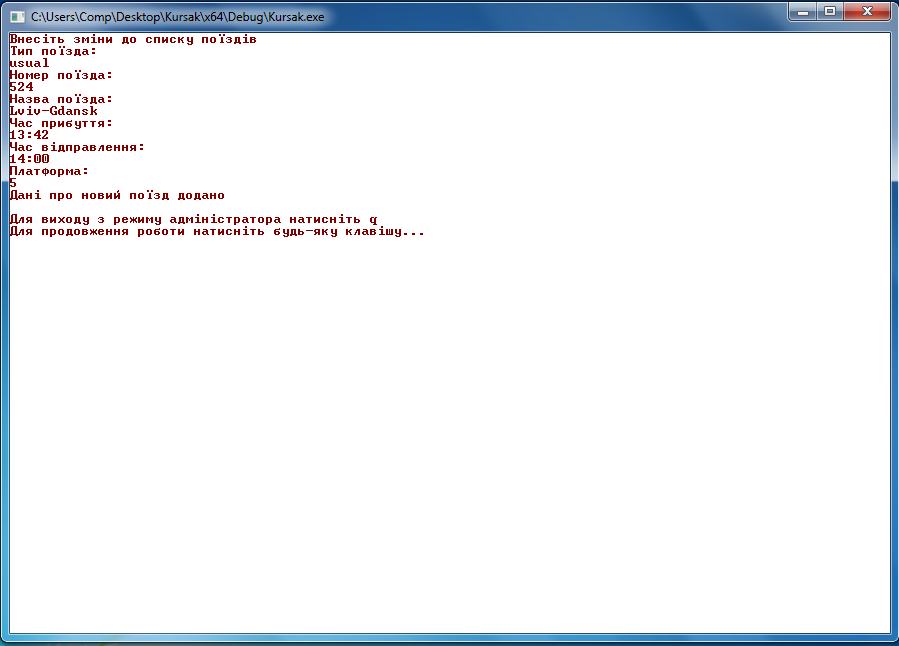


Рис.1 Виконання операції додавання нового потягу до бази даних

Виконання запису нових даних у файл відбувається наступним чином: відкривається файл для запису, поелементно записуються дані про працівника, файл закривається, а на екран виводиться повідомлення, що дані додано успішно.

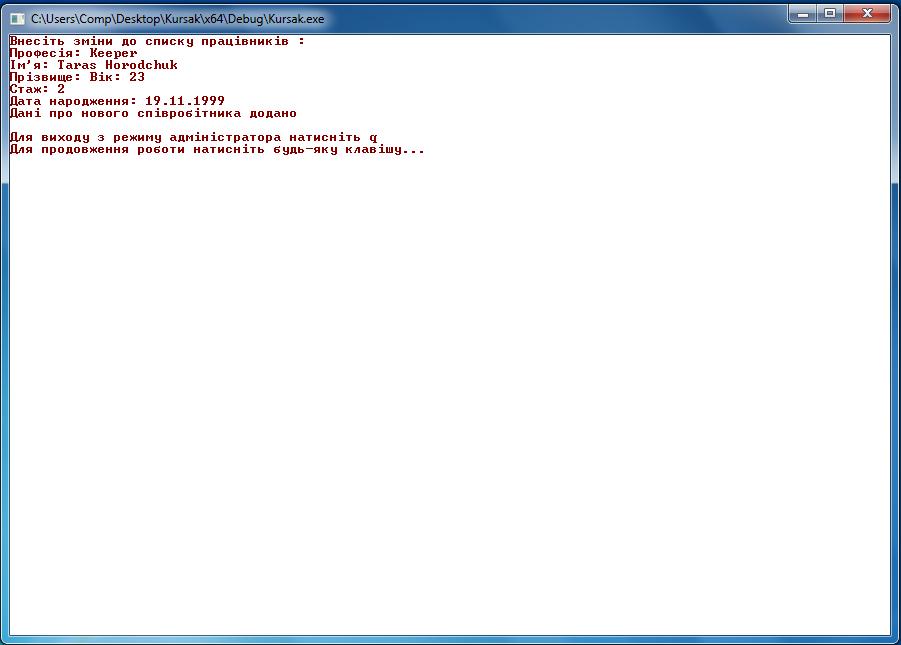


Рис.2 Виконання запису у файл даних про нового працівника

Виконання зчитування з файлу списку працівників відбувається так: файл відкривається для зчитування, проводиться поелементне зчитування, після чого файл закривається, а дані які зчитані з файлу виводяться на екран.

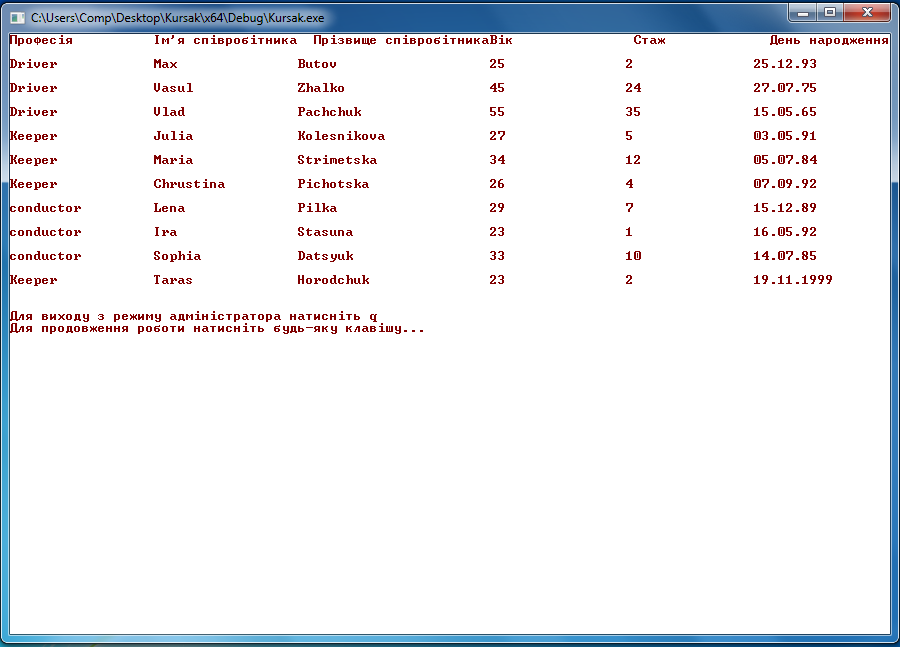
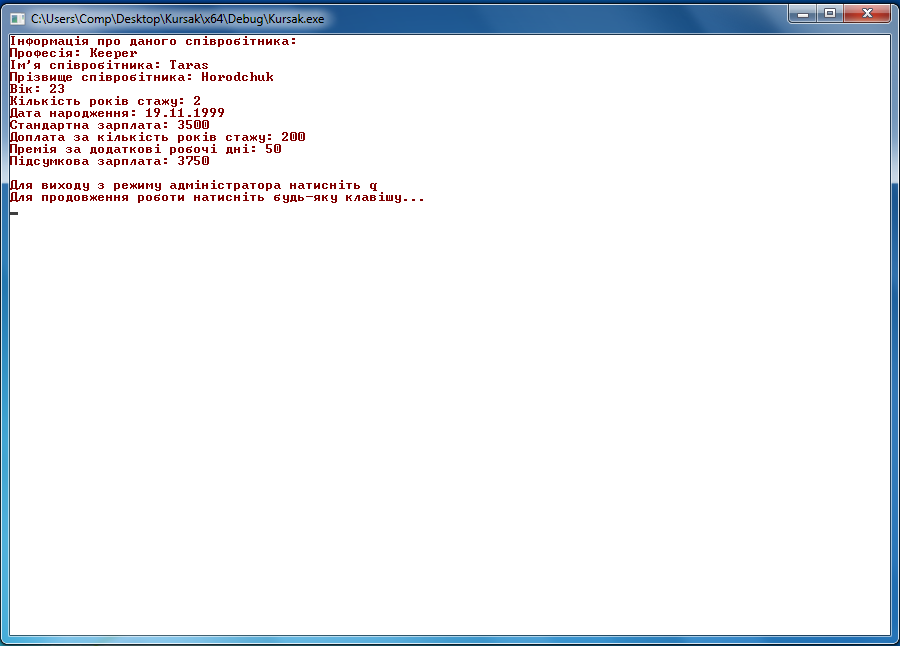


Рис.3 Виконання зчитування з файлу даних про працівників

Пошук вибраного вами працівника, для здійснення виплати йому заробітної плати, здійснюється за допомогою зчитування з клавіатури потрібного імені співробітника. Потім відкривається файл для зчитування зі списком працівників і поелементно зчитуючи з файлу усі дані, ми порівнюємо введене ім’я, з тим, що було зчитано з файлу. При виявленні збігу, перед тим, як інформація про працівника буде виведена на екран, потрібно ввести кількість дадаткових робочих днів які відпрацював даний робітник, для виписання йому премії.



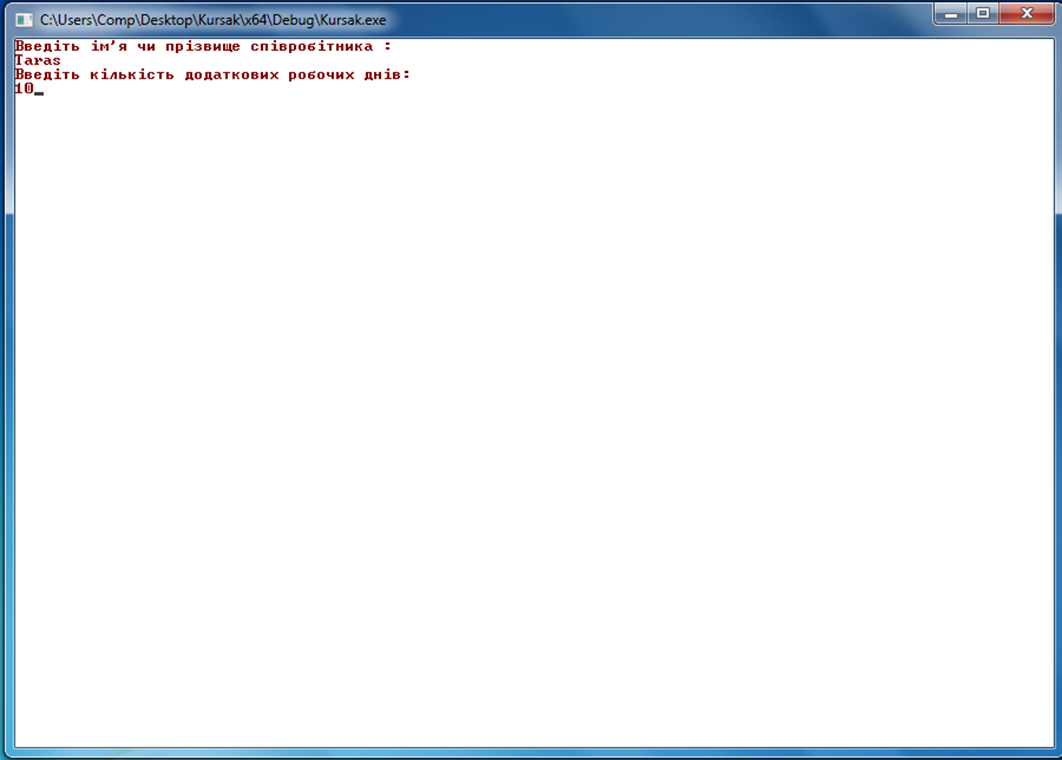


Рис.4,5 Виконання пошуку інформації про працівника для виплати зарплати

Виконання друкування квитанції про оплату пального відбувається в декілька етапів. На першому етапі здійснюється зчитування з клавіатури потрібного номера чи назви поїзда, на який потрібно видрукувати квитанцію. Далі відкривається файл для зчитування з інформацією про відстані і, поелементно зчитуючи з файлу усі дані, ми співставляємо нашу назву чи номер поїзда, з тими, що були зчитані з файлу. При виявленні збігу, потрібно ввести середню швидкість з якою рухатиметься даний поїзд на цьому маршуті. Після введення середньої швидкості програма виконає необхідні обчислення, після чого програма відкриє новий файл куди запише необхідні дані для формування квитанції про оплату.

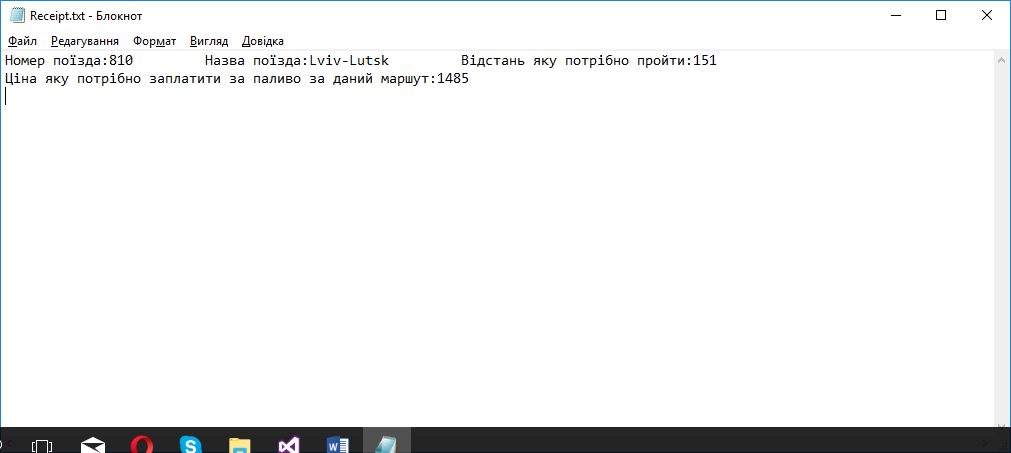


Рис.6 Результат виконання друку квитанції

Виконання зчитування з файлу списку потягів відбувається так: файл відкривається для зчитування, проводиться поелементне зчитування, після чого файл закривається.

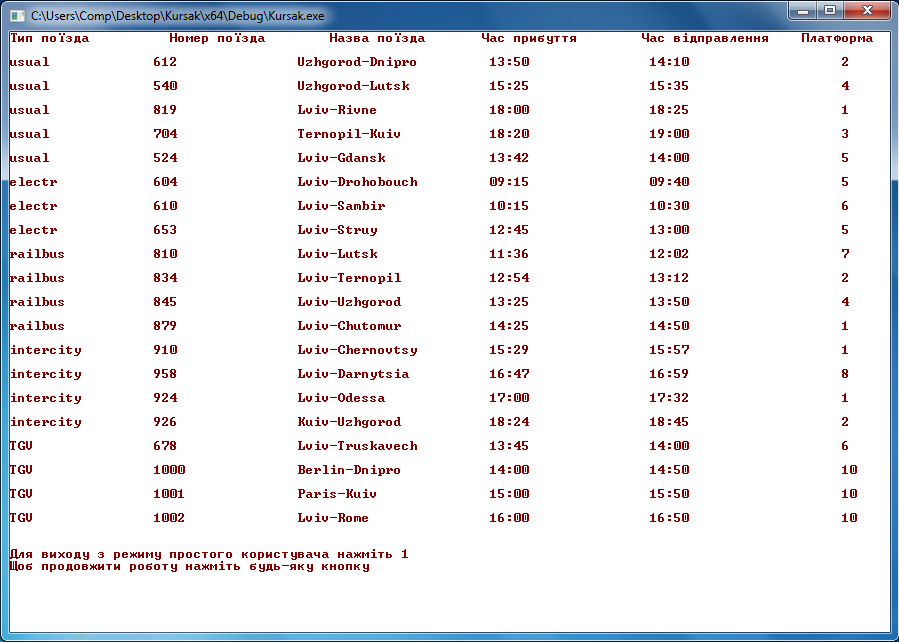


Рис.7 Результат виконання операції, яка виводить розкад на екран

Далі зчитаний список поїздів сортується за типом поїзда, після чого він виводиться на екран.

Пошук вибраного вами поїзда, для знаходження інформації про нього, здійснюється за допомогою зчитування з клавіатури потрібного номера чи назви поїзда. Далі відкривається файл для зчитування зі списком поїздів і поелементно зчитуючи з файлу усі дані, ми порівнюємо наш номер чи назву поїзда, з тими, що були зчитані з файлу. При виявленні збігу інформація про поїзд виводиться на еран. Якщо ж пасажир, з цікавості, бажає переглянути інформацію про технічні характеристики поїзда, він може їх переглянути нажавши відповідну клавішу на клавіатурі. Після цього на екран будуть виведені характеристики даного поїзда.

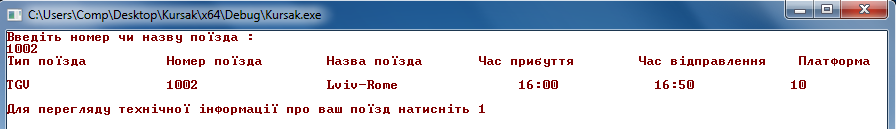


Рис.8 Результат виконання операції, яка виводить дані про поїзд на екран

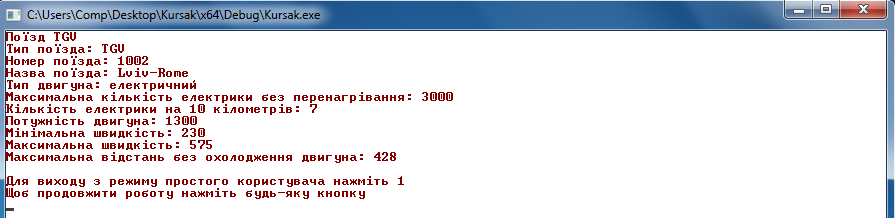


Рис.9 Результат виконання операції, яка виводить технічні дані про поїзд на екран

Купівля квитка відбувається в декілька етапів. На першому етапі відбувається пошук пошук поїзда та інформації про нього. Це здійснюється за допомогою зчитування з клавіатури потрібного номера чи назви поїзда. Потім відкривається файл для зчитування зі списком поїздів і поелементно зчитуючи з файлу усі дані, ми порівнюємо наш номер чи назву поїзда, з тими, що було зчитано з файлу. При вияленні збігу інформація про даний поїзд записується у файл для створення квитка.

На другому етапі відбувається пошук вільних місць. Відкривається файл для зчитування зі списком вільних місць у поїзді і поелементно зчитуючи з файлу усі дані, ми порівнюємо ім’я нашого поїзда з тими, що були зчитані з файлу. При виявлені збігу, інформація про вільні місця виводиться на екран.

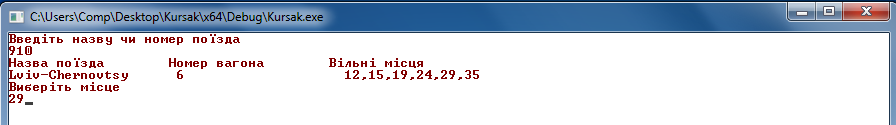
На третьому етапі відбувається вибір місця. Пасажир вводить номер місця яке хоче придбати, після чого цей номер порівнюється з місцями які доступні, якщо ж виявлено збіг інформація про це місце видаляється з довідника, а сам довідник закривається.

Рис.10 Результат виконання операцій за допомогою яких відбувається вибір місця

На четвертому етапі, якщо пасажир має якусь пільгову категорію він може придбати квиток, натиснувши відповідну клавішу і, вибравши пільгову категорію. Якщо ж пасажир немає ніякої пільгової категорії він обирає звичайну форму купівлі квитка.

На п’ятому етапі, пасажир має можливість одразу придбати квиток або ж забронювати його. Різниця між цими можливостями полягає в тому, що якщо пасажир хоче забронювати обране місце йому не доведеться оплачувати одразу ціну за квиток, але у цьому випадку ціна квитка зросте на 10 грн.

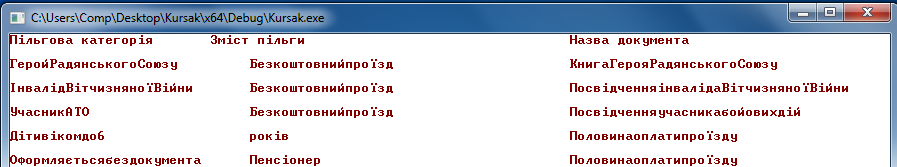


Рис.11 Операції з вибору пасажиром пільгової категорії (якщо вона є) та методу купівлі чи бронювання квитка

Після вибору купівлі чи бронювання квитка, інформація про місце, номер вагона і ціну записується у файл для завершення формування квитка. Після чого цей файл закривається, а на екран виводиться повідомлення, що квиток надруковано.

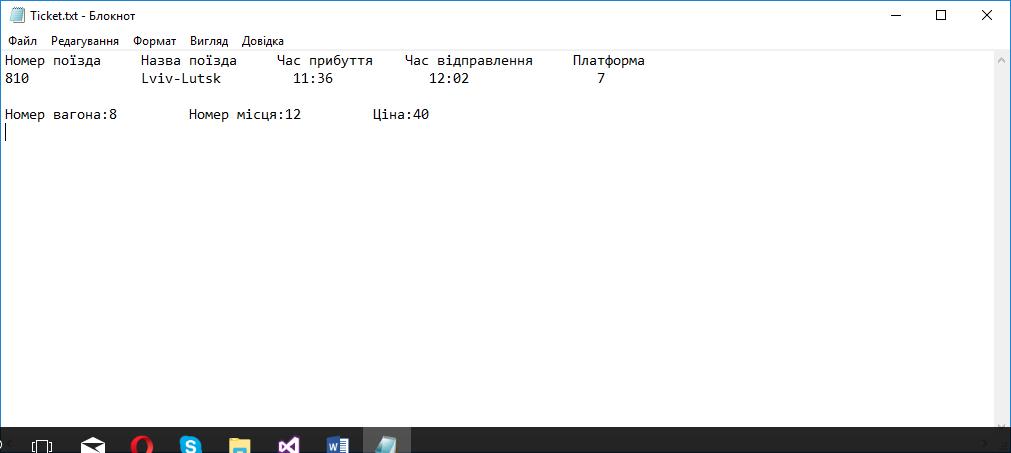


Рис.12 Результат виконання друку квитка

Виконання зчитування з файлу інформації про ціни на додаткові послуги відбувається наступним чином: файл відкривається для зчитування, проводиться поелементне зчитування, після чого файл закривається, а дані, що було зчитано з файлу виводяться на екран.

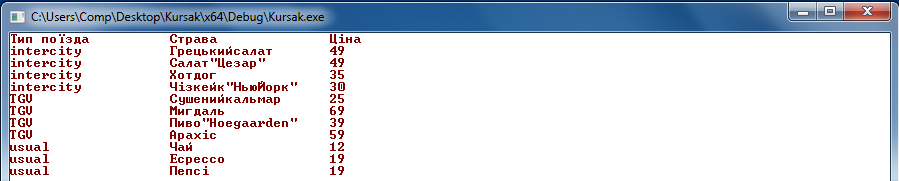


Рис.13 Результат виконання операції ,яка виводить дані про ціни на додаткові послуги на екран

Виконання зчитування з файлу інформації про пільговий проїзд відбувається таким чином: файл відкривається для зчитування, проводиться поелементне зчитування, після чого файл закривається, а дані які зчитані з файлу виводяться на екран.

Рис.14,15 Можливості користувача та адміністратора

Кодова частина виконання всіх цих операцій наведена у додатку А.

**4. Тестування програми**

При вході у систему потрібно визначитися з режимом доступу: адміністратор чи пасажир.

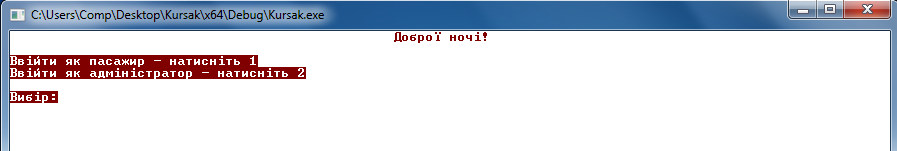


Рис.15 Виконання меню для входу в систему

Спочатку оберемо режим доступу адміністратор.



Рис.16 Виконання входу в систему від імені адміністратора

Після цього ми отримаємо перелік операцій, які можна здійснювати в даному режимі.

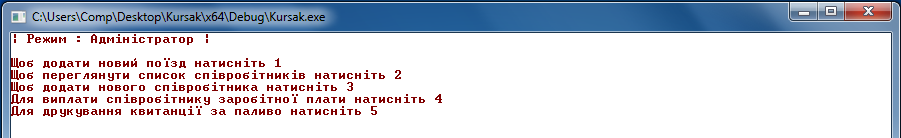


Рис.17 Виконання операцій які може здійснювати адміністратор

Для виконання операції друкування квитанції потрібно ввести пароль вищого доступу.



Рис.18 Виконання входу в систему з паролем вищого доступу

Якщо ж ми оберемо режим доступу типу «Пасажир» ми отримаємо набір операцій, які можна здійснювати в такому режимі.

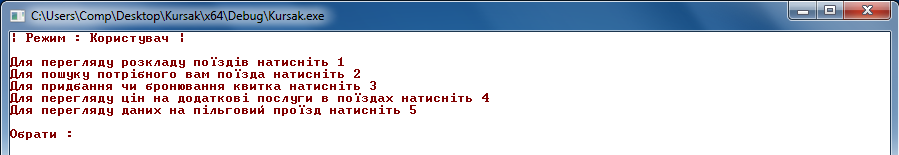
**

Рис.19 Перелік операцій які може здійснювати пасажир

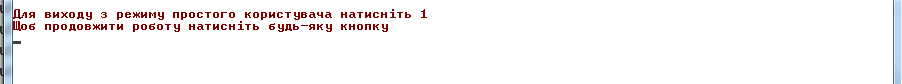
 Зробивши потрібні вам дії, результати яких описані в попередньому розділі, ви можете повернутись до попереднього меню чи вийти з режиму користувача під яким ви зараз працюєте.

Рис.20 Операцій для виходу або продовження роботи в даному режимі

Також передбачена можливість входу від імені іншого користувача чи завершити роботу з програмою.

Код, завдяки якому виконуються дані операції знаходиться у додатку А.

**5. Аналіз продуктивності (часу роботи) реалізованої програми згідно індивідуального завдання.**

Оскільки, моя програма реалізує процеси завдяки яким функціонує залізничний вокзал, то зрозуміло, що великої кількості інформації в обігу програми не буде (максимальна кількість поїздів, яку може прийняти і обслужити вокзал, а також його персоналу є обмежена). Значної різниці в часі виконання програми спостерігатися не буде.

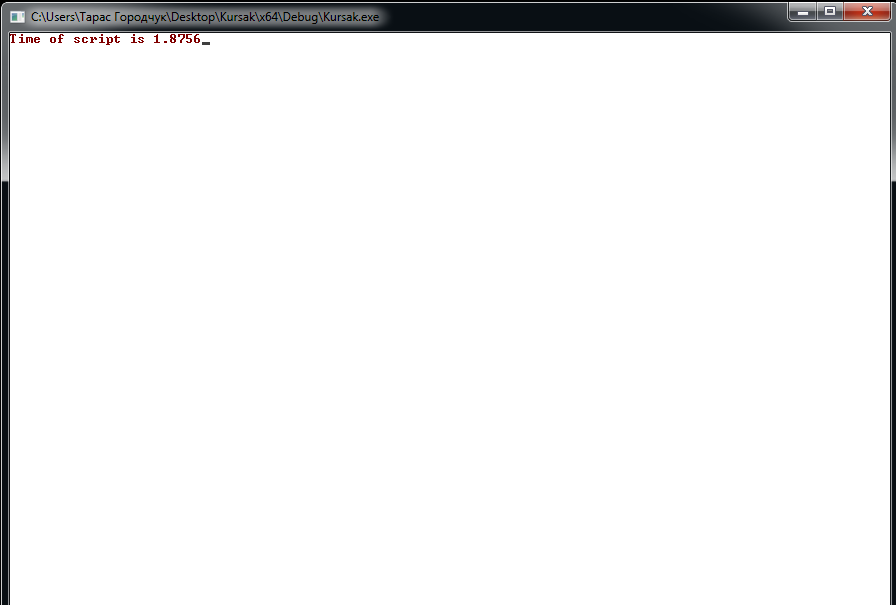


Рис.21 Тестування часу виконання програми

Використавши функцію clock(), я виконав тестування часу роботи скрипта. В результаті досліджень було виявлено, що час роботи програми при збільшенні обсягу виконання операцій відрізняється незначно.

***Висновок***

У процесі виконання поставленого завдання, а саме реалізація такої предметної галузі як залізничний вокзал, я закріпив свої знання з вивченого за один семестр курсу об’єктно-орієнтованого програмування. Закріпив на практиці знання із створення класів, утворення одинарного наслідування та інкапсуляцію даних.

З переваг моєї програми можна виділити те , що вона є безкоштовною, а також має високу швидкодію, наявність простого у використанні інтерфейсу, що спрощує керування нею. До переваг можна віднести й те, що люди можуть купувати квитки та дізнаватися всю потрібну їм інформацію не підходячи до каси.

З точки зору програмування однією із позитивних сторін моєї програми є те, що я використовував масиви об’єктів – так звані вектори, адже у сучасному світі ООП саме такий метод опрацювання об’єктів є пріоритетним.

Недоліком моєї програми можна вважати те, що вона не охоплює усіх аспектів керування залізничним вокзалом.

В результаті виконання даної курсової роботи можна зробити висновок, що станом на сьогодні залізничний вокзал є потужним механізмом, який обслуговує велику кількість пасажирів. Правильне його функціонування є запорукою успішного пасажироперевезення великої кількості людей. Зваживши всі плюси та мінуси можна сказати, що моя програма спрямована для підтримки правильності його функціонування.

**Література**

* Лавріщева К.М. Програмна інженерія—К.: Академперіодика, 2008.- 319 с.
* Основы информатики и программирования: Учебное пособие// Е. А. Роганов — М.: МГИУ, 2001. — Вузол VI.(рос.)
* Armstrong, Deborah J. (February 2006). The Quarks of Object-Oriented Development. Communications of the ACM 49 (2). с. 123–128. ISSN 0001-0782.
* Meyer, Bertrand (1997). Object-Oriented Software Construction. Prentice Hall. ISBN 0-13-629155-4.
* Електронні ресурси

1. <http://programming.in.ua/programming/basisprogramming/25-oop.html>
2. <https://uk.wikipedia.org/wiki/Об%27єктноорієнтоване_програмування#.D0.86.D1.81.D1.82.D0.BE.D1.80.D1.96.D1.8F>
3. <https://uk.wikipedia.org/wiki/C%2B%2B>
4. <https://ru.wikipedia.org/wiki/Объектноориентированное_программирование#.D0.98.D1.81.D1.82.D0.BE.D1.80.D0.B8.D1.8F>
5. <https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_програмування>
6. <http://www.progs.in.ua/Progs/rozdilu/OOP.php>
7. <http://damp.biz/plyusi-i-minusi-obyektno-oriyentovanogo-programuvannya/>
8. <https://uk.wikipedia.org/wiki/Пасажир>
9. <https://uk.wikipedia.org/wiki/Поїзд#.D0.A0.D1.96.D0.B7.D0.BD.D0.BE.D0.B2.D0.B8.D0.B4.D0.B8_.D0.BF.D0.BE.D1.97.D0.B7.D0.B4.D1.96.D0.B2>
10. <https://uk.wikipedia.org/wiki/Персонал>
11. <https://uk.wikipedia.org/wiki/Електропоїзд>
12. <https://uk.wikipedia.org/wiki/Рейковий_автобус>
13. <https://uk.wikipedia.org/wiki/Інтерсіті>
14. <https://uk.wikipedia.org/wiki/TGV>

**ДОДАТОК А**

#pragma once

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

class AdditionalServices

{

private:

string typeTrain;

string nameOfDish;

int price;

public:

void lookPriceForAdditionalService(string name); //Створюємо метод для перегляду iнформацiї про цiни на додатковi послуги в поїздах

};

Лістинг 1. Код файлу «AdditionalServices.h»

#pragma once

#include "DistanceProcessing.h"

#include "TrainProcessing.h"

#include "WorkerProcessing.h"

#include "Driver.h"

#include "Keeper.h"

#include "Conductor.h"

#include "Maid.h"

class Administrator

{

public:

void featureList();

void addTrain();

void lookWorkerList();

void addWorker();

void searchWorker();

void fuelReceipt();

};

Лістинг 2. Код файлу «Administrator.h»

#pragma once

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

class Benefits

{

private:

string listCategories;

string ContentPreferences;

string DocumentTitleFreeTravel;

public:

void lookBenefits(string name); //Створюємо метод для перегляду даних про пiльговий проїзд

};

Лістинг 3. Код файлу «Benefits.h»

#pragma once

#include"Train.h"

class CargoTrain:public Train

{

public:

protected:

int fuelCapacity;

int fuelPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

CargoTrain(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");;

~CargoTrain();

void showInfo();

void showTechDitails();

};

Лістинг 4. Код файлу «CargoTrain.h»

#pragma once

#include "Worker.h"

class Conductor: public Worker {

protected:

int normalSalary;

int surchangeRecord;

int allSalary;

int premium;

public:

Conductor(string typeOfWorker = "", string nameWorker = "", string surname = "", int age = 0, int experience = 0, string birthday = 0, int numberWorkingDays = 0);

~Conductor();

void showInfo();

void showAllInform();

};

Лістинг 5. Код файлу «Conductor.h»

#pragma once

#include<fstream>

#include<iomanip>

#include <vector>

#include <string>

#include <iostream>

using namespace std;

class Distance

{

private:

string typeTrain;

int numberTrain;

string nameTrain;

int length;

public:

//конструктор за замовчуванням

Distance(string typeTrain, int numberTrain, string nameTrain, int length) {

this->typeTrain = typeTrain;

this->numberTrain = numberTrain;

this->nameTrain = nameTrain;

this->length = length;

}

~Distance();

//Створюємо методи get для кожної властивостi

string getTypeTrain();

int getNumberTrain();

string getNameTrain();

int getlength();

};

Лістинг 6. Код файлу «Distance.h»

#pragma once

#include"Distance.h"

class DistanceProcessing

{

private:

int speed;

float fuel10Km;

float amountFuel;

float electr10Km;

float amountElectr;

float priceFuel;

float priceElectr;

int costPath;

public:

string \*getStrArray(string name, int numerator);

Distance\*\* createDistanceList(string\* lineArray, int numerator); //Створюємо масив з класу Distance

int getNumberLine(string);

Distance\*\* findDistance(Distance \*\* array, string property, int numerator, int trainNumerator);

int getNumeratorOfSearch(Distance \*\* array, string property, int numerator);

string convertToString(int numberTrain, string nameTrain, int length);

float fuelInformation(string type, int leght);

string convertToString(int costPath);

};

Лістинг 7. Код файлу «DistanceProcessing.h»

#pragma once

#include "Worker.h"

class Driver: public Worker {

protected:

int normalSalary;

int surchangeRecord;

int allSalary;

int premium;

public:

Driver(string typeOfWorker = "", string nameWorker = "", string surname = "", int age = 0, int experience = 0, string birthday = 0, int numberWorkingDays = 0);

~Driver();

void showInfo();

void showAllInform();

};

Лістинг 8. Код файлу «Driver.h»

#pragma once

#include "Train.h"

class ElectricityTrain : public Train {

protected:

int maxProcessingElectricity;

int electPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

ElectricityTrain(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");

~ElectricityTrain();

void showInfo();

void showTechDitails();

};

Лістинг 9. Код файлу «ElectricityTrain.h»

#pragma once

#include "Train.h"

class IntercityTrain:public Train

{

protected:

int fuelCapacity;

int fuelPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

IntercityTrain(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");;

~IntercityTrain();

void showInfo();

void showTechDitails();

};

Лістинг 10. Код файлу «IntercityTrain.h»

#pragma once

#include "Worker.h"

class Keeper : public Worker {

protected:

int normalSalary;

int surchangeRecord;

int allSalary;

int premium;

public:

Keeper(string typeOfWorker = "", string nameWorker = "", string surname = "", int age = 0, int experience = 0, string birthday = 0, int numberWorkingDays = 0);

~Keeper();

void showInfo();

void showAllInform();

};

Лістинг 11. Код файлу «Keeper.h»

#pragma once

#include "Worker.h"

class Keeper : public Worker {

protected:

int normalSalary;

int surchangeRecord;

int allSalary;

int premium;

public:

Keeper(string typeOfWorker = "", string nameWorker = "", string surname = "", int age = 0, int experience = 0, string birthday = 0, int numberWorkingDays = 0);

~Keeper();

void showInfo();

void showAllInform();

};

Лістинг 12. Код файлу «Keeper.h»

#pragma once

#include "Worker.h"

class Maid : public Worker {

protected:

int normalSalary;

int surchangeRecord;

int allSalary;

int premium;

public:

Maid(string typeOfWorker = "", string nameWorker = "", string surname = "", int age = 0, int experience = 0, string birthday = 0, int numberWorkingDays = 0);

~Maid();

void showInfo();

void showAllInform();

};

Лістинг 13. Код файлу «Maid.h»

#pragma once

#include "Train.h"

class PassengerTrain : public Train {

protected:

int fuelCapacity;

int fuelPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

PassengerTrain(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");;

~PassengerTrain();

void showInfo();

void showTechDitails();

};

Лістинг 14. Код файлу «PassengerTrain.h»

#pragma once

#include "Train.h"

class RailbusTrain: public Train {

protected:

int fuelCapacity;

int fuelPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

RailbusTrain(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");;

~RailbusTrain();

void showInfo();

void showTechDitails();

};

Лістинг 15. Код файлу «RailbusTrain.h»

#pragma once

#include"TrainProcessing.h"

class Schedule

{

public:

void dataOnScreen(Train\*\* array, int numerator, int i);

void lookSchedule(Train\*\* array, int numerator);

};

Лістинг 16. Код файлу «Schedule.h»

#pragma once

#include"Train.h"

class TGV:public Train

{

protected:

int maxProcessingElectricity;

int electPerKm;

int enginePower;

int maxSpeed;

int minSpeed;

public:

TGV(string typeOfTrain = "", int numberTrain = 0, string nameTrain = "");;

~TGV();

void showInfo();

void showTechDitails();

};

Лістинг 17. Код файлу «TGV.h»

#pragma once

#include<fstream>

#include<iomanip>

#include <vector>

#include <string>

#include <iostream>

using namespace std;

class Ticket

{

private:

string nameTrain;

int numberCar;

string place;

int price;

public:

Ticket(string nameTrain, int numberCar, string place, int price) {

this->nameTrain = nameTrain;

this->numberCar = numberCar;

this->place = place;

this->price = price;

}

~Ticket();

string getNameTrain();

int getNumberCar();

void setPlace(string place);

string getPlace();

int getPrice();

};

Лістинг 18. Код файлу «Ticket.h»

#pragma once

#include"Ticket.h"

class TicketProcessing

{

private:

string place;

int old;

string numeric;

string number;

public:

string \*getStrArray(string name, int numerator); //Створюємо метод який повертає вказiвник на масив рядкIв

Ticket\*\* createTicketList(string\* lineArray, int numerator); //Створюємо масив з класу WorkerTable

int getNumberLine(string); //Створюємо метод для пiдрахування кiлькостi рядкiв

Ticket\*\* findFreePlace(Ticket \*\* array, string property, int numerator, int trainNumerator);

int getNumeratorOfSearch(Ticket \*\* array, string property, int numerator);

void BuyTicket(Ticket\*\* array, int numerator, int price); //Створюємо метод для купiвлi квитка

string choicePlace(Ticket\*\* ticket, string place);

string convertToString(Ticket ticket);

int privilegedPosition(int price);

string bookingBuying(int numberar, string place, int price);

int deletePlace(string nameFile, string nameTrain, string place);

};

Лістинг 19. Код файлу «TicketProcessing.h»

#pragma once

#include<fstream>

#include<iomanip>

#include <vector>

#include <string>

#include <iostream>

using namespace std;

class Train

{

protected:

string typeOfTrain;

int numberTrain;

string nameTrain;

int maxDistance;

private:

string arrivalTime;

string departureTime;

int platform;

public:

Train(string typeOfTrain, int numberTrain, string nameTrain, string arrivalTime, string departureTime, int platform) {

this->typeOfTrain = typeOfTrain;

this->numberTrain = numberTrain;

this->nameTrain = nameTrain;

this->arrivalTime = arrivalTime;

this->departureTime = departureTime;

this->platform = platform;

}

Train(string typeOfTrain, int numberTrain, string nameTrain) {

this->typeOfTrain = typeOfTrain;

this->numberTrain = numberTrain;

this->nameTrain = nameTrain;

}

~Train();

void showInfo();

//Створюємо методи set i get для кожної властивостi

void setTypeOfTrain(string typeOfTrain);

string getTypeOfTrain();

void setNumberTrain(int numberTrain);

int getNumberTrain();

void setNameTrain(string nameTrain);

string getNameTrain();

void setArrivalTime(string arrivalTime);

string getArrivalTime();

void setDepartureTime(string departureTime);

string getDepartureTime();

void setPlatform(int platform);

int getPlatform();

};

Лістинг 20. Код файлу «Train.h»

#pragma once

#include"Train.h"

class TrainProcessing

{

public:

string \*getStrArray(string name, int numerator); //Створюємо метод який повертає вказiвник на масив рядкIв

Train\*\* createTrainList(string\* lineArray, int numerator); //Створюємо масив з класу WorkerTable

int getNumberLine(string); //Створюємо метод для пiдрахування кiлькостi рядкiв

void lookTrain(Train\*\* array, int numerator); //Сторюємо метод для перегляду списку поїздiв

Train\*\* findTrain(Train \*\* array, string property, int numerator, int trainNumerator);

int getNumeratorOfSearch(Train \*\* array, string property, int numerator);

void setNewData(string nameFile, string newData); //Створюємо метод для записування нових даних у довiдник

string convertToString(string typeOfTrain, int numberTrain, string nameTrain, string arrivalTime, string departureTime, int platform); //Створюємо метод для конвертування даних в тип string

string convertToStringForTicket(int numberTrain, string nameTrain, string arrivalTime, string departureTime, int platform); //Створюємо метод для конвертування даних в тип string якi будуть записуватися у вiдомостi квитка

};

Лістинг 21. Код файлу «TrainProcessing.h»

#pragma once

#include "Benefits.h"

#include "AdditionalServices.h"

#include "Schedule.h"

#include "TicketProcessing.h"

#include "TrainProcessing.h"

#include "ElectricityTrain.h"

#include "PassengerTrain.h"

#include "RailbusTrain.h"

#include "IntercityTrain.h"

#include "TGV.h"

#include "CargoTrain.h"

class User

{

public:

void featureList();

void lookSchedule();

void searchTrain();

void buyTicket();

void lookPriceAdditionalService();

void lookListBenefits();

};

Лістинг 22. Код файлу «User.h»

#pragma once

#include<fstream>

#include<iomanip>

#include <vector>

#include <string>

#include <iostream>

using namespace std;

class Worker

{

protected:

string typeOfWorker;

string nameWorker;

string surname;

int age;

int experience;

string birthday;

int numberWorkingDays;

public:

//Конструктор за замовчуванням

Worker(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday) {

this->typeOfWorker = typeOfWorker;

this->nameWorker = nameWorker;

this->surname = surname;

this->age = age;

this->experience = experience;

this->birthday = birthday;

}

Worker(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday, int numberWorkingDays) {

this->typeOfWorker = typeOfWorker;

this->nameWorker = nameWorker;

this->surname = surname;

this->age = age;

this->experience = experience;

this->birthday = birthday;

this->numberWorkingDays = numberWorkingDays;

}

~Worker();

void showInfo();

//Прописуємо методи set i get для кожної властивостi

void setTypeOfWorker(string typeOfWorker);

string getTypeOfWorker();

void setNameWorker(string nameWorker);

string getNameWorker();

void setSurname(string surname);

string getSurname();

void setAge(int age);

int getAge();

void setExperience(int experience);

int getExperience();

void setBirthday(string birthday);

string getBirthday();

};

Лістинг 23. Код файлу «Worker.h»

#pragma once

#include"Worker.h"

class WorkerProcessing

{

public:

string \*getStrArray(string name, int numerator); //Створюємо метод який повертає вказiвник на масив рядкIв

Worker\*\* createWorkerList(string\* lineArray, int numerator); //Створюємо масив з класу WorkerTable

int getNumberLine(string); //Створюємо метод для пiдрахунку кiлькостi рядкiв

void lookWorker(Worker\*\* array, int numerator); //Створюємо метод для перегляду списку робочих

Worker\*\* findWorker(Worker \*\* array, string property, int numerator, int trainNumerator);

int getNumeratorOfSearch(Worker \*\* array, string property, int numerator);

void setNewData(string nameFile, string newData); //Створюємо метод для запису нових даних у довiдник

string convertToString(string typeOfWorker, string name, string surname, int age, int experience, string platform); //Створюємо метод для конвертування даних у тип string

};

Лістинг 24. Код файлу «WorkerProcessing.h»

#include "AdditionalServices.h"

void AdditionalServices::lookPriceForAdditionalService(string name) { //Прописуємо метод для пергляду цiн на додатковi послуги в поїздах

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл!" << endl;

}

cout << setw(20) << "Тип поїзда" << setw(20) << "Страва" << setw(20) << "Цiна" << endl;

while (!infile.eof())

{

infile >> typeTrain;

infile >> nameOfDish;

infile >> price;

if (!infile.eof())

{

//Виводимо данi на екран

cout.width(20);

cout << typeTrain;

cout.width(20);

cout << nameOfDish;

cout.width(20);

cout << price << endl;

}

}

}

Лістинг 25. Код файлу «AdditionalServices.cpp»

#include "Administrator.h"

WorkerProcessing dataWorker;

DistanceProcessing dataDistance;

void Administrator::addTrain() {

TrainProcessing dataTrain;

string typeOfTrain;

int numberTrain;

string nameTrain;

string arrivalTime;

string departureTime;

int platform;

string buff;

cout << "Внесiть змiни до списку поїздiв\n";

cout << "Тип поїзда:\n";

cin >> typeOfTrain;

cout << "Номер поїзда:\n";

cin >> numberTrain;

cout << "Назва поїзда:\n";

cin >> nameTrain;

cout << "Час прибуття:\n";

cin >> arrivalTime;

cout << "Час вiдправлення:\n";

cin >> departureTime;

cout << "Платформа:\n";

cin >> platform;

buff = dataTrain.convertToString(typeOfTrain, numberTrain, nameTrain, arrivalTime, departureTime, platform);

dataTrain.setNewData("DovidnukTrain.txt", buff);

cout << "Данi про новий поїзд додано" << endl;

cout << endl;

}

void Administrator::lookWorkerList() {

int numerator = dataWorker.getNumberLine("DovidnukWorker.txt");// Пiдрахування кiлькостi рядкiв

string\* workerArray = dataWorker.getStrArray("DovidnukWorker.txt", numerator);

Worker\*\* pointerWorkerArray = dataWorker.createWorkerList(workerArray, numerator);

dataWorker.lookWorker(pointerWorkerArray, numerator);

cout << endl;

}

void Administrator::addWorker() {

string typeOfWorker;

string nameWorker;

string surname;

int age;

int experience;

string birthday;

string buff;

cout << "Внесiть змiни до списку працiвникiв : \n";

cout << "Професiя: ";

cin >> typeOfWorker;

cout << "Iм'я: ";

cin >> nameWorker;

cout << "Прiзвище: ";

cin >> surname;

cout << "Вiк: ";

cin >> age;

cout << "Стаж: ";

cin >> experience;

cout << "Дата народження: ";

cin >> birthday;

buff = dataWorker.convertToString(typeOfWorker, nameWorker, surname, age, experience, birthday); //Виклик методу, який перетворює типи введених даних в данi типу string

dataWorker.setNewData("DovidnukWorker.txt", buff);

cout << "Данi про нового спiвробiтника додано" << endl;

cout << endl;

}

void Administrator::searchWorker() {

string buff;

int number;

cout << "Введiть iм'я чи прiзвище спiвробiтника : " << endl;

cin >> buff;

int numerator = dataWorker.getNumberLine("DovidnukWorker.txt");

string\* workerArray = dataWorker.getStrArray("DovidnukWorker.txt", numerator);

Worker\*\* pointerWorkerArray = dataWorker.createWorkerList(workerArray, numerator);

int workerNumerator = dataWorker.getNumeratorOfSearch(pointerWorkerArray, buff, numerator);

Worker\*\* pointerStringArray = dataWorker.findWorker(pointerWorkerArray, buff, numerator, workerNumerator);

//Виводимо данi пошуку на екран

if (workerNumerator > 0) {

cout << "Введiть кiлькiсть додаткових робочих днiв: " << endl;

cin >> number;

system("cls");

if (pointerStringArray[0]->getTypeOfWorker() == "Driver") {

Driver dataDriver(pointerStringArray[0]->getTypeOfWorker(), pointerStringArray[0]->getNameWorker(), pointerStringArray[0]->getSurname(), pointerStringArray[0]->getAge(), pointerStringArray[0]->getExperience(), pointerStringArray[0]->getBirthday(), number);

dataDriver.showInfo();

dataDriver.showAllInform();

}

if (pointerStringArray[0]->getTypeOfWorker() == "Keeper") {

Keeper dataKeeper(pointerStringArray[0]->getTypeOfWorker(), pointerStringArray[0]->getNameWorker(), pointerStringArray[0]->getSurname(), pointerStringArray[0]->getAge(), pointerStringArray[0]->getExperience(), pointerStringArray[0]->getBirthday(), number);

dataKeeper.showInfo();

dataKeeper.showAllInform();

}

if (pointerStringArray[0]->getTypeOfWorker() == "conductor") {

Conductor dataConductor(pointerStringArray[0]->getTypeOfWorker(), pointerStringArray[0]->getNameWorker(), pointerStringArray[0]->getSurname(), pointerStringArray[0]->getAge(), pointerStringArray[0]->getExperience(), pointerStringArray[0]->getBirthday(), number);

dataConductor.showInfo();

dataConductor.showAllInform();

}

if (pointerStringArray[0]->getTypeOfWorker() == "maid") {

Maid dataMaid(pointerStringArray[0]->getTypeOfWorker(), pointerStringArray[0]->getNameWorker(), pointerStringArray[0]->getSurname(), pointerStringArray[0]->getAge(), pointerStringArray[0]->getExperience(), pointerStringArray[0]->getBirthday(), number);

dataMaid.showInfo();

dataMaid.showAllInform();

}

}

else

cout << "Даного спiвробiтника не знайдено." << endl;

cout << endl;

}

void Administrator::fuelReceipt() {

TrainProcessing dataTrain;

string buff;

int buff2;

cout << "Для отримання квитанцiї за паливо введiть код доступу :" << endl;

cin >> buff;

system("cls");

if (buff == "admin") {

cout << "Введiть назву чи номер поїзда: " << endl;

cin >> buff;

system("cls");

int numerator = dataDistance.getNumberLine("DovidnukDistanse.txt");

string \*distanceArray = dataDistance.getStrArray("DovidnukDistanse.txt", numerator);

Distance\*\* pointerDistanceArray = dataDistance.createDistanceList(distanceArray, numerator);

int distanceNumerator = dataDistance.getNumeratorOfSearch(pointerDistanceArray, buff, numerator);

Distance\*\* DistanceStringArray = dataDistance.findDistance(pointerDistanceArray, buff, numerator, distanceNumerator);

if (distanceNumerator > 0) {

buff = dataDistance.convertToString(DistanceStringArray[0]->getNumberTrain(), DistanceStringArray[0]->getNameTrain(), DistanceStringArray[0]->getlength());

dataTrain.setNewData("Receipt.txt", buff);

buff2 = dataDistance.fuelInformation(DistanceStringArray[0]->getTypeTrain(), DistanceStringArray[0]->getlength());

buff = dataDistance.convertToString(buff2);

dataTrain.setNewData("Receipt.txt", buff);

cout << endl;

cout << endl;

}

else

cout << "За вашим запитом поїздiв не знайдено" << endl;

}

else

cout << "Ви ввели невiрний пароль!" << endl;

}

void Administrator::featureList()

{

Administrator dataAdministrator;

int number;

string numeric;

while (true)

{

cout << "| Режим : Адмiнiстратор |\n\n";

cout << "Щоб додати новий поїзд натиснiть 1\n";

cout << "Щоб переглянути список спiвробiтникiв натиснiть 2\n";

cout << "Щоб додати нового спiвробiтника натиснiть 3\n";

cout << "Для виплати спiвробiтнику заробiтної плати натиснiть 4\n";

cout << "Для друкування квитанцiї за паливо натиснiть 5\n";

cin >> numeric;

system("cls");

try {

number = stoi(numeric);

}

catch (exception e) {

cout << "Некоректнi данi!" << endl;

number = 6;

}

switch (number)

{

case 1:

dataAdministrator.addTrain();

break;

case 2:

dataAdministrator.lookWorkerList();

break;

case 3:

dataAdministrator.addWorker();

break;

case 4:

dataAdministrator.searchWorker();

break;

case 5:

dataAdministrator.fuelReceipt();

break;

}

cout << "Для виходу з режиму адмiнiстратора натиснiть q" << endl;

cout << "Для продовження роботи натиснiть будь-яку клавiшу..." << endl;

cin >> numeric;

system("cls");

if (numeric == "q")

break;

}

}

Лістинг 26. Код файлу «Administrator.cpp»

#include "Benefits.h"

void Benefits::lookBenefits(string name) { //Прописуємо метод для перегляду iнформацiї про пiльговий проїзд

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

cout << setw(25) << "Пiльгова категорiя" << setw(45) << "Змiст пiльги" << setw(50) << "Назва документа" << setw(15) << endl;

while (!infile.eof())

{

//Зчитуємо данi з файлу

infile >> listCategories;

infile >> ContentPreferences;

infile >> DocumentTitleFreeTravel;

if (!infile.eof())

{

//Виводимо данi на екран

cout.width(30);

cout << listCategories;

cout.width(40);

cout << ContentPreferences;

cout.width(45);

cout << DocumentTitleFreeTravel;

cout.width(15);

cout << endl;

}

}

}

Лістинг 27. Код файлу «Benefits.cpp»

#include "CargoTrain.h"

#include <iostream>

CargoTrain::CargoTrain(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int fuelCapacity = this->fuelCapacity = 2000;

int fuelPerKm = this->fuelPerKm = 6;

int enginePower = this->enginePower = 1500;

int maxSpeed = this->maxSpeed = 80;

int minSpeed = this->minSpeed = 5;

maxDistance = fuelCapacity / fuelPerKm;

}

CargoTrain::~CargoTrain()

{

}

void CargoTrain::showInfo() {

cout << "Вантажний поїзд" << endl;

this->Train::showInfo();

}

void CargoTrain::showTechDitails() {

cout << "Тип двигуна: дизельний" << endl;

cout << "Об'єм баку для палива: " << this->fuelCapacity << endl;

cout << "Витрата палива на 10 кiлометрiв: " << this->fuelPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна дистанцiя: " << this->maxDistance << endl;

}

Лістинг 28. Код файлу «CargoTrain.cpp»

#include "Conductor.h"

#include <iostream>

Conductor::Conductor(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday, int numberWorkingDays) : Worker(typeOfWorker, nameWorker, surname, age, experience, birthday, numberWorkingDays) {

this->numberWorkingDays = numberWorkingDays;

int normalSalary = this->normalSalary = 3500;

int surchangeRecord = this->surchangeRecord = 100 \* experience;

int premium = this->premium = 10 \* numberWorkingDays;

int allSalary = this->allSalary = normalSalary + surchangeRecord + premium;

}

Conductor::~Conductor()

{

}

void Conductor::showInfo() {

cout << "Iнформацiя про даного спiвробiтника" << endl;

this->Worker::showInfo();

}

void Conductor::showAllInform() {

cout << "Стандартна зарплата: " << this->normalSalary << endl;

cout << "Доплата за кiлькiсть рокiв стажу: " << this->surchangeRecord << endl;

cout << "Премiя за додатковi робочi днi: " << this->premium << endl;

cout << "Пiдсумкова зарплата: " << this->allSalary << endl;

}

Лістинг 29. Код файлу «Conductor.cpp»

#include "Distance.h"

Distance::~Distance() {

}

string Distance::getTypeTrain() {

return this->typeTrain;

}

int Distance::getNumberTrain() {

return this->numberTrain;

}

string Distance::getNameTrain() {

return this->nameTrain;

}

int Distance::getlength() {

return this->length;

}

Лістинг 30. Код файлу «Distance.cpp»

#include "DistanceProcessing.h"

Distance\*\* DistanceProcessing::createDistanceList(string \* lineArray, int numerator) {

Distance \*\* infoArray = new Distance\*[numerator];

for (int i = 0; i < numerator; i++)

{

//Парсинг рядка

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = lineArray[i].find\_first\_of(';', pos)) != string::npos) {

localArray.push\_back(lineArray[i].substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(lineArray[i].substr(pos));

infoArray[i] = new Distance(localArray[0], atoi(localArray[1].c\_str()), localArray[2], atoi(localArray[3].c\_str()));

}

return infoArray;

}

int DistanceProcessing::getNumberLine(string nameFile) {

int numerator = 0;

char \*str = new char[1024];

ifstream infile;

infile.open(nameFile);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

// Пiдраховуємо кiлькiсть рядкIв

while (!infile.eof())

{

infile.getline(str, 1024, '\n');

numerator++;

}

numerator--;

return numerator;

infile.close();

}

string\* DistanceProcessing::getStrArray(string name, int numerator) { // Метод, що створює масив рядкIв з рядкiв файлу та повертає вказiвник

string \*lineArray = new string[numerator];

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

for (int i = 0; i < numerator; i++)

{

getline(infile, lineArray[i]);

}

return lineArray;

infile.close();

}

Distance\*\* DistanceProcessing::findDistance(Distance \*\* array, string property, int numerator, int trainNumerator) {

if (trainNumerator > 0) {

Distance\*\* newList = new Distance\*[trainNumerator];

int j = 0;

for (int i = 0; i < numerator; i++) {

if (to\_string(array[i]->getNumberTrain()) == property) {

newList[j] = array[i];

j++;

}

else if (array[i]->getNameTrain() == property) {

newList[j] = array[i];

j++;

}

}

return newList;

}

}

int DistanceProcessing::getNumeratorOfSearch(Distance \*\* array, string property, int numerator) {

int j = 0;

for (int i = 0; i < numerator; i++) {

if (to\_string(array[i]->getNumberTrain()) == property) {

j++;

}

else if (array[i]->getNameTrain() == property) {

j++;

}

}

return j;

}

string DistanceProcessing::convertToString(int numberTrain, string nameTrain, int length) { //Метод для конвертування даних в тип string

string sNumberTrain = to\_string(numberTrain);

string sNameTrain = nameTrain;

string slength = to\_string(length);

string buff = "Номер поїзда:" + sNumberTrain + " Назва поїзда:" + sNameTrain + " Вiдстань, яку потрiбно подолати:" + slength;

return buff;

}

float DistanceProcessing::fuelInformation(string type, int length) {

cout << "Середня швидкiсть даного поїзда : " << endl;

cin >> speed;

system("cls");

if (type == "usual") {

if (speed >= 30 && speed <= 80) {

fuel10Km = speed / 4 \* 0.2;

amountFuel = length / 10 \* fuel10Km;

costPath = priceFuel\*amountFuel;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

if (type == "electr") {

if (speed >= 10 && speed <= 50) {

electr10Km = speed / 4 \* 0.1;

amountElectr = length / 10 \* electr10Km;

costPath = priceFuel\*amountElectr;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

if (type == "railbus") {

if (speed >= 40 && speed <= 100) {

fuel10Km = speed / 4 \* 0.3;

amountFuel = length / 10 \* fuel10Km;

costPath = priceFuel\*amountFuel;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

if (type == "intercity") {

if (speed >= 70 && speed <= 140) {

fuel10Km = speed / 5 \* 0.4;

amountFuel = length / 10 \* fuel10Km;

costPath = priceFuel\*amountFuel;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

if (type == "TGV") {

if (speed >= 150 && speed <= 320) {

electr10Km = speed / 5 \* 0.4;

amountElectr = length / 10 \* electr10Km;

costPath = priceFuel\*amountElectr;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

if (type == "CargoTrain") {

if (speed >= 5 && speed <= 80) {

fuel10Km = speed / 4 \* 0.4;

amountFuel = length / 10 \* fuel10Km;

costPath = priceFuel\*amountFuel;

cout << "Квитанцiю про цiну за паливо роздруковано!";

}

else

cout << "Даний тип поїзда не може рухатися Iз даною швидкiстю!";

}

return costPath;

}

string DistanceProcessing::convertToString(int costPath) {

string sCostPath = to\_string(costPath);

string buff = "Цiна за паливо за даним маршрутом:" + sCostPath;

return buff;

}

Лістинг 31. Код файлу «DistanceProcessing.cpp»

#include "Driver.h"

#include <iostream>

Driver::Driver(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday, int numberWorkingDays) : Worker(typeOfWorker, nameWorker, surname, age, experience, birthday, numberWorkingDays) {

this->numberWorkingDays = numberWorkingDays;

int normalSalary = this->normalSalary = 3500;

int surchangeRecord = this->surchangeRecord = 100 \* experience;

int premium = this->premium = 10 \* numberWorkingDays;

int allSalary = this->allSalary = normalSalary + surchangeRecord + premium;

}

Driver::~Driver()

{

}

void Driver::showInfo(){

cout << "Iнформацiя про даного спiвробiтника: " << endl;

this->Worker::showInfo();

}

void Driver::showAllInform() {

cout << "Стандартна зарплата: " << this->normalSalary << endl;

cout << "Доплата за кiлькiсть рокiв стажу: " << this->surchangeRecord << endl;

cout << "Премiя за додатковi робочi днi: " << this->premium << endl;

cout << "Пiдсумкова зарплата: " << this->allSalary << endl;

}

Лістинг 32. Код файлу «Driver.cpp»

#include "ElectricityTrain.h"

#include <iostream>

ElectricityTrain::ElectricityTrain(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int maxProcessingElectricity = this->maxProcessingElectricity = 600;

int electPerKm = this->electPerKm = 4;

int enginePower =this->enginePower = 400;

int maxSpeed = this->maxSpeed = 50;

int minSpeed = this->minSpeed = 10;

maxDistance = maxProcessingElectricity / electPerKm;

}

ElectricityTrain::~ElectricityTrain()

{

}

void ElectricityTrain::showInfo() {

cout << "Електричний поїзд : " << endl;

this->Train::showInfo();

}

void ElectricityTrain::showTechDitails() {

cout << "Тип двигуна: електричний" << endl;

cout << "Максимальна кiлькiсть електрики без перенагрiвання: " << this->maxProcessingElectricity << endl;

cout << "Кiлькiсть кiловат на 10 кiлометрiв: " << this->electPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна вiдстань без охолодження двигунiв: " << this->maxDistance << endl;

}

Лістинг 33. Код файлу «ElectricityTrain.cpp»

#include "IntercityTrain.h"

#include <iostream>

IntercityTrain::IntercityTrain(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int fuelCapacity = this->fuelCapacity = 900;

int fuelPerKm = this->fuelPerKm = 5;

int enginePower = this->enginePower = 700;

int maxSpeed = this->maxSpeed = 140;

int minSpeed = this->minSpeed = 70;

maxDistance = fuelCapacity / fuelPerKm;

}

IntercityTrain::~IntercityTrain()

{

}

void IntercityTrain::showInfo() {

cout << "Поїзд iнтерсiтi" << endl;

this->Train::showInfo();

}

void IntercityTrain::showTechDitails() {

cout << "Тип двигуна: дизельний" << endl;

cout << "Об'єм баку для палива: " << this->fuelCapacity << endl;

cout << "Витратат палива на 10 кiлометрiв: " << this->fuelPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна вiдстань: " << this->maxDistance << endl;

}

Лістинг 34. Код файлу «IntercityTrain.cpp»

#include "Keeper.h"

#include <iostream>

Keeper::Keeper(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday, int numberWorkingDays) : Worker(typeOfWorker, nameWorker, surname, age, experience, birthday, numberWorkingDays) {

this->numberWorkingDays = numberWorkingDays;

int normalSalary = this->normalSalary = 3500;

int surchangeRecord = this->surchangeRecord = 100 \* experience;

int premium = this->premium = 10 \* numberWorkingDays;

int allSalary = this->allSalary = normalSalary + surchangeRecord + premium;

}

Keeper::~Keeper()

{

}

void Keeper::showInfo() {

cout << "Iнформацiя про даного спiвробiтника: " << endl;

this->Worker::showInfo();

}

void Keeper::showAllInform() {

cout << "Стандартна зарплата: " << this->normalSalary << endl;

cout << "Доплата за кiлькiсть рокiв стажу: " << this->surchangeRecord << endl;

cout << "Премiя за додатковi робочi днi: " << this->premium << endl;

cout << "Пiдсумкова зарплата: " << this->allSalary << endl;

}

Лістинг 35. Код файлу «Keeper.cpp»

#include "Maid.h"

#include <iostream>

Maid::Maid(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday, int numberWorkingDays) : Worker(typeOfWorker, nameWorker, surname, age, experience, birthday, numberWorkingDays) {

this->numberWorkingDays = numberWorkingDays;

int normalSalary = this->normalSalary = 3500;

int surchangeRecord = this->surchangeRecord = 100 \* experience;

int premium = this->premium = 10 \* numberWorkingDays;

int allSalary = this->allSalary = normalSalary + surchangeRecord + premium;

}

Maid::~Maid()

{

}

void Maid::showInfo() {

cout << "Iнформацiя про даного спiвробiтника: " << endl;

this->Worker::showInfo();

}

void Maid::showAllInform() {

cout << "Стандартна зарплата: " << this->normalSalary << endl;

cout << "Доплата за кiлькiсть рокiв стажу: " << this->surchangeRecord << endl;

cout << "Премiя за додатковi робочi днi: " << this->premium << endl;

cout << "Пiдсумкова зарплата: " << this->allSalary << endl;

}

Лістинг 36. Код файлу «Maid.cpp»

#include <conio.h>

#include <iostream>

#include <Windows.h>

#include <string>

#include <iomanip>

#include <cstdlib>

#include "Administrator.h"

#include "User.h"

using namespace std;

#define ADMIN\_PASSWORD "512678"

int main(int argc, char\*\* argv) {

setlocale(LC\_ALL,"Ukrainian");

unsigned int start\_time = clock();

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

HWND window\_header = GetConsoleWindow();

SetWindowPos(window\_header, HWND\_TOP, 183, 40, 2100, 300, NULL);

system("mode 110,50");

system("color F0");

SYSTEMTIME st;

GetLocalTime(&st);

string choice;

string password;

int number;

User user;

Administrator administrator;

while(true) {

if(st.wHour >= 6 && st.wHour <= 11) cout << "\t\t\tДоброго ранку!" << endl;

if(st.wHour >= 12 && st.wHour <= 18) cout << "\t\t\tДоброго дня!" << endl;

cout.width(45);

if(st.wHour >= 19 && st.wHour <= 23) cout << "\t\t\tДоброго вечора!" << endl;

if(st.wHour >= 0 && st.wHour <= 5) cout << "\t\t\tДоброї ночi!" << endl;

cout << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((9 << 3) | 15));

cout << setiosflags(ios::left) << setw(31) << "Ввiйти як пасажир - натиснiть 1" << endl;

cout << "Ввiйти як адмiнiстратор - натиснiть 2" << endl << endl;

cout << "Вибiр:";

SetConsoleTextAttribute(hConsole, (WORD) ((15 << 4) | 4));

cout << " ";

cin >> choice;

system("cls");

try {

number = stoi(choice);

}

catch(exception e) {

number = 3;

}

switch(number) {

case 1:

user.featureList();

for(int i = 0; i < 15; i++) cout << endl;

cout << setiosflags(ios::right) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl ;

cout << setiosflags(ios::right) << "\* Курсова робота \"Залiзничний вокзал\" \*" << endl;

cout << setiosflags(ios::right) << "\* Виконав: ст.гр.КI-12 \*" << endl;

cout << setiosflags(ios::right) << "\* Городчук Тарас Вiталiйович \*" << endl << endl;

cout << setiosflags(ios::right) << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl ;

break;

case 2:

cout << "Для входу вiд iменi адмiнiстратора введiть код доступу : " << endl;

cin >> password;

system("cls");

if(password == ADMIN\_PASSWORD) {

// привiлегiї адмiнiстратора

administrator.featureList();

break;

}

else

cout << "Некоректний пароль!" << endl;

case 3:

cout << "Данi введено некоректно!" << endl;

break;

}

cout << "Для виходу з програми натиснiть q..." << endl;

cout << "Для того щоб увiйти заново натиснiть будь-яку клавiшу..." << endl;

cin >> choice;

system("cls");

if(choice == "q") {

break;

}

}

\_getch();

return 0;

}

Лістинг 37. Код файлу «main.cpp»

#include "PassengerTrain.h"

#include <iostream>

PassengerTrain::PassengerTrain(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int fuelCapacity = this->fuelCapacity = 600;

int fuelPerKm = this->fuelPerKm = 2;

int enginePower = this->enginePower = 800;

int maxSpeed = this->maxSpeed = 80;

int minSpeed = this->minSpeed = 30;

maxDistance = fuelCapacity / fuelPerKm;

}

PassengerTrain::~PassengerTrain()

{

}

void PassengerTrain::showInfo() {

cout << "Звичайний дизельний поїзд : " << endl;

this->Train::showInfo();

}

void PassengerTrain::showTechDitails() {

cout << "Тип двигуна: дизельний" << endl;

cout << "Об'єм баку для палива: " << this->fuelCapacity << endl;

cout << "Витрата палива на 10 кiлометрiв: " << this->fuelPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна вiдстань: " << this->maxDistance << endl;

}

Лістинг 38. Код файлу «PassengerTrain.cpp»

#include "RailbusTrain.h"

#include <iostream>

RailbusTrain::RailbusTrain(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int fuelCapacity = this->fuelCapacity = 400;

int fuelPerKm = this->fuelPerKm = 3;

int enginePower = this->enginePower = 600;

int maxSpeed = this->maxSpeed = 100;

int minSpeed = this->minSpeed = 40;

maxDistance = fuelCapacity / fuelPerKm;

}

RailbusTrain::~RailbusTrain()

{

}

void RailbusTrain::showInfo() {

cout << "Рейковий автобус" << endl;

this->Train::showInfo();

}

void RailbusTrain::showTechDitails() {

cout << "Тип двигуна: дизельний" << endl;

cout << "Об'єм баку для палива: " << this->fuelCapacity << endl;

cout << "Витрата палива на 10 кiлометрiв: " << this->fuelPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна вiдстань: " << this->maxDistance << endl;

}

Лістинг 39. Код файлу «RailbusTrain.cpp»

#include "Schedule.h"

void Schedule::lookSchedule(Train \*\* array, int numerator)

{

Schedule dataSchedule;

cout << setw(20) << "Тип поїзда" << setw(20) << "Номер поїзда" << setw(19) << "Назва поїзда" << setw(20) << "Час прибуття" << setw(20) << "Час вiдправлення" << setw(16) << "Платформа" << setw(20) << endl;

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "usual") {

dataSchedule.dataOnScreen(array, numerator, i);

}

}

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "electr") {

dataSchedule.dataOnScreen(array, numerator, i);

}

}

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "railbus") {

dataSchedule.dataOnScreen(array, numerator, i);

}

}

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "intercity") {

dataSchedule.dataOnScreen(array, numerator, i);;

}

}

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "TGV") {

dataSchedule.dataOnScreen(array, numerator, i);

}

}

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfTrain() == "CargoTrain") {

dataSchedule.dataOnScreen(array, numerator, i);

}

}

}

void Schedule::dataOnScreen(Train \*\* array, int numerator, int i)

{

if (i < numerator) {

cout.width(18);

cout << array[i]->getTypeOfTrain();

cout.width(18);

cout << array[i]->getNumberTrain();

cout.width(24);

cout << array[i]->getNameTrain();

cout.width(20);

cout << array[i]->getArrivalTime();

cout.width(24);

cout << array[i]->getDepartureTime();

cout.width(18);

cout << array[i]->getPlatform() << endl;

}

}

Лістинг 40. Код файлу «Schedule.cpp»

#include "TGV.h"

#include <iostream>

TGV::TGV(string typeOfTrain, int numberTrain, string nameTrain) : Train(typeOfTrain, numberTrain, nameTrain) {

int maxProcessingElectricity = this->maxProcessingElectricity = 3000;

int electPerKm = this->electPerKm = 7;

int enginePower = this->enginePower = 1300;

int maxSpeed = this->maxSpeed = 575;

int minSpeed = this->minSpeed =230 ;

maxDistance = maxProcessingElectricity / electPerKm;

}

TGV::~TGV()

{

}

void TGV::showInfo() {

cout << "Поїзд TGV" << endl;

this->Train::showInfo();

}

void TGV::showTechDitails() {

cout << "Тип двигуна: електричний" << endl;

cout << "Максимальна кiлькiсть електрики без перенагрiвання: " << this->maxProcessingElectricity << endl;

cout << "Кiлькiсть електрики на 10 кiлометрiв: " << this->electPerKm << endl;

cout << "Потужнiсть двигуна: " << this->enginePower << endl;

cout << "Мiнiмальна швидкiсть: " << this->minSpeed << endl;

cout << "Максимальна швидкiсть: " << this->maxSpeed << endl;

cout << "Максимальна вiдстань без охолодження двигуна: " << this->maxDistance << endl;

}

Лістинг 41. Код файлу «TGV.cpp»

#include "Ticket.h"

Ticket::~Ticket() {

}

string Ticket::getNameTrain() {

return this->nameTrain;

}

int Ticket::getNumberCar() {

return this->numberCar;

}

void Ticket::setPlace(string place) {

this->place = place;

}

string Ticket::getPlace() {

return this->place;

}

int Ticket::getPrice() {

return this->price;

}

Лістинг 42. Код файлу «Ticket.cpp»

##include "TicketProcessing.h"

Ticket\*\* TicketProcessing::createTicketList(string \* lineArray, int numerator) {

Ticket \*\* infoArray = new Ticket\*[numerator];

for (int i = 0; i < numerator; i++)

{

//Парсинг рядка

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = lineArray[i].find\_first\_of(';', pos)) != string::npos) {

localArray.push\_back(lineArray[i].substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(lineArray[i].substr(pos));

infoArray[i] = new Ticket(localArray[0], atoi(localArray[1].c\_str()), localArray[2], atoi(localArray[3].c\_str()));

}

return infoArray;

}

int TicketProcessing::getNumberLine(string nameFile) {

int numerator = 0;

char \*str = new char[1024];

ifstream infile;

infile.open(nameFile);

if (!infile)

{

cout << "Неможливо вiдкрити файл!" << endl;

}

while (!infile.eof())

{

infile.getline(str, 1024, '\n');

numerator++;

}

numerator--;

return numerator;

}

string\* TicketProcessing::getStrArray(string name, int numerator) {

string \*lineArray = new string[numerator];

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

for (int i = 0; i < numerator; i++)

{

getline(infile, lineArray[i]);

}

return lineArray;

infile.close();

}

Ticket\*\* TicketProcessing::findFreePlace(Ticket \*\* array, string property, int numerator, int trainNumerator) {

if (trainNumerator > 0) {

Ticket\*\* newList = new Ticket\*[trainNumerator];

int j = 0;

for (int i = 0; i < numerator; i++) {

if (array[i]->getNameTrain() == property) {

newList[j] = array[i];

j++;

}

}

return newList;

}

}

int TicketProcessing::getNumeratorOfSearch(Ticket \*\* array, string property, int numerator) {

int j = 0;

for (int i = 0; i < numerator; i++) {

if (array[i]->getNameTrain() == property) {

j++;

}

}

return j;

}

void TicketProcessing::BuyTicket(Ticket\*\* array, int numerator, int price) //Прописуємо метод для купiвлi квитка

{

cout << setw(20) << "Назва поїзда" << setw(20) << "Номер вагона" << setw(19) << "Вiльнi мiсця" << endl;

for (int i = 0; i < numerator; i++) {

cout.width(21);

cout << array[i]->getNameTrain();

cout.width(21);

cout << array[i]->getNumberCar();

cout.width(24);

cout << array[i]->getPlace() << endl;

}

}

string TicketProcessing::choicePlace(Ticket\*\* ticket, string place) {

string placa;

placa = ticket[0]->getPlace();

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = placa.find\_first\_of(',', pos)) != string::npos) {

localArray.push\_back(placa.substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(placa.substr(pos));

bool flag = false;

for (int i = 0; i < localArray.size(); i++) {

if (place == localArray[i]) {

flag = true;

}

}

return place;

}

int TicketProcessing::privilegedPosition(int price)

{

while (true) {

cout << "Для придбання пiльгового квитка натиснiть 1,\nДля придбання звичайного квитка натиснiть 2" << endl;

cin >> numeric;

system("cls");

if (numeric == "1") {

while (true)

{

cout << "Оберiть пiльгову категорiю : " << endl;

cout << "Чорнобилець - 1" << endl;

cout << "Студент - 2" << endl;

cout << "Дитина вiком вiд 6 рокiв - 3" << endl;

cout << "Ветеран - 4" << endl;

cout << "Пенсiонер - 5" << endl;

cin >> number;

if (number == "1") {

price = (price + 30) / 4;

break;

}

if (number == "2") {

price = price / 2;

break;

}

if (number == "3") {

price = price \* 2 / 3;

break;

}

if (number == "4") {

price = price / 4;

break;

}

if (number == "5") {

int minOld = 65;

cout << "Введiть свiй вiк:" << endl;

cin >> old;

if (old < minOld) {

cout << "Ви не є пенсiонером" << endl;

break;

}

else

price = (log(price + 100 - old) \* minOld / 10);

break;

}

if (number != "1" && number != "2" && number != "3" && number != "4" && number != "5")

{

cout << endl;

cout << "Некоректне введення!" << endl << endl;

}

}

break;

}

if (numeric == "2") {

price = price;

break;

}

if (numeric != "1" && numeric != "2")

{

cout << endl;

cout << "Некоректне введення!" << endl << endl;

}

return price;

}

}

string TicketProcessing::bookingBuying(int numberCar, string place, int price)

{

while (true)

{

cout << "Для бронювання квитка натиснiть 1, для купiвлi натиснiть 2" << endl;

cin >> numeric;

system("cls");

if (numeric == "1") {

price = price + 10;

string sNumberCar = to\_string(numberCar);

string sPlace = place;

string sPrice = to\_string(price);

string buff = "Номер вагона:" + sNumberCar + " Номер мiсця:" + sPlace + " Цiна:" + sPrice;

cout << "Квитанцiю про бронювання мiсця надруковано!" << endl;

return buff;

break;

}

if (numeric == "2") {

string sNumberCar = to\_string(numberCar);

string sPlace = place;

string sPrice = to\_string(price);

string buff = "Номер вагона:" + sNumberCar + " Номер мiсця:" + sPlace + " Цiна:" + sPrice;

cout << "Ваш квиток надруковано!" << endl;

return buff;

break;

}

if (numeric != "1" && numeric != "2")

{

cout << endl;

cout << "Некоректне введення!" << endl << endl;

}

}

}

string TicketProcessing::convertToString(Ticket ticket) {

string str;

string sNameTrain(ticket.getNameTrain());

string sNumberCar = to\_string(ticket.getNumberCar());

string sPlace(ticket.getPlace());

string sPrice = to\_string(ticket.getPrice());

str = sNameTrain + ";" + sNumberCar + ";" + sPlace + ";" + sPrice + ";" + "\n";

return str;

}

int TicketProcessing::deletePlace(string nameFile, string nameTrain, string place) {

int placeNumber;

int buffNumber;

TicketProcessing dataTicket;

int numerator = dataTicket.getNumberLine(nameFile);

string\* lineArray = dataTicket.getStrArray(nameFile, numerator);

Ticket\*\* infoArray = dataTicket.createTicketList(lineArray, numerator);

Ticket\*\* newList = new Ticket\*[1];

int j = 0, number;

for (int i = 0; i < numerator; i++) {

if (infoArray[i]->getNameTrain() == nameTrain) {

newList[j] = infoArray[i];

j++;

number = i;

}

}

string placa = newList[0]->getPlace();

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = placa.find\_first\_of(',', pos)) != string::npos) {

localArray.push\_back(placa.substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(placa.substr(pos));

bool flag = false;

for (int i = 0; i < localArray.size(); i++) {

if (place == localArray[i]) {

placeNumber = i;

flag = true;

localArray.erase(localArray.begin() + placeNumber);

string numbersString;

for (int i = 0; i < localArray.size() - 1; i++) {

numbersString += localArray[i] + ",";

}

numbersString += localArray[localArray.size() - 1];

newList[0]->setPlace(numbersString);

infoArray[number] = newList[0];

ofstream fout;

fout.open("DovidnukPlace.txt", ios\_base::trunc);

for (int i = 0; i < numerator; i++) {

fout << dataTicket.convertToString(\*infoArray[i]);

}

fout.close();

}

buffNumber = 1;

}

if (!flag) {

buffNumber = 2;

cout << "Обране вами мiсце вже заброньовано!\n";

}

return buffNumber;

}

Лістинг 43. Код файлу «TicketProcessing.cpp»

#include "Train.h"

Train::~Train() {

}

void Train::showInfo() {

cout << "Тип поїзда: " << this->typeOfTrain << endl;

cout << "Номер поїзда: " << this->numberTrain << endl;

cout << "Назва поїзда: " << this->nameTrain << endl;

}

void Train::setTypeOfTrain(string typeOfTrain) {

this->typeOfTrain = typeOfTrain;

}

string Train::getTypeOfTrain() {

return this->typeOfTrain;

}

void Train::setNumberTrain(int numberTrain) {

this->numberTrain = numberTrain;

}

int Train::getNumberTrain() {

return this->numberTrain;

}

void Train::setNameTrain(string nameTrain) {

this->nameTrain = nameTrain;

}

string Train::getNameTrain() {

return this->nameTrain;

}

void Train::setArrivalTime(string arrivalTime) {

this->arrivalTime = arrivalTime;

}

string Train::getArrivalTime() {

return this->arrivalTime;

}

void Train::setDepartureTime(string departureTime) {

this->departureTime = departureTime;

}

string Train::getDepartureTime() {

return this->departureTime;

}

void Train::setPlatform(int platform) {

this->platform = platform;

}

int Train::getPlatform() {

return this->platform;

}

Лістинг 44. Код файлу «Train.cpp»

#include"TrainProcessing.h"

Train\*\* TrainProcessing::createTrainList(string \* lineArray, int numerator) {

Train \*\* infoArray = new Train\*[numerator];

for (int i = 0; i < numerator; i++)

{

//Парсинг рядка

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = lineArray[i].find\_first\_of(';', pos)) != string::npos) {

localArray.push\_back(lineArray[i].substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(lineArray[i].substr(pos));

infoArray[i] = new Train(localArray[0], atoi(localArray[1].c\_str()), localArray[2], localArray[3], localArray[4], atoi(localArray[5].c\_str()));

}

return infoArray;

}

int TrainProcessing::getNumberLine(string nameFile) {

int numerator = 0;

char \*str = new char[1024];

ifstream infile;

infile.open(nameFile);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

// Пiдраховуємо кiлькiсть рядкIв

while (!infile.eof())

{

infile.getline(str, 1024, '\n');

numerator++;

}

numerator--;

return numerator;

infile.close();

}

string\* TrainProcessing::getStrArray(string name, int numerator) {

string \*lineArray = new string[numerator];

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

for (int i = 0; i < numerator; i++)

{

getline(infile, lineArray[i]);

}

return lineArray;

infile.close();

}

void TrainProcessing::lookTrain(Train\*\* array, int numerator)

{

cout << setw(20) << "Тип поїзда" << setw(20) << "Номер поїзда" << setw(19) << "Назва поїзда" << setw(20) << "Час прибуття" << setw(20) << "Час вiдправлення" << setw(16) << "Платформа" << setw(20) << endl;

for (int i = 0; i < numerator; i++) {

cout.width(20);

cout << array[i]->getTypeOfTrain();

cout.width(20);

cout << array[i]->getNumberTrain();

cout.width(24);

cout << array[i]->getNameTrain();

cout.width(17);

cout << array[i]->getArrivalTime();

cout.width(16);

cout << array[i]->getDepartureTime();

cout.width(18);

cout << array[i]->getPlatform() << endl;

}

}

Train\*\* TrainProcessing::findTrain(Train \*\* array, string property, int numerator, int trainNumerator) {

if (trainNumerator > 0) {

Train\*\* newList = new Train\*[trainNumerator];

int j = 0;

for (int i = 0; i < numerator; i++) {

if (to\_string(array[i]->getNumberTrain()) == property) {

newList[j] = array[i];

j++;

}

else if (array[i]->getNameTrain() == property) {

newList[j] = array[i];

j++;

}

}

return newList;

}

}

int TrainProcessing::getNumeratorOfSearch(Train \*\* array, string property, int numerator) {

int j = 0;

for (int i = 0; i < numerator; i++) {

if (to\_string(array[i]->getNumberTrain()) == property) {

j++;

}

else if (array[i]->getNameTrain() == property) {

j++;

}

}

return j;

}

void TrainProcessing::setNewData(string nameFile, string newData) {

ofstream fileOut;

fileOut.open(nameFile, ios\_base::app);

if (!fileOut) {

cout << "Неможливо вiдкрити файл" << endl;

}

else

{

fileOut << newData << endl;

}

fileOut.close();

}

string TrainProcessing::convertToString(string typeOfTrain, int numberTrain, string nameTrain, string arrivalTime, string departureTime, int platform) { //Прописуємо метод для конвертування даних в тип string

string sTypeOfTrain = typeOfTrain;

string sNumberTrain = to\_string(numberTrain);

string sNameTrain = nameTrain;

string sArrivalTime = arrivalTime;

string sDepartureTime = departureTime;

string sPlatform = to\_string(platform);

string buff = sTypeOfTrain + ";" + sNumberTrain + ";" + sNameTrain + ";" + sArrivalTime + ";" + sDepartureTime + ";" + sPlatform;

return buff;

}

string TrainProcessing::convertToStringForTicket(int numberTrain, string nameTrain, string arrivalTime, string departureTime, int platform) {

string sNumberTrain = to\_string(numberTrain);

string sNameTrain = nameTrain;

string sArrivalTime = arrivalTime;

string sDepartureTime = departureTime;

string sPlatform = to\_string(platform);

string buff = sNumberTrain + " " + sNameTrain + " " + sArrivalTime + " " + sDepartureTime + " " + sPlatform + "\n";

return buff;

}

Лістинг 45. Код файлу «TrainProcessing.cpp»

#include "User.h"

TrainProcessing dataTrain;

Schedule dataSchedule;

TicketProcessing dataTicket;

AdditionalServices dataAdditionalService;

Benefits dataBenefits;

void User::lookSchedule() {

int numerator = dataTrain.getNumberLine("DovidnukTrain.txt");

string\* stringArray = dataTrain.getStrArray("DovidnukTrain.txt", numerator);

Train\*\* trainArray = dataTrain.createTrainList(stringArray, numerator);

dataSchedule.lookSchedule(trainArray, numerator);

cout << endl;

}

void User::searchTrain() {

string buff;

string numeric;

cout << "Введiть номер чи назву поїзда : " << endl;

cin >> buff;

int numerator = dataTrain.getNumberLine("DovidnukTrain.txt");

string\* stringArray = dataTrain.getStrArray("DovidnukTrain.txt", numerator);

Train\*\* trainArray = dataTrain.createTrainList(stringArray, numerator);

int trainNumerator = dataTrain.getNumeratorOfSearch(trainArray, buff, numerator);

Train\*\* pointerStringArray = dataTrain.findTrain(trainArray, buff, numerator, trainNumerator);

if (trainNumerator > 0) {

dataTrain.lookTrain(pointerStringArray, trainNumerator);

cout << "Для перегляду технiчної iнформацiї про ваш поїзд натиснiть 1\n";

cin >> numeric;

system("cls");

if (numeric == "1") {

if (pointerStringArray[0]->getTypeOfTrain() == "electr") {

ElectricityTrain elctrTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

elctrTrain.showInfo();

elctrTrain.showTechDitails();

cout << endl;

}

if (pointerStringArray[0]->getTypeOfTrain() == "usual") {

PassengerTrain dieslTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

dieslTrain.showInfo();

dieslTrain.showTechDitails();

cout << endl;

}

if (pointerStringArray[0]->getTypeOfTrain() == "railbus") {

RailbusTrain railTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

railTrain.showInfo();

railTrain.showTechDitails();

cout << endl;

}

if (pointerStringArray[0]->getTypeOfTrain() == "intercity") {

IntercityTrain interTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

interTrain.showInfo();

interTrain.showTechDitails();

cout << endl;

}

if (pointerStringArray[0]->getTypeOfTrain() == "TGV") {

TGV TGVTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

TGVTrain.showInfo();

TGVTrain.showTechDitails();

cout << endl;

}

if (pointerStringArray[0]->getTypeOfTrain() == "CargoTrain") {

CargoTrain CargTrain(pointerStringArray[0]->getTypeOfTrain(), pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain());

CargTrain.showInfo();

CargTrain.showTechDitails();

cout << endl;

}

}

}

else

cout << "За вашим запитом поїздiв не знайдено" << endl;

}

void User::buyTicket() {

string buff;

string place;

int buff2;

cout << "Введiть назву чи номер поїзда" << endl;

cin >> buff;

int numerator = dataTrain.getNumberLine("DovidnukTrain.txt");

string\* stringArray = dataTrain.getStrArray("DovidnukTrain.txt", numerator);

Train\*\* trainArray = dataTrain.createTrainList(stringArray, numerator);

int trainNumerator = dataTrain.getNumeratorOfSearch(trainArray, buff, numerator);

Train\*\* pointerStringArray = dataTrain.findTrain(trainArray, buff, numerator, trainNumerator);

if (trainNumerator > 0) {

buff = dataTrain.convertToStringForTicket(pointerStringArray[0]->getNumberTrain(), pointerStringArray[0]->getNameTrain(), pointerStringArray[0]->getArrivalTime(), pointerStringArray[0]->getDepartureTime(), pointerStringArray[0]->getPlatform());

dataTrain.setNewData("Ticket.txt", buff);

buff = pointerStringArray[0]->getNameTrain();

numerator = dataTicket.getNumberLine("DovidnukPlace.txt");

string\* strTicketArray = dataTicket.getStrArray("DovidnukPlace.txt", numerator);

Ticket\*\* ticketArray = dataTicket.createTicketList(strTicketArray, numerator);

trainNumerator = dataTicket.getNumeratorOfSearch(ticketArray, buff, numerator);

Ticket\*\* ticketStringArray = dataTicket.findFreePlace(ticketArray, buff, numerator, trainNumerator);

if (trainNumerator > 0) {

dataTicket.BuyTicket(ticketStringArray, trainNumerator, ticketStringArray[0]->getPrice());

cout << "Виберiть мiсце" << endl;

cin >> place;

int buffNumber = dataTicket.deletePlace("DovidnukPlace.txt", buff, place);

if (buffNumber == 1) {

buff = dataTicket.choicePlace(ticketStringArray, place);

buff2 = dataTicket.privilegedPosition(ticketStringArray[0]->getPrice());

buff = dataTicket.bookingBuying(ticketStringArray[0]->getNumberCar(), buff, buff2);

dataTrain.setNewData("Ticket.txt", buff);

}

if (buffNumber == 2) {

}

}

}

else {

cout << "По вашому запиту поїздiв не знайдено" << endl;

}

cout << endl;

}

void User::lookPriceAdditionalService() {

dataAdditionalService.lookPriceForAdditionalService("DovidnukService.txt");

cout << endl;

}

void User::lookListBenefits() {

dataBenefits.lookBenefits("DovidnukBenefits.txt");

cout << endl;

}

void User::featureList()

{

string numeric;

int number;

string buff;

User dataUser;

while (true)

{

cout << "| Режим : Користувач |\n\n";

cout << "Для перегляду розкладу поїздiв натиснiть 1\n";

cout << "Для пошуку потрiбного вам поїзда натиснiть 2\n";

cout << "Для придбання чи бронювання квитка натиснiть 3\n";

cout << "Для перегляду цiн на додатковi послуги в поїздах натиснiть 4\n";

cout << "Для перегляду даних на пiльговий проїзд натиснiть 5\n\n";

cout << "Обрати : ";

cin >> numeric;

system("cls");

try {

number = stoi(numeric);

}

catch (exception e) {

cout << "Введiть коректнi данi!!!" << endl;

number = 6;

}

switch (number)

{

case 1:

dataUser.lookSchedule();

break;

case 2:

dataUser.searchTrain();

break;

case 3:

dataUser.buyTicket();

break;

case 4:

dataUser.lookPriceAdditionalService();

break;

case 5:

dataUser.lookListBenefits();

break;

case 6:

break;

}

cout << "Для виходу з режиму простого користувача натиснiть 1" << endl;

cout << "Щоб продовжити роботу натиснiть будь-яку кнопку" << endl;

cin >> numeric;

system("cls");

if (numeric == "1")

break;

}

}

Лістинг 46. Код файлу «User.cpp»

#include "Worker.h"

Worker::~Worker() {

}

void Worker::showInfo() {

cout << "Професiя: " << this->typeOfWorker << endl;

cout << "Iм'я спiвробiтника: " << this->nameWorker << endl;

cout << "Прiзвище спiвробiтника: " << this->surname << endl;

cout << "Вiк: " << this->age << endl;

cout << "Кiлькiсть рокiв стажу: " << this->experience << endl;

cout << "Дата народження: " << this->birthday << endl;

}

void Worker::setTypeOfWorker(string typeOfWorker) {

this->typeOfWorker = typeOfWorker;

}

string Worker::getTypeOfWorker() {

return this->typeOfWorker;

}

void Worker::setNameWorker(string nameWorker) {

this->nameWorker = nameWorker;

}

string Worker::getNameWorker() {

return this->nameWorker;

}

void Worker::setAge(int age) {

this->age = age;

}

int Worker::getAge() {

return this->age;

}

void Worker::setExperience(int experience) {

this->experience = experience;

}

int Worker::getExperience() {

return this->experience;

}

void Worker::setBirthday(string birthday) {

this->birthday = birthday;

}

string Worker::getBirthday() {

return this->birthday;

}

void Worker::setSurname(string surname) {

this->surname = surname;

}

string Worker::getSurname() {

return this->surname;

}

Лістинг 47. Код файлу «Worker.cpp»

#include "WorkerProcessing.h"

Worker\*\* WorkerProcessing::createWorkerList(string \* lineArray, int numerator) {

Worker \*\* infoArray = new Worker\*[numerator];

for (int i = 0; i < numerator; i++)

{

//Парсинг рядка

vector<std::string> localArray;

size\_t pos = 0, found;

while ((found = lineArray[i].find\_first\_of(';', pos)) != string::npos) {

localArray.push\_back(lineArray[i].substr(pos, found - pos));

pos = found + 1;

}

localArray.push\_back(lineArray[i].substr(pos));

infoArray[i] = new Worker(localArray[0], localArray[1], localArray[2], atoi(localArray[3].c\_str()), atoi(localArray[4].c\_str()), localArray[5]);

}

return infoArray;

}

int WorkerProcessing::getNumberLine(string nameFile) {

int numerator = 0;

char \*str = new char[1024];

ifstream infile;

infile.open(nameFile);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

// Пiдраховуємо кiлькiсть рядкIв

while (!infile.eof())

{

infile.getline(str, 1024, '\n');

numerator++;

}

numerator--;

return numerator;

infile.close();

}

string\* WorkerProcessing::getStrArray(string name, int numerator) {

string \*lineArray = new string[numerator];

ifstream infile;

infile.open(name);

if (!infile)

{

cout << "Неможливо вiдкрити файл" << endl;

}

for (int i = 0; i < numerator; i++)

{

getline(infile, lineArray[i]);

}

return lineArray;

infile.close();

}

void WorkerProcessing::lookWorker(Worker\*\* array, int numerator)

{

cout << setw(18) << "Професiя" << setw(20) << "Iм'я спiвробiтника" << setw(22) << "Прiзвище спiвробiтника" << setw(18) << "Вiк" << setw(17) << "Стаж" << setw(20) << "День народження" << setw(20) << endl;

for (int i = 0; i < numerator; i++) {

cout.width(18);

cout << array[i]->getTypeOfWorker();

cout.width(18);

cout << array[i]->getNameWorker();

cout.width(24);

cout << array[i]->getSurname();

cout.width(17);

cout << array[i]->getAge();

cout.width(16);

cout << array[i]->getExperience();

cout.width(18);

cout << array[i]->getBirthday() << endl;

}

}

Worker\*\* WorkerProcessing::findWorker(Worker \*\* array, string property, int numerator, int workerNumerator) { //

if (workerNumerator > 0) {

Worker \*\* newList = new Worker\*[workerNumerator];

int j = 0;

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfWorker() == property) {

newList[j] = array[i];

j++;

}

else if (array[i]->getNameWorker() == property) {

newList[j] = array[i];

j++;

}

}

return newList;

}

}

int WorkerProcessing::getNumeratorOfSearch(Worker \*\* array, string property, int numerator) {

int j = 0;

for (int i = 0; i < numerator; i++) {

if (array[i]->getTypeOfWorker() == property) {

j++;

}

else if (array[i]->getNameWorker() == property) {

j++;

}

}

return j;

}

void WorkerProcessing::setNewData(string nameFile, string newData) {

ofstream fileOut;

fileOut.open(nameFile, ios\_base::app);

if (!fileOut) {

cout << "Неможливо вiдкрити файл" << endl;

}

else

{

fileOut << newData << endl;

}

fileOut.close();

}

string WorkerProcessing::convertToString(string typeOfWorker, string nameWorker, string surname, int age, int experience, string birthday) {

string sTypeOfWorker = typeOfWorker;

string sNameWorker = nameWorker;

string sSurname = surname;

string sAge = to\_string(age);

string sExperience = to\_string(experience);

string sBirthday = birthday;

string buff = sTypeOfWorker + ";" + sNameWorker + ";" + sSurname + ";" + sAge + ";" + sExperience + ";" + sBirthday;

return buff;

}

Лістинг 48. Код файлу «WorkerProcessing.cpp»