МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

"ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ" НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА СИСТЕМ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ

**Курсова робота**

з дисципліни «Об’єктно-орієнтоване програмування»

***на тему: «Дослідження та порівняння технік навчання з підкріпленням на базі задачі «Лабіринт»»***

РОБОТУ ВИКОНАВ:

студент групи ДА-82 Зайцев К.Ю. Прийняв:

\_ \_ Булах Б.В.

„\_\_\_” \_ 2019

Київ – 2019

# ЗМІСТ

[ВСТУП 3](#_TOC_250006)

РОЗДІЛ I. ТЕОРЕТИЧНИЙ АНАЛІЗ НАУКОВОЇ ЛІТЕРАТУРИ ТА ВИХІДНІ ПОНЯТТЯ РОБОТИ 5

* 1. [Загальні відомості про автоматизований диспетчерський контроль залізничного вузла 5](#_TOC_250005)
  2. [Оптимальний диспетчерський контроль 7](#_TOC_250004)
  3. [Ядро залізничних систем управління 10](#_TOC_250003)

РОЗДІЛ II. ПОБУДОВА І РЕАЛІЗАЦІЯ ПРОЕКТУ 12

[2.1. Розробка програми 12](#_TOC_250002)

РОЗДІЛ III. ІНТЕРФЕЙС КОРИСТУВАЧА 19

[ВИСНОВКИ 22](#_TOC_250001)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 24](#_TOC_250000)

# ВСТУП

Диспетчер є невід’ємною частиною роботи залізниці по всьому світу з середини 19 століття. Проте й до сьогодні навіть з такими високотехнологічними сигнальними системами як CTC, PTC та ABS, поїзди не можуть безпечно та ефективно слідувати за маршрутами без централізованого керування.

На разі ринок сповнений різноманітними системи диспетчерського контролю. Ці системи створювалися після узгоджень вимог до роботи залізниці у конкретних країнах та залізничних маршрутів, з якими їм потрібно було працювати, що говорить про практично відсутню універсальність. Суттєво, що більшість програмних архітектур було розроблено досить давно, на час, коли вони співпадали з тенденціями програмування та новітніми технологіями, тож на підґрунті вони усі мають однакову структуру. Більш конкретно, такі системи побудовані з фундаментальних компонент, таких як блок-системи контролю інтервалів руху «потяг-потяг» для потягів, що переміщуються між станціями; контроль напрямків руху на для одноколійної секції; автоматизована система забезпечення безпеки потягу (АСЗБП), вбудована у блок-системи; низка пов’язаних між собою девайсів для здійсненню контролю за коліями на територіях, що належать залізничній станції.

Системи варіюються в залежності від того, як вони ініціюють детекцію потягів та реалізують АСЗБП. Однак, не дивлячись на те, що ці системи розміщуються на шкалі від ненадійних з функцією автоматичного посилання сигналу про небезпеку до найбільш просунутих радіо-систем контролю потягів (комунікативних систем контролю потягів), вони не відрізняються у використанні мережі девайсів, що відповідають за дорожній контроль та забезпечення безпеки потягів, що рухаються у межах самої залізничної станції.

Після появи систем централізованого контролю руху (eng. - “centralized traffic control (CTC)”), точного контролю потягів (eng. - “positive train control

(PTC)”), автоматичних сигналів блоків (eng. - “automatic block signals(ABS) ”), потреба в диспетчері, який курсував вздовж залізничних колій, зникла.

Системи контролю залізниці, що пройшли через суттєві зміни та покращення, наразі складають композицію з апаратного забезпечення та підсистем. Який підхід має бути прийнятий для забезпечення функціоналу, безпеки та надійності систем контролю залізниць та впровадження до них сервіс-орієнтованої складової? Проблема потребує переосмислення архітектур програмних продуктів, що вже проявили себе на ринку, та розробки більш бажаних з точки зору автоматизації та надійності систем керування, ґрунтуючись на новітніх комунікаційних технологіях та засобах обробки інформації.

За мету для даної роботи було поставлено розробку нової централізованої системи, в якій функції контролю безпеки на залізничній станції та менеджменту руху потягів поєднані з сервіс-орієнтованими можливостями, керування якою можна здійснювати з єдиного центру управління.

У відповідності з метою роботи поставлені наступні завдання:

* проаналізувати стан проблеми в науковій літературі та Інтернет ресурсах;
* розглянути структуру і можливості мови програмування C;
* створити власну систему диспетчерського контролю з використанням засобів програмування мови C, а саме: розробити програму, що управляє процесом приходу/відправлення потягів, реєструє аварійні ситуації, видає інформацію користувачу про проведення посадки на потяги, стан потягів.

# РОЗДІЛ I

**ТЕОРЕТИЧНИЙ АНАЛІЗ НАУКОВОЇ ЛІТЕРАТУРИ ТА ВИХІДНІ ПОНЯТТЯ РОБОТИ**

# Загальні відомості про автоматизований диспетчерський контроль залізничного вузла.

Автоматизована система диспетчерського контролю (АСДК) – це інтегрований комплекс з прикладних бібліотек та програмного забезпечення сервісного спрямування для роботи з базами даних.

Така система характеризується наступними параметрами:

* + - Модульна структура.
    - Відкритість архітектури програмного забезпечення.
    - Надійність реалізації функцій керування об’єктами.

Еволюція комп’ютеризованих диспетчерських систем продемонстрована у таблиці 2. Ланцюг починається з базової автоматизованої диспетчерської системи, проходить через системи автоматизованого контролю, до глобальної системи, що поєднує диспетчерські функції з контролем кожного потягу [3, с. 86].

Таблиця 1. Еволюція комп’ютеризованих диспетчерських систем DISPATCHER INFORMATION SYSTEM

Local and remote entry of data:

Train characteristics

Train location and status, etc

Record keeping:

Automatic preparation of train sheets Preparation of train performance reports

COMPUTER ASSISTED DISPATCH

Automatic signal clearing: Clearing signals ahead of trains Clearing trains out of sidings

Manual selection of meetjpass locations

Computer calculatcd ETAS at next conflict

Computer suggests dispatch decisions

Dispatcher accepts (or over-rides) decisions and aligns switches

AUTOMATED DISPATCH

Computer selection of meet,'pass locations Computer control of switches and signals Dispatcher over-ride in exceptional circumstances

INTEGRATED DISPATCH AND TRAIN CONTROL

Continuous feedback of dispatch decisions to modify train operating decisions.

На базовому рівні, автоматизоване регулювання звільнює диспетчера станції від рутинної діяльності, проте залишає його на місці головного контролера всередині системи. Усі критичні рішення мають бути прийняті диспетчером. З ускладненням таких систем, додаткові функції були додані в наступному порядку:

# Автоматичне управління сигналами

За даним місцем призначення потягу, система може обрати відповідний шлях та скинути всі сигнали перед потягом, забезпечуючи відсутність можливих конфліктів. Ця система здатна відправляти потяги з додаткових колій станції.

# Комп’ютеризоване обчислення часу прибуття

До базових функцій систем диспетчерського контролю належить обчислення очікуваного часу прибуття кожного потягу на усі локації у розкладі. Найпростіший підхід полягає у використанні усередненого часу для кожного класу потягів. Більш ускладнені системи включають деталі стосовно потягів та характеристики шляхів, з використанням такого забезпечення як калькулятор продуктивності потягу (eng.-“TPC”), що оцінює час руху, базуючись на відношеннях потужності до маси потягу, прискорення до уповільнення та спеціальних величинах та обмеженнях для ділянок колії [5, c. 50].

# Диспетчерська логіка

Фінальний рівень складності комп’ютеризованих диспетчерських систем включає в себе здатність обробляти конфлікти та пропонувати можливі рішення. Залежно від реалізації, цей функціонал системи варіює від стандартизованих правил для вирішення конфліктів (стандартних оперативних процедур) до оптимального управління – комп’ютер виконує пошук серед усіх можливих рішень конфліктів та обирає оптимальний. Рішення від усіх неоптимальних типів мають бути здійсненними та не призводити до блокування колій. Ця проблема була вирішена Петерсеном (1983).

# Вбудовані системи управління

Диспетчерські системи найвищого рівня мають неперервно надсилати інформацію щодо прийняття рішень контролю машиністу потяга чи бортового комп’ютера для модифікації поточних параметрів руху потяга із метою мінімізації витрат пального та затримок. Наприклад, якщо потяг запізнюється, ця інформація може бути використана машиністом для зменшення витрат пального раннім гальмуванням.

АСДК використовується як база для створення:

* + - Діючих, інформаційних та управляючих систем.
    - Центральної приймаючої та транслюючої станції.
    - Системи для збору та передачі технологічної інформації.
    - Високорівневої системи контролю аварійних ситуацій на базі центральної системи.

# Оптимальний диспетчерський контроль

Змоделюємо математично проблему контролю для одноколійного шляху між двома станціями. Колія складається з сегментів, обмежених перемикачами, що дають потягу можливість переходити на іншу колію. Проблема оптимального контролю визначає, де потяги мають зустрітися чи здійснювати обгін та на якій колії має бути кожен потяг для мінімізації

пріоритетної величини. Початкові умови – локація потягу та очікуваний час до місця прибуття. Рухи потягів для наступних *T* годин визначені [4, c. 192- 195].

Розглянемо шлях із *K* сегментів, *М* з яких дозволяють потягам пройти повз один одного. Нехай набір з *I* потягів рухається в напрямі 1 і набір *J* – в напрямі 2. Визначимо кінцевий час для *i*-потягу з *I* з *k*-сегменту як 𝒙𝒊𝒌, час для *j*-потягу з *J* з *k*-сегменту як 𝑦𝑗𝑘. 𝑚𝑖𝑗 ∈ 𝑀 - місце зустрічі потягів *i* та *j*, під час якого i знаходиться на 𝐼𝑖𝑗 колії. 𝐼𝑖𝑗 = 0, якщо i залишається на основній колії та 𝐼𝑖𝑗 = 1, якщо і переходить на допоміжну. Потяг j знаходиться на колії 1-𝐼𝑖𝑗.

Нехай 𝑞𝑖1𝑖2 – місце, де 𝑖1 обганяє 𝑖2, 𝑟𝑗1𝑗2 – місце, де 𝑗1 обганяє 𝑗2. В усіх випадках потяг, що обганяє, залишається на основній колії, в той час як повільніший рухається по допоміжній. Нехай 𝑡1𝑖𝑘 та 𝑡2𝑗𝑘 – вільні час для руху потягів i та j вздовж сегменту k. Також нехай 𝑠1𝑖𝑚 та 𝑠2𝑗𝑚 – мінімальний транзитний час, включаючи затримку на поворотах, якщо потяги проходять сегмент на допоміжній колі. (Це залежить від швидкості повороту та ліміту швидкості на допоміжній колії). Нехай ℎ𝑖𝑗 – мінімальний інтервал часу між потягами після зустрічі чи обгону.

Мінімізація загальної затримки еквівалентна мінімізації суми часів прибуття потягів на станції, виходячи з поточних положень 𝑘1𝑖 та 𝑘2𝑗 кожного потягу. Тоді проблема планування можна представити у наступному програмному вигляді:

вибрати 𝑚𝑖𝑗, 𝑙𝑖𝑗 , 𝑞𝑖1𝑖2 , 𝑟𝑗1𝑗2 для min ∑𝑖∈𝐼 𝑥𝑖𝑘 + ∑𝑗∈𝐽 𝑦𝑗𝑘

зважуючи на:

часові обмеження

𝑥𝑖𝑘 ≥ 𝑥𝑖,𝑘−1 + 𝑡1𝑖𝑘; 𝑘 = 𝑘1𝑖 + 1, … , 𝐾

𝑦𝑖𝑘 ≥ 𝑦𝑖,𝑘−1 + 𝑡2𝑗𝑘 ; 𝑘 = 𝑘2𝑗 − 1, … , 1

зустрічні обмеження

𝑥𝑖𝑚𝑖𝑗 ≥ 𝑦𝑗,𝑚𝑖𝑗+1 + ℎ𝑖𝑗

𝑦𝑗𝑚𝑖𝑗 ≥ 𝑥𝑗,𝑚𝑖𝑗−1 + ℎ𝑗𝑖

для всіх 𝑖 ∈ 𝐼, 𝑗 ∈ 𝐽

𝑥𝑖𝑚𝑖𝑗 ≥ 𝑦𝑗,𝑚𝑖𝑗−1 + 𝑙𝑖𝑗 𝑠1𝑖𝑚𝑖𝑗

𝑦𝑗𝑚𝑖𝑗 ≥ 𝑦𝑗,𝑚𝑖𝑗+1 + (1 − 𝑙𝑖𝑗 )𝑠2𝑗𝑚𝑖𝑗

та обмеження для обгону

𝑥𝑖1𝑚 ≥ 𝑥𝑖2𝑚 + ℎ𝑖1𝑖2; 𝑞𝑖1𝑖2 < 𝑚

𝑥𝑖2𝑚 ≥ 𝑥𝑖1𝑚 + ℎ𝑖2𝑖1; 𝑞𝑖1𝑖2 ≥ 𝑚

𝑥𝑖2𝑚 ≥ 𝑥𝑖2𝑚−1 + 𝑠1𝑖2𝑚; 𝑞𝑖1𝑖2 = 𝑚

для всіх 𝑖1 𝑖2 ∈ 𝐼

𝑦𝑗1𝑚 ≥ 𝑦𝑗2𝑚 + ℎ𝑗1𝑗2; 𝑟𝑗1𝑗2 < 𝑚

𝑦𝑗2𝑚 ≥ 𝑦𝑗1𝑚 + ℎ𝑗2𝑗1; 𝑟𝑗1𝑗2 ≥ 𝑚

𝑦𝑗2𝑚 ≥ 𝑦𝑗2𝑚+1 + 𝑠2𝑗2𝑚; 𝑟𝑗1𝑗2 = 𝑚

для всіх 𝑗1𝑗2 ∈ 𝐽

Пряме рішення цієї задачі недосяжне з обчислювальної точки зору доки існує декілька потягів з кінцевою кількістю обгонів. Наразі ця проблема знаходиться на стадії дослідження. Отвей та Зальцхорн (2001) описали складнощі пов’язані зі спробами вирішити її методом грубої сили.

Обчислювальний час зростає експоненційно з розмірами проблеми. Розміри залежать від факторів, таких як: кількість зустрічей/обгонів, ширина колії та швидкість потягу, поточна завантаженість шляхів, різноманіття типів потягів. Наприклад, якщо інтенсивність руху на шляхах в кожному напрямі подвоюється – кількість зустрічей зростає в чотири рази. Складність зростає, якщо розглядати проблему для ліній з декількома коліями. Це пов’язано з появою нових шляхів вирішення конфліктів, які треба проаналізувати для знаходження оптимального.

Розглянемо ситуацію, де існує три варіанти руху під час зустрічі (головна колія та допоміжні з обох боків). Якщо під час планування треба вирішити 10 зустрічей (враховуючи 3-4 потяги в кожному напрямі), існує

𝟑𝟏𝟎 = 𝟓𝟗 𝟎𝟎𝟎 комбінацій місць зустрічей для розглядання. Якщо подвоїти трафік – буде принаймні 40 зустрічей, які потрібно вирішити під час планування. В цьому випадку треба розглянути 𝟑𝟒𝟎 = 𝟏𝟎𝟏𝟗 комбінацій. Це означає, що цілочисельні методи програмного рішення як метод гілок-та- кордонів можна вжити тільки у випадку низької інтенсивності руху. Поточні дослідження ведуться в напрямі евристичних методів, які показують гарні результати (Петерсен, Тейлор (1982)).

# Ядро залізничних систем управління

Комплексна інформаційна система (КІС) – це програмне забезпечення, призначене для роботи з базами даних, мовами програмування, архітектурами систем та користувацьким інтерфейсом для створення найбільш ефективної системи менеджменту інформації залізничної станції. Система працює автономно з можливістю отримання прав доступу оператором для зміни складових бази даних (додавання нових потягів, зміна вимог управління і т.п.). Для зв’язку між системою та усією залізничною мережею, друга має бути покрита телекомунікаційною мережею, що також забезпечує комунікацію між станцією, потягом та двома наступними потягами в даному напрямі. Від реалізації КІС залежить безпека, швидкість та економічність диспетчерських систем.

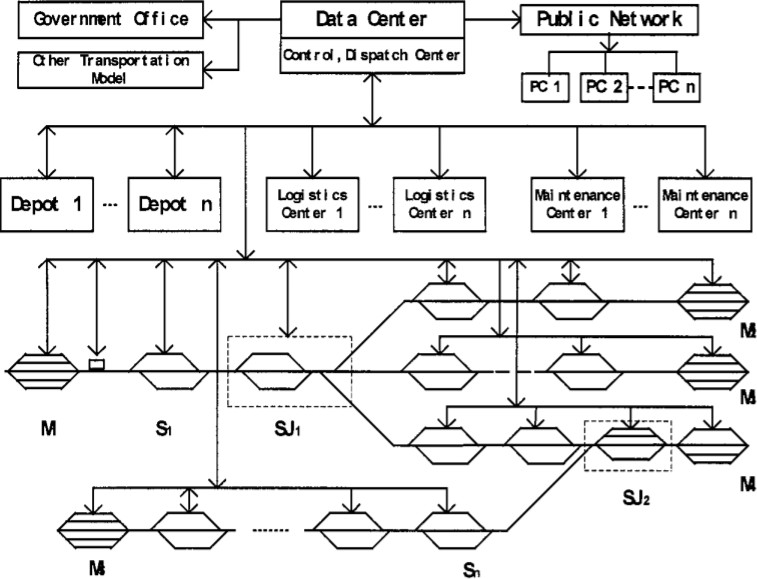
В КІС залізнична мережа ділиться на дві частини: залізничну (мости, тунелі та колії, тип потягу, характеристики локомотиву) та диспетчерську. За допомогою статичної бази даних, динамічна інформація із станом шляхів та диспетчерськими діями подаються в єдиному форматі в реальному часі. Директор станції, користуючись КІС, може отримати поточну ситуацію на мережі та історію диспетчерського контролю і використати це для прийняття рішень [1, c.3].

Архітектура залізничної мережі показана на схемі 1. Вона включає вузлові станції, звичайні станції, сортувальні станції, депо, лінії, центр обслуговування, будівлю вокзалу та центр управління залізницею. Залізнична мережа – це незалежна система, що з’єднана з іншими системами

(транспортною, Інтернет і т.п.). Вона має надавати відповідну інформацію для наведених систем. В залізничній мережі є 6 складових: (1) – дата-центр та центр управління; (2) – станції депо; (3) – логістичні центри; (4) центр

технічного обслуговування; (5) – система з вузлових 𝑀𝑖, звичайних 𝑆𝑖 та сортувальних станцій 𝑆𝐽𝑖; (6) придорожні системи в блоках. Як система, ці частини координуються з центру управління, тож КІС стає центральним елементом усієї залізничної мережі. За допомогою КІС, центр управління може брати дані з усіх шести частин та керувати системою безпечно та ефективно [2, c. 1576].

Схема 1. Архітектура залізничної мережі



# РОЗДІЛ II

**ПОБУДОВА І РЕАЛІЗАЦІЯ ПРОЕКТУ**

# Розробка програми

Для моделювання роботи залізничної станції було створено декілька структур, що представляють ключові об’єкти залізничної мережі.

Основною структурою з якою працює програма є структура диспетчера:

typedef struct schdlr { train\* trains;

} dispatcher;

*trains* – масив вказівників на структуру типу *train*

Структура, що відповідає за об’єкт потягу:

typedef struct one\_train { int ID;

char Destination[20]; departure Departure; arrival Arrival;

char State[15]; int Halt;

char\* Status; int rail;

} train;

*ID* – номер потягу, *Destination* – пункт призначення потягу, *Departure, Arrival*

– структури для зберігання часу відправлення/прибуття потягу на станцію, *State* – стан потягу, *Halt* – час перебування потягу на даній станції, *Status* – статус потягу: «Триває посадка», «Посадка завершена», «Потяг очікується», *rail* – номер колії, на яку прийнято даний потяг.

Структура для збереження часу прибуття та відправлення потягів.

typedef struct time\_table { int hour;

int min;

} arrival, departure;

*hour* – час в годинах прибуття/відправлення потягу, *min* – час в хвилинах

прибуття/відправлення потягу.

Структура залізничної мережі станції.

struct Railroad

{

int size;

int capacity; train\* array;

};

*size* – поточна кількість потягів на станції чи на під’їзді до неї, *capacity* –

максимальна кількість колій, на які станція може приймати потяги, *array* – масив вказівників на структури типу train, що відображає поточне розташування потягів на коліях.

Структура для передачі бази даних на обробку потоками програми

struct thread\_args {

train\* trains; int lines;

};

*trains* – масив вказівників на структуру типу *train*, *lines* – поточна кількість

потягів у базі даних

Для реалізації диспетчерської та сервісної складових, використано два потоки для обробки диспетчерських та користувацьких запитів. При створенні, в потоки передається актуальна база даних потягів, яка оновлюється кожного разу функцією *update\_boarding()*, в яку передається масив потягів, що зберігається в структурі диспетчера, та їх поточна кількість. Для передачі даних для обробки потоками використовується структура типу thread\_args з відповідними полями:

dispatcher.trains = trains; update\_boarding(&(dispatcher.trains), lines);

struct thread\_args \*args = malloc (sizeof (struct thread\_args)); args->trains = dispatcher.trains;

args->lines = lines; pthread\_t thread1, thread2;

const char \*message;

while(1){ printf("\n1.DISPATCHER\n"); printf("2.USER\n");

printf("3.EXIT\n"); printf("\nChoose program mode: "); int action\_number;

scanf("%d", &action\_number); switch (action\_number) {

case 1:

pthread\_create(&thread1, NULL, DISP\_mode, (void\*) args); pthread\_join(thread1, NULL);

break; case 2:

pthread\_create(&thread2, NULL, USER\_mode, (void\*) args); pthread\_join(thread2, NULL);

break; case 3:

return 0; default:

printf("Enter the correct number please!\n"); break;

}

}

Робота в режимі диспетчера організована за допомогою функції, яку викликає відповідний потік при ініціалізації. В аргументи передаємо структуру, що містить базу даних, з якою буде працювати диспетчер. Виконувані диспетчером дії полягають в проведенні операцій над потягами, що прибувають та відправляються із залізничної станції. Ці дії включають: відслідковування процесу прибуття/відправлення потягів зі станції, реєстрування надзвичайних ситуацій, додавання нового потягу у базу даних, редагування параметрів потягу у базі даних і т.д.

void \*DISP\_mode( void \* \_args)

{

struct thread\_args \*args = (struct thread\_args \*) \_args; struct Railroad\* rail\_road = createRailroad(10);

int lines = args->lines; while (1) {

char\* status; char\* state;

status = malloc(sizeof(char)\*10); state = malloc(sizeof(char)\*10);

// to get current time s = time(NULL);

current\_time = localtime(&s); printf("\nCurrent time: %02d:%02d:%02d",

current\_time->tm\_hour, current\_time->tm\_min, current\_time->tm\_sec);

printf("\n1. Add train to the base\n"); printf("2. Edit train in base\n"); printf("3. Remove train from base\n"); printf("4. Print the whole base\n");

printf("5. Follow trains arrival / departure\n"); printf("6. Register emergency\n");

printf("7. Return to MODE menu\n"); printf("\nChoose next action : "); int action\_number;

scanf("%d", &action\_number); switch (action\_number) { case 1:

add\_train\_to\_base(&(args->trains), &lines); break;

case 2:

edit\_train\_in\_base(&(args->trains), lines); break;

case 3:

remove\_train\_from\_base(&(args->trains), &lines); break;

case 4:

print\_train\_base(&(args->trains), lines); break;

case 5:

train\_tracking(&(args->trains), lines, rail\_road); break;

case 6:

reg\_emergency(&(args->trains), lines, rail\_road); break;

case 7:

pthread\_exit(NULL); break;

default:

printf("Enter the correct number please!\n"); break;

}

}

}

Для роботи у режимі диспетчера необхідно мати можливість змінювати поточну ситуацію на залізничній мережі вокзалу. Для цього створено

структуру railroad, полями якої є поточна кількість потягів на станції, загальна місткість та масив з потягів, що перебувають в межах станції.

Для створення залізничної мережі вокзалу використано функцію createRailroad(int capacity), в якій аргументом є константа, що визначає максимальну кількість потягів, яку вокзал може прийняти одночасно.

struct Railroad\* createRailroad(int capacity)

{

struct Railroad\* rail\_road = (struct Railroad\*) malloc(sizeof(struct Railroad));

rail\_road->capacity = capacity;

rail\_road->array = (train\*) malloc(rail\_road->capacity \* sizeof(train)); rail\_road->size = 0;

return rail\_road;

}

Робота з залізничною мережею передбачає контроль прибуття та відправлення

потягів на станцію та отримання інформації про потяг на певній колії. Функція для управління прибуттям потягів:

void arr\_to\_rail\_road(struct Railroad\* rail\_road, train item)

{

if (isFull(rail\_road)) return;

rail\_road->array[rail\_road->size] = item; item.rail = rail\_road->size;

rail\_road->size = rail\_road->size + 1; printf("%d added to rail\_road\n", item.ID);

}

Якщо на станції є вільні колії, потягу приписується відповідний індекс у

масиві мережі вокзалу. Полю *rail* структури *item* (що відповідає за даний потяг) присвоюється значення відповідної колії. Поточна кількість потягів збільшується на одиницю.

Слідкування за потоком потягів на станції реалізовано у функції train\_tracking:

void train\_tracking(train\*\* trains, int lines, struct Railroad\* rail\_road){

s = time(NULL);

current\_time = localtime(&s);

for(int i=0;i<lines;i++){ if((\*trains)[i].Arrival.hour==current\_time->tm\_hour){

if(((\*trains)[i].Arrival.min-current\_time-

>tm\_min<10)&&((\*trains)[i].Arrival.min-current\_time->tm\_min>0)) arr\_to\_rail\_road(rail\_road, (\*trains)[i]);



int k=0; k<rail\_road->size; k++

rail\_road->array[k] = rail\_road->array[k + 1]

\*del\_item = rail\_road->array[0]

}

}

check\_departure(rail\_road); if(rail\_road->array[0].ID)

printf("Next to depart - %d train\n", rail\_road->array[0].ID);

}

Після запуску функції з диспетчерського меню, в базі даних потягів

шукаються потяги, що знаходяться на підході до станції. Для цього за допомогою логічних умов перевіряється різниця між часом прибуття потягу, зазначеному у базі даних, та поточним часом, що оновлюється після кожного запуску функції за допомогою методу *localtime* з бібліотеки *time.h.*

Після додавання потягів на станцію, відбувається перевірка наявності потягів, що відправляються зі станції, з подальшим видаленням їх з масиву потягів, що знаходяться на станції за допомогою функції depart\_from\_rail\_road:



depart\_from\_rail\_road (struct Railroad\* rail\_road, train\*

del\_item)

isEmpty(rail\_road)

return NULL

end

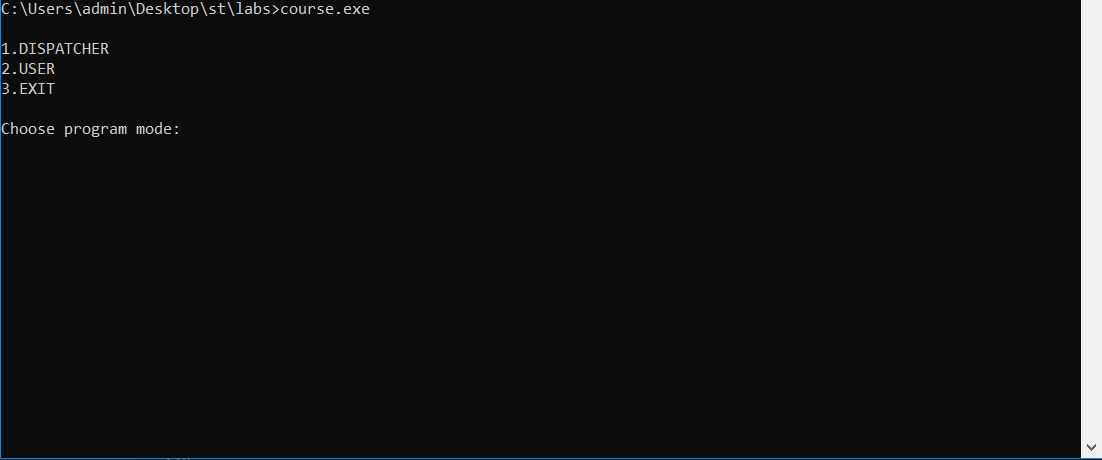
|  |  |
| --- | --- |
| return del\_item | rail\_road->size = rail\_road->size - 1; |
|  |

*isEmpty -* функція для перевірки наявності потягів на станції*, del\_item –* вказівник на структуру типу train, що використовується для збереження інформації про потяг, що відбув зі станції (логіка прибуття потягів на залізницю передбачає, що першим відбуде саме той потяг, що був прийнятим найпершим)

# РОЗДІЛ III

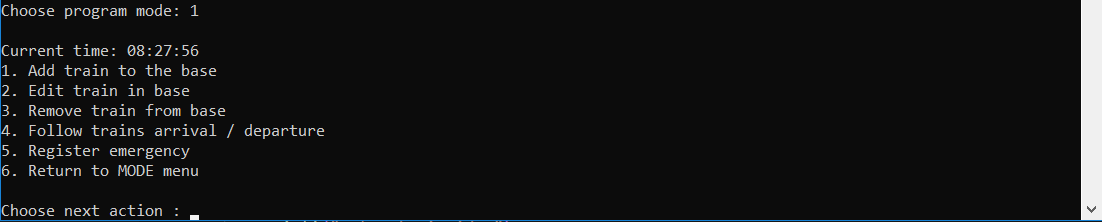
**ІНТЕРФЕЙС КОРИСТУВАЧА**

Програму розроблено для роботи у консолі операційної системи. Для її запуску необхідно скомпілювати вихідний файл з розширенням *.с* та створити виконуючий файл *.exe.* Після введення команди *<name\_of\_the\_program>.exe,* користувач потрапляє у наступне меню:



Наступні кроки здійснюються по введенні користувачем запропонованої цифри – номеру пункту у меню.

Для переходу у режим диспетчера, користувач має ввести *1:*



Меню диспетчера передбачає наступний функціонал: додати потяг, редагувати параметри потягу, видалити потяг з бази даних, слідкувати за процесом прибуття/відправлення потягів та реєструвати надзвичайні ситуації.

Натискання на *4* переведе користувача у режим слідкування за потоком потягів, що має наступний інтерфейс:



У вікні відображено потяги, що наразі знаходяться на станції та потяги, що з неї відправилися.

Реєстрація надзвичайних ситуацій передбачає введення номера потягу, з яким виникла проблема та часу затримки його прибуття:



В локальній базі даних диспетчера час прибуття потягу оновлюється з урахуванням затримки.

Після повернення до головного меню, можна обрати пасажирський режим роботи програми:



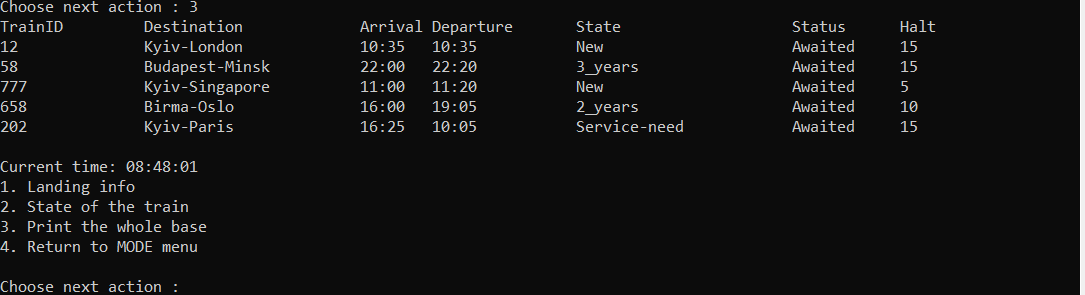
Перед користувачем декілька опцій: отримати інформацію про конкретний потяг, стан потягу чи побачити поточну базу даних потягів вокзалу.

Функція отримання інформації про посадку виглядає наступним чином:



Користувач має ввести номер потяга, що його цікавить, та отримати статус потяга стосовно знаходження на станції: *Прибув, Очікується, Відправився.*

Щоб побачити всі потяги, зареєстровані на станції, користувач може скористатися опцією *Print the whole base:*



Після цього надрукується поточна база даних потягів залізничної станції (статична, може відрізнятися від локальної бази даних, з якою працює диспетчер станції).

Для виходу з режимів диспетчера та користувача потрібно обрати пункт

*Return to MODE menu*.



Звідки, після обрання опції *EXIT*, користувач зможе зупинити виконання програми.

# ВИСНОВКИ

У ході виконання курсової роботи було досліджено алгоритми для створення автоматизованої диспетчерської системи залізничної станції. Реалізовано ключовий функціонал такої системи, що полягає в управлінні прибуттям та відправленням потягів та роботі з базою даних залізничної мережі. Додатково розроблено сервісну частину диспетчерської системи.

Було створено широку теоретичну базу різноманітних видів диспетчерських систем та їх фактичної реалізації на залізницях по всьому світу, що уміщує в собі теоретичну інформацію про математичні та програмні принципи побудови диспетчерських систем, деякі проблеми в практичній реалізації. Розібрано загальну структуру та ключові компоненти таких систем. Звідки можна зробити наступні висновки:

* ключовою компонентою сучасних залізничних мереж є комплексна інформаційна система, що обєднує мережу в одне ціле;
* ефективність вирішення конфліктів автоматизованим диспетчером обернено пропорційна кількості задіяних потягів та колій на локації;
* для вирішення конфліктів на маршрутах потрібно розглядати низку проблем теорії оптимізації, не використовуючи метод грубою сили через експоненційне зростання обчислень;
* система, що приймає рішення, має оцінювати економічну складову руху потягів та вирішувати проблеми на залізничній мережі, приймаючи до уваги наслідки зміни параметрів руху потягів.

У процесі проектування програмного забезпечення мовою C було розроблено низку функцій для управління процесом прибуття та відправлення потягів із залізничної станції, реєстрації надзвичайних випадків та роботою з базою даних потягів.

Звідки було набуто доволі великі пізнання у роботі зі структурами даних, декількома потоками програми, низкою бібліотек мови програмування C та

файлами.

Також задля роботи користувача з програмою було розроблено розширене меню, що передбачає можливість керування залізничною станцією в режимі диспетчера та отримання довідкової інформації в режимі пасажира.

Відповідно до цього було розроблену інструкцію для полегшення роботи з даної програмою. Структуру самої програми представлено у *додатку А*. Результатом роботи є програма, яку можна використовувати у навчальних цілях або на залізничних станціях для централізованого керування мережею залізниці та легкої комунікації між пасажирами та станцією.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* + 1. Bin Ning, Xuewei Li, “A comprehensive information system for railway networks,” Department of Control Engineering, August 2015, pp.1-8.
    2. M. Hafez Fahmy, H.Hemeda, “Computer applications in railway operation,” Alexandria Engineering Journal, Vol. 55, Issue 2, June 2016, Pages 1573- 1580.
    3. Petersen, C.D., “An Introduction to Computer Assisted Train Dispatch, ” Transportation Science, Vol. 3, No. 1, May 1996, pp. 85-91.
    4. Petersen, E.R. and Taylor, A.J., “A Structural Model for Rail Line Simulation and Optimization,” Transportation Science, Vol. 16, No. 2, May 1982, pp. 192-205.
    5. Petersen, E.R. and Taylor, A.J., “Line Block Prevention in Rail Line Dispatch and Simulation Models,” INFOR, Vol. 21, No. 1, February 1983, pp. 46-51.

ІНТЕРНЕТ РЕСУРСИ

* + 1. Автоматизовані системи диспетчерського контролю.<http://khartep.com/en/article/view/id/14/>
    2. Алгоритми керування потоком потягів станції.<https://link.springer.com/chapter/10.1007/978-3-319-72153-8_12>.
    3. Диспетчер залізниці. <https://en.wikipedia.org/wiki/Train_dispatcher>
    4. Програмне забезпечення диспетчерської системи залізничної станції.<https://www.sintef.no/en/software/train-dispatcher/>
    5. Системи управління залізничними мережами Європи. [https://www.mobility.siemens.com/mobility/global/en/interurban- mobility/rail-solutions/rail-automation/train-control-system/european-train- protection-system/Pages/european-train-control-system.aspx](https://www.mobility.siemens.com/mobility/global/en/interurban-mobility/rail-solutions/rail-automation/train-control-system/european-train-protection-system/Pages/european-train-control-system.aspx)

# ДОДАТКИ

*Додаток А. Структурна схема програми*

Головне меню



Режим диспетчера

Вихід

Режим пасажира

Додати потяг у базу

Інформація про посадку

Корегувати параметри потяга

Інформація про стан потягів

Видалити потяг з бази

Надрукувати базу

Повернення у головне меню

Управління потоком потягів

Реєстрація надзвичайних випадків

Повернення у головне меню