Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №11

з навчальної дисципліни

“Базові методології та технології програмування”

КОМАНДНА РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ

ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Заріцкий В. А.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

Мета роботи полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів С++ для керування динамічною пам’яттю та бінарними файловими потоками.**Варіант №2**

**Завдання до лабораторної роботи**

1. У складі команди ІТ-проєкта розробити програмні модулі оброблення динамічної структури даних.
2. Реалізувати програмний засіб на основі розроблених командою ІТ-проєкта модулів.

**Склад IT-команди**

* Заріцкий Віктор (<https://github.com/odorenskyi/Zaritskyi-Viktor-KB24>)
* Кондратенко Дмитро (<https://github.com/odorenskyi/Kondratenko-Dmytro-KB24>)
* Колесник Віктор(<https://github.com/odorenskyi/Kolesnyk-Viktor-KB24>)

# Аналіз задач ІТ‑проєкту та вимог до ПЗ

1. Завантаження бази з файлу при старті програми.
2. Відображення всієї бази на екрані або вивід у текстовий файл (за вибором користувача).
3. Додавання нового запису («особова картка»).
4. Пошук запису за прізвищем працівника.
5. Видалення запису за табельним номером.
6. Збереження бази у файл при завершенні роботи.

# Специфікації ПЗ, концептуальні рішення, архітектура

— Головний файл (main.cpp) — єдиний «оркестратор» з консольним меню.

— Три незалежні модулі (по 2 функції кожний), усі залежать тільки від спільного заголовка struct\_type\_project\_2.h.

Схема взаємодії:

main.cpp → Module\_File\_Zaritskyi

main.cpp → Module\_CRUD\_Kondratenko

main.cpp → Module\_SearchDel\_Kolesnyk

Module\_File\_Zaritskyi → struct\_type\_project\_2.h

Module\_CRUD\_Kondratenko → struct\_type\_project\_2.h

Module\_SearchDel\_Kolesnyk → struct\_type\_project\_2.h

Алгоритми викликів із main.cpp:

1. Завантажити: loadDatabase(head, filename)
2. Меню оператора: виклик addRecord, printAllRecords, findRecordByName або deleteRecord
3. На виході: saveDatabase(head, filename)

# Вибір динамічної структури даних

Тип: однозв’язний список.

Обґрунтування:

* Динамічність (будь-яка кількість записів).
* Простота вставки/видалення та проходження лінійно (O(n) — прийнятно для кількох сотень записів).

Вузол списку (Node):

struct Node {

EmployeeRecord record;

Node\* next;

Node(EmployeeRecord rec) : record(rec), next(nullptr) {}

};

Таблиця 1 – Розподіл підзадач (по модулях)

|  |  |  |
| --- | --- | --- |
| **Модуль** | **Студент** | **Функції** |
| Module\_File\_Zaritskyi | Заріцкий Віктор | loadDatabase(), saveDatabase() |
| Module\_CRUD\_Kondratenko | Кондратенко Дмитро | addRecord(), printAllRecords() |
| Module\_SearchDel\_Kolesnyk | Колесник Віктор | findRecordByName(), deleteRecord() |

Таблиця 2 – План робіт за ISO/IEC 12207

|  |  |
| --- | --- |
| **Етап** | **Відповідальні** |
| 1. Планування | Усі |
| 2. Реалізація модулів | За модулями |
| 3. Інтеграція та тестування | Усі |
| 4. Верифікація відповідності вимогам | Усі |
| 5. Валідація, демонстрація викладачу | Усі + викладач |

# Модуль Module\_File\_Zaritskyi

Модуль Module\_File\_Zaritskyi забезпечує функції для роботи з бінарним представленням бази даних у пам’яті та на диску:

* loadDatabase(Node\*& head, const std::string& filename) – завантажує список записів із бінарного файлу у зв’язаний список у пам’яті;
* saveDatabase(Node\* head, const std::string& filename) – зберігає поточний зв’язаний список у бінарний файл.

Обидві функції повертають bool, що сигналізує про успіх (true) чи невдачу (false) операції.

## Функція loadDatabase

**Призначення:**

* Ініціалізує порожній список (у разі наявності даних – звільняє пам’ять).
* Зчитує спочатку кількість записів (recordCount), потім кожен запис у цикл­у.
* Побудоває нові вузли Node із відновленими EmployeeRecord і ланцюжить їх у список.

**Особливості:**

* Перед зчитуванням старий список звільняється, щоб уникнути витоку пам’яті.
* Після читання кожного блоку перевіряється file.fail(), і в разі помилки відбувається очищення вже створених вузлів.

## Функція saveDatabase

**Призначення:**

* Обчислює кількість записів у списку (recordCount).
* Записує спочатку recordCount, потім послідовно бінарізує поля кожного EmployeeRecord:
  + прості типи (Date, int, short int, enum) через reinterpret\_cast,
  + строки — через запис size\_t len + c\_str() з довжиною .

**Особливості:**

* Прямий бінарний дамп структур, що дає високу швидкість.
* Після успіху виводить кількість збережених записів — зручно для відлагодження.

## Обробка помилок

* Обидві функції повертають false і виводять діагностичне повідомлення через std::cout, якщо не вдається відкрити файл чи виникають помилки введення/виведення.
* При критичній помилці в loadDatabase модуль очищує вже зчитані дані, щоб не залишати неповний список.

* 

