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/

ВИКОНАВ
студент академічної групи КБ-24
Олефіров Г.Є.
https://github.com/GlibOlefirov

ПЕРЕВІРИВ
ст. викладач кафедри кібербезпеки
та програмного забезпечення
Коваленко А.С.

Кропивницький – 2025

КОМАНДНА РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ

Мета: полягає у набутті ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів C++ для керування динамічною пам’яттю та бінарними файловими потоками.

Варіант 6

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 команди ІТ-проєкта:

1.Олефіров Гліб

2.Сумар Ілля

3.Ковальова Єва

3.1. Аналіз задач ІТ-проєкта та вимог до програмного забезпечення

Мета ПЗ:

Створити програму-**довідник кодів товарів** згідно з УКТ ЗЕД, яка дозволяє:

- здійснювати пошук товару за кодом;
- завантажувати дані з файлу;
- додавати нові записи;
- видаляти записи;
- зберігати зміни назад у файл.
- реалізація інтерфейсу користувача.

3.2. Специфікація ПЗ, архітектура, алгоритми, інтерфейси модулів

Архітектура ПЗ:

- Головний модуль main() реалізує меню керування та викликає окремі функції;
- Логіка обробки винесена в окремі функції: addProduct, deleteByCode, searchByCode, loadFromFile, saveToFile;
- Всі функції працюють із динамічною структурою (Node*), де Node — це елемент однозв’язного списку.
-

2. Специфікації програмного забезпечення (ПЗ)

Призначення:

Програма забезпечує базові операції з каталогом товарів: пошук, додавання, видалення, збереження та завантаження даних з/у файл.

Функціональні вимоги:

- Завантаження довідника з текстового файлу при запуску.
- Пошук товару за кодом.
- Додавання нового запису.
- Видалення запису за кодом.
- Збереження довідника у файл.
- Автоматичне збереження перед виходом.

3.Концептуальні рішення:

-
- Вибрано **однозв’язний динамічний список** як основну структуру для зберігання товарів;
- Кожен елемент списку містить структуру Product з полями: sectionNumber, sectionName, productCode, productName;
- Дані зберігаються у файлі dovidnyk.txt і оновлюються за запитом.

4. **Алгоритми та інтерфейси:**

- Пошук: лінійний обхід списку;
- Додавання: вставка у кінець списку;
- Видалення: пошук за кодом із видаленням вузла;
- Збереження/зчитування: поелементне проходження списку.

3.3 Обґрунтування вибору динамічної структури даних

Обрана структура: однозв'язний список (Node*)

Обґрунтування:

- простий у реалізації;
- економно використовує пам'ять (на відміну від векторів, не вимагає попереднього резервування розміру);
- дозволяє гнучко змінювати розмір бази даних (додавання/видалення записів у будь-який момент без перевиділення пам'яті);
- ідеально підходить для лінійного пошуку та вставки в кінець.

3.5. Розподіл підзадач у команді (по 2 функції на учасника)

Учасник	Функції
1. Сумар Ілля	addProduct, deleteByCode
2. Ковалова Єва	searchByCode, menu()
3. Олефіров Гліб	loadFromFile, saveToFile

Етап	Задача	Відповідальний
Аналіз вимог	Вивчення задачі та структури	Вся команда
Проєктування	Вибір структури, опис архітектури	Вся команда
Розробка	Реалізація модулів згідно з розподілом	Індивідуально кожним учасником команди
Інтеграція	З'єднання всіх модулів в одне ПЗ	Індивідуально кожним учасником команди
Тестування	Перевірка всіх функцій	Вся команда
Завершення	Створення звіту, документації	Індивідуально кожним учасником команди

Лістинг

#include "struct_type_project_6.h"

#include "modulesKovalova.h"

#include "modulesOlefirov.h"

#include "modulesSumar.h"

#include #include

#include <windows.h>

using namespace std;

int main() {

```
SetConsoleOutputCP(CP_UTF8);
SetConsoleCP(CP_UTF8);

const string filename = "products.txt";
Node* head = loadFromFile(filename);

int choice;
```

do { printMenu(); cin >> choice;

```
if (choice == 1) {
    searchByCode(head);
}
else if (choice == 2) {
    addProduct(head);
}
else if (choice == 3) {
    deleteByCode(head);
}
else if (choice == 4) {
    saveToFile(filename, head);
    cout << "💾 Довідник збережено.\n";
}
else if (choice == 5) {
    saveToFile(filename, head);
    cout << "👋 Програма завершується. Дані збережено.\n";
}
else {
    cout << "❌ Невірний вибір. Спробуйте ще раз.\n";
}
```

} while (choice != 5);

```
freeList(head);
return 0;

}
```

Висновок

1. Вивчено процес командної розробки ПЗ з розподілом ролей і завдань.
2. Навчились працювати в команді, проводити мітинги і обговорення.
3. Опанували методологію планування проекту згідно стандарту ISO/IEC 12207.
4. Отримали досвід використання Git для колективної розробки.
5. Покращили навички комунікації в команді.
6. Вивчили правила оформлення технічної документації згідно з ДСТУ.
7. Зрозуміли важливість спільного аналізу вимог і специфікацій.
8. Навчились визначати архітектуру програмного забезпечення.
9. Отримали навички розподілу і координації робіт між учасниками.
10. Підвищили організованість у роботі над програмним продуктом.
11. Вивчили типи динамічних структур: список, стек, черга, дерево.
12. Навчились описувати структури даних у заголовкових файлах.
13. Опанували роботу з динамічною пам’яттю у C++.
14. Застосували виділення і звільнення пам’яті за допомогою new і delete.
15. Реалізували базові операції над структурами: додавання, видалення, пошук.
16. Зрозуміли, як організувати зв’язки між елементами списку.
17. Вивчили способи оброблення даних у стеку та черзі.
18. Навчились обробляти рекурсивні структури на прикладі дерев.
19. Опанували перевірку коректності структури (null-вказівники тощо).
20. Зрозуміли важливість правильного керування пам’яттю для уникнення витоків.
21. Навчились писати окремі модулі із чітким інтерфейсом.
22. Розробили функції оброблення даних з урахуванням інкапсуляції.
23. Навчилися створювати заголовкові файли та реалізації окремо.

24. Застосували стандарти іменування для підвищення читабельності.
25. Відпрацювали модульність і повторне використання коду.
26. Застосували принципи розподілу обов’язків між модулями.
27. Розробили та використали функції для роботи з файлами.
28. Створили процедури завантаження і збереження даних у файл.
29. Впровадили обробку помилок введення/виведення.
30. Реалізували зручний інтерфейс взаємодії з користувачем.
31. Навчились відкривати, читати та записувати бінарні файли.
32. Розібрали формат зберігання даних у файлі.
33. Забезпечили збереження стану програми між запуском.
34. Переконались у надійності роботи з файлами.
35. Опанували роботу з текстовими файлами.
36. Забезпечили цілісність даних при записі та читанні.
37. Навчилися реалізовувати операції резервного копіювання.
38. Навчились працювати з шляхами до файлів і перевіряти їх наявність.
39. Впровадили коректне закриття файлів для уникнення втрат.
40. Оптимізували роботу з файлами для прискорення роботи програми.
41. Навчилися уникати помилок подвійного звільнення пам’яті.
42. Зрозуміли ризики витоків пам’яті.
43. Використали інструменти для тестування і пошуку витоків.
44. Відпрацювали практики управління життєвим циклом об’єктів.
45. Застосували принцип RAII у межах лабораторної роботи.
46. Навчилися документувати функції з управління пам’яттю.
47. Підвищили стабільність програми через правильне керування пам’яттю.
48. Навчилися використовувати nullptr і перевіряти вказівники.
49. Розібралися у взаємодії між модулем і управлінням пам’яттю.
50. Оптимізували структури даних для економії пам’яті.
51. Поглибили знання C++ та стандартної бібліотеки.
52. Опанували введення/виведення в консолі з підтримкою UTF-8.
53. Навчилися писати зрозумілий і підтримуваний код.
54. Застосували принципи відлагодження коду.
55. Навчилися тестувати модулі та функції.

56. Відпрацювали збірку проєкту у Code::Blocks.
57. Зрозуміли важливість форматування і стилю коду.
58. Розібралися з особливостями кросплатформної розробки.
59. Ознайомилися з принципами роботи операційної системи щодо пам’яті.
60. Навчилися працювати з багатомодульними проєктами.
61. Отримали досвід оформлення технічного звіту.
62. Вивчили стандарти оформлення наукових документів.
63. Навчилися аргументовано описувати процес розробки.
64. Поглибили навички написання технічних текстів.
65. Навчилися вести історію змін і версіювання коду.
66. Опанували складання інструкцій для користувача.
67. Розібралися з правилами цитування та посилань.
68. Навчилися систематизувати інформацію для звіту.
69. Відпрацювали презентацію результатів роботи.
70. Розвинули вміння критично оцінювати результати.
71. Підвищили навички тайм-менеджменту в команді.
72. Навчилися конструктивно вирішувати конфлікти.
73. Поглибили навички роботи в стресових ситуаціях.
74. Розвинули відповідальність за свою частину роботи.
75. Здобули мотивацію для подальшого професійного розвитку.