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”

На тему:

“ Реалізація програмних модулів оброблення динамічних структур даних та бінарних файлів”

ВИКОНАВ

студент академічної групи

КБ-24

Сумар Ілля

ПЕРЕВІРИВ

викладач Коваленко А. С.

м. 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.

**Завдання :**

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

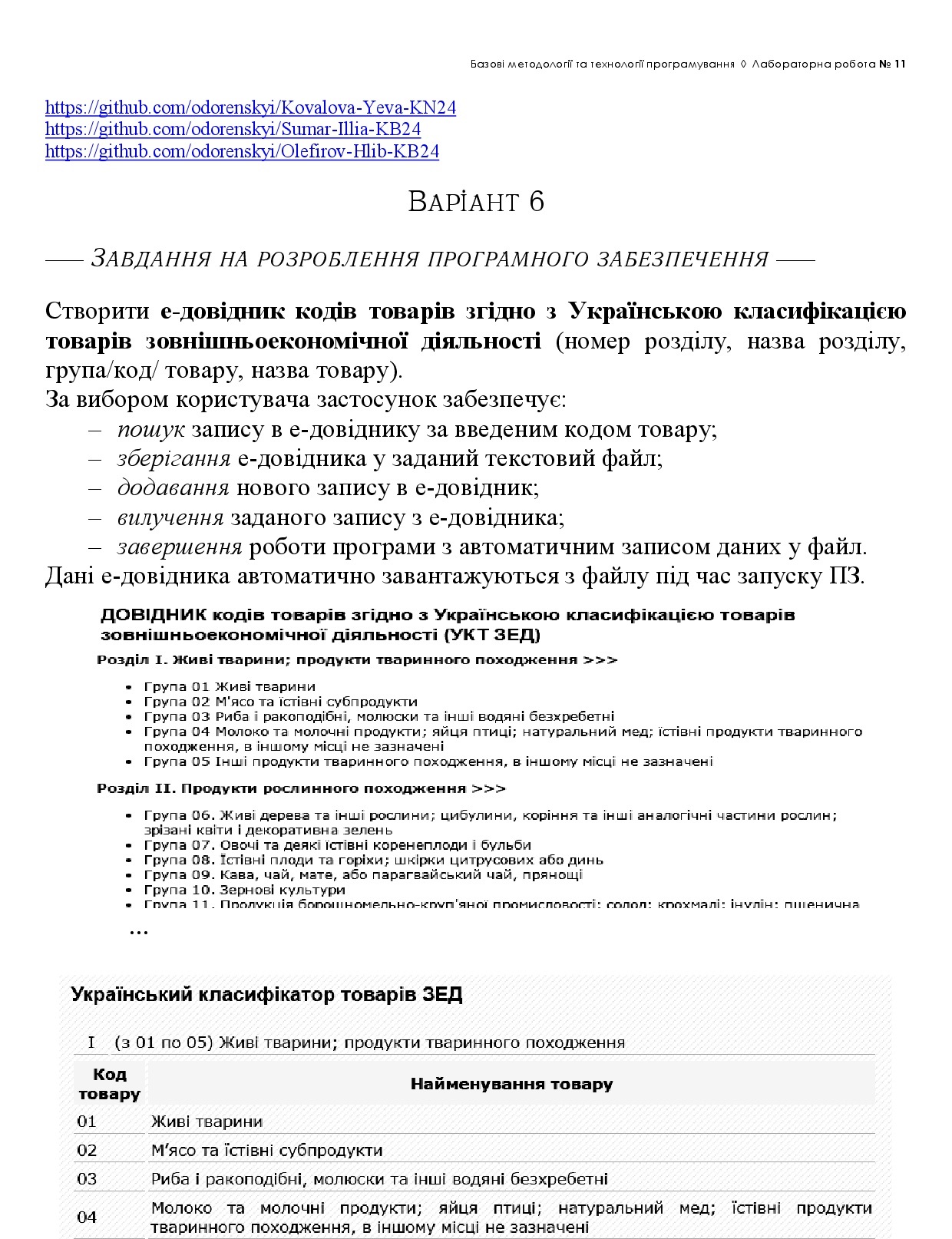
**Склад команди IT-проєкта:**

1. Сумар Ілля
2. Олефіров Гліб
3. Ковальова Єва

**План роботи з виконання IT-проєкта:**

1. Завантажити Git-репозиторій - https://github.com/odorenskyi/student-name;
2. В файл README.md, записати: тему, мету, завдання та варіант.
3. Створити теки Report, prj, Software, TestSuite.
4. В текі \lab11\Report створити Документ Microsoft Word «Звіт до Лабораторної роботи № 11», в ньому написати титульну сторінку, тему, мету, завдання, варіант, склад команди.
5. Розробити детальний аналіз отриманих півзадач та проектування. Для розроблення даних півзадач потрібно використати створений на meeting, заголовковий файл struct\_type\_project\_6.h.
6. Розробити свої дві підзадачі та записати їх в модуль «ModulesПрізвище» підключивши заголовковий файл struct\_type\_project\_2.h, вміст файлу включити до звіту.
7. Завантажити фа**Не знайдено елементи покажчика.**йли на GitHub.
8. Коли всі учасники розроблять свої модулі та проведуть модульні тестування, при успішному тестування, завантажте Git-репозиторії інших учасників.
9. В Code::Blocks IDE створити проект консольного додатка prj\_2\_Прізвище.
10. Відповідно до стандарту ISO/IEC 12207 реалізувати програмний засіб згідно з завданням, використовуючи функції (модулі), розроблені учасниками команди ІТ-проекта.
11. Провести системне тестування консольного додатка prj\_2\_Прізвище та свого модуля. При успішному тестування переходи на наступний крок, якщо результат негативний повернутися до додатка, та виправити помилки.
12. Підготувати й зберегти у \Lab11\Report звіт про виконання лабораторної роботи, оформлений згідно з ДСТУ 3008:2015 “Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання”, зі змістом, визначеним цим порядком виконання лабораторної роботи.
13. Представити до захисту звіт з виконаної лабораторної роботи і проект у Git-репозиторії https://github.com/odorenskyi/student\_name.

**Завдання:**



**Аналіз та постановка задачі:**

**Аналіз виконання Task 11:**

На мітингу, який відбувся в Discord, ми з командою розділили завдання для кожного учасника із команди.

|  |  |
| --- | --- |
| Учасник команди | Функції для реалізації |
| 1. Сумар Ілля | addProduct(), deleteByCode() |
| 2. Ковалова Єва | searchByCode, saveToFile |
| 3. Олефіров Гліб | |  | | --- | |  |  |  | | --- | | loadFromFile, ShowMenu() | |

Я реалізовував функції для додавання так видалення продукту. Потрібно щоб при додаванні запису користувач вводив параметри з клавіатури. І повідомляти користувача про успішну додавання або видалення продукту. При видаленні продукту, треба щоб користувач вводив код продукту і якщо такий код є, то вивести що продукт видалено, інакше запис не знайдено.

**Постановка виконання Task 11:**

Вхідні дані:

Функція для додавання продукту:

Дані що вводить користувач:

* Номер розділу
* Назва розділу
* Код товару
* Назва товару

Функція для видалення запису:

Код товару

Вихідні дані:

Функція для додавання запису:

Повідомлення про те що запис додано

Функція для видалення запису:

Повідомлення про видалення запису або що його не знайдено

**Висновок:**

Під час виконання лабораторної роботи № 11 на тему **«**Реалізація програмних модулів оброблення динамічних структур даних та бінарних файлів**»** з предмету «Базові Методології та Технології Програмування» я відпрацював на практиці роботу з бінарними файлами та динамічними структурами даних. Я набув 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.

Під час мітингу ми вирішили використати клас Node замість простої структури, оскільки це дає нам більше гнучкості, зручності та відповідає принципам об'єктно-орієнтованого програмування (ООП) основні причини:

**1. Розширюваність:**  
Клас легко розширити — наприклад, ми можемо додати методи виводу, копіювання, перевірки або збереження даних прямо в Node, не змінюючи логіку програми.

**2. Відповідність навчальній меті:**  
Ми вивчаємо об'єктно-орієнтоване програмування, і використання класу Node допомагає закріпити ці знання на практиці.

**Вихідний код заголовкового файлу struct\_type\_project\_6:**

#ifndef STRUCT\_TYPE\_PROJECT\_6\_H\_INCLUDED

#define STRUCT\_TYPE\_PROJECT\_6\_H\_INCLUDED

#include <string>

using namespace std;

struct Product {

    string sectionNumber;

    string sectionName;

    string productCode;

    string productName;

};

struct Node {

    Product data;

    Node\* next;

};

#endif // STRUCT\_TYPE\_PROJECT\_6\_H\_INCLUDED

**Вихідний код модуля ModulesSumar:**

#include "struct\_type\_project\_6.h"

#include <iostream>

#include <algorithm>

using namespace std;

bool isNumber(const string& str) {

    return !str.empty() && all\_of(str.begin(), str.end(), ::isdigit);

}

bool isAlphabetic(const string& str) {

    return !str.empty();

}

void addProduct(Node\*& head) {

    Product p;

    cin.ignore();

    cout << "\n➕ Додавання нового запису:\n";

    while (true) {

        cout << "Номер розділу (тільки цифри): ";

        getline(cin, p.sectionNumber);

        if (isNumber(p.sectionNumber)) break;

        cout << "❌ Помилка! Введіть лише цифри.\n";

    }

    while (true) {

        cout << "Назва розділу (тільки літери): ";

        getline(cin, p.sectionName);

        if (isAlphabetic(p.sectionName)) break;

        cout << "❌ Помилка! Введіть лише літери.\n";

    }

    while (true) {

        cout << "Код товару (тільки цифри): ";

        getline(cin, p.productCode);

        if (isNumber(p.productCode)) break;

        cout << "❌ Помилка! Введіть лише цифри.\n";

    }

    while (true) {

        cout << "Назва товару (тільки літери): ";

        getline(cin, p.productName);

        if (isAlphabetic(p.productName)) break;

        cout << "❌ Помилка! Введіть лише літери.\n";

    }

    Node\* newNode = new Node{p, nullptr};

    if (!head) {

        head = newNode;

    } else {

        Node\* current = head;

        while (current->next) current = current->next;

        current->next = newNode;

    }

    cout << "✔️ Запис додано.\n";

}

void deleteByCode(Node\*& head) {

    string code;

    cout << "\n🗑️ Введіть код товару для видалення: ";

    cin >> code;

    Node\* current = head;

    Node\* prev = nullptr;

    while (current) {

        if (current->data.productCode == code) {

            if (!prev) {

                head = current->next;

            } else {

                prev->next = current->next;

            }

            delete current;

            cout << "✅ Запис видалено.\n";

            return;

        }

        prev = current;

        current = current->next;

    }

    cout << "❌ Запис не знайдено.\n";

}

void freeList(Node\* head) {

    while (head) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

}

**Вихідний код заголовкового файлу ModulesSumar:**

#ifndef SUMAR\_H\_INCLUDED

#define SUMAR\_H\_INCLUDED

#include "struct\_type\_project\_6.h"

bool isNumber(const std::string& str);

bool isAlphabetic(const std::string& str);

void addProduct(Node\*& head);

void deleteByCode(Node\*& head);

void freeList(Node\* head);

#endif // SUMAR\_H\_INCLUDED

**Вихідний код prj\_6\_Sumar**

#include "struct\_type\_project\_6.h"

#include "modulesKovalova.h"

#include "modulesOlefirov.h"

#include "modulesSumar.h"

#include <iostream>

#include <cstdlib>

#include <windows.h>

// Прототипи функцій з модулів

using namespace std;

int main() {

     SetConsoleOutputCP(CP\_UTF8);

    SetConsoleCP(CP\_UTF8);

    setlocale(LC\_ALL, "uk\_UA.UTF-8");

    const string filename = "products.txt";

    Node\* head = loadFromFile(filename);

    int choice;

    do {

        printMenu();

        cin >> choice;

        cin.ignore(); // Очистити буфер після введення числа

        switch (choice) {

            case 1:

                searchByCode(head);

                break;

            case 2:

                addProduct(head);

                break;

            case 3:

                deleteByCode(head);

                break;

            case 4:

                saveToFile(filename, head);

                cout << "💾 Довідник збережено.\n";

                break;

            case 5:

                saveToFile(filename, head);

                cout << "👋 Програма завершується. Дані збережено.\n";

                break;

            default:

                cout << "❌ Невірний вибір. Спробуйте ще раз.\n";

        }

    } while (choice != 5);

    freeList(head);

    return 0;

}

**Аргументи виконання лабораторної роботи:**

1. **Засвоєння структури пов’язаних списків.**
2. **Навички роботи з динамічною пам’яттю (new/delete).**
3. **Розуміння важливості перевірки коректності введення.**
4. **Робота з текстовими файлами (файловий ввід/вивід).**
5. **Створення модульної структури програми.**
6. **Ознайомлення з побудовою меню і обробкою вибору користувача.**
7. **Практика написання функцій і розподілення коду на частини.**
8. **Використання функцій перевірки типу введених даних.**
9. **Реалізація простого, але корисного додатку.**
10. **Практика проектування алгоритмів пошуку, додавання та видалення.**
11. **Можливість перетворити структури на класи.**
12. **Вивчення принципів інкапсуляції.**
13. **Підготовка до використання наслідування.**
14. **Уміння створювати методи всередині класів.**
15. **Створення приватних/публічних полів.**
16. **Створення конструкторів і деструкторів.**
17. **Реалізація get/set методів.**
18. **Побудова гнучкої структури даних.**
19. **Розмежування відповідальності між об’єктами.**
20. **Основи побудови абстрактних типів даних.**
21. **Навчання роботі з багатьма файлами у проєкті.**
22. **Вміння дотримуватись принципу DRY (Don’t Repeat Yourself).**
23. **Логічне розділення коду на модулі.**
24. **Правильне використання заголовкових файлів.**
25. **Формування навички коментування коду.**
26. **Створення масштабованого коду.**
27. **Використання захисту від подвійного включення (include guard).**
28. **Формування звички до чистого та організованого коду.**
29. **Покращення навичок компіляції багатофайлових проєктів.**
30. **Розуміння взаємозв’язків між модулями.**
31. **Відповідність реальним завданням у розробці ПЗ.**
32. **Розвиток аналітичного мислення.**
33. **Уміння писати зручні для користувача інтерфейси в консолі.**
34. **Навички роботи з меню та навігацією в програмі.**
35. **Покращення навичок відлагодження програм.**
36. **Підготовка до автоматизації завдань.**
37. **Розуміння, як зберігати та обробляти велику кількість даних.**
38. **Уміння реалізовувати пошук у структурі даних.**
39. **Звичка писати логічно послідовний код.**
40. **Покращення дисципліни в написанні функціональних блоків.**
41. **Звітна робота для оцінки знань.**
42. **Підготовка до іспиту або заліку.**
43. **Закріплення теми “однозв'язні списки”.**
44. **Демонстрація знань викладачеві.**
45. **Можливість отримати вищу оцінку.**
46. **Приклад для портфоліо студента.**
47. **Виконання частини курсової або дипломної роботи.**
48. **Власний проект, який можна вдосконалювати.**
49. **Практика на реальному прикладі (товари, коди, довідники).**
50. **Відчуття завершеного корисного проєкту.**
51. **Перший крок до розробки баз даних.**
52. **Підготовка до вивчення шаблонів STL (наприклад, std::list).**
53. **Уміння працювати з текстовими форматами даних.**
54. **Навички модульного тестування.**
55. **Підвищення впевненості у використанні C++.**
56. **Побудова мислення програміста.**
57. **Засвоєння підходів рефакторингу коду.**
58. **Уміння працювати з помилками компіляції.**
59. **Вивчення способів оптимізації простих алгоритмів.**
60. **Усвідомлення зв’язку між теорією і практикою.**
61. **Розуміння, як працює обробка даних на низькому рівні.**
62. **Уміння перевіряти правильність даних (валідація).**
63. **Досвід роботи з потенційно уразливою пам’яттю (heap).**
64. **Можливість перевірити, як обробляються помилки.**
65. **Основи захисту даних у структурах.**
66. **Побудова надійної обробки вводу користувача.**
67. **Формування підходу до побудови безпечного ПЗ.**
68. **Навички безпечного зчитування з файлів.**
69. **Осмислення ризиків неправильного видалення об’єктів.**
70. **Підготовка до тем: буферні переповнення, витоки пам’яті.**
71. **Задоволення від роботи, яка реально працює.**
72. **Розвиток самостійності та ініціативи.**
73. **Поштовх до глибшого вивчення C++.**
74. **Можливість пишатися своїм результатом.**
75. **Крок до наступного рівня програмування (ООП, STL, GUI тощо).**

**Відповіді на контрольні запитання:**

**1. Що відповідно до міжнародного стандарту ISO/IEC 12207 розуміється під процесом комплексування (інтегрування) ПЗ?**

**Комплексування (інтегрування) — це процес збирання окремих модулів (компонентів) програмного забезпечення в одне ціле.**

**Мета — забезпечити взаємодію компонентів та перевірити правильність їхньої роботи разом. Приклад як ми підключали наші модулі разом.**

**2. Виконайте порівняльний аналіз вказівника та посилання С++**

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Вказівник (pointer)** | **Посилання (reference)** |
| Синтаксис | int\* ptr = &x; | int& ref = x; |
| Можна змінювати адресу? | Так | Ні (після ініціалізації фіксована) |
| Може бути null? | Так (nullptr) | Ні |
| Вимагає ініціалізації? | Не обов’язково | Так |
| Зручність у використанні | Менш безпечно, але гнучко | Простіше та безпечніше |

**3. Перелічіть допустимі операції над вказівниками С++**

1. **Присвоєння адреси (int\* p = &x;)**
2. **Арифметика вказівників (p++, p--, p + i)**
3. **Зрівняння (if (p == q))**
4. **Розіменування (\*p)**
5. **Присвоєння іншого вказівника (p = q)**
6. **Перевірка на nullptr**
7. **Використання з масивами (\*(arr + i))**

**4. Яке призначення операції опосередкованої адресації та який синтаксис її запису мовою програмування С++?**

**Призначення** — звернення до значення за вказаною адресою.  
 **Синтаксис:**

int x = 5;

int\* ptr = &x;

std::cout << \*ptr; // опосередкована адресація: отримаємо 5

**\*ptr** — це операція **розіменування**, вона дозволяє читати або змінювати значення, яке зберігається за вказаною адресою.

**5. Які функції-члени об’єктів fstream C++ забезпечують відкриття потоку, запису даних у потік та читання з нього?**

**Відкриття файлу:**  
 file.open("data.txt", ios::in | ios::out);

**Запис у файл:**  
 file << "Hello"; або file.write(buf, size);

**Читання з файлу:**  
 file >> str; або file.read(buf, size);

Стандартні класи:

* ifstream – читання
* ofstream – запис
* fstream – і те, і те

6. Які члени (поля) елемента динамічної структури підлягають зберіганню у файл, а які ні? Відповідь обґрунтуйте.

**Зберігаються:** всі дані користувача, наприклад поля типу string, int, float у структурі Product.

**Не зберігаються**: службові поля, як-от Node\* next; — вони вказують на інші об'єкти в пам'яті, але не мають змісту у файлі (бо адреси в оперативній пам'яті недійсні після перезапуску).

**Обґрунтування**: Файл повинен містити лише корисні дані, які реально потрібно зберегти між запусками.

**7. Чим бінарний файловий потік відрізняється від текстового з погляду читання/запису інформації?**

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Текстовий потік** | **Бінарний потік** |
| Формат зберігання | Людиночитний текст | Сирі байти |
| Обробка символів | Автоматично перетворює символи (переноси рядків) | Не змінює жодні байти |
| Розмір | Зазвичай більший (через форматування) | Менший і точний |
| Придатність для | Логів, конфігурацій, CSV, TXT | Зображень, структур даних, архівів |
| Методи читання | <<, >>, getline | read(), write() |