Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Базові методології та технології програмування

Лабораторна робота №11

Тема: «КОМАНДНА РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: викладач
Коваленко А.С.

Кропивницький 2025

Варіант - 4

Мета роботи - 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. ..

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

Група: КН-24

1. Куріщенко Павло;

Підзадачі:

1) Виведення всієї бази на екран або у текстовий файл (на вибір користувача);

2) Пошук запису за введеним диспетчером прізвищем студента.

2. Булюкін Володимир;

Підзадачі:

1) Завантаження бази з текстового файлу;

2) Завершення роботи програми з автоматичним записом бази у файл.

3. Радомська Діана.

Підзадачі:

1) Додавання нового запису в базу

2) Видалення заданого оператором запису з бази

Аналіз задач ІТ-проєкта та вимог до ПЗ:

Функціональні вимоги:

1. **Виведення всієї бази** (Виводити на екран або зберігати у текстовий файл).
2. **Додавання записів** (Інтерактивне введення нових студентів у базу).
3. **Пошук:** (Пошук записів за прізвищем).
4. **Видалення** (Видалення обраного запису оператором).
5. **Автоматичне збереження** (Збереження бази у файл при завершенні роботи).
6. **Автоматичне завантаження** (Читання бази з файлу при старті програми).

Формати вводу/виводу:

Ввід: із клавіатури.

Вивід: у консоль або текстовий файл.

Обраний вид динамічної структури (однозв'язний список):

Для реалізації бази даних «Деканат: облік студентів» обрано **однозв'язний список**, оскільки він:

- дозволяє **динамічно змінювати розмір** бази без попереднього резервування пам'яті;
- забезпечує **швидке додавання, видалення та перегляд записів**;
- простий у реалізації та зручний для **лінійного пошуку за прізвищем**, що повністю відповідає вимогам завдання.

Інші структури (дерева, стек, черга) або складніші у реалізації, або обмежують доступ до даних.

Обрані типи:

- string — для зберігання текстових даних (ПІБ, громадянство, адреса тощо);

- Date — уніфікований тип для дат (дата народження, дата заповнення, звільнення).

План виконання ІТ-проєкта:

Етап	Хто виконує
Підготовка і узгодження ідеї	Вся команда
Написання своїх частин коду	Кожен за своїми підзадачами
Збирання всього в одне ціле	Разом
Перевірка: чи все працює як треба	Вся команда
Презентація проєкта викладачу	Вся команда та викладач

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Опис модуля(Modules_Kurishchenko_ShowAndSearch**): Пошук і виведення бази даних:**

Модуль містить функції для пошуку студентів за прізвищем та для виведення всієї бази даних на екран чи в файл.

Основні функції:

1) printAllRecords()

Призначення:

Виводить всю базу даних студентів або на екран, або у файл students_output.txt за вибором користувача.

Основні кроки виконання:

1. Пропонує користувачу вибрати, куди виводити дані: 1 — на екран, 2 — у файл.

2. Перевіряє коректність вибору (обробляє помилки введення).

3. Якщо обрано файл — відкриває students_output.txt у режимі перезапису.

4. Проходить по всьому зв’язаному списку head, для кожного вузла:

- Виводить заголовок СТУДЕНТ №N.

- Використовує функцію printRecord() для друку детальної інформації про студента.

5. Якщо вивід був у файл — закриває файл і повідомляє користувача про успішне збереження.

Особливості реалізації:

- Завдяки параметру ostream* out функція printRecord() універсальна — вона може виводити як у консоль (cout), так і у файл.
- Виведення форматоване з рамками та заголовками для кожного студента.

Лістинг функції:

```
void printAllRecords() {
    cout << "Куди бажаєте вивести базу даних?\n"
         << "1. На екран\n"
         << "2. У файл (students_output.txt)\n"
         << "Ваш вибір: ";

    int choice;
    cin >> choice;
    if (cin.fail() || (choice > 2 || choice < 1)) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cerr << "Некоректне введення. Введіть 1 або 2.\n\n";
        return;
    }
    cin.ignore();

    ostream* out = nullptr;
    ofstream outFile;

    if (choice == 2) {
        outFile.open("students_output.txt", ios::trunc);
        if (!outFile.is_open()) {
            cerr << "Не вдалося відкрити файл для запису.\n";
            return;
        }
```

```
        out = &outFile;
    }
    else out = &cout;

    Node* current = head;
    int index = 1;

    while (current) {
        const Student& s = current->data;
        *out << "+-------------------------------------------------
----------+\n";
        *out << "                              СТУДЕНТ №" << index++ <<
"\n";
        *out << "+-------------------------------------------------
----------+";
        printRecord(out, s);
        current = current->next;
    }

    if (choice == 2) {
        outFile.close();
        cout << "Дані успішно збережено у файл
'students_output.txt'\n\n";
    }
}
```

2) **searchRecordByLastName()**

Призначення:

Здійснює пошук у базі студентів за прізвищем і виводить детальну інформацію про всі збіги.

Основні кроки виконання:

1. Запитує у користувача прізвище студента.
2. Проходить по зв’язаному списку head.
3. Для кожного елемента перевіряє, чи збігається поле lastName.
4. Якщо знаходить співпадіння:

- Виводить заголовок ІНФОРМАЦІЯ ПРО СТУДЕНТА.

- Використовує функцію printRecord() для форматованого виводу.

5. Якщо жодного збігу не знайдено — повідомляє, що студент не знайдений.

Особливості реалізації:

- Збережено чітке розділення логіки: основна функція лише викликає printRecord() для виводу, не дублюючи форматування.
- Вивід одразу в консоль, без варіанту з файлом.

Лістинг функції:

```
void searchRecordByLastName() {
    string lastName;
    cout << "Введіть прізвище студента для пошуку: ";
    getline(cin, lastName);

    Node* current = head;
    bool found = false;

    while (current != nullptr) {
        if (current->data.name.lastName == lastName) {
            found = true;
            const Student& s = current->data;

            cout << "\n+-----------------------------------------
----------------+\n";
            cout <<   "|                   ІНФОРМАЦІЯ ПРО СТУДЕНТА
|\n";
            cout <<   "+-----------------------------------------
----------------+";
            printRecord(&cout, s);
        }
        current = current->next;
    }

    if (!found) {
```

```
        cout << "Студент з прізвищем \"" << lastName << "\" не
знайдений.\n\n";
    }
}
```

Лістинг .h файлу визначення структур:

```
#ifndef STRUCT_TYPE_PROJECT_4_H #define STRUCT_TYPE_PROJECT_4_H
#include
using namespace std;
struct DepartmentInfo {
string institute;
string faculty;
string department;
};
struct CodeName {
string code;
string name;
};
struct EducationInfo {
string institutionName;
DepartmentInfo department;
string educationLevel;
CodeName trainingDirection;
CodeName specialty;
CodeName specialization;
};
struct FullName {
string lastName;
string firstName;
string middleName;
};
struct Date {
int day;
int month;
int year;
};
struct Address {
```

```
string postalCode;
string region;
string district;
string locality;
};
struct Student {
FullName name;
Date birthDate;
Address birthPlace;
string citizenship;
string graduatedFrom;
string graduationYear;
string familyStatus;
Address address;
EducationInfo education;
};
struct Node {
Student data;
Node* next;
Node(const Student& studentData) : data(studentData), next(nullptr)
{}
};
#endif // STRUCT_TYPE_PROJECT_4_H
```

Лістинг main.cpp файлу:

```
#include "interface.h"

int main() {
    system("chcp 65001 > nul");
    handleUserChoice(); // Основна функція для вибору операцій
    return 0;
}
```

Висновок

У процесі виконання лабораторної роботи №11 я здобув важливий практичний досвід у командній розробці програмного забезпечення з

використанням динамічних структур даних та бінарних файлів у середовищі C++. Робота вимагала глибокого аналізу, планування, колективної взаємодії та технічної реалізації. Нижче наведено перелік знань і навичок, яких я набув:

1. Навчився працювати в складі IT-команди над спільним проєктом.
2. Здобув досвід участі в технічних мітингах (обговорення задач і планування).
3. Вперше самостійно обґрунтував вибір структури даних для зберігання інформації.
4. Засвоїв принципи роботи з динамічною пам’яттю в C++.
5. Реалізував структури типу «список» для зберігання інформації про студентів.
6. Створив власний заголовковий файл з описом структури.
7. Використав функції для додавання, пошуку, видалення та збереження даних.
8. Працював із вказівниками та розумінням адресної арифметики.
9. Зрозумів різницю між вказівником і посиланням у C++.
10. Навчився відкривати, зчитувати і записувати бінарні файли.
11. Застосував стандартні засоби керування файлами (fstream).
12. Зрозумів відмінності між бінарними та текстовими потоками.
13. Навчився зберігати лише необхідні дані у файлі, виключаючи службові поля.
14. Дослідив вимоги ISO/IEC 12207 до процесу розробки ПЗ.
15. Навчився створювати план реалізації проєкту.
16. Реалізував програму, яка автоматично завантажує та зберігає базу.
17. Вдосконалив навички написання модульних функцій.
18. Розробив функцію виведення бази даних у файл.
19. Навчився формувати меню з різними діями користувача.
20. Здійснив тестування роботи своєї частини проєкту.
21. Сформував трасувальну таблицю для перевірки логіки роботи.

22.Застосував Git для синхронізації командної роботи.

23.Отримав досвід компіляції та лінкування з бібліотеками колег.

24.Дотримувався принципів інкапсуляції даних.

25.Зрозумів як створювати повторно використовувані модулі.

26.Розвинув здатність читати та інтегрувати чужий код.

27.Визначив залежності між модулями та налаштував взаємодію.

28.Навчився протоколювати хід виконання функцій.

29.Визначив, які поля структури критичні для збереження.

30.Розробив алгоритм пошуку студента за прізвищем.

31.Вивчив методику очищення динамічної пам’яті.

32.Реалізував логіку валідації введених даних.

33.Навчився сортувати записи за алфавітом.

34.Використав структуровані дані для представлення анкети студента.

35.Описав функціональну специфікацію ПЗ.

36.Розробив README.md проєкту з описом усіх функцій.

37.Усвідомив важливість належного оформлення коду.

38.Вивчив можливості препроцесора для уникнення дублювання коду.

39.Використовував директиву #ifndef для захисту від повторного включення.

40.Усвідомив важливість перевірки стану потоку після відкриття файлу.

41.Виявив і виправив помилки в обробці пустої бази.

42.Сформував навичку роботи з валідацією структур.

43.Навчився створювати функції для зчитування з файлу.

44.Побудував механізм для безпечного видалення елементів.

45.Додав функцію завершення програми з автоматичним збереженням.

46.Вивчив синтаксис функцій-членів у структурі.

47.Ознайомився з концепцією абстрактних типів даних (ADT).

48.Працював над розробкою модульної архітектури.

49.Реалізував логіку уникнення дублікатів під час додавання.

50.Отримав навички роботи з консольною взаємодією.

51.Розвинув розуміння безпеки роботи з файлами.

52.Вдосконалив здатність до відлагодження коду.

53.Усвідомив важливість збереження логічної цілісності даних.

54.Навчився зберігати структуру вмісту файлу після змін.

55.Використовував структури даних для представлення реального об’єкта (студента).

56.Вдосконалив навички командної взаємодії в GitHub.

57.Вчився писати документацію до функцій.

58.Розробив систему керування діями користувача через меню.

59.Розумію як створювати кросплатформенне ПЗ.

60.Дослідив, як взаємодіють модулі з пам’яттю.

61.Вчився застосовувати модифікатори доступу до змінних.

62.Усвідомив переваги модульного тестування.

63.Навчився використовувати константні параметри у функціях.

64.Засвоїв принцип «розділяй і володарюй» у побудові логіки програми.

65.Реалізував серіалізацію структур у файл.

66.Додав обробку помилок відкриття/читання/запису файлів.

67.Зрозумів, чим важливий порядок зберігання елементів у пам’яті.

68.Створив документацію до власної частини проєкту.

69.Побачив практичну користь об’єктно-орієнтованих підходів.

70.Опрацював приклади з методичок і адаптував до власного проєкту.

71.Навчився формувати структуру проєкту в Code::Blocks.

72.Використовував інструменти налаштування компіляції.

73.Розширив загальні знання з C++.

74.Узагальнив технічні труднощі та знайшов способи їх подолання.

75.Оцінив власний внесок у спільний результат.

76.Досяг мети лабораторної роботи — створити програму, що реалізує облік студентів з використанням динамічних структур даних, колективної розробки та автоматизації зберігання даних.