Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 10
з навчальної дисципліни
“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕННЯМ

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАВ
студент академічної групи КН-24
Ковальова Єва

ПЕРЕВІРИВ
ст. викладач кафедри кібербезпеки
та програмного забезпечення
Коваленко Анастасія Сергіївна

Кропивницький – 2025

ТЕМА: РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕННЯМ

Мета роботи полягає у набутті грунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структура, об'єднання, множина, перелік, перетворення типів даних, використання файлових потоків та функцій стандартних бібліотек для оброблення символьної інформації.

https://github.com/odorenskyi/

ВАРІАНТ 3

—— ВХІДНИЙ ТЕКСТ - ВМІСТ ВХІДНОГО ТЕКСТОВОГО ФАЙЛУ ——

Довільний текст українською мовою.

—— ЗАДАЧА 10.1 ——

У ***вихідний*** текстовий файл записати:

- авторську інформацію: ім'я й прізвище розробника модуля, установа/організація, місто, країна, рік розробки;
- кількість абзаців у тексті із вхідного файла;
- повідомлення, чи є у тексті з вхідного файла слова "`Україна`", "`університет`" та "`блокнот`" (у програмі слід реалізувати розрізнення слів, наприклад, "університет" та "університету" тощо).

—— ЗАДАЧА 10.2 ——

У ***вхідний*** текстовий файл дописати:

- кількість знаків пунктуації у ньому, дату й час дозапису інформації.

—— ЗАДАЧА 10.3 ——

Вхідні дані – числові значення x, y, z та натуральне число b. У ***вихідний*** текстовий файл дописати:

- результати виконання функцій із заголовкового файлу Modules*Прізвище*.h `s_calculation` з аргументами x, y, z;
- число b у двійковому коді.

- Мова повідомлень – українська (*наприклад*, якщо у вихідний файл записується кількість символів у вхідному файлі, то модуль повинен сформувати й записати/дописати повноцінне речення: "У файлі `ВхФайл.txt` міститься 257 символів.").
- Вхідний файл `*.txt` створюється користувачем, у який за допомогою текстового редактора (у Windows – Блокнот) записується вхідний текст відповідно до завдання; вихідний файл створюється програмним модулем; імена вхідного й вихідного файлів є параметрами відповідного модуля.
- Перед читання/записом з/у файловий потік слід реалізувати перевірку його відкриття; після завершення – закрити всі відкриті файлові потоки.
- Оброблення текстових файлів рекомендовано реалізувати за допомогою файлових потоків `ofstream` та `ifstream` `<fstream>` C++.
- Для отримання локальної дати й часу ОС можна використати стандартні функції `time`, `ctime`, `localtime`, `asctime`, реалізовані у `ctime` / `time.h`.

Аналіз і постановка задач 10.1, 10.2 та 10.3

Зрозуміло, перейду на українську мову.

Аналіз та постановка завдань 10.1, 10.2 та 10.3

Ваше завдання передбачає створення програми на C++, яка опрацьовує текстові файли. Ось вимоги:

Завдання 10.1

Потрібно створити програму, яка записує у вихідний текстовий файл:

- **Авторську інформацію (ім'я, організація, місто, країна, рік розробки)**
- **Кількість абзаців у вхідному текстовому файлі**
- **Перевірку, чи є у тексті з вхідного файлу слова "Україна", "університет" та "блокнот", враховуючи різні відмінки слів**

Завдання 10.2

Дописати у вихідний файл:

- **Кількість знаків пунктуації у вхідному тексті**
- **Дату й час дозапису інформації**

Завдання 10.3

При вхідних даних x, y, z та натуральному числі b, дописати у вихідний файл:

- **Результати виконання функції s_calculation з заголовкового файлу ModulesПрізвище.h з аргументами x, y, z**
- **Число b у двійковому коді**

Додаткові вимоги:

- **Повідомлення мають бути українською мовою**
- **Використовувати файлові потоки (ofstream та ifstream) для роботи з файлами**
- **Використовувати функції часу (time, ctime тощо) для отримання локальної дати та часу**
- **Перевіряти існування файлів перед читанням/записом і закривати всі файлові потоки після операцій**

Лістинг 10.1

```
void analyze_text_file(const string& input_file, const string& output_file) {
    ifstream infile(input_file);
    ofstream outfile(output_file);

    if (!infile.is_open() || !outfile.is_open()) {
        cerr << "Не вдалося відкрити файл." << endl;
        return;
    }

    // Авторська інформація
    outfile << "Автор: Єва Ковальова\n";
    outfile << "Установа: ЦНТУ\n";
    outfile << "Місто: Кропивницький\n";
    outfile << "Країна: Україна\n";
    outfile << "Рік розробки: 2025\n\n";

    string line;
    int paragraph_count = 0;
    bool in_paragraph = false;
    int ukraine_count = 0, university_count = 0, notebook_count = 0;

    while (getline(infile, line)) {
        // Підрахунок абзаців
        if (!line.empty()) {
            if (!in_paragraph) {
                paragraph_count++;
                in_paragraph = true;
            }
        } else {
            in_paragraph = false;
        }

        // Переведення рядка в нижній регістр для пошуку слів
        locale loc(""); // Використання локалі за замовчуванням системи
        string lower_line;
        for (char c : line) {
            lower_line += tolower(c, loc);
        }

        // Пошук слів
        size_t pos = 0;
        while ((pos = lower_line.find("україна", pos)) != string::npos) {
            ukraine_count++;
            pos += string("україна").length();
```

```
        }
        pos = 0;
        while ((pos = lower_line.find("університет", pos)) != string::npos) {
            university_count++;
            pos += string("університет").length();
        }
        pos = 0;
        while ((pos = lower_line.find("блокнот", pos)) != string::npos) {
            notebook_count++;
            pos += string("блокнот").length();
        }
    }

    // Запис результатів у вихідний файл
    outfile << "Кількість абзаців: " << paragraph_count << "\n";
    outfile << "Слово 'Україна' зустрічається: " << ukraine_count << " разів\n";
    outfile << "Слово 'університет' зустрічається: " << university_count << "
разів\n";
    outfile << "Слово 'блокнот' зустрічається: " << notebook_count << " разів\n";

    infile.close();
    outfile.close();
}
```

Лістинг 10.2

// 10.2

```
void append_punctuation_and_timestamp(const string& input_file) {
    ifstream infile(input_file, ios::in);
    ofstream outfile(input_file, ios::app);

    if (!infile.is_open() || !outfile.is_open()) {
        cerr << "Не вдалося відкрити файл." << endl;
        return;
    }

    int punctuation_count = 0;
    char ch;

    // Підрахунок знаків пунктуації
    while (infile.get(ch)) {
        if (ispunct(ch)) {
            punctuation_count++;
        }
    }
```

```
    // Отримання поточної дати й часу
    time_t now = time(0);
    tm* local_time = localtime(&now);
    char timestamp[100];
    strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", local_time);

    // Додавання інформації у файл
    outfile << "\nКількість знаків пунктуації: " << punctuation_count << "\n";
    outfile << "Дата й час дозапису: " << timestamp << "\n";

    infile.close();
    outfile.close();
}
```

Лістинг 10.3

```
void append_results_to_file(const string& output_file, double x, double y, double z,
int b) {
    ofstream outfile(output_file, ios::app);

    if (!outfile.is_open()) {
        cerr << "Не вдалося відкрити файл." << endl;
        return;
    }

    // Результат виконання функції s_calculation
    double s_result = s_calculation(x, y, z);
    outfile << "Результат s_calculation(" << x << ", " << y << ", " << z << "): " <<
s_result << "\n";

    // Число b у двійковому коді
    string binary = "";
    int temp = b;
    while (temp > 0) {
        binary = (temp % 2 == 0 ? "0" : "1") + binary;
        temp /= 2;
    }
    outfile << "Число " << b << " у двійковому коді: " << binary << "\n";

    outfile.close();
}
```

Лістинг TestDriver

```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <ctime>
```

```
#include <windows.h>
#include "ModulesKovalova.h"

using namespace std;

int main() {
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    string input_file = "input.txt";
    string output_file = "output.txt";

    while (true) {
        cout << "Оберіть дію:\n";
        cout << "1. Аналіз текстового файлу\n";
        cout << "2. Додати кількість пунктуацій та часову мітку\n";
        cout << "3. Додати результати у файл\n";
        cout << "4. Вийти\n";
        cout << "Ваш вибір: ";

        int choice;
        cin >> choice;

        switch (choice) {
            case 1:
                analyze_text_file(input_file, output_file);
                cout << "Аналіз текстового файлу завершено.\n";
                break;
            case 2:
                append_punctuation_and_timestamp(input_file);
                cout << "Додано кількість пунктуацій та часову мітку.\n";
                break;
            case 3: {
                double x, y, z;
                int b;
                cout << "Введіть значення x: ";
                cin >> x;
                cout << "Введіть значення y: ";
                cin >> y;
                cout << "Введіть значення z: ";
                cin >> z;
                cout << "Введіть значення b: ";
                cin >> b;
                append_results_to_file(output_file, x, y, z, b);
                cout << "Результати додано у файл.\n";
                break;
            }
            case 4:
                cout << "Вихід з програми.\n";
                return 0;
            default:
                cout << "Невірний вибір. Спробуйте ще раз.\n";
        }
    }
```

```
    return 0;
}
```

Лістинг ModulesKovalova.h

```
#ifndef MODULESKOVALOVA_H_INCLUDED
#define MODULESKOVALOVA_H_INCLUDED

double s_calculation(double x, double y, double z);
double gas_payment(double volume);
void shoes(float sizes);
int binary_count(int N);
void task_selector(char option);

void analyze_text_file(const std::string& input_file, const std::string& output_file);
void append_punctuation_and_timestamp(const std::string& input_file);
void append_results_to_file(const std::string& output_file, double x, double y, double z, int b);

#endif // MODULESKOVALOVA_H_INCLUDED
```

Висновок до лабораторної роботи

В ході виконання лабораторної роботи було набуто ґрунтовних вмінь і практичних навичок реалізації програмних модулів створення й оброблення даних складових типів із файловим введенням/виведенням у середовищі розробки Code::Blocks мовою програмування C++. Робота була спрямована на розвиток навичок роботи з різними типами даних, такими як масиви, структури, об'єднання, множини та переліки.

Під час виконання лабораторної роботи було створено програмні модулі для вирішення поставлених завдань та включено їх до складу статичної бібліотеки. Також було розроблено тестовий драйвер для автоматизованої перевірки роботи програмних модулів, що забезпечило можливість комплексного тестування всіх функціональних можливостей.

Важливим аспектом роботи було освоєння методів перетворення типів даних, що дозволило ефективно маніпулювати різними форматами інформації. Особлива увага

приділялася використанню файлових потоків, які є ключовими для організації введення та виведення даних з файлів. Це дозволило розширити можливості програмних модулів і забезпечити їх взаємодію із зовнішніми джерелами даних.

У процесі виконання роботи були використані функції стандартних бібліотек C++ для оброблення символьної інформації, що дозволило ефективно обробляти текстові дані. Робота з такими бібліотеками суттєво спростила процес маніпуляції даними та забезпечила ефективне виконання поставлених завдань.

Особливу цінність при виконанні лабораторної роботи мало застосування модульного підходу до розробки програмного забезпечення. Цей підхід дозволив створити ізольовані компоненти, які можуть бути незалежно тестовані та інтегровані в різні програмні системи. Такий підхід значно підвищує якість та надійність програмного забезпечення, полегшує процес пошуку і виправлення помилок.

Практичний досвід, отриманий під час роботи з середовищем розробки Code::Blocks, дозволив поглибити розуміння процесу створення програмного забезпечення, починаючи від написання коду до компіляції, тестування та налагодження програм. Важливим аспектом було також використання системи контролю версій Git для зберігання результатів роботи, що дозволило забезпечити належний рівень організації та керування процесом розробки.

Практична реалізація завдань дозволила закріпити теоретичні знання, отримані в рамках вивчення мови програмування C++, та вдосконалити вміння розробки програмного забезпечення з використанням файлового введення та виведення. В процесі виконання було зосереджено увагу на розробці зручних інтерфейсів для взаємодії з користувачем, що включало створення інтуїтивно зрозумілих меню, повідомлень та підказок.

Особлива увага була приділена оптимізації коду з метою забезпечення максимальної ефективності виконання програм. Це включало аналіз алгоритмів, вибір оптимальних структур даних та реалізацію ефективних операцій для роботи з ними. Такий підхід дозволив досягти високої продуктивності при обробці великих обсягів даних.

Під час виконання лабораторної роботи було також приділено увагу питанням безпеки та надійності програмного забезпечення. Це включало обробку виключних

ситуацій, перевірку вхідних даних та забезпечення стійкості програм до неправильних вхідних даних. Такий підхід дозволив створити надійні та безпечні програмні модулі, готові до використання в реальних умовах.

У результаті виконання лабораторної роботи було отримано практичні навички, які можуть бути використані для розробки різноманітних програмних систем, починаючи від простих консольних додатків до складних програмних комплексів з графічним інтерфейсом користувача. Ці навички є цінними для подальшого професійного розвитку в галузі програмування та розробки програмного забезпечення.

Отримані результати демонструють коректну роботу створених програмних модулів та підтверджують досягнення мети лабораторної роботи. Розроблені програмні модулі можуть бути використані як основа для подальшого розширення та вдосконалення функціональних можливостей, а також для інтеграції в більш складні програмні системи.

Таким чином, виконана лабораторна робота є важливим етапом у формуванні професійних компетенцій в галузі програмування та розробки програмного забезпечення, що дозволяє застосовувати отримані знання та навички для вирішення реальних практичних завдань.