МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ

ЛАБОРАТОРНА РОБОТА №10

З ДИСЦИПЛІНИ

БАЗОВІ МЕТОДОЛОГІЇ ТА ТП

ТЕМА ЛАБОРАТОРНОЇ РОБОТИ:

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

Студент:Новак П.А.

Група КІ-24

Викладач: Коваленко А.С.

м.Кропивницький

2025

**Мета роботи:** полягає у набутті ґрунтовних 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.

**ОБЛАДНАННЯ МАТЕРІАЛИ, ПРОГРАМНІ ЗАСОБИ**

Для виконання лабораторної роботи необхідні:

– персональний комп’ютер з ОС Windows 10 / 11, Linux 32-bit / 64- bit або Mac OS X;

– вільне кросплатформове середовище розроблення програмного забезпечення Code::Blocks (www.codeblocks.org) для платформи Windows 10 / 11, Linux 32-bit / 64-bit, або Mac OS X;

– текстовий редактор (OpenOffice Writer, Microsoft Word або ін.);

– Git-репозиторій <https://github.com/odorenskyi/student-name>.

**ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ**

1. Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище лабораторних робіт №8–9).

2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.

**РЕАЛІЗАЦІЯ ЗАВДАНЬ 10.1-10.3**

**ModulesNovak.cpp :**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <ctime>

#include "ModulesNovak.h"

#include <fstream>

#include <string>

#include <bitset>

std::string s\_calculation(int x, int y, int z) {

return std::to\_string(x + y + z);

}

void solve\_10\_1(const std::string& filename) {

std::ofstream out(filename);

std::string author = "Ім'я Прізвище, ХНУРЕ, Україна, 2025";

std::string input = "Хай щастить";

int letter\_count = 0;

bool is\_word = true;

for (char ch : input) {

if (isalpha((unsigned char)ch)) ++letter\_count;

else {

is\_word = false;

break;

}

}

out << author << "\n";

out << "Кількість літер: " << (is\_word ? letter\_count : 0) << "\n";

out << (is\_word ? "Хай щастить" : "Анатолій Таран") << "\n";

out << "Хай щастить у Вашій хаті, мамо:\n"

"Всім, хто пересунеться Ваш поріг:\n"

"Добрим людям, і птахам так само,\n"

"І котові, що в теплі принишк;\n"

"Хай щастить кожненькій деревині,\n"

"Що до хати віття притулив,\n"

"Хай щастить малесенькій травині\n"

"Й вітрові, що прилітав здаля.\n";

out.close();

}

void solve\_10\_2(const std::string& filename) {

std::ofstream out(filename, std::ios::app);

out << "\nХай щастить у Вашій хаті, мамо:\n"

"Всім, хто пересунеться Ваш поріг:\n"

"Добрим людям, і птахам так само,\n"

"І котові, що в теплі принишк;\n"

"Хай щастить кожненькій деревині,\n"

"Що до хати віття притулив,\n"

"Хай щастить малесенькій травині\n"

"Й вітрові, що прилітав здаля.\n";

time\_t now = time(0);

out << "Дата та час: " << ctime(&now);

out.close();

}

void solve\_10\_3(const std::string& inputFile, const std::string& outputFile, int x, int y, int z)

{

std::ifstream in(inputFile);

if (!in.is\_open()) {

throw std::runtime\_error("Не вдалося відкрити вхідний файл");

}

std::ofstream out(outputFile, std::ios::app);

if (!out.is\_open()) {

throw std::runtime\_error("Не вдалося відкрити вихідний файл");

}

std::string result = s\_calculation(x, y, z);

out << "Результат s\_calculation: " << result << "\n";

int b;

in >> b;

if (in.fail()) {

throw std::runtime\_error("Помилка читання числа з файлу");

}

std::string binary = std::bitset<32>(b).to\_string();

out << "Число " << b << " у двійковому коді: " << binary << "\n";

in.close();

out.close();

}

**ModulesNovak.h :**

#pragma once

#include <string>

void solve\_10\_1(const std::string& filename);

void solve\_10\_2(const std::string& filename);

void solve\_10\_3(const std::string& inputFile, const std::string& outputFile, int x, int y, int z);

std::string s\_calculation(int x, int y, int z);

**TestDrivers.cpp:**

#include <iostream>

#include <fstream>

#include "ModulesNovak.h"

int main() {

std::cout << "=== Тестування задач 10.1 - 10.3 ===\n";

solve\_10\_1("output\_10\_1.txt");

std::cout << "10.1 пройдена\n";

solve\_10\_2("output\_10\_2.txt");

std::cout << "10.2 пройдена\n";

int x = 5, y = 10, z = 15, b = 25;

solve\_10\_3("output\_10\_3.txt", x, y, z, b);

std::cout << "10.3 пройдена\n";

return 0;

}

**NovakTask.cpp:**

#include <iostream>

#include <string>

#include "ModulesNovak.h"

#include <locale>

using namespace std;

void runTask10\_1() {

string inputFile;

cout << "\n[10.1] Введіть ім'я вхідного файлу (наприклад: input10\_1.txt): ";

cin >> inputFile;

solve\_10\_1(inputFile.c\_str());

cout << "Задача 10.1 виконана. Перевірте файл: " << inputFile << endl;

}

void runTask10\_2() {

string inputFile;

cout << "\n[10.2] Введіть ім'я вхідного файлу (той самий або новий): ";

cin >> inputFile;

solve\_10\_2(inputFile.c\_str());

cout << "Задача 10.2 виконана. Перевірте файл: " << inputFile << endl;

}

void runTask10\_3() {

string inputFile, outputFile;

int x, y, z;

cout << "\n[10.3] Введіть значення x, y, z (натуральні числа): ";

cin >> x >> y >> z;

cout << "Введіть ім'я вхідного файлу: ";

cin >> inputFile;

cout << "Введіть ім'я вихідного файлу: ";

cin >> outputFile;

solve\_10\_3(inputFile, outputFile, x, y, z);

cout << "Задача 10.3 виконана. Результат у файлі: " << outputFile << endl;

}

int main() {

setlocale(LC\_ALL, "ukr");

cout << "=== Демонстраційна програма (Lab10) ===" << endl;

int choice;

do {

cout << "\nОберіть завдання для демонстрації:\n";

cout << "1 - Завдання 10.1\n";

cout << "2 - Завдання 10.2\n";

cout << "3 - Завдання 10.3\n";

cout << "0 - Вийти\n";

cout << "Ваш вибір: ";

cin >> choice;

switch (choice) {

case 1: runTask10\_1(); break;

case 2: runTask10\_2(); break;

case 3: runTask10\_3(); break;

case 0: cout << "Завершення програми.\n"; break;

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

}

} while (choice != 0);

return 0;

}

**Задача 10.1:**

**Умова:**  
З текстового файлу зчитується один рядок, у якому може бути як слово, так і знак пунктуації. Якщо це слово, то до файлу дописується інформація про довжину слова і частину вірша. Якщо це знак пунктуації — додається інформація з прізвищем першого космонавта. У будь-якому випадку, дописується дата виконання.

**Постановка задачі:**  
Розробити функцію, яка зчитує один рядок з вхідного текстового файлу, аналізує його (визначає тип – слово чи знак пунктуації) та формує вихідний текстовий файл відповідно до описаної умови. Якщо файл вже існує – його вміст перезаписується.

**Задача 10.2:**

**Умова:**  
У вхідному текстовому файлі зберігається частина вірша. Необхідно автоматично додати до нього ще один рядок вірша, а також інформацію про автора та дату обробки.

**Постановка задачі:**  
Розробити функцію, яка відкриває вхідний файл, дописує до нього новий рядок вірша, додає інформацію про автора та дату. Якщо файл відсутній – створюється новий.

**Задача 10.3:**

**Умова:**  
Вхідні параметри — три натуральні числа x, y, z. Необхідно обчислити значення виразу s = x \* y \* z, після чого результат записати у вихідний файл у двійковому представленні (у вигляді рядка з '0' та '1').

**Постановка задачі:**  
Розробити функцію, яка приймає три цілочисельні параметри та два імені файлів (вхідного і вихідного). Після обчислення виразу s = x \* y \* z, результат переводиться у двійкову систему числення і записується у файл.

**АРГУМЕНТИ**

1. Вивчається створення статичних бібліотек.
2. Формується розуміння модульної структури проєкту.
3. Тренується використання масивів.
4. Вивчається структура даних struct.
5. Вивчається використання об’єднань (union).
6. Освоюється тип даних enum (перелік).
7. Опановується множинна логіка.
8. Практикується перетворення типів даних.
9. Засвоюється робота з текстовими файлами.
10. Формуються навички роботи з символьною інформацією.
11. Розвивається навичка написання бібліотек C++.
12. Покращується вміння організовувати код.
13. Стимулюється повторне використання коду.
14. Вчить перевіряти працездатність коду через тестування.
15. Допомагає налагодити автоматизовану перевірку результатів.
16. Дає досвід роботи з IDE Code::Blocks.
17. Пояснює принцип компіляції та лінкування статичних бібліотек.
18. Дозволяє поглибити знання про інтерфейси функцій.
19. Навчає правильного розділення логіки (функції, файли).
20. Розвиває навички читання та аналізу специфікацій завдань.
21. Показує реальні приклади обробки символьних даних.
22. Формує аналітичне мислення при роботі з типами.
23. Підсилює знання про буферизоване введення/виведення.
24. Вчить створювати та використовувати хедери (.h файли).
25. Розвиває вміння писати тестові сценарії.
26. Сприяє глибшому розумінню C++ стандартної бібліотеки.
27. Навчає використовувати структури даних ефективно.
28. Вчить дотримуватись стилю іменування функцій.
29. Дозволяє побачити важливість інкапсуляції.
30. Вчить компілювати проєкт із кількох модулів.
31. Формує навички рефакторингу коду.
32. Демонструє важливість коректного тестування.
33. Вивчається вплив типів на продуктивність.
34. Тренується робота з текстовими файлами: читання і запис.
35. Дозволяє дослідити обмеження C++ типів.
36. Розвиває здатність писати переносимий код.
37. Навчає аналізувати вхідні дані та обробляти помилки.
38. Стимулює роботу з власними прикладами тестування.
39. Пояснює, як будувати бібліотеки для інших програм.
40. Розвиває вміння працювати з великим кодом.
41. Підвищує впевненість у роботі з типами даних.
42. Покращує розуміння стандартних функцій обробки рядків.
43. Показує практичну цінність структурування коду.
44. Вчить працювати з константами та визначеннями (define, const).
45. Сприяє повторному використанню коду в інших проєктах.
46. Навчає основам документування коду.
47. Підвищує дисципліну програмування.
48. Дає приклади роботи з різними типами задач.
49. Дозволяє застосувати теорію на практиці.
50. Є завершенням важливого блоку лабораторних робіт №8–10.

**КОНТРОЛЬНІ ЗАПИТАННЯ**

1. **Призначення і синтаксис try-throw-catch:**

**Призначення** — обробка виняткових ситуацій під час виконання програми (наприклад, ділення на нуль, відсутність файлу тощо).

**Синтаксис:**

cpp

try {

// код, який може викликати виняток

if (error) throw "Помилка!";

}

catch (const char\* msg) {

std::cout << "Виняток: " << msg << std::endl;

}

**2. Приклад міжмодульної змінної:**

**Файл globals.h:**

cpp

extern int globalCounter;

**Файл main.cpp:**

cpp

#include "globals.h"

int globalCounter = 0;

**Файл module.cpp:**

cpp

#include "globals.h"

void increment() {

globalCounter++;

}

**3. Область видимості об’єктів у функції main:**

Змінні, типи, константи, оголошені всередині main(), мають **локальну область видимості**, тобто доступні лише у межах тіла функції main.

**4. Порівняльний аналіз enum і масиву:**

|  |  |  |
| --- | --- | --- |
| **Критерій** | **enum** | **Масив** |
| Тип | Перелік значень | Набір однотипних даних |
| Пам'ять | Зберігається як int | Може бути будь-який тип |
| Індексація | Немає прямої індексації | Доступ за індексом |
| Семантика | Позначення символічних значень | Зберігання набору даних |

**5. Потік і файловий потік:**

**Потік** — абстракція для вводу/виводу даних.  
**Файловий потік** — потік, що працює з файлами (ifstream, ofstream, fstream).

**Відмінність:** файлові потоки потребують відкриття/закриття файлів і можуть зчитувати/записувати дані постійно.

**Алгоритм читання/запису з/у файл:**

cpp

#include <fstream>

#include <string>

void readWriteFile(const std::string& filename) {

std::ofstream fout(filename);

fout << "Привіт, файл!\n";

fout.close();

std::ifstream fin(filename);

std::string line;

while (getline(fin, line)) {

std::cout << line << std::endl;

}

fin.close();

}

**7. Об’єм файлу з числом π до 1 000 000 знаків:**

* 1 символ = 1 байт (у текстовому форматі)
* π = 3. + 1,000,000 цифр = **1,000,002 символи**
* **Розмір ≈ 1,000,002 байти (≈1 МБ)**

**8. Перевірка відкриття файлу:**

cpp

std::ifstream fin("data.txt");

if (!fin.is\_open()) {

std::cerr << "Файл не відкрито!" << std::endl;

}

**Мета** — уникнення роботи з неіснуючим файлом, захист програми від помилки.

**9. Текстовий vs. двійковий потік:**

|  |  |  |
| --- | --- | --- |
| **Ознака** | **Текстовий потік** | **Двійковий потік** |
| Формат | Символи (ASCII) | Бінарні байти |
| Редагування | Можна у текстовому редакторі | Неможливо |
| Обробка | Автоматичне перетворення | Чіткий контроль |

**10. Режими відкриття файлів у C++:**

* ios::in – читання
* ios::out – запис
* ios::app – дописування
* ios::trunc – очищення вмісту
* ios::binary – двійковий режим

**11. Класи файлових потоків (<fstream>):**

* ifstream — для читання
* ofstream — для запису
* fstream — для читання і запису
* **12. Константи ios і їх призначення:**

|  |  |
| --- | --- |
| **Константа** | **Призначення** |
| ios::in | Відкриття для читання |
| ios::out | Відкриття для запису |
| ios::app | Додавання в кінець файлу |
| ios::trunc | Очищення вмісту |
| ios::binary | Двійковий режим |
| ios::ate | Початок із кінця файлу |

**13. Основні функції-члени fstream:**

cpp

fstream file;

file.open("file.txt", ios::in);

file.close();

if (file.eof()) std::cout << "Кінець файлу\n";

**14. Алгоритм видалення слова з файлу:**

1. Зчитати весь файл у пам’ять.
2. Видалити слово з рядків.
3. Перезаписати файл.

cpp

std::ifstream in("file.txt");

std::ofstream out("temp.txt");

std::string word, target = "видалити";

while (in >> word) {

if (word != target) out << word << " ";

}

in.close(); out.close();

remove("file.txt");

rename("temp.txt", "file.txt");

**15. Складові типи даних C/C++:**

* struct, union, enum, масиви
* Оголошуються через ключові слова (struct, union)
* Працюють як агреговані типи, доступ через крапку (.)

**16. Множина в C/C++:**

Немає вбудованої множини — використовуються:

* Масиви + лінійний пошук
* std::set<int> у C++:

cpp

#include <set>

std::set<int> s;

s.insert(5);

if (s.count(5)) std::cout << "Є";

s.erase(5);

**17. Явне та неявне перетворення:**

* **Неявне:**

cpp

int x = 3.14; // округлення автоматично

* **Явне:**

cpp

int x = (int)3.14;

int y = static\_cast<int>(3.14);

**18. Коли використовують union:**

* Коли потрібно зекономити пам'ять, і в один момент використовується тільки одне з полів.

cpp

union Data {

int i;

float f;

char c;

};

**19. Алгоритм пошуку числа в файлі:**

cpp

std::ifstream in("data.txt");

int x;

while (in >> x) {

if (x == target) {

std::cout << "Знайдено!";

break;

}

}

in.close();

**20. Що містить symbol\_transaction у:**

cpp

short symbol\_transaction = sizeof short('R');

* 'R' — це int (виклик short('R') не має сенсу)
* sizeof short('R') = **2 або 4** залежно від платформи

У більшості випадків:

cpp

sizeof(short('R')) → sizeof(int) → 4

Отже, symbol\_transaction = 4