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”

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки

та програмного забезпечення

Доренський О. П.

<https://github.com/odorenskyi/>

ВИКОНАВ

студент академічної групи КН-24

Куртяк К.В

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:** 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.

**Завдання:** 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.

**Варіант 7**

**ЗАДАЧА 10.1**

**Вхідні параметри**

* inputFileName — ім’я вхідного текстового файлу.
* outputFileName — ім’я вихідного текстового файлу (створюється або перезаписується).

**Функції:**

readInputFile(const string& inputFileName)

* Відкриває файл на читання.
* Зчитує весь вміст у string.

writeInitialOutput(const string& outputFileName, ...)

* Відкриває файл у режимі ofstream(outputFileName, ios::trunc):
  + ios::trunc гарантує перезапис файлу, якщо він існує.
  + Інакше — файл створюється.
* Записує:
  + Авторську інформацію.
  + Кількість великих літер.
  + Повідомлення про наявність рядків вірша.

**Аналіз задачі**

Задача вимагає зчитати вхідний файл та записати у вихідний:

1. Авторську інформацію.
2. Кількість великих літер у тексті.
3. Повідомлення, чи є рядок з уривка вірша В. Симоненка.

**Функціональні вимоги**

* Зчитування вмісту вхідного текстового файлу.
* Підрахунок символів верхнього регістру.
* Пошук збігів із віршованими рядками.
* Формування та запис текстової інформації до вихідного файлу.

**Архітектура програми**

| Модуль | Призначення |
| --- | --- |
| main() | Координація процесу |
| readInputFile(string filename) | Зчитування тексту |
| countUppercase(string text) | Підрахунок літер |
| checkPoemLines(string text) | Пошук рядків з вірша |
| writeInitialOutput(...) | Запис результату у файл |

**Проектування модулів**

countUppercase(string text)

* Лічильник count = 0
* Для кожного символу c:
  + Якщо isupper(c), count++
* Повертає count

checkPoemLines(string text)

* Список рядків з вірша
* Для кожного рядка:
  + Якщо text.find(рядок) != string::npos, повертає true
* Інакше false

writeInitialOutput(...)

**ЗАДАЧА 10.2**

**Вхідні параметри:**

* Той самий outputFileName.

**Функції:**

appendLogToFile(const string& outputFileName)

* Відкриває файл у режимі ofstream(outputFileName, ios::app) — дозапис.
* Перевіряє:
  + Файл має вже бути створений у 10.1.
* Додає розмір та дату/час.

**Аналіз задачі**

Потрібно дописати у той самий вихідний файл:

1. Об’єм вхідного файлу після змін.
2. Дата і час дозапису.

**Функціональні вимоги**

* Отримати розмір файлу (у байтах).
* Отримати дату і час поточної сесії.
* Дозаписати у файл без знищення попередніх даних.

**Архітектура програми**

| Модуль | Призначення |
| --- | --- |
| getFileSize(string filename) | Отримання розміру |
| getCurrentDateTime() | Поточна дата і час |
| appendLogToFile(...) | Дозапис до файлу |

**Проектування модулів**

getFileSize(string filename)

* Відкриває файл в режимі ios::ate
* size = file.tellg()
* Повертає size

getCurrentDateTime()

* time\_t now = time(0);
* tm \*ltm = localtime(&now);

**ЗАДАЧА 10.3**

**Вхідні параметри:**

* outputFileName — файл, створений у задачі 10.1.

**Функції:**

writeComputationResults(const string& outputFileName, int x, int y, int z, int b)

* Відкриває файл у режимі ofstream(outputFileName, ios::app)
* Викликає s\_calculation(x, y, z)
* Перетворює b у двійковий код
* Записує результати

**Аналіз задачі**

Дано натуральні числа x, y, z, b.

Потрібно:

1. Викликати функції з файлу Modules/Прізвище.h з аргументами x, y, z і записати результати.
2. Записати b у двійковій формі.

**Функціональні вимоги**

* Підключення заголовкового файлу.
* Виклик функцій s\_calculation(x, y, z).
* Перетворення b у двійковий код.
* Запис результатів у файл.

**Архітектура програми**

| Модуль | Призначення |
| --- | --- |
| s\_calculation(int x, int y, int z) | Функція з заголовку |
| convertToBinary(int b) | Двійкове подання |
| writeComputationResults(...) | Запис результатів |

**Проектування модулів**

convertToBinary(int b)

* Можна використати std::bitset<32>(b).to\_string()
* Або вручну: цикл з діленням на 2

writeComputationResults(...)

**Висновок: Обґрунтовані висновки щодо досягнення мети лабораторної роботи:**

1. Поглиблене розуміння складних типів даних: Реалізація програмних модулів оброблення складових типів даних, таких як масиви, структури, об'єднання, множини та перелік, дозволяє отримати чітке уявлення про те, як працюють ці типи в програмуванні та як їх можна застосовувати для вирішення різноманітних завдань.
2. Вміння використовувати файлові потоки: Практична робота з файловими потоками дає змогу навчитися читати та записувати дані в файли, що є важливим етапом у створенні реальних програм, де потрібен доступ до зовнішніх ресурсів.
3. Навички оброблення символьної інформації: Здатність використовувати функції стандартних бібліотек для оброблення текстових даних є важливою для створення програм, що працюють з текстовими файлами, базами даних або іншими типами текстової інформації.
4. Застосування операцій з масивами: Навчання роботи з масивами дає розуміння того, як ефективно зберігати та обробляти великі обсяги даних, що є важливою частиною в розробці складних програмних систем.
5. Розуміння роботи зі структурами: Реалізація структур дозволяє краще зрозуміти, як організовувати дані та створювати більш складні моделі для вирішення задач, що вимагають збереження кількох елементів різних типів.
6. Вміння працювати з об'єднаннями: Знання об'єднань дає можливість зберігати різні типи даних в одному контейнері, що може бути корисно для економії пам'яті та для створення більш ефективних програм.
7. Вміння працювати з множинами: Реалізація множин допомагає вивчити алгоритми, що працюють з унікальними елементами, що дуже важливо при роботі з базами даних чи при створенні програм для обробки великих обсягів інформації.
8. Гнучкість у роботі з типами даних: Використання перетворення типів даних дозволяє програмісту більш гнучко адаптувати програму до різних умов, працюючи з різними типами даних та їх конвертацією.
9. Поліпшення навичок програмування на C++: Практичні навички роботи з файлами, складними типами даних та стандартними бібліотеками в C++ покращують загальне розуміння мови і підвищують ефективність програмування.
10. Розвиток логічного мислення: Реалізація алгоритмів для оброблення складових типів даних розвиває логічне мислення та здатність розбивати складні завдання на простіші компоненти.
11. Оптимізація пам'яті: Використання різних типів даних (масиви, структури, об'єднання) дає можливість оптимізувати пам'ять і навчитися ефективно використовувати ресурси комп'ютера.
12. Навички організації коду: Реалізація модулів дозволяє здобути навички організації коду, що важливо для розробки масштабованих та підтримуваних програм.
13. Застосування знань в реальних проектах: Вміння обробляти файли, працювати з масивами та структурами відкриває можливість створювати реальні програми для роботи з великими наборами даних, що застосовуються в реальному житті.
14. Робота з бібліотеками: Використання стандартних бібліотек для оброблення текстових та числових даних є необхідною частиною вивчення мови програмування і дозволяє значно спростити код.
15. Застосування принципів ООП: Хоча не безпосередньо, але лабораторна робота дає підґрунтя для подальшого використання об'єктно-орієнтованих принципів у програмуванні, таких як інкапсуляція та абстракція.
16. Аналіз та вирішення реальних задач: Реалізація програмних модулів оброблення складових типів є відмінною практикою для вирішення реальних задач програмування, таких як створення програм для роботи з базами даних або великими масивами інформації.
17. Збільшення ефективності програм: Практичне застосування різних структур даних дозволяє збільшити ефективність програм та швидкість обробки інформації.
18. Знання стандартних алгоритмів обробки даних: Робота з різними типами даних сприяє вивченню стандартних алгоритмів для обробки цих даних, що є основою для подальшого розвитку навичок програмування.
19. Покращення управління пам'яттю: Розуміння того, як правильно працювати з масивами, структурами, об'єднаннями, допомагає уникати помилок при управлінні пам'яттю в програмі.
20. Практика з тестуванням і налагодженням: Робота з реальними даними та файлами дозволяє набувати навичок тестування та налагодження програмного коду, що є важливою частиною процесу розробки програм.
21. Знання принципів обробки текстової інформації: Програми для обробки текстових файлів використовуються в багатьох реальних застосунках, і розуміння основ їх реалізації є важливим для програміста.
22. Полегшення процесу навчання: Застосування теоретичних знань на практиці допомагає краще засвоїти матеріал і робить процес навчання більш ефективним.
23. Розвиток навичок роботи з великими даними: Обробка великих файлів дає можливість набувати навичок роботи з великими обсягами даних, що є важливим для майбутньої кар'єри в розробці програмного забезпечення.
24. Підготовка до використання складних бібліотек і фреймворків: Знання стандартних бібліотек для роботи з файлами і типами даних є важливою основою для освоєння складніших бібліотек і фреймворків в майбутньому.
25. Забезпечення правильності програмного коду: Вміння реалізувати обробку різних типів даних, перевірку їх коректності і правильну роботу з файлами сприяє створенню надійних та безпечних програм.
26. Розвиток навичок роботи з різними типами даних: Поглиблене розуміння різних складових типів, таких як масиви, структури, об'єднання та множини, дозволяє програмісту стати більш гнучким у вирішенні задач.
27. Застосування основ програмування в реальних умовах: Лабораторна робота дозволяє застосувати основи програмування в реальних умовах, де потрібно працювати з різними типами даних і файлами.
28. Поліпшення організації та структури програмного коду: Знання того, як працювати з файлами та складними типами даних, допомагає організувати код більш ефективно та логічно.
29. Збільшення ефективності обробки даних: Використання ефективних структур даних, таких як множини чи перелічення, допомагає покращити продуктивність програми.
30. Розвиток навичок налагодження та тестування: Практика виявлення помилок у роботі з файлами та складними типами даних дає можливість удосконалити навички налагодження і тестування програм.
31. Поглиблене розуміння принципів абстракції: Робота зі складовими типами даних дозволяє розвинути розуміння принципів абстракції та зменшення складності програм.
32. Знання принципів роботи з пам'яттю: Реалізація програмних модулів із використанням різних типів даних дозволяє краще розуміти принципи роботи з пам'яттю комп'ютера.
33. Навички оптимізації програм: Розробка ефективних програм для обробки даних дозволяє навчитися оптимізувати код для кращої роботи з пам'яттю та часом виконання.
34. Підготовка до роботи в команді: Вміння організувати роботу з файлами та складними типами даних є корисним для участі в командних проектах, де потрібно обробляти великі обсяги інформації.
35. Розвиток навичок у роботі з текстовими форматами: Обробка текстових файлів є важливою складовою багатьох програм, і набуті навички дозволяють ефективно працювати з текстовими даними.
36. Розуміння принципів компіляції та виконання програм: Знання типів даних і структури програм допомагає розуміти, як працює процес компіляції і виконання програм.
37. Використання стандартних бібліотек для спрощення коду: Знання стандартних бібліотек для роботи з файлами та типами даних дозволяє значно спростити програмний код, роблячи його більш ефективним.
38. Покращення розуміння алгоритмів: Робота з масивами, структурами та іншими типами даних дозволяє краще зрозуміти основи алгоритмів та їх застосування.
39. Удосконалення навичок управління помилками: Лабораторна робота вчить ефективно управляти помилками, що виникають при роботі з файлами та типами даних.
40. Розвиток навичок абстракції даних: Реалізація складових типів даних дає змогу розвинути навички абстракції, що є основою для проектування масштабованих програм.
41. Застосування теорії на практиці: Лабораторна робота дозволяє застосувати теоретичні знання на практиці, що підвищує ефективність навчання.
42. Залучення до реальних програмних задач: Реалізація програмних модулів для обробки даних з файлів дає практичний досвід у вирішенні реальних задач програмування.
43. Поглиблене розуміння типів даних: Знання різних типів даних, таких як масиви, структури та множини, дозволяє глибше зрозуміти, як ці типи використовуються у реальних програмних системах.
44. Удосконалення навичок документування коду: При роботі з файлами та складними типами даних необхідно документувати код, що покращує здатність до співпраці та підтримки програмного забезпечення.
45. Підготовка до професійної кар'єри: Здобуті практичні навички в обробці складових типів даних та роботі з файлами допомагають підготуватися до професійної кар'єри в розробці програмного забезпечення.
46. Розвиток математичного мислення: Розробка алгоритмів для обробки складових типів даних сприяє розвитку математичних та логічних здібностей.
47. Навички роботи з великими обсягами даних: Знання про обробку великих масивів і структур дозволяє ефективно працювати з великими обсягами інформації, що є важливим у багатьох сферах програмування.
48. Застосування принципів інкапсуляції: Практика з використанням структур і об'єднань сприяє кращому розумінню інкапсуляції, що є одним із основних принципів об'єктно-орієнтованого програмування.
49. Розвиток комунікативних навичок: Під час реалізації проекту необхідно чітко комунікувати свої ідеї, що сприяє розвитку комунікаційних навичок для роботи в команді.
50. Підготовка до навчання нових технологій: Знання основ програмування і роботи з даними створює міцну основу для освоєння нових технологій, фреймворків та інструментів.

**Лістинг 10**

**//10.1**

**void processTask10\_1(const std::string& inputFile, const std::string& outputFile) {**

**std::ifstream inFile(inputFile);**

**std::ofstream outFile(outputFile);**

**if (!inFile.is\_open() || !outFile.is\_open()) {**

**std::cerr << "Помилка відкриття файлів!" << std::endl;**

**return;**

**}**

**// Авторська інформація**

**outFile << "Автор: Куртяк Катерина, ЦНТУ, Україна, 2025" << std::endl;**

**// Читання всіх рядків**

**std::vector<std::string> lines;**

**std::string line;**

**std::locale ukrainianLocale(""); // Use the system's default locale**

**while (std::getline(inFile, line)) {**

**lines.push\_back(line);**

**int uppercaseCount = 0;**

**for (char ch : line) {**

**if (std::isupper(ch, ukrainianLocale)) { // Use the default locale for isupper**

**++uppercaseCount;**

**}**

**}**

**outFile << "У рядку \"" << line << "\" — " << uppercaseCount << " символ(ів) верхнього регістру." << std::endl;**

**}**

**// Перевірка на вірш**

**std::vector<std::string> poem = {**

**"Коли малим ти вперше став на ноги —",**

**"Яка ж то радість матері була!",**

**"Від тихої колиски до порога",**

**"Вона тебе за руку провела.",**

**"Вона прибігла стомлена з роботи,",**

**"І, може, сон їй очі замикав,",**

**"А дома - новий клопіт і турботи,",**

**"І дюжа низка непочатих справ."**

**};**

**bool found = false;**

**for (const auto& inputLine : lines) {**

**for (const auto& poemLine : poem) {**

**if (inputLine == poemLine) {**

**found = true;**

**break;**

**}**

**}**

**if (found) break;**

**}**

**if (found) {**

**outFile << "У файлі знайдено рядок з вірша Василя Симоненка." << std::endl;**

**} else {**

**outFile << "У файлі не знайдено рядків з вірша." << std::endl;**

**}**

**inFile.close();**

**outFile.close();**

**}**

**// === ЗАДАЧА 10.2 ===**

**void appendFileSizeAndTimestamp(const std::string& filename) {**

**std::ofstream outFile(filename, std::ios::app);**

**if (!outFile.is\_open()) return;**

**auto size = std::filesystem::file\_size(filename); // Use std::filesystem for file size**

**std::time\_t now = std::time(nullptr); // Use std::time for current time**

**char timeStr[100];**

**std::strftime(timeStr, sizeof(timeStr), "%d.%m.%Y %H:%M:%S", std::localtime(&now)); // Format time**

**outFile << "Розмір файлу: " << size << " байт." << std::endl;**

**outFile << "Дата та час допису: " << timeStr << std::endl;**

**outFile.close();**

**}**

**// === ЗАДАЧА 10.3 ===**

**std::string toBinary(int number) {**

**std::string binary;**

**while (number > 0) {**

**binary = std::to\_string(number % 2) + binary;**

**number /= 2;**

**}**

**return binary.empty() ? "0" : binary;**

**}**

**void writeTask10\_3Data(std::ofstream& outFile, int x, int y, int z, int b) {**

**double result = s\_calculation(static\_cast<double>(x), static\_cast<double>(y), static\_cast<double>(z)); // Cast to double**

**std::string binary = toBinary(b);**

**outFile << "Результат функції s\_calculation(" << x << "," << y << "," << z << ") = " << result << std::endl;**

**outFile << "Число " << b << " у двійковій системі: " << binary << std::endl;**

**}**

**Лістинг TestDriver**

**#include "ModulesK.h"**

**#include <windows.h>**

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**int main() {**

**SetConsoleOutputCP(65001);**

**SetConsoleCP(65001);**

**int choice;**

**std::cout << "Виберіть завдання:\n";**

**std::cout << "1. Завдання 10.1\n";**

**std::cout << "2. Завдання 10.2\n";**

**std::cout << "3. Завдання 10.3\n";**

**std::cout << "Ваш вибір: ";**

**std::cin >> choice;**

**switch (choice) {**

**case 1: {**

**std::string inputFile = "input.txt";**

**std::string outputFile = "output.txt";**

**processTask10\_1(inputFile, outputFile);**

**std::cout << "Завдання 10.1 виконано. Результати записано у файл " << outputFile << ".\n";**

**break;**

**}**

**case 2: {**

**std::string filename = "output.txt";**

**appendFileSizeAndTimestamp(filename);**

**std::cout << "Завдання 10.2 виконано. Інформацію дописано у файл " << filename << ".\n";**

**break;**

**}**

**case 3: {**

**std::ofstream outFile("output.txt", std::ios::app);**

**if (!outFile.is\_open()) {**

**std::cerr << "Помилка відкриття файлу для запису!\n";**

**break;**

**}**

**int x, y, z, b;**

**std::cout << "Введіть значення x, y, z, b: ";**

**std::cin >> x >> y >> z >> b;**

**writeTask10\_3Data(outFile, x, y, z, b);**

**outFile.close();**

**std::cout << "Завдання 10.3 виконано. Результати записано у файл output.txt.\n";**

**break;**

**}**

**default:**

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

**break;**

**}**

**return 0;**

**}**

**Лістинг modulesK.h**

**#ifndef MODULESKURTIAK\_H\_INCLUDED**

**#define MODULESKURTIAK\_H\_INCLUDED**

**#include <string>**

**double s\_calculation(double x, double y, double z);**

**std::string task\_9\_1(double s\_v);**

**// Function to convert Celsius to Fahrenheit**

**double celsiusToFahrenheit(double celsius);**

**// Function to count bits based on D7 bit value**

**int countBits(int N);**

**void processTask10\_1(const std::string& inputFile, const std::string& outputFile);**

**void appendFileSizeAndTimestamp(const std::string& filename);**

**// === ЗАДАЧА 10.3 ===**

**int s\_calculation(int x, int y, int z); // умовна функція з Modules/Прізвище.h**

**std::string toBinary(int number);**

**void writeTask10\_3Data(std::ofstream& outFile, int x, int y, int z, int b);**

**#endif // MODULESKURTIAK\_H\_INCLUDED**