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

СЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕННЯМ

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ  
студент академічної групи

КБ-22-2 Ковальов Максим

ПЕРЕВІРИВ

викладач кафедри кібербезпеки  
та програмного забезпечення Олександр Собінов

Кропивницький – 2023

**Лабораторна робота №10**

**Реалізація програмних модулів оброблення даних**

**сладових типів з файловим введенням/виведенням**

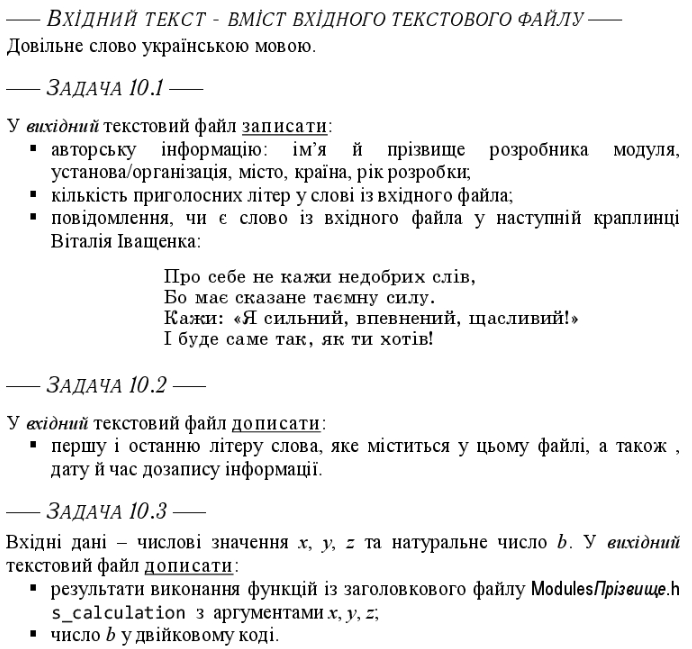
*Мета роботи* полягає у набутті ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структура, об’єднання, множина, перелік, перетворення типів даних, використання файлових потоків та функцій стандартних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

**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.

**Варіант 4**

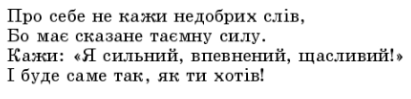


**Рішення**

**Завдання 10.1.**

**Алгоритм:**

1. Початок
2. Самостійно створюємо у проекті файл word.txt з довільним словом та файл text10\_1.txt з рядками тексту:



1. Створюємо структуру даних PersonInfo з змінними string *name*, *surname*,

*organization*, *city*, *country* та int *year*.

1. Створюємо функцію *write\_info* та передаємо у функцію ім’я файлу та структуру PersonInfo.
2. Відкриваємо файловий потік у режимі запису *op\_file*.
3. Перевіряємо чи відкритий файл *op\_file*, якщо ні то виводимо помилку.
4. У файл записуємо дані зі структури та закриваємо файловий потік *op\_file*.
5. Створюємо функцію *consonants\_num()* та передаємо у функцію ім’я файлу.
6. Відкриваємо файловий потік у режимі читання *input\_f* та створюємо рахунок приголосних зі значенням 0.
7. Перевіряємо чи відкритий файл *op\_file*, якщо ні то виводимо помилку.
8. Створюємо змінну *string* з назвою *ukr\_word* та передаємо у змінну перший рядок з файлу *input\_f*.
9. Створюємо множину приголосних літер з назвою *consonants*.
10. Створюємо цикл *for* для перевірки кожної літери слова на приголосну з множини *consonants*.
11. Повертаємо кількість приголосних *c\_count* тазакриваємо *input\_f.*
12. Створюємо функцію *write\_consosnants\_num* та передаємо у неї ім’я файлу та кількість приголосних.
13. Відкриваємо файловий потік *output\_f* у режимі дозапису(::out) та переставленим вказівником на кінець(::app).
14. Перевіряємо чи відкритий файловий потік *output\_f*, якщо ні то виводимо помилку.
15. Дозаписуємо у файл кількість приголосних та закриваємо файловий потік *output\_f.*
16. Створюємо функцію *does\_it\_has\_in()* та передаємо у функцію ім’я файлу.
17. Створюємо булеву змінну *found = false,* та дві змінні типу *string* *ukr\_word*, word.
18. Відкриваємо файлові потоки зі словом *word\_file* та *text10\_1.txt* у режимі читання.
19. Перевіряємо чи дійсно відкриті два файли одночас, якщо ні виводимо помилку.
20. Отримуємо слово з рядка файлу *word\_file* та створюємо цикл while з умовою перебору кожного слова буферної змінної *word* та зрівнюємо її з *ukr\_word*, та якщо слово знайдено то *found = true*.
21. Повертаємо значення змінної *found*.

Закриваємо файлові потоки *word\_file* та *text10\_1.txt.*

1. Створюємо функцію *write\_does\_it\_has\_in()* та передаємо у неї ім’я файлу та булеву *found.*
2. Відкриваємо файлові потоки *output\_f* у режимі дозапису(::app) та *input\_f*(“*word.txt”)* у режимі читання.
3. Створюємо *string ukr\_word.*
4. Перевіряємо чи відкрився *output\_f* та *input\_f* отримуємо рядок зі словом з файлу *input\_f* та передаємо у змінну ukr\_word.
5. Перевіряємо змінну *found* та якщо істина то слово *ukr\_word* присутнє в тексті, інакше ні.
6. Закриваємо файлові потоки *word\_file* та *text10\_1.txt.*
7. Кінець

**Код:**

// 1 -------------

// Структура для запису даних

struct PersonInfo {

std::string name;

std::string surname;

std::string organization;

std::string city;

std::string country;

int year;

};

void write\_info(const char\* filename, const PersonInfo& person){

std::ofstream op\_file(filename);

if(op\_file){

op\_file << "Авторська інформація:" << std::endl;

op\_file << "Ім'я: " << person.name << std::endl;

op\_file << "Прізвище: " << person.surname << std::endl;

op\_file << "Організація: " << person.organization << std::endl;

op\_file << "Місто: " << person.city << std::endl;

op\_file << "Країна: " << person.country << std::endl;

op\_file << "Рік: " << person.year << std::endl;

}else{

std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

}

op\_file.close();

}

// 1 -------------

// 2 -------------

int consonants\_num(const char\* filename){

// Кількість приголосних з word.txt

std::ifstream input\_f(filename); // Відкриваємо файл для читання word.txt

int c\_count = 0;

if(input\_f){

std::string ukr\_word;

std::getline(input\_f, ukr\_word);

// Создаємо множину приголосних літер

std::unordered\_set<char> consonants = {'б', 'в', 'г', 'ґ', 'д', 'ж', 'з', 'й', 'к', 'л', 'м', 'н', 'п', 'р', 'с', 'т', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь'};

for (char c : ukr\_word) {

if (consonants.count(c) == 1 && std::isalpha(c)) {

c\_count++;

}

}

}else{

std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

return -1;

}

input\_f.close();

return c\_count;

}

void write\_consosnants\_num(const char\* filename, int c\_count){

// Відкриваємо файл для запису task10\_1.txt

std::ofstream output\_f(filename, std::ios::out | std::ios::binary | std::ios::app);

// Дозапис кількості приголосних у task10\_1.txt

if(output\_f){

output\_f << "Кількість приголосних: "<< c\_count << std::endl;

}else{

std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

}

output\_f.close();

}

// 2 -------------

// 3 -------------

bool does\_it\_has\_in(const char\* filename){

bool found = false;

std::string ukr\_word;

std::string word;

// ------------ Відкриваємо файл для зчитання слова яке потрібно шукати

std::ifstream word\_file("word.txt");

std::ifstream read\_file(filename);

if(read\_file && word\_file){

std::getline(word\_file, ukr\_word);

while (read\_file >> word) {

if (ukr\_word == word) {

found = true;

break;

}

}

}else{

std::cerr << "Не вдалося відкрити файл" << std::endl;

}

read\_file.close();

word\_file.close();

return found;

}

void write\_does\_it\_has\_in(const char\* filename, bool found){

// ------------ Дозапис результата

std::ofstream output\_f(filename, std::ios::binary | std::ios::app);

std::ifstream input\_f("word.txt");

std::string ukr\_word;

if(output\_f){

if(input\_f){

std::getline(input\_f, ukr\_word);

}

if(found == true){

output\_f << "Слово \""<< ukr\_word << "\" присутнє в тексті" << std::endl;

}else{

output\_f << "Слова \""<< ukr\_word << "\" немає в тексті" << std::endl;

}

}else{

std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

}

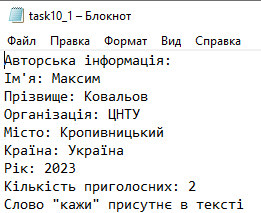
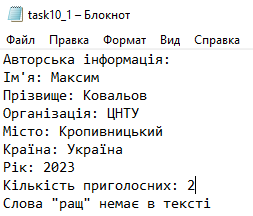
input\_f.close();

output\_f.close();

}

// 3 -------------

**Результат:**

****

**Завдання 10.2:**

**Алгоритм:**

1. Початок.
2. Створюємо функцію *f\_l\_char()* та передаємо у неї ім’я файлу.
3. Відкриваємо файлові потоки *input\_f* у режимі читання та *output\_f* у режимі дозапису(::app)
4. Створюємо змінні *string word* та *int word\_length*.
5. Перевіряємо чи відкриті файли *input\_f* та *output\_f*, якщо ні то помилка.
6. Отримуємо слово з рядка *input\_f* у *word* та визначаємо його довжину у *word\_length*
7. Записуємо у файл першу букву word[0] та останню word[word\_length - 1].
8. Закриваємо файлові потоки.
9. Створюємо функцію *time\_date()* та передаємо у неї ім’я файлу.
10. Відкриваємо файловий потік *output\_f*  у режимі дозапису(::app) та преміщаємо вказівник у кінець(::ate).
11. Перевіряємо чи відкритий файл *output\_f*, якщо ні то помилка.
12. Отримуємо поточний час, перетворюємо його та записуємо його у *output\_f.*
13. Закриваємо файловий потік output\_f.
14. Кінець

**Код:**

void f\_l\_char(const char\* filename){

std::ifstream input\_f(filename);

std::ofstream output\_f(filename, std::ios::app | std::ios::ate);

std::string word;

int word\_length;

if(input\_f && output\_f){

std::getline(input\_f, word);

word\_length = word.length();

output\_f << "\nПерша буква слова: " << word[0] << std::endl

<< "Остання буква слова: " << word[word\_length - 1] << std::endl;

}else{

std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

}

input\_f.close();

output\_f.close();

}

void time\_date(const char\* filename){

std::ofstream output\_f(filename, std::ios::app | std::ios::ate);

if(output\_f){

auto now = std::chrono::system\_clock::now();

std::time\_t now\_c = std::chrono::system\_clock::to\_time\_t(now);

output\_f << "Дата та час дозапису інформації: " << std::ctime(&now\_c);

}else{

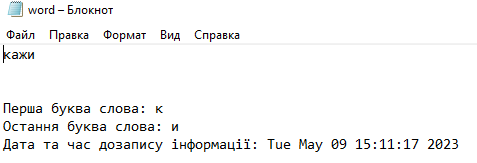
std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

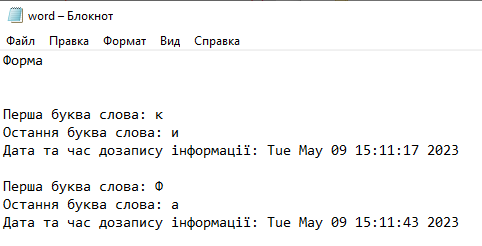
}

output\_f.close();

}

**Результат:**

****

****

**Завдання 10.3:**

**Алгоритм:**

1. Початок
2. Створюємо функцію string *decToBinary()* та передаємо у неї змінну *b* типу *int.*
3. Створюємо змінну result = “” типу string та цикл while з умовою n > 0.
4. Перевіряємо чи парне число n, якщо так то добавляємо у result “0”, інакше “1”
5. Присвоюємо число n до n та ділимо його на 2.
6. Як закінчився цикл, повертаємо result.
7. Створюємо функцію *write\_s\_calc()* та передаємо у неї ім’я файлу, *float s\_calc, string number, int b.*
8. Відкриваємо файловий потік *output\_f*  у режимі дозапису(::app) та преміщаємо вказівник у кінець(::ate).
9. Перевіряємо чи відкритий файл *output\_f*, якщо ні то помилка.
10. Записуємо у файл результат функції *s\_calculation()* та вхідне число *b* у двійковій системі вихідне *number*.
11. Закриваємо файловий потік.

**Код:**

std::string decToBinary(int n){

std::string result = "";

while (n > 0) {

if (n % 2 == 0)

result = "0" + result;

else

result = "1" + result;

n /= 2;

}

return result;

}

void write\_s\_calc(const char\* filename, float s\_calc, std::string number, int b){

std::ofstream output\_f(filename, std::ios::app | std::ios::ate);

if(output\_f){

output\_f << "Результат функції s\_calculation: " << s\_calc << std::endl;

output\_f << "Число \"" << b << "\" у двійковій системі: " << number << std::endl;

}else{

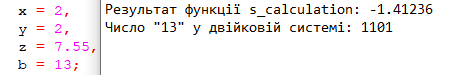
std::cerr << "Не вдалося відкрити цей файл: " << filename << std::endl;

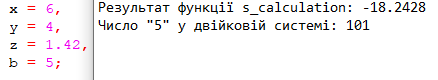
}

output\_f.close();

}

**Результат:**

****

****

**Висновок**

В ході цієї лабораторної роботи я набув ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структур, об’єднань, множин, перелік, перетворень типів даних, використання файлових потоків та функцій стандартних бібліотек для оброблення символьної інформації.