

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/>

### ВИКОНАВ

студент академічної групи

КБ-22-2 Ковальов Максим

### ПЕРЕВІРИВ

викладач кафедри кібербезпеки  
та програмного забезпечення  
Олександр Собінов

## **Лабораторна робота №10**

### **Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням**

*Мета роботи* полягає у набутті ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування C++ 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 символічної інформації.

### **Завдання до лабораторної роботи**

1. Реалізувати програмні модулі розв'язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.a (проект ModulesПрізвище лабораторних робіт №8–9).
2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв'язування задач 10.1–10.3.

## Варіант 4

— *ВХІДНИЙ ТЕКСТ - ВМІСТ ВХІДНОГО ТЕКСТОВОГО ФАЙЛУ* —  
Довільне слово 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 приголосних літер у слові із вхідного файла;
- повідомлення, чи є слово із вхідного файла у наступній крапинці Віталія Іващенко:

Про себе не кажи недобрих слів,  
Бо має сказане таємну силу.  
Кажи: «Я сильний, впевнений, щасливий!»  
І буде саме так, як ти хотів!

— *ЗАДАЧА 10.2* —

У *вихідний* текстовий файл дописати:

- першу і останню літеру слова, яке міститься у цьому файлі, а також , 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.hs_calculation` з аргументами  $x$ ,  $y$ ,  $z$ ;
- число  $b$  у двійковому коді.

## Рішення

### Завдання 10.1.

#### Алгоритм:

1. Початок
2. Самостійно створюємо у проєкті файл `word.txt` з довільним словом та файл `text10_1.txt` з рядками тексту:

Про себе не кажи недобрих слів,  
Бо має сказане таємну силу.  
Кажі: «Я сильний, впевнений, щасливий  
І буде саме так, як ти хотів!

3. Створюємо структуру даних `PersonInfo` з змінними `string name, surname, organization, city, country` та `int year`.
4. Створюємо функцію `write_info` та передаємо у функцію ім'я файлу та структуру `PersonInfo`.
5. Відкриваємо файловий потік у режимі запису `op_file`.
6. Перевіряємо чи відкритий файл `op_file`, якщо ні то виводимо помилку.
7. У файл записуємо дані зі структури та закриваємо файловий потік `op_file`.
8. Створюємо функцію `consonants_num()` та передаємо у функцію ім'я файлу.
9. Відкриваємо файловий потік у режимі читання `input_f` та створюємо рахунок приголосних зі значенням 0.
10. Перевіряємо чи відкритий файл `op_file`, якщо ні то виводимо помилку.
11. Створюємо змінну `string` з назвою `ukr_word` та передаємо у змінну перший рядок з файлу `input_f`.
12. Створюємо множину приголосних літер з назвою `consonants`.
13. Створюємо цикл `for` для перевірки кожної літери слова на приголосну з множини `consonants`.
14. Повертаємо кількість приголосних `c_count` та закриваємо `input_f`.
15. Створюємо функцію `write_consosnants_num` та передаємо у неї ім'я файлу та кількість приголосних.
16. Відкриваємо файловий потік `output_f` у режимі дозапису(`::out`) та переставленим вказівником на кінець(`::app`).
17. Перевіряємо чи відкритий файловий потік `output_f`, якщо ні то виводимо помилку.
18. Дозаписуємо у файл кількість приголосних та закриваємо файловий потік `output_f`.
19. Створюємо функцію `does_it_has_in()` та передаємо у функцію ім'я файлу.

20. Створюємо булеву змінну *found = false*, та дві змінні типу *string ukr\_word*, *word*.
21. Відкриваємо файлові потоки зі словом *word\_file* та *text10\_1.txt* у режимі читання.
22. Перевіряємо чи дійсно відкриті два файли одночас, якщо ні виводимо помилку.
23. 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*.
24. Повертаємо значення змінної *found*.  
Закриваємо файлові потоки *word\_file* та *text10\_1.txt*.
25. Створюємо функцію *write\_does\_it\_has\_in()* та передаємо у неї ім'я файлу та булеву *found*.
26. Відкриваємо файлові потоки *output\_f* у режимі дозапису(*::app*) та *input\_f("word.txt")* у режимі читання.
27. Створюємо *string ukr\_word*.
28. Перевіряємо чи відкрився *output\_f* та *input\_f* отримуємо рядок зі словом з файлу *input\_f* та передаємо у змінну *ukr\_word*.
29. Перевіряємо змінну *found* та якщо істина то слово *ukr\_word* присутнє в тексті, інакше ні.
30. Закриваємо файлові потоки *word\_file* та *text10\_1.txt*.
31. 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);

        // Створюємо 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 результату
    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:

 task10_1 – Блокнот	 task10_1 – Блокнот
Файл Правка Формат Вид Справка	Файл Правка Формат Вид Справка
Авторська інформація:	Авторська інформація:
Ім'я: Максим	Ім'я: Максим
Прізвище: Ковальов	Прізвище: Ковальов
Організація: ЦНТУ	Організація: ЦНТУ
Місто: Кропивницький	Місто: Кропивницький
Країна: Україна	Країна: Україна
Рік: 2023	Рік: 2023
Кількість пригосних: 2	Кількість пригосних: 2
Слова "рац" немає в тексті	Слово "кажи" присутнє в тексті

## 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:

word – Блокнот  
Файл Правка Формат Вид Справка  
кажи

Перша буква слова: к  
Остання буква слова: и  
Дата та час запису інформації: Tue May 09 15:11:17 2023

word – Блокнот  
Файл Правка Формат Вид Справка  
Форма

Перша буква слова: к  
Остання буква слова: и  
Дата та час запису інформації: Tue May 09 15:11:17 2023

Перша буква слова: Ф  
Остання буква слова: а  
Дата та час запису інформації: Tue May 09 15:11:43 2023

## 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 додаємо 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:

```
x = 2, Результат функції s_calculation: -1.41236
y = 2, Число "13" у двійковій системі: 1101
z = 7.55,
b = 13;
```

```
x = 6, Результат функції s_calculation: -18.2428
y = 4, Число "5" у двійковій системі: 101
z = 1.42,
b = 5;
```

## **Висновок**

В ході цієї лабораторної роботи я набув ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування C++ 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.