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

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки та програмного забезпечення
Доренський О. П.

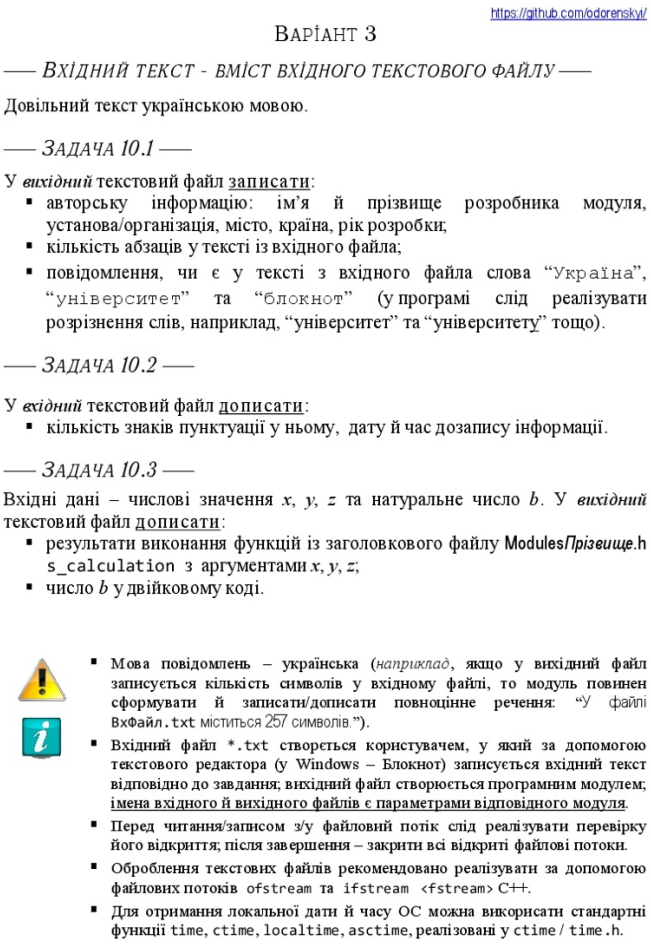
ВИКОНАВ
студент академічної групи КБ-23
Литвин М. В.

ПЕРЕВІРИВ
викладач кафедри кібербезпеки та програмного забезпечення
Дрєєва Г.М.

Кропивницький – 2024

Мета: Набуття 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 символьної інформації.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ



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`.

Варіант 3

Алгоритм

1. очаток програми:
    - Імпортувати необхідні бібліотеки та файл ModulLytvyn.h.
    - Визначити головні функції та структури даних.
    - Вивести привітання або інформацію про розробника.
2. Введення основного меню:
    - Показати користувачеві список опцій:
        1. Обчислення.
        2. Обробка файлів.
        3. Вихід з програми.
3. Обробка вибору користувача:
    - Зчитати вибір користувача.

- Виконати відповідну дію залежно від обраної опції:
  - Якщо обрано опцію "Обчислення", перейти до кроку 4.
  - Якщо обрано опцію "Обробка файлів", перейти до кроку 5.
  - Якщо обрано опцію "Вихід з програми", завершити виконання програми.

4. Обчислення:
   - Запросити введення необхідних даних для обчислення.
   - Викликати функцію s_calculation для обчислення значення s.
   - Вивести результат обчислень.
5. Обробка файлів:
   - Зчитати імена вхідного та вихідного файлів.
   - Передати їх до відповідної функції для обробки файлів.
   - Показати результати обробки файлів.
6. Повторення:
   - Повернутися до кроку 2, якщо користувач не обрав виходу з програми.
7. Завершення програми:
   - Вивести прощальне повідомлення.
   - Завершити виконання програми.

Лістинг коду програми з коментарями

```
#include <iostream> // Підключення бібліотеки введення-виведення
#include <clocale> // Підключення стандартних функцій локалі
#include <ctime> // Підключення функцій для роботи з часом
#include <cmath> // Підключення математичних функцій
#include <limits> // Підключення обмежень числових типів
#include <string> // Підключення бібліотеки рядків
#include <fstream> // Підключення бібліотеки файлового введення-виведення
#include <bitset> // Підключення бібліотеки для роботи з бітовими полями
#include <codecvt> // Підключення бібліотеки для конвертації символів
#include "ModulLytvyn.h" // Підключення користувацького заголовкового файлу "ModulLytvyn.h"

using namespace std; // Використання стандартного простору імен

void SoftwareDeveloper() // Оголошення функції

void s_calculation(char choice); // Оголошення функції з параметром

// Функція для відображення інформації про розробника
void s_calculation(int x, int y, int z) {
    system("chcp 1251"); // Встановлення кодування консолі для відображення українських символів
    system("cls"); // Очищення екрану консолі
```

```
    setlocale(LC_ALL, "ukr"); // Встановлення локалізації на українську
    cout << "\n ------------------------------------------------\n"
         << "  | Maksym Lytvyn, CUNTU, opd@kntu.kr.ua |"
         << "  | Максим Литвин, ЦНТУ, opd@kntu.kr.ua |  "
         << "\n   ----------- © All Rights Reserved ---------------\n\n\n\n";
}

using namespace std; // Повторне використання стандартного простору імен

wstring characters = L".,!?:;)\" (@'№%*_-+=\\/0123456789абвгґдеєжзиіїйклмнопрстуфхцчшщьюя"; // Набір символів, включаючи українські літери та пунктуацію

// Функція для обчислення значення s за заданими параметрами
float s_calculation(int x, int y, int z) {
      return (1.0/2.0)*pow(fabs(2*z-pow(x,2))/sin(x),3)/sqrt(1+fabs(cos(x))+2*3.14*y); // Обчислення s
}

// Функція для розрахунку фінансового депозиту
Deposit getPayment(float value, int months) {
      Deposit deposit;
      if (months == 6 || months == 12) { // Перевірка дійсності тривалості депозиту
            float interest;
            deposit.totalInterest = months == 12 ? 13 : 11/(float)2; // Обчислення загального відсотка
            interest = deposit.totalInterest / 100 / months; // Обчислення щомісячного відсотка
            deposit.monthlyInterestPaid = round(value*interest*100)/100; // Обчислення щомісячної виплати відсотків
      } else {
            deposit.totalInterest = -1; // Позначення недійсного відсотка
            deposit.monthlyInterestPaid = -1; // Позначення недійсної щомісячної виплати
      }
      return deposit;
}

// Функція для визначення розміру одягу за словацькою системою
Size getSize(int slovakSize) {
      Size size;
      if (slovakSize >= 6 && slovakSize <= 10) { // Перевірка дійсності словацького розміру
            size.french = slovakSize - 4; // Перетворення на французький розмір
            switch (slovakSize) // Визначення міжнародного розміру
```

```
            {
                  case 6:
                        size.international = "S";
                        break;
                  case 7:
                        size.international = "M";
                        break;
                  case 8:
                        size.international = "L";
                        break;
                  case 9:
                        size.international = "XL";
                        break;
                  case 10:
                        size.international = "XXL";
                        break;
                  default:
                        size.international = "Error"; // Позначення помилки для
недійсного словацького розміру
            }
      } else {
            size.french = -1; // Позначення помилки для недійсного словацького
розміру
            size.international = "Error"; // Позначення помилки для недійсного
словацького розміру
      }
      return size;
}

// Функція для підрахунку кількості встановлених бітів у числі
int t9_3(int number) {
      unsigned int count = 0;
      if (number > 0 || number <= 7483650) { // Перевірка дійсності числа
            bool set = number & 1;
            while (number) {
                  count += (number & 1) == set; // Інкрементування лічильника для
кожного встановленого біту
                  number >>= 1; // Зсув числа вправо
            }
      } else {
            return -1; // Позначення помилки для числа, що виходить за межі
діапазону
      }
      return count; // Повернення кількості встановлених бітів
}
```

```
// Функція для перевірки, чи належить символ до заданого набору
bool validateCharacter(wchar_t character) {
	for (int i = 0; i < characters.length(); i++)
		if (character == characters.at(i))
			return true; // Повернення true, якщо символ дійсний
	return false; // Повернення false, якщо символ не дійсний
}

// Функція для обробки вхідного файлу та запису результатів у вихідний файл
int t10_1(string inputFile, string outputFile) {
	wstring line;
	wstring words[4] = {L"програма", L"модуль", L"студент", L"програміст"}; // Масив цільових слів
	bool found = false; // Прапорець, що позначає, чи знайдені цільові слова
	int number = 0; // Лічильник символів у файлі
	wifstream indata;
	indata.open(inputFile);
	ofstream outdata;
	outdata.open(outputFile);

	if (!indata || !outdata)
		return 1; // Позначення помилки для невдалого відкриття файлу

	indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування вхідного файлу UTF-8

	// Обробка кожного рядка у вхідному файлі
	while (getline(indata,line)) {
		number += line.length(); // Інкрементування лічильника символів

		// Перетворення кожного символу на нижній регістр та перевірка його дійсності
		for (int i = 0; i < line.length(); i++) {
			line[i] = towlower(line[i]);
			if (!validateCharacter(line[i]))
				return 2; // Позначення помилки для недійсного символу
		}

		// Пошук цільових слів у кожному рядку
		if (!found) {
			for (int i = 0; i < 4; i++) {
				int index = line.find(words[i]);
				if (index != wstring::npos && line.length()-words[i].length()-index == 0) { // Перевірка, чи слово знаходиться в кінці рядка
```

```
                        found = true;
                        goto out_found;
                    }

                    // Пошук цільових слів, за якими слідують певні символи
                    for (int j = 0; j < 9; j++) {
                        wstring wordToFind = words[i] + characters.at(j);
                        if (line.find(wordToFind) != wstring::npos) {
                            found = true;
                            goto out_found;
                        }
                    }
                }
            }
            out_found: continue; // Продовження обробки наступного рядка
        }

        // Запис інформації про розробника та кількості символів у вихідний файл
        outdata << "Розробник: Гончарук Олександр" << endl << "Установа/організація: Центральноукраїнський національний технічний університет" << endl << "Місто: Кропивницький" << endl << "Країна: Україна" << endl << "Рік розробки: 2023" << endl << endl;

        outdata << "Кількість символів у файлі: " << number << endl;

        outdata << "У вхідному файлі " << ((found) ? "наявні" : "відсутні") << " слова \"Україна\", \"університет\", \"блокнот\"." << endl;

        return 0;
}

// Функція для обробки вхідного файлу та підрахунку кількості цифр
int t10_2(string file) {
        wchar_t character;
        time_t rawtime;
        time(&rawtime);
        int number = 0; // Лічильник цифр у файлі

        wifstream indata;
        indata.open(file);
        ofstream outdata;
        outdata.open(file, ios_base::app);

        if (!indata || !outdata)
            return 1; // Позначення помилки для невдалого відкриття файлу
```

```
	indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування вхідного файлу UTF-8

	while (indata >> character) {
		if (!validateCharacter(towlower(character)))
			return 2; // Позначення помилки для недійсного символу
		for (int i = 0; i < 10; i++)
			if (character == characters[i + 22]) // Перевірка наявності цифри
				number += 1; // Інкрементування лічильника
	}

	outdata << endl << "Кількість цифр у файлі: " << number << endl;
	outdata << "Дата й час дозапису: " << ctime(&rawtime); // Запис дати та часу дозапису

	return 0;
}

// Функція для обробки вхідних даних та запису результатів у вихідний файл
int t10_3(string file, int x, int y, int z, int b) {
	wchar_t character;
	time_t rawtime;
	time(&rawtime);
	int number = 0;

	ofstream data;
	data.open(file, ios_base::app);

	if (!data)
		return 1; // Позначення помилки для невдалого відкриття файлу

	data << endl << "s = " << s_calculation(x, y, z) << endl; // Запис результату обчислення s
	if (b > 0) {
		string binary = bitset<numeric_limits<int>::digits>(b).to_string(); // Перетворення числа у двійковий формат
		binary.erase(0, binary.find_first_not_of('0')); // Видалення зайвих нулів з початку рядка
		data << "b у двійковому коді: " << binary << endl; // Запис у вихідний файл у двійковому форматі
	}
	else
```

```
            data << "b — не натуральне число" << endl; // Повідомлення про недійсність числа b

      return 0;
}
```

Алгоритм:

1. Створення вхідного файлу: Функція createInput призначена для створення вхідного текстового файлу з переданим їй вмістом. Вона відкриває файл і записує вміст у нього. Якщо операція відкриття файлу не вдалася, вона повертає false, в іншому випадку - true.
2. Тестування функцій модуля: Кожна з функцій test1_1, test1, test2_1, test2, test3 призначена для тестування відповідної функції з модуля ModulLytvyn. Кожна з них викликає відповідну функцію тестування, записує результати у вихідний файл та перевіряє, чи вони правильні.
3. Головна функція main: У цій функції описано основний процес тестування. Вона містить масиви вхідних даних для тестів та запускає цикли для тестування різних функцій з різними вхідними даними. Результати тестів виводяться на екран.

Лістинг програми з коментарем:

```
#include <iostream> // Підключення бібліотеки для введення та виведення даних
#include <locale> // Підключення бібліотеки для роботи з локалями
#include <fstream> // Підключення бібліотеки для роботи з файлами
#include <codecvt> // Підключення бібліотеки для кодування та декодування тексту
#include <string> // Підключення бібліотеки для роботи з рядками
#include "ModulLytvyn.h" // Підключення власного модуля

using namespace std; // Використання простору імен стандартної бібліотеки

string input = "input.txt"; // Змінна для назви вхідного файлу
string output = "output.txt"; // Змінна для назви вихідного файлу

// Функція для створення вхідного файлу з заданим вмістом
bool createInput(wstring content) {
    wofstream data; // Об'єкт для запису у файл
    data.open(input); // Відкриття вхідного файлу

    data.imbue(locale(locale(),  new  codecvt_utf8<wchar_t>));  //  Встановлення кодування UTF-8
```

```
    if (!data) { // Перевірка на успішне відкриття файлу
        cout << "Не вдається створити вхідний файл." << endl; // Вивід повідомлення про помилку
        return false; // Повернення значення false
    }

    data << content << endl; // Запис вмісту у файл

    return true; // Повернення значення true
}

// Функція для перевірки результату тестування 1.1
bool test1_1() {
    wstring line; // Змінна для збереження рядка з файлу
    const wstring lines[3] = {L"Розробник: Гончарук Олександр", L"Установа/організація: Центральноукраїнський національний технічний університет", L"Рік розробки: 2023"}; // Масив з очікуваними рядками
    bool linesFound[3] = {false, false, false}; // Масив для позначення знайдених рядків
    int currentLine = 0; // Змінна для відстеження поточного рядка

    if(!createInput(L"Тест")) // Створення вхідного файлу
        return false; // Повернення значення false

    t10_1(input, output); // Виклик функції для тестування

    wifstream indata; // Об'єкт для зчитування з файлу
    indata.open(output); // Відкриття вихідного файлу

    indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування UTF-8

    while (getline(indata,line)) { // Цикл читання рядків з файлу
        if (line.find(lines[currentLine]) != wstring::npos) { // Перевірка наявності поточного рядка у файлі
            linesFound[currentLine] = true; // Позначення знайденого рядка
            currentLine++; // Перехід до наступного рядка
        }
        if (linesFound[0] && linesFound[1] && linesFound[2]) // Перевірка, чи всі потрібні рядки знайдено
            return true; // Повернення значення true
    }
    return false; // Повернення значення false, якщо не всі рядки знайдено
}
```

```
// Функція для тестування 1
bool test1(wstring data, bool wordPresent) {
    wstring line; // Змінна для збереження рядка з файлу

    if(!createInput(data)) // Створення вхідного файлу
        return false; // Повернення значення false

    t10_1(input, output); // Виклик функції для тестування

    wifstream indata; // Об'єкт для зчитування з файлу
    indata.open(output); // Відкриття вихідного файлу

    indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування UTF-8

    while (getline(indata,line)) { // Цикл читання рядків з файлу
        if ((wordPresent && line.find(L"Програма") != wstring::npos) || (!wordPresent && line.find(L"Модуль") != wstring::npos)) // Перевірка наявності слова у файлі
            return true; // Повернення значення true
    }

    return false; // Повернення значення false, якщо слово не знайдено
}

// Функція для тестування 2.1
bool test2_1() {
    wstring line; // Змінна для збереження рядка з файлу

    if(!createInput(L"Тест")) // Створення вхідного файлу
        return false; // Повернення значення false

    t10_2(input); // Виклик функції для тестування

    wifstream indata; // Об'єкт для зчитування з файлу
    indata.open(input); // Відкриття вихідного файлу

    indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування UTF-8

    while (getline(indata,line)) { // Цикл читання рядків з файлу
        if (line.find(L"Кількість цифр у файлі: ") != wstring::npos) // Перевірка наявності певного рядка у файлі
            return true; // Повернення значення true
    }
```

```
    return false; // Повернення значення false, якщо рядок не знайдено
}

// Функція для тестування 2
bool test2(wstring data, int digits) {
    wstring line; // Змінна для збереження рядка з файлу

    if(!createInput(data)) // Створення вхідного файлу
        return false; // Повернення значення false

    t10_2(input); // Виклик функції для тестування

    wifstream indata; // Об'єкт для зчитування з файлу
    indata.open(input); // Відкриття вихідного файлу

    indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування UTF-8

    while (getline(indata,line)) { // Цикл читання рядків з файлу
        if (line.find(L"Кількість цифр у файлі: " + to_wstring(digits)) != wstring::npos) // Перевірка наявності рядка з певною кількістю цифр
            return true; // Повернення значення true
    }

    return false; // Повернення значення false, якщо рядок не знайдено
}

// Функція для тестування 3
bool test3(int x, int y, int z, int b, wstring s, wstring bin) {
    wstring line; // Змінна для збереження рядка з файлу
    bool firstValid = false; // Прапорець для позначення першого правильного рядка

    t10_3(output, x, y, z, b); // Виклик функції для тестування

    wifstream indata; // Об'єкт для зчитування з файлу
    indata.open(output); // Відкриття вихідного файлу

    indata.imbue(locale(locale(), new codecvt_utf8<wchar_t>)); // Встановлення кодування UTF-8

    while (getline(indata,line)) { // Цикл читання рядків з файлу
        if (!firstValid && (line.find(L"s = " + s) != wstring::npos)) // Перевірка наявності першого правильного рядка
            firstValid = true; // Позначення першого правильного рядка
```

```
        else if (firstValid && (line.find(L"b у двійковому коді: " + bin) != wstring::npos)) // Перевірка наявності другого правильного рядка
            return true; // Повернення значення true
    }

    return false; // Повернення значення false, якщо рядки не знайдено
}

int main() {
    const wstring data1_1[] = {L"Програма", L"Модуль", L"Студент", L"Програміст"}; // Масив вхідних даних для тесту 1
    const bool wordPresent[] = {true, false, true, false}; // Масив для вказівки наявності слова у вхідних даних

    const wstring data2_1[] = {L"Привіт", L"0123456789", L"12", L"00000"}; // Масив вхідних даних для тесту 2
    const int digits[] = {0, 10, 2, 5}; // Масив для вказівки кількості цифр

    const int x[] = {7, 2, 9, 0, 7}; // Масив значень x
    const int y[] = {2, 45, 0, 0, 5}; // Масив значень y
    const int z[] = {1, 6, 1, 0, 4}; // Масив значень z
    const int b[] = {1, 3, 8, 100, 127}; // Масив значень b
    const wstring s[] = {L"7.84054", L"268.187", L"3.84147", L"1", L"35775.4"}; // Масив значень s
    const wstring bin[] = {L"1", L"11", L"1000", L"1100100", L"1111111"}; // Масив двійкових значень b

    setlocale(LC_ALL, ""); // Встановлення локалі для коректного виводу

    for (int i = 0; i < 3; i++) { // Цикл по всім тестам
        for (int j = 0; j < 5; j++) { // Цикл по всім вхідним даним
            bool value = false; // Значення результату тесту
            wstring description; // Опис тесту

            if (i < 2 && j == 0) { // Перевірка на тест 1.1 та 2.1
                description = L"Власний тест"; // Опис тесту
                if (i == 0) // Вибір функції для тестування
                    value = test1_1(); // Виклик функції для тестування
                else if (i == 1)
                    value = test2_1(); // Виклик функції для тестування
            }
            else if (i == 0) { // Перевірка на тест 1
                value = test1(data1_1[j-1], wordPresent[j-1]); // Виклик функції для тестування
```

```
                description = L"Вхідні дані: " + data1_1[j-1] + L", Очікуваний результат: " + (wordPresent[j-1] ? L"Присутнє" : L"Відсутнє"); // Опис тесту
            }
            else if (i == 1) { // Перевірка на тест 2
                value = test2(data2_1[j-1], digits[j-1]); // Виклик функції для тестування
                string tmpstr = to_string(digits[j-1]); // Перетворення числа в рядок
                wstring wstr(tmpstr.begin(), tmpstr.end()); // Перетворення рядка у wchar_t
                description = L"Вхідні дані: " + data2_1[j-1] + L", Очікувана кількість: " + wstr; // Опис тесту
            }
            else if (i == 2) { // Перевірка на тест 3
                value = test3(x[j], y[j], z[j], b[j], s[j], bin[j]); // Виклик функції для тестування
                string xtmpstr = to_string(digits[j-1]); // Перетворення числа в рядок
                wstring xwstr(xtmpstr.begin(), xtmpstr.end()); // Перетворення рядка у wchar_t
                string ytmpstr = to_string(digits[j-1]); // Перетворення числа в рядок
                wstring ywstr(ytmpstr.begin(), ytmpstr.end()); // Перетворення рядка у wchar_t
                string ztmpstr = to_string(digits[j-1]); // Перетворення числа в рядок
                wstring zwstr(ztmpstr.begin(), ztmpstr.end()); // Перетворення рядка у wchar_t
                string btmpstr = to_string(digits[j-1]); // Перетворення числа в рядок
                wstring bwstr(btmpstr.begin(), btmpstr.end()); // Перетворення рядка у wchar_t
                description = L"Вхідні дані: x = " + xwstr + L", y = " + ywstr + L", z = " + zwstr + L", b = " + bwstr + L", Очікуваний результат: s = " + s[j] + L", b у двійковому коді: " + bin[j]; // Опис тесту
            }

            wcout << "Тест " << i + 1 << "." << j + 1 << " (" << description << ") " << (value ? L"пройдений" : L"не пройдений") << endl; // Виведення результату тестування
        }
    }

    return 0; // Повернення значення 0 в кінці програми
}
```

Висновок

Завантажений власний Git-репозиторій https://github.com/odorenskyi/Lytvyn-Maksym-KB23. У \Lab10 заповнено файл README.md, створено теки prj, Software, TestSuite, Report. До звіту включено тему мету завдання. Здійснено аналіз і постановку задач 10.1, 10.2, 10.3. Виконано аналіз вимог, проектування архітектури, детальне проектування програмних модулів розв’язування задач 10.1, 10.2, 10.3. Розроблено набори контрольних прикладів до задач 10.1, 10.2, 10.3 задля виконання модульного тестування (Unit testing) модулів C++. В Code::Blocks IDE відкрито проект статичної бібліотеки ModulLytvyn з \Lab8\prj, створений під час виконання лабораторної роботи №8. За отриманими під час проектування програмних модулів артефактами виконано конструювання функцій: мовою програмування C++ реалізовано функції, які за наданим інтерфейсом реалізовують розв’язування задач 10.1, 10.2 та 10.3 відповідно. Реалізовано тестовий драйвер для виконання розроблених тестових наборів до задач 10.1-10.3 і за його допомогою виконано модульне тестування функцій зі статичної бібліотеки. Результати unit-тестування задокументувано. В ході роботи над лабораторною роботою набув ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування, застосування операторів C/C++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.