

## 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.

### ЧАС ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

4 академічні години.

### ОБЛАДНАННЯ, МАТЕРІАЛИ, ПРОГРАМНІ ЗАСОБИ




Для виконання лабораторної роботи необхідні:

- персональний комп'ютер з ОС Windows XP / Vista / 7 / 8.x / 10, Linux 32-bit / 64-bit або Mac OS X;
- вільне кросплатформове середовище розроблення програмного забезпечення Code::Blocks ([www.codeblocks.org](http://www.codeblocks.org)) для платформи Windows XP / Vista / 7 / 8.x / 10, 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.a` (проект `ModulesПрізвище` лабораторних робіт №8–9).
2. Реалізувати *тестовий драйвер* автоматизованої перевірки програмних модулів розв'язування задач 10.1–10.3.

### Порядок виконання лабораторної роботи ТА МЕТОДИЧНІ ВКАЗІВКИ

1. Завантажити власний Git-репозиторій <https://github.com/odorenskyi/student-name> (в `\Lab4\tasks` містяться умови задач 10.1–10.3)
2. У `\Lab10` заповнити файл `README.md`, створити теки `prj`, `TestSuite`, `Software`, `Report`; отриманий вміст теки `\Lab10` завантажити до Git-репозиторію <https://github.com/odorenskyi/student-name>; надалі здійснювати означену дію за позначкою .
3. До звіту з лабораторної роботи (далі — звіт) записати мету роботи, номер варіанту, завдання.
4. Почергово виконати аналіз і постановку задач 10.1, 10.2, 10.3, аналіз вимог до ПЗ та вмісту вхідного текстового файлу (див. умови задач), проектування архітектури, детальне проектування програмних модулів; одержані артефакти задокументувати й включити до звіту 
  - під час проектування функцій слід врахувати, що вхідні дані модуля – ім'я вхідного і/або вихідного файлу; якщо вихідний файл існує, то його вміст знищується, інакше – файл створюється.
5. Розробити *три* тест-сьюти (набори контрольних прикладів) задля проведення автоматизованого unit-тестування програмних модулів розв'язування задач 10.1–10.3; отримані тестові артефакти задокументувати (наприклад, у вигляді таблиці), зберегти у `\Lab10\TestSuite` та включити до звіту 
  - рекомендовано таку структуру тест-кейса: Preliminary Steps – ім'я вхідного файлу та його вміст (текст) і/або ім'я вихідного файлу; Action (test steps) – виклик відповідного модуля з вхідними даними (значення аргументів); Expected Result – вміст вхідного/вихідного текстового файлу.
6. В Code::Blocks IDE відкрити проект статичної бібліотеки `ModulesПрізвище` з `\Lab8\prj`, створений під час виконання лабораторної роботи №8.
7. За отриманими під час проектування програмних модулів артефактами виконати конструювання функцій: мовою програмування C++ реалізувати функції, які за наданим інтерфейсом

- реалізують розв'язування задач 10.1, 10.2 та 10.3 відповідно;
- проект *ModulesПрізвище* доповнюється новими функціями.
8. Скомпілювати проект статичної бібліотеки *ModulesПрізвище*.
  9. Відкрити проект заголовкового файлу *ModulesПрізвище*, створений під час виконання лабораторної роботи № 8, та доповнити його прототипами реалізованих функцій 10.1–10.3.
  10. У середовищі Code::Blocks створити в теці \Lab10\prj проект консольного додатка, іменувати його *TestDriver*.
  11. Мовою програмування C++ реалізувати консольний застосунок – тестовий драйвер для *модульного* тестування функцій розв'язування задач 10.1–10.3 за допомогою розроблених тест-сьютів з \Lab10\TestSuite та вхідного і/або вихідного текстового файлу.
    - необхідно реалізувати протоколювання процесу тестування: запис у файл вхідних даних (аргументів функції, яка тестується), отриманий результат та статус тест-кейса (passed або failed);
    - назва вхідного/вихідного файлу(ів) передається відповідним функціям з *ModulesПрізвище.h* як аргументи;
    - для включення функцій бібліотеки *libModulesПрізвище.a* до вихідного коду драйвера слід використати заголовковий файл *ModulesПрізвище.h* та належно налаштувати опції компілятора (Build options...);
    - контрольні приклади рекомендовано представити константними масивами, елементи яких у циклі передаються на оброблення відповідною функцією з *ModulesПрізвище.h*;
    - є можливим реалізація тестовим драйвером або повністю автоматизованого, або напівавтоматизованого *unit*-тестування:
      - для *автоматизованого* слід реалізувати алгоритм виконання відповідного тест-кейса: *A1*) драйвер за *Preliminary Steps* створює вхідний файл та записує в нього вхідний текст і/або створює вихідний файл, *A2*) за *Action* викликається функція з аргументами – ім'я/іменами файлів, *A3*) відкривається модифікований функцією з кроку *A2* файл, зчитується текст з нього та порівнюється з текстом із поля *Expected Result*; *A4*) результат порівняння (*Test Result passed / failed*) виводиться у стандартний файловий потік;
      - для *напівавтоматизованого* кроки *A1*, *A3* та *A4* алгоритму автоматизованого виконання тест-кейса реалізуються тестувальником вручну за допомогою текстового редактора, а крок *A2* — тестовим драйвером.
  12. Створений застосунок *TestDriver.exe* перемістити у \Lab10\Software.
  13. За допомогою *TestDriver.exe* виконати автоматизоване тестування розроблених функцій розв'язування задач 10.1–10.3.

- у випадку *невиконання* тест-кейса(ів) слід виконати відлагодження відповідної функції, перекомпілювати проект статичної бібліотеки *ModulesПрізвище*, модульне тестування повторити.
14. Результати модульного тестування відповідних функцій статичної бібліотеки *libModulesПрізвище.a* задокументувати шляхом включення (копіювання) результатів роботи тестового драйвера \Lab4\Software\TestDriver.exe до звіту.
  15. Вихідний код (текст) проектів *ModulesПрізвище* та *TestDriver* включити до звіту як додатки.
  16. Проаналізувати хід виконання лабораторних завдань і самостійно одержані результати, на основі чого сформулювати обґрунтовані висновки<sup>1</sup> з виконаної лабораторної роботи, викласти їх обсягом *не менше двох сторінок* машинного (комп'ютерного) тексту та включити до звіту.
  17. Підготувати й зберегти у \Lab10\Report звіт про виконання лабораторної роботи, оформлений згідно з ДСТУ 3008:2015 “Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання”, та зі змістом, визначеним цим порядком виконання лабораторної роботи.
  18. Представити до захисту звіт з виконаної лабораторної роботи і проект у Git-репозиторії [https://github.com/odorenskyi/student\\_name](https://github.com/odorenskyi/student_name).

### КОНТРОЛЬНІ ЗАПИТАННЯ І ЗАВДАННЯ

1. Яке призначення та синтаксис запису блоку-контроля *try - throw - catch* у мові програмування C++?
2. Наведіть приклад опису й використання міжмодульної змінної.
3. Яку область видимості матимуть об'єкти (змінні, типи, константи тощо), описані в тілі функції *main* C++?
4. Здійсніть порівняльний аналіз змінної типу *enum* та масиву.

<sup>1</sup> *висновки*, як результат розумової діяльності студента, повинні, зокрема, містити стислий виклад самостійно здобутих результатів в процесі виконання завдань, реалізованих власних проектних рішень, шляхи вирішення проблем, які виникли під час виконання лабораторної роботи; окремим абзацом слід конкретизовано викласти висновок про досягнення мети лабораторної роботи; структура висновків має бути логічною і охоплювати весь процес виконання лабораторної роботи.