ПЕРЕДМОВА

Важливою складовою навчального процесу є лабораторні заняття, в ході яких здобувачі вищої освіти у спеціально обладнаних лабораторіях та під керівництвом викладача виконують експериментально-дослідницьку роботу в межах навчальної дисципліни, набувають ґрунтовних практичних навичок за спеціальністю.

Якщо під час лекцій студентам надаються основи науково-теоретичних знань, формується науковий світогляд, розкриваються найбільш складні питання матеріалу дисципліни "Базові методології та технології програмування", то лабораторні заняття спрямовані на поглиблення й уточнення знань, здобутих на лекціях і в процесі самостійної роботи, формування інтелектуальних умінь та навичок аналізу, узагальнення, планування, опанування ІТ-засобів тощо.

В забезпеченні означеного ключову роль відіграють організаційно-методична складова навчального процесу та відповідальне ставлення до нього студента. Відповідно до Положення про організацію освітнього процесу у ЦНТУ, лабораторне заняття включає проведення інструктажу з техніки безпеки, поточного **контролю підготовленості студента** до виконання конкретної лабораторної роботи, тобто завдань з теми заняття, оформлення індивідуального звіту з виконаної роботи та його захист перед науково-педагогічним працівником. Водночас, в межах самостійної роботи **у вільний від аудиторних занять час студент зобов'язаний**, зокрема, опрацювати матеріал відповідної теми навчальної дисципліни, літературу, першоджерела, тобто належно підготуватись до виконання лабораторної роботи.

Отже, слід **ретельно готуватись до *кожного* заняття**. Підготовка до чергової лабораторної роботи здійснюється здобувачем вищої освіти самостійно з обов'язковим опрацюванням навчальної, довідникової, наукової літератури задля ґрунтовного вивчення теоретичних положень дисципліни "Базові методології та технології програмування", винесених на лабораторну роботу, а також самоконтролем підготовленості до виконання завдань за темою заняття.

ЛАБОРАТОРНА РОБОТА № 9

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ РОЗГАЛУЖЕНИХ ТА ІТЕРАЦІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи полягає у набутті 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.

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) для відповідної 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.);
- файл-шаблон тестового набору Artifact_TEST_SUITE_lab.doc;
- Git-репозиторій https://github.com/odorenskyi/*student-name*.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Реалізувати функції розв'язування задач 9.1–9.3 як складових статичної бібліотеки libModules*Прізвище*.a (проект Modules*Прізвище*, створений під час виконання лабораторної роботи № 8).
2. Реалізувати програмне забезпечення розв'язування задачі 9.4 на основі функцій статичної бібліотеки libModules*Прізвище*.a.

Порядок виконання лабораторної роботи та методичні вказівки

1. Завантажити власний Git-репозиторій https://github.com/odorenskyi/*student-name* (в \Lab9\tasks містяться умови задач 9.1–9.4).
2. У \Lab9 заповнити файл README.md, створити теки prj, TestSuite, Software, Report; отриманий вміст теки \Lab9 завантажити до Git-репозиторію https://github.com/odorenskyi/*student-name*; надалі здійснювати означену дію за позначкою.
3. До звіту з лабораторної роботи (далі — звіт) включити мету роботи, номер варіанту, завдання.
4. Почергово здійснити аналіз і постановку задач 9.1, 9.2 та 9.3.
5. Виконати аналіз вимог, проектування архітектури, детальне проектування *програмних модулів* розв'язування задач 9.1–9.3; одержані артефакти задокументувати й включити до звіту.
6. Розробити три набори контрольних прикладів до задач 9.1–9.3 для виконання *модульного тестування* (*Unit Testing*) функцій; отримані результати задокументувати та включити до звіту.
7. В Code::Blocks IDE відкрити проект статичної бібліотеки Modules*Прізвище* з \Lab8\prj, створений під час виконання лабораторної роботи № 8.
8. За отриманими під час проектування програмних модулів артефактами виконати конструювання функцій: мовою програмування C++ реалізувати функції, які за наданим інтерфейсом реалізовують розв'язування задач 9.1, 9.2 та 9.3 відповідно;
   - проект статичної бібліотеки Modules*Прізвище*, який вже містить реалізовану функцію `s_calculation`, доповнюється функціями розв'язування задач 9.1, 9.2 та 9.3.
9. Скомпілювати проект статичної бібліотеки Modules*Прізвище*.
10. Відкрити проект *заголовкового файлу* Modules*Прізвище* із \Lab8\prj, створений під час виконання лабораторної роботи № 8, та доповнити його прототипами (заголовками) функцій розв'язування задач 9.1–9.3.

11. У середовищі Code::Blocks створити в теці \prj проект консольного застосунку, іменувати його TestDriver.
12. Реалізувати тестовий драйвер для виконання *модульного* тестування функцій розв'язування задач 9.1, 9.2 та 9.3 за допомогою розроблених наборів контрольних прикладів.
    - для включення функцій бібліотеки libModules*Прізвище*.a до вихідного коду драйвера слід використати заголовковий файл Modules*Прізвище*.h та належно налаштувати опції компілятора (Build options…);
    - необхідно реалізувати протоколювання процесу тестування тестовим драйвером: виведення вхідних даних (аргументів функції, яка тестується), отриманий від функції результат та статус кожного тест-кейса (passed або failed).
13. За допомогою розробленого тестового драйвера (застосунку TestDriver.exe) виконати модульне тестування функцій розв'язування задач 9.1–9.3 з бібліотеки libModules*Прізвище*.a;
    - у випадку невиконання тест-кейса(ів) слід відлагодити проект Modules*Прізвище* (зазвичай з повторним виконанням детального проектування і/або конструювання), після чого процес модульного тестування повторити.
14. Результати unit-тестування задокументувати шляхом включення (копіювання) результатів роботи TestDriver.exe до звіту;
    - текст з консольного вікна додатка у буфер операційної системи Windows можна скопіювати за допомогою контекстого меню, попередньо виділивши його.
15. Вихідний код проектів Modules*Прізвище* та TestDriver включити до звіту як додатки.
16. Здійснити аналіз і постановку задачі 9.4.
17. Виконати аналіз вимог до ПЗ, проектування архітектури, детальне проектування програмного забезпечення розв'язування задачі 9.4; отримані артефакти задокументувати й включити до звіту.
18. Розробити тест-сьют для виконання *системного* тестування ПЗ задачі 9.4, який повинен складатись з достатньої кількості тест-кейсів (є допустимим використання простої структури тестового випадка: Test Case ID → Action (test steps) → Expected Result → Test Result (passed/failed/blocked); отриманий

тестовий артефакт (файл з тестовим набором) зберегти у \TestSuite.

19. В Code::Blocks IDE створити у теці \prj проект консольного застосунку *Прізвище*_task.
20. Мовою програмування C++ реалізувати артефакти проектування програмного забезпечення розв'язування задачі 9.4 з використанням функцій статичної бібліотеки libModules*Прізвище*.a (заголовковий файл Modules*Прізвище*.h), скомпілювати проект.
21. Виконати *системне* тестування створеного *Прізвище*_task.exe за допомогою тестового набору із \TestSuite, результати тестування задокументувати й включити до звіту як додаток;
    - у випадку невиконання тест-кейса(ів) слід виконати відлагодження проекта, після чого тестування ПЗ повторити.
22. Розроблений *Прізвище*_task.exe скопіювати у \Software, вихідний код проекта *Прізвище*_task включити до звіту як додаток.
23. Проаналізувати хід виконання лабораторних завдань і самостійно одержані результати, на основі чого сформулювати обґрунтовані висновки[1] з виконаної лабораторної роботи, викласти їх обсягом не менше двох сторінок машинного (комп'ютерного) тексту та включити до звіту;
    - у висновках (підсумках) варто також зазначити особисті враження від процесу виконання завдань лабораторної роботи, аргументовано викласти вмотивовані пропозиції, обґрунтовані зауваження, конструктивну критику[2], рекомендації тощо.

[1] *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.

[2] *критика* є розглядом і оцінкою чогось з метою виявлення й усунення вад, хиб; під *конструктивною* слід розуміти критику, після якої стає зрозумілим, як саме виправити помилку й не допускати її в майбутньому.

24. Підготувати й зберегти у \Lab9\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
25. Представити до захисту звіт з виконаної лабораторної роботи і проект у Git-репозиторії https://github.com/odorenskyi/*student_name*.

Контрольні запитання і завдання

1. Яким оператором C/C++ можливо повноцінно замінити тернарний оператор? Відповідь обґрунтуйте й доведіть експериментально.
2. Що в програмуванні розуміють під пріоритетом виконання операцій та асоціативністю?
3. Яку область видимості мають змінні, оголошені в тілі циклу або умови (вибору)? Відповідь обґрунтуйте та доведіть експериментально.
4. Якою є асоціативність операцій арифметичних, логічних, логічних порозрядних, інкреметна, декремента, тернарної та порівняння в мові програмування C/C++?
5. Перелічіть випадки, за яких доцільно використовувати тернарний оператор C/C++, й наведіть приклад його запису.
6. Яке значення міститиме змінна `cnt` після виконання наступної інструкції: `cnt--;` ?
7. Чим константна змінна, оголошена за допомогою кваліфікатора типів `const`, відрізняється від змінної? Сформулюйте правило, коли змінну варто оголошувати саме константною.
8. Яких типів можуть бути операнди логічних операторів C/C++?
9. Яке значення міститиме змінна `cnt` при: `bool cnt = !!0;` ?
10. Сформулюйте правило запису виразу ініціалізації у циклах з параметром (`for`) C++.