**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

До лабораторної роботи №1

**На тему: “Ознайомлення із основами створення програм з використанням мови С++. Опції компіляції. Типи збірок. Виконавчі файли, статичні та динамічні бібліотеки”**

**З дисципліни:** *“Об’єктно-орієнтовне програмування”*

**Лектор:**

доц. каф. ПЗ

Тетяна КОРОТЄЄВА

**Виконав:**

ст. гр. ПЗ-16

Тарас КАБАЧОК

**Прийняв:**

асист. каф. ПЗ

Владислав ДАЛЯВСЬКИЙ

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2025 р.

Львів – 2025

**Тема роботи:** Ознайомлення із основами створення програм з використанням мови С++. Опції компіляції. Типи збірок. Виконавчі файли, статичні та динамічні бібліотеки.

**Мета роботи:** Засвоїти принципи створення програм написаних мовою С++, навчитись перетворювати текст програми на виконавчий файл за допомогою командного рядка. Навчитись задавати опції компіляції, створювати різні типи збірок - виконавчі файли, статичні та динамічні бібліотеки.

**Теоретичні відомості**

**Структура програми:**

Програма чи бібліотека складається із однієї чи багатьох одиниць трансляції (translation unit). Зазвичай, одиницею трансляції є файл із розширенням .c для коду C та із розширенням .cpp чи .cxx для коду C++. Кожна одиниця трансляції складається із функцій та змінних. Вони можуть бути як внутрішні (для використання лише в межах однієї одиниці трансляції), так і зовнішні (для зв’язків між одиницями та доступу ззовні). Для зручності використання функцій та змінних між різними одиницями трансляції існують файлизаголовки (header files) із розширенням .h чи .hpp. Їхній вміст підставляється препроцесором у місце директиви #include.

**Структура C файлу:**

Для початку варто розуміти відмінність між оголошенням (declaration) та визначенням

(definition). Визначення пов’язує ім’я із реалізацією цього імені – даними або кодом:

• визначення змінної вказує компілятору виділити певне місце під цю змінну та, можливо, заповнити його певним значенням;

• визначення функції вказує компілятору згенерувати код цієї функції.

Оголошення повідомляє компілятору, що визначення із таким ім’ям та типом існує деінде у програмі, ймовірно, в іншому C файлі. Для змінних, визначення поділяють на два види:

• глобальні змінні, які існують протягом усього життя програми і які доступні з багатьох різних

функцій;

• локальні змінні, які існують лише протягом виконання певної функції та доступні лише з цієї функції.

**Компіляція:**

Завдання компілятора (compiler) – перетворити текст C файлу в об’єктний файл. На UNIX

платформах такі файли мають розширення .o, на Windows - .obj. Вміст об’єктного файлу – це

здебільшого два типи сутностей: • код, що відповідає визначенням функцій у C файлі; дані, що

відповідають визначенням глобальних змінних у C файлі (для ініціалізованих глобальних змінних також містить їхні значення).

**Компонування (лінкування):**

Позаяк оголошення функції чи змінної є для компілятора лише обіцянкою, що десь у програмі існує визначення для цієї функції чи змінної, задача компонувальника полягає в тому, щоб виконати цю обіцянку, тобто знайти відповідні визначення та розв’язати залежності.

**Динамічні бібліотеки:**

Файли динамічних бібліотек мають розширення .so (Linux), .dll (Windows) чи .dylib (Mac).

Під час компонування програми із такими бібліотеками, символи позначаються такими, які

потребують компонування на етапі виконання.

**Хід роботи**

**Частина 1. Компіляція. Компонування.**

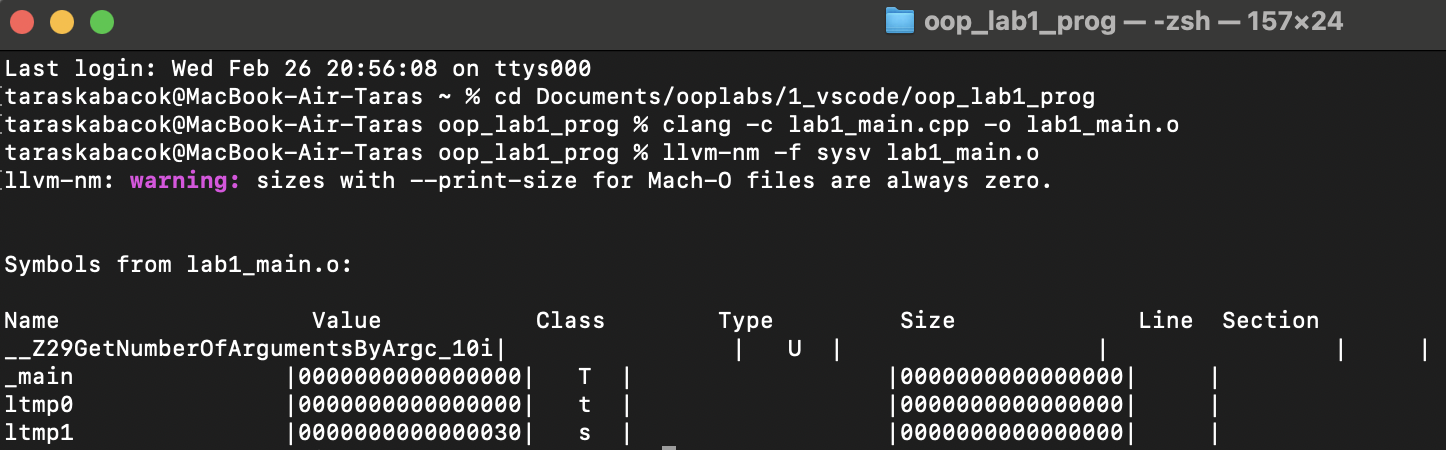
****

Рис. 1. На рисунку зображено результат компілювання файлу **“lab1\_main.cpp”** та подальшого переглядувмісту обʼєктного файлу **“lab1\_main.o”**

* \_\_Z29GetNumberOfArgumentsByArgc\_10i – клас U невизначене посилання, порожнє значення яке генерує компілятор для оголошених, але ще не визначених функцій.
* \_main – клас T до якого можна звертатися з інших файлів.
* ltmp0 та ltmp1 – тимчасові символи (local temporary labels) компілятору LLVM створені автоматично

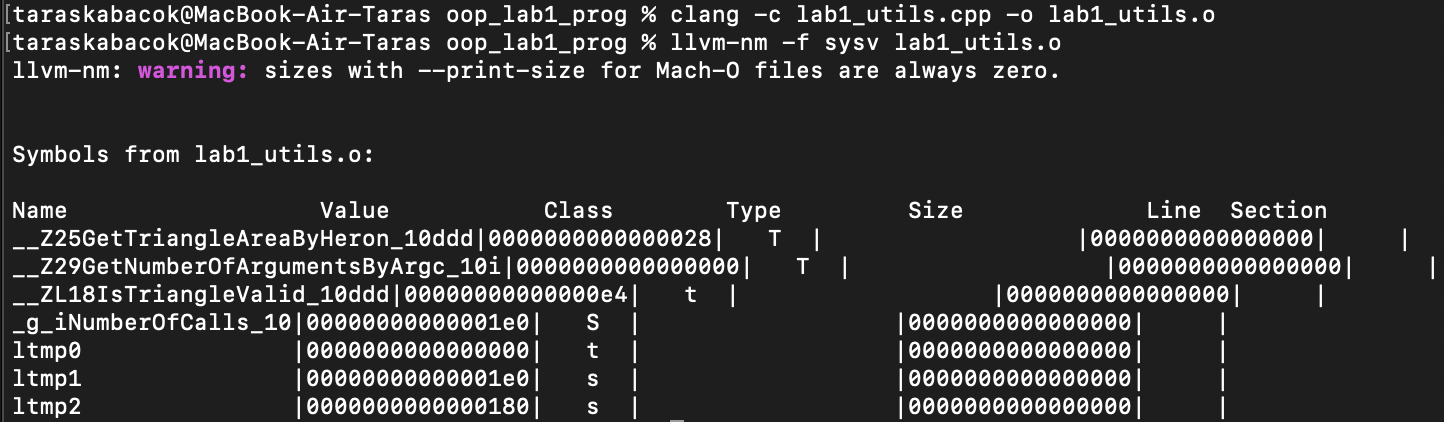


Рис. 2. На рисунку зображено результат компілювання файлу **“lab1\_utils.cpp”** та подальшого переглядувмісту обʼєктного файлу **“lab1\_utils.o”**

* \_\_Z25GetTriangleAreaByHeron\_10ddd – клас T до якого можна звертатися з інших файлів.
* \_\_Z29GetNumberOfArgumentsByArgc\_10i – клас T до якого можна звертатися з інших файлів.
* \_\_ZL18IsTriangleValid\_10ddd – клас t локальний для цього файлу.
* \_g\_iNumberOfCalls\_10 – клас S неініціалізована глобальна змінна, можна звертатися із інших файлів.
* ltmp0, ltmp1 та ltmp2 – тимчасові символи (local temporary labels) компілятору LLVM створені автоматично

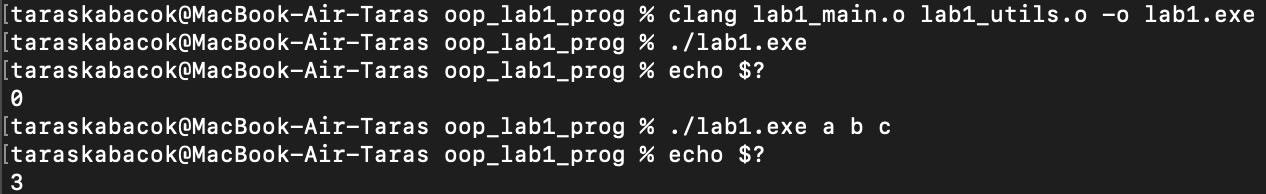


Рис.3. На рисунку зображено результат компонування файлів у виконавчий і виконання lab1.exe з різними параметрами.

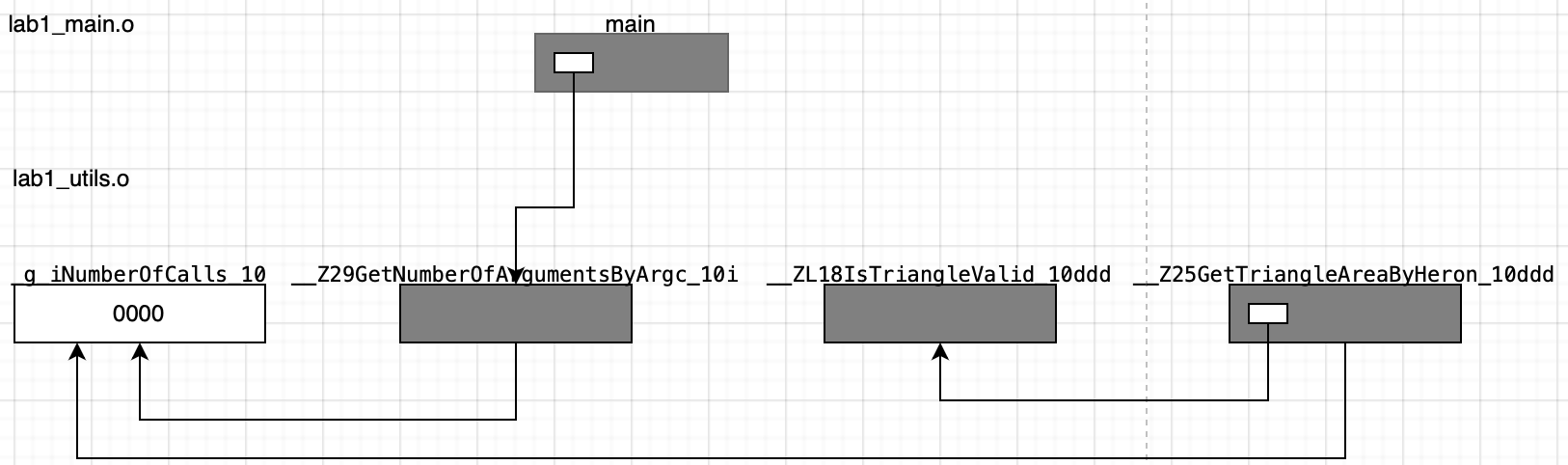


Рис. 4. Діаграма структури програми.

**Частина 2. Опції компіляції, типи збірок.**

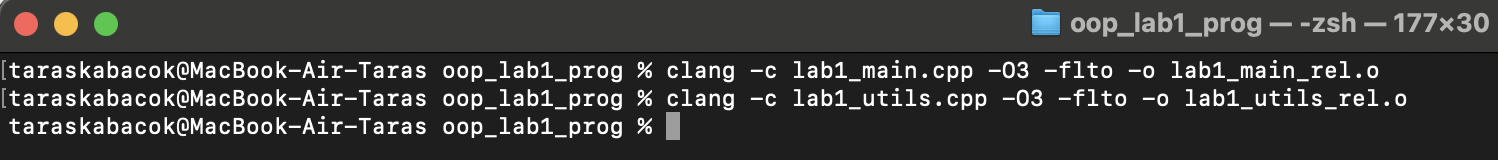
****

Рис. 5. Компіляція двох файлів main та utils, створення двох обʼєктних файлів для збірки реліз.

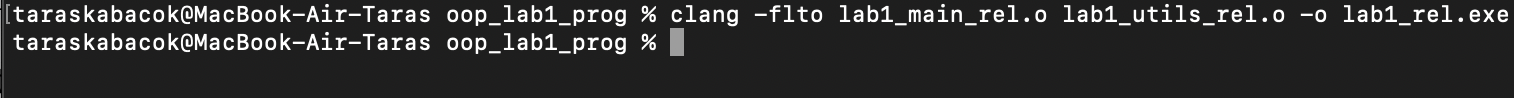


Рис. 6. Компонування обʼєктів в виконавчий файл, збірка реліз.

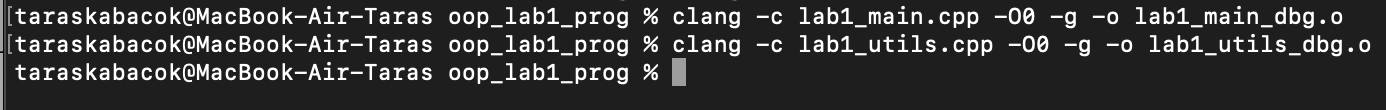


Рис. 7. Компіляція двох файлів main та utils, створення двох обʼєктних файлів для збірки відлагодження.

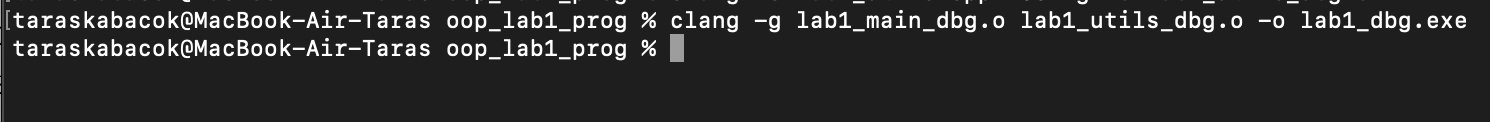


Рис. 8. Компонування обʼєктів в виконавчий файл, збірка відлагодження.

Таблиця 1. Порівняння розмірів файлів збірок

|  |  |  |
| --- | --- | --- |
|  | Розмір для збірки ‘Release’ (байт) | Розмір для збірки ‘Debug’ (байт) |
| lab1\_main.o | 3136 | 2016 |
| lab1\_utils.o | 3136 | 11488 |
| lab1.exe | 16896 | 17904 |

**Частина 3. Статичні та динамічні бібліотеки.**

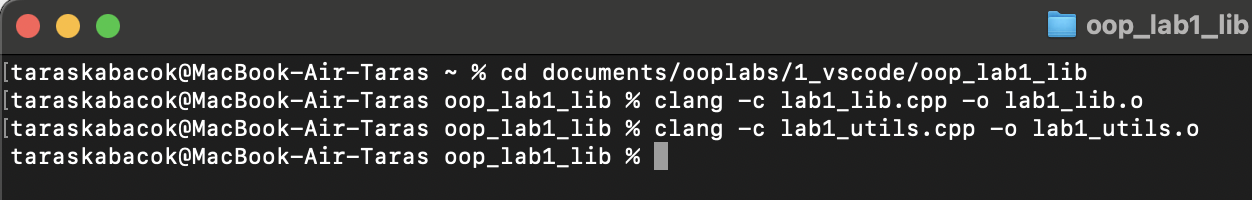
****

Рис. 9. Компіляція двох файлів lib та utils, створення двох обʼєктних файлів.

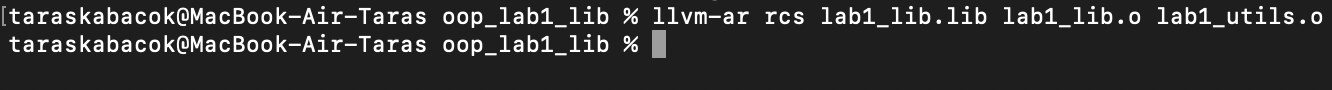


Рис. 10. Обʼєднання попередньо скомпільованих обʼєктів в статичну бібліотеку.

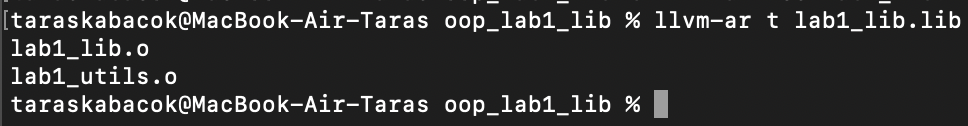


Рис. 11. Перевірка наявності двох файлів в статичній бібліотеці.



Рис. 12. Перекомпілювання в динамічну бібліотеку.

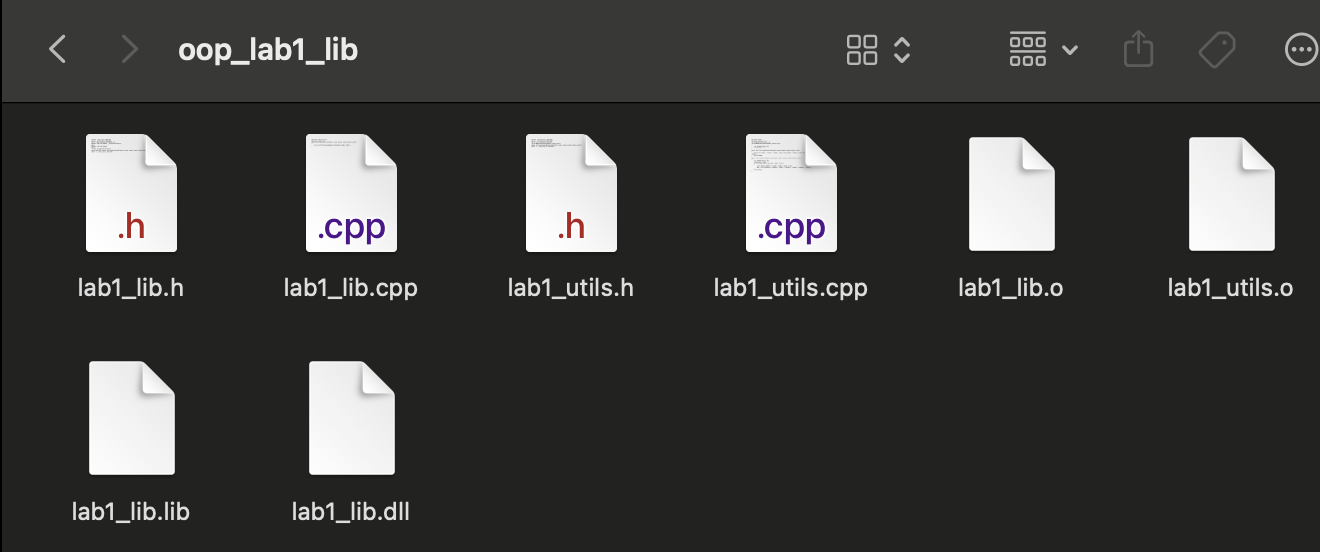


Рис. 13. Вміст папки oop\_lab1\_lib

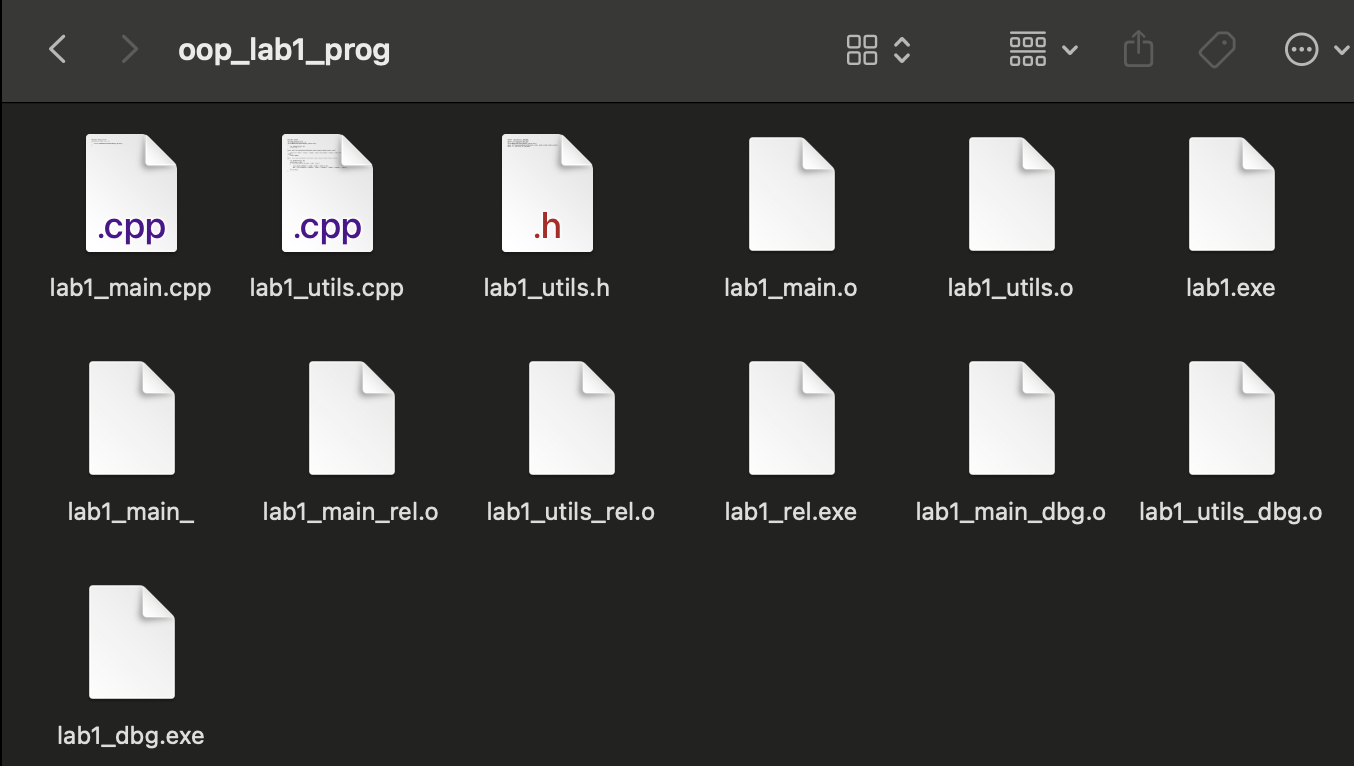


Рис. 14. Вміст папки oop\_lab1\_prog

**Висновки**

В ході виконання даної роботи я навчився застосовувати різні опції компіляції програм на C++, працювати з об'єктними та виконуваними файлами, створювати версії програм для релізу і відлагодження. Окрім цього, я освоїв створення та використання динамічних і статичних бібліотек.