

# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

## Кафедра комп'ютерної інженерії

Дисципліна: Програмування

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній рівень: Перший (бакалаврський)

Семестр: I

---

## ЛАБОРАТОРНА РОБОТА № 1

**Тема:** Розгортання інженерного середовища розробки, конвеєр компіляції GCC, базові типи даних мови C та система контролю версій Git

Тривалість: 4 академічні години

---

### 1. МЕТА РОБОТИ

- Ознайомитися з робочим місцем системного програміста: редактором Visual Studio Code, компілятором GCC та командним рядком (терміналом).
  - Опанувати базовий робочий процес розподіленої системи контролю версій Git та сервісу GitHub.
  - Дослідити на практиці чотири етапи конвеєра компіляції мови C (препроцесування, компіляція, асемблювання, компонування).
  - Дослідити розміри базових типів даних у пам'яті комп'ютера за допомогою оператора `sizeof` та зафіксувати ефект переповнення розрядної сітки (Integer Overflow).
  - Набути практичних навичок форматowanego введення ( `scanf` ) та виведення ( `printf` ) при розв'язанні інженерно-розрахункових завдань.
- 

### 2. УНІВЕРСАЛЬНІ КРИТЕРІЇ ОЦІНЮВАННЯ (10 БАЛІВ)

Ці критерії є єдиними для всіх лабораторних робіт з дисципліни «Програмування»:

| №     | Складова оцінки                      | Макс. бал | Опис вимог                                                                                                                                                                                                                                                                                                                 |
|-------|--------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | Функціональність та коректність коду | 4         | Програма повністю виконує поставлене завдання, успішно компілюється з прапорцем <code>-Wall</code> без помилок та попереджень. Реалізовано базову валідацію вхідних даних (відсутність падінь при некоректному введенні, захист від ділення на нуль).                                                                      |
| 2     | Якість коду та інженерні стандарти   | 2         | Зрозумілі назви змінних та функцій (англійською мовою, <code>Snake_case</code> або <code>camelCase</code> ). Акуратне форматування коду (єдиний стиль відступів у 4 пробіли). Наявність пояснювальних коментарів до складних блоків. Відсутність «магічних чисел» (використання <code>const</code> ).                      |
| 3     | Культура ведення репозиторію Git     | 2         | Своєчасний <code>push</code> до GitHub. Атомарні коміти, створені по ходу виконання роботи (не менше 3–4 комітів на роботу). Зрозумілі повідомлення комітів за стандартом Conventional Commits ( <code>feat: </code> , <code>fix: </code> , <code>docs: </code> ). Відсутність скопійованих бінарних файлів у репозиторії. |
| 4     | Очний захист роботи                  | 2         | Студент вільно пояснює логіку кожного рядка власного коду, відповідає на теоретичні запитання щодо архітектури та пам'яті, здатний внести незначні зміни до коду на вимогу викладача в режимі реального часу.                                                                                                              |
| РАЗОМ |                                      | 10        |                                                                                                                                                                                                                                                                                                                            |

*Штрафні санкції:* Несвоєчасна здача роботи без поважної причини призводить до зниження сумарної оцінки на 20% за кожні два тижні затримки.

### 3. НЕОБХІДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

1. Текстовий редактор / IDE: **Visual Studio Code** (із встановленим розширенням `C/C++` від Microsoft).
2. Набір компіляторів: **GCC** (GNU Compiler Collection через MinGW-w64 для Windows, або нативний GCC для Linux/macOS).

3. Система контролю версій: **Git** (версії 2.30 або новішої).
  4. Обліковий запис на платформі **GitHub**.
- 

## 4. ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

### Етап 0. Конфігурація Git та структури проєкту (перед початком кодингу)

1. Відкрийте термінал і налаштуйте персональні дані автора (якщо це не було зроблено раніше):

```
git config --global user.name "Ім'я Прізвище"  
git config --global user.email "your_email@domain.com"
```

2. Створіть кореневий каталог для дисципліни та перейдіть у нього:

```
mkdir -p ~/projects/programming-sem1  
cd ~/projects/programming-sem1
```

3. Ініціалізуйте Git-репозиторій:

```
git init
```

4. Створіть файл `.gitignore`, щоб тимчасові та скомпільовані файли не потрапляли до сховища. Додайте до нього такі рядки:

```
# Виконувані файли  
*.exe  
*.out  
*.app  
  
# Проміжні об'єктні файли  
*.o  
*.obj  
*.i  
*.s  
  
# Папки середовищ розробки  
.vscode/  
.idea/
```

5. Створіть файл опису проєкту `README.md` з інформацією про назву курсу, вашу академічну групу та автора.
6. Зробіть перший фіксуєчий коміт:

```
git add .gitignore README.md
git commit -m "docs: initial repository setup with gitignore and readme"
```

7. Створіть теку для першої лабораторної роботи:

```
mkdir lab01
cd lab01
```

---

## Етап 1. Практичне дослідження конвеєра компіляції GCC

1. У теці lab01 створіть файл pipeline.c :

```
#include <stdio.h>
#define ARCH_ID 123

int main(void) {
    // Лабораторна робота %1: Конвеєр компіляції
    printf("Computer Engineering Specialty Code: %d\n", ARCH_ID);
    return 0;
}
```

2. Виконайте вручну 4 кроки перетворення програми через термінал:

- **Фаза 1 (Препроцесинг):**

```
gcc -E pipeline.c -o pipeline.i
```

*Завдання:* Відкрийте файл pipeline.i . Переконайтеся, що коментар видалено, макрос ARCH\_ID замінено числом 123 , а на початку файлу з'явився вміст файлу stdio.h .

- **Фаза 2 (Компіляція в асемблерний код):**

```
gcc -S pipeline.i -o pipeline.s
```

*Завдання:* Відкрийте файл pipeline.s . Знайдіть назву функції main та рядок виклику call printf (або call puts ).

- **Фаза 3 (Асемблювання в машинний код):**

```
gcc -c pipeline.s -o pipeline.o
```

*Завдання:* Переконайтеся, що згенеровано двійковий файл pipeline.o , який людина не може прочитати як текст.

- **Фаза 4 (Компонування в готовий бінарник):**

```
gcc pipeline.o -o pipeline_app
```

- **Запуск програми:**

```
./pipeline_app      # для Linux / macOS  
pipeline_app.exe    # для Windows
```

3. Зафіксуйте створений сирцевий файл у Git:

```
git add pipeline.c  
git commit -m "feat(lab1): complete compilation pipeline research"
```

---

## Етап 2. Аудит пам'яті: Розміри типів та шістнадцятковий вивід

1. У теці lab01 створіть файл memory\_audit.c .

2. Реалізуйте програму, яка:

- Визначає розмір у байтах для типів: char , short , int , long , long long , float , double , void\* за допомогою оператора sizeof . Для виведення використовуйте специфікатор %zu .
- Оголошує змінну типу unsigned char byte\_test = 255; .
- Виводить значення byte\_test у десятковому форматі ( %u ) та в шістнадцятковому форматі з префіксом ( 0x%02X ).
- Виконує додавання одиниці: byte\_test = byte\_test + 1; та повторно виводить отримане значення, демонструючи явище переповнення (Integer Overflow).

3. Скомпілюйте програму з прапорцем -Wall та перевірте її виконання:

```
gcc -Wall memory_audit.c -o memory_audit  
./memory_audit
```

4. Додайте файл до індексу Git і зафіксуйте зміни:

```
git add memory_audit.c  
git commit -m "feat(lab1): implement memory audit and overflow demonstration"
```

---

## Етап 3. Інженерний розрахунок (Індивідуальний варіант)

1. У теці lab01 створіть файл task3.c .

2. Відповідно до свого номера у списку групи оберіть варіант завдання з **Розділу 5**.

3. Реалізуйте програму на мові C, яка:

- Запитує у користувача вхідні дані за допомогою зрозумілих текстових підказок

( printf ).

- Зчитує значення змінних за допомогою функції `scanf()` (пам'ятайте про оператор `&` ).
- Перевіряє коректність введених даних: перевіряє повернене значення `scanf` (чи ввів користувач саме число) та запобігає діленню на нуль або кореню з від'ємного числа.
- Виконує обчислення за формулою свого варіанта.
- Виводить результат із зазначенням одиниць вимірювання та з точністю **до 2 або 3 знаків після коми**.

4. Скомпілюйте, протестуйте на різних вхідних даних та зафіксуйте у Git:

```
git add task3.c
git commit -m "feat(lab1): solve individual calculation task"
```

## Етап 4. Відправка результатів на GitHub

1. Перейдіть у свій браузер на сайт [GitHub.com](https://github.com) і створіть новий публічний репозиторій з назвою `programming-sem1`.
2. У терміналі зв'яжіть локальний репозиторій із сервером (виконується з кореневої теки проєкту):

```
git remote add origin https://github.com/ВАШ_НИКНЕЙМ/programming-sem1.git
git branch -M main
git push -u origin main
```

3. Переконайтеся, що на сайті GitHub відображаються файли `pipeline.c`, `memory_audit.c`, `task3.c`, файл `README.md`, а скомпільованих файлів ( `.exe`, `.o` ) у списку немає.

## 5. ВАРІАНТИ ЗАВДАНЬ ДЛЯ ЕТАПУ 3

*Примітка:* Номер варіанта обирається як: **Варіант = (Номер у списку журналу - 1) % 20 + 1.**

| Вар. | Фізична /<br>Інженерна задача | Вхідні<br>параметри          | Розрахункова формула                       | Вихідні             |
|------|-------------------------------|------------------------------|--------------------------------------------|---------------------|
| 1    | Закон Ома для<br>ділянки кола | Напруга $U$<br>(В), опір $R$ | $I = \frac{U}{R}, \quad P = \frac{U^2}{R}$ | Струм<br>потужність |

| Вар. | Фізична /<br>Інженерна задача                | Вхідні<br>параметри                                          | Розрахункова формула                                                                          | Вихідні                        |
|------|----------------------------------------------|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------|--------------------------------|
|      |                                              | (Ом)                                                         |                                                                                               | (Вт)                           |
| 2    | Кінетична та<br>потенціальна<br>енергія      | Маса $m$ (кг),<br>швидкість $v$<br>(м/с), висота<br>$h$ (м)  | $E_k = \frac{m \cdot v^2}{2}, \quad E_p = m \cdot g \cdot h \quad (g = 9.81)$                 | Енергія<br>(Дж)                |
| 3    | Паралельне<br>з'єднання<br>резисторів        | Опори $R_1, R_2$<br>(Ом)                                     | $R_{eq} = \frac{R_1 \cdot R_2}{R_1 + R_2}$                                                    | Еквівалентний<br>опір $R$      |
| 4    | Енергія<br>зарядженого<br>конденсатора       | Ємність $C$<br>(Ф), напруга<br>$U$ (В)                       | $W = \frac{C \cdot U^2}{2}, \quad q = C \cdot U$                                              | Енергія<br>(Дж), заряд<br>(Кл) |
| 5    | Реактивний опір<br>катушки<br>індуктивності  | Частота<br>сигналу $f$<br>(Гц),<br>індуктивність<br>$L$ (Гн) | $X_L = 2 \cdot \pi \cdot f \cdot L \quad (\pi \approx 3.14159)$                               | Опір $X_L$                     |
| 6    | Реактивний опір<br>конденсатора              | Частота<br>сигналу $f$<br>(Гц), ємність<br>$C$ (Ф)           | $X_C = \frac{1}{2 \cdot \pi \cdot f \cdot C}$                                                 | Опір $X_C$                     |
| 7    | Розрахунок опору<br>мідного<br>провідника    | Довжина $l$<br>(м), переріз $S$<br>(мм <sup>2</sup> )        | $R = \rho \cdot \frac{l}{S} \quad (\rho_{Cu} = 0.0175 \text{ Ом} \cdot \text{мм}^2/\text{м})$ | Опір $R$                       |
| 8    | Закон<br>всесвітнього<br>тяжіння             | Маси $m_1, m_2$<br>(кг), відстань<br>$r$ (м)                 | $F = G \cdot \frac{m_1 \cdot m_2}{r^2} \quad (G = 6.674 \times 10^{-11})$                     | Сила тяжіння<br>$F$ (Н)        |
| 9    | Період коливань<br>математичного<br>маятника | Довжина<br>нитки $l$ (м)                                     | $T = 2 \cdot \pi \cdot \sqrt{\frac{l}{g}} \quad (g = 9.81)$                                   | Період $T$                     |
| 10   | Період коливань<br>пружинного<br>маятника    | Маса<br>тягарця $m$<br>(кг),<br>жорсткість $k$<br>(Н/м)      | $T = 2 \cdot \pi \cdot \sqrt{\frac{m}{k}}$                                                    | Період $T$                     |
| 11   | Перетворення<br>температурних<br>шкал        | Температура<br>$T_C$ (°C)                                    | $T_F = T_C \cdot \frac{9}{5} + 32, \quad T_K = T_C + 273.15$                                  | Фаренгейт<br>Кельвін           |

| Вар. | Фізична / Інженерна задача                        | Вхідні параметри                                            | Розрахункова формула                                                     | Вихідні дані                                |
|------|---------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------|
| 12   | Час передачі файлу каналом зв'язку                | Розмір файлу $V$ (МБ), пропускна здатність $S$ (Мбіт/с)     | $t = \frac{V \cdot 8}{S}$                                                | Час передачі (с)                            |
| 13   | Розрахунок ділителя напруги                       | Вхідна напруга $U_{in}$ (В), опори $R_1, R_2$ (Ом)          | $U_{out} = U_{in} \cdot \frac{R_2}{R_1 + R_2}$                           | Вихідна напруга (В)                         |
| 14   | Теплова енергія Джоуля-Ленца                      | Струм $I$ (А), опір $R$ (Ом), час $t$ (с)                   | $Q = I^2 \cdot R \cdot t$                                                | Теплова енергія (Дж)                        |
| 15   | Сила Лоренца                                      | Заряд $q$ (Кл), швидкість $v$ (м/с), індукція $B$ (Тл)      | $F = q \cdot v \cdot B$ (при куті $90^\circ$ )                           | Сила Лоренца (Н)                            |
| 16   | Доцентрове прискорення та сила                    | Маса $m$ (кг), радіус $R$ (м), лінійна швидкість $v$ (м/с)  | $a = \frac{v^2}{R}, \quad F = m \cdot a$                                 | Прискорення (м/с <sup>2</sup> )<br>Сила (Н) |
| 17   | Гідростатичний тиск стовпа рідини                 | Густина рідини $\rho$ (кг/м <sup>3</sup> ), глибина $h$ (м) | $P = \rho \cdot g \cdot h \quad (g = 9.81)$                              | Тиск $P$ (Па)                               |
| 18   | Довжина хвилі електромагнітного випромінювання    | Частота передавача $f$ (МГц)                                | $\lambda = \frac{c}{f \cdot 10^6} \quad (c = 3 \times 10^8 \text{ м/с})$ | Довжина хвилі $\lambda$ (м)                 |
| 19   | Резонансна частота коливального контуру (Томсона) | Індуктивність $L$ (Гн), ємність $C$ (Ф)                     | $f_0 = \frac{1}{2 \cdot \pi \cdot \sqrt{L \cdot C}}$                     | Частота $f_0$ (Гц)                          |
| 20   | Закон Кулона                                      | Заряди $q_1, q_2$ (Кл), відстань $r$ (м)                    | $F = k \cdot \frac{\ q_1 \cdot q_2\ }{r^2} \quad (k = 8.99 \times 10^9)$ | Сила взаємодії (Н)                          |

## 6. ЗМІСТ ЗВІТУ ТА ФОРМА ЗДАЧІ

Робота захищається очно в комп'ютерному класі.

Для перевірки викладачу надається **посилання на публічний GitHub-репозиторій**, який містить:

1. Файл `README.md` з інформацією про студента та виконану роботу.
  2. Файл `.gitignore`.
  3. Теку `lab01` із сирцевими файлами:
    - `pipeline.c`
    - `memory_audit.c`
    - `task3.c`
  4. Історію комітів (Commit History), що підтверджує самостійне та поетапне написання коду.
- 

## 7. КОНТРОЛЬНІ ЗАПИТАННЯ ДЛЯ ОЧНОГО ЗАХИСТУ (2 БАЛИ)

1. Які чотири етапи проходить сирцевий код на шляху до створення `.exe` файлу?
2. Чим відрізняється робота препроцесора від роботи компілятора?
3. Що зберігається у файлах із розширеннями `.i`, `.s` та `.o`?
4. Для чого використовується оператор `sizeof`? Чому його обчислення не навантажує процесор під час виконання програми?
5. Чому результат операції `255 + 1` для типу `unsigned char` дорівнює `0`? Поясніть фізику явища переповнення.
6. Навіщо у виклику `scanf("%lf", &voltage)` перед назвою змінної обов'язково передавати символ `&`?
7. Що станеться, якщо у виразі `int a = 5, b = 2; double c = a / b;` не використати явне приведення типів (`double`)?
8. Що робить команда `git add`, і чим вона відрізняється від `git commit`?
9. Чому файли з розширенням `.exe` та `.o` обов'язково слід додавати до `.gitignore`?