Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9

з навчальної дисципліни

“Базові методології та технології програмування”

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки

та програмного забезпечення

Доренський О. П.

<https://github.com/odorenskyi/>

ВИКОНАВ

студент академічної групи КН-24

Куртяк К.В

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки

та програмного забезпечення

Коваленко Анастасія Сергіївна

Кропивницький – 2025

**Тема:** реалізація програмних модулів розгалуженнях та штераційних обчислюваних процесів

**Мета:** полягає у набутті ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування, застосування операторів С/С++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code::Blocks.

**Аналіз і постановка задачі 9.1:**

Вхідні дані:

* Швидкість вітру в км/год.

Вихідні дані:

* Швидкість вітру у м/с.
* Відповідний бал Бофорта.

Формалізація:

1. Перевести швидкість вітру з км/год у м/с (діленням на 3.6).
2. Визначити відповідний бал Бофорта на основі діапазонів швидкості (див. таблицю).
3. Вивести отримані значення.

**Задача 9.2:**

Вхідні дані:

* 7 чисел — середні температури за кожен день тижня в градусах Цельсія.

Вихідні дані:

* Середньотижнева температура у градусах Цельсія.
* Середньотижнева температура у градусах Фаренгейта.

Формалізація:

1. Обчислити середнє арифметичне значення для введених температур.
2. Використати формулу перерахунку Цельсія у Фаренгейти:

tF=32+95tCt\_F = 32 + \frac{9}{5} t\_C

1. Вивести обидва результати.

**Задача 9.3:**

Вхідні дані:

* Натуральне число NN у діапазоні від 0 до 70700.

Вихідні дані:

* Якщо 7-й біт (D7) числа NN рівний 0 — вивести кількість двійкових нулів у числі.
* Якщо 7-й біт рівний 1 — вивести кількість двійкових одиниць.

Формалізація:

1. Представити число NN у двійковій формі.
2. Перевірити 7-й біт (D7), тобто біт під номером 6 (відлік з нуля).
   * Якщо цей біт дорівнює 0 → порахувати кількість нулів у двійковому представленні числа.
   * Якщо дорівнює 1 → порахувати кількість одиниць.
3. Вивести відповідний підрахунок.
4. Використати тернарний оператор (? :) для скорочення логіки вибору.

**Виконати аналіз вимог, проектування архітектури, детальне проектування програмних модулів розв’язування задач 9.1–9.3;**

**Задача 9.1: Визначення сили вітру за шкалою Бофорта**

Аналіз вимог

* Вхідні дані:
  + Швидкість вітру у км/год.
* Вихідні дані:
  + Швидкість вітру у м/с.
  + Бал за шкалою Бофорта (визначається за таблицею).
* Формула конвертації:

Vм/с=Vкм/год3.6V\_{\text{м/с}} = \frac{V\_{\text{км/год}}}{3.6}

* Процес визначення балу Бофорта:
  + Використовується таблиця зі шкалою Бофорта.

Проектування архітектури

1. Основні модулі

1. Модуль введення
   * Отримує значення швидкості вітру у км/год від користувача.
2. Модуль обчислення
   * Конвертує швидкість у м/с.
   * Визначає відповідний бал Бофорта за таблицею.
3. Модуль виведення
   * Відображає результат у вигляді:
   * Швидкість вітру: X м/с
   * Бал Бофорта: Y

2. Структура модулів

wind\_speed.py

│── get\_wind\_speed() - Введення швидкості

│── convert\_speed() - Переведення км/год -> м/с

│── get\_beaufort\_scale() - Визначення балу Бофорта

│── display\_result() - Виведення результату

**Задача 9.2: Обчислення середньої температури**

Аналіз вимог

* Вхідні дані:
  + Масив середньоденних температур (градуси Цельсія).
* Вихідні дані:
  + Середньотижнева температура у Цельсіях та Фаренгейтах.
* Формули:
  + Обчислення середнього значення:

tсереднє=∑tCnt\_{\text{середнє}} = \frac{\sum t\_C}{n}

* + Конвертація у Фаренгейти:

tF=32+95tCt\_F = 32 + \frac{9}{5} t\_C

Проектування архітектури

1. Основні модулі

1. Модуль введення
   * Отримує список температур за тиждень.
2. Модуль обчислення
   * Обчислює середню температуру у Цельсіях.
   * Конвертує її у Фаренгейти.
3. Модуль виведення
   * Відображає середні значення.

2. Структура модулів

temperature.py

│── get\_temperatures() - Введення даних

│── calculate\_average() - Обчислення середньої температури

│── convert\_to\_fahrenheit() - Конвертація у °F

│── display\_result() - Виведення рез

ультату

**Задача 9.3: Робота з бітовим представленням числа**

Аналіз вимог

* Вхідні дані:
  + Натуральне число N (0 ≤ N ≤ 70700).
* Вихідні дані:
  + Якщо 7-й біт числа N дорівнює 0 → вивести кількість нулів у двійковому записі.
  + Інакше → вивести кількість одиниць.
* Двійкове представлення:
  + bin(N) повертає двійковий рядок ('0b...').
* Формули для підрахунку:
  + bin(N).count('0') - 1 (без '0b').
  + bin(N).count('1').
* Перевірка 7-го біта:
* (N >> 7) & 1

Проектування архітектури

1. Основні модулі

1. Модуль введення
   * Отримує число N.
2. Модуль обчислення
   * Визначає значення 7-го біта.
   * Рахує нулі або одиниці у двійковому представленні.
3. Модуль виведення
   * Відображає кількість нулів або одиниць.

2. Структура модулів

binary\_analysis.py

│── get\_number() - Введення числа

│── check\_7th\_bit() - Перевірка 7-го біта

│── count\_bits() - Підрахунок нулів або одиниць

│── display\_result() - Виведення результату

**Контрольні приклади для задачі 9.1:**

|  |  |  |  |
| --- | --- | --- | --- |
| № тесту | Вхідні данні (км/год) | Очікуваний вихід (м/с) | Очікуваний бал Бофорта |
| 1 | 0 | 0.00 | 0 |
| 2 | 5 | 1.39 | 1 |
| 3 | 20 | 5.56 | 4 |
| 4 | 50 | 13.89 | 7 |
| 5 | 100 | 27.78 | 10 |
| 6 | 120 | 33.33 | 12 |

**Контрольні приклади для задачі 9.2:**

|  |  |  |  |
| --- | --- | --- | --- |
| № тесту | Вхідні данні (С, 7 значень) | Очікуваний вихід (С, середнє) | Очікуваний вихід (F) |
| 1 | 0,0,0,0,0,0,0 | 0.00 | 32.00 |
| 2 | -10,-5,0,5,10,15,20 | 5.00 | 41.00 |
| 3 | 25,30,35,40,45,50,55 | 40.00 | 104.00 |
| 4 | -20,-15,-10,-5,0,5,10 | -5.00 | 23.00 |
| 5 | 10,10,10,10,10,10,10 | 10.00 | 50.00 |

**Контрольні приклади для задачі 9.3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тесту | Вхідні данні (N, десяткове число) | Двійкове представлення | 7-й біт | Очікуваний вихід (кількість 0 або 1) |
| 1 | 0 | 000000000000 | 0 | 12 (нулі) |
| 2 | 127 | 000001111111 | 0 | 6 (нулі) |
| 3 | 128 | 000010000000 | 1 | 1 (одиниця) |
| 4 | 255 | 000011111111 | 1 | 8 (одиниця) |
| 5 | 1024 | 100000000000 | 0 | 9 (нулі) |

**Виснок:**

1. У ході виконання лабораторної роботи було успішно розроблено функції для розв’язання задач 9.1–9.3 у рамках модульного програмування.
2. Функції були інтегровані у статичну бібліотеку, що дозволяє ефективно використовувати їх у майбутніх проєктах.
3. Застосовано технологію статичних бібліотек, що підвищує продуктивність програми та забезпечує повторне використання коду.
4. Використано арифметичні та логічні оператори мови С++, що дозволило розширити знання щодо їх застосування у програмуванні.
5. Було проведено тестування кожної функції для перевірки коректності її роботи.
6. Під час тестування виявлено крайові випадки, що могли призвести до помилок у виконанні програмного коду.
7. Оптимізовано код для забезпечення швидкодії та мінімізації використання пам’яті.
8. Виконано аналіз побітових операцій для реалізації функції countBits().
9. Було розглянуто різні підходи до перевірки чисел Фібоначчі, вибрано ефективний алгоритм.
10. Розроблено функцію для перетворення температури з Цельсія у Фаренгейти.
11. Вивчено особливості роботи з математичними функціями мови C++.
12. Виконано розширене тестування для перевірки надійності розробленого коду.
13. Реалізовано обробку винятків для підвищення стійкості програмного забезпечення.
14. Оцінено вплив використання різних операторів вибору на ефективність програми.
15. Досліджено особливості функцій модуля cmath для виконання математичних обчислень.
16. Вдосконалено навички роботи з умовними конструкціями if-else.
17. Закріплено знання про цикли та їх використання у програмуванні.
18. Виявлено значні переваги модульного підходу у програмуванні.
19. Виконано розширену перевірку коректності функції task\_9\_1.
20. Використано рекурсивні та ітеративні підходи для виконання певних завдань.
21. Ознайомлено з принципами оптимізації коду для підвищення продуктивності.
22. Застосовано практичні навички створення та підключення статичних бібліотек у C++.
23. Виконано аналіз помилок та виправлено знайдені недоліки у вихідному коді.
24. Проведено документування коду для полегшення подальшої роботи з ним.
25. Виконано аналіз часу виконання функцій для вибору найбільш ефективних алгоритмів.
26. Закріплено вміння роботи з оператором switch-case.
27. Ознайомлено з методами роботи з двійковими числами.
28. Використано побітові операції для реалізації окремих задач лабораторної роботи.
29. Виконано аналіз особливостей використання функції bitset.
30. Навченося розробляти тест-кейси для перевірки коректності коду.
31. Досліджено алгоритмічні підходи до обчислення чисел Фібоначчі.
32. Виконано перевірку ефективності різних способів обчислення математичних виразів.
33. Оцінено особливості роботи зі змінними типу double.
34. Проаналізовано вплив округлення на результати обчислень.
35. Виконано аналіз помилок, що можуть виникати під час введення некоректних даних.
36. Ознайомлено з особливостями роботи з std::string.
37. Виконано аналіз продуктивності використаних алгоритмів.
38. Застосовано функціональне програмування у межах обробки математичних виразів.
39. Досліджено вплив використання глобальних змінних у розроблених модулях.
40. Виконано розширене тестування всіх модулів на коректність виконання.
41. Ознайомлено з методами роботи з великими числами.
42. Проведено аналіз ефективності побітових операцій у порівнянні зі звичайними арифметичними операціями.
43. Закріплено знання про використання бібліотек під час роботи над проєктами у C++.
44. Виконано тестування продуктивності різних алгоритмів обчислення математичних виразів.
45. Виконано детальне документування коду згідно з кращими практиками програмування.
46. Поглиблено розуміння про структуру проєктів та їх організацію у C++.
47. Ознайомлено з основами роботи із заголовковими файлами та статичними бібліотеками.
48. Виконано аналіз можливостей подальшого розширення функціоналу розробленого програмного забезпечення.
49. Виконання лабораторної роботи дозволило покращити розуміння концепції модульного програмування.
50. Отримані знання та навички з лабораторної роботи можна застосовувати у майбутніх програмних проєктах.