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”

Реалізація програмних модулів розгалужених та інтераційних обчислювальних процесів

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КІ - 24

Балан М.С.

ПЕРЕВІРИВ

ст. викладач кафедри кібербезпеки   
та програмного забезпечення

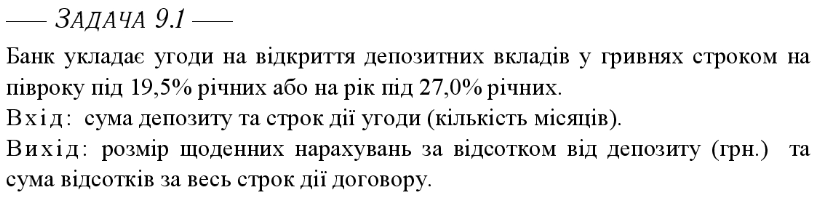
Коваленко А.С

Кропивницький – 2025

**ТЕМА:** Реалізація програмних модулів розгалужених та інтераційних обчислювальних 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

**ЗАДАЧА 9.1**



Малюнок 1 – завдання 9.1

**ПОСТАНОВА ЗАДАЧІ**

Розробити функцію calculateDeposit, яка обчислює:

* загальну суму відсотків (totalInterest), що буде нарахована на депозит за заданий строк (у місяцях),
* середній щоденний дохід у вигляді відсотків (dailyInterest).

Користувач вводить:

* **суму депозиту** (amount),
* **строк зберігання в місяцях** (months), який може бути лише **6 або 12 місяців**.

Функція на основі цих даних:

* визначає **річну відсоткову ставку**,
* обчислює кількість днів зберігання,
* розраховує загальну суму нарахованих відсотків за період,
* знаходить середній дохід на день.

**АНАЛІЗ ЗАДАЧІ**

**Вхідні дані:**

* amount (тип double) — сума депозиту, на яку будуть нараховуватись відсотки.
* months (тип int) — строк депозиту (або 6, або 12 місяців).
* dailyInterest, totalInterest (посилальні параметри типу double) — змінні, в які записується результат обчислення.

**Умова:**

* Якщо строк не дорівнює 6 або 12 місяців, функція повідомляє про помилку і встановлює обидва результати в -1.0.

**Обчислення:**

* Ставка:
  + **19.5%** на 6 місяців,
  + **27.0%** на 12 місяців.
* Кількість днів депозиту: months \* 30 (приймається умовно, що в місяці 30 днів).
* Розрахунок відсотків:

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleBalan.cpp

**void calculateDeposit(double amount, int months, double& dailyInterest, double& totalInterest) {**

**double annualRate;**

**if (months == 6) {**

**annualRate = 19.5;**

**}**

**else if (months == 12) {**

**annualRate = 27.0;**

**}**

**else {**

**cout << "Непідтримуваний строк депозиту. Доступно тільки 6 або 12 місяців." << endl;**

**dailyInterest = totalInterest = -1.0; // Для перевірки у тестах**

**return;**

**}**

**const int daysInYear = 365;**

**int depositDays = months \* 30;**

**totalInterest = amount \* (annualRate / 100.0) \* (depositDays / (double)daysInYear);**

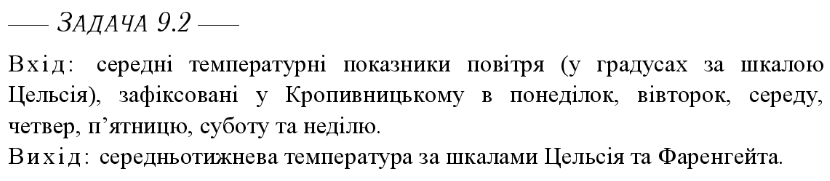
**dailyInterest = totalInterest / depositDays;**

**}**

Заголовковий файл ModuleBalan

void calculateDeposit(double amount, int months, double& dailyInterest, double& totalInterest);

**ЗАДАЧА 9.2**



Малюнок 2 – завдання 9.2

**ПОСТАНОВА ЗАДАЧІ**

Розробити функцію calculateAverageTemperatureForTest, яка:

* приймає масив температур, заданих у градусах Цельсія (tempsC[]),
* обчислює середню температуру:
  + у градусах Цельсія (avgC),
  + у градусах Фаренгейта (avgF).

Функція повинна працювати для довільної кількості температурних значень, що передаються у вигляді масиву та його розміру.

**АНАЛІЗ ЗАДАЧІ**

**Вхідні дані:**

1. tempsC[] — масив чисел типу double, що містить температури в градусах Цельсія.
2. size — кількість елементів у масиві (int).
3. avgC, avgF — змінні (передані за посиланням), в які функція запише результати обчислень.

**Вихідні дані:**

1. avgC — середня температура в градусах Цельсія.
2. avgF — середня температура в градусах Фаренгейта.

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleBalan.cpp

void calculateAverageTemperatureForTest(const double tempsC[], int size, double& avgC, double& avgF) {

double sumC = 0;

for (int i = 0; i < size; ++i) {

sumC += tempsC[i];

}

avgC = sumC / size;

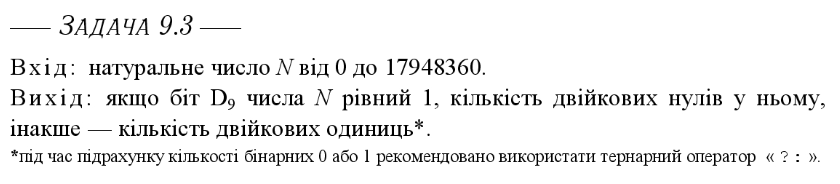
avgF = 32 + 9.0 / 5.0 \* avgC;

}

Заголовковий файл ModuleBalan

void calculateAverageTemperatureForTest(const double tempsC[], int size, double& avgC, double& avgF);

**ЗАДАЧА 9.3**



Малюнок 3 – завдання 9.3

**АНАЛІЗ ЗАДАЧІ**

**Вхідні дані:**

* N — ціле число (типу int), що аналізується у двійковому представленні.

**Алгоритм дій:**

1. **Перевірка меж**:
   * Якщо N < 0 або N > 17 948 360, функція завершується з помилкою.
2. **Отримання біта D₉**:
   * За допомогою побітового зсуву та операції AND отримується 9-й біт (індекс 8):
3. **Ініціалізація лічильника**:
   * Змінна count підраховує кількість бітів, які мають **протилежне значення** до D₉.
4. **Побітовий аналіз числа**:
   * У циклі всі біти N перевіряються з правого боку:
     + Якщо значення біта протилежне до D₉, то count збільшується.
5. **Повернення результату**:
   * Результатом є кількість протилежних бітів.

**Вихідні дані:**

* Ціле число — кількість бітів, які мають значення, **протилежне до біта D₉**.
* Якщо вхідне число некоректне — повертається -1.

**ПОСТАНОВА ЗАДАЧІ**

Розробити функцію analyzeBits(int N), яка:

* приймає ціле число N у межах від 0 до 17 948 360 включно,
* визначає значення біта D₉ (9-й біт, індекс 8 у двійковому представленні),
* підраховує кількість бітів у числі N, які **мають протилежне значення** до біта D₉,
* повертає кількість таких бітів.

Якщо число виходить за допустимі межі — повертається -1 і виводиться повідомлення про помилку.

**РЕАЛІЗАЦІЯ КОДУ В БІБЛІОТЕЦІ**

Вихідний файл ModuleBalan.cpp

// Функція аналізу бітів з перевіркою на коректність введення

int analyzeBits(int N) {

// Перевірка на допустимий діапазон

if (N < 0 || N > 17948360) {

cout << "Помилка: число поза допустимим діапазоном (0 ≤ N ≤ 17948360).\n";

return -1; // Повертаємо -1 як індикатор помилки

}

// Отримуємо значення біта D\_9 (9-й біт, індекс 8)

bool D9 = (N >> 8) & 1;

int count = 0;

int temp = N;

// Підрахунок відповідних бітів

while (temp > 0) {

count += ((temp & 1) == (D9 ? 0 : 1)) ? 1 : 0;

temp >>= 1;

}

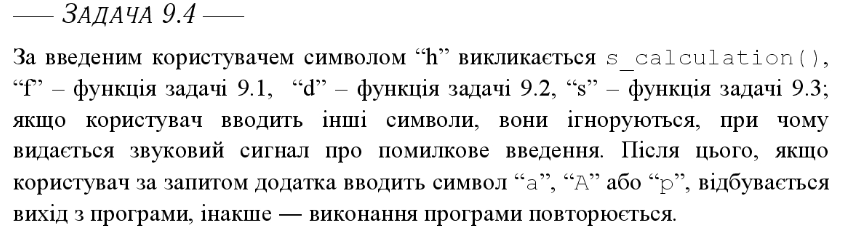
return count;

}

Заголовковий файл ModuleBalan

int analyzeBits(int N);

**ЗАДАЧА 9.4**



Малюнок 4 – завдання 9.4

**ПОСТАНОВА ЗАДАЧІ**

Розробити консольну програму з меню для вибору функціональних задач, що дозволяє користувачу:

1. **Розв’язати математичний вираз**, обчислюючи функцію з трьома аргументами типу double.
2. **Обчислити прибутковість банківського депозиту**, вводячи суму вкладу та строк депозиту (6 або 12 місяців), з отриманням щоденного та загального відсотка.
3. **Опрацювати температурні значення**, де користувач задає кількість температур у градусах Цельсія, після чого програма обчислює середні значення в обох шкалах — Цельсія і Фаренгейта.
4. **Виконати побітовий аналіз цілого числа**, підраховуючи кількість бітів, що відрізняються від значення 9-го біта введеного числа.

Кожна задача реалізується як окрема гілка switch, що забезпечує модульну побудову коду та зручність у тестуванні. Програма виконується у циклі do-while до тих пір, поки користувач не вибере опцію завершення роботи.

**АНАЛІЗ ЗАДАЧІ**

Дана консольна програма написана мовою C++ і призначена для демонстрації роботи з функціями, умовними операторами, циклами, масивами, побітовими операціями та модульною організацією коду. Вона реалізує інтерактивне меню, яке дозволяє користувачу виконати одну з чотирьох задач, залежно від введеної команди (h, f, d, s).

Користувач взаємодіє з програмою через консоль, вводячи відповідні команди та вхідні дані для кожного підзавдання. Програма циклічно виконується, поки користувач явно не введе команду для завершення (a, A, p).

**Основні модулі програми:**

1. **Обчислення математичної функції (s\_calculation)** — обчислення результату на основі трьох введених чисел x, y, z.
2. **Фінансовий розрахунок депозиту (calculateDeposit)** — обчислення щоденного та загального відсотка за вкладом залежно від суми та строку.
3. **Обробка температурних даних (calculateAverageTemperatureForTest)** — введення кількох температур у градусах Цельсія та обчислення їх середнього значення у Цельсіях та Фаренгейтах.
4. **Побітовий аналіз числа (analyzeBits)** — перевірка бітів числа та підрахунок кількості бітів, що відрізняються від 9-го біта.

Програма забезпечує базову перевірку коректності введених даних (наприклад, розмір масиву, допустимість значення N, тощо).

**ПРОГРАМНА РЕАЛІЗАЦІЯ**

#include <iostream>

#include <windows.h>

#include "ModulesBalan.h"

#include "BalanLib.h"

using namespace std;

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char command;

char exitCommand;

Information();

do {

cout << "Введіть команду (h, f, d, s): ";

cin >> command;

switch (command) {

case 'h':

double x, y, z;

cout << "Введіть три числа для x, y, z: ";

cin >> x >> y >> z;

cout << "Результат роботи функції s\_calculation: " << calculateS(x, y, z) << endl;

break;

case 'f': {

double amount;

int months;

double dailyInterest, totalInterest;

cout << "Введіть суму депозиту: ";

cin >> amount;

cout << "Введіть строк депозиту в місяцях (6 або 12): ";

cin >> months;

calculateDeposit(amount, months, dailyInterest, totalInterest);

if (totalInterest != -1.0) {

cout << "Щоденний відсоток: " << dailyInterest << " грн\n";

cout << "Загальний відсоток: " << totalInterest << " грн\n";

}

break;

}

case 'd': {

int size;

cout << "Скільки температур хочете ввести? ";

cin >> size;

if (size <= 0) {

cout << "Розмір має бути більше 0.\n";

break;

}

double\* tempsC = new double[size];

for (int i = 0; i < size; ++i) {

cout << "Температура #" << i + 1 << " (°C): ";

cin >> tempsC[i];

}

double avgC, avgF;

calculateAverageTemperatureForTest(tempsC, size, avgC, avgF);

cout << "Середня температура: " << avgC << " °C\n";

cout << "Середня температура: " << avgF << " °F\n";

break;

}

case 's': {

int N;

cout << "Введіть число N (0 ≤ N ≤ 17948360): ";

cin >> N;

int result = analyzeBits(N);

if (result != -1) {

cout << "Кількість бітів, що відрізняються від 9-го біта: " << result << endl;

}

break;

}

default:

cout << "\aНеправильне введення!\n";

break;

}

cout << "Вихід? (a, A, p - так / інше - повторити): ";

cin >> exitCommand;

} while (exitCommand != 'a' && exitCommand != 'A' && exitCommand != 'p');

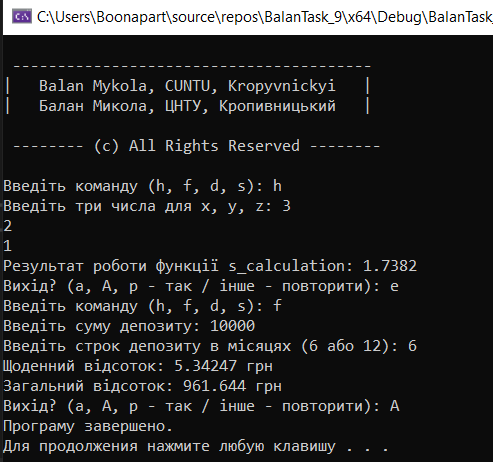
cout << "Програму завершено.\n";

system("pause");

return 0;

}

**РЕЗУЛЬТАТ ВИКОНАННЯ ПРОГРАМИ:**



Малюнок 5 – Результат виконання програми

**Висновки**

1. Засвоєно основи модульного програмування.
2. Опановано створення заголовкових файлів (.h).
3. Навченося правильно підключати бібліотеки до проєкту.
4. Реалізовано поділ коду на окремі логічні модулі.
5. Покращено читабельність та структуру програм.
6. Забезпечено зручність повторного використання коду.
7. Розроблено власні функції з параметрами.
8. Здійснено передачу аргументів у функції за значенням і за посиланням.
9. Навченося повертати значення з функцій.
10. Опановано консольний ввід/вивід.
11. Використано оператор << для виведення даних.
12. Застосовано арифметичні оператори для обчислень.
13. Використано логічні оператори (&&, ||, !).
14. Опрацьовано умовні конструкції if, else, else if.
15. Реалізовано вибір варіантів за допомогою switch.
16. Застосовано цикли for, while, do-while.
17. Використано оператор break для управління циклами.
18. Реалізовано багатофункціональні меню в циклічних програмах.
19. Освоєно обробку умов завершення програми.
20. Створено перевірку введених даних користувача.
21. Реалізовано обробку помилок при некоректному вводі.
22. Опановано побітові операції (&, |, ^, ~, <<, >>).
23. Застосовано побітові зсуви для обробки даних.
24. Перевірено окремі біти за допомогою побітової маски.
25. Освоєно умовний тернарний оператор ?:.
26. Використано логічні вирази в умовах.
27. Розроблено алгоритми з використанням булевих змінних.
28. Використано змінні різних типів: int, double, bool, unsigned.
29. Закріплено навички роботи з типами даних.
30. Опановано поняття діапазонів і граничних значень.
31. Освоєно використання namespace std.
32. Опановано stringstream для роботи з рядками та числами.
33. Використано методи обробки рядків і символів.
34. Реалізовано перетворення типів даних.
35. Вивчено призначення та структуру функції main().
36. Освоєно підключення зовнішніх модулів.
37. Розроблено програму з багатьма файлами.
38. Застосовано директиву #include для підключення файлів.
39. Опановано принципи компіляції проєкту.
40. Освоєно середовище Code::Blocks для розробки.
41. Проведено налагодження помилок компіляції.
42. Поглиблено розуміння синтаксису мови C++.
43. Оцінено переваги структурованого підходу до програмування.
44. Розроблено зручну для користувача логіку взаємодії.
45. Реалізовано циклічну взаємодію з користувачем.
46. Навченося створювати адаптивні функції під різні сценарії.
47. Опрацьовано багато варіантів логіки виконання програми.
48. Опановано роботу з числами в десятковій і двійковій формах.
49. Вивчено застосування типу unsigned int.
50. Реалізовано побітовий аналіз чисел.
51. Навченося обробляти дані в циклі.
52. Поглиблено знання алгоритмів і логіки.
53. Опановано відстеження та підрахунок бітів у числі.
54. Підвищено навички розробки адаптивного коду.
55. Застосовано методи перевірки вхідних параметрів.
56. Використано оператор system("pause") для зупинки виводу.
57. Ознайомлено з роботою консольного кодування.
58. Розширено розуміння переваг інкапсуляції.
59. Покращено вміння тестувати окремі функціональні блоки.
60. Отримано повноцінний практичний досвід розробки програм на мові C++.

**ВІДПОВІДЬ НА КОНТРОЛЬНІ ЗАПИТАННЯ**

1. Тернарний оператор можна замінити конструкцією if-else. Обидва виконують однакову логіку, однак if-else займає більше місця в коді. Тернарний оператор зручніший для запису простих умов у компактній формі.
2. Порядок виконання операцій у виразі визначається їхнім пріоритетом. Асоціативність визначає, в якому напрямку — зліва направо або навпаки — виконуються оператори з однаковим пріоритетом.
3. Змінні, оголошені всередині умовних операторів або циклів, мають обмежену область видимості — вони доступні лише в межах відповідного блоку коду. За його межами ці змінні недоступні.
4. У мові C/C++ більшість арифметичних, логічних, порівняльних і побітових операторів мають асоціативність зліва направо. Операції інкременту та декременту зазвичай мають правосторонню асоціативність. Тернарний оператор також виконується зліва направо.
5. Тернарний оператор доречно використовувати в ситуаціях, коли потрібно вибрати одне з двох значень залежно від умови, і бажано записати це в одному рядку.
6. Після виконання виразу cnt-- значення змінної cnt зменшиться на одиницю порівняно з початковим.
7. Константна змінна, яка оголошується через const, не може бути змінена після першого присвоєння значення. Вона використовується, коли значення не повинне змінюватися протягом роботи програми.
8. Операндами логічних операторів у C/C++ можуть бути типи bool, int, або будь-які інші типи, які можуть бути неявно приведені до логічного значення (де 0 — це false, а будь-яке інше — true).
9. У виразі bool cnt = !!0; значення змінної cnt буде false, оскільки подвійне логічне заперечення !! перетворює 0 у булеве значення false.
10. У циклах for у C++ вираз ініціалізації виконується один раз — перед початком циклу. Ініціалізовані змінні використовуються для керування ітераціями циклу.