Крос-платформне програмування

ЛАБОРАТОРНА РОБОТА № 1

Створення та виконання простої програми. Знайомство з середовищем розробки. Програмування алгоритмів лінійної структури. Консольний ввід-вивід числових даних та виконання простих арифметичних обчислень.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи
.

Львів

2024

Мета роботи: набуття навичок практичної роботи в інтегрованому середовищі; ознайомлення з структурою програми на мовах Java та Kotlin, елементами синтаксису цих мов програмування та деякими типами даних на прикладі простої програми.

Завдання до лабораторної роботи.

1. Для обидвох задач (Завдання 1 та Завдання 2), відповідно до варіанта, розробити алгоритм розв'язування, та зобразити його графічно у вигляді блок-схеми.
2. Запрограмувати розроблений алгоритм мовою ***Java***.
3. Запрограмувати розроблений алгоритм мовою ***Kotlin***.
4. Оформити письмовий звіт про виконання роботи.

Звіт повинен містити:

- титульний аркуш:
- для кожного завдання:
  - ✓ умову задачі відповідного варіанту;
  - ✓ блок-схему алгоритму;
  - ✓ програму на Java;
  - ✓ результати виконання програми для деякого набору вхідних даних;
  - ✓ програму на Kotlin;
  - ✓ результати її виконання для того ж набору вхідних даних.

Короткі теоретичні відомості

Історія ***кросплатформності*** у програмуванні починається зі створення об'єкно-орієнтованої мови та технології програмування ***Hot Java***. Зокрема, її створення, за словами очевидців, розпочинається з невдоволення одного з провідних іженерів компанії ***Sun Microsystems*** Патріка Ноутона потребою в перекомпіляції програмних кодів ***C++*** для різних пристроїв, над програмним керуванням якими в той час працювала компанія. Так виникає та впроваджується ідея використання проміжного коду (байт-коду) та виконання програм під керуванням віртуальної машини (***Java Virtual Machine*** – ***JVM***). Пізніше такі ж ідеї лягли в основу нової технології програмування ***.NET Framework*** від компанії ***Microsoft***.

Кросплатформність – це здатність технологій програмування створювати ПЗ в умовах, коли програмна чи (і) апаратна платформи для розробки відрізняється від цільової платформи, на якій це ПЗ працюватиме. Часто цю властивість програмного забезпечення трактують більш широко і ототожнюють з ***мультиплатформністю*** (багатоплатформністю) — здатністю програм працювати більш ніж на одній програмній та апаратній платформі, та технології, що дозволяють досягти цієї властивості.

Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення. Залежно від засобів реалізації кросплатформність поділяють на:

- ✓ *кросплатформність на рівні мов програмування та інструментів розробки;*
- ✓ *кросплатформність на рівні середовища виконання;*
- ✓ *кросплатформність на рівні операційної системи;*
- ✓ *кросплатформність на рівні апаратної частини.*

В сучасній розробці кросплатформність програмного забезпечення, зазвичай, досягається використанням проміжних платформ, наприклад, віртуальних машин, як в ***Java***. Скомпільована ***Java-програма*** (байт-код) працюватиме на будь-якій платформі, де встановлена відповідна ***JVM***, в тому числі і на мобільних пристроях, з якими часто асоціюють кросплатформне програмування. У свій час ***Java*** була основним засобом для розробки мобільних додатків, – спочатку у вигляді пакету ***Java Mobile Edition***, пізніше, як технологія для програмування під ОС ***Android***.

Мова програмування Java повністю ґрунтується на об'єктно-орієнтованому підході (об'єктно-орієнтованій парадигмі програмування). В об'єктно-орієнтованому програмуванні (ООП) програми утворюються з сукупності об'єктів, що взаємодіють між собою. Об'єкт є іменованою моделлю реальної сутності, що володіє конкретними значеннями своїх властивостей та визначеною поведінкою. Набір властивостей та поведінка об'єкта описуються в класі, на основі якого створюється об'єкт. З іншого боку, клас об'єднує у собі набір даних та методи їх обробки, а об'єкт є представником класу, який

володіє конкретними значеннями описаних у класі даних. Навіть найпростіша програма на Java повинна складатися принаймні з одного класу. Наприклад, програма, що виводить на екран текст «Hello, world!», може мати вигляд:

```
public class HelloWorld {
     public static void main(String[] args) {
          System.out.println("Hello, world!");
     }
}
```

В першому рядку коду оголошується клас HelloWorld, з якого складається програма. Далі в фігурних дужках записана тіло класу, в якому є лише головний метод main, що містить єдину команду для виводу на консоль тексту «Hello, world!».

На сьогодні одним з центральних орієнтирів кросплатформної (точніше, мультиплатформної) розробки стала чеська компанія ***JetBrains***, яка, зокрема, з 2001 постачає на ринок найпопулярнішу на наших теренах ***IDE*** для ***Java*** - ***Intellij Idea***. В 2010 році компанія започаткувала нову мову програмування ***Kotlin***, спочатку, як внутрішній проєкт. Команда розробників мала на меті створити сучасну, лаконічну та прагматичну мову для створення надійного та ефективного програмного забезпечення. Офіційно про створення нової технології програмування компанія оголосила вже у липні 2011. Проєкт випустили під opensource-ліцензією Apache 2.0, надавши зацікавленим розробникам можливість робити свій внесок у розвиток мови.

Ранні випуски ***Kotlin*** випускалися, здебільшого, щоб зібрати відгуки користувачів з метою вдосконалення мови. А вже у лютому 2016 р. вийшов офіційний випуск ***Kotlin 1.0***, який здобув популярність, насамперед, в мобільній розробці, і у травні 2017 р. компанія ***Google*** оголосила про офіційну підтримку ***Kotlin*** як рекомендованої мови для розробки ***Android-додатків***. ***Kotlin*** продовжує розвиватися завдяки регулярним оновленням, підтримуючи при цьому взаємодію з платформами та кодовими базами ***Java***.

Згодом компанією JetBrains розролено фреймворк ***Kotlin Multiplatform***, що, власне, дозволила розробникам створювати мультиплатформний код для різних платформ, в т. ч., Android та iOS.

Kotlin вважається об’єктно-орієнтованою мовою програмування, на 100% сумісною з ***Java***, але підтримує і процедурне програмування за допомогою функцій, подібно до ***C++, JavaScript*** чи ***Python***. У випадку простих обчислювальних задач цього цілком достатньо.

«Дослівний переклад» найпростішої об’єкно-орієнтованої ***Java***-програми на ***Kotlin*** виглядатиме так:

```
object HelloWorld {
 @JvmStatic
 fun main(args: Array<String>) {
      println("Hello, world!");
 }
}
```

А в «процедурному варіанті»:

```
fun main() {
      println("Hello, world!");
}
```

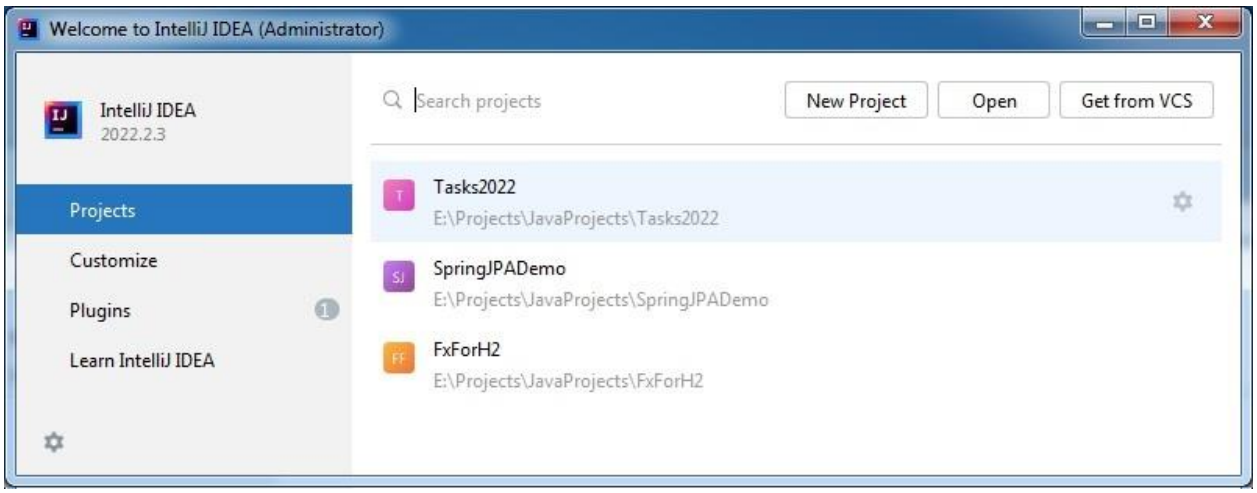
В якості середовища для програмування прикладів та розв’язування задач в рамках цього навчального посібника пропонуємо використовувати середовище ***Intellij Idea***. Завантажити безкоштовну версію продукту можна з офіційного сайту компанії за адресою https://www.jetbrains.com/idea/download.

Щоб створити програму на Java за допомогою Intellij Idea спочатку потрібно:

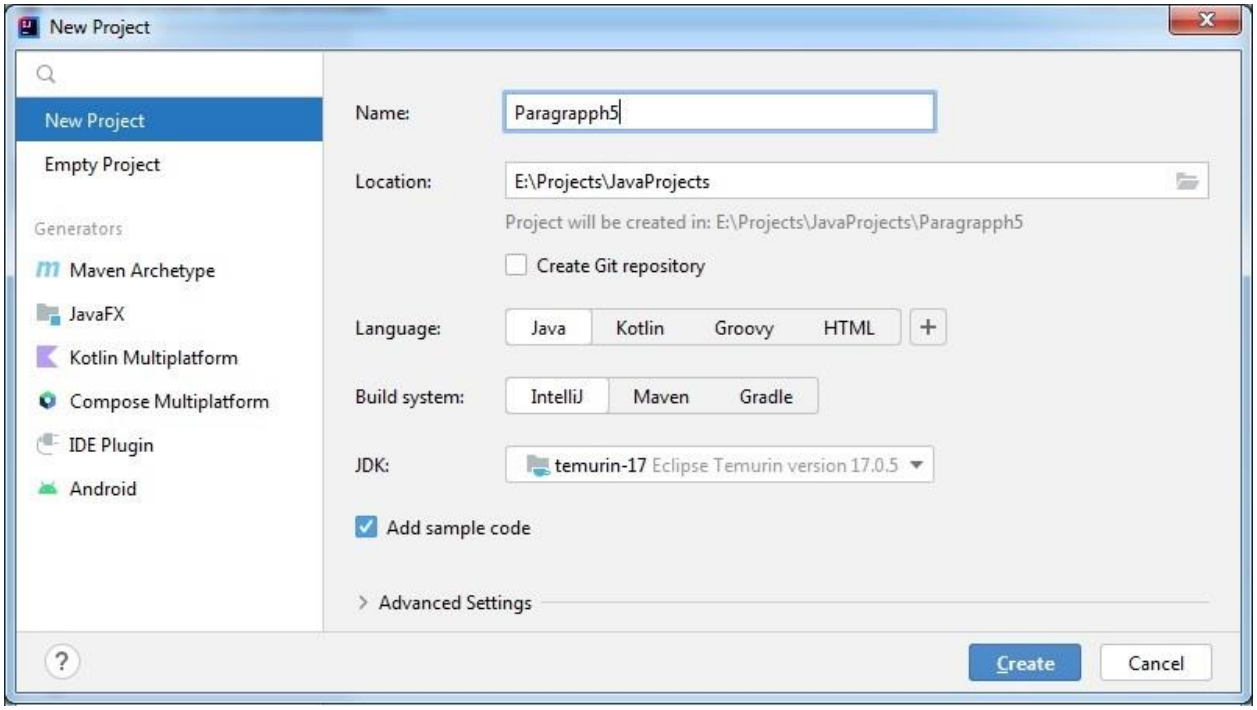
- *завантажити та встановити пакет JDK;*
- *завантажити та встановити Intellij Idea;*
- *запустити Intellij Idea та створити Java-проект;*
- *до створеного проєкту додати новий Java-клас;*
- *в класі оголосити статичний main-метод.*

Далі можна переходити до написання коду програми.

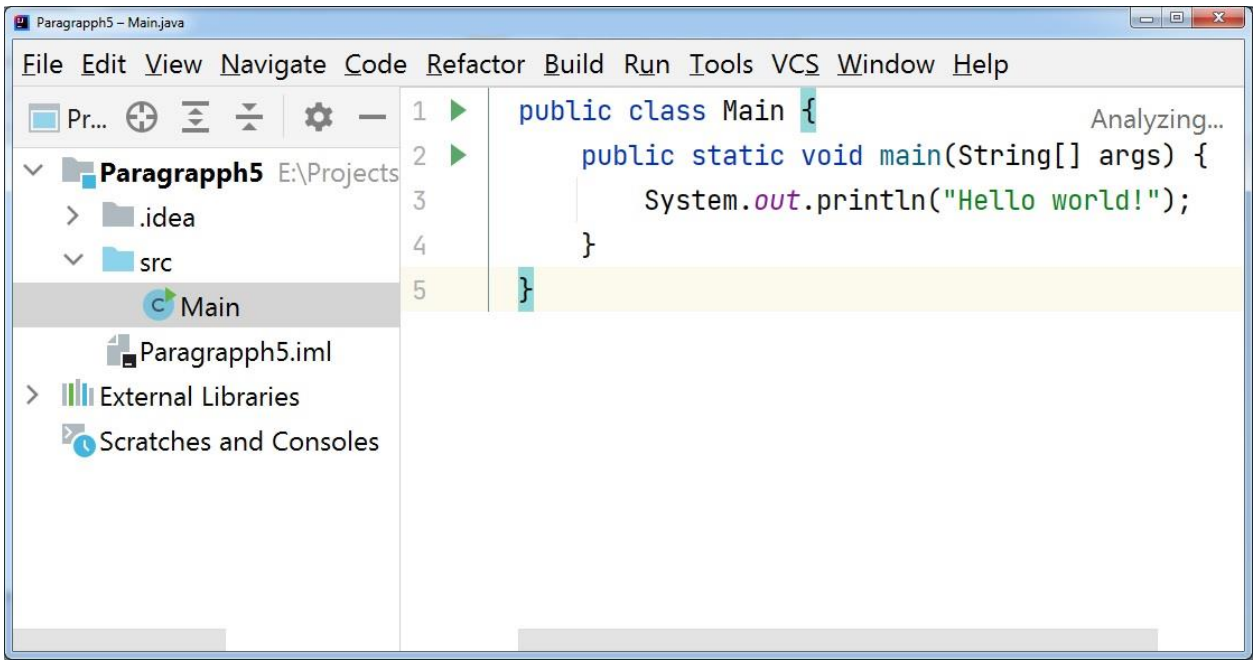
Перший запуск щойно встановленої ***Intellij Idea*** буде починатися зі стартового вікна, подібного до показаного на малюку. Наступні запуски, зазвичай, починаються з проєкту, що був відкритий за попереднього сеансу роботи IDE. Якщо закрити поточний проєкт, виконавши команду **File => Close Project**, – ми знову побачимо стартове вікно ***Intellij Idea***.



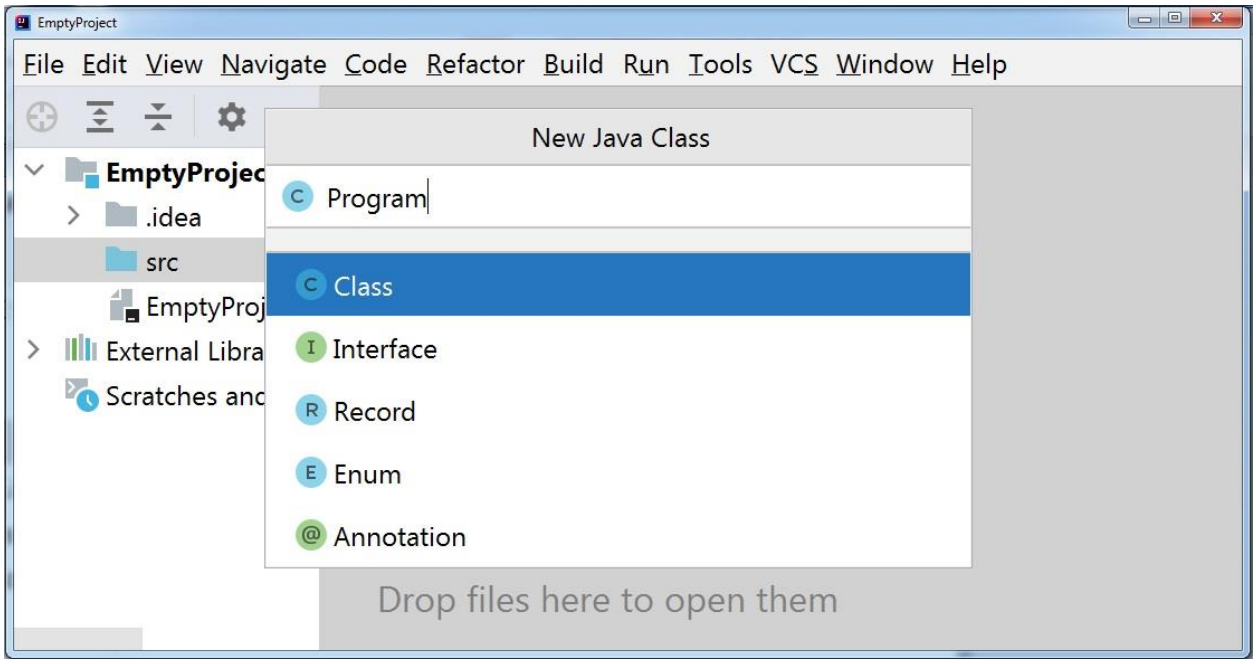
Розпочати створення нового проєкту можна кнопкою **New Project** на стартовому вікні ***Intellij Idea***, або **File => New => Project** з меню основного вікна програми.



На наступному кроці у вікні ***New Project*** слід обрати потрібні параметри проєкту, зокрема, папку для розташування проєкту та його назву. Для створення класичного консольного java-застосунку тип проєту у списку зліва слід залишити на пункті “New Project”. Так само варта залишити вибраними опції “Java” та “Intellij”. У розкривний список з підписом “JDK”, зазвичай, Intellij успішно знаходить і показує поточну версію ***Java Development Kit*** втановлену в системі, хоча інколи доводиться вказувати тут шлях до пакету ***JDK*** вручну. Ще одна опція “Add simple code” дозволяє при створенні проєкту отримати готовий клас ***`Main`*** з готовим методом ***`main`***, в якому слід писати код програми.



Якщо прапорець “Add simple code” не позначити, то ми отримаємо порожній проєкт, а клас для написання коду програми потрібно буде до нього додати. Це можна зробити з контекстного меню, викликаного на вузлі “src” у дереві проєкту зліва.

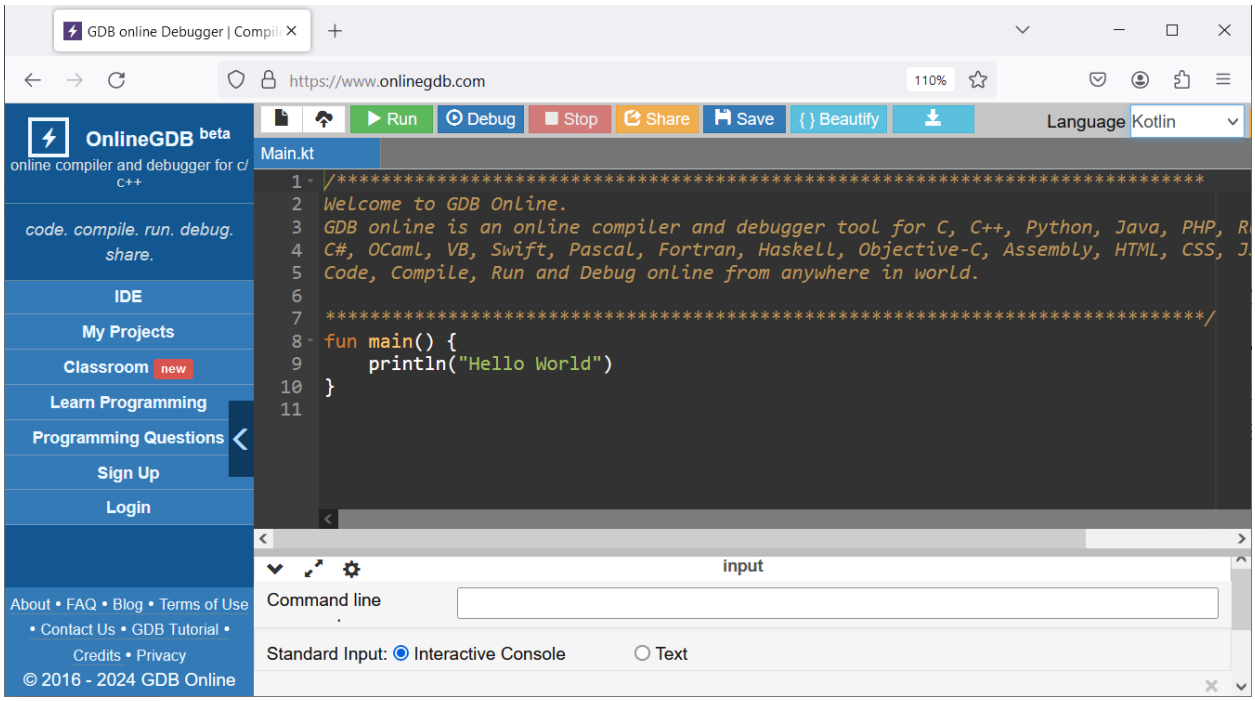


Далі можемо переходити до написання самого коду. Зазначимо лише, що найпростіший спосіб запустити створену програму – натиснути позначку у формі зеленого трикутника зліва від назви класу, чи таку ж позначку зліва від заголовку метода ***`main`***.

Проєкт на ***Kotlin*** створюється в ***IntelliJ Idea*** аналогічно.

Для створення простих програм (як у завданнях до цієї лабораторної роботи) встановлювати в ***IntelliJ Idea*** на ПК не обов’язково, бо для цього можна скористатися одним з онлайнових середовищ в мережі Internet.

Наприклад мультимовний редактор програм GDB online Debugger, розміщений за адресою https://www.onlinegdb.com/. Серед кількох десятків підтримуваних мов програмування, він надає можливіть програмувати і на ***Java***, і на ***Kotlin***.



Завдання 1.

Вважаючи усі вхідні і вихідні дані дійсними числами типу **float**, написати програму для розв’язання задачі згідно варіанту.

Варіанти завдань.

1. Задано два кола із загальним центром і радіусами R_1 і R_2 ($R_1 > R_2$). Знайти площі цих кіл S_1 та S_2, а також площу S_3 кільця, зовнішній радіус якого дорівнює R_1, а внутрішній радіус дорівнює R_2. Як значення π використовувати 3.14.

2. Задано координати двох протилежних вершин прямокутника: (x_1, y_1), (x_2, y_2). Сторони прямокутника паралельні осям координат. Знайти периметр і площу даного прямокутника.

3. Знайти відстань між двома точками із заданими координатами (x_1, y_1) і (x_2, y_2) на площині.

4. Задано координати трьох вершин трикутника: (x_1, y_1), (x_2, y_2), (x_3, y_3). Знайти його периметр, використовуючи формулу для відстані між двома точками на площині.

5. Задано сторони прямокутника a і b. Знайти його площу S і периметр P.

6. Задано довжини ребер A, B, C прямокутного паралелепіпеда. Знайти його об’єм та площу поверхні.

7. Задано число A. Обчислити A^{15}, використовуючи дві допоміжні змінні і п'ять операцій множення. Для цього послідовно знаходити A^2, A^3, A^5, A^{10}, A^{15}. Вивести всі знайдені степені числа A.

8. Задано значення температури T в градусах Фаренгейта. Визначити значення цієї ж температури в градусах Цельсія. Температура за Цельсієм TC і температура TF за Фаренгейтом зв'язані наступним співвідношенням: TC = (TF - 32) • 5/9.

9. Задано значення температури T в градусах Цельсія. Визначити значення цієї ж температури в градусах Фаренгейта. Температура за Цельсієм TC і температура за Фаренгейтом TF пов'язані наступним співвідношенням: TC = (TF - 32) • 5/9.

10. Відомо, що X кг шоколадних цукерок коштує A гривень, а Y кг ірисок - B гривень. Визначити, скільки коштує 1 кг шоколадних цукерок, 1 кг ірисок, а також у скільки разів шоколадні цукерки дорожчі ірисок.

11. Задано два ненульових числа. Знайти суму, різницю, добуток і частку їх квадратів.

12. Задано два ненульових числа. Знайти суму, різницю, добуток і частку їх модулів.

13. Задано катети прямокутного трикутника a і b. Знайти його гіпотенузу c і периметр P.

14. Швидкість першого автомобіля V_1 км / год, другого - V_2 км / год, відстань між ними S км. Визначити відстань між ними через T годин, якщо вони рухаються в одному напрямку.

15. Швидкість першого автомобіля V_1 км / год, другого - V_2 км / год, відстань між ними S км. Визначити відстань між ними через T годин, якщо автомобілі рухаються назустріч один одному.

16. Задано довжину кола L. Визначити його радіус R та площу S круга, який це коло обмежує, враховуючи, що $L = 2\pi R$, $S = \pi R^2$.

17. Задано площу круга S. Визначити його радіус R та довжину кола L, яке його обмежує, враховуючи, що $S = \pi R^2$, $L = 2\pi R$.

18. Знайти потенціальну енергію тіла масою m, піднятого на висоту h над поверхнею Землі ($E = mgh$).

19. Знайти об'єм V та площу бічної поверхні $S_б$ циліндра, якщо відомо радіус R його основи та висоту H.

20. Знайти об'єм V та площу бічної поверхні $S_б$ конуса, якщо відомо радіус R його основи та висоту H.

21. Тіло починає рухатися без початкової швидкості з прискоренням a. Визначити швидкість v, яку розвине тіло за t сек., та шлях S, який воно пройде за цей час.

22. За заданим радіусом кулі обчислити її об’єм та площу поверхні за формулами $V = \frac{4}{3}\pi R^3$, $S = 4\pi R^2$.

23. Задано об’єм кулі. Визначити її радіус та площу поверхні, враховуючи, що $V = \frac{4}{3}\pi R^3$, $S = 4\pi R^2$.

24. Задана площа ділянки в гектарах. Вивести площу ділянки спочатку в метрах квадратних, а потім кілометрах квадратних.

25. Задана маса вантажу в центнерах. Вивести цю масу спочатку в кілограмах, а потім в тонах.

26. Задана маса вантажу в фунтах. Вивести цю масу спочатку в кілограмах, а потім в центнерах, якщо 1 фунт = 0,45359237 кілограма.

27. Водій на автозаправці залив у бак X літрів пального. Визначити суму до сплати, якщо відома ціна C за літр пального та відсоток P фіксованої (не залежить від кількості пального) знижки.

28. Виріб збирають з трьох вузлів собівартістю A, B та C грн. відповідно. На оплату складання одного виробу затрачають X грн. Визначити ціну виробу, якщо від його продажу планується отримання прибутку P%.

29. Задано довжини сторін трикутника a, b та с. Обчислити довжини його медіан $m_a = \frac{\sqrt{2b^2+2c^2-a^2}}{2}$, $m_b = \frac{\sqrt{2a^2+2c^2-b^2}}{2}$, $m_c = \frac{\sqrt{2a^2+2b^2-c^2}}{2}$.

30. Металічний куб з довжиною ребра a переплавили в кулю. Визначити радіус R кулі, враховуючи, що об'єм кулі $V = \frac{4}{3}\pi R^3$, а об'єм куба $V = a^3$.

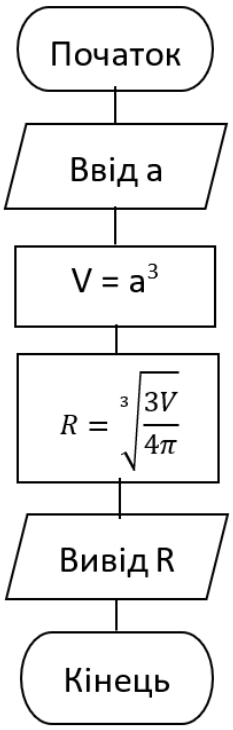
Приклад розв'язування задачі 30

30. Металічний куб з довжиною ребра a переплавили в кулю. Визначити радіус R кулі, враховуючи, що об'єм кулі $V = \frac{4}{3}\pi R^3$, а об'єм куба $V = a^3$.

Вхідними даними тут є довжина ребра куба a, її значення програма повинна отримати від користувача напочатку, далі можна знайти об'єм куба за формулою $V = a^3$. Якщо вважати, що при переплавці металу його об'єм не змінюється, то отримане значення V буде об'ємом кулі. А от для знаходження її радіуса формулу, задану в умові, слід записати у вигляді $R = \sqrt[3]{\frac{3V}{4\pi}}$, виразивши радіус R через об'єм V. Ця формула і є формулою, за якою програма обчислюватиме розв'язок задачі.

Алгоритм розв'язування задачі буде таким:

- *на початку роботи програми користувач вводить число, що задає довжину ребра куба a;*
- *програма знаходить об'єм куба (a, отже, і об'єм кулі) за формулою* $V = a^3$;
- *за отриманим значенням об'єму* V *та заданим в програмі значенням числа* π *програма обчислює величину радіуса кулі за фомулою* $R = \sqrt[3]{\frac{3V}{4\pi}}$;
- *обчислену довжину радіуса кулі програма повідомляє користувачеві і завершує свою роботу.*



Графічне зображення сформульованого вище словесно алгоритму подано у вигляді блок-схеми на малюнку.

В більшості мов програмування базові бібліотеки не мають функції для обчислення кореня кубічного, тому формулу для знаходження R використовуємо у вигляді

$$R = \sqrt[3]{\frac{3V}{4\pi}} = \left(\frac{3V}{4\pi}\right)^{\frac{1}{3}}.$$

Програма на Java.

«Приладнати» об'єктно-орієнтовану ***Java*** до структурного розв'язування простої обчислювальної задачі можна, дотримуючись наступних засад:

- ✓ *код програми записується в класі, клас має ім'я, що записується після службового* ***class*** *та тіло у фігурних дужках, що містить поля (дані) та методи (функції) класу;*
- ✓ *для того, щоб код класу можна було виконати як самостійну програму він повинен містити «головний метод»* ***public static void main (String[] args)****, з якого розпочнеться виконання програми;*
- ✓ *тіло методу складається з послідовності інструкцій програми, поміщеної в фігурні дужки, кожна команда закінчується знаком"****;****";*
- ✓ *виконання методу може завершується командою* ***return****, після якої вказується значення, що буде повернене в якості результату методу, метод може не повертати жодного значення, в такому випадку тип його результату позначається службовим словом* ***void****;*
- ✓ *в тілі методу, що не повертає значення (****void****) команда* ***return*** *не обов'язкова, за її відсутності виконання методу завершиться після виконання усіх його команд;*
- ✓ *змінні та константи, що використовуються тілі методу можуть бути оголошені будь-де, але до їх першого використання, змінні оголошені в класі за межами методу називають полями класу, вони доступні в усіх методах класу;*
- ✓ *для створення іменованих констант використовується службове слово* ***final****, записане перед типом константи;*
- ✓ *коментарі, тобто, пояснення до програми, записують у рядку, після пари знаків «слеш» ("****//****"), або між парами символів "****/****" та "*****/****".*

Найпростіший спосіб консольного введення числових даних в програмах на Java, – використання методів класу java.util.Scanner. Саме його використано в коді далі. Для консольного виводу дійсних чисел у форматі з фіксованою кількістю знаків після десяткової коми можна використовувати метод printf класу PrintStream, який працює аналогічно до одноіменної функції C/C++. Операція піднесення до степеня в ***Java*** відсутня, для обчислення a^x слід скористатися статичним методом ***pow(a, x)*** бібліотечного класу ***Math***.

Решта особливостей Java пропонуємо розглянути в наступному коді:

```
package com.tasks.part1;

//пакет містить клас Scanner, використаний для вводу даних

import java.util.*;

class Task30 {

  public static void main (String[] args) {

    System.out.println("Введіть довжину ребра куба");

    double a; //довжина ребра куба

    // Створюємо об'єкт класу Scanner для вводу даних
```

```
        Scanner scanner = new Scanner(System.in);
        a = scanner.nextDouble();//інструкція вводу даних
        double v;//оголошення змінної для об'єму
        v = a * a * a;   double r3, r;
        r3 = 3 * v /(4 * Math.PI);
        /* усі математичні функції, в тому числі, піднесення
        до степеня є статичними методами бібліотечного кладсу
        Math, тому викликаються від його імені*/
        r = Math.pow(r3, 1.0/3.0);
        //вивід результату в форматі з фіксованою крапкою
        //та трьома знаками після коми
        System.out.printf("Радіус кулі: %10.3f",r);
    }
}
```

Програма на Kotlin.

Для створення програм на ***Kotlin*** використовують такі концепції:

- ✓ *програма може складатися з функцій та об'єктів бібліотечних чи описаних власноруч класів;*
- ✓ *виконання програми починається з головної функції («точки входу»)* **`fun main(args: Array<String>)`**;
- ✓ *функція* **`main(args: Array<String>)`** *може бути описана всередині класу, тоді її необхідно позначити анотацією* **`@JvmStatic`**;
- ✓ *кожна функція складається з заголовку та тіла, заголовок розпочинається з ключового* **`fun`**, *містить назву функції та список параметрів функції в круглих дужках, а також тип результату функції, вказаний через двокрапку після списку параметрів;*
- ✓ *якщо параметри відсутні, дужки в заголовку функції залишають порожніми, якщо функція не повертає визначеного результату, тип в заголовку можна не вказувати;*
- ✓ *тіло функції складається з послідовності інструкцій програми, у фігурних дужках, кожна інструкція розміщується в окремому рядку, дві інструкції розміщені в одному рядку розділяють знаком "**;**";*
- ✓ *виконання функції, що повертає результат має завершуватися командою* **`return`**, *після якої вказується значення, що буде повернене;*
- ✓ *в тілі функції, яка не повертає значення команда* **`return`** *не обов'язкова, за її відсутності виконання функції завершиться після досягнення закриваючої фігурної дужки, що обмежує тіло функції;*

- ✓ *змінні та константи, що використовуються в тілі функції можуть бути оголошені будь-де, але до їх першого використання для присвоєння чи вводу та виводу даних;*
- ✓ *для створення іменованих констант використовується службове слово* ***val****, для змінних –* ***var****, записане перед її іменем;*
- ✓ *тип змінної чи константи можна вказати після її імені через двокрапку, або ж його буде визначено за типом значення, яким її ініціалізують;*
- ✓ *коментарі записують у рядку, починаючи з пари знаків «слеш» (“****//****”), або між парами символів “****/****” та “*****/****”;*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (а для порівняння значень – знак “****= =****”).*

Піднесення до степеня не має окремого оператора, так само як і в ***Java***, для обчислення a^x можна скористатися тим же методом ***pow(a, x)*** бібліотечного Java-класу ***java.lang.Math*** з пакету ***JDK***.

Програма-розв’язок задачі 30 на ***Kotlin***:

```
fun main(args: Array<String>) {
    val PI = 3.141593
    var a: Double //довжина ребра куба
    println("Введіть дожину ребра куба")
    //інструкція вводу дійсного числа
    a = readln().toDouble()
    var v: Double //оголошення змінної для об'єму
    v = a * a * a
    val r: Double
    val r3 = 3 * v / (4 * PI)
    /* усі математичні функції, в тому числі, піднесення
    до степеня є статичними методами бібліотечного класу
    Math, з пакету Java Development Kit*/
    r = java.lang.Math.pow(r3, 1.0 / 3.0)
    //вивід результату р 5-ма знаками після коми
    println("Радіус кулі %10.5f".format(r))
}
```

Завдання 2.

Використуваючи цілочисельні дані та операції на цілими числами, скласти програму для розв’язання задачі згідно варіанту.

Варіанти завдань.

1. Користувач вводить чотирицифрове натуральне число. Знайти суму та добуток його цифр.

2. Задано трицифрове натуральне число. Вивести число, отримане перестановкою його цифр у зворотному порядку.

3. Задано трицифрове ціле число. У ньому закреслили першу цифру ліворуч та дописали її ж праворуч. Вивести отримане число.

4. Задано двоцифрове число. Вивести суму квадратів та добуток його цифр.

5. Задано трицифрове ціле число. Вивести число, отримане при перестановці цифр сотень та десятків заданого числа (наприклад, з отриманого числа 376 утвориться число 736).

6. Задано чотирицифрове натуральне число. Знайти різницю добутку його крайніх цифр та добутку внутрішніх цифр.

7. Задано трицифрове ціле число. Вивести число, отримане перестановкою цифр десятків та одиниць вихідного числа (наприклад, 123 перейде до 132).

8. Задано ціле число, більше за 999. Використовуючи лише одну операцію ділення та одну операцію отримання остачі від ділення, знайти цифру, що задає кількість сотень цього числа.

9. Задано трицифрове ціле число. Вивести спочатку його останню цифру (одиниці), а потім його середню цифру (десятки).

10. Задано ціле число, що означає відстань L у міліметрах. Використовуючи операції цілочисельного ділення та остачі від ділення, записати відстань L у метрах, сантиметрах та міліметрах.

11. Масу M у кілограмах задано цілим числом, більшим за 1000. Використовуючи ділення націло та остачу від ділення, записати масу M в тонах, центнерах та кілограмах (наприклад 12345 кг = 12 тон 3 центнери і 45 кг).

12. Наведено розмір файлу в байтах. Визначити кількість повних кілобайтів, які займає даний файл (1 кілобайт = 1024 байти).

13. Задано трицифрове ціле число. У ньому закреслили першу праворуч цифру та дописали її ліворуч. Вивести отримане число.

14. Задано двоцифрове число. Знайти суму та добуток його цифр.

15. Задано двоцифрове число. Вивести число, отримане перестановкою його цифр.

16. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року був понеділок.

17. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року була субота.

18. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 13 січня цього року була п’ятниця.

19. Дні тижня пронумеровані таким чином: 1 – понеділок, 2 – вівторок, …, 6 – субота, 7 – неділя. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року була середа.

20. Дні тижня пронумеровані таким чином: 1 – понеділок, 2 – вівторок, …, 6 – субота, 7 – неділя. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року був вівторок.

21. Задано ціле число, більше за 999. Використовуючи лише одну операцію ділення та одну операцію отримання остачі від ділення, знайти цифру, що задає кількість десятків цього числа.

22. Задано цілі додатні числа A, B, C. В прямокутнику розміру A × B розміщено максимально можливу кількість квадратів зі стороною C (без накладень). Знайти кількість квадратів, розміщених на прямокутнику, а також площу незайнятої частини прямокутника.

23. Користувач вводить два двоцифрові числа. Утворити з них чотирицифрове число за правилом: 1 цифра = 1 цифра першого числа, 2 цифра = 1 цифра другого числа, 3 цифра = 2 цифра першого числа, 4 цифра = 2 цифра другого числа.

24. Задано п’ятицифрове натуральне число. Визначити добуток його першої, третьої та останньої цифр.

25. Задано шестицифрове натуральне число. Визначити суму та добуток його першої та останньої цифри.

26. Користувач вводить два двоцифрові числа. Утворити з них чотирицифрове число шляхом дописування після першого числа цифр другого числа в зворотному порядку.

27. Користувач вводить два трицифрові числа. Утворити з них чотирицифрове число за правилом: 1 цифра = остання цифра першого числа, 2 цифра = остання цифра другого числа, 3 цифра = 1 цифра першого числа, 4 цифра = 1 цифра другого числа.

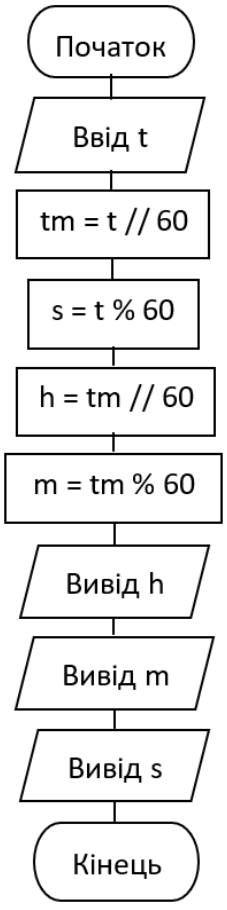
28. У трицифровому числі k закреслили першу цифру. Якщо отримане числ2 помножити на 10 і добуток додати до першої цифри числа k, то отримаємо число n. За заданим числом n, 1< n <999 знайти число k.

29. Від трицифрового числа k відняли його останню цифру. Після ділення результату на 10 до частки ліворуч дописали останню цифру числа k та отримали число m. За заданим числом m знайти число k, вважаючи, що число m належить відрізку [100;999], а середня цифра не дорівнює нулю.

30. Задана кількість секунд, що пройшла від початку доби до поточного моменту. Вивести поточний момент часу в годинах, хвилинах та секундах.

Приклад розв'язування задачі 2

Варіант 30. Задана кількість секунд, що пройшла від початку доби до поточного моменту.



Вивести поточний момент часу в годинах, хвилинах та секундах. Задачі другого блоку цього розділу відрізняються від попередніх тим, що передбачають виконання обчислень з цілими числами, зокрема ділення з остачею. Відповідно, в програмах розв'язування задачі 0 слід використовувати так звані «цілочисельні» арифметичні операції обраної мови програмування. Сам алгоритм розв'язування залишається лінійним:

- *користувач вводить ціле число t, що задає кількість секунд, яка пройшла від опівночі;*
- *програма знаходить цілу частину від ділення заданого числа на 60, число tm, що буде означати кількість повних хвилин, що пройшла від опівночі;*
- *програма знаходить остачу s від ділення введеного користувачем числа t на 60, яка показує кількість секунд, що пройшла з останньої повної хвилини;*
- *обчислену на другому кроці цього алгоритму кількість хвилин tm знову ділимо на 60: знайдена ціла частина h буде вказувати кількість годин від опівночі, а остача m, – кількість хвилин, що минула від початку поточної години;*

- *знайдені кількості годин, хвилин і секунд програма повинна послідовно повідомити користувачеві і після цього завершити свою роботу.*

Програма на Java.

Для вводу даних в коді нижче використано не допоміжний клас ***Scanner***, як у прикладі до задачі 30, а спосіб, із вводом текстової стрічки та подальшим отриманням числових даних з неї. Для цього в ***Java*** використовуються класи ***InputStreamReader*** та ***BufferedReader***. Щоправда, метод зчитування стрічки тексту ***readLine()*** передбачає опрацювання виняткової ситуації ***IOException***, але його можна делегувати зовнішньому коду (в нашому випадку середовищу виконання програми) оператором ***throws***.

Цілочисельне ділення в ***Java,*** так само, як і ***C/C++***, окремого оператора не має, а виконується операторм «/» для цілочисельних операндів автоматично. Коду на ***Java*** для поділу часу на години, хвилини, секунди:

```
package com.tasks.par1;


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Task60 {
    public static void main(String[] args)
                                throws IOException {
        int t, tm; // змінні можна оголошувати наперед
        System.out.println("Введіть час в секундах");
        // об'єкти для читання даних вхідного потоку
        InputStreamReader inputStreamReader =
                new InputStreamReader(System.in);
        BufferedReader bufferedReader =
                new BufferedReader(inputStreamReader);
        // зчитування тексту з консолі
        String str = bufferedReader.readLine();
        // отримання числа з введеного тексту
        t = Integer.parseInt(str);
        tm = t / 60;
```

```
        int s = t % 60;
        int h = tm / 60;
        int m = tm % 60;
        // форматований вивід в консоль
        System.out.printf("Час %02d:%02d:%02d", h, m, s);
    }
}
```

Програма на Kotlin.

Стовідсоткова сумісність ***Kotlin*** та ***Java*** дозволяє на перекласти вище поданий код з ***Java*** на ***Kotlin*** «дослівно». В ***IntelliJ IDEA*** для цього, зокрема є спеціальний інструмент, що запускається командою ***Convert Java File to Kotlin File*** (комбінація клавіш Ctrl+Alt+Shift+K). Після перекладу отримаємо цілком робочий об’єктно-орієнтований Kotlin-код:

```
package com.tasks.par1
import java.io.BufferedReader
import java.io.IOException
import java.io.InputStreamReader

object Task60 {
  @Throws(IOException::class)
  @JvmStatic
  fun main(args: Array<String>) {
    val t: Int
    val tm: Int // змінні можна оголошувати наперед
    println("Введіть час в секундах")
    // об'єкти для читання даних вхідного потоку
    val inputStreamReader =
            InputStreamReader(System.`in`)
    val bufferedReader =
            BufferedReader(inputStreamReader)
    // зчитування тексту з консолі
    val str = bufferedReader.readLine()
    // отримання числа з введеного тексту
```

```
        t = str.toInt()
        tm = t / 60
        val s = t % 60
        val h = tm / 60
        val m = tm % 60
        // форматований вивід в консоль
        System.out.printf("Час %02d:%02d:%02d", h, m, s)
      }
    }
```

Отримана програма при першому запуску буде працювати повільніше за свою «Java-копію», оскільки її для виконаня її байт-коду на віртуальній машині ***Java*** (***JVM***) спочатку потрібно виконати компіляцію з ***Kotlin***.

Сам код можна суттєво скоротити, скориставшись реалізованою в ***Kotlin*** структурною парадигмою. За такого підходу для розв'язування простої обчислювальної задачі нам достатньо створити головну функцію, а не цілий клас, як в ***Java*** (***JVM***). А також використовувати для вводу даних функцію ***readln()***, замість екземплярів класів ***InputStreamReader*** та ***BufferedReader***.

Процедурно-орієнтований код розв'язування задачі 60 на ***Kotlin***:

```
package com.tasks.par1


fun main(args: Array<String>) {
    println("Введіть число секунд, " +
       "що пройшли від початку доби: ")
    // зчитування цілого числа з консолі
    //інструкція вводу даних
    var t: Int = readln().toInt()
    val tm = t / 60
    val s = t % 60
    val h = tm / 60
    val m = tm % 60
    println("%02d".format(h) + ":" +
        "%02d:%02d".format(m,s))
}
```