*Міністерство освіти і науки України*

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря
Сікорського» Інститут телекомунікаційних систем

Кафедра телекомунікацій

*Протокол з лабораторної роботи №2*

*«Оператори та типи даних»*

Виконав студ. гр.ТЗ-11

Варіант №16

Пікалов Андрій

Київ 2023

# ЗАВДАННЯ:

Таблиця 1. Таблиця варіантів

| Варіант | Завдання № 1 | Завдання № 2 | Завдання № 3 |
|---|---|---|---|
| 1 | 1 | 2 | 3 |
| 2 | 4 | 5 | 6 |
| 3 | 7 | 8 | 9 |
| 4 | 10 | 11 | 12 |
| 5 | 13 | 14 | 15 |
| 6 | 16 | 17 | 18 |
| 7 | 19 | 20 | 21 |
| 8 | 22 | 23 | 24 |
| 9 | 25 | 26 | 27 |
| 10 | 28 | 29 | 30 |
| 11 | 1 | 11 | 21 |
| 12 | 2 | 12 | 22 |
| 13 | 3 | 13 | 23 |
| 14 | 4 | 14 | 24 |
| 15 | 5 | 15 | 25 |
| 16 | 6 | 16 | 26 |
| 17 | 7 | 17 | 27 |
| 18 | 8 | 18 | 28 |
| 19 | 9 | 19 | 29 |
| 20 | 10 | 20 | 30 |
| 21 | 1 | 20 | 29 |
| 22 | 11 | 2 | 30 |
| 23 | 21 | 12 | 3 |
| 24 | 4 | 22 | 13 |
| 25 | 14 | 5 | 23 |
| 26 | 24 | 15 | 6 |
| 27 | 7 | 25 | 16 |
| 28 | 17 | 8 | 26 |
| 29 | 27 | 18 | 9 |

# ВИКОНАННЯ РОБОТИ (вигляд коду):

```
public class Lab_2 {

    public static void main(String[] args) {

        double a1 = -1.23;
        double b1 = -0.34;
        double c1 = 0.707;
        double d1 = 2.312;

        double squareRoot = Math.sqrt(Math.sin(c1) + Math.exp(d1));
        double log1 = Math.log10(Math.abs(b1/a1));
        double res1 = 3*(log1 + squareRoot);
        System.out.println("1st exercise: " + res1);
        System.out.println( );

        double a2 = 0.58;
        double b2 = 0.34;
        double c2 = 1.25;
        double d2 = -1.89;

        double fraction = 2*Math.sin(a2)/Math.acos(-2*b2);
```

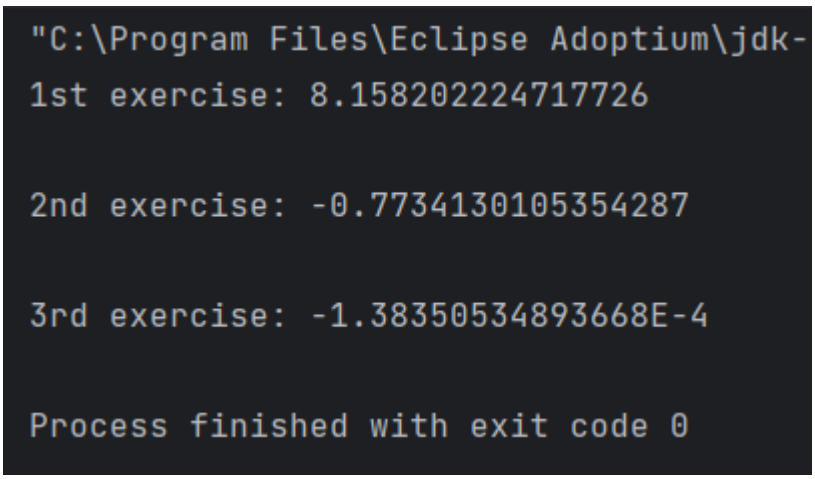
```
        double log2 = Math.log(c2*Math.abs(2*d2));
        double square = Math.sqrt(log2);
        double res2 = fraction - square;
        System.out.println("2nd exercise: " + res2);
        System.out.println( );

        double a3 = 1.27;
        double b3 = 10.99;
        double c3 = 4.0;
        double d3 = -25.32;

        double numerator = Math.pow(Math.tan(a3), 1.0 / c3);
        double denomerator = 2 - Math.sinh(b3)/Math.log(Math.abs(d3+c3));
        double res3 = numerator/denomerator;
        System.out.println("3rd exercise: " + res3);
    }

}
```

**Результат виконання**




## КОНТРОЛЬНІ ПИТАННЯ:

1. **В чому полягає різниця між ключовими та зарезервованими словами?**

Зарезервовані слова не можна використовувати для ініціалізування змінних.

2. **Які примітивні типи даних Ви знаєте? Для кожного з них наведіть приклади, коли найбільш ефективно використовувати саме цей тип (наприклад: довжина файлу, кількість зірок на небі, рахунок у футбольному матчі, маса всесвіту, заробітна платня, ...).**

1) Цілі числа - byte, short, int, long

2) Числа з плаваючою точкою (інакше речові) - float, double

3) Логічний - boolean

4) Символьний – char

Наприклад: чек у магазині – float, кількість місяців у році – int, настільна гра «правда/неправда» - Boolean.

3. **Що таке знакові та беззнакові типи? До якої групи відноситься кожен з примітивних типів даних?**

Знаковий тип призначений як для від'ємних так і для додатніх чисел (включаючи нуль), а беззнакові типи – лише для значень, великих або рівних нулю.

4. **Яке максимальне число можна записати у змінну типу short, char, int, long?**

- short приймає значення від -32768 до 32767 і займає 2 байти пам'яті
- int від -2147483648 до 2147483647 і займає 4 байти пам'яті
- long від -9223372036854775808 до +9223372036854775807 і займає 8 байтів пам'яті
- char від 0 до 65536, замймає 2 байта

5. **Змінна типу byte може приймати значення у діапазоні -128..127. Чому цей діапазон саме такий? Чому від'ємних значень більше ніж додатних?**

Тому що byte займає 1 байт пам'яті. Пояснення, чому від'ємних значень більше:

| | |
|---|---|
| 10000000 = -128 (-128 + 0) | 00000000 = 0 (0 + 0) |
| 10000001 = -127 (-128 + 1) | 00000001 = 1 (0 + 1) |
| 10000011 = -125 (-128 + 3) | 00000011 = 3 (0 + 3) |
| 10000111 = -121 (-128 + 7) | 00000111 = 7 (0 + 7) |
| the biggest number possible | 011111111 = 127 |

6. **Що таке система числення? Як переводити значення з однієї системи числення у іншу?**

Система числення – символічний метод запису чисел, сукупність правил і законів, що застосовуються для позначення будь-якого невід'ємного числа.

Різноманітні прийоми числових переводів визначаються відповідно до видів чисел. Операції проводяться з трьома типами чисел – цілими, від'ємними, дробовими. Необхідність перевести число між системами числення породжує правило №1: спочатку перетворити числовий символ у десяткову систему.

Розрахунки можна проводити вручну, самостійно розписуючи послідовності, або шляхом використання універсального електронного калькулятора.

**7. Що таке «Доповняльний код»?**

**Доповняльний код –** спосіб подання від'ємних чисел у комп'ютерах.

**8. Пояснити результат роботи такого фрагмента коду:**

```
int a = 2_000_000_000;
int b = 2_000_000_000;
int c = a + b;
System.out.println(c);
```

Тип змінних a і b int, переповнює ліміт значень, які можливо прийняти.

System.out.println виведе -294967296.

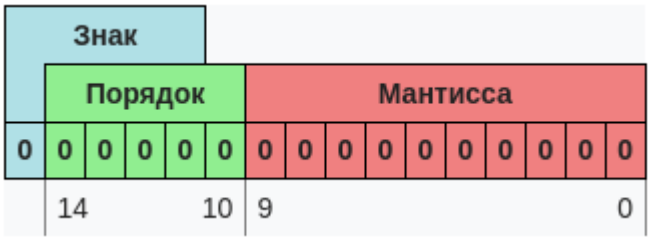
**9. Чим тип float відрізняється від double?**

Double виводить більш точне значення, має більше цифр після коми.

**10. Що таке число з рухомою комою? Що таке мантиса та степінь?**

**Число з рухомою комою** — форма подання дійсних чисел, в якій число зберігається у формі мантиси і показника степеня.

**Мантиса –** частина двійкого числа, яка підноситься у степінь.




Порядок = степінь

**11.Що таке Double.NaN?**

Double.NaN - це спеціальне значення (Not-a-Number) в мові програмування Java, яке використовується для позначення результатів неможливих або неозначених обчислень з числовими значеннями типу double.

Основне призначення Double.NaN полягає в тому, щоб вказати, що результат обчислення не є дійсним числом. Наприклад, якщо ви ділите число на нуль або виконуєте математичні операції, які не мають сенсу (наприклад, ділення нуля на нуль), результатом може бути Double.NaN. Це допомагає програмістам розпізнати, що щось пішло не так у обчисленнях і потрібно перевірити або виправити код.

**12. Пояснити результат роботи такого фрагменту коду:**

System.out.println(Double.NaN==Double.NaN);

**False**

Це пов'язано з особливістю роботи зі спеціальним значенням **Double.NaN** (Not-a-Number) у мові програмування Java і не тільки. **Double.NaN** є спеціальним значенням, яке вказує на неозначеність або неможливість обчислень. Особливість полягає в тому, що **Double.NaN** не рівне ніякому іншому значенню, включаючи інше **Double.NaN**.

**13. Пояснити різницю між преінкрементом та постінкрементом. Навести приклади, коли ці операції призводять до різних результатів.**

Преінкремент і постінкремент - це два способи збільшення значення змінної на 1 в мові програмування. Різниця між ними полягає в тому, коли саме відбувається збільшення значення та коли повертається початкове значення змінної.

Преінкремент (++x): У цьому випадку змінна збільшується на 1 перед тим, як вона використовується в виразі. Тобто, спочатку збільшується значення змінної, а потім використовується нове значення.

Приклад:

int x = 5;

int y = ++x;

// Тепер x = 6, y = 6

Постінкремент (**x++**): У цьому випадку змінна використовується в виразі, і після цього її значення збільшується на 1. Тобто, спочатку використовується початкове значення змінної, а потім вона збільшується.

Приклад:

int x = 5;

int y = x++;

// Тепер x = 6, y = 5

Різниця між ними стає важливою в тих випадках, коли потрібно контролювати порядок виразів або коли ви використовуєте результат операції преінкременту або постінкременту в інших виразах. Переінкремент і постінкремент можуть призвести до різних результатів, і важливо зрозуміти, як вони працюють для правильного використання в програмах.

**14. В чому різниця між логічною та побітовою операцією AND? Чому звичайна операція AND (&) є в обох варіантах, а її короткозамкнута версія (&&) лише тільки логічна?**

Логічна операція AND (**&&**) та побітова операція AND (**&**) мають різний призначення і використовуються в різних контекстах. Логічна операція AND використовується для порівняння логічних умов (булевих значень), вона повертає **true**, якщо обидві умови **true**. Побітова операція AND використовується для виконання операцій над окремими бітами чисел і вона застосовується до кожного біта окремо.

Побітова операція AND (**&**) виконує логічне AND над кожним бітом відповідних позицій чисел, незалежно від їхнього значення. Логічна операція AND (**&&**), натомість, використовується для перевірки логічних умов і вона має короткозамкнуту поведінку, що означає, що в разі, якщо перша умова вже визначає результат (наприклад, **false**), друга умова не обчислюється. Це може призвести до покращення продуктивності і уникнення непотрібних обчислень.

Отже, різниця полягає в контексті та призначенні операцій. Логічна операція AND використовується для порівняння булевих умов, а побітова операція AND використовується для роботи з окремими бітами чисел. І короткозамкнута версія **&&** використовується в контексті логічних умов з можливістю оптимізації обчислень.

**15. В чому різниця між OR та XOR?**

Оператор OR (АБО) виконується 2ма бітами (a і b). Результат дорівнює 0, якщо a і b дорівнюють 0, інакше він дорівнює 1.

Оператор XOR (виключаюче АБО) виконується також двома бітами a і b. Результат виконання операції дорівнює 1, коли один з бітів = 1. У інших ситуаціях результат = 0.

**16. Проаналізувати наступний фрагмент програми. Передбачити його результат. Запустити. Пояснити.**

```
boolean a = true && false | false;

System.out.println(a = false);
```

boolean b = true && false || false;

System.out.println(b = true);

1. **boolean a = true && false | false;**
    - Спочатку виконується операція **&&**, яка має вищий пріоритет. **true && false** дорівнює **false**.
    - Потім виконується побітова операція **|** (OR). **false | false** дорівнює **false**.
    - Отже, **a** отримує значення **false**.
    - В останньому виразі **System.out.println(a = false);** значення **false** присвоюється змінній **a**, і це значення виводиться на консоль.

Результат виводу першого рядка буде: false.

2. **boolean b = true && false || false;**
    - Спочатку виконується операція **&&**, яка має вищий пріоритет. **true && false** дорівнює **false**.
    - Потім виконується операція **||** (OR), яка також має вищий пріоритет. Оскільки перший операнд **false**, то результат виразу вже відомий, і другий операнд **false** навіть не обчислюється.
    - Отже, **b** отримує значення **false**.
    - В останньому виразі **System.out.println(b = true);** значення **true** присвоюється змінній **b**, і це значення виводиться на консоль.

Результат виводу другого рядка також буде: false.

Отже, результати виводу обох рядків будуть **false**, оскільки обидва вирази призводять до присвоєння значення **false** змінним **a** і **b** відповідно.