**Лабораторна робота № 2**

**УМОВНІ ТА ЦИКЛІЧНІ КОНСТРУКЦІЇ В JAVA.**

**Мета роботи:** засвоїти технологію створення програм, в яких реалізуються розгалуження і цикли.

**Постановка задачі**

У цій роботі мова піде про керуючі конструкції: умовні оператори і оператори циклу.

*Умовні оператори* використовуються і призначені для створення точок розгалуження в програмі, коли той чи інший код виконується в залежності від того, виконується чи ні деяка умова. Оператори циклу потрібні для багаторазового виконання однотипних дії.

Для умовного оператора if розглянемо стандартний спосіб виклику, що складається з двох ключових слів: if і else. Синтаксис виклику умовного оператора такий:

if(*умова*) {*команди*\_1}

else {*команди*\_2}

Після ключового слова if в круглих дужках вказується *умова*. Цей вираз повертає логічний результат. Якщо значення умова дорівнює true (*умова виконана*), то виконуються команди, які розміщені в блоці з фігурних дужок відразу після if - інструкції (*команди\_1*). Якщо ж значенням умови є false (умова не виконана), то виконуються команди в блоці після ключового слова else (*команди\_2*).

Досить популярною є конструкція з вкладених умовних операторів. Ідея досить проста: в одному умовному операторі викликається інший умовний оператор, в якому може бути виклик третього умовного оператора, і так далі. Синтаксис команди з таким ієрархічним викликом умовних операторів наступний:

if (*умова\_1*) {*команди\_1*}

else if (*умова\_2*) {*команди\_2*}

else if (*умова\_3*) {*команди\_3*}

…

else if (*умова\_N*) {*команди\_N*}

else {*команди\_N+1*}

*Оператор багатоваріантного розгалуження* - оператор switch може виявитися значно зручніше конструкції if / else, якщо необхідно реалізувати вибір з багатьох варіантів. Синтаксис у оператора наступний.

switch (*вираз*) {

case *значення\_1:*

*команди\_1*

break;

case *значення \_2:*

*команди\_2*

break;

**…**

Case *значення \_N:*

*команди\_N*

break;

default:

*команди\_N+1*

*}*

Спочатку вказується ключове слово switch. В круглих дужках після цього слова вказується вираз, значення якого перевіряється в результаті виконання оператора. Результатом виразу може бути ціле число або символ. Потім йде блок з фігурних дужок. Блок складається з case-інструкцій. У case-інструкції після ключового слова case вказується можливе значення виразу, і завершується все двокрапкою. Далі йдуть команди. Це саме ті команди, які виконуються, якщо вираз приймає значення, вказане після ключового слова case.

*Невизначені цикли* while і їх модифікація do-while дозволяють створювати в програмі цикли з умовами, тобто виконувати якусь кількість разів одні й ті ж команди поки умова дорівнює true. Спочатку розглянемо оператор циклу while. Оператор має досить простий синтаксис

while (*умова*) {

*команди*

}

Виконується оператор while так. Спочатку обчислюється значення виразу, вказаного в круглих дужках після ключового слова while (перевіряється умова). Якщо значення виразу дорівнює true (умова істинно), виконується блок команд в фігурних дужках. Потім знову перевіряється умова. Якщо умова істинно, знову обчислюються команди основного тіла оператора while і перевіряється умова. Як тільки для виразу-умови отримано значення false, виконання оператора while припиняється і управління передається наступному після нього оператору.

Умова циклу while перевіряється з самого початку. Отже, можлива ситуація, при якій оператори, що містяться в блоці, не будуть виконані ніколи.

Якщо необхідно, щоб блок виконувався, хоча б один раз, перевірку умови потрібно перенести в кінець, це можна зробити за допомогою модифікованого оператора while, який записується в такий спосіб

do {

*команди*

} while(*умова*);

Після ключового слова do в фігурних дужках вказуються команди основного тіла операторів циклу. Після блоку команд слідує ключове слово while, після якого в круглих дужках вказано умова (вираз логічного типу). Закінчується вся ця конструкція крапкою з комою.

Умова перевіряється лише після виконання тіла циклу. Потім тіло циклу повторюється, знову перевіряється умова і так далі.

*Визначені цикли for* - дуже поширена мовна конструкція. У ній число повторення контролюється змінної, яка виконує роль лічильника і оновлюється на кожному кроці ітерації. Синтаксис оператора наступний:

for (*ініціалізація; умова; зміна значень*){

*команди*

}

Починається все з ключового слова for, після якого йдуть круглі дужки. В круглих дужках три блоки виразів. Блоки розділяються крапкою з комою. Команди всередині блоку розділяються комами.

Перший елемент оператора for зазвичай виконує ініціалізацію лічильника, другий формує умову виконання тіла циклу, а третій визначає спосіб поновлення лічильника.

Починається виконання оператора циклу for з виконання команди блоку ініціалізації. Ця команда виконується один-єдиний раз і тільки спочатку роботи оператора циклу. Після їх виконання обчислюється вираз, розміщене в другому блоці. Вираз повинен мати логічний тип, тому зазвичай називається умовою. Оператор циклу виконується, поки істинно умова.

**Введення з клавіатури**

**1. Читання з консолі, System.in**

У попередніх лекціях ми ознайомилися з командами виведення на екран. Для цього ми використовували об'єкт System.out і його методи print() і println().

Так само як для виведення, для введення даних теж є спеціальний об'єкт — System.in. Але, на жаль, він не настільки зручний, як нам хотілося б. Цей об'єкт дає змогу зчитувати дані з клавіатури по одному символу за раз.

Ось чому ми скористаємося ще одним класом, який у парі з об'єктом System.in дасть нам усе, що потрібно.

**2. Клас Scanner**

Клас Scanner (повне ім'я java.util.Scanner) вміє зчитувати дані з різних джерел: консолі, файлів, інтернету. Якщо ми хочемо, щоб він зчитував дані з клавіатури, ми маємо передати йому об'єкт System.in як параметр — джерело даних. А об'єкт типу Scanner вже сам розбереться, що з ним робити.

Зчитування з клавіатури за допомогою об'єкта типу Scanner матиме приблизно такий вигляд:

Scanner console = new Scanner(System.in);

(Створюємо об'єкт класу Scanner.)

String name = console.nextLine();

(Читаємо з клавіатури рядок тексту.)

int age = console.nextInt();

(Читаємо з клавіатури число.)

Приклад повної програми, в якій використовується клас Scanner:

import java.util.Scanner;

public class Solution {

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

String name = console.nextLine();

int age = console.nextInt();

System.out.println("Name: " + name);

System.out.println("Age: " + age);

}

}

**Завдання для виконання**

1. Вивести на екран числа від 1 до 5 (використовуючи оператор циклу).
2. Вивести на екран числа від 5 до 1 (використовуючи оператор циклу).
3. Вивести на екран таблицю множення.
4. Напишіть програму, у якій користувач вводить будь-яке позитивне число. Програма підсумовує всі числа від 1 до введеного числа і виводить суму на екран.
5. У циклі програма запитує у користувача на введення два числа і виводить результат їх множення. Після виводу результату множення програма запитує про необхідність завершення виконання. Якщо користувач вводить число 1, то програма завершує цикл. Якщо введено будь-яке інше число, то програма повторює запит у користувача двох чисел і перемножує їх.