Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

Звіт
про виконання лабораторної роботи №8
на тему:
«Логічний та імперативний підходи»

Виконала:
Студентка групи ФеС-32
Філь Дарина

Львів - 2025

Мета роботи:

Порівняти між логічний та імперативний підходи.

Теоретичні відомості:

Логічна парадигма програмування — це підхід до програмування, який базується на використанні логіки для опису проблеми, замість традиційного опису алгоритмів. У цій парадигмі програміст визначає факти та правила, з яких система робить висновки для вирішення задачі.

Основними елементами логічного програмування є:

1. Факти — це твердження, які приймаються як істинні. Наприклад, у базі знань можна мати факт: батько(Іван, Петро)., що означає, що Іван — батько Петра.

2. Правила — це умовні твердження, що вказують, як можна отримати нові факти. Наприклад: батько(X, Y) :- чоловік(X), дитина(Y). (Якщо X — чоловік, а Y — дитина, то X — батько Y).

Програмування в логічній парадигмі зазвичай використовує пролог — одну з найвідоміших мов програмування в цій парадигмі.

Основні принципи:

● Декларативність: програміст не визначає, як саме повинні виконуватись операції, а лише що є істинним в контексті задачі.

● Розв'язання задачі через запити: програма генерує відповіді на запити, обираючи з наданих фактів і правил.Приклад виводу результати програми світлофор:

Щоб запустити програми на Prolog на Windows, вам потрібно встановити реалізацію Prolog, таку як SWI-Prolog, та налаштувати середовище для роботи з програмами.

Завдання:

1. Використовуючи SWI-Prolog (скорочено SWI або swipl) або іншу реалізацію мови prolog, написати програму світлофор (завдання 1).
2. Реалізувати програму з п.2 мовою python (або іншою мовою, використавши імперативний підхід).
3. Написати програму на мові prolog, яка розв'язує квадратне рівняння (завдання 2).
4. Порівняти результати з лабораторною роботою 1. У висновку вказати для яких задач зручніше використовувати дані парадигми і чому (порівняти завдання 1 та завдання 2)?

Хід роботи

1. Використовуючи SWI-Prolog (скорочено SWI або swipl) або іншу реалізацію мови prolog, написати програму світлофор (завдання 1).

```
trafic_light:-
    write('Enter light (green, orange, red) or exit'), read(X),
    (X == 'exit' -> write('Exiting program...'), nl, false;
     X == 'red' -> write('Stop'), nl;
     X == 'orange' -> write('Wait'), nl ;
     X == 'green' -> write('Go'), nl;
     write('Invalid input'), nl),
    trafic_light.
```

Вивід:

```
?- trafic_light.
Enter light (green, orange, red) or exit
|: red.
Stop
Enter light (green, orange, red) or exit|: green.
Go
Enter light (green, orange, red) or exit|: orange.
Wait
Enter light (green, orange, red) or exit|: blue.
Invalid input
Enter light (green, orange, red) or exit|: exit.
Exiting program...
false.
```

2. Реалізувати програму з п.2 мовою python (або іншою мовою, використавши імперативний підхід).

```python
def sum_of_squares(N):  2 usages
    result = 0
    for item in range(1, N + 1):
        result += item**2
    return result

print(sum_of_squares(0))
print(sum_of_squares(3))
```

Вивід:

```
PS E:\СПП\Lab8> python sum_of_squares.py
0
14
```

3. Написати програму на мові prolog, яка розв'язує квадратне рівняння (завдання 2).

```
quadratic(A, B, C):-
    D is B*B - 4 * A * C,(
    D < 0 -> write('Is Complex'),nl, false;

    D =:= 0 ->
    R1 is (- B + sqrt(D))/ (2 * A),
    write(R1), nl ;

    D > 0 ->
    R1 is (- B + sqrt(D))/ (2 * A),
    R2 is (- B - sqrt(D))/ (2 * A),

    write(R1), nl,
    write(R2)).
```

Вивід:

```
?- quadratic(1, -7, 10)
5.0
2.0
true.

?- quadratic(4, 4, 1).
-0.5
true.

?- quadratic(2, 3, 5).
Is Complex
false.
```

Висновок:

У межах цієї лабораторної роботи я реалізувала однакові задачі в двох різних парадигмах програмування: імперативній (на Python) та логічній (на Prolog). Перше завдання — обчислення суми квадратів — виявилося значно зручнішим для реалізації в імперативному стилі завдяки простому використанню циклів. Друге завдання — розв'язання квадратного рівняння — також було легше реалізувати у Python, оскільки мова дозволяє просто і наочно працювати з умовними конструкціями. Хоча логічна парадигма є дуже потужною, вона більше підходить для задач, де потрібно знаходити рішення на основі заданих правил чи знань — наприклад, в експертних системах або при пошуку в графах. Натомість імперативний підхід зручніший для послідовних математичних розрахунків і алгоритмічної логіки.

Додаток:

quadratic.pl

```
quadratic(A, B, C):-
    D is B*B - 4 * A * C,(
    D < 0 -> write('Is Complex'),nl, false;

    D =:= 0 ->
    R1 is (- B + sqrt(D))/ (2 * A),
    write(R1), nl ;

    D > 0 ->
    R1 is (- B + sqrt(D))/ (2 * A),
    R2 is (- B - sqrt(D))/ (2 * A),

    write(R1), nl,
    write(R2)).
```

quadratic.py

```python
def quadratic(a, b, c):
    d = b**2 - 4 * a * c
    if d < 0:
        return "Is Complex"
    elif d == 0:
        x = (-b + d**0.5) / (2 * a)
        return f"{x}"
    else:
        x1 = (-b + d**0.5) / (2 * a)
        x2 = (-b - d**0.5) / (2 * a)
        return f"{x1}\n{x2}"

print(quadratic(1, -3, 2))
```

sum_of_squares.py

```python
def sum_of_squares(N):
    result = 0
    for item in range(1, N + 1):
        result += item**2
    return result

print(sum_of_squares(0))
print(sum_of_squares(3))
```