Міністерство освіти і науки України

Чорноморський національний університет імені Петра Могили

**Старченко В. В.**

# ЛОГІЧНЕ ПРОГРАМУВАННЯ (PROLOG)

***Методичні вказівки***

**Випуск 414**

Миколаїв–2023

УДК 004.434.021:510.755](076)

**С77**

*Рекомендовано до друку вченою радою Чорноморського національного університету імені Петра Могили від 29.09.2022 протокол № 7*

***Рецензент:***
***Обрубов А. В. –*** канд. техн. наук, доцент кафедри суднових електроенергетичних систем НУК ім. адм. Макарова.

**С77** **Старченко В. В.** Логічне програмування (Prolog) : метод. вказівки / В. В. Старченко – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. – 48 с. – (Методична серія ; вип. 414).

Методичні вказівки призначені для студентів бакалаврату всіх спеціальностей галузі знань 12 «Інформаційні технології», що вивчають дисципліни «Функціональне і логічне програмування», «Інтелектуальне керування робототехнічними системами», «Методи та системи штучного інтелекту» та ін., пов’язані з реляційним програмуванням, розробкою медичних експертних систем, програмуванням роботів, моделюванням та дослідженнями систем штучного інтелекту.

УДК 004.434.021:510.755](076)

**ISSN 1811-492X**

© Старченко В. В., 2023
© ЧНУ ім. Петра Могили, 2023

# ЗМІСТ

ВСТУП ..... 1
РОЗДІЛ 1 Практичні роботи ..... 2
1 Практична робота № 1 ..... 2
1.1 Теоретичні відомості ..... 2
1.2 Тренувальні завдання ..... 3
1.3 Приклад виконання завдання ..... 5
1.4 Залікові завдання ..... 6
1.5 Контрольні питання ..... 6
2 Практична робота № 2 ..... 8
2.1 Теоретичні відомості ..... 8
2.2 Тренувальні завдання ..... 9
2.3 Залікові завдання ..... 11
2.4 Контрольні питання ..... 12
3 Практична робота № 3 ..... 13
3.1 Теоретичні відомості ..... 13
3.2 Залікові завдання ..... 13
3.3 Контрольні питання ..... 15
4 Практична робота № 4 ..... 16
4.1 Теоретичні відомості ..... 16
4.2 Залікові завдання ..... 17
4.3 Контрольні питання ..... 19
5 Практична робота № 5 ..... 21
5.1 Теоретичні відомості ..... 21
5.2 Залікові завдання ..... 21
5.3 Контрольні питання ..... 24
6 Практична робота № 6 ..... 25
6.1 Теоретичні відомості ..... 25
6.2 Залікові завдання ..... 26
6.3 Контрольні питання ..... 27
7 Практична робота № 7 ..... 28
7.1 Теоретичні відомості ..... 28
7.2 Залікові завдання ..... 29
7.3 Контрольні питання ..... 31

РОЗДІЛ 2 Самостійні роботи .....33
1. Вимоги до звіту .....33
2. Порядок подання звіту .....33
3. Порядок зарахування роботи.....33
4. Завдання до самостійної роботи № 1 .....34
5. Контрольні питання до самостійної роботи № 1.....37
6. Завдання до самостійної роботи № 2 .....37
7. Контрольні питання до самостійної роботи № 2.....44
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ .....45

# ВСТУП

Головною метою цих методичних вказівок є допомога студентам у процесі опанування методології створення комп`ютерних програм з використанням логічного підходу шляхом ознайомлення їх з мовою програмування Prolog - найбільш розповсюдженою у світі реалізацією логічного підходу.

Під час вивчення основ логічного програмування студенти оволодіють основними принципами логічного підходу до створення інформаційно-обчислювальних комплексів. Опанують математичні основи логічного програмування, синтаксис мови Prolog, формати представлення даних різних типів та структур, стандартні предикати і функції. Ознайомляться з особливостями процесу уніфікації термів, роботою механізмів повернення та рекурсивних викликів, як основи роботи Prolog-програм. Також студенти навчаться використовувати систему програмування Prolog для розв`язування типових задач символьної обробки даних, маніпулювання складними структурами даних, формулювання і розв`язування задач штучного інтелекту.

Наведені практичні роботи можуть бути використані під час проведення занять з дисциплін «Функціональне і логічне програмування», «Алгоритми та методи обчислень», «Розпізнавання образів», «Інтелектуальне керування робототехнічними системами», «Методи та системи штучного інтелекту», «Інтелектуальний аналіз даних» для студентів бакалаврату всіх спеціальностей галузі знань 12 «Інформаційні технології».

# РОЗДІЛ 1
# Практичні роботи

## 1 Практична робота № 1

***Загальні поняття мови Prolog***

**Мета:** познайомитися із загальними поняттями мови Prolog, середовищем системи програмування Arity Prolog та процедурою розробки і виконання Prolog-програм.

### 1.1 Теоретичні відомості

Програмування мовою Prolog складається з визначення відношень та постановки питань, що стосуються цих відношень.

Програма мовою Prolog складається з тверджень (фраз, клауз). Речення бувають трьох типів:

— факти;

— правила;

— питання.

Кожне речення закінчується крапкою.

Відношення може визначатися за допомогою фактів (елементів відношення), які перераховують об’єкти, для яких це відношення виконується. Воно може визначатися також за допомогою правил, що спираються на інші правила, або факти. Факти містять елементи множин (константи), або зовсім не містять нічого (нулярні відношення). Константи–це атоми. Атом–ідентифікатор, що починається з малої літери, або будь-який рядок, обмежений одинарними лапками: ’Це теж атом’. Правила містять константи і змінні (абстрактні об’єкти). Змінна–це ідентифікатор, що починається з великої літери.

Процедура–це упорядкована послідовність тверджень про одне й те ж відношення.

Питання схожі на запити до деякої бази даних. Відповідь системи на питання являє собою множину об’єктів, які задовольняють запит.

Процес, у результаті якого Prolog-система встановлює, чи задовольняє об’єкт запиту, достатньо складний і включає в себе логічне виведення, дослідження різноманітних варіантів і, можливо, повернення. Все це робиться автоматично самою Prolog-системою і приховано від користувача.

Кожна Prolog-програма одночасно має три інтерпретації: декларативну, процедурну і поведінкову. В ідеалі процедурної інтерпретації взагалі не повинно бути (так званий чистий Prolog), але практична реалізація Prolog-систем, питання ефективності Prolog-програм примушують зважати на процедурну інтерпретацію. Програмуючи мовою Prolog, треба враховувати процедурні деталі, а також поведінку системи логічного висновку.

**1.2 Тренувальні завдання**

1. Враховуючи, що відношення *parent* визначено (див. рис. 1.1), знайти, якими будуть відповіді Prolog-системи на питання:

    1. *.?- parent( джим, X ).*
    2. *.?- parent( X, джим ).*
    3. *.?- parent( пам, X), parent( X, пат ).*
    4. *.?- parent( пам, X ), parent( X, Y ), parent( Y, джим).*

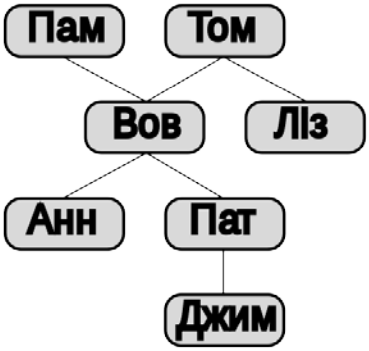


*Рис. 1.1*

2. Сформулюйте мовою Prolog такі питання про відношення *parent*:
    1. Хто народив Пат?
    2. Чи є у Ліз дитина?
    3. Хто є дідом батька Пат?
3. Відтранслюйте такі затвердження в правила мовою Prolog:
    1. Будь-хто, в кого є дитина, – щасливий (введіть одноаргументне відношення *щасливий*).
    2. Будь-який *X*, який має дитину, в якої є сестра, має двох дітей (введіть нове відношення *мати двох дітей*).
4. Визначте відношення *онук*, використовуючи відношення *parent*. *Відповідь:*

    *онук(X, Y) :- parent(X, Y), сестра(Z, Y).*
5. Визначити відношення *тітка( X, Y )* через відношення *parent* та *сестра*.
6. Розглянемо варіант відношення *пращур*:

    *пращур(X, Z) :- parent(X, Z).*

    *пращур(X, Z) :- parent(Y, Z), пращур(X, Y).*

    Чи правильне таке визначення?
7. Спробуйте зрозуміти, як Prolog-система, використовуючи програму (див. рис. 1.1) виводить відповіді на вказані нижче питання. Чи будуть зустрічатися повернення при виводі відповідей на будь-які з цих питань?
    1. *?- parent( пам, боб).*
    2. *?- мати( пам, боб).*
    3. *?- parent_of_parent( пам, енн ).*
    4. *?- parent_of_parent( боб, джим ).*

**1.3 Приклад виконання завдання**

*parent( пам, боб).*

*parent( том, боб).*

*parent( том, ліз).*

*parent( боб, енн).*

*parent( боб, пат).*

*parent( пат, Джим).*

*жінка( пам).*

*жінка( ліз).*

*жінка( енн).*

*жінка( пат).*

*чоловік( том).*

*чоловік( боб).*

*чоловік( джим).*

*паросток( Y, X) :- parent( X, Y).*

*мати( X, Y) :- parent( X, Y), жінка( X).*

*parent_of_ parent( X, Z) :- parent( X, Y),parent( Y, Z).*

*сестра( X, Y) :- parent( Z, X),parent( Z, Y),*

*жінка( X, Y),різні( X, Y).*

*пращур( X, Z) :- parent( X, Z).*

*пращур( X, Z) :- parent( X, Y), пращур( Y, Z).*

### 1.4 Залікові завдання

Складіть власне генеалогічне дерево, використовуючи факти *male*, *female*, *parent*. Визначте правила для якомога більшого числа сімейних відношень, у тому числі для відношень:

1. Чоловік.
2. Брат.
3. Сестра.
4. Дід.
5. Баба.
6. Онук.
7. Онука.

1. Прадід.
2. Прабаба.
3. Двоюрідний брат.
4. Двоюрідна сестра.
5. Дядько.
6. Тітка.

### 1.5 Контрольні питання

1. З яких елементів складається програма мовою Prolog?
2. Які типи речень використовуються у мові Prolog?
3. Який формат має речення мовою Prolog?
4. Як визначається відношення у мові Prolog?
5. Яка арність відношень у мові Prolog?
6. Що таке факт у мові Prolog?
7. Що таке правило у мові Prolog?
8. Що таке атом у мові Prolog?
9. Що таке константа у мові Prolog?
10. Що таке змінна у мові Prolog?
11. Що таке процедура у мові Prolog?
12. Який формат має процедура у мові Prolog?
13. Що таке запит у мові Prolog?

14. Як сформувати запит у мові Prolog?
15. Що таке процедура логічного виведення у мові Prolog?
16. Що таке процедура повернення у мові Prolog?
17. Які інтерпретації має програма мовою Prolog?

# 2 Практична робота № 2

***Синтаксис і семантика Prolog-програм***

**Мета:** познайомитися з синтаксисом і семантикою Prolog-програм.

## 2.1 Теоретичні відомості

Прості об’єкти мови Prolog – це *атоми, змінні, числа.*

Структурні об’єкти, або структури, використовуються для представлення об'єктів, які складаються з декількох компонент.

Структури будуються за допомогою *функторів*. Кожний функтор визначається своїм *ім'ям* і *арністю*, наприклад, *father/2*. Функтор може бути порожнім, наприклад, (1, 2, 3).

У мові Prolog немає визначення типу об’єкта. Тип об’єкта визначається виключно за його синтаксичною формою.

Область дії змінних–одна фраза. Тому одне й те ж ім'я у двох різних фразах визначає дві різні змінні.

Структури можуть бути природним чином зображені у вигляді *дерев*. Prolog можна розглядати як мову обробки дерев.

Задача операції *уніфікації* (зіставлення)–зробити терми ідентичними, підбираючи відповідну конкретизацію змінних в обох термах. Змінні в межах фрази можна зіставити за допомогою оператора ’=’, наприклад, *X* = *Y*.

Уніфікація, якщо вона завершується успішно, у вигляді результату видає найбільш загальну конкретизацію змінних.

Кома між цілями означає їх *кон'юнкцію*. Крапка з комою між цілями означає їх *диз'юнкцію*.

Декларативний сенс програм на «чистому Prolog» не залежить від порядку фраз та від порядку цілей у фразах.

## 2.2 Тренувальні завдання

1. Які із зазначених виразів презентують правильні об’єкти на думку Prolog-системи? Що це за об’єкти (атоми, числа, змінні, структури)?
    1. Діана .
    2. .діана.
    3. ’Діана’.
    4. _діана.
    5. ’Діана йде на південь’.
    6. їде( діана, південь).
    7. 45.
    8. 5( X, Y ).
    9. +( північ, захід).
    10. три( Чорні( Кішки) ).
2. Чи будуть наступні операції успішними або неуспішними? Якщо вони успішні, то яка буде конкретизація змінних?
    1. *точка( A, B ) = точка( 1, 2 ).*
    2. *точка( A, B ) = точка( X, Y, Z ).*
    3. *плюс( 2, 2) = 4.*
    4. *+( 2, D ) = +( E, 2 ).*
    5. *трикутник( точка( -1, 0 ), P2, P3 ) =*
       *трикутник( P1, точка( 1, 0 ), точка( 0, Y )).*
3. Деяка конкретизація визначає родину трикутників.

   *трикутник( точка( -1, 0 ), P2, P3 ) =*

   *трикутник( P1, точка( 1, 0 ), точка( 0, Y )).*

   Як би ви описали цю родину?

   *Відповідь:*

   Така конкретизація визначає родину трикутників, у яких обидві

вершини розташовані на осі *X* в точках 1 і -1, а третя – в довільній точці.

4. Використовуючи таке представлення відрізків *line( X1, Y1, X2, Y2 )*, напишіть терм, який би відповідав будь-якому відрізку на вертикальній прямій *X* = 5.
5. Припустимо, що прямокутник заданий термом *rectangle( P1, P2, P3, P4 )*, де *P* – вершини прямокутника, впорядковані за зростанням. Визначити відношення *regular(R)*, яке має сенс, якщо *R* – прямокутник з вертикальними та горизонтальними сторонами.
6. Розглянемо програму:

   *f( 1,один).*

   *f( s( 1), два).*

   *f( s( s( 1) ), три).*

   *f(( s( s( s( X) ) ), N) :- f( X, N).*

   Як Prolog-система відповість на такі питання?

   1. *?-f( s( 1),A).*
   2. *?-f( s( s( 1) ), два).*
   3. *?-f( s( s( s( s( s( 1) ) ) ) ) ), C).*
   4. *?-f( D, три).*
7. У програмі йдеться про те, що дві людини є родичами, якщо:
   1. Один є пращуром іншого, або.
   2. У них є загальний пращур, або.
   3. У них є загальний нащадок.

   *родич( X, Y) :- пращур( X, Y).*

   *родич( X, Y) :- пращур( Y, X).*

*родич( X, Y) :- пращур( Z, X), пращур( Z, Y).*

*родич( X, Y) :- пращур( X, Z), пращур( Y, Z).*

Чи можна скоротити цю програму, використовуючи запис з крапками з комою?

8. Перепишіть зазначену програму, не користуючись крапками з комою.

   *перетворити( Число, Слово) :-*

   *Число = 1, Слово = один;*

   *Число = 2, Слово =два;*

   *Число = 3, Слово =три.*

**2.3 Залікові завдання**

1. Що буде, якщо Prolog-системі поставити таке питання?

   *?- X = f( X).*

   Успішним або неуспішним буде тут зіставляння? За визначенням уніфікації в логіці, зіставляння повинно бути неуспішним. А що буде відповідно до нашого визначення зіставлення? Поясніть, чому немала кількість реалізацій Prolog відповідає на вищеподане питання так:

   *X = f( f( f( f( f( f( f( f(.....*

2. Використовуючи відношення *conc*, напишіть ціль, яка відповідає викресленню трьох останніх елементів списку *L*, результат – новий список *L1*.

   *Вказівка: L* – конкатенація *L1* і трьохелементного списку.

3. Напишіть послідовність цілей для породження списку *L2*, який отримуємо із списку *L* за допомогою викреслення його трьох перших і трьох останніх елементів.

4. Визначте відношення: *останній(Елемент, Список)* так, щоб *Елемент* був останнім елементом списку *Список*. Напишіть два варіанта визначення:

1. З використанням відношення *conc*.
2. Без використання цього відношення.

## 2.4 Контрольні питання

1. Які об'єкти не є атомарними у мові Prolog?
2. Які об'єкти не є структурними у мові Prolog?
3. Як визначається тип об`єкта у мові Prolog?
4. Який формат має функтор у мові Prolog?
5. Чи може у мові Prolog функтор бути порожнім?
6. Яка область дії змінних у мові Prolog?
7. У чому полягає задача уніфікації термів у мові Prolog?
8. Як відбувається процес уніфікації термів у мові Prolog?
9. За допомогою якої операції відбувається уніфікація термів у мові Prolog?
10. Яким буває результат уніфікація термів у мові Prolog?
11. Який сенс має кома між цілями у програмі мовою Prolog?
12. Який сенс має крапка з комою між цілями у програмі мовою Prolog?
13. Чи залежить декларативний сенс Prolog програми від порядку фраз у неї?
14. Чи залежить процедурний сенс Prolog програми від порядку фраз у неї?
15. Чи залежить поведінковий сенс Prolog програми від порядку фраз у неї?

# 3 Практична робота № 3

***Програмування задач з обробки простих списків***

**Мета:** опанувати методи програмування задач з обробки списків.

## 3.1 Теоретичні відомості

Списки у мові Prolog представляють спеціальну структуру, що введена для спрощення процедур обробки переліків термів. Перелік термів вказується у квадратних дужках.

*[ перелік термів ]*

Усі списки анонімні, тобто не можуть мати власного імені. У списках виділяють голову та хвіст. Розподільником між головою та хвостом є вертикальна риска.

*[ голова | хвіст ]*

Голова списку не може бути порожньою і зображає собою перелік перших елементів списку. Хвіст– список елементів, що йдуть за головою, він може бути порожнім.

Приклади:

*[1, 2, 3, 4, 5], [a, b, c, d, e]*– прості списки.

*[(1), (2), (3, 4)]*– список структур.

*[[1, 2], [a, b], [x, y, z* – список списків.

## 3.2 Залікові завдання

1. Написати предикат, який інвертує список, крім першого елемента.
2. Написати предикат, який інвертує список, крім останнього елемента.
3. Написати предикат, який інвертує список крім другого елемента.

4. Написати предикат, який інвертує список крім передостаннього елемента.
5. Написати предикат, який знаходить мінімальний елемент списку.
6. Написати предикат, який між елементами списку вставляє одиниці.
7. Написати предикат, який зі списку виду *[a, 1, a, 2, b, 3, a, 4, . . . ]* видаляє усі літери ’*a*’.
8. Написати предикат, який замінює місцями передостанній і останній елементи списку.
9. Написати предикат, який переставляє останній елемент списку на його початок.

10. Написати предикат, який передостанній елемент списку переставляє на його початок.

11. Написати предикат, який повертає 10-й елемент списку.

12. Видалити *N*-й елемент зі списку.

13. Видалити 10-й елемент від кінця списку.

14. Написати предикат, який будує список виду *[a-1, b-2, c-3, d-4]*.

15. Написати предикат, який будує список виду *[1-a, 2-b, 3-c, 4-d]*.

16. Написати предикат, який будує список виду *[a-4, b-3, c-2, d-1]*.

17. Написати предикат, який будує список виду *[a-b, c-d, . . . ]*.

18. Написати предикат, який будує список виду *[1-2, 3-4, . . . ]*.

19. Дано список виду *[a, 1, b, 2, c, 3, d, 4, . . . ]*. Написати предикат, який інвертує тільки цифри.

20. Дано список виду *[a, 1, b, 2, c, 3, d, 4, . . . ]*. Написати предикат, який інвертує тільки літери.

21. Дано список виду *[a, 1, b, 2, c, 3, d, 4, . . . ]*. Написати предикат, який літери збирає у першій половині списку, а цифри у другий.

22. Дано список виду *[a, 1, b, 2, c, 3, d, 4, . . . ]*. Написати предикат, який цифри збирає у першій половині списку, а літери у другий.

23. Написати предикат, який генерує список з *N* елементів *'a'*.

24. Написати предикат, який генерує список з *N* елементів, які є числами у порядку зростання. Тобто *[1, 2, 3, 4, ..., N]*.

25. Написати предикат, який генерує список з *N* елементів, які є числами у порядку зменшення.

26. Дано список виду *[a-b, d-c, c-d, d-e, . . . ]*. Написати предикат, який інвертує елементи у парах.

27. Дано список виду *[a, b, d, c, c, d, d, e, . . . ]*. Написати предикат, який підраховує кількість елементів, які дублюються.

28. Є список *[a, b, c, d, . . . ]*. Надрукувати його у стовпчик.

### 3.3 Контрольні питання

1. Який формат мають списки у мові Prolog?
2. Чи існують у мові Prolog іменовані списки?
3. З яких елементів складається список у мові Prolog?
4. Який формат має голова списку у мові Prolog?
5. Який формат має хвіст списку у мові Prolog?
6. Який розподільник треба вказати між головою та хвостом списку у мові Prolog?
7. Який розподільник треба вказати між атомарними елементами списку у мові Prolog?
8. Чи може бути порожньою голова списку у мові Prolog?
9. Чи може бути порожнім хвіст списку у мові Prolog?
10. Чи може голова списку містити вкладені списки у мові Prolog?
11. Чи може хвіст списку містити вкладені списки у мові Prolog?
12. Чи може містити список містити структурні об'єкти у мові Prolog?

# 4 Практична робота № 4

***Програмування задач з обробки вкладених списків***

**Мета:** опанувати методи програмування задач з обробки багаторівневих списків.

## 4.1 Теоретичні відомості

У мові Prolog визначення списку є рекурсивним. Це дозволяє при обробці списків широко використовувати рекурсію. Рекурсія може бути прямою та хвостовою. Як приклад створимо предикат, який визначає довжину списку. Для цього складемо такі логічні твердження:

– довжина порожнього списку дорівнює нулю;

– довжина непорожнього списку на один більше довжині хвоста (якщо у голові один елемент).

З прямою рекурсією цей предикат буде виглядати так:

```
length([], 0).
length([_|T], N):-
length(T, N1),
N is N1+1.
```

Запит до нього буде таким:

```
?- length([a, b, c], X).
```

З хвостовою рекурсією цей предикат буде виглядати так:

```
length([], N, N).
length([_ | T], N0, N):-
N1 is N0+1,
length(T, N1, N).
```

Запит до нього буде таким:

```
?- length([a, b, c], 0, X).
```

**4.2 Залікові завдання**

1. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить рівень її вкладеності. Початковий список може бути, наприклад, такого виду: *[[1, 2, [4, z, 6]], [7, 9], 8]*.
2. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить її абсолютну позицію.
3. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, які розташовані перед нею.
4. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, які розташовані після неї.
5. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, які розташовані у тому ж підсписку, що і літера.
6. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, які розташовані у списках, підлеглих до того, у якому розташована літера.
7. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, рівень вкладеності яких менше, ніж у літери.
8. Написати предикат, який шукає літеру серед цифр у списку з підсписками та виводить суму цифр, рівень вкладеності яких більше, ніж у літери.
9. Підрахувати характеристичне число списку з підсписками, який містить літери та цифри. Характеристичне число нараховується як сума добутків відповідних чисел на рівень вкладеності підсписку, в якому вони розташовані. Рівень вкладеності списку найвищого рівня

прийняти рівним одиниці. Літери ігнорувати. Наприклад: Дано список *[1, [2, a], b, [5, c, 7]]*. Його характеристичне число обраховується так:
$X = (\mathbf{1} \cdot 1) + (\mathbf{2} \cdot 2) + (\mathbf{2} \cdot 5 + \mathbf{2} \cdot 7)$, де жирним шрифтом позначено рівень вкладеності підсписку.

10. Дано список з цифр, що йдуть у довільному порядку. Написати предикат, який повертає суму першої і останньої цифри.

11. Дано список довільної довжини. Написати предикат, який повертає перший і останній елемент цього списку у вигляді двоелементного списку.

12. Дано два списки парної довжини. Написати предикат, який будує з них список такої структури: спочатку два елементи з першого списку, потім два елементи з другого списку, й далі..

13. Дано два списки, причому довжина другого списку вдвічі більше за довжину першого. Написати предикат, який між елементами першого списку вставляє по два елементи з другого. Тобто зі списків виду *[ 1, 2, 3, 4 ]* і *[ a, b, c, d, e, f, g, h ]* будує список виду *[ 1, a, b, 2, c, d, 3, e, f, 4, g, h ]*.

14. Дано список довільної довжини. Написати предикат, який повертає перший, другий, передостанній і останній елементи цього списку у вигляді чотирьохелементного списку.

15. Обернути квадратну матрицю навколо головної діагоналі.

16. Знайти у матриці з натуральних чисел найбільший елемент.

17. Обчислити середнє арифметичне усіх елементів матриці.

18. Бінарне відношення задане своєю матрицею. З’ясувати, чи є воно рефлексивним.

19. Бінарне відношення задане своєю матрицею. З’ясувати, чи є воно симетричним.

20. Бінарне відношення задане своєю матрицею. З'ясувати, чи є воно асиметричним.

21. Бінарне відношення задане своєю матрицею. З'ясувати, чи є воно антисиметричним.

22. У квадратній матриці переставити місцями парні і не парні рядки.

23. У квадратній матриці переставити місцями парні і не парні стовпчики.

24. Перетворити масив довжиною $N^2$ елементів у матрицю з розмірністю $N \times N$ елементів.

25. Перетворити матрицю з розмірністю $N \times N$ елементів у масив довжиною $N^2$ елементів.

26. Побудувати декартів добуток двох множин.

27. Побудувати симетризований декартів добуток двох множин.

28. Дана матриця бінарного відношення. З'ясувати, чи є воно функціональним.

### 4.3 Контрольні питання

1. Що таке рекурсія?
2. Якою може бути рекурсія?
3. Чи можна використовувати рекурсивні процедури для обробки списків у мові Prolog?
4. Чи можна використовувати рекурсивні процедури для обробки структур у мові Prolog?
5. Де треба розташувати рекурсивний виклик у процедурі з прямою рекурсією у мові Prolog?

6. Де треба розташувати рекурсивний виклик у процедурі з хвостовою рекурсією у мові Prolog?
7. Яку мінімальну кількість тверджень треба сформулювати для створення рекурсивної процедури у мові Prolog?
8. Як сформулювати запит до рекурсивної процедури у мові Prolog?
9. Як уникнути зациклювання при створенні рекурсивної процедури у мові Prolog?
10. Як Prolog-система обробляє рекурсивні процедури?

# 5 Практична робота № 5

***Створення простих програм на мові Prolog***

**Мета:** отримати навички створення програм мовою Prolog.

## 5.1 Теоретичні відомості

Prolog-програма може розглядатися з трьох точок зору: *декларативної*, *процедурної* та *поведінкової*. Декларативний сенс полягає у тому, що Prolog-система намагається визначити, чи є ця ціль (мета) істиною (досяжною) і, якщо так, то при яких значеннях змінних вона досягається. Порядок цілей і речень у Prolog-програмі при цьому не є суттєвим.

З процедурної точки зору навпаки, порядок цілей і речень у Prolog-програмі є суттєвим. Від нього залежить ефективність роботи програми. Невдалий порядок може навіть привести до нескінченних рекурсивних викликів проміжних цілей та зациклюванню програми.

З поведінкової точки зору розглядається процес пошуку Prolog-системою відповіді на поставлене питання. Тобто послідовність поставлення та досягнення проміжних цілей, автоматичного повернення та перебору різних варіантів рішень.

## 5.2 Залікові завдання

1. У півфінал вийшли чотири команди: Реал, Динамо, Спартак, Мілан. Перед початком ігор були прогнози 3-ох тренерів:
    1. 1 – Динамо, 2 – Спартак.
    2. 2 – Динамо, 3 – Мілан.
    3. 2 – Реал, 4 – Мілан.

Після ігор виявилося, що кожен фахівець помилився лише в одному припущенні. Як розподілилися місця? Мовою Prolog складіть базу знань і розробіть правила для вирішення задачі.

2. Обід складається з трьох страв: перше, друге і десерт. Страви можуть бути такі: м’ясні, рибні, молочні, овочеві, круп’яні. Як десерт вживають соки, компоти, фрукти. Складіть базу знань про блюда і розробіть правила, складання обіду для людей різних категорій:
    1. Вегетаріанців.
    2. Любителів м’ясної їжі.
    3. Любителів молочної їжі.
    4. Обід заданої калорійності.

| База знань до задачі | |
|---|---|
| course(sausage,meat). | calority(sausage,1). |
| course(cutlet,meat). | calority(cutlet,2). |
| course(ham,meat). | calority(ham,3). |
| course(sour_cream,milk). | calority(sour_cream,4). |
| course(kefir,milk). | calority(kefir,5). |
| course(curd,milk). | calority(curd,6). |
| course(salad,vegetable). | calority(salad,7). |
| course(mashed_potatoes, vegetable). | calority(mashed_potatoes,8). |
| course(soup_vegetable, vegetable). | calority(soup_vegetable,9). |
| course(buckwheat,grain). | calority(buckwheat,10). |

| | |
|---|---|
| course(porridge,grain). | calority(porridge,11). |
| course(lobster,fish). | calority(lobster,12). |
| course(whale,fish). | calority(whale,13). |
| dessert(juice). | calority(juice,14). |
| dessert(fruit). | calority(fruit,15). |
| dessert(stewed_fruit). | calority(stewed_fruit,16). |
| dessert(wine). | calority(wine,17). |
| dessert(icecream). | calority(icecream,18). |

3. Чотири спортсмени (Андрій, Іван, Роман і Сергій) у змаганнях посіли чотири перших місця, причому жодне з місць не було поділене між ними. Один з уболівальників сказав, що друге місце посів Роман, а третє Сергій. Другий стверджував, що переміг Роман, а Іван був другим. Третій сказав, що друге місце посів Андрій, а Сергій останнє. Як розподілились місця між учасниками змагань, якщо відомо, що в кожному з трьох висловлень про зайняті місця одне висловлення істинне, а друге–хибне? Мовою Prolog складіть базу знань і розробіть правила для вирішення задачі.
4. На змаганнях легкоатлетів зі спортивної ходьби Іван відстав від Богдана та ще від двох спортсменів (тобто прийшов четвертим). Віктор фінішував після Дмитра, але раніше Сашка. Дмитро випередив Богдана, але прийшов після Євгена. Яке місце посів кожний спортсмен? Мовою Prolog складіть базу знань і розробіть правила для вирішення задачі.

### 5.3 Контрольні питання

1. У чому полягає декларативний сенс Prolog-програми?
2. У чому полягає процедурний сенс Prolog-програми?
3. У чому полягає поведінковий сенс Prolog-програми?
4. Чи є суттєвим порядок цілей і речень у Prolog-програмі з декларативної точки зору?
5. Чи є суттєвим порядок цілей і речень у Prolog-програмі з процедурної точки зору?
6. Чи є суттєвим порядок цілей і речень у Prolog-програмі з поведінкової точки зору?
7. З якої точки зору треба розглядати Prolog-програму для того, щоб підвищити її ефективність?
8. З якої точки зору треба розглядати Prolog-програму під час аналізу поведінки системи?

# 6 Практична робота № 6

***Робота з операторами***

**Мета:** засвоїти методику роботи з операторами.

## 6.1 Теоретичні відомості

Визначимо логічні оператори: *заперечення (\), кон'юнкції (&), диз'юнкції (□), імплікації (→)* і *еквівалентності (~)*. Значення рівня пріоритету для заперечення приймемо рівним 500, кон'юнкції–600, диз'юнкції–700, імплікації–800, еквівалентності–900. Введемо до бази знань факт *a*. Наявність його в базі означає, що атом *a*, який буде використовуватись у записі формули, має значення «істина». Побудуємо таблицю істинності для операції *кон'юнкція.*

Текст програми:

```
:-op(500, fy, \).
:-op(600, yfx, &).
:-op(700, yfx, v).
:-op(800, xfx, →).
:-op(900, yfx, <=>).
\X:- not X.
X v Y:- X;Y.
X & Y:- X,Y.
X → Y:- \X v Y.
X<=>Y:- (X → Y) & (Y → X).
a.
t2(b,b). t2(b,a). t2(a,b). t2(a,a).
table2 :- t2(X,Y),
((call(X & Y),
write('X= '), write(X), write(', Y= '), write(Y),
write(', Z= 1'),!);
(write('X= '), write(X), write(', Y= '), write(Y),
write(', Z= 0'))),
nl,fail.
```

Приклад запиту:

*JIP:-table2.*
*X= b, Y= b, Z= 0*
*X= b, Y= a, Z= 0*
*X= a, Y= b, Z= 0*
*X= a, Y= a, Z= 1*
*No*

**6.2 Залікові завдання**

1. Помістити в базу факти *x* і *y*. Наявність їх в базі означає, що атоми *x* і *y*, які будуть використовуватися в записі формули, мають значення «істина».
2. Скласти програму інтерпретації довільної формули з двома атомами.
3. Додати операції *штрих Шефера* ($\boxed{x \ \& \ y}$) і *стрілка Пірса* ($\boxed{x \ \square \ y}$) і перевірити їх роботу. Стандартний набір аргументів задати двохарними фактами.
4. Перевірити роботу логічних операторів на формулах: $\boxed{x}$, $x \ \& \ y$, $x \ \square \ y$, $x \to y$, $x \sim y$.
5. Скласти програму інтерпретації довільної формули з трьома атомами. Перевірити її роботу на формулах:
    1. $(x \ \& \ y) \ \square \ z$.
    2. $(x \ \& \ y) \ \square \ (x \ \& \ z)$.
    3. $x \ \& \ y \to z$.
    4. $x \to y \ \square \ z$.
    5. $(x \to y) \sim (x \to z)$.

    Стандартний набір аргументів задати трьохарними фактами.

### 6.3 Контрольні питання

1. Що таке логічні оператори?
2. Якими бувають логічні оператори?
3. Які властивості мають логічні оператори?
4. Які логічні оператори є одноарними?
5. Які логічні оператори є двоарними?
6. Як визначаються логічні оператори у мові Prolog?
7. Як призначити равні приоритетів логічним операторам у мові Prolog?
8. Що таке таблиця істинності?
9. Як побудувати таблицю істинності у мові Prolog?
10. Як згенерувати стандартний набір даних для таблиці істинності у мові Prolog?
11. Як створити запит для генерації таблиці істинності у мові Prolog?
12. Скільки рядків буде мати таблиця істинності для двоарного логічного виразу?
13. Скільки рядків буде мати таблиця істинності для трьохарного логічного виразу?

# 7 Практична робота № 7

***Бінарні дерева***

**Мета:** ознайомитися з методами обробки даних, представлених за допомогою бінарних дерев.

## 7.1 Теоретичні відомості

*Деревом* називається граф, у якого одна вершина коренева, а інші вершини мають тільки одного батька, вони є нащадками кореневої вершини. *Листям* дерева називається його вершина, яка не має нащадків. *Кроною* дерева називається сукупність усіх листів. *Висотою* дерева називається найбільша довжина маршруту від кореня до листа.

Для складання програм зручно використовувати таке рекурсивне визначення бінарного дерева: дерево або порожнє, або складається з кореня, а також лівого й правого піддерев, які у свою чергу також є деревами.

У вершинах дерева може зберігатися інформація будь-якого типу. У межах цієї роботи будемо вважати, що у вершинах дерева розташовуються цілі числа.

Приклад бінарного дерева наведено на цьому рисунку.

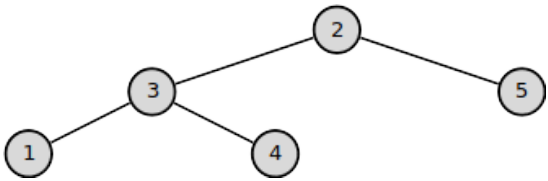


*Рис. 7.1*

Опис цього дерева мовою Prolog може бути таким:

*tree(tr(2,*

```
tr(5, empty, empty),
tr(3,
    tr(4, empty, empty),
    tr(1, empty, empty)))).
```

Ідентифікатор *empty*, який використано у цьому описі, не є зарезервованим словом мови Prolog. Замість нього цілком можна вживати яке-небудь інше позначення для порожнього дерева.

### 7.2 Залікові завдання

Дано червоно-чорне бінарне дерево, кожній вершині якого зіставлене певне натуральне число.

1. Написати предикат, який підраховує кількість червоних вершин.
2. Написати предикат, який підраховує суму чисел у червоних вершинах.
3. Написати предикат, який підраховує суму чисел у вершинах, які є листами.
4. Написати предикат, який підраховує суму чисел у вершинах, які не є листами.
5. Написати предикат, який підраховує кількість вершин, які є листами.
6. Написати предикат, який підраховує кількість вершин, які не є листами.
7. Написати предикат, який підраховує суму парних чисел у червоних вершинах.
8. Написати предикат, який підраховує суму парних чисел у вершинах, які є листами.

9. Написати предикат, який підраховує суму парних чисел у вершинах, які не є листами.

10. Написати предикат, який підраховує кількість вершин, які є листами з парними числами.

11. Написати предикат, який підраховує кількість вершин, які не є листами з парними числами.

12. Написати предикат, який підраховує кількість вершин, на заданому рівні *H*.

13. Написати предикат, який підраховує кількість червоних вершин на заданому рівні *H*.

14. Написати предикат, який підраховує суму чисел у вершинах на заданому рівні *H*.

15. Написати предикат, який підраховує суму чисел у червоних вершинах на заданому рівні *H*.

16. Написати предикат, який підраховує суму парних чисел у червоних вершинах на заданому рівні *H*.

17. Написати предикат, який підраховує кількість листів на заданому рівні *H*.

18. Написати предикат, який підраховує кількість червоних листів на заданому рівні *H*.

19. Написати предикат, який підраховує суму чисел у вершинах, які є листами на заданому рівні *H*.

20. Написати предикат, який підраховує суму чисел у вершинах, які не є листами на заданому рівні *H*.

21. Написати предикат, який підраховує суму чисел у червоних вершинах, які є листами на заданому рівні *H*.

22. Написати предикат, який підраховує суму чисел у червоних вершинах, які не є листами на заданому рівні *H*.

23. Написати предикат, який підраховує суму парних чисел у червоних вершинах, які є листами на заданому рівні *H*.

24. Написати предикат, який підраховує суму парних чисел у червоних вершинах, які не є листами на заданому рівні *H*.

25. Написати предикат, який повертає номер рівня, на якому сума чисел у вершинах є найбільшою.

26. Написати предикат, який повертає номер рівня, на якому сума чисел у червоних вершинах є найбільшою.

27. Написати предикат, який повертає номер рівня, на якому сума парних чисел у червоних вершинах є найбільшою.

### 7.3 Контрольні питання

1. Що таке граф?
2. Якими бувають графи?
3. Що таке деревовидний граф?
4. Що таке вершина дерева?
5. Що таке ребро дерева?
6. Що таке коренева форма дерева?
7. Якими бувають дерева кореневої форми?
8. Що таке листя дерева?
9. Що таке крона дерева?
10. Що таке висота дерева?
11. Що таке бінарне дерево?
12. Як правильно розфарбувати дерево?
13. Як скласти рекурсивну програму мовою Prolog для побудови бінарного дерева?

14. Як скласти рекурсивну програму мовою Prolog для обходу вершин бінарного дерева?
15. Як скласти рекурсивну програму мовою Prolog для перебудови структури бінарного дерева?

# РОЗДІЛ 2
# Самостійні роботи

**1. Вимоги до звіту**

1. Звіт подається в друкованому вигляді, написаний на одній стороні аркуша А4.
2. Звіт повинен мати стандартний титульний аркуш.
3. В основному надписі ставиться підпис студента і дата подання звіту.
4. Звіт повинен включати такі розділи:
    1. умови задачі,
    2. опис алгоритму,
    3. правильно відформатований текст програми з необхідними коментарями,
    4. приклади роботи програми.

**2. Порядок подання звіту**

1. Правильно оформлений звіт подається викладачеві.
2. Неправильно оформлений звіт вважається неподаним.
3. Несвоєчасно поданий звіт вважається неподаним.

**3. Порядок зарахування роботи**

1. Викладач перевіряє роботу.
2. При виявленні помилок, невідповідного виконання, нераціональних рішень робота повертається на доопрацювання.
3. Студент, роботу якого визнано правильно виконаною, допускається до її захисту.

4. Захист починається з виконання тестових прикладів, які надає викладач для перевірки правильності роботи програми. Правильна програма повинна працювати для будь-яких допустимих даних.

**4. Завдання до самостійної роботи № 1**

***Списки***

1. Неорієнтований граф заданий списком суміжних вершин: *graph( [ a-b, a-c, a-d, b-c, b-d, c-d, c-e, d-e ] ).* Скласти програму пошуку списку вершин, що утворюють Ейлерів ланцюг: *eichain(Graph,Chain).*
2. Дерево задається списком суміжних вершин у вигляді *tree( [ a-g, b-g, g-k, k-c, g-f, f-h, h-d, h-e ] ).* Скласти список висячих вершин. Визначити кількість висячих вершин.
3. Скласти програму визначення довжини маршруту між довільними вершинами дерева *tree( [ a-g, b-g, g-k, k-c, g-f, f-h, h-d, h-e ] ).*
4. Сітка доріг є графом з навантаженими ребрами: *map( [ (a-b, 10), (a-d, 10), (a-h, 70), (a-g, 25), (b-c, 10), (c-h, 10), (d-e, 5), (e-f, 15), (f-h, 20), (g-h, 30) ] ).* Цифри – відстань між містами. Скласти програму визначення найкоротшого маршруту між двома заданими містами. Визначити відстань між пунктами *a* і *h*.
5. Є неорієнтований граф *graph( [ a-b, a-d, a-h, a-g, b-c, c-h, d-e, e-f, f-h, g-h ] ).* Знайти всі цикли графа у вигляді списку списків вершин з точністю до початкової вершини і напрямку обходу. Визначити кількість таких циклів.
6. Є неорієнтований граф *graph( [ a-b, a-d, a-h, a-g, b-c, c-h, d-e, e-f, f-h, g-h ] ).* Визначити цикл, що містить найбільше число ребер.

7. Є неорієнтований граф *graph( [ a-c, a-d, b-d, b-e, c-e ] ).* Знайти список вершин, що утворюють Ейлерів цикл.
8. Є неорієнтований граф *graph( [ a-b, a-c, b-f, b-e, c-d ] ).* Скласти список вершин графа з наведенням їх степенів у вигляді *[ (a, 2), (b, 4), ... ]*. Визначити число вершин і число ребер.
9. Є неорієнтований граф *graph( [a-b, a-e, a-d, b-e, b-c, c-g, c-d, d-f, f-e, f-g, g-e ] ).* Довести, що цей граф є Гамільтоновим, склавши список вершин Гамільтонового циклу.
10. Є лабіринт у вигляді неорієнтованого графа: *maze( [ a-enter, a-b, b-c, c-enter, enter-f, enter-e, e-d, e-g, f-gold, gold-g, g-exit ] ).* Знайти найкоротший шлях у вигляді списку вершин від входу (*enter*) до виходу (*exit*), такий, що проходить через вершину, що містить скрб (*gold*).
11. Неорієнтований граф заданий списком вершин і списком ребер *vertecies( [ a, b, c, d, e, f, g, h ] ), edges( [ b-g, b-c, b-e, g-f, f-c, e-d ] ).* Знайти список ізольованих вершин.
12. Є неорієнтований граф *g( V, E ) = g( [ a, b, c, d, e, f, g ], [ a-b, a-c, a-d, e-f ] ).* Знайти зв'язні компоненти у вигляді *Gi(V i,Ei).*
13. На графі заданий циклічний маршрут *Path = [ a-c, c-e, e-d, d-c, c-b, b-a ]*, де *a, b, c, d, e* – вершини графа. Визначити, чи є він простим циклом.
14. Маршрут на графі визначається списком *M = [ a-b, c-e, e-d, d-c, c-b, b-a, a-d ]*, де *a, b, c, d, e* – вершини графа. Визначити, чи є він циклічним.
15. Маршрут на графі задається списком *S = [ a-c, c-e, e-d, d-c, c-b, b-a, a-c ]*, де *a, b, c, d, e* – вершини графа. Визначити, чи є цей маршрут ланцюгом.

16. Є маршрут на графі *L = [ d-c, c-b, b-a, a-c, c-e ], a, b, c, d, e* – вершини графа. Визначити, чи є він простим ланцюгом.
17. Є граф *graph( [ a-b, b-c, b-d, c-d, b-e, e-f, f-g, f-h, g-h, f-c, f-j, f-k, i-j ] )*. Скласти список степенів вершин графа, а також списки вершин з однаковим степенем.
18. Є граф *graph( [ a-b, b-e, e-d, d-a, c-a, c-b, c-d, c-e ] )*. Довести, що він не є Ейлеровим.
19. Є граф *graph( [ a-e, a-c, e-c, e-d, b-e, b-d ] )*. Довести, що він унікурсальний.
20. Є граф *g( V, E), V = [ a, b, c, d, e, f, g ], E = [ (a, e), (a, g), (a, d), (d, e) ]*. Скласти список ізольованих вершин графа.
21. Є дерево *tree( [ a-b, b-c, c-d, c-e, b-f, f-g, f-h, f-i, b-j ] )*. Скласти список висячих вершин.
22. Є граф *graph( [ a-b, a-c, b-c, b-d, b-f, c-e, b-d, d-e, e-f ] )*. Знайти каркас, що не містить ребро *а-в*.
23. Є граф *graph( [ a-b, a-c, c-d, b-d, b-e ] )*. Довести, що він не є Гамільтоновим.
24. Є плоский граф *graph( [ a-b, b-c, c-e, c-d, d-a, d-b, d-c, d-e ] )*. Визначити число граней графа.
25. Є граф *graph( [ a-b, b-c, c-d, d-e, e-f, f-a, h-a, h-d, h-g, g-e])*. Знайти граф, йому гомеоморфний (не містить вершин степеня 2).
26. Є граф *graph( [ a-e, a-b, a-d, b-e, b-d, b-c, c-d, e-d ] )*. Знайти Ейлерів ланцюг, якщо він існує.
27. Є граф *graph( [ a-b, a-c, a-g, b-c, c-d, c-e, d-e, d-f, g-f, g-e ] )*. Розбити граф на дві неперетинні частини, кожна з яких є ланцюгом.
28. Є неорієнтований граф *graph( [ a-b, a-e, a-c, b-c, b-e, b-d, c-d, d-e ] )*. Знайти Гамільтоновий цикл.

29. Є неорієнтований граф *graph( [ a-b, b-c, c-d ] )*. Знайти доповнення графа.
30. Є граф, у якого *V = [ a, b, c, d, e, f, g]* – вершини, а *E = [ b-c, b-g, c-g, c-d, g-e ]* – ребра. Видалити з графа ізольовані і висячі вершини.

## 5. Контрольні питання до самостійної роботи № 1

1. Що таке простий граф?
2. Що таке ланцюг?
3. Що таке орієнтований граф?
4. Що таке неорієнтований граф?
5. Що таке циклічний граф?
6. Що таке Ейлерів граф?
7. Що таке Гамільтонів граф?
8. Що таке суміжні вершини графа?
9. Що таке суміжні ребра графа?
10. Що таке висяча вершина графа?
11. Що таке граф з навантаженими ребрами?
12. Що таке степінь вершини графа?
13. Що таке маршрут на графі?
14. Що таке унікурсальний граф?
15. Що таке каркас графа?

## 6. Завдання до самостійної роботи № 2

***Робота з базою знань***

1. Привілейований користувач–«власник», вводить до бази знань факти «*intvalue(X)*», де *X*–довільне ціле число. Решта користувачів також можуть додавати числа в базу, але числа, які вони можуть вводити, мають бути в межах $X_{min} < X < X_{max}$. Програма при першому запуску знайомиться з «власником», а при наступних входженнях вітає його на ім'я і переходить до стану введення терму. При введенні терму «*end*» відбувається змінення власника.
2. База з фактів «*element(X)*» утворюється за принципом, пов'язаним з номером події введення *X*: перші три факти заносяться безумовно у порядку їх введення. У подальшому номери непарних подій заносяться на початок, парні в кінець бази, причому для подій з номерами кратними 3 факти не створюються. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
3. Головний користувач програми вводить пароль і починає створювати базу з фактів «*object(X)*», де *X* в межах сеансу –дані одного типу. Решта користувачів теж мають змогу додавати факти, але тільки того типу, що увів головний користувач під час останнього сеансу роботи з системою. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
4. Програма керується за допомогою такого меню:
    1. Введення числа.
    2. Введення правила.
    3. Вихід.

Правило визначає числа, що допустимі для введення. Правило можна змінювати під час роботи і воно зберігається до наступного сеансу.

Спочатку допустимими є всі числа. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.

5. Робота програми введення фактів «*number(X)*» не залежить від сеансу і керується за допомогою такого меню:
    1. Введення числа–початок.
    2. Введення числа–закінчення.
    3. Перегляд (для привілейованого користувача).
    4. Вихід.

    Перегляд–видає значення *X* за порядком розміщення фактів в базі. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
6. Програма використовується для створення і оновлення файлу фактів виду «*number(X)*», в якому числа йдуть в порядку зростання. Надходити вони можуть в довільному порядку і в різних сеансах роботи. Дублювання фактів не допустимо. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
7. Програма веде англо-український і українсько-англійський словники і за введенням українського або англійського слова видає його переклад. Якщо такого слова немає, запитує відповідний переклад і доповнює словник. Вважати, що слова вводяться без помилок. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
8. Програма дозволяє вводити англійські слова і зберігати їх у базі за абеткою. На запит генерує текстовий файл зі слів. Введення слів здійснюється за декілька сеансів. Робота програми не залежить від

входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.

9. Перший гравець вводить в базу 5 чисел першого десятка, другий, намагаючись вгадати, що увів перший, вводить свої 5 і виграє хід, якщо вгадає більше половини чисел. У випадку виграшу ходу, другий повторює спробу. Якщо другий не набув права ходу, перший робить спробу вгадати, що увів другий. І так доти, доки хтось не вгадає всю послідовність. Вгадані числа виводяться на екран. Робота програми не залежить від входів та виходів з системи і завжди може бути подовжена з того місця, де її робота була зупинена.
10. Дитяча гра. Програма формує факти «*sentence(N, X)*», де *N*-номер речення, *X*-рядки, що дають відповіді на питання «*хто?*», «*з ким?*», «*де?*», «*коли?*», «*що робили?*». Всі гравці, а їх може бути скільки завгодно, але не менше двох, по черзі відповідають на одне й те ж питання так, що ніхто не знає відповідей інших гравців. При цьому кожна наступна відповідь додається у інше речення, що визначається циклічним зсувом. Після закінчення введення видаються всі складені таким чином речення. Число речень дорівнює числу гравців. Робота програми не залежить від входів та виходів з системи і завжди може бути подовжена з того місця, де її робота була зупиненна.
11. Гра «хрестики-нулики» з можливістю переривання і поновлення гри з місця зупинки.
12. Предметна область складається зі столу і чотирьох кубиків *a, b, c, d*. Взаємне положення об’єктів характеризується предикатом «*on(x, y)*», де *x, y* – різні об’єкти. Скласти програму, яка зманює стан світу кубиків. Вважати, що в початковий момент всі кубики знаходяться на столі. Стан світу зберігається при виході з програми.

13. Запрограмувати автомат з пам’яттю $y = f(x,y,s_{[i-1]})$, де

    $s_{[i-1]} = (x_{[i-1]},y_{[i-1]})$,

    $x_{[0]} = 0$, $y_{[0]} = 0$,

    $i$ – номер кроку автомату.

    $z = x_{[i-1]}$ □ $x$ & $y$,

    $w = y_{[i-1]}$ & $y$ □ $x$.

    Під час виходу з системи стан запам’ятовується.

| **z** | **w** | дія |
|---|---|---|
| 1 | 1 | a |
| 1 | 0 | b |
| 0 | 1 | c |
| 0 | 0 | d |

14. Запрограмувати автомат з 3-ома перемикачами *x*, *y*, *z*, які можуть знаходитися в одному з двох станів. Дії автомату описані таблицею. За один раз змінюється одне значення і виконується одна дія. Наприкінці сеансу стан автомату зберігається. Дії вибрати самостійно. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.

| **x** | **y** | **z** | дія |
|---|---|---|---|
| 1 | 1 | 1 | a |
| 1 | 1 | 0 | b |
| 1 | 0 | 1 | c |
| 1 | 0 | 0 | d |
| 0 | 1 | 1 | e |

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | f |
| 0 | 0 | 1 | g |
| 0 | 0 | 0 | h |

15. Шифр замка заданий чотирма цифрами. Після 5-ти невдалих спроб відкрити замок лунає сигнал. Спроби можуть бути розподілені у часі. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
16. На складі спочатку є *M* виробів «а», *N* виробів «б», *K* виробів «в». Вироби постійно забираються зі складу і постачаються на склад. Скласти програму ведення складу з видачею кількості кожного виробу. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
17. Створити автомат $f(z,w)$, де $z$ береться з черги, а $w$ береться зі стеку. Якщо черга і стек порожні, ніяких дій не виконувати. Черга і стек можуть бути поповнені в будь-який час. Стан автомату зберігається при завершені сеансу роботи. Дії вибрати самостійно. Робота програми не залежить від входів та виходів з системи і завжди може бути подовжена з того місця, де її робота була зупинена.

| **z** | **w** | дія |
|---|---|---|
| 1 | 1 | a |
| 1 | 0 | b |
| 0 | 1 | c |
| 0 | 0 | d |

18. Виріб складається з чотирьох складових частин «*a*», «*b*», «*c*», «*d*», які можуть надходити у цех у довільному порядку по одному. За

наявності всіх потрібних частин виріб збирається і відправляється на склад. Скласти програму ведення роботи цеху і складу. Робота цеху може призупинятися (вихід з системи) і поновлюватися. Робота програми не залежить від входів та виходів з системи і завжди може бути подовжена з того місця, де її робота була зупинена.

19. Вироби складаються з деталей «*a*», «*b*», «*c*», «*d*»:
    1. Виріб 1 з «*a*», «*b*», «*c*».
    2. Виріб 2 з «*b*», «*c*», «*d*».

    Скласти програму комплектації виробів, яка б визначала число скомплектованих виробів. Вироби мають комплектуватися рівномірно, деталі надходять одна за одною. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
20. Гроші за куплені товари можуть надходити до каси крамниці купюрами і дрібними і видаватися з каси у вигляді решти. Розробити програму «Каса», що приймає гроші купюрами і дрібними і видає решту, слідкує за правильністю суми, що надходить до каси. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
21. Двоє гравців по черзі вибирають крамниці, причому беруть або з однієї купки, або однакову кількість з обох одночасно. Виграє той, хто візьме останній камінець. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.
22. З купи, яку утворюють *N* сірників кожен з двох гравців бере будь-яке число сірників, не більше 3-ох. Програє той, хто візьме останній. Створити програму, що моделює гру. Гра може мати перерви. Робота

програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.

23. Розробити програму, що моделює додавання на рахівниці. Сума повинна зберігатися на рахівниці постійно, незалежно від сеансу роботи, до моменту скидання рахівниці. Передбачити зчитування числового результату. Робота програми не залежить від входів та виходів з системи і завжди може бути продовжена з того місця, де її робота була зупинена.

**7. Контрольні питання до самостійної роботи № 2**

1. Що таке база знань Prolog системи?
2. Який порядок роботи з базою знань у Prolog системі?
3. Що таке клауза у мові Prolog?
4. Як додати клаузу до бази знань у Prolog системі?
5. Як видалити клаузу з бази знань у Prolog системі?
6. У чому полягає суть процесу консультування бази знань у Prolog системі?

# СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ


1. Бойко В., Василенко М., Рачук В., Слатвінська В. Нове (друге) народження мови «Пролог» (Prolog) в контексті систем підтримки прийняття рішень. *Інформаційні технології та суспільство.* 2022. № 1 (3). С. 16–22. DOI: 10.32689/maup.it.2022.1.2.
2. Любченко К. М. Мова програмування Prolog. Базовий курс : навч.-метод. посіб. Черкаси : ЧНУ імені Богдана Хмельницького, 2016. 136 с.
3. Магола Д. Логическое программирование в среде Visual Prolog: Компьютерный практикум. Palmarium Academic Publishing, 2014. 136 с.
4. Марков В. М. Сучасне логічне програмування на мові Visual Prolog 7.5. БХВ-Петербург, 2016. 544 с.
5. Шумейко О. О., Кнуренко В. М. Visual Prolog. Опануй на прикладах : навч. посіб. / Дніпропетровськ : Біла К. О., 2014. 404 с.
6. Ющенко Ю. О. Логічне програмування – Prolog. НаУКМАб, 2022. URL: https://www.youtube.com/watch?v=n4pXLejbl5U (дата звернення 12.08.2022)
7. Boizumault P. The implementation of Prolog. Princeton University Press, 2014. 314 p.
8. Bratko I. Prolog programming for artificial intelligence. Addison-4th ed. Wesley, 2011. 673 p.
9. Chen Yi. Introduction to programming languages: Programming in C, C++, Scheme, Prolog, C#, and Python. Kendall Hunt Publishing Company, 2020. 529 p.
10. Dawe M.S., Dawe C. M. PROLOG for computer science. Springer Science & Business Media, 2012. 189 p.

11. Johansson A.-L., Eriksson-Granskog A., Edman A. Prolog versus you: An introduction to logic programming. Springer-Verlag, 2012. 293 p.

12. Key S. Prolog programming success in a day: Beginners guide to fast, easy and efficient learning of Prolog programming. CreateSpace Independent Publishing Platform, 2015. 60 p.

13. Mellish C. S. Programming in Prolog: Using the ISO Standard. 5th ed. Springer, 2003. 295 p.

14. Merritt D. Expert systems in Prolog. Springer Independently Published, 2017. 239 p.

15. Shoham Yo. Artificial intelligence techniques in Prolog. Morgan Kaufmann Publ., 2014. 315 p.

# ДЛЯ НОТАТОК

**Старченко**
**В'ячеслав Володимирович**

# ЛОГІЧНЕ ПРОГРАМУВАННЯ (PROLOG)

*Методичні вказівки*

**Випуск 414**

---

Редактор *А. Бурмус*
Технічний редактор, комп'ютерна верстка *К. Гросу-Грабарчук*
Друк *С. Волинець*. Фальцовально-палітурні роботи *О. Мішалкіна*.

Підп. до друку 17.04.2023
Формат 60х84 1/16. Папір офсет.
Гарнітура «Times New Roman». Друк ризограф.
Ум. друк. арк. 3,02. Обл.-вид. арк.1,29.
Тираж 5 пр. Зам. № .6598

54003, м. Миколаїв, вул. 68 Десантників, 10.
Тел.: 8 (0512) 50-03-32, 8 (0512) 76-55-81, e-mail: rector@chmnu.edu.ua.
Свідоцтво суб'єкта видавничої справи ДК № 36124 від 05.04.2018.