МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до самостійних робіт з дисципліни

“Функціональне та логічне програмування”

для студентів професійного напряму
6.050103 **«Програмна інженерія»**
спеціальності 7.05010301 **«Програмне забезпечення систем”**

всіх форм навчання

2016

Методичні вказівки до самостійних робіт з дисципліни “Функціональне та логічне програмування” для студентів професійного напряму 6.050103 «Програмна інженерія» спеціальності 7.05010301 «Програмне забезпечення систем” всіх форм навчання /Укл.: І.В. Левада, Л.Ю. Дейнега. – Запоріжжя: ЗНТУ, 2016. –54 с.

Укладачі: ст. викладач І.В. Левада
ст. викладач Л.Ю. Дейнега

Рецензент: доцент, к.т.н. А.В. Пархоменко

Відповідальний за випуск: ст. викладач І.В. Левада

Затверджено
на
засіданні
кафедри
Протокол №
від

22 Утворити бінарне дерево, вузли якого містять відомості про спеціальні символи мови програмування і їх призначення у вигляді структури: symb(“спец. символ”, “призначення”). Дані вводити з клавіатури. Одержане дерево вивести на екран.

**23 Дан файл, компонентами якого є рядки, які містять слова і відповідну частину мови. Утворити інший файл, відсортований за допомогою дерева.

**24 Будемо називати гіперструктурою тексту структуру, в якій кожна компонента має декілька рядків тексту і посилання на інші компоненти. Такі гіперструктури показують логічний зв’язок тексту.

Задайте гіперструктуру тексту у вигляді дерева і виведіть на екран тексти гіперструктури, що логічно пов’язані між собою для обраного з клавіатури вузла.

25 Утворити бінарне дерево, вузли якого містять відомості про предикати Visual Prolog і їх призначення у вигляді структури: pred(“предикат”, “призначення”). Дані вводити з файлу. Одержане дерево вивести на екран.

4. ЛІТЕРАТУРА

1 Адаменко А.Н., Кучуков А.М. Логическое программирование и Visual Prolog. – СПб.: БХВ-Петербург, 2003. – 992 с.

2 Ин Ц., Соломон Д. Использование Турбо-Пролога. – М.: Мир, 1993 ,-605с.

3 Доорс Дж. и др. Пролог– язык программирования будущего. – М.: Финансы и статистика, 1990,-141с..

4 Клоксин У., Меллиш К. Программирование на языке Пролог. – М.: Мир, 1987, -336с.

ЗМІСТ

14 Дан список відомостей про лікарські препарати, які мають таку структуру:[[ціна , назва], …, [ціна, назва]].

Відсортувати список за назвою за допомогою бінарного дерева. Одержане дерево вивести на екран.

15 Дано бінарне дерево, яке містить у вузлах прізвища хворих і їх діагнози у вигляді списку: [прізвище, [діагноз1,... ,діагнозN]]. Ввести с клавіатури прізвища хворих і їх діагнози. Поповнити дерево новими даними. Одержане дерево вивести на екран.

16 Дано бінарне дерево, яке містить у вузлах відомості про потреби покупців деталей у вигляді структури: det(“покупець”, “назва деталі”). Вивести на екран в відсортованому вигляді відомості про покупців деталей, відібраних за введеним з клавіатури покупцем.

17 Дано бінарне дерево вузли, якого містять відомості про назви міст і кількість населення в цих містах у вигляді [“назва міста”, “кількість населення”]. Поповнити відомості про міста з фактів динамічної бази даних. Одержане дерево вивести на екран.

18 Дано бінарне дерево, вузли якого містять відомості про відстань між містами країни у вигляді karta („місто1”, „місто2”, відстань). Визначити за картою будь-яку відстань між двома містами і вивести шлях і відстань на екран.

19 Дано бінарне дерево, вузли якого містять відомості про колір графічного елементу і значення кольору, як ціле число у вигляді структур: col (“колір”, ціле). Написати програму, що знаходить за кольором значення кольору як ціле число; поповнює дані про кольори. Вводити дані з клавіатури. Поповнене дерево вивести на екран.

20 Дано бінарне дерево, вузли якого містять відомості про діагнози і список препаратів, що застосовують при вказаному захворюванні. Вилучити з списків препаратів препарат введений з клавіатури. Одержане дерево вивести на екран.

21 Утворити бінарне дерево, вузли якого містять відомості про стандартні модулі мови програмування у вигляді структури: mod (“назва модуля”, “платформа”). Дані вводити з файлу. Одержане дерево вивести на екран.

5 Дано бінарне дерево вузли, якого містять відомості про шифри і назви деталей на складі у вигляді [“шифр”, “назва деталі”]. Поповнити відомості про деталі з фактів динамічної бази даних. Одержане дерево вивести на екран.

6 Дано бінарне дерево, вузли якого містять відомості про шифри книг бібліотеки. Поповнити шифри книг з файлу. Одержане дерево вивести на екран.

7 Дано бінарне дерево, вузли якого містять відомості про робітників виробничої дільниці у вигляді структур: rab(“прізвище”, “таб. №”). Вилучити з дерева відомості про двох робітників, що звільнилися. Прізвище і таб. № ввести с клавіатури. Одержане дерево вивести на екран.

8 Дано бінарне дерево, вузли якого містять відомості про асортимент продуктового магазину, назви продуктів. Вилучити з асортименту назви продуктів введених з клавіатури. Одержане дерево вивести на екран.

9 Утворити бінарне дерево, вузли якого містять відомості про старост груп спеціальності у вигляді структури: star (“прізвище”, “група”). Дані вводити з файлу. Одержане дерево вивести на екран.

10 Утворити бінарне дерево, вузли якого містять відомості про середній бал студентів групи у вигляді структури: stud (“прізвище”, “бал”). Дані вводити з клавіатури. Одержане дерево вивести на екран.

**11 Дан файл, компонентами якого є рядки, які містять слова і їх тлумачення. Утворити інший файл, відсортований за допомогою дерева.

**12 Будемо називати гіперструктурою тексту структуру, в якій кожна компонента має декілька рядків і посилання на інші компоненти. Такі гіперструктури показують логічний зв’язок тексту.

Утворіть гіперструктуру тексту, що вводиться з клавіатури і містить спеціальні терміни і їх визначення у вигляді дерева. Одержане дерево вивести на екран.

13 Дан файл, який містить прізвища хворих на дільниці поліклініки. Відсортувати прізвища хворих за допомогою бінарного дерева. Одержане бінарне дерево вивести на екран.

ПЕРЕДМОВА

Мета самостійної роботи: Самостійна робота під керівництво викладача цілеспрямована на **розвиток творчих здібностей і активну самостійну розумову діяльність** студентів. Вона розширює можливості навчального процесу. В самостійній роботі вивчаються теоретичні відомості з мови Visual Prolog, які не були розглянуті на лекціях і не були використані в лабораторних роботах.

Вміст: Методичні вказівки з дисципліни "Логічне програмування" містять стислі теоретичні відомості з мови Visual Prolog з нерозглянутих тем, методичні вказівки до вивчення матеріалу тем. Методичні вказівки також містять: контрольні питання для перевірки засвоєння теоретичного матеріалу, індивідуальні завдання для закріплення вивченого, перелік літератури, яку треба використати під час вивчення матеріалу.

Загальні вказівки: Самостійна робота студентів, яка керується, здійснюється у формі систематичних самостійних занять над теоретичним матеріалом який подано у методичних вказівках і літературі, що рекомендована. В ході вивчення матеріалу треба спробувати приклади, що подані в матеріалі, в роботі. Відразу після вивчення теми студенти повинні відповісти на контрольні питання і закріпити одержані знання під час виконання індивідуальних завдань.

№ завдань співпадають з номером прізвища студента у журналі.

Студент зобов’язаний не менше одного разу на місяць дати можливість викладачу ознайомитися з ходом самостійної роботи(вивчення теоретичного матеріалу і виконання завдань).

Студент може одержати консультацію викладача у разі необхідності.

В кінці семестру викладач виконує остаточний контроль та **оцінювання результатів самостійної роботи**.

1. АРИФМЕТИКА ТА ПОРІВНЯННЯ

Мета:

- Одержати знання про арифметичні операції, операції порівняння, стандартні функції і предикати, бінарні предикати Visual Prolog;
- Одержати знання про компоненти арифметичного виразу і правила обчислення його значення у Visual Prolog;
- Навчитися писати и налагоджувати програми на Visual Prolog, які містять арифметичні вирази.

1.1.Теоретичні відомості

1.1.1. Арифметичний вираз

Арифметичний вираз – це компактний запис обчислювального алгоритму. Компонентами арифметичного виразу можуть бути операції, операнди і дужки. Окремим випадком арифметичного виразу є число або конкретизована змінна.

Операндами у виразу можуть бути числа, символи, змінні, що конкретизовані числами чи кодами символів, а також стандартні функції. Числа можуть записуватися у системах числення з основами 10, 8, 16.

Операндами можуть також бути змінні типів користувача, які базуються на стандартних типах.

Операції, що використовуються у виразу, це:

+ , - , / , * , mod , цілочислове ділення div , логічна операція not , предикати порівняння: <, >, <=, >=, =, <>, ><.

Операція mod знаходить залишок від ділення операнду1 на операнд2, де обидва операнди цілі числа.

Обчислення значення арифметичного виразу відбувається при використанні операції порівняння. Пролог не дозволяє використовувати арифметичний вираз без операції порівняння. Тому не можна записувати на місці аргументу предикату арифметичний вираз.

3.3.3 Результати роботи програми. Вказати за яким принципом організовано дерево.

3.4.Контрольні питання

3.4.1 Яким чином можна використовувати бінарні рекурсивні дерева для швидкого пошуку даного? Яка в цьому випадку буде структура дерева?

3.4.2 Для чого ще можна використати тип складного об’єкту – дерево?

3.4.3 Які ви знаєте способи обходу дерева?

3.5. Література

/1. с.331 – 344; 3.с.52-57/

3.6.Індивідуальні завдання

1 Дан файл, який містить прізвища учнів класу. Відсортувати прізвища учнів за допомогою бінарного дерева. Одержане бінарне дерево вивести на екран.

2 Дан список відомостей про робітників цеху, які мають таку структуру:[[таб. № , прізвище], …, [таб. №, прізвище]].

Відсортувати список за таб. № за допомогою бінарного дерева. Одержане дерево вивести на екран. Таб. № розглядати як рядок.

3 Дано бінарне дерево, яке містить у вузлах прізвища авторів і назви книг, які є в бібліотеці у вигляді списку: [автор, назва книги]. Ввести с клавіатури прізвища авторів для нових книг і їх назви. Поповнити дерево новими даними згідно вище вказаному принципу організації дерева. Одержане дерево вивести на екран.

4 Дано бінарне дерево, яке містить у вузлах відомості про постачальників деталей у вигляді структури: det(“постачальник”, “назва деталі”). Вивести на екран назви деталей, відібраних за введеним з клавіатури постачальником.

Процедура do починає працювати з порожнім деревом. Вона вводить число і передає його і дерево в предикат form.

Для кожного нового числа предикат form будує дерево з числом у вузлі і порожніми гілками. Після чого шукає місце, в яке вставляти побудоване дерево в дерево, яке одержано від процедури do. Одержане нове дерево знову рекурсивно оброблюється.

Робота процедури do закінчується за введеним 0. Побудоване дерево виводиться на екран. При утворенні дерева виконується сортування значень вузлів.

3.1.5. Сортування рекурсивних дерев

Не відсортоване дерево, утворене довільно, можна відсортувати використовуючи алгоритм прямого обходу дерева.

При обході дерева виконують такі дії:

- Для порожнього дерева нічого не робити;
- Інакше: переставити у вірному порядку всі елементи лівого піддерева, потім розташувати елемент вузла, і переставити у вірному порядку елементи правого піддерева.

3.2.Методичні вказівки до виконання самостійної роботи

3.2.1 У завданнях, в яких дані дерева, використовуйте дерево-константу впорядковану за обраним вами принципом.

3.2.2 У завданнях, в яких результатом роботи є дерево, вузли дерева впорядковуйте за принципом обраним вами.

3.2.3 У завданнях, які змінюють дане дерево, порядок розташування елементів повинен зберігатися.

3.3.Зміст звіту

Звіт повинен містити наступні пункти:

3.3.1 Ціль роботи.
3.3.2 Завдання і текст програми.

Розглянемо приклади виразів і їх обчислення.

Приклад1. X= 3.6 / (sin (Z) + Y)

Змінній X привласнюється значення виразу, що розташовано праворуч від предикату порівняння „=”. Для обчислення виразу змінні Y та Z повинні бути конкретизовані числами. Функція sin(Z) повертає свій результат на місце її виклику.

Приклад2. X= 0xB + 7 + 0o11

Результат: X= 27

В прикладі використовуються константи в системах числення з основою 16(0xB) і основою 8(0o11).

Приклад3. Знаходиться залишок від ділення 14 на 3:

X= 14 mod 3
X= 2
Перевірка: 14=4*3+2

Приклад4: Виконується ділення одного цілого(17) на друге ціле(2). Результат ціле число(8).

X= 17 div 2
X= 8

1.1.2. Типи операндів і результатів арифметичних операцій

Типи даних можна змішувати в одному виразі за правилами з таблиці 1.1.

Таблиця 1.1 — Типи даних в арифметичних операціях

Операції	Операнд1	Операнд2	Результат
+, - ,*, mod, div	Ціле	Ціле	Ціле
/	Ціле	Ціле	Дійсне або ціле
+,-,*,/	Ціле	Дійсне	Дійсне
+,-,*,/	Дійсне	Ціле	Дійсне
+,-,*,/	Дійсне	Дійсне	Дійсне
<, >, <=, >=, =, <>, ><	Значення арифметичних виразів Символи Рядки	Значення арифметичних виразів Символи Рядки	True або Fail

Правило 1: У виразах можна використовувати операнди сумісних типів.

Сумісні типи – це типи, які Пролог автоматично перетворює, одне в одне для виконання операцій.

Сумісні типи це:

- string і symbol;
- real і integer;
- integer і char.

До сумісних типів відносяться також типи користувача і їх базові типи. Наприклад, сумісними будуть типи integer і nomer, data і integer, якщо типи nomer і data об'явлено:

Domains

nomer = integer

data = integer

Проте типи data і nomer не сумісні.

При виконанні дій Пролог перетворює дане типу char в код (тип integer):

X ='a'+1=97+1. Результат X=98.

Правило2: При обчисленні значення виразу, що містить числа різних типів, результат буде мати автоматично тип операнду, який ширший.

Правило3: При виконанні дій над числами без знакових типів і числами знакових типів результат буде автоматично мати знаковий тип.

Наприклад, тип результату операції над числами типу ushort і long буде long.

Vuvid (X, Y, Z):- X= c(C), write(C), obxid (Z).

Вивід: 26*2 + 36*4

Процедура виконує рух по лівій гілці дерева. Граничною умовою є піддерево, де вузол число, а піддерева порожні.

По досягненню граничної умови, значення вузлів цього піддерева виводяться у вигляді: число операція число.

При поверненні з рекурсії процедура переміщується на попередній вузол, який розташовано вище по дереву. Виводиться значення цього вузла – операція і виконується перехід на праву гілку цього вузла. Права гілка оброблюється аналогічно лівій.

3.1.4. Створення дерева

Побудуємо дерево дійсних чисел впорядкованих таким чином, що числа лівої гілки завжди менші за число вузла, а числа правої гілки завжди більше число вузла:

```
Domains
Tr = tree(real, tr, tr);
     empty

Predicates
Do (tr, tr)
Form (real, tr, tr)

goal
clearwindow, do(empty, Res), write(Res).

clauses

do (D, Res):- readreal (T), T<>0, !, form (T, D, DT), do (DT, Res).
do (D, D).

form (T, empty, tree(T, empty, empty)):- !.

Form (T, tree(El, L, R), tree(El, Nl, R)):-T<El, !, form(T, L, Nl).
Form (T, tree(El, L, R), tree(El, L, Nr)):-!, form(T, R, Nr).
```

Розглянемо рекурсивне дерево у вузлах якого розташовані цілі числа або арифметичні операції так, що при зворотному обході дерева буде надруковано вираз: 26*2 + 36*4.

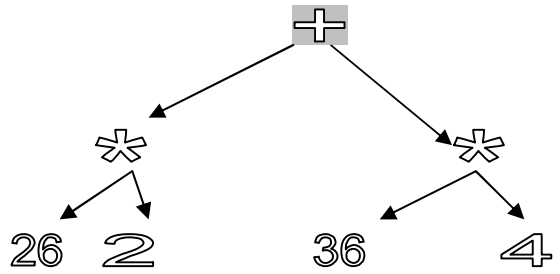

Рисунок 3.2 Дерево для друку виразу

Далі подано приклад програми, яка реалізує зворотний обхід дерева для друку виразу, що розташовано на рис.3.2.

```
Domains
el = i(integer);
     c(char)
tr = tree(el, tr, tr);
     empty

Predicates
obxid(tr)
vuvid(el, tr, tr)
c(char)
i(integer)

goal
D=tree('+',tree('*',tree(26,empty,empty),tree(2,empty,empty)),
tree('*',tree(36, empty, empty),tree(4,empty, empty))),  obxid(D).
clauses
obxid(tree(i(N), empty, empty).                  % гранична умова
obxid (tree (X, Y, Z))  :- obxid(Y), vuvid(X, Y, Z).

Vuvid(X,Y,Z):- X= c(C),Y= tree(N1,_,_), N1= i(K1), Z= tree(N2,_,_),
                    N2= i(K2), write(K1,C,K2).
```


1.1.3. Властивості операцій

Щоб обчислити значення арифметичного виразу, необхідно знати порядок виконання операцій, для цього використовуються наступні властивості операцій:

- позиція
- пріоритет
- асоціативність

Позиція вказує, де стоїть операція стосовно своїх операндів, а також яка кількість операндів у операції. Позиція буває:

- інфіксна – операція стоїть між операндами X + У;
- префіксна – операція стоїть перед операндом -3;
- постфіксна – операція стоїть після операнду 3!.

Інфіксна позиція можлива для двох операндів. **Префіксна** або **постфіксна** позиції можливі для одного операнду.

Компілятор використовує властивість „позиція" при аналізі виразу і виявленні операндів і операцій.

Пріоритет вказує, коли дана операція повинна бути виконана по відношенню до інших операцій виразу. Операція з вищим пріоритетом виконується раніше. Пріоритет у програмах задається числом. Більш високому пріоритету ставиться в відповідність менше число.

Таблиця 1.2 показує відносні пріоритети операцій, предикатів та дужок та їх асоціативність(про асоціативність див. далі).

Таблиця 1.2 — Пріоритети операцій

Операції, предикати, дужки	Пріоритет	Асоціативність
Дужки	-1	Не визначена
Префіксні операції: +,-	1	Праворуч асоціативні
Інфіксні операції: *, / , предикати div, mod	2	Ліворуч асоціативні
Інфіксні операції: +,-	3	Ліворуч асоціативні
Предикати порівняння <,>,<=,>=,=,<>,><	4	Не визначена
Предикат not	5	Праворуч асоціативний

Пріоритети задаються відносно. Це означає, що важно не число, яке використовує певний компілятор як номер пріоритету операції, а відношення між числами - номерами пріоритетів операцій.

Наприклад, номер пріоритету для інфіксної операції „+”, повинен бути більше номеру пріоритету інфіксної операції „*”.

Розглянемо порядок виконання операцій у прикладі.

Приклад1: X= 2 + 4*3.

Операція ‘*’ має більший пріоритет, ніж пріоритет інфіксної операції ‘+’.

Тому обчислюємо:

Перша операція: 4*3=12. Друга операція: 2+12=14.

Приклад2: X= 4*(51-12)

Як видно з таблиці операція в дужках виконується першою.

Тому обчислюємо:

X= 4*39

X= 156

Операції, що мають однаковий пріоритет, утворюють клас одного пріоритету. Операції класу характеризуються певним числом.

Асоціативність вказує, яка операція виразу виконається раніше, при умові, що операції відносяться до одного класу.

Операції бувають ліворуч асоціативні і праворуч асоціативні.

Ліворуч асоціативні операції виконуються зліва направо. Праворуч асоціативні операції виконуються справа наліво.

У таблиці2 вказано асоціативність операцій.

Операція ділення ‘/’ відноситься до ліворуч асоціативних операцій. Тому вираз з прикладу X = 12 /6 / 2 обчислюємо в такий спосіб:

X = (12 / 6) / 2

X=2/2

X=1

До праворуч асоціативних операцій можна віднести операцію „зведення в ступінь”. Обчислюємо вираз:

$_2 3^2 = {}_2 9 = {}_{512}$

Розглянемо приклад програми, яка реалізує прямий обхід дерева з дійсних чисел, що розташовано на рис.3.1.

```
domains
tr=tree(real, tr, tr);
   empty

Predicates
obxid(tr)

goal
D = tree(2.5,tree(13.4,tree(26.8,empty, empty),tree(43.4,empty, empty)),
              tree(47.1,tree(50.3, empty, empty),tree(61.1,empty, empty))),
 obxid(D).

clauses
obxid(empty).                    % гранична умова
obxid (tree (X, Y, Z))  :- write(X), nl, obxid(Y), obxid(Z).
```

Вивід: 2.5
13.4
26.8
43.4
47.1
50.3
61.1

Зворотний обхід рекурсивного дерева

Принцип організації дерева (LTR):

1) Початковою точкою відліку є нижній вузол самого лівого піддерева.
2) Значення лівого піддерева передують значенню поточного вузла.
3) Значення правого піддерева прямують за значенням поточного вузла.

tree(4.1, empty, empty)
tree(5.3, empty, empty)

3.1.3. Обхід рекурсивного дерева

Програми, які використовують в роботі дерева, застосовують процедуру „обхід дерева”. Обходити дерево можна різними способами. Для ідентифікації способу обходу позначимо вузол – T, ліве піддерево – L, праве піддерево R.

Прямий обхід рекурсивного дерева

Принцип організації прямого обходу рекурсивного дерева (T L R):

1) Початковою точкою обробки є вузол.
2) Після чого обробити ліве піддерево, потім обробити праве піддерево.

Вказані дії рекурсивно повторюються спочатку для лівого піддерева , поки піддерево не стане порожнім. Потім аналогічно для правого піддерева.

На рисунку 3.1 показано рекурсивне дерево, у вузлах якого розташовані дійсні числа так, щоб при прямому обході дерева одержані числа були впорядковані за зростанням.

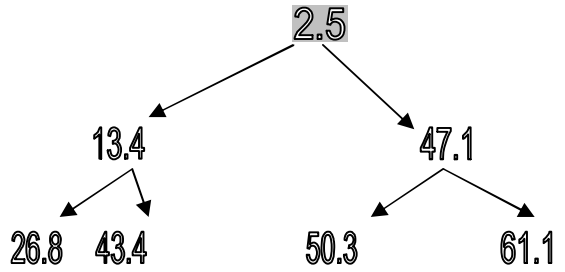

Рисунок 3.1 Рекурсивне дерево дійсних чисел

Існують операції, які не відносять ні до ліворуч асоціативних, ні до праворуч асоціативних. Для таких операцій все одно в якому порядку їх виконувати. До таких операцій відносять операції порівняння: =, <>,<=,>=,<,>. Наприклад: X<Y<Z.

У компіляторі та виконуючої системі Visual Prolog для кожної операції зазначена її асоціативність.

1.1.4. Стандартні функції і предикати

У виразах можна використовувати функції, як операнди.
В Пролозі існують наступні стандартні функції:

sin(X) – обчислює і повертає синус X.

Приклад1: do:-Y1= sin(3.141592653),
Y= round (Y1).
Відповідь: Y1= 0.00000000058979333121, Y= 0

Приклад2: do:- Y= 0, Y= sin (3.141592653). Предикат do повертає fail. Цей приклад має місце і для інших стандартних функцій. Якщо змінну зліва від „=” конкретизовано і її значення не дорівнює результату, що повертає функція.

cos(X) – обчислює і повертає косинус X.
Приклад: do:-Y= cos(3.141592653).
Відповідь: Y= -1

tan(X) – обчислює і повертає тангенс X.
Приклад: do:-Y1= tan(3.141592653),
Y= round (Y1).
Відповідь: Y1= 0.00000000058979333121, Y= 0

Аргументи вказаних тригонометричних функцій – це кути в радіанах. Для перетворення радіани в кути треба виконати дії:

Gradus = Radians*180/3.141592653

arctan(X) – обчислює і повертає арктангенс X в радіанах.
Приклад: do:-Y= arctan(1).
Відповідь: Y=0.7853981634

exp(X) – обчислює і повертає e^x .
Приклад: do:- Y=exp(1).
Відповідь: Y=2.7182818285

ln(X) – обчислює і повертає логарифм натуральний X.
Приклад: do:- Y= ln(2.7182818285).
Відповідь: Y= 1

log(X) – обчислює і повертає логарифм десятковий X.
Приклад: do:- Y= log(1000).
Відповідь: Y=3

round(X) – округляє дійсне число X до найближчого цілого і повертає дійсне число.

Приклад1: do:- Y= round(3.781).
Відповідь: Y=4.0

Приклад2: do:- Y= round(6.129).
Відповідь: Y=6.0

Приклад3: do:- Y=round(-2.189).
Відповідь: Y= -2.0

trunc(X) – відкидає дробову частину X, повертає дійсне число.

Приклад: do:- Y= trunc(3.781).
Відповідь: Y=3.0

Приклад: do:-Y= trunc(-2.189).
Відповідь: Y= -2.0

sqrt(X) – обчислює і повертає квадратний корінь із X;
Приклад: do:- Y= sqrt(25).
Відповідь: Y=5

abs(X) – обчислює абсолютне значення X. Для додатних чисел повертає само число. Для від'ємних чисел повертає X*(-1);

Приклад1: do:- Y= abs(-6.2).
Відповідь: Y= 6.2

Подана в прикладі структура - рекурсивне дерево утворюється двома типами дерев:

- тип утворюється функтором tree і компонентами структури: вузлом(ціле число) і двома іншими деревами того ж самого типу;
- тип подається порожнім деревом з функтором empty.

Порожнє дерево обмежує рекурсивну структуру, таким же чином, як [ ] обмежує список.

За необхідністю рекурсивне дерево може мати скільки завгодно гілок.

Назви типів і функтори програміст обирає сам.

Visual Prolog дозволяє визначити будь-які рекурсивні типи. При цьому покажчики(адресні посилання) створюються і визначаються автоматично.

Ясно, що Visual Prolog дозволяє компонувати дерево з різних типів дерев, але одержана структура повинна відповідно оброблюватися процедурами.

3.1.2. Увід - вивід дерева

Дерево можна вводити предикатом redterm.

Приклад.

```
domains
tr=tree(real, tr, tr);
   empty
Predicates
do(tr)

goal
readterm(tr, D),Do(D).
clauses
do(empty).
do(tree(X,Y,Z)):- write(X), nl, write(Y), nl, write(Z).
```

Вводиться дерево:
tree(3.2, tree(4.1, empty, empty), tree(5.3, empty, empty))

Вивід: 3.2

3. РЕКУРСИВНІ СТРУКТУРИ ДАНИХ. ДЕРЕВА ЯК ТИПИ ДАНИХ

Мета:

Вивчити засоби Visual Prolog побудови рекурсивних дерев. Отримати практичні навички роботи з рекурсивними деревами.

3.1.Теоретичні відомості

3.1.1. Рекурсивні дерева

Для наочного графічного подавання складеної структури часто використовують дерево, в якому функтори структури подаються як вузли дерева, а компоненти як гілки дерева. Див рис.2.1 теми „Об'єкти даних складеної структури". На рисунку подано дерево граматичного розбору для речення на природній мові.

Дерево, як тип складеного об'єкту, використовують в програмах для швидкого пошуку певного даного, сортування даних, для керування роботою програми, тощо.

Нами буде розглянуто тип даного – рекурсивне дерево. Об'єкт рекурсивного типу будується так, що його частини є об'єкти того ж самого типу.

Тому тип даного – рекурсивне дерево у Visual Prolog подається у вигляді:

<Тип дерева> = <функтор>(<тип вузла>, <тип дерева>,..., <тип дерева>);
<функтор порожнього дерева>

Наприклад:

```
Domains
 tr = tree(integer, tr, tr);
      empty
```


Приклад2: do:- Y= abs(18).
Відповідь: Y=18

Приклад3: do:- Y= abs(0).
Відповідь: Y=0

random(X) – генерує випадкове число, де 0<=X<1.
Приклад: do:- Y= random(X).

random(Макс, X) – генерує випадкове ціле число
(i, o)
0<=X<Макс. Ціле число Макс вказує програміст.
Приклад: do:- Y= random(15,X).

randominit(X) – генерує випадкове початкове значення, як функцію системного часу. Обидві форми random використовують це початкове значення і базовий генератор випадкових чисел.
Приклад: do:- Y= randominit(X).

Val (тип, X) – обчислює вираз X і перетворює результат до вказаного числового типу. Контролює на переповнення.
Приклад: do:- readint(X), Y=val (ushort, 65000 + X).

1.1.5. Бінарні предикати

Бінарні предикати працюють з цілими числами по бітах, виконуючи над ними логічні операції та операції зрушення. При виконанні операцій знакові розряди розглядаються теж.

Бінарні предикати дозволяють одержати частину цілого числа, упакувати кілька чисел в одне ціле число, тощо.

Предикат **Bitand**(X,Y,Z) – виконує по бітах операцію AND над цілими числами. (i, i, o)

Наприклад: do:- bitand (3,6,Z), write(Z).

00011 = 3
00110 = 6
Результат: 00010 = 2

Розглянемо приклад застосування даного бінарного предикату.

Завдання: В ціле число X упаковані шифр цеху і шифр ділянки. Причому шифр ділянки займає 4 молодших двійкових розрядів числа, а шифр цеху, розряди двійкового числа, що залишилися. Одержати шифр ділянки.

Нехай X = 000000000001 0011_2 = 19_{10} . Виділимо маскою 15 чотири молодших розряди числа за допомогою бінарного предикату bitand. Старші розряди стануть 0.

Do(X):-bitand (X,15,Z), write ("Шифр ділянки "Z).

Вивід: Шифр ділянки Z = 3.

Предикат **Bitor**(X,Y,Z) – виконує по бітах операцію OR над цілими числами. (i, i, o)

Наприклад: do:- bitor (3,6,Z), write(Z).

00011 = 3
00110 = 6
Результат: 00111 = 7

Предикат **Bitxor**(X,Y,Z) - виконує по бітах операцію "XOR" над цілими числами.

(i, i, o)

Наприклад: do:- bitxor(5,3,Z), write(Z).

00101 = 5
00011 = 3
Результат: 00110 = 6

Предикат **Bitnot** (X, Z) – виконує інверсію бітів цілого числа X.

(i, o)

Наприклад: do:- bitnot(5, Z),write(Z).
Результат: Z = -6.

Порахувати середній бал для учнів 10^a класу.

19 Дан список, що має компонентами структури, які містять відомості про робітників заводу: прізвище та ім'я по батькові, робота(місце, посада, стаж, професія). Створити з обраної структури 2 структури, які містять: прізвище, посаду, місце роботи і прізвище стаж роботи.

20 Дан список, що має компонентами структури, які містять відомості про робітників заводу: прізвище та ім'я по батькові, робота(місце, посада, стаж, професія). Підрахувати кількість робітників заводу стаж яких більше 20 років.

21 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:

```
Default (Files (load ("*.pro"), new _ file(" work .pro")),
        Compile("memory"),
        Setup(Save _ sys _file("Prolog .sys")))
```

Виділити програмно всі можливі компоненти структури.

22 Написати програму, що вводить і зберігає в файлі структури, які містять відомості про рейтинг студентів і про результати сесії. Програма повинна використовувати один предикат вводу для різних структур. Відомості для зберігання виберіть самі.

23 Написати програму, що формує єдину структуру про рейтинг студента і про результати сесії. Програма повинна вводити відомості з клавіатури, а зберігати у файлі. Інформацію для структур оберіть самі.

24 Написати програму, що формує структури, які містять розклад занять групи 432, і зберігає структури в динамічній базі даних. Програма повинна вводити відомості з клавіатури. Інформацію для структур оберіть самі.

25 Написати програму, що формує в динамічній базі даних структури з відомостями про розклад літаків. Програма повинна вводити відомості з клавіатури. Інформацію для структур оберіть самі.

Виділити програмно компоненти структури, які відносяться до компіляції і задають ім'я файлу для зберігання установок в середовищі.

13 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:

Files/load - "*. Pro ", Files/new _file - " work . pro ",
Compile - " memory ", Setup/Save _ sys _file - "Prolog . sys"

Побудувати можливу структуру, що зберігає установки за замовченням для дій: завантажити файл, компілювати.

14 Написати програму, що вводить і зберігає дані про фільми вашої домашньої фільмотеки, дані про компакт диски для вашого комп'ютеру в динамічні базі даних. Програма повинна використовувати один предикат вводу для різних структур. Відомості для зберігання виберіть самі.

15 Дан файл, що має компонентами структури, які містять відомості про учнів: прізвище та ім'я по батькові, стать, адресу(місто, вулицю, дім, квартиру). Вивести на екран відомості про всіх хлопчаків, вік яких менше 10 років.

16 Дан файл, що має компонентами структури, які містять відомості про студентів: прізвище та ім'я по батькові, стать, адресу(місто, вулицю, дім, квартиру). Порахувати кількість студентів, які проживають у місті, назва якого вводиться з клавіатури.

17 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:

```
Default (Files(load("*.pro"), new _file (" work . pro")),
        Compile("memory"),
        Setup(Save _ sys _ file(" Prolog .sys")))
```

Замінити установку за замовчанням про розширення нового файлу та додати установку про розширення проекту.

18 Дан файл, що має компонентами структури, які містять відомості про учнів: прізвище та ім'я по батькові, стать, адресу(місто, вулицю, дім, квартиру), клас, оцінки за навчальними предметами.

Предикат **Bitleft** (X,Y,Z) - зрушує ліворуч біти свого операнду X, включаючи знаковий, на число бітів зазначених в операнду Y.

(i, i, o)

Результат міститься в операнду Z. Значення старших розрядів і знакового розряду зникають, їх заповнюють нові значення бітів числа, що здвигається. Молодші розряди заповнюються 0. Предикат bitleft працює аналогічно множенню X на 2^y.

Приклад: do:- bitleft (2,3,Z),write(Z).
Відповідь: Z=16. Порівняйте: $2*2^3=16$.

Предикат **Bitright** (X,Y,Z) – працює аналогічно предикату bitleft , але зрушує розряди праворуч.

(i, i, o)

Предикат bitright працює аналогічно цілочисловому діленню.

Приклад: do:- bitright (5,1,Z).

Відповідь: Z=2. Порівняйте 5 div 2 = 2.

1.1.6. Операції порівняння

Операції порівняння можуть порівнювати операнди одного типу або сумісних типів.

Нижче, в таблиці 1.3 подані приклади операцій порівняння.

Таблиця 1.3 — Приклади операцій порівняння

Операція	Зміст	Операції порівняння	Відповідь
X=Y	Дорівнює	3.1=2	Неправда
X<Y	Менше	2<3	Істина
X>Y	Більше	5.1>3	Істина
X< >Y	Не дорівнює	3*4< > 11	Істина
X≤ Y	Менше або дорівнює	X=1.2,X<=2	Істина
X≥ Y	Більше або дорівнює	X>=3	Помилка, змінна X не конкретизована

Операції порівняння порівнюють символьні дані по кодах. Дивись тему „Рядки”.

1.1.7. Приклади програм на Пролозі

Приклад1.

Завдання. Написати програму на Пролозі, яка розв’язує квадратне рівняння: AX2+BX+C=0.

```
Predicates

urawnen(real, real, real)
otwet(real, real, real)

Clauses

urawnen(A,B,C) :- D=B*B-4*A*C, otwet(A,B,D), nl.

otwet(_,_,D):-  D<0, write("Нема дійсних коренів").
otwet(A,B,D):- D=0, X= -B/(2*A),
                write("Два  дійсних однакових кореня = ",X).

otwet(A,B,D):-  W=sqrt(D),
                X1 =(-B+W)/(2*A), X2=(-B-W)/(2*A),
                write("X1=",X1),write("X2=",X2).

Goal
urawnen( 1,-15,50).
```

Одержимо результат: X1=10;X2=5.

В даній програмі були використанні арифметичні операції: '*', '-' , '/'. Операції порівняння: '=', '<'. Арифметична функція: sqrt(D).

Приклад 2.

Серед арифметичних операцій немає операції степеню. Скористаємося функцією exp, для написання програми, яка виконує цю операцію. За допомогою експоненти степінь обчислюється за наступною формулою:

Xn=exp(n * ln(x)).

8 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:

```
Default (Files(load(“*.pro”), new _file (“ work . pro”)),
        Compile(“memory”),
        Setup(Save _ sys _ file(“ Prolog .sys”)))
```

Замінити установку за замовчанням про розширення файлу на „asc” та установку результату компіляції на „exe file”.

9 Дан файл, що має компонентами структури, які містять відомості про учнів: прізвище та ім’я по батькові, стать, адресу(місто, вулицю, дім, квартиру), клас, оцінки за навчальними предметами. Створити з обраної структури 2: структуру з даними про учня і відомості про навчання.

10 Дан тип речення: група іменника, група дієслова.

Де, група іменника: прикметник, іменник, а група дієслова: дієслово прийменник іменник. Придумайте речення вказаного типу.

Для визначення в яку групу входить певний іменник, введіть ознаку для іменників відмінок.

Утворіть набір фактів, які мають слова речення з вказівкою частини мови і відмінка.

Побудуйте програмно структуру речення вказаного типу і виведіть її на екран.

11 Дан тип речення: група іменника і група дієслова.

Група іменника: прикметник, іменник.

Група дієслова: дієслово прийменник іменник.

Придумайте речення вказаного типу і побудуйте структуру-константу речення.

Виділити програмно зі структури і вивести на екран всі слова речення і вивести їх на екран у вигляді речення.

12 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:

```
Default (Files (load (“*.pro”), new _ file(“ work .pro”)),
        Compile(“memory”),
        Setup(Save _ sys _file(“Prolog .sys”)))
```

2 Дан тип речення: група іменника і група дієслова.
Група іменника: прикметник, іменник.
Група дієслова: дієслово прийменник іменник.
Придумайте речення вказаного типу і задайте структуру речення.
Виділити програмно зі структури і вивести на екран групи: іменника, дієслова.

3 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:
Default (Files (load (“*.pro”), new _ file(“ work .pro”)),
Compile(“memory”),
Setup(Save _ sys _file(“Prolog .sys”)))
Виділити програмно компоненти структури, які відносяться до меню Files.

4 Нехай структура містить установки роботи системи Турбо - Пролог за замовченням:
Default (Files (load (“*.pro”), new _ file(“ work .pro”)),
Compile(“memory”),
Setup(Save _ sys _file(“Prolog .sys”)))
Побудувати програмно можливу структуру , що зберігає установки за замовченням для дій: утворити файл, компілювати.

5 Написати програму, що вводить структури і зберігає дані про книги вашої домашньої бібліотеки, дані про касети вашої фонотеки в динамічній базі даних. Програма повинна використовувати один предикат вводу для різних структур. Відомості для зберігання виберіть самі.

6 Дан файл, що має компонентами структури, які містять відомості про учнів: прізвище та ім’я по батькові, стать, адресу(місто, вулицю, дім, квартиру). Вивести на екран відомості про всіх учнів, що проживають на вказаній вулиці. Назву вулиці вводить з клавіатури.

7 Дан файл, що має компонентами структури, які містять відомості про студентів: прізвище та ім’я по батькові, стать, адресу(місто, вулицю, дім, квартиру). Скомпонувати структуру для кожного студента, яка містить: прізвище студента і місто.

Завдання. Обчислити $Y = X^n$, де n натуральне, X дійсне число.

```
Predicates
stepen

Clauses
stepen:-write(" Введіть степінь"), readint(N),
        write("Введіть X"),readreal(X),
        Y= exp (N * ln (X)),write(" Результат ",Y).

Goal

Stepen.
```

1.2.Методичні вказівки до виконання самостійної роботи

При написанні програми треба звернути увагу на те, що типи значень змінних виразу і тип результату впливають на роботу програми.

Наприклад:
```
Predicates
        Do(integer)

        Goal
        Do(N),write(N).

        Clauses
         Do(N):- N = sqrt(2).
```
Замість очікуваної відповіді 1.4 на екран буде виведено 1, тому що тип результату integer. Якщо змінити тип на real, то відповідь буде вірною.

1.3.Зміст звіту

Звіт повинен містити наступні пункти:
1.3.1 Ціль роботи.
1.3.2 Таблицю арифметичних операцій і їх опис.
1.3.3 Таблицю операцій порівняння і їх опис.
1.3.4 Таблицю стандартних функцій та предикатів, бінарних предикатів і їх опис.
1.3.5 Завдання і текст програми.

1.4.Контрольні питання

1.4.1 Перелічити відомі вам арифметичні операції в порядку пріоритету виконання.

1.4.2 Назвіть тип результату для кожної з арифметичних операцій, в залежності від типу операндів.

1.4.3 Назвіть відомі вам операції порівняння. Для яких типів даних можна їх виконувати?

1.4.4 Які стандартні функції і предикати відомі вам?

1.4.5 Опишіть роботу бінарних предикатів.

1.4.6 Які властивості операцій вам відомі? Як ці властивості використовує Visual Prolog при роботі з виразами?

1.4.7 Що таке класи операцій за пріоритетом? В якому порядку виконуються операції одного класу?

1.5.Література

/1. с.380 – 392; 572 – 574; 591 – 593; 3. с. 57-64; 4. с.49-56 /

1.6.Індивідуальні завдання

1 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = \frac{\sqrt{|X-1| - \sqrt[3]{|Y|}}}{1 + \frac{X^2}{2} + \frac{Y^2}{4}}, \qquad B = X(\operatorname{arctg} Z + e^{-(x+3)}),$$

і з'ясовує чи можна побудувати трикутник зі сторонами A, B,C, де C=5.

2 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = \frac{3 + e^{Y-1}}{1 + X^2|Y - \operatorname{tg} Z|}, \qquad B = 1 + |Y - X| + \frac{(Y-X)^2}{2} + \frac{|Y-X|^3}{3},$$

2.2.Методичні вказівки до виконання самостійної роботи

При виконанні завдань в яких треба будувати структуру речення реалізуйте процес, який описано в прикладі побудови дерева граматичного розбору.

2.3.Зміст звіту

Звіт повинен містити наступні пункти:

2.3.1 Ціль роботи.

2.3.2 Завдання і текст програми.

2.3.5 Результати роботи програми.

2.4.Контрольні питання

2.4.1 В яких випадках в програмі зручно користуватися об'єктами складеної структури?

2.4.2 Як визначаються складені домени?

2.4.3Опишіть процес програмного формування об'єкту багаторівневої складеної структури на прикладі дерева граматичного розбору.

2.5.Література

/1. с.298 – 314; 2. с.93 – 110/

2.6.Індивідуальні завдання

1 Дан тип речення: група іменника, група дієслова.

Група іменника: прикметник, іменник.

Група дієслова: дієслово прийменник іменник.

Придумайте речення вказаного типу.

Для визначення в яку групу входить певний іменник, введіть семантичну ознаку для іменників: для першого іменника діяч, для другого іменника об'єкт.

Утворіть набір фактів, які мають слова речення з вказівкою частини мови і семантики іменників. Побудуйте програмно структуру речення вказаного типу і виведіть її на екран.

поверненні результату структурою-константою. Але конкретизована змінна в свою чергу входить в структуру вищого рівня і тому структура нижчого рівня укладається в структуру вищого рівня.

Процес повторюється поки не буде одержано дерево граматичного розбору.

Одержування компонент об'єкту складеної багаторівневої структури

Одержати компоненти багаторівневої структури можна за допомогою уніфікації.

Розглянемо програму, яка одержує компоненти дерева граматичного розбору.

```
domains
struktura_r = dgr ( gr_im, gr_di)          % 1 рівень

gr_im = grupa_imennuka1 (pr, im);          % 2 рівень
        grupa_imennuka2 (im)               % 3 рівень
gr_di = grupa_dieslova (di, gr_im)         % 2 рівень

pr = prujmennuk(string)                    % 3 рівень
im = imennuk(string)                       %  4 рівень
di = dieslovo(string)                      %  3 рівень

Predicates
Do(struktura_r)

Goal
DGR = dgr(grupa_imennuka1(prujmennuk("На"),
                          imennuk("столі")),
          grupa_dieslova  (dieslovo("стояла"),
                           grupa_imennuka2(imennuk("ваза")))),
Do(DGR).

Clauses
Do(DGR):-DGR= dgr(Gr_Im,Gr_Di),write(Gr_Im), nl, write(Gr_Di), nl,
          DGR=dgr(grupa_imennuka1((Pr, _),_), write(Pr).
```

Вивід:

```
grupa_imennuka1((prujmennuk("На"), imennuk("столі"))
grupa_dieslova(dieslovo("стояла"),grupa_imennuka2(imennuk("ваза")))
prujmennuk("На")
```


і обчислює мінімальну відстань від точки площини з координатами A і B до вершин трикутника: (-0.5, 0.5),(0.5, 0.5), (0.5,-0.5).

3 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = \frac{X + Y/(X^2+4)}{e^{-(x+2)} + 1/(X+4)^2}(Y+1), \qquad B = \frac{1+\cos(Y-2)}{X^4/2 + \sin^2 Z},$$

і виводить послідовність чисел 5.5, A і B у такому порядку, щоб вона була зростаючою.

4 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = Y + \frac{X}{Y^2 + \left|\frac{X^2}{Y + X^3/3}\right|}, \qquad B = (1 + tg^2 Z/2),$$

і знаходить максимум значень A і B і чисел з послідовності P, де P=[5.8,3,-4.5]. Кожне число послідовності задавати у вигляді факту.

5 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = \frac{2\cos(X - \pi/6)}{1/2 - \sin^2 Y}, \qquad B = 1 + \frac{Z^2}{3 + Z^2/5},$$

і виводить числа A, B і їх середнє арифметичне за спаданням.

6 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z обчислює A і B за формулами:

$$A = \frac{1 + \sin^2(X+Y)}{2 + |X - 2X/(1 + X^2Y^2)|} + X, \qquad B = \cos^2(\operatorname{arctg}\frac{1}{Z}),$$

і знаходить суми цифр значень [A], [B], де [R] ціла частина числа R. Вивести числа на екран разом з сумами.

7 Написати і налагодити програму, яка вводить з клавіатури числа X, Y, Z, обчислює A і B за формулами:

$$A = \ln\left|(Y - \sqrt{|X|})(X - \frac{Y}{Z + X^2/4})\right|, \qquad B = X - \frac{X^2}{3!} + \frac{X^3}{5!}.$$

З'ясувати, чи попадає точка з координатами A і B у коло з R=6 і центром (1,1).

8 Ввести дійсне число h.
Визначити чи має рівняння: $Ax^2+Bx+C=0$ дійсні корені, якщо:

$$A = \sqrt{\frac{|\sin 8h| + 17}{(1 - \sin 4h(\cos(h^2 + 18))^2}},$$

$$B = 1 - \sqrt{\frac{3}{3 + |tgAh^2 - \sin Ah|}};$$

$$C = Ah^2\sin(Bh) + Bh\cos(Ah).$$

Інакше відповіддю повинно бути повідомлення, що дійсних коренів немає.

9 Ввести дійсні числа A1, B1, C1, A2, B2, C2. З'ясувати чи вірно, що |A1B2-A2B1 |>=0.0001 і якщо вірно, то знайти розв'язок системи лінійних рівнянь:

$$A1x+B1y+C1=0,$$

$$A2x+B2y+C2=0.$$

Розподіл і аналіз

Процедура Append розбиває список слів на два підсписки за схемою:

1) [] [„На”, „столі”, „стояла”, „ваза”]
2) [„На”] [„столі”, „стояла”, „ваза”]
3) [„На” „столі”] [„стояла” „ваза”]
4) [„На” „столі” „стояла”] [„ваза”]
5) [„На” „столі” „стояла” „ваза”] []

Кожна процедура програми, що використовує процедуру Append, розподіляє список слів на два підсписки, і після кожного розподілу аналізує підсписки.

Під час аналізу визначається, чи є одержані підсписки слів вказаними граматичними групами, або, чи має список один елемент, який є необхідною частиною мови.

Якщо слова підсписку не мають необхідних характеристик, то процедура Append виконує наступний розподіл. Розподіл списку слів на підсписки закінчується, якщо підсписки мають вказані характеристики.

Визначення граматичної групи

Визначення, чи є підсписок вказаною граматичною групою, виконується іншою процедурою, яка в свою чергу може використовувати процедуру Append для перевірки з чого складається одержаний нею підсписок.

Процес повторюється до тих пір, поки список не буде мати одне слово. Якщо одержане слово має вказані граматичні характеристики(частину мови і відмінок), що визначається за фактом, то починається синтез структури.

Синтез структури речення

Кожна процедура побудови структури має другим аргументом структуру, компонентами якої є змінні. Якщо змінні конкретизуються значеннями при виконанні процедури, то другий аргумент стає структурою – константою.

Виклик процедури побудови структури на місці другого аргументу має вільну змінну. Тому вона конкретизується при

```
Fact (“стояла”, „дієслово”, “”).
Fact (“ваза”, „іменник”, “називний”).

Append([], L, L).
Append([H | L1], L2, [H | L3]):-append(L1, L2, L3).

pobudova_Dgr(R, dgr(Gr_im, Gr_di) ):- append(R1, R2, R),                (1)
                                      pobudova_Gr_Im1(R1, Gr_Im),
                                      pobudova_Gr_Di(R2,Gr_di).

pobudova_Gr_im1(R, grupa_imennuka1(Pr,Im)):-append(R1, R2, R),    (2)
                                            pobudova_pr(R1,Pr),
                                            pobudova_im(R2,Im).
Pobudova_Pr( [Pr | []],prujmennuk(Pr) ):-
                                  Fact(Pr, “прийменник”, „місцевий”).   (3)
Pobudova_Im( [Im | []], imennuk(Im) ):-
                                  Fact(Im, “іменник”, “місцевий”);      (4)
                                  Fact(Im, “іменник”, “називний”).
pobudova_Gr_Di( R, grupa_dieslova(D, Gr_im ) ):- append(R1, R2, R), (5)
                                      pobudova_Di(R1, D),
                                      pobudova_Gr_im2(R2, Gr_im).
Pobudova_Di([ D | [] ], dieslovo(D) ):- fact(D, “дієслово”, “”).        (6)
pobudova_Gr_im2(R, grupa_imennuka2(Im) ):- pobudova_Im(R,Im).    (7)
```

Робота програми

Дано речення, як список слів, і набір фактів, які містять слова речення з граматичною інформацією.

Розглянемо процес побудови структури речення – дерева граматичного розбору.

Побудова структури виконується в два етапи:

- розподіл списку на підсписки слів і аналіз підсписків слів;
- синтез структур різних рівнів.

(При виконанні записаної нерівності система сумісна і має один розв’язок).

10 Ввести дійсні числа a, b, c (a< >0). Повністю дослідити біквадратне рівняння

$$ax^4 + bx^2 + c = 0 \ .$$

Якщо дійсних коренів немає, то повинно бути виведено повідомлення, інакше повинно бути виведено 2 або 4 кореня.

11 Ввести натуральне число n. Обчислити n коренів:

$$\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}$$

12 Ввести натуральне число n. Обчислити:

$$\sqrt{3+\sqrt{6+\ldots+\sqrt{3(n-1)+\sqrt{3n}}}}$$

13 Ввести дійсне число a і натуральне n. Обчислити:

$$\frac{1}{1a}+\frac{1}{a(a+1)}+\ldots+\frac{1}{a(a+1)\ldots(a+n)}$$

14 Ввести дійсне число a і натуральне n. Обчислити:

$$\frac{a(x-2)(x-4)(x-8)\ldots(x-64)}{(x-1)(x-3)(x-7)\ldots(x-n)}$$

15 Ввести дійсні числа a, h і натуральне число n. Обчислити:

$$f(a)+2f(a+h)+2f(a+2h)+\ldots+2f(a+(n-1)h)+2f(a+nh),$$

де $f(x)=(x^2+1)\cos^2(nx)$.

16 Ввести натуральні числа m, n. За допомогою алгоритму Евкліда, Знайти НСД(m, n).

Алгоритм Евкліда знаходження найбільшого спільного дільника (НСД) невід'ємних цілих чисел такий: Нехай m і n цілі невід'ємні числа, які одночасно не дорівнюють 0, і нехай m ≥ n. Тоді, якщо n = 0, то НСД (n, m) = m. Якщо n < >0, то для чисел m, n, і r, де r – залишок від ділення m на n буде НСД (m,n) = НСД (n,r).

Наприклад,

НСД (15, 6)=НСД (6, 3)=НСД (3, 0)=3.

17 Ввести натуральні числа m і n. Знайти натуральні числа p і q, які не мають спільних дільників, і де m/n=p/q.

18 Знайти всі прості не скорочуванні дроби, які знаходяться між 0 і 1, знаменники яких не перевищують m. (Дріб задається двома натуральними числами – чисельником і знаменником).

19 Знайти найменше спільне кратне цілих додатних чисел m, n.

20 Написати і налагодити програму, яка перетворює ціле число, що записано у 16-річній системі числення, у десяткове число.

21 Написати і налагодити програму, яка вводить ціле десяткове число і записує його в 16-ричній системі числення.

22 Написати і налагодити програму, яка вводить ціле десяткове число і записує його у 8-ричній системі числення.

23 Натуральне число називають досконалим, якщо воно дорівнює сумі всіх своїх дільників за винятком самого себе. Число 6 досконале, тому що 6=1+2+3. Число 8 не досконале, тому що 8 < > 1+2+4. Ввести натуральне число n. Одержати всі досконалі числа, які менше n.

24 Ввести натуральне число n. Одержати всі менші ніж n прості числа.

Структура не може мати однакові функтори, тому групи іменників мають № : grupa_imennuka1, grupa_imennuka2.

2.1.8. Робота з об'єктами багаторівневих складених структур

Побудова об'єктів багаторівневих складених структур

Розглянемо приклад програми, яка будує структуру речення – дерево граматичного розбору.

```
domains
struktura_r = dgr ( gr_im, gr_di)          % 1 рівень

gr_im = grupa_imennuka1 (pr, im);          % 2 рівень
        grupa_imennuka2 (im)               % 3 рівень
gr_di = grupa_dieslova (di, gr_im)         % 2 рівень

pr = prujmennuk(string)                    % 3 рівень
im = imennuk(string)                       %  4 рівень
di = dieslovo(string)                      %  3 рівень

list = string*

predicates
pobudova_Dgr(list, struktura_r)    % Побудова дерева
pobudova_Gr_im1(list, gr_im)       %  Побудова групи іменника 1
pobudova_Gr_im2(list, gr_im)       %  Побудова групи іменника 2
pobudova_Gr_Di(list, gr_di)        % Побудова  групи дієслова
pobudova_Pr(list, pr)              % Побудова прийменника
pobudova_Im(list, im)              % Побудова іменника
pobudova_Di(list, di)              % Побудова дієслова

fact (string, string, string)      % Факти з граматичною інформацією
append(list, list, list)           %  Процедура, що розбиває список на
                                   %        усі можливі підсписки

Goal
R= [„На", „столі", „стояла", „ваза"], pobudova_Dgr(R, DGR),
 write (DGR).

Clauses
Fact ("На", „прийменник" , "місцевий").
Fact ("столі", „іменник", "місцевий").
```

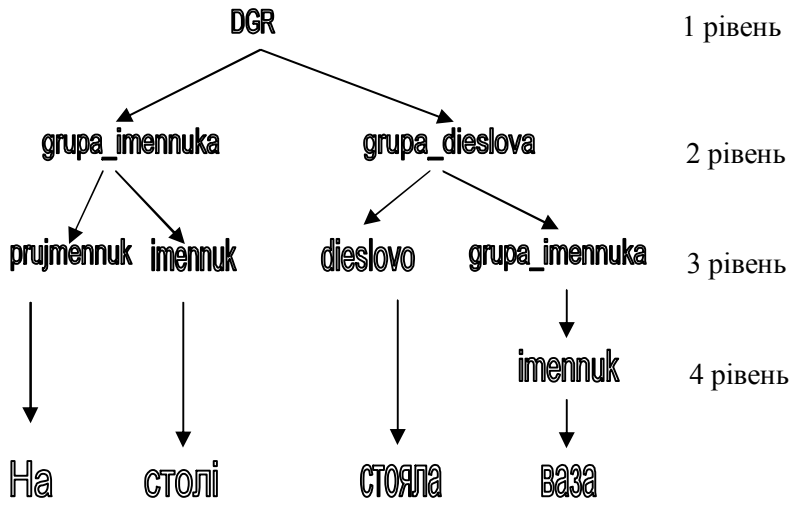

Рисунок 2.1 Дерево граматичного розбору

В дереві на рис.2.1 показано функтори(імена структур) і значення аргументів внутрішніх структур(слова).

Опис багаторівневих структур виконується по рівням. Для вказаної структури опис буде:

```
domains
struktura_r = dgr ( gr_im, gr_di)          % 1 рівень
gr_im = grupa_imennuka1 (pr, im);          % 2 рівень
        grupa_imennuka2 (im)               %3 рівень
gr_di = grupa_dieslova (di, gr_im)         % 2 рівень
pr = prujmennuk(string)                    % 3 рівень
im = imennuk(string)                       %  4 рівень
di = dieslovo(string)                      %  3 рівень
```

В програмі для речення „На столі стояла ваза” структура, що описана буде мати вигляд:

```
DGR = dgr(grupa_imennuka1(prujmennuk(“На”),
                          imennuk(“столі”)),
          grupa_dieslova (dieslovo(“стояла”),
            grupa_imennuka2(imennuk(“ваза”))))
```


25 Написати і налагодити програму, яка визначає знак і значення тригонометричних функцій в залежності від аргументу для функцій виду:

sin(a), cos(a), tg(a), ctg(a), де a>360 градусів, результати роботи вивести на друк в такій формі: sin(1440)=0.

2. ОБ’ЄКТИ ДАНИХ СКЛАДЕНОЇ СТРУКТУРИ

Мета:

- Одержати знання про організацію об’єктів складеної структури у Visual Prolog;
- Навчитися:

- організовувати логічно пов’язану між собою інформацію, як об’єкт складеної багаторівневої структури;
- одержувати з об’єкту складеної багаторівневої структури його компоненти різних рівнів;
- будувати програмно об’єкти складеної багаторівневої структури;
- визначати програмні ситуації і ситуації предметної області в яких треба використовувати дані, як об’єкти складеної структури, або компоненти складеної структури.

2.1.Теоретичні відомості

2.1.1. Загальні відомості про об’єкти складеної структури

Об’єкти даних складеної структури (надалі просто структури) дозволяють інтерпретувати різну інформацію, яка логічно пов’язана між собою, як єдине ціле.

Структури організуються таким чином, щоб потім можна було легко виділити з них компоненти.

Розглянемо приклад структури.

Адреса: Київська 10-а, складається з двох даних – вулиці і № дому. Об’єднаємо два даних в одно ціле – структуру з ім’ям address:

address (“Київська”,”10-а”)

Поданий запис - це об'єкт даних, який називають структурою - константою. Він починається з імені структури address. Ім’я структури прийнято називати функтором За функтором у дужках вказані два простих об’єкти даних, які є компонентами структури - константи.

Структура об’являється в секції Domains і використовується на місці аргументу предикату.

Розглянемо приклад об’яви структури, яка розглянута вище та її використання:

```
Domains
a = address(vul, dim)
vul, dim = string

Predicates
do(a)

Goal
Do(address(“Мира”, “17-а”)).

Clauses
Do(Adr):- write(Adr).
```

В секції domains оголошується тип структури(домен) - a; address – функтор структури; типи компонент структури – vul, dim. Типи компонент структури належить до стандартного типу string.

Visual Prolog дозволяє визначати типи користувача після визначення типу структури.

2.1.2. Відмінність об’єкту складеної структури від предикату

Форма запису розглянутої структури тотожна формі запису предикату. Тому треба знати правила, як відрізнити структуру від предикату.

Структура – це дане і використовується на місці даного, як аргумент предикату. Структура описується в секції Domains.

Предикат використовується для подавання фактів, правил, цілей і описується в секції Predicates або Facts.


```
Domains
n=i(integer);
    r(real);
    s(string)


database
fact(n)


Predicates
do
repeat


Goal
Do,save("ST.asc").


Clauses
repeat.
repeat:-repeat.

Do:-repeat, readterm(n,T), assertz(fact(T)), nl, write("Закончить?(y/n)"),
    readchar(C), nl, C='y', save(“f.bd”,a).
```

2.1.7. Об’єкти багаторівневих складених структур

Visual Prolog дозволяє конструювати структури о декількох рівнях. Тобто компоненти структури можуть самі бути структурами.

Розглянемо приклад: Побудувати структуру речення „На столі стояла ваза” за частинами мови.

Структуру речення за частинами мови називають деревом граматичного розбору. Її компонентами є групи іменника і групи дієслова. В свою чергу групи іменника можуть складатися з іменника, прийменника, тощо. А група дієслова складається з дієслова і групи іменника. Наприклад для вище вказаного речення дерево граматичного розбору буде:

Приклад1.

```
domains
vlast = rab (sub, prof, stag, posada);
        misce (sub, adr);
         xust (sub, x)

adr = address(misto, vul, dim, kvart)
sub, prof, posada, misto, vul, x = string
stag, dim, kvart = integer

Facts
Obj(vlast)

Predicates
do(vlast)

Goal
Readterm(vlast, X), do(X).

Clauses
do(X):- assertz(obj(X),a), nl,save("f.db",a).
```

Вводяться і заносяться у внутрішню базу даних:

```
Obj (rub ("Кузьменко", "програміст", 3, "ст. інженер"))
Obj (misce ("Кузьменко", address ("Шкільна", "2" , "127"))
Obj (xust ("Кузьменко", "графіка"))
```

Застосовуючи складений домен, до визначення об'єкту легко можна додати інші властивості, додавши новий тип властивостей в секції Domains до типу Vlast.

Останній приклад програми показує, що одним аргументом можна визначати різну кількість компонент структур.

За допомогою структур, можна також задавати один аргумент предикату для різних типів даних.

Приклад2.

Ввести одним предикатом вводу дані різних типів, сформувати внутрішню базу даних і зберегти її у файлі.

2.1.3. Увід - вивід об'єктів складеної структури

Структури читають з пристроїв стандартним предикатом readterm:

readterm(тип структури, змінна)
(i, o)

Предикат вводить структуру і привласнює її вільній змінній. Під час вводу за типом контролюється вид структури.

Вивід структури виконується стандартним предикатом write.

Приклад: Написати програму, що вводить структуру з ім'ям address і виводить її на екран.

```
Domains
a =  address ( vul, dim)
vul, dim = string

Predicates
Do ()

Goal
Do.

Clauses
Do:- readterm( a, Adr), write(Adr).
```

При об'яві структури біля типів компонент структури можна вказувати коментарій: «вулиця», «дім». Компілятор коментарій не компілює.

2.1.4. Робота з об'єктами складеної структурами

Привласнювання структур

Структуру можна привласнити вільній змінній за допомогою процедури уніфікації при використанні предикату '=' або при зіставленні предикатів.

Приклад1:

```
X = date ("Травень", 1, 2005)
```

Вільній змінній X привласнюється значення date ("Май", 1, 2005) при використанні предикату ‘=’.

Приклад2:

```
Domains
dat = data(mis, chuslo, rik )
mis = string
chuslo, rik = integer

Predicates
do(dat)

Goal
Do(data(“Березень”, 8, 2005)).

Clauses
Do(Dat):- write(Dat).
```

Цільове твердження Do(data(“Березень”, 8, 2005)) зіставляється з умовним твердженням Do(Dat). При цьому структура data(“Березень”, 8, 2005) уніфікується з вільною змінною Dat. В результаті вільній змінній Dat привласнюється структура data(“Березень”, 8, 2005).

Виділення компонент структури

При роботі зі структурами часто потрібно вміти виділяти компоненти структури.

Виділити зі структури компоненти можна за допомогою процедури уніфікації при використанні предикату ‘=’ або при зіставленні предикатів.

Приклад1.

```
Do :- date ("Май", 1, 2005) = date (M, D, G), write(M, ‘ ’, D, ‘ ‘,G ).
```

Структура-константа зіставляється зі структурою, компоненти якої подані як вільні змінні M,D,G, за допомогою предикату ‘=’. При цьому змінні одержують значення: M = "Май", D = 1, G = 2005.

Розглянемо приклад програми, де виділяються компоненти структур при зіставленні предикатів.


```
domains
rb = rab (sub, prof, stag, posada) /* суб’єкт, професія, стаж роботи, посада  */
msc =  misce (sub, adr) /* суб’єкт, адреса */
adr = address(misto, vul, dim, kvart) /* місто, вулиця, дім, квартира */
xst = xust (sub, x) /* суб’єкт, хист */
sub, prof, posada, misto, vul, x = string
stag, dim, kvart = integer
```

При такому описі структур для кожного типу властивостей суб’єкту треба вводити в предикат новий аргумент.

```
Predicates
obj (rb, msc, xst)
```

Пролог дозволяє оголошувати складений домен, який дозволяє для опису об’єкту використовувати тільки один аргумент, який описує ту чи іншу властивість.

```
Domains
vlast = rab (sub, prof, stag, posada);
        misce (sub, adr);
        xst = xust (sub, x)

adr = address(misto, vul, dim, kvart)
sub, prof, posada, misto, vul, x = string
stag, dim, kvart = integer

Predicates
Obj(vlast)
```

Крапка з комою читається як "або".

Складений домен дозволяє записувати будь-який тип властивостей з перелічених на місці одного аргументу.

Розглянемо приклад програми, яка записує в динамічну базу даних властивості об’єктів.

Але в факті не зафіксовано, що означає кожний аргумент.

Факт, що використовує замість простого об'єкту структуру, фіксує властивості об'єкту.

Наприклад:

term (tip ("integer"), min (-32768), max (32767), platforma_bit (16))

Розглянемо програму в якій опис об'єкту подається фактом, а властивості об'єкту подані у вигляді структури.

Приклад.

Завдання: Вивести на екран відомості про типи даних певної мови. Для кожного типу вказується ідентифікатор типу, мінімальне і максимальне значення, платформа в бітах.

```
Domains
tp= tip(string)       /* Опис властивостей об'єкту */
mn= min(integer)
mx= max(integer)
bit= platforma_bit(integer)

Predicates
Term(tp, mn, mx, bit) /* Опис об'єкту фактом */

Clauses
term ( tip ("integer"), min (-32768), max (32767), platforma_bit (16)).
term ( tip ("short"), min (-32768), max (32767), platforma_bit (16)).

Goal
term(T, Mn, Mx, Pl),nl.
```

Результат: tip ("integer") min (-32768) max (32767) platforma_bit (16)
tip ("short") min (-32768) max (32767) platforma_bit (16)

2.1.6. Оголошення складених доменів

Об'єкт можна описувати з різних аспектів. Тому опис об'єкту може складатися з властивостей різних типів. Наприклад, нехай суб'єкт має наступні властивості: роботу, місце проживання, хист. Кожну властивість суб'єкту можна описувати доменом простим чи складеним.

Приклад2.

Завдання: Нехай подано набір фактів з адресами школярів. Адреса складається з назви вулиці та № дому. Вивести на екран адреси всіх школярів.

```
Domains
fam, vul, dim = string
adr = address ( vul, dim )     /*Структура описує адрес */

Predicates
do
inform (fam, adr)  /*Предикат описує факт: Прізвище, адреса */

Goal
do.

Clauses
inform ("Іванов", address ("Київська","10-а")).
inform ("Петров", address ("Дніпровська","18")).
inform ("Сидорів", address ("Київська","1")).

do:- inform (F, address (Vul, Dim )), write (F, ' ' ,Vul, ' ', Dim),nl,
fail.
do.
```

Результат: Іванов Київська 10-а
Петров Дніпровська 18
Сидорів Київська 1

При зіставленні першого факту і поточної цілі
inform (F, adress (Vul, Dim)), змінній F привласнюється прізвище школяра, а змінним Vul, Dim привласнюються компоненти структури - назва вулиці і № дому з факту. Після чого одержана інформація виводиться на екран. Механізм звороту повторює ці дії для кожного факту.

Побудова структури

Приклад. Розглянемо програму, в якій будується об'єкт складеної структури.

Завдання: Побудувати програмно структуру, яка описує властивості кубика.

```
Domains
Color, mat = string
Rebro = real
kub  =  kubik ( rebro, color, mat )

Predicates
do

Goal
do.

Clauses
do:- write(“Введіть довжину ребра ”), readreal(D),
     write(“Введіть колір кубика ”), readln(C),
     write(“Введіть матеріал кубика”), readln(M),
     K= kubik(D, C ,M ), write (K).
do.
```

2.1.5. Призначення об’єктів складеної структури

Використання об’єктів складеної структури для спрощення програми.

Структуру можна використовувати для того, щоб передавати великий набір значень як єдиний об'єкт, а при необхідності застосовувати уніфікацію для їхнього поділу. Такий підхід спрощує програму, не треба оголошувати велику кількість аргументів предикату.

Приклад.

Завдання: Нехай у предмета є такі характеристики: довжина, ширина, висота, вага, матеріал, колір. Написати програму, що вводить характеристики предмету і передає їх в програму пошуку, яка шукає назву предмету з такими характеристиками.

```
Domains
pr = predmet (dovg, shir, vusot, vaga, mater, color)
dovg, shir, vusot, vaga = real
```



```
nazv, mater, color = string
Predicates
Vvid(pr)
Poshuk(pr)
Svedeny(nazv, dovg, shir, vusot, vaga, mater,
color)
Goal
Vvid (Strukt), poshuk(Strukt).

Clauses
Svedeny("статуетка",6.2, 4.0, 15.0, 300.5,
"мармур","білий").
Svedeny("ваза", 10.0, 10.0, 10.0, 10.0, "скло",
"синій").
Svedeny("книга", 21.4, 13.2, 3.1, 80.0, "папір",
"червоний").

Vvid(Strukt):- write(" уведіть характеристики
предмету як структуру"), nl,
                Write("predmet (Довжина,
Ширина,Висота,Матеріал, Колір)"),
                nl, readterm (pr, Strukt).
Poshuk(Strukt):- Strukt = predmet(Dov, Sh, Vus,
Vag, Mat, Col),
          Svedeny(Naz, Dov, Sh, Vus, Vag, Mat,Col),
Write(Naz),nl.
```

Процедура Poshuk одержує характеристики предмету у вигляді структури. Після чого застосовуючи уніфікацію виділяє компоненти у змінні Dov, Sh, Vus, Vag, Mat, Col для пошуку; шукає назву предмету з такими характеристиками і видає на екран.

Використання об’єктів складеної структури для опису предметної області.

Нехай предметною областю буде певна мова програмування. Опишемо тип даного „integer” фактом:

term („integer”, -32768, 32767, 16).