Лабораторна робота 5
ПОШУК РОЗВ'ЯЗКУ В ПРОСТОРІ СТАНІВ СИСТЕМИ

Мета роботи:

1. Ознайомлення з застосуванням опису предметної області у просторі станів; набуття навичок складання декларативних програм для розв'язання логічних задач.

2. Набуття навичок оформлення результатів роботи програми у вигляді таблиць..

Завдання роботи: розробити описи станів систем та скласти відповідні програми для задач, наведених у пунктах завдання.

Короткі теоретичні відомості

Метод простору станів часто застосовується в теорії керування та теорії ігор. Його застосування полягає в тому, що сукупність характеристик предметної області описують сукупністю компонентів вектору станів, а після цього визначають закони чи правила, за якими ці компоненти можуть змінюватись.

Для дискретних задач простір станів зручно представляти у вигляді неорієнтованого або орієнтованого графа, вершини якого відповідають описам станів предметної області, а дуги – операторам, що переводять систему з одного стану в інший.

Процес пошуку розв'язку (цільової вершини на графі) полягає в послідовному виконанні алгоритмічно ідентичних кроків, на кожному з яких визначено поточний стан (поточну вершину) і відбувається розгляд станів або вершин, безпосередньо пов'язаних з поточною. Вказівники на знайдені вершини розміщуються у списковій структурі (черга або стек), після цього та ж послідовність операцій повторюється з черговою вершиною. При цьому для кожної знайденої вершини виконується перевірка того, чи не є вона цільовою. Якщо цільова вершина знайдена, то збережені в процесі пошуку вказівники дозволяють відновити повний шлях від початкової вершини до цільової, який і буде розв'язком задачі.

Для складних задач часто існує декілька різних способів формального представлення стану системи. Вдало обране подання задачі може суттєво спростити її розв'язання. Тому, якщо ви помічаєте, що є певні проблеми з розв'язанням задачі, спробуйте застосувати творчий підхід і описати задачу іншим способом.

Порядок виконання роботи

1. Ознайомтесь з короткими теоретичними відомостями на початку документу.

2. Нехай задано карту-схему деякої місцевості (приклад карти наведено на рис. 4). На карті позначено населені пункти і шляхи між ними. Вінні-Пух хоче з пункту A потрапити у пункт Z.
Розгляньте наступну програму, в якій описано дану задачу:

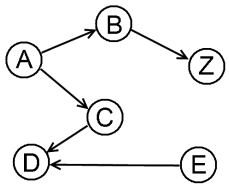


Рис. 4. Приклад карти місцевості

```
predicates
    road (symbol,symbol)
                can_reach
(symbol,symbol)
clauses
    road (a,b).
    road (a,c).
    road (b,z).
    road (c,d).
    road (e,d).
can_reach (X,Y) if road (X,Y).
can_reach (X,Y) if can_reach (X,Q), road (Q,Y).
```

Виконайте цю програму в середовищі Turbo Prolog. Поставте запит до програми так, щоб машиною було знайдено шлях з пункту A в пункт Z. Простежте за послідовністю пошуку розв'язку задачі. Для цього можна використати предикати друку (write), додавши їх в праві частини правил, і/або режим трасування.
Поставте до цієї ж програми також інші запити та вивчіть алгоритм їх виконання.

Підказка. Уникайте графів, в яких є циклічні шляхи, бо в такому випадку програма схильна до зациклювання. Для того, щоб уникнути цієї проблеми, засобів мови Пролог, вивчених вами досі, недостатньо.

3. Намалюйте в зошиті власний граф (схему доріг). Відповідно до нього опишіть систему в Пролозі і визначте, чи зможе Вінні-Пух потрапити в пункт призначення. Вважайте, що рух по кожній дорозі може бути одно- або двостороннім.

4. Ускладніть ситуацію. Тепер пункти (вершини графа) з'єднані шляхами (ребрами) двох видів: дорогами й тунелями (або норами). Змініть програму для цього випадку.

5. Якщо Вінні-Пух поїсть, то при цьому він так розтовстіє, що в нори він уже не влізе, а зможе пересуватися тільки по дорозі. Змініть програму так, щоб урахувати – їв він чи ні.

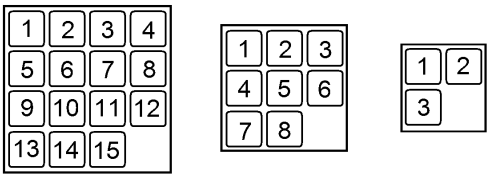
```
Тобто, тепер у вас будуть, наприклад, цільові запити типу:
може_потрапити(z, їв)     або     може_потрапити(z, не_їв)
(можна зробити і інакше)
```

6. Дороги й нори залишаються як і раніше, Вінні-Пух виходить у шлях голодним, а в пункті (m) стоїть бочка з медом. Зрозуміло, що Вінні-Пух перш за все біжить туди, де стоїть мед. І якщо він в цю точку зможе потрапити, то добре поїсть і після цього вже не зможе влізти в жодну нору. Починаючи із цього моменту, він зможе пересуватися тільки по відкритій місцевості, тобто тільки по дорогах.
Тепер його мета – пункт (z).
Ваше завдання – урахувати цю обставину в програмі.

7. Розробіть опис у вигляді станів системи та напишіть програму на Пролозі для розв'язання старовинної задачі: «Селянину необхідно перевезти через ріку вовка, козу та капусту. На річці є невеликий човен, в який може вміститися лише селянин та вовк, або селянин та коза, або селянин та капуста. Отож, в той час, поки селянин перевозить у човні на інший берег один з своїх товарів, інші два очікують його на берегах. Вовка з козою не можна залишати на одному березі без догляду людини, бо вовк з'їсть козу. Так само не можна залишати козу наодинці з капустою. Вкажіть послідовність дій, яка дозволить селянину переправити усіх трьох неушкодженими.»

```
Цю задачу можна описувати різними способами. Один з наочних способів
полягає в тому, що для опису дозволених переміщень використовуються
предикати такої структури:
move(селянин_до, вовк_до, коза_до, капуста_до,
     селянин_після, вовк_після, коза_після, капуста_після).
```

8. Розробіть опис у просторі станів системи та напишіть програму на Пролозі для розв'язання задачі, подібної до гри-головоломки «15» (рис. 5, а).



а б в

Рис. 5. Головоломки до завдання 4:
а – гра в «15»; б – гра в «8»; в – гра в «3».

Головоломка «15» складається з п’ятнадцяти занумерованих квадратних фішок, розташованих на квадратному полі розміром 4х4 місця. Одне місце залишається вільним, тому на нього можна пересунути одну з сусідніх фішок; на те місце, яке при цьому звільнилося, пересунути іншу, і т.д. Мета гри «15» полягає в тому, щоб, починаючи з довільного розташування фішок на гральному полі, перемістити їх в початкове (впорядковане) розташування, як на рис. 5, а. Оскільки програма для задачі з 15-ма фішками вийде досить громіздкою, рекомендується розв’язати спрощену задачу для гри у «8» (рис. 5, б), а розпочати розв’язання з найпростішої гри у «3» (рис. 5, в), з якою вам буде досить просто випробувати свою ідею опису системи. Якщо на грі у «3» ви зможете втілити *Ідею* (з великої літери), то реалізація головоломок «8» і «15» уже ніяких інтелектуальних зусиль не вимагає – все відбувається шляхом простого нарощування розміру.

9. Ознайомтеся з *Додатками А та Б*, де описані функції введення-виведення; з *Додатками В, Г*, де описані прийоми оформлення виведення у вигляді таблиці.

Подивіться також *Додаток Д*, хоча з цією інформацією ви вже знайомі.

Додаток Е не має жодного стосунку до цієї лабораторної роботи і наведений тут просто так. Не вивчайте його, бо після його вивчення можна виконати другу модульну контрольну, а це небезпечно.

10. Оформіть у вигляді таблиці виведення однієї з ваших програм. Наприклад, Ви можете виводити у вигляді табличок стан на ігровому полі головоломки «15» чи «8», можете надрукувати у вигляді таблиці з чотирма або п’ятьма колонками протокол переміщення селянина, вовка, кози і капусти через річку, тощо.

11. Підготуйте звіт про виконання лабораторної роботи.

Контрольні запитання

1. Поясніть, що таке опис системи в просторі станів?
2. Поясніть, яким чином відбувається пошук шляху інтерпретатором Прологу при виконанні програм Ваших завдань.
3. Запропонуйте опис розглянутих задач засобами теорії графів.

4. Модифікуйте задачу про мандрівника так, щоб при виборі маршруту було враховано довжини шляхів. Складіть відповідну програму мовою Prolog.

ДОДАТОК А

Синтаксис предикатів консольного виведення інформації в Turbo Prolog (1)

OUTPUT PREDICATES (предикаты вывода информации на печать)

write(Variable|Constant *)

nl - новая строка (new line)

writef(FormatString, Variable|Constant*) – печать по формату. Формат указывается подобно тому, как это делается в Си.

In the format string the following options are known after a percentage sign:
В строке формата после знака % ставятся следующие спецификаторы:

%d Normal decimal number. (chars and integers)
%u As an unsigned integer. (chars and integers)
%x As a hexadecimal number. (chars and integers).
%X As a long hexadecimal number. (strings, database reference numb).
%s Strings. (symbols and strings).
%c As a char. (chars and integers).
%g Reals in shortest posible format (default for reals)
%e Reals in exponetial notation
%f Reals in fixed notation
%lf Only for C compatibility (fixed reals)

(и еще кое-что пропущено как явно лишнее)

Теперь смотрите, что об этом написано в другом источнике. Тут информация полнее:

write(e1,e2, … , eN) – выводит константы или переменные в системное или текущее окно. Аргументы не могут быть свободными переменными.
writef(формат строки, e1,e2, … , eN) – выводит константы или переменные в системное или текущее окно в заданном формате. Формат строки содержит обычные символы, которые выводятся без модификации, и форматы спецификации формы %–M.PF.

спецификация	значение
– (дефис)	Показывает, что поля выравниваются слева. по умолчанию – справа.
M поле	Десятичное число, описывающее минимальный размер поля
.P поле	Описывает или точное представление числа с плавающей точкой, или максимальное количество выводимых в строке символов
F поле	Описывает специальные форматы вывода F формат вещественного числа с фиксированной точкой 12.08 E формат вещественного числа в экспоненциальной форме 3.3e-6 G короткий формат вещественного D формат символов или целых как десятичное число U формат символов или целых как десятичное без знака X формат символов или целых как шестнадцатеричного числа C формат символов или целых чисел как символа S строка символов

Пример. Рассмотрите следующий пример программы:

```
        predicates
ppp
a

        clauses
a.
ppp if A=1,B=1,
writef("%4%6%1%8%1%5%1","|",A,"|",B,"|",'m',"|"),nl,fail.
        goal
ppp
```

Тут числа после процента задают размер поля для каждого из перечисленных далее элементов. Это и имелось в виду в таблице под названием «поле M».

… Крім того, у рядку виводу предикатів write, writef можуть бути присутні такі комбінації:
\n - newline (переведення рядка)
\t - tabulator (символ табуляції)
\nnn - character with code nnn (символ з цифровим кодом nnn)

ДОДАТОК Б

Синтаксис предикатів консольного введення інформації в Turbo Prolog

ВВЕДЕННЯ З КЛАВІАТУРИ
readchar(*змінна_типу_char*) - читає один символ з поточного пристрою введення (за замовчуванням - клавіатура)
readint(*змінна_типу_integer*) - читає ціле число із пристрою введення
readreal(*змінна_типу_real*) - читає дійсне число із пристрою введення
readln(*змінна_типу_string*) - читає символи з поточного пристрою введення, поки не прочитає символ повернення каретки (ASCII 13)

ДОДАТОК В. **Синя програма**

Приклад оформлення виведення Turbo Prolog псевдографічними елементами («рамочки», «таблички»)

Синя програма на вигляд страшна і якась погана, але не поспішайте втрачати свідомість. Не хвилюйтеся, натисніть на штангу інтервала. (*Реальна цитата із книжки по Прологу*)

Спробуйте перенести це в Пролог і запустити. Скоро ви переконаєтесь, що нічого страшного немає. Незважаючи на всі забобони, які можуть у вас з'явитися при першому погляді на цей хаос і жах, насправді програма просто елементарна.

Всі ці \196\196\193 і т.п. – це коди псевдографічних символів (лінії, кутки й інші, якими програма малює рамку таблички). (В додатку А такий формат описаний як "**\nnn**".) Про те, що означають тексти на зразок %1%6%1%5%1%8%1%5%1 , – також написано у Додатку А.

Для того, щоб було легше розібратися, прочитайте *Додатки Г і Д*.

```
domains
name   = symbol
age    = integer
height = real
sex    = char

predicates
person(name,age,height,sex)
tb1
tb2
tb3
tb4
w1
w2

clauses
person("Dima",  21, 1.8,  'm').
person("Ilya",  20, 1.81, 'm').
person("Kolya", 26, 1.92, 'm').
person("Lena",  18, 1.73, 'f').
person("Olya",  20, 1.65, 'f').
person("Masha", 24, 1.75, 'f').
person("Ivan",  20, 1.79, 'm').
person("Dasha", 22, 1.66, 'f').

tb1:- write("all data"),nl,
      write("\218\196\196\196\196\196\194\196\196\196\196\196\194\196\196\196\196\196\196\196\196\194\196\196\196\196\196\191"),nl,
```

```
      write("\179 name \179 age \179 height \179 sex \179"),nl,
       write("\195\196\196\196\196\196\196\197\196\196\196\196\196\197\196\196\196\196\196\196\196\19
6\197\196\196\196\196\196\180"),nl,fail.
tb1:- person(A,B,C,D),
      writef("%1%6%1%5%1%8%1%5%1","\179",A,"\179",B,"\179",C,"\179",D,"\179"),nl,fail.
tb1:-
write("\192\196\196\196\196\196\196\193\196\196\196\196\196\193\196\196\196\196\196\196\196\196\19
3\196\196\196\196\196\217").


tb2:- write("female, age > 20"),nl,
      write("\218\196\196\196\196\196\196\194\196\196\196\196\196\194\196\196\196\196\196\191"),nl,
      write("\179 name \179 age \179 sex \179"),nl,
      write("\195\196\196\196\196\196\196\197\196\196\196\196\196\197\196\196\196\196\196\180"),nl,fail
.
tb2:- person(A,B,_,'f'),B>20,
      writef("%1%6%1%5%1%5%1","\179",A,"\179",B,"\179",'f',"\179"),nl,fail.
tb2:- write("\192\196\196\196\196\196\196\193\196\196\196\196\196\193\196\196\196\196\196\217").


tb3:- write("all, age >= 20"),nl,
       write("\218\196\196\196\196\196\196\194\196\196\196\196\196\194\196\196\196\196\196\196\196\19
6\194\196\196\196\196\196\191"),nl,
      write("\179 name \179 age \179 height \179 sex \179"),nl,
       write("\195\196\196\196\196\196\196\197\196\196\196\196\196\197\196\196\196\196\196\196\196\19
6\197\196\196\196\196\196\180"),nl,fail.
tb3:- person(A,B,C,D),B>=20,
      writef("%1%6%1%5%1%8%1%5%1","\179",A,"\179",B,"\179", C,"\179",D,"\179"),nl,fail.
tb3:-
write("\192\196\196\196\196\196\196\193\196\196\196\196\196\193\196\196\196\196\196\196\196\196\19
3\196\196\196\196\196\217").


tb4:- write("male, height > 1.75"),nl,
       write("\218\196\196\196\196\196\196\194\196\196\196\196\196\196\196\196\194\196\196\196\196\19
6\191"),nl,
      write("\179 name \179 height \179 sex \179"),nl,
       write("\195\196\196\196\196\196\196\197\196\196\196\196\196\196\196\196\197\196\196\196\196\19
6\180"),nl,fail.
tb4:- person(A,_,B,'m'),B>1.75,
      writef("%1%6%1%8%1%5%1","\179",A,"\179",B,"\179",'m',"\179"),nl,fail.
tb4:-
write("\192\196\196\196\196\196\196\193\196\196\196\196\196\196\196\196\193\196\196\196\196\196\21
7").

w1:-  makewindow(1, 15, 3, "Info", 0, 0, 16, 35).
w2:-  makewindow(2, 2, 3, "Info", 0, 37, 16, 35).

goal
w1,tb1,
readchar(_),
w2,tb2,
readchar(_),
removewindow(1,1),
removewindow(2,1),
w1,tb3,
readchar(_),
w2,tb4,
```

```
readchar(_),
removewindow(),
shiftwindow(1),
clearwindow(),
window_attr(15),
cursor(6,13),
write("Bye! :)"),
readchar(_).
```

ДОДАТОК Г

Як ще можна виводити таблиці (поради)

Таблиця виводиться в три етапи : виведення заголовка, виведення тіла таблиці і заключна частина. Ця послідовність залишається незмінною, незалежно від того, які символи використовуються для малювання таблички. У попередньому додатку був наведений приклад того, як намалювати таблицю за допомогою псевдографічних символів. Але можна обійтися і без них.

Наприклад, побудуємо таблицю з даними із чиєїсь записної книжки: ім'я - телефон - дата народження.

Спочатку виводимо заголовок (шапку) таблиці:

```
pr_table:- write("-----------------------------------"), nl,
           write("I  name   I   phone  I   date       I"), nl,
           write("-----------------------------------"), nl,
           fail.
```

(Як ви бачили, в синій програмі (додаток В) замість мінусів і букв I було використано більш придатні псевдографічні символи, але ідея та ж сама.)

потім тіло таблиці:

```
pr_table:- hand_book(N, P, D),
           write("I  ", N, "  I  ", P, "  I  ", D, "  I"), nl, fail.
```

(Наявність *fail* призводить до того, що ця дія повторюється, аж доки не буде виведено всі книги, наявні в програмі. Після цього виконання перейде до наступного рядка.)

і нарешті - заключна частина:

```
pr_table:-       write("-----------------------------------"), nl.
```

Запит на вивід таблиці формується в секції goal: pr_table.
Для того, щоб Пролог виконав всі три правила, служить предикат відкату - fail, розміщений після перших двох правил.

ДОДАТОК Д. Синтаксис предикатів *window

(тобто makewindow, removewindow, clearwindow і всяке_інше_window)

Пролог-програма може забезпечити багатовіконний інтерфейс. Для цього передбачені вбудовані предикати:

– створення вікна makewindow
– видалення вікна removewindow
– очищення вікна clearwindow
– перемикання активного вікна gotowindow, shiftwindow.

Синтаксис предиката makewindow

makewindow(№ вікна, кольори тексту й тла, кольори рамки, заголовок рамки,
координати лівого верхнього кута вікна: № рядка 0..23, № стовпчика 0..79,
висота вікна, ширина вікна)

Коди кольорів наведені в таблиці.

Для одержання потрібної колірної гами коди необхідно додати, наприклад:

Синій символ - 1
На зеленому тлі - 32
Мерехтить - 128

Загальний код дорівнює сумі кодів кольори тексту, тла й спецефектов: 1+32+128=161, от цю цифру й ставимо у функцію makewindow.

Кольори	Код кольорів Тексту / тла
Чорний – black	0/0
Синій – blue	1/16
Зелений – green	2/32
Блакитний – cyan	3/48
Червоний – red	4/64
Фіолетовий – magenta	5
Коричневий – brown	6
Ясно-сірий – lightgray	7
Темно-сірий – darkgray	8
Яскраво-синій – lightblue	9
Яскраво-зелений – lightgreen	10

Яскраво-блакитний – lightcyan	11
Рожевий – lightred	12
Малиновий – lightmagenta	13
Жовтий – yellow	14
Білий – white	15
мерехтіння	128
підкреслення ???	1 (????????)

removewindow - видалення поточного вікна

removewindow(№ вікна, параметр)
removewindow(WindowNo,Refresh)
(Integer,Integer) - (i,i)
Refresh = 0 Don't refresh background.
= 1 Do refresh background

clearwindow - очищення поточного вікна
gotowindow(№ вікна) - перехід до вікна із заданим номером
shiftwindow(№ вікна) - перемикає або повертає номер активного вікна

ДОДАТОК Е

Як виводити інтерактивне меню

Давайте розглянемо приклад.

```
clauses
  mainmenu:-printmenu, read(X), do_variant(X).
  printmenu:-
    write("      Оберіть потрібну дію:"), nl,
    write("1 – завантажити дані з файлу"), nl,
    write("2 – провести розрахунки"), nl,
    write("3 – зберегти результати на диску"), nl,
    write("4 – вихід з програми"), nl.

  do_variant(1):- ....  ,
                  % тут - опис дій, які потрібні для завантаження даних
  mainmenu,
                  % знову надрукувати меню і спитати про чергову дію
  fail.           % на всяк випадок, в разі необхідності.

  do_variant(2):- ....    ,
                  %  опис дій, які потрібні для проведення розрахунків
  mainmenu,
                  % знову надрукувати меню і спитати про чергову дію
  fail.            % на всяк випадок, в разі необхідності.
..........
.......... % Далі описуємо решту варіантів.

  do_variant(4).
            %  В кінці стоїіть КРАПКА.
            %  Тобто, в цьому випадку доказ успішний,
            %  і виконання програми припиняється.
```