2.14. Простые объекты

Любой объект, который представляет сам себя, называется простым объектом.

Простой объект данных может быть или переменной, или константой:

- Константа определяет конкретный объект, не подлежащий изменениям.
- Переменная представляет неопределённый объект, значение которого устанавливается в процессе согласования с любым допустимым конкретным объектом.

2.15. Сложные объекты – структуры

Объект, который представляет другой объект или совокупность объектов, называют структурой.

Структура – это целостный объект, имеющий несколько компонентов, каждый из которых так же может быть структурой.

У структуры имеется имя и один или больше аргументов. Аргументы структуры могут быть как простыми, так и сложными объектами.

На рис. 1 показана схема структуры **os** (операционная система), у которой объекты **description** (описание), **property** (свойства) и **rating** (рейтинг) также являются структурами. Остальные объекты – простые.

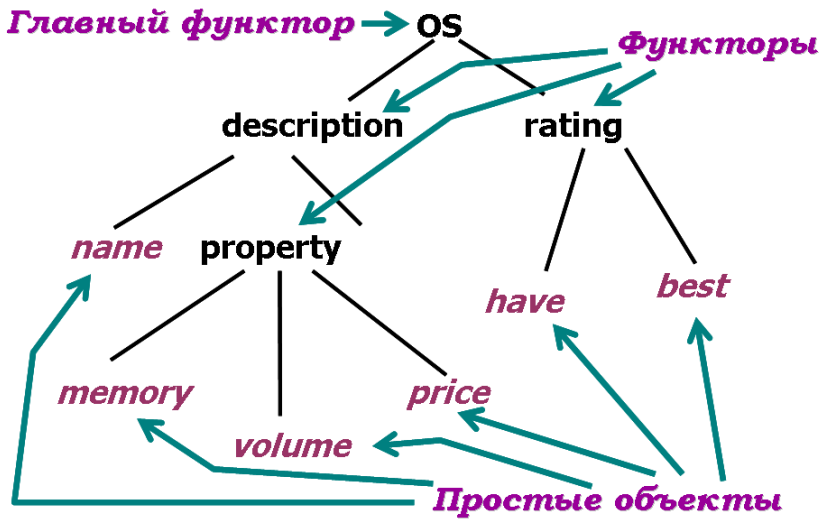

Рис. 1. Схема структуры *Операционная система*

Имя, которое идентифицирует структуру и объединяет аргументы в целостный объект, называется функтор (functor).

Самый первый функтор называют главным функтором.

Хотя сложный объект не является предикатом, об организации его компонент говорят как о предикатной структуре.

На языке *Пролог* структура (рис. 1) записывается следующим образом:

os (description (windowsNT, property (16, 120, 2500)), rating (14.2, 11))

Примеры сложных объектов:

дата (1, сентябрь, 2009) *% 1 сентября 2009 года*
дата (День, май, 2009) *% Числа мая 2009 года*
% Имена и фамилии авторов книги с названием Lambda-Calculus
book (author (Fname, Lname), "Lambda-Calculus")

Объявление структур производится в секции **domains**.

Формат объявления:

доменноеИмя = функтор (СписокТиповОбъектов)

Доменное имя и функтор при объявлении могут совпадать. Следует помнить, что доменное имя пишется слева от знака равенства, а функтор – справа.

Объявление структуры *Операционная система* (рис. 1) будет следующим:

```
domains
        % структуры
        os = os (description, rating)
        description = d (name, property)
        property = p (memory, volume, cost)
        rating = r (have, best)

        % простые объекты
        name = symbol
        memory, volume = integer
        cost, have, best = real
```

Здесь доменные имена: **os, description, property, rating name, memory, volume, cost, have, best**. Функторы: **os, d, p, r**.

Пример 2-6. Фрагмент программы ввода и вывода одной структуры.

```
domains
        % объявление структур
        os = os (description, rating)
        description = d (name, property)
        property = p (memory, volume, cost)
        rating = r (have, best)
        % объявление простых объектов
        name = symbol
        memory, volume = integer
        cost, have, best = real
predicates
        readOs (os)             % ввод структуры
        nondeterm run           % целевое утверждение
```

```
goal  run.
clauses
        run :-
                readOs (Os ),                   % ввод одной структуры
                nl, nl, write (Os),             % вывод структуры
                nl, write ( "Подтверждаете ввод (y/n)? " ),    readchar (Ch),
                Ch = 'y'.
        run :-
                nl, write ( "Повторите ввод ещё раз" ), nl, nl, run.
        % ввод одного объекта
        readOs (os (d (Name, p (Memory, Volume, Cost)), r (Have, Best))) :-
                write ("ХАРАКТЕРИСТИКИ ОС"),
                nl, nl, write ("Тип ОС:  "),            readln (Name),
                write ("Объём ОЗУ, Мбайт:  "),    readint (Memory),
                write ("Память на МД, Мбайт: "),  readint (Volume),
                write ("Стоимость, тыс. руб.:  "),  readreal (Cost),
                nl, write ("РЕЙТИНГ"), nl, nl,
                write ("Имеют ОС (%):  "),          readreal (Have),
                write ("Считают лучшей (%):  "), readreal (Best).
```

Пример 2-7. Фрагмент программы ввода и вывода трёх структур.

```
domains
        % объявление структур
        os = os (description, rating)
        description = d (name, property)
        property = p (memory, volume, cost)
        rating = r (have, best)
        % объявление простых объектов
        name = symbol
        memory, volume = integer
        cost, have, best = real
        % номер структуры
        j = integer
predicates
        nondeterm rw3os (j, os)         % ввод-вывод трёх структур
        run                             % целевое утверждение
goal  run.
clauses
        % цель
        run :- rw3os (1, _), exit.
        % нерекурсивная фраза для завершения повторений
        rw3os (4, _) :-
                write ("Введена информация о трёх ОС.\n").
        % рекурсивная фраза
        rw3os (J, os(d (Name, p (Memory, Volume, Cost)), r (Have, Best))) :-
                write ("ХАРАКТЕРИСТИКИ ОС"),
                nl, nl, write ("Тип ОС:  "),            readln (Name),
                write ("Объём ОЗУ, Мбайт:  "),    readint (Memory),
```

```
        write ("Память на МД, Мбайт: "),  readint (Volume),
        write ("Стоимость, тыс. руб.:  "),  readreal (Cost),
        nl, write ("РЕЙТИНГ"), nl, nl,
        write ("Имеют ОС (%):  "),          readreal (Have),
        write ("Считают лучшей (%):  "), readreal (Best),
        % переопределение значения номера объекта
        K = J + 1,
        % рекурсивный вызов с новым значением аргумента
        rw3os (K, _),
        J1 = 4 – J,          % точка возврата
        % вывод структуры
        nl, writef  ("%) %\n", J1, Name),
        writef  ("%\t%\t%\n", Memory, Volume, Cost),
        writef  ("%\t%\n", Have, Best).
```

Структуры последовательно вводятся и размещаются в стеке до тех пор, пока номер объекта станет равным 4. Далее согласуется *нерекурсивная фраза*, и становится возможным согласовать подцели второго правила **rw3os**, начиная с точки возврата.

Структуры последовательно извлекаются из стека и выводятся в порядке, обратном их вводу.

Пример лабораторной работы № 3-2 приведён в [1].

2.16. Множественное объявление

2.16.1. Множественное объявление объектов

Множественное объявление объектов вместе с функторами используют, чтобы один и тот же предикат мог работать с объектами разных типов.

Символ *точка с запятой* в объявлении отображает связку **OR (ИЛИ)**.

Пример 2-8. Множественное объявление объектов.

```
domains
        article = book (author, title) ;
        horse (name) ;
        pencil (num) ;
        pen
        author, title, name = string
        num = integer
predicates  nondeterm item (article)
clauses
        item (book ( "Лихтарников", "Логические задачи")).
        item (book ( "Волкова", "Искусство формализации")).
        item (horse ( "сивка-бурка")).
        item (pencil (2)).
        item (pen).
goal  item (X).
```

Решение:

X=book("Лихтарников","Логические задачи")
X=book("Волкова","Искусство формализации")
X=horse("сивка-бурка")
X=pencil(2)
X=pen
5 Solutions

Доменное имя **article** может быть использовано в объявлении разных предикатов, предназначенных для описания отношений между объектами разных типов. Объект **pen** в примере задан функтором без аргументов, что аналогично символьной константе.

Пример 2-9. Использование множественного объявления объектов.

```
domains
        возраст = i (integer); r (real); s (string)
predicates
        сколькоЛет (возраст)
clauses
        сколькоЛет (i (5)).
        сколькоЛет (r (17.5)).
        сколькоЛет (s ("мне 14 лет")).

% Варианты целевых утверждений с ответами
goal   сколькоЛет (i (Возраст)).
Возраст = 5

goal   сколькоЛет (r (Возраст)).
Возраст = 17.5

goal   сколькоЛет (s (Возраст)).
Возраст = Мне 14 лет
```

2.16.2. Множественное объявление предикатов

Объявление в программе предиката с одним и тем же именем для объектов разного типа *называется множественным*.

Во множественном объявлении одинаковые предикаты должны стоять в секции **predicates** один под другим.

Пример 2-10. Использование множественного объявления предикатов.

```
predicates
        add (real, real, real)
        add (integer, integer, integer)
        add (symbol, symbol)
clauses
        add (X, Y, Res) :- Res = X + Y.
        add ( X, S ) :- concat (X, "!!!", Res) .
```

% Варианты целевых утверждений с ответами
goal add (2.5, 2.5, S).
S = 5

goal add (3, 2, Sum).
Sum = 5

goal add (vivat, H).
H = vivat!!!

Предикат **add** благодаря множественному объявлению и правилам может быть использован для сложения целых и вещественных чисел, а также для присоединения к символьной строке трёх восклицательных знаков.

2.17. Арифметика в Прологе

Каждая операция есть предикат.

2.17.1. Обычные арифметические операции

К обычным относятся арифметические операции, показанные в табл.7.

Таблица 7

Обычные арифметические операции

Знак операции	Название операции
+	сложение
–	вычитание
*****	умножение
/	деление
mod	остаток от деления целых чисел
div	целочисленное деление

2.17.2. Операции отношения

Операции отношения относятся в *Прологе* к арифметике. Это также встроенные предикаты, но для удобства используется обычная нотация:

< <= > >= = <>

Сравнивать можно объекты **только** всех встроенных стандартных типов. Исключение составляет операция сравнения на равенство для структур, аргументы которых поддаются согласованию.

2.17.3. Равенство и предикат равенства

Предикат равенства можно использовать для работы с простыми объектами и со сложными объектами, имеющими одинаковую предикатную структуру.

Предложение **N = A + 5** эквивалентно предикату **eq (A, 5, N).**

Известно, что каждая переменная может иметь два состояния *(свободная или связанная)*. Для рассматриваемого предложения возможные состояния переменных показаны в табл. 8.

Таблица 8

Возможные состояния переменных в предложении N = A + 5

A	N
свободна	свободна
свободна	связана
ˇ **связана**	ˇ **свободна**
ˇ **связана**	ˇ **связана**

Известно, что для правильного вычисления выражения к моменту вычисления все переменные в правой части **(A + 5)** должны быть связаны, следовательно, реальны два последних варианта.

В зависимости от состояния объекта в левой части предикат равенства работает либо на конкретизацию этого объекта, либо на сравнение с объектом правой части предиката.

Если переменная в левой части предиката равенства свободна, то она конкретизируется значением правой части.

Если переменная в левой части предиката равенства связана, то она сравнивается на равенство со значением в правой части.

2.17.4. Встроенные предикаты

Встроенные предикаты предоставляет то, что обычно называют *обычными математическими функциями* (табл. 9).

Использование этих предикатов аналогично использованию стандартных математических функций в языке C.

Таблица 9

Встроенные предикаты вычисления математических функций

Предикат	Назначение	Предикат	Назначение
abs (X)	абсолютное значение	**cos (X)**	тригонометрические функции, требующие, чтобы переменная была представлена в радианах
exp (X)	экспонента, основание **e**	**sin (X)**	
ln (X)	натуральный логарифм	**tan (X)**	
log (X)	десятичный логарифм	**arctan (X)**	
round (X)	округление	**trunc (X)**	усечение младших разрядов

Существуют также встроенные предикаты для выполнения побитовых операций **and**, **or**, **xor**, **not**, а также сдвига и генерации псевдослучайных чисел.

Методические указания

1. *Довбуш Г. Ф.* Лабораторные работы по логическому программированию. Методические указания. – СПб: ПГУПС, 2005.