|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра общей информатики

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**по дисциплине**

«Функциональное программирование»

“Программирование на языке Prolog”

Выполнили студенты группыИКБО-11-17 Ибрагимова Ч.Т.

Принял Аввакумов Г.Е.

Лабораторная работа выполнена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2019г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

«Зачтён» «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2019г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Москва 2019

# Содержание

[Содержание 2](#__RefHeading___Toc768_3988325111)

[Цель работы 3](#__RefHeading___Toc420_714581061)

[Постановка задачи 3](#__RefHeading___Toc770_3988325111)

[Этапы выполнения 4](#__RefHeading___Toc499_2476617287)

# **Цель работы**

Освоение основных понятий языка Пролог.

# **Постановка задачи**

1. Нарисуйте схему родства. Составьте программу Родственные отношения, которая кроме родственных отношений parent (родитель) и ancestor (предок) программа должна содержать одно или несколько из следующих отношений: brother (брат); sister (сестра); grand-father (дедушка); grand-mother (бабушка); uncle (дядя).
2. Составьте программу, используя отношения likes ("нравится") и can\_buy ("может купить").
3. Составьте собственную программу, состоящую из фактов и правил. Проверьте ее работу.

# Этапы выполнения

1. Нарисуйте схему родства. Составьте программу Родственные отношения, которая кроме родственных отношений parent (родитель) и ancestor (предок) программа должна содержать одно или несколько из следующих отношений: brother (брат); sister (сестра); grand-father (дедушка); grand-mother (бабушка); uncle (дядя).

Во время анализа программы была составлена схема родства.

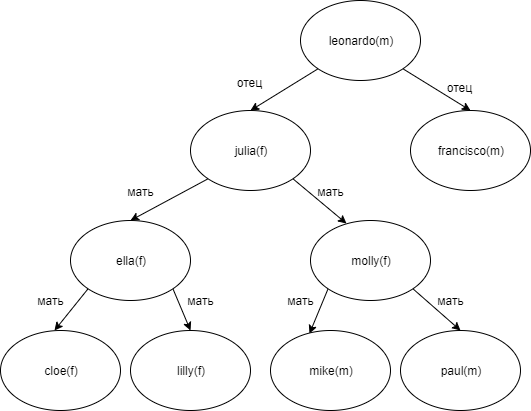


Рис. 1. Схема родства.

Далее была создана программа с новыми реализованными отношениями.

|  |
| --- |
| domains  name=symbol  predicates  nondeterm parent(name,name)  nondeterm brother(name,name)  nondeterm ancestor(name,name)  nondeterm sister(name,name)  nondeterm female(name)  nondeterm male(name)  nondeterm mother(name,name)  nondeterm father(name,name)  nondeterm grandfather(name,name)  nondeterm grandmother(name,name)  nondeterm uncle(name,name)  nondeterm child(name, name)    clauses  parent(leonardo, julia).  parent(leonardo, francisco).  parent(julia, molly).  parent(julia, ella).  parent(ella, cloe).  parent(ella, lilly).  parent(molly, mike).  parent(molly, paul).    male(mike).male(paul).male(leonardo).male(francisco).  female(lilly).female(cloe).  female(molly).female(julia).  child(Y,X):-parent(X,Y).  mother(X,Y):-parent(X,Y),female(X).  father(X,Y):-parent(X,Y),male(X).  ancestor(X,Z):-parent(X,Z).  ancestor(X,Z):-parent(X,Y),ancestor(Y,Z).  brother(X,Y):-parent(Z,X),parent(Z,Y),male(X), X<>Y.  sister(X,Y):-parent(Z,X),parent(Z,Y),female(X), X<>Y.  grandfather(X,Y):- father(X,Z), parent(Z,Y).  grandmother(X,Y):- mother(X,Z), parent(Z,Y).  uncle(X,Y):- brother(X,Z),parent(Z,Y), X<>Z.  goal  %sister(cloe,lilly).  %sister(X,Y).  %brother(X,Y).  %grandfather(X,Y).  %grandmother(X,Y).  uncle(X,Y). |

Листинг 1. Исходный код программы.

При последовательном тестировании каждой закомментированной строки выдаёт следующий результат:

* sister(X,Y) результат:

X=julia, Y= francisco

X=molly, Y=ella

X=cloe, Y=lilly

3 Solutions

* brother(X,Y) результат:

X=francisco, Y=julia

X=mike, Y=paul

2 Solution

* grandmother(X,Y) результат:

X=julia, Y=mike

X=julia, Y=paul

X=julia, Y=cloe

X=julia, Y=lilly

4 Solutions

* grandfather(X,Y) результат:

X= leonardo, Y=ella

X= leonardo, Y=molly

2 Solutions

* uncle(X,Y) результат:

X=francisco, Y=ella

X=francisco, Y=molly

2 Solutions

1. Составьте программу, используя отношения likes ("нравится") и can\_buy ("может купить").

|  |
| --- |
| predicates  nondeterm likes(symbol,symbol)  nondeterm can\_buy(symbol, symbol)  nondeterm person(symbol)  nondeterm exist(symbol)  nondeterm haveMoney(symbol)  clauses  person(lilly).  person(molly).  person(paul).  person(bob).  person(gosha).  person(gleb).  person(mike).  haveMoney(molly).  haveMoney(bob).  haveMoney(gleb).  haveMoney(mike).  haveMoney(gosha).  likes(lilly, car).  likes(molly, candy).  likes(paul, milk).  likes(mike, dreams).  likes(gosha, girls).  likes(gleb, house).  exist(car).  exist(candy).  exist(milk).  exist(dreams).  exist(house).  can\_buy(X, Y) :- person(X),exist(Y), likes(X, Y), haveMoney(X).  goal  can\_buy(molly, candy). |

Листинг 2. Исходный код программы.

* can\_buy(X, Y). результат:

X=molly, Y=candy

X=gleb, Y=house

X=mike, Y=dreams

3 Solutions

1. Составьте собственную программу, состоящую из фактов и правил. Проверьте ее работу.

|  |
| --- |
| domains  animal\_name, color, name = string  Predicates  nondeterm cat( animal\_name )  nondeterm dog( animal\_name )  color( animal\_name, color )  nondeterm have( name, animal\_name )  nondeterm rodoslovnaya(animal\_name )  nondeterm animal( animal\_name)  Clauses  cat( butsi ).cat( korni ).cat( mac ).  dog( rover ).dog( fles ).dog( spot ).  color( butsi, brown ).  color( korni, black ).  color( mac, yellow ).  color( rover, yellow ).  color( spot, white ).  color( fles, black\_and\_white ).  animal(X):-cat( X ); dog( X ).  rodoslovnaya( X ):-animal( X ), have( tom, X ).  rodoslovnaya( X ):-animal( X ), have( keit, X ).  have( tom, X ):-color( X, black ); color( X, brown ).  have( keit, X ):-dog( X ),  not( color( X, white )),  not( have( tom, X )).  have( alan, mac ):-not( have( keit, butsi )),  not( rodoslovnaya( spot )).  goal  animal( X ), not( have( \_ , X )). |

Листинг 3. Исходный код программы.

результат:

X=spot

1 Solution