**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Факультет №8 «Прикладной математики и физики»**

**Кафедра вычислительной математики и программирования**

**Курсовой проект**

**по курсу**

**«Логическое программирование»**

|  |  |
| --- | --- |
| **Студент:** | Сорокин Д.М. |
| **Группа:** | 8О-207Б, №13 |
| **Преподаватели:** | Сошников Д.В. Левинская М.А |
|  |  |
| **Оценка:** |  |
| **Дата:** |  |

2 курс, 1 семестр

Москва 2016

# Задание

1. Создать родословное дерево своего рода на несколько поколений (3-4) назад в стандартном формате GEDCOM.
2. Преобразовать файл в формате GEDCOM в набор утверждений на языке Prolog, используя одно из следующих представлений с использованием предиката **parents(потомок, отец, мать).**
3. Реализовать предикат проверки/поиска следующих родственников **троюродный брат или сестра**
4. Реализовать программу на языке Prolog, которая позволит определять степень родства двух произвольных индивидуумов в дереве, например:

*?- relative(brother, ‘Петя’, X).*

*X = ‘Вася’*

*?- relative(W,’Петя’,’Вася’).*

*X = brother X = child – father % т.е. Вася является ребенком отца Пети*

*X = child – mother % т.е. Вася является ребенком матери Пети*

# Описание метода решения

1. Реализовать парсер файла GEDCOM на языке C#, который будет выписывать предикаты согласно варианту. Вывод парсера направить в файл \*.pl
2. Парсер написан мною так, что программа на прологе из него выходит уже готовая. То есть предикаты отношений согласно варианту я писал в код парсера.

Теперь относительно самой программы на прологе:

1. После предикатов parents, реализовать предикат согласно варианту (**triple**), его я буду реализовывать через предикаты **double, bro\_or\_sist.**

* **bro\_or\_sist –** предикат проверяет на брата или сестру. Смотрит общие ли родители у выбранных людей, и не и тот же этот человек
* **double –** проверяет на двоюродное родство. Чтобы люди были в двоюродном родстве, их родители должны быть братьями или сестрами, данный предикат проверят всевозможные варианты родства между родителями.
* **Triple –** принцип работы такой же как и у **double,** только теперь родителей проверяют на двоюродное родтсво.

1. Далее реализую предикаты отношений **relative**. Стандартные отношения отце, мать, ребенок и т.д.
2. После них, если не будет найдено обычных отношений, с помощью поиска в глубину будем выводить путь от одного родственника до другого.

# Текст программы

Парсер на С#

**using** **System**;

**using** **System.IO**;

**using** **System.Collections.Generic**;

**using** **System.Data**;

**using** **System.Linq**;

**using** **System.Runtime.CompilerServices**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**namespace** **CP\_C\_Sharp**

{

**class** **Program**

{

**public** **static** StreamReader Reader = File.OpenText("Sorokin Family.ged");

**static** **void** **Main**(**string**[] args)

{

**var** dictOfMembers = **new** Dictionary<**string**, **string**>();

dictOfMembers["He"] = "Неизвестен";

dictOfMembers["She"] = "Неизвестена";

List<**string**> predicats = **new** List<**string**>();

**string** tmp = "";

**string** id = "";

**string** name = "";

**while** ((tmp = Reader.ReadLine()) != **null**)

{

**if** (tmp.StartsWith("0 @I"))

id = tmp.Substring(**3**, **7**);

**if** (tmp.StartsWith("1 NAME"))

{

name = tmp.Substring(**7**);

name = name.Replace("/", "");

**if** (name == " ")

name = "Неизвестно";

dictOfMembers[id] = name;

id = "";

}

**if** (tmp.StartsWith("0 @F"))

predicats = PredicatsWriter(dictOfMembers, predicats);

}

File.WriteAllLines("cp\_sorokin.pl", predicats);

File.AppendAllText("cp\_sorokin.pl", @"

bro\_or\_sist(X, Y) :-

parents(X, F, M),

parents(Y, F, M),

X\=Y.

double(X, Y) :-

parents(X, F1, M1),

parents(Y, F2, M2),

X\=Y,

(bro\_or\_sist(F1, F2);

bro\_or\_sist(F1, M2);

bro\_or\_sist(M1, F2);

bro\_or\_sist(M1, M2)).

triple(X, Y) :-

parents(X, F1, M1),

parents(Y, F2, M2),

X\=Y,

(double(F1, F2);

double(F1, M2);

double(M1, F2);

double(M1, M2)).

relative(child, Child, Parent) :-

parents(Child, \_, Parent);

parents(Child, Parent, \_).

relative(husband, Husband, Wife) :-

parents(\_, Husband, Wife).

relative(wife, Wife, Husband) :-

parents(\_, Husband, Wife).

relative(bro\_or\_sist, X, Y) :-

bro\_or\_sist(X, Y).

relative(double, X, Y) :-

double(X, Y).

relative(triple, X, Y) :-

triple(X, Y).

relative(father, Father, Child) :-

parents(Child, Father, \_).

relative(mother, Mother, Child) :-

parents(Child, \_, Mother).

relative(Way, First, Last) :-

dpath(First, Last, Way), !.

print\_ans([]).

print\_ans([H|Tail]):-

print\_ans(Tail),

write(H), write(' - ').

next(Curr, HasBeen, Y, Rel) :-

(relative(Rel, Curr, Y);

relative(Rel, Y, Curr)),

not(member(Y, HasBeen)).

dFS(X, X, \_, \_).

dFS(Curr, Last, T, [Rev|Tl]) :-

next(Curr, T, Next, Rev),

dFS(Next, Last, [Curr|T], Tl),

!.

dpath(First, Last, Way) :-

dFS(First, Last, [], RevWay),

reverse(RevWay, Way),

print\_ans(Way),

!.");

}

**private** **static** List<**string**> PredicatsWriter(Dictionary<**string**, **string**> dict, List<**string**> predicats)

{

**string** tmp = " ";

**string** husband = "He";

**string** wife = "She";

List<**string**> children = **new** List<**string**>();

**while** (!tmp.StartsWith("1 RIN"))

{

tmp = Reader.ReadLine();

**if** (tmp == **null**)

{

Console.WriteLine("Incorrect .ged file");

**return** predicats;

}

**if** (tmp.StartsWith("1 HUSB"))

husband = tmp.Substring(**8**, **7**);

**if** (tmp.StartsWith("1 WIFE"))

wife = tmp.Substring(**8**, **7**);

**if** (tmp.StartsWith("1 CHIL"))

children.Add(tmp.Substring(**8**, **7**));

}

//foreach (var child in children)

//{

// string pred = string.Format("parents(\"{0}\", \"{1}\", \"{2}\").", dict[child], dict[husband].TrimStart(), dict[wife].TrimEnd());

// predicats.Add(pred);

//}

// спасибо visual studio за упрощение кода

predicats.AddRange(children.Select(child =>

$"parents(\"{dict[child]}\", \"{dict[husband].TrimStart()}\", \"{dict[wife].TrimEnd()}\")."));

**return** predicats;

}

}

}

Вывод программы:

**parents**("Борис Сорокин", "Михаил Сорокин", "Татьяна Василенко").

**parents**("Денис Сорокин", "Михаил Сорокин", "Татьяна Василенко").

**parents**("Татьяна Василенко", "Иван Василенко", "Лидия Тимофеева").

**parents**("Наталья Василенко", "Иван Василенко", "Лидия Тимофеева").

**parents**("Михаил Сорокин", "Борис Сорокин", "Софья Каганович").

**parents**("Валерия Рыбакова", "Михаил Рыбаков", "Наталья Василенко").

**parents**("Захар Горбатенко", "Семен Горбатенко", "Наталья Василенко").

**parents**("Александр Каганович", "Михаил Каганович", "Мина Каганович").

**parents**("Юрий Коганович", "Михаил Каганович", "Мина Каганович").

**parents**("Софья Каганович", "Михаил Каганович", "Мина Каганович").

**parents**("Армине Мектерян", "Ваган Мектерян", "Рузанна").

**parents**("Карина Мектерян", "Ваган Мектерян", "Рузанна").

**parents**("Илья Барабанщиков", "Александр Барабанщиков", "Татьяна Харитонова").

**parents**("Яна Барабанщикова", "Александр Барабанщиков", "Татьяна Харитонова").

**parents**("Павел Барабанщиков", "Александр Барабанщиков", "Татьяна Харитонова").

**parents**("Борис Сорокин", "Василий Сорокин", "Полина Кондратьева").

**parents**("Полина Кондратьева", "Василий Кондратьев", "Неизвестена").

**parents**("Иван Василенко", "Фёдор Василенко", "Неизвестена").

**parents**("Татьяна Харитонова", "Алексей Харитонов", "Галина Козловских").

**parents**("Александр Барабанщиков", "Евгений Барабанщиков", "Алла").

**parents**("Александр Ратенберг", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Мина Каганович", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Шейл Ратенберг", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Мера Ратенберг", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Мендель Ратенберг", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Сима Ратенберг", "Прапрадед Ратенберг", "Прапрабабушка").

**parents**("Семён Ратенберг", "Александр Ратенберг", "Этебейле").

**parents**("Мария Ратенберг", "Александр Ратенберг", "Этебейле").

**parents**("Борис Ратенберг", "Александр Ратенберг", "Этебейле").

**parents**("Лев Ратенберг", "Александр Ратенберг", "Этебейле").

**parents**("Ляля Гутман", "Виталий Гутман", "Мария Ратенберг").

**parents**("Виталий Гутман", "Моисей Гутман", "Неизвестена").

**parents**("Нина Ратенберг", "Шейл Ратенберг", "Неизвестно").

**parents**("Мина Ратенберг", "Шейл Ратенберг", "Неизвестно").

**parents**("Эмма Ратенберг", "Шейл Ратенберг", "Неизвестно").

**parents**("Мендель Ратенберг", "Шейл Ратенберг", "Неизвестно").

**parents**("Рувим Хентов", "Арон Хентов", "Мера Ратенберг").

**parents**("Рая Хентова", "Арон Хентов", "Мера Ратенберг").

**parents**("Леонид Хентов", "Рувим Хентов", "Фаня Штейн").

**parents**("Михаил Хентов", "Рувим Хентов", "Фаня Штейн").

**parents**("Борис Хентов", "Михаил Хентов", "Светлана").

**parents**("Полина Хентова", "Михаил Хентов", "Светлана").

**parents**("Саша Штейн", "Джон Штейн", "Полина Хентова").

**parents**("Сидни Штейн", "Джон Штейн", "Полина Хентова").

**parents**("Лев Файнерман", "Семён Файнерман", "Рая Хентова").

**parents**("Семён Файнерман", "Лев Файнерман", "Наталья Брауде").

**parents**("Наталья Брауде", "Александр Брауде", "Неизвестена").

**parents**("Александр Брауде", "Исаак Брауде", "Неизвестена").

**parents**("Софья Вульфсон", "Соломон Вульфсон", "Сима Ратенберг").

**parents**("Виктор Керзон-Золин", "Марк Керзон", "Зоя Золина").

**parents**("Зоя Золина", "Михаил Золин", "Неизвестена").

**parents**("Марк Керзон", "Виктор Керзон", "Неизвестена").

Далее следует фрагмент из парсера

…

# Протокол работы программы

# Выводы

Данная курсовая работа безусловно очень интересная. Для меня особенно, потому что свое семейное дерево я составлял этим летом вместе со своей бабушкой долгими июльскими вечерами. И, как можно видеть по предикатам parents, получилось у нас довольно много родственников. Приятно работать с чем-то реальным, когда тебе близка информация, с которой работаешь.

Помимо этого, занимательно было подключить в работу С#, который сам писал программу на прологе! В общем, данная работа мне очень понравилась!

Работа показывает, как можно применять пролог на практике. С помощью него, довольно просто можно реализовать поиск по родословному дереву и установить связь между двумя произвольно выбранными людьми.

Не остались без внимания и знания, полученные в ходе курса «Логическое программирования». Как можно видеть, моя программа использует поиск в глубину для установки отношений между двумя родственниками.

И в заключение повторюсь: этот курсовой проект был мне очень интересен, получил большое удовольствие от работы над ним, спасибо преподавателям за такой подбор заданий!