Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Отчет по курсовому проэкту**

**по курсу «Логическое Программирование»**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Сошников Д.В. |
| Преподаватель: | Левинская М. А. |
| Оценка: |  |
| Дата: |  |

Москва

2019

**Введение.**

В результате выполнения данной курсовой работы я получу следующие навыки и знания: умение работать с языком Prolog, понимание разницы между императивным и логическим программированием.

**Задание.**

1.Создать родословное дерево своего рода на несколько поколений (3-4) назад в стандартном формате GEDCOM с использованием сервиса MyHeritage.com

2.Преобразовать файл в формате GEDCOM в набор утверждений на языке Prolog, используя следующее представление:

parents(потомок, отец, мать).

3.Реализовать предикат проверки/поиска Шурина.

4.Реализовать программу на языке Prolog, которая позволит определять степень родства двух произвольных индивидуумов в дереве.

**Получение родословного дерева.**

Для получения родословного дерева я зашла на сайт MyHeritage.com и ,используя инструкцию, которая там присутствует, построила дерево. В нем находится 10 человек.

**Конвертация родословного дерева.**

Программа на С++ читает файл .ged по строкам , каждую из каторых потом обрабатывает, используя структуры person и family. В итоге создается вектор семей который определенным образом выводится в файл.

person:

struct person {

std::string name;

std::string fam;

char sex;

std::string indi;

};

family:

struct family {

std::vector<person> all;

};

Обработка:

std::vector<family> familys;

std::vector<person> persons;

family family1;

person individ;

bool flag = false;

char help[4];

std::ifstream file("Family\_Tree.ged");

while (file) {

std::string str;

getline(file, str);

if ((str[0] == '0') && (str[2] == '@') && (str[3] == 'I') && (str[4] == '5')) {

for (size\_t i = 2; i < 11; ++i) {

individ.indi.push\_back(str[i]);

}

} else if ((str[0] == '2') && (str[2] == 'G') && (str[3] == 'I') && (str[4] == 'V')) {

int i = 7;

while (str[i] != '\r') {

individ.name.push\_back(str[i]);

++i;

}

} else if ((str[0] == '2') && (str[2] == 'S') && (str[3] == 'U') && (str[4] == 'R')) {

int i = 7;

while (str[i] != '\r') {

individ.fam.push\_back(str[i]);

++i;

}

} else if ((str[0] == '1') && (str[2] == 'S') && (str[3] == 'E') && (str[4] == 'X')) {

individ.sex = str[6];

person individ2;

individ2.indi = individ.indi;

individ2.name = individ.name;

individ2.fam = individ.fam;

individ2.sex = individ.sex;

persons.push\_back(individ2);

individ.name.erase();

individ.fam.erase();

individ.indi.erase();

} else if ((str[0] == '1') && (str[2] == 'H') && (str[3] == 'U') && (str[4] == 'S')) {

flag = true;

for (int i = 7; i < 16; ++i) {

individ.indi.push\_back(str[i]);

}

int s = 0;

while (persons[s].indi != individ.indi) {

++s;

}

person notfull;

notfull.fam = persons[s].fam;

notfull.name = persons[s].name;

notfull.indi = persons[s].indi;

notfull.sex = persons[s].sex;

individ.indi.erase();

family1.all.push\_back(notfull);

} else if ((str[0] == '1') && (str[2] == 'W') && (str[3] == 'I') && (str[4] == 'F')) {

for (int i = 7; i < 16; ++i) {

individ.indi.push\_back(str[i]);

}

int s = 0;

while (persons[s].indi != individ.indi) {

++s;

}

person notfull;

notfull.fam = persons[s].fam;

notfull.name = persons[s].name;

notfull.indi = persons[s].indi;

notfull.sex = persons[s].sex;

individ.indi.erase();

family1.all.push\_back(notfull);

} else if ((str[0] == '1') && (str[2] == 'C') && (str[3] == 'H') && (str[4] == 'I')) {

for (int j = 0; j < 4; ++j) {

help[j] = str[j + 2];

}

for (int i = 7; i < 16; ++i) {

individ.indi.push\_back(str[i]);

}

int s = 0;

while (persons[s].indi != individ.indi) {

++s;

}

person notfull;

notfull.fam = persons[s].fam;

notfull.name = persons[s].name;

notfull.indi = persons[s].indi;

notfull.sex = persons[s].sex;

individ.indi.erase();

family1.all.push\_back(notfull);

}

if ((strcmp(help, "CHILD")) && (out\_of\_family(str)) && (flag == true)) {

family family2;

family2.all = family1.all;

family1.all.clear();

familys.push\_back(family2);

for (int i = 0; i < 4; ++i) {

help[i] = '0';

}

flag = false;

}

}

Вывод файл:

std::ofstream out("prolog.pl");

bool flag2=false;

for (size\_t k = 0; k < familys.size(); ++k) {

for (size\_t i = 2; i < familys[k].all.size(); ++i) {

out << "parent( \"" << familys[k].all[i].name << " " << familys[k].all[i].fam <<

"\",\"" << familys[k].all[0].name << " " << familys[k].all[0].fam <<

"\",\"" << familys[k].all[1].name << " " << familys[k].all[1].fam <<

"\")" << "\n";

}

if(k==0){

for (size\_t i = 2; i < familys[k].all.size(); ++i) {

if((familys[k].all[i].indi==familys[1].all[0].indi)||(familys[k].all[i].indi==familys[1].all[1].indi)){

flag2= true;

}else if((familys[k].all[i].indi==familys[2].all[0].indi)||(familys[k].all[i].indi==familys[2].all[1].indi)){

flag2= true;

}

if(flag2== false){

if(familys[k].all[i].sex=='M'){

out << "parent(\"\",\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\",\"\")"<<"\n";

}else{

out << "parent(\"\",\"\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\")"<<"\n";

}

}

flag2= false;

}

}else if(k==1){

for (size\_t i = 2; i < familys[k].all.size(); ++i) {

if((familys[k].all[i].indi==familys[0].all[0].indi)||(familys[k].all[i].indi==familys[0].all[1].indi)){

flag2= true;

}else if((familys[k].all[i].indi==familys[2].all[0].indi)||(familys[k].all[i].indi==familys[2].all[1].indi)){

flag2= true;

}

if(flag2== false){

if(familys[k].all[i].sex=='M'){

out << "parent(\"\",\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\",\"\")"<<"\n";

}else{

out << "parent(\"\",\"\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\")"<<"\n";

}

}

flag2= false;

}

}else{

for (size\_t i = 2; i < familys[k].all.size(); ++i) {

if((familys[k].all[i].indi==familys[0].all[0].indi)||(familys[k].all[i].indi==familys[0].all[1].indi)){

flag2= true;

}else if((familys[k].all[i].indi==familys[1].all[0].indi)||(familys[k].all[i].indi==familys[1].all[1].indi)){

flag2= true;

}

if(flag2== false){

if(familys[k].all[i].sex=='M'){

out << "parent(\"\",\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\",\"\")"<<"\n";

}else{

out << "parent(\"\",\"\""<<familys[k].all[i].name <<" "<<familys[k].all[i].fam<<

"\")"<<"\n";

}

}

flag2= false;

}

}

out << "\n";

}

out.close();

**Предикат поиска родственника.**

При поиске Шурина программа проверяет следующие условия: переданный человек является женского пола, имеется ли у неё брат; и на их основе выводит ответ.

Факты:

parent("Alexey Suvorov","Sergey Suvorov","Valentina Tishenko").

parent("Andrey Suvorov","Sergey Suvorov","Valentina Tishenko").

parent("","Andrey Suvorov","").

parent("Yanina Kotilevskaya","Vladimir Kotelev","Nina Kotilevskaya").

parent("Sofya Suvorova","Alexey Suvorov","Yanina Kotilevskaya").

parent("Ksenia Suvorova","Alexey Suvorov","Yanina Kotilevskaya").

parent("Alexandr Suvorov","Alexey Suvorov","Yanina Kotilevskaya").

parent("","","Sofya Suvorova").

parent("","","Ksenia Suvorova").

parent("","Alexandr Suvorov","").

Поиск шурина:

shyrin(X):-

parent(\_,\_,X)->

(parent(X,Y,Z),

parent(X1,Y,Z),

parent(\_,X1,\_)->

write(X1)

;false)

;false.

Пример работы:

?-shyrin("Ksenia Suvorova"),nl.

Alexandr Suvorov

Yes.

?-shyrin("Alexandr Suvorov"),nl.

No.

**Определение степени родства.**

Программа проверяет, что ей надо искать и в соответствии с этим ищет человека или степень родства.

Факты:

parent("Alexey Suvorov","Sergey Suvorov","Valentina Tishenko").

parent("Andrey Suvorov","Sergey Suvorov","Valentina Tishenko").

parent("","Andrey Suvorov","").

parent("Yanina Kotilevskaya","Vladimir Kotelev","Nina Kotilevskaya").

parent("Sofya Suvorova","Alexey Suvorov","Yanina Kotilevskaya").

parent("Ksenia Suvorova","Alexey Suvorov","Yanina Kotilevskaya").

parent("Alexandr Suvorov","Alexey Suvorov","Yanina Kotilevskaya").

parent("","","Sofya Suvorova").

parent("","","Ksenia Suvorova").

parent("","Alexandr Suvorov","").

Предикаты:

brother(Z,X,X1):-

(Z ?= "brother")->

(parent(X,Y,Y1),

parent(X1,Y,Y1),

parent(\_,X1,\_),

X1 \= X).

brother\_1(Z,X,X1):-

parent(X,Y,Y1),

parent(X1,Y,Y1),

parent(\_,X1,\_),

X1 \= X,

Z=["brother","child-mother","child-father"].

sister(Z,X,X1):-

(Z ?= "sister")->

(parent(X,Y,Y1),

parent(X1,Y,Y1),

parent(\_,\_,X1),

X1 \= X).

sister\_1(Z,X,X1):-

parent(X,Y,Y1),

parent(X1,Y,Y1),

parent(\_,\_,X1),

X1 \= X,

Z=["sister","child-mother","child-father"].

mother(Z,X,X1):-

(Z ?= "mother")->

(parent(X,\_,X1)).

mother\_1(Z,X,X1):-

parent(X,\_,X1),

Z=["mother","mother-simbling"].

father(Z,X,X1):-

(Z ?= "father")->

(parent(X,X1,\_)).

father\_1(Z,X,X1):-

parent(X,X1,\_),

Z=["father","father-simbling"].

child(Z,X,X1):-

(Z ?= "child")->(

(parent(\_,\_,X))->

(parent(X1,\_,X));parent(X1,X,\_)).

child\_1(Z,X,X1):-

(parent(\_,\_,X))->

(parent(X1,\_,X),(Z="child";Z="husband-child"));parent(X1,X,\_),

Z=["child","wife-child"].

grand\_mother(Z,X,X1):-

(Z ?= "grand\_mother")->

(parent(X,Y,Y1),(parent(Y,\_,X1);parent(Y1,\_,X1))).

grand\_mother\_1(Z,X,X1):-

parent(X,Y,Y1),(parent(Y,\_,X1);parent(Y1,\_,X1)),

Z=["grand\_mother","mother-parent"].

grand\_father(Z,X,X1):-

(Z ?= "grand\_father")->

(parent(X,Y,Y1),(parent(Y,X1,\_);parent(Y1,X1,\_))).

grand\_father\_1(Z,X,X1):-

parent(X,Y,Y1),(parent(Y,X1,\_);parent(Y1,X1,\_)),

Z=["grand\_father","father-parent"].

uncle(Z,X,X1):-

(Z ?= "uncle")->

(parent(X,Y,Y1),

((parent(Y1,M,M1),parent(X1,M,M1),X1 \= Y1);

(parent(Y,M,M1),parent(X1,M,M1),X1 \= Y))).

uncle\_1(Z,X,X1):-

(Z ?= "")->

(parent(X,Y,Y1),

((parent(Y1,M,M1),parent(X1,M,M1),X1 \= Y1);

(parent(Y,M,M1),parent(X1,M,M1),X1 \= Y)),

Z=["uncle","simbling-parent"]).

niece(Z,X,X1):-

(Z ?= "niece")->

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,

((parent(X1,\_,M));(parent(X1,M,\_))),parent(\_,\_,X1)).

niece\_1(Z,X,X1):-

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,

((parent(X1,\_,M));(parent(X1,M,\_))),parent(\_,\_,X1)),

Z=["niece","simbling-daughter"].

nephew(Z,X,X1):-

(Z ?= "nephew")->

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,

((parent(X1,\_,M));(parent(X1,M,\_))),parent(\_,X1,\_)).

nephew\_1(Z,X,X1):-

parent(X,Y,Y1),parent(M,Y,Y1),M \= X,

((parent(X1,\_,M));(parent(X1,M,\_))),parent(\_,X1,\_),

Z=["nephew","simbling-son"].

sister\_in\_law(Z,X,X1):-

(Z ?= "sister\_in\_law")->

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,X1),!).

sister\_in\_law\_1(Z,X,X1):-

parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,X1),!,

Z=["sister\_in\_law","simbling-wife"].

brother\_in\_law(Z,X,X1):-

(Z ?= "brother\_in\_law")->

((parent(\_,X,Y);(parent(\_,Y,X))),parent(Y,M,M1),parent(X1,M,M1),

X1 \= Y,parent(\_,X1,\_),!).

brother\_in\_law\_1(Z,X,X1):-

(parent(\_,X,Y);(parent(\_,Y,X))),parent(Y,M,M1),parent(X1,M,M1),

X1 \= Y,parent(\_,X1,\_),!,Z=["brother\_in\_law","husband-simbling"].

mother\_in\_law(Z,X,X1):-

(Z ?= "mother\_in\_law")->

(((parent(\_,X,Y),!);(parent(\_,Y,X),!)),parent(Y,\_,X1)).

mother\_in\_law\_1(Z,X,X1):-

((parent(\_,X,Y),!);(parent(\_,Y,X),!)),parent(Y,\_,X1),

Z=["mother\_in\_law","partner-mother"].

father\_in\_law(Z,X,X1):-

(Z ?= "father\_in\_law")->

(((parent(\_,X,Y),!);(parent(\_,Y,X),!)),parent(Y,X1,\_)).

father\_in\_law\_1(Z,X,X1):-

((parent(\_,X,Y),!);(parent(\_,Y,X),!)),parent(Y,X1,\_),

Z=["father\_in\_law","partner-father"].

daughter\_in\_law(Z,X,X1):-

(Z ?= "daughter\_in\_law")->

(((parent(Y,X,\_));(parent(Y,\_,X))),parent(\_,Y,X1),!).

daughter\_in\_law\_1(Z,X,X1):-

((parent(Y,X,\_));(parent(Y,\_,X))),parent(\_,Y,X1),!,

Z=["daughter\_in\_law","child-wife"].

son\_in\_law(Z,X,X1):-

(Z ?= "son\_in\_law")->

(((parent(Y,X,\_));(parent(Y,\_,X))),parent(\_,X1,Y),!).

son\_in\_law\_1(Z,X,X1):-

((parent(Y,X,\_));(parent(Y,\_,X))),parent(\_,X1,Y),!,

Z=["son\_in\_law","child-husband"].

father\_of\_sister\_in\_law(Z,X,X1):-

(Z ?= "father\_of\_sister\_in\_law")->

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,M1),

parent(M1,X1,\_),!).

father\_of\_sister\_in\_law\_1(Z,X,X1):-

parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,M1),

parent(M1,X1,\_),Z="father\_of\_sister\_in\_law",!.

mother\_of\_sister\_in\_law(Z,X,X1):-

(Z ?= "mother\_of\_sister\_in\_law")->

(parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,M1),

parent(M1,\_,X1),!).

mother\_of\_sister\_in\_law\_1(Z,X,X1):-

parent(X,Y,Y1),parent(M,Y,Y1),M \= X,parent(\_,M,M1),

parent(M1,\_,X1),Z="mother\_of\_sister\_in\_law",!.

wife(Z,X,X1):-

(Z ?= "wife")->

(parent(\_,X,X1),!).

wife\_1(Z,X,X1):-

parent(\_,X,X1),!,

Z=["wife","child-mother"].

husband(Z,X,X1):-

(Z ?= "husband")->

(parent(\_,X1,X),!).

husband\_1(Z,X,X1):-

parent(\_,X1,X),!,

Z=["husband","child-father"].

matchmaker(Z,X,X1):-

(Z ?= "matchmaker")->

(((parent(Y,X,\_));(parent(Y,\_,X))),

((parent(\_,Y,Y1));(parent(\_,Y1,Y))),!,

((parent(Y1,X1,\_));(parent(Y1,\_,X1)))).

matchmaker\_1(Z,X,X1):-

((parent(Y,X,\_));(parent(Y,\_,X))),

((parent(\_,Y,Y1));(parent(\_,Y1,Y))),

((parent(Y1,X1,\_));(parent(Y1,\_,X1))),

Z=["matchmaker","child-partner\_parent"].

relative(X,Y,Z):-

(X ?= "")->(

brother\_1(X,Y,Z);sister\_1(X,Y,Z);mother\_1(X,Y,Z);

father\_1(X,Y,Z);child\_1(X,Y,Z);wife\_1(X,Y,Z);

husband\_1(X,Y,Z);grand\_mother\_1(X,Y,Z);grand\_father\_1(X,Y,Z);

uncle\_1(X,Y,Z);niece\_1(X,Y,Z);nephew\_1(X,Y,Z);

sister\_in\_law\_1(X,Y,Z);brother\_in\_law\_1(X,Y,Z);mother\_in\_law\_1(X,Y,Z);

father\_in\_law\_1(X,Y,Z);daughter\_in\_law\_1(X,Y,Z);

son\_in\_law\_1(X,Y,Z);father\_of\_sister\_in\_law\_1(X,Y,Z);

mother\_of\_sister\_in\_law\_1(X,Y,Z);matchmaker\_1(X,Y,Z)),!;

((Z ?= "")->

(brother(X,Y,Z),sister(X,Y,Z),mother(X,Y,Z),

father(X,Y,Z),child(X,Y,Z),wife(X,Y,Z),

husband(X,Y,Z),grand\_mother(X,Y,Z),grand\_father(X,Y,Z),

uncle(X,Y,Z),niece(X,Y,Z),nephew(X,Y,Z),

sister\_in\_law(X,Y,Z),brother\_in\_law(X,Y,Z),mother\_in\_law(X,Y,Z),

father\_in\_law(X,Y,Z),daughter\_in\_law(X,Y,Z),

son\_in\_law(X,Y,Z),father\_of\_sister\_in\_law(X,Y,Z),

mother\_of\_sister\_in\_law(X,Y,Z),matchmaker(X,Y,Z));

(brother(X,Z,Y),sister(X,Z,Y),mother(X,Z,Y),

father(X,Z,Y),child(X,Z,Y),wife(X,Z,Y),

husband(X,Z,Y),grand\_mother(X,Z,Y),grand\_father(X,Z,Y)),

uncle(X,Z,Y),niece(X,Z,Y),nephew(X,Z,Y),

sister\_in\_law(X,Z,Y),brother\_in\_law(X,Z,Y),mother\_in\_law(X,Z,Y),

father\_in\_law(X,Z,Y),daughter\_in\_law(X,Z,Y),

son\_in\_law(X,Z,Y),father\_of\_sister\_in\_law(X,Z,Y),

mother\_of\_sister\_in\_law(X,Z,Y),matchmaker(X,Z,Y))

.

get\_one([],\_).

get\_one([H|[]],H):-!.

get\_one([H|T],X):-

(X=H;get\_one(T,X)).

Пример работы:

?-relative(Z,"Andrey Suvorov","Alexey Suvorov"),get\_one(Z,X),write(X),nl.

brother

Yes.

child-mother

Yes.

child-father

Yes.

No.

?-relative("brother",Z,"Alexey Suvorov"),write(Z),nl.

Andrey Suvorov

Yes.

No.

**Выводы.**

При выполнении данного курсового проекта я приобрела навыки работы с языком Prolog, узнала о сайте MyHeritage.com, где можно построить родословное дерево, узнала как конвертировать файлы .get и определила разницу между императивным программированием и логическим программированием.