МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)   
  
  
Институт информатики и кибернетики  
Кафедра программных систем  
  
Дисциплина  
**Логическое программирование  
  
  
  
ОТЧЕТ**по лабораторной работе №3  
Вариант №12

Студент: Мананников М.А.,   
Группа: 6303-020302D  
  
Преподаватель: Лобанков А.А.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2024

**Теоретические сведения**

http\_handler(Path, Closure, Options):

Регистрация пути Closure в качестве обработчика HTTP-запросов. Path - это абсолютный или относительный путь.

dynamic:

Информирует о том, что определение предиката(-ов) может измениться во время выполнения (используя assert/1 и/или retract/1). Динамические предикаты можно превратить в статические, используя compile\_predicates/1.

assert/1:

Добавление предложения (факта или правила) в базу данных. Предикат asserta/1 вставляет в первую позицию, а assertz/1 в последнюю. Устаревший assert/1 эквивалентен assertz/1.

findall/3:

Создает список экземпляров, которые удовлетворяют условию. Возвращает пустой список, если нет совпадений.

retract/1:

Удаление предложения (факта или правила) из базы данных.

http\_redirect/3:

Перенаправление на новое местоположение.

reply\_html\_page/3:

Возвращает HTML-страницу, которой предшествует HTTP-заголовок, как того требует библиотека (http\_wrapper).

**Задание**

Задание 1

Создать базу данных о заданной предметной области в виде множества фактов языка Пролог (не менее 5 фактов). Информацию о каждом компоненте БД представить в виде структуры.

Задание 2

Разработать набор предикатов, осуществляющих взаимодействие с БД, при помощи которых можно реализовать все статические запросы, приведенные в варианте задания.

Задание 3

Используя HTTP\_server\_libraries написать CRUD (Create/Read/Update/Delete) для своей базы фактов. Предоставить пользователю возможность сбросить базу данных к исходному (тестовому) варианту заполнения.

Задание 4

Реализовать вывод результатов всех поисковых запросов к базе фактов на html страницы.

Задание 5

Оформить отчет. Ответить на контрольные вопросы.

**Предметная область**

Предметная область – библиотека. Каждый автор может быть описан структурой из трех компонент: имени, фамилии и годом рождения. Каждая книга может быть описана структурой: названием, автором, издательством, числом экземпляров. Каждое издательство может быть описана структурой: названием, номером, годом издания, количеством страниц. Реализовать следующие типы запросов:

* Найти автора, у которого книга имеет самый ранний год издания;
* Подсчитать число изданий заданной книги;
* Найти всех книги, изданные в заданном издательстве за последние десять лет;
* Найти все книги заданного автора;
* Найти все книги цена, которых превышает заданную сумму.

**Примеры работы программы**

На рисунках 1-14 приведены экранные снимки работы программы.

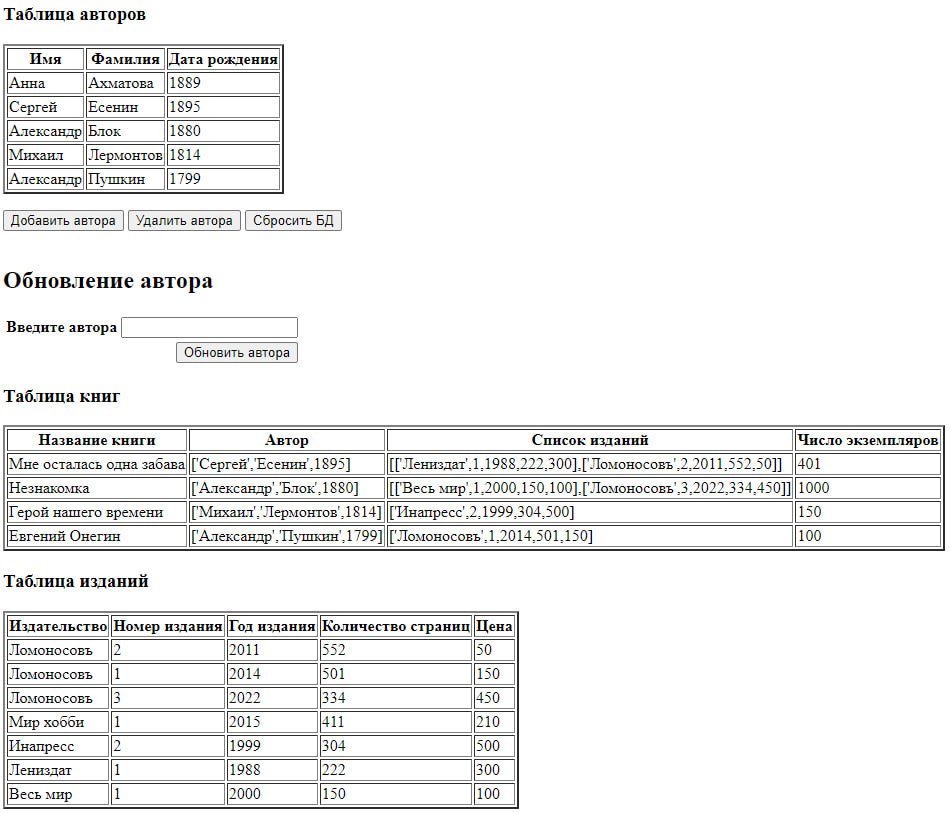
****

Рисунок 1 ­– Пример таблиц базы знаний ­­­

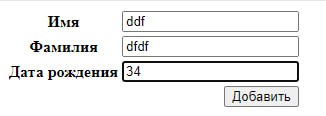


Рисунок 2 – Пример добавления автора



Рисунок 3 – Пример таблицы “Авторы” после добавления нового автора

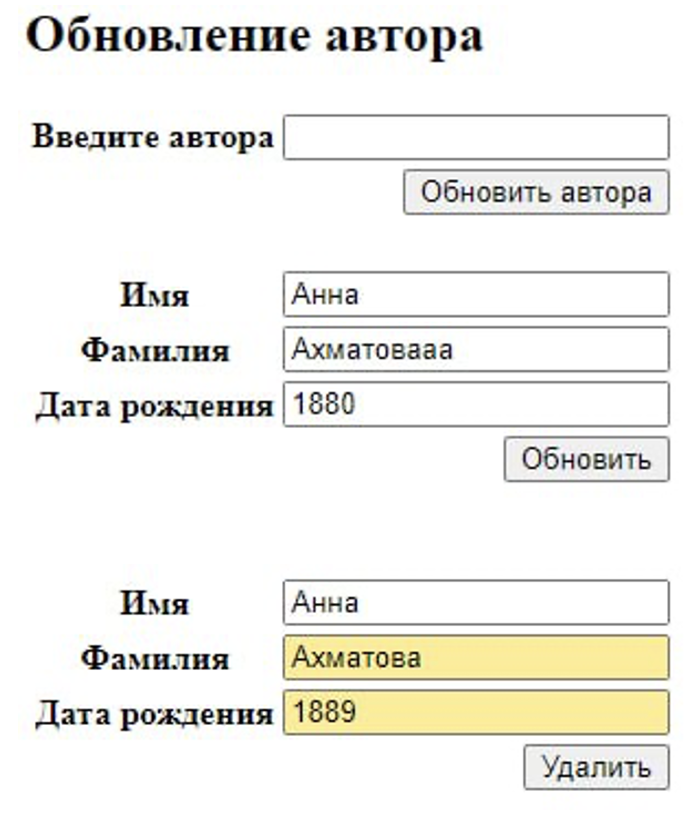


Рисунок 4 – Функционал для удаления и изменения элементов сущности “Авторы”

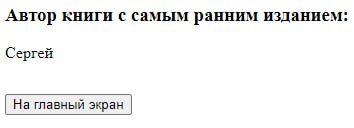


Рисунок 5 – Результат первого запроса



Рисунок 7 – Форма для выполнения второго запроса

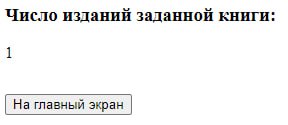


Рисунок 8 – Результат второго запроса



Рисунок 9 – Форма для выполнения третьего запроса

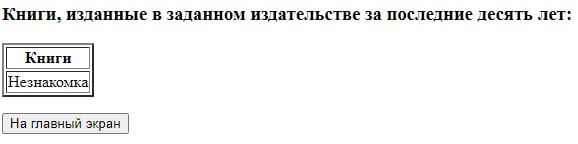


Рисунок 10 – Результат третьего запроса

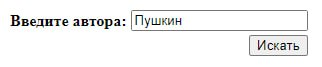


Рисунок 11 – Форма для выполнения четвертого запроса

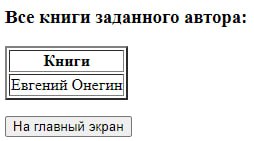


Рисунок 12 – Результат четвертого запроса



Рисунок 13 – Форма для выполнения пятого запроса

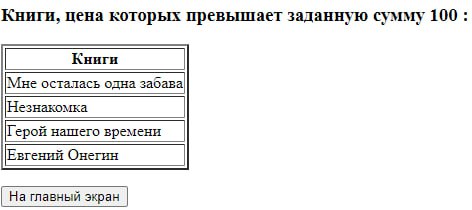


Рисунок 14 – Результат выполнения пятого запроса **Листинг программы**

:-use\_module(library(http/http\_server)).

:-use\_module(library(http/http\_dispatch)).

:-use\_module(library(http/http\_parameters)).

:-use\_module(library(http/html\_write)).

:-use\_module(library(http/http\_error)).

%библиотека для полного стектрейса ошибок

:- use\_module(library(http/http\_error)).

:- dynamic book/4.

:- dynamic author/3.

:- dynamic edition/5.

% Обработчик для корневого URL

:- http\_handler(root(.), home\_page, []).

%хэндлер страницы для добавления нового требования

:- http\_handler(root(add\_requirements\_page), add\_requirements\_page, []).

%хэндлер для добавления нового требования

:- http\_handler(root(add\_requirements), add\_requirements, [method(post)]).

%хэндлер страницы для удаления требования

:- http\_handler(root(delete\_requirements\_p), delete\_requirements\_p, []).

%хэндлер для удаления требования

:- http\_handler(root(delete\_requirements), delete\_requirements, [method(post)]).

%хэндлер для ресета БД

:- http\_handler(root(reset\_DB), reset\_DB, [method(post)]).

%хендлер страницы обновления бд

:- http\_handler(root(update\_db\_p), update\_db\_p, [method(post)]).

%хэндлер для удаления требования

:- http\_handler(root(update\_db), update\_db, [method(post)]).

% Определение хэндлеров для различных запросов

:- http\_handler(root(task\_1), task\_1\_page, []).

:- http\_handler(root(task\_2), task\_2\_page, []).

:- http\_handler(root(task\_2\_V), task\_2\_page\_V, []).

:- http\_handler(root(task\_3), task\_3\_page, []).

:- http\_handler(root(task\_3\_V), task\_3\_page\_V, []).

:- http\_handler(root(task\_4), task\_4\_page, []).

:- http\_handler(root(task\_4\_V), task\_4\_page\_V, []).

:- http\_handler(root(task\_5), task\_5\_page, []).

:- http\_handler(root(task\_5\_V), task\_5\_page\_V, []).

:- http\_handler(root(stop), stop, []).

% Запуск и остановка сервера

server(Port):- http\_server(http\_dispatch, [port(Port)]).

server:-server(8080).

stop(Port):- http\_stop\_server(Port, http\_dispatch).

stop:- stop(8080).

%инициализация базы фактов

reset\_DB(\_):-

retractall(book(\_, \_, \_, \_)),

retractall(author(\_, \_, \_)),

retractall(edition(\_, \_, \_, \_, \_)),

%инициализация БД

%Каждая книга может быть описана структурой: название, автор, список изданий, число экземпляров.

asserta(book('Евгений Онегин', ['Александр','Пушкин', 1799], ['Ломоносовъ', 1, 2014, 501, 150], 100)),

asserta(book('Герой нашего времени', ['Михаил','Лермонтов', 1814], ['Инапресс', 2, 1999, 304, 500], 150)),

asserta(book('Незнакомка', ['Александр','Блок', 1880], [['Весь мир', 1, 2000, 150, 100],['Ломоносовъ', 3, 2022, 334, 450]], 1000)),

asserta(book('Мне осталась одна забава', ['Сергей','Есенин', 1895], [['Лениздат', 1, 1988, 222, 300],['Ломоносовъ', 2, 2011, 552, 50]], 401)),

asserta(book('Реквием', ['Анна','Ахматова', 1889], ['Мир хобби', 1, 2015, 411, 210])),

%Автор может быть описан структурой: имя, фамилия, год рождения.

asserta(author('Александр','Пушкин', 1799)),

asserta(author('Михаил','Лермонтов', 1814)),

asserta(author('Александр','Блок', 1880)),

asserta(author('Сергей','Есенин', 1895)),

asserta(author('Анна','Ахматова', 1889)),

%Издание может быть описано структурой: издательство, номер издания, год издания, количество страниц, цена.

asserta(edition('Весь мир', 1, 2000, 150, 100)),

asserta(edition('Лениздат', 1, 1988, 222, 300)),

asserta(edition('Инапресс', 2, 1999, 304, 500)),

asserta(edition('Мир хобби', 1, 2015, 411, 210)),

asserta(edition('Ломоносовъ', 3, 2022, 334, 450)),

asserta(edition('Ломоносовъ', 1, 2014, 501, 150)),

asserta(edition('Ломоносовъ', 2, 2011, 552, 50)),

http\_redirect(moved, '/', \_Request).

%Обработка главной страницы

home\_page(\_Request):-

%Собираем столбцы в список

findall(NameBook, book(NameBook, \_, \_, \_), NameBooks),

findall(Author, book(\_, Author, \_, \_), Authors),

findall(ListEdition, book(\_, \_, ListEdition, \_), ListEditions),

findall(Kolvo, book(\_, \_, \_, Kolvo), Kolvos),

%Добавляем заголовки таблицы

generate\_rows1(NameBooks, Authors, ListEditions, Kolvos, Rows1),

ins(Rows1, tr(

[

th('Название книги'),

th('Автор'),

th('Список изданий'),

th('Число экземпляров')

]

), Rows\_with\_Headers1),

%Собираем столбцы в список

findall(NameAuthor, author(NameAuthor, \_, \_), NameAuthors),

findall(SurnameAuthor, author(\_, SurnameAuthor, \_), SurnameAuthors),

findall(DateofBirthday, author(\_, \_, DateofBirthday), DateofBirthdays),

%Добавляем заголовки таблицы

generate\_rows2(NameAuthors, SurnameAuthors, DateofBirthdays, Rows2),

ins(Rows2, tr(

[

th('Имя'),

th('Фамилия'),

th('Дата рождения')]

), Rows\_with\_Headers2),

%Собираем столбцы в список

findall(Company, edition(Company, \_, \_, \_, \_), Companys),

findall(NumberCompany, edition(\_, NumberCompany, \_, \_, \_), NumberCompanys),

findall(Dateofizdan, edition(\_, \_, Dateofizdan, \_, \_), Dateofizdans),

findall(Kolvostranic, edition(\_, \_, \_, Kolvostranic, \_), Kolvostranics),

findall(Cost, edition(\_, \_, \_, \_, Cost), Costs),

%Добавляем заголовки таблицы

generate\_rows3(Companys, NumberCompanys, Dateofizdans, Kolvostranics, Costs, Rows3),

ins(Rows3, tr(

[

th('Издательство'),

th('Номер издания'),

th('Год издания'),

th('Количество страниц'),

th('Цена')

]

), Rows\_with\_Headers3),

reply\_html\_page(

title('Мананников М 6303'),

[

h1('Лабораторная работа 3'),

h2('Библиотека'),

h2('Поисковые запросы:'),

ul([

li(a([href('/task\_1')], 'Найти автора, у которого книга имеет самый ранний год издания')),

li(a([href('/task\_2')], 'Подчитать число изданий заданной книги')),

li(a([href('/task\_3')], 'Найти все книги, изданные в заданном издательстве за последние десять лет')),

li(a([href('/task\_4')], 'Найти все книги заданного автора')),

li(a([href('/task\_5')], 'Найти все книги, цена которых превышает заданную сумму'))

]),

h3('Таблица авторов'),

table(

[border(2)],

Rows\_with\_Headers2

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/add\_requirements\_page')], 'Добавить автора'))

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/delete\_requirements\_p')], 'Удалить автора'))

),

form(

[style('display: inline-block'), method(post)],

p(button([type(submit), formaction(location\_by\_id('reset\_DB'))], 'Сбросить БД'))

),

form(

[action=location\_by\_id(update\_db\_p), method(post)],

[

h2('Обновление автора'),

table([

tr([th('Введите автора'), td(input([name(name)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Обновить автора'])))

])

]

),

h3('Таблица книг'),

table(

[border(2)],

Rows\_with\_Headers1

),

h3('Таблица изданий'),

table(

[border(2)],

Rows\_with\_Headers3

)

]

).

%Добавление элемента в начало списка

ins(L, El, [El|L]).

% Генерация строк таблицы для книг

generate\_rows1([], [], [], [], []).

generate\_rows1([NameBook|NameBooks], [Author|Authors], [ListEdition|ListEditions], [Kolvo|Kolvos], [tr([td(NameBook), td(AuthorStr), td(ListEditionStr), td(Kolvo)])|Rows1]):-

format(atom(AuthorStr), '~q', [Author]),

format(atom(ListEditionStr), '~q', [ListEdition]),

generate\_rows1(NameBooks, Authors, ListEditions, Kolvos, Rows1).

% Генерация строк таблицы для авторов

generate\_rows2([], [], [], []).

generate\_rows2([NameAuthor|NameAuthors], [SurnameAuthor|SurnameAuthors], [DateofBirthday|DateofBirthdays], [tr([td(NameAuthor), td(SurnameAuthor), td(DateofBirthday)])|Rows2]):-

generate\_rows2(NameAuthors, SurnameAuthors, DateofBirthdays, Rows2).

%Генерация строк таблицы для издательств

generate\_rows3([], [], [], [], [], []).

generate\_rows3([Company|Companys], [NumberCompany|NumberCompanys], [Dateofizdan|Dateofizdans], [Kolvostranic|Kolvostranics], [Cost|Costs], [tr([td(Company), td(NumberCompany), td(Dateofizdan), td(Kolvostranic), td(Cost)])|Rows3]):-

generate\_rows3(Companys, NumberCompanys, Dateofizdans, Kolvostranics, Costs, Rows3).

%Страница с добавлением нового автора

add\_requirements\_page(\_Request):-

reply\_html\_page(

title('Добавление автора'),

[form(

[action=location\_by\_id(add\_requirements), method(post)],

[

table([

tr([th('Имя'), td(input([name(nameauthor)]))]),

tr([th('Фамилия'), td(input([name(surnameauthor)]))]),

tr([th('Дата рождения'), td(input([name(dateofBirthday)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Добавить'])))

])

]

)]

)

.

%Добавление автора

add\_requirements(Request):-

http\_parameters(

Request,

[

nameauthor(Nameauthor, []),

surnameauthor(SurnameAuthor, []),

dateofBirthday(DateofBirthday, [])

]

),

atom\_number(DateofBirthday, DateofBirthdayNum),

assertz(author(Nameauthor, SurnameAuthor, DateofBirthdayNum)),

http\_redirect(moved, '/', Request).

%Страница с удалением автора

delete\_requirements\_p(\_Request):-

reply\_html\_page(

title('Удаление автора'),

[form(

[action=location\_by\_id(delete\_requirements), method(post)],

[

table([

tr([th('Имя'), td(input([name(nameauthor)]))]),

tr([th('Фамилия'), td(input([name(surnameauthor)]))]),

tr([th('Дата рождения'), td(input([name(dateofBirthday)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Удалить'])))

])

]

)]

).

%Удаление автора

delete\_requirements(Request):-

http\_parameters(

Request,

[

nameauthor(Nameauthor, []),

surnameauthor(Surnameauthor, []),

dateofBirthday(DateofBirthday, [])

]

),

atom\_number(DateofBirthday, DateofBirthdayNum),

retract(author(Nameauthor, Surnameauthor, DateofBirthdayNum)),

http\_redirect(moved, '/', Request).

%Страница обновления автора

update\_db\_p(Request):-

http\_parameters(

Request,

[

name(Name, [])

]

),

retract(author(Name,\_, \_)),

reply\_html\_page(

title('Обновление автора'),

[form(

[action=location\_by\_id(update\_db), method(post)],

[

table([

tr([th('Имя'), td(input([name(name), value(Name), readonly]))]),

tr([th('Фамилия'), td(input([name(surnameauthor)]))]),

tr([th('Дата рождения'), td(input([name(dateofBirthday)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Обновить'])))

])

]

)]

).

%Обновление автора

update\_db(Request):-

http\_parameters(

Request,

[

name(Name, []),

surnameauthor(Surnameauthor, []),

dateofBirthday(DateofBirthday, [])

]

),

atom\_number(DateofBirthday, DateofBirthdayNum),

assertz(author(Name, Surnameauthor, DateofBirthdayNum)),

http\_redirect(moved, '/', Request).

%Реализовать следующие типы запросов:

%1. Найти автора, у которого книга имеет самый ранний год издания

task\_1(Author) :-

findall([A, MinYear],

(book(\_, Authors, Editions, \_),

member(A, Authors),

member([\_, \_, Year|\_], Editions),

find\_min\_year(Editions, Year, MinYear)),

AuthorsYears),

find\_min\_year\_author(AuthorsYears, Author, MinYear),

!.

find\_min\_year\_author([[A, MinYear]|\_], A, MinYear).

find\_min\_year\_author([\_|Rest], Author, MinYear) :-

find\_min\_year\_author(Rest, Author, MinYear).

find\_min\_year([], MinYear, MinYear).

find\_min\_year([[\_, \_, Y|\_]|Rest], CurrentMin, MinYear) :-

Y < CurrentMin,

find\_min\_year(Rest, Y, MinYear).

find\_min\_year([[\_, \_, Y|\_]|Rest], CurrentMin, MinYear) :-

Y >= CurrentMin,

find\_min\_year(Rest, CurrentMin, MinYear).

task\_1\_page(\_Request):-

task\_1(Author),

reply\_html\_page(

title('Автор книги с самым ранним изданием'),

[h3('Автор книги с самым ранним изданием:'), p(Author),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/')], 'На главный экран'))

)]

).

%2. Подчитать число изданий заданной книги

task\_2(Book, Count) :-

book(Book, \_, Editions, \_),

length(Editions, RawCount),

(RawCount =:= 5 -> Count = 1 ; Count = RawCount).

task\_2\_page(\_Request):-

reply\_html\_page(

title('Введите книгу'),

[form(

[action=location\_by\_id(task\_2\_page\_V), method(post)],

[

table([

tr([th('Введите книгу:'), td(input([name(book)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Искать'])))

] )

]

)]

).

task\_2\_page\_V(Request):-

http\_parameters(

Request,

[book(B, [])]

),

task\_2(B, Count),

reply\_html\_page(

title('Число изданий заданной книги'),

[h3('Число изданий заданной книги:'), p(Count),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/')], 'На главный экран'))

)]

).

%3. Найти все книги, изданные в заданном издательстве за последние десять лет

generate\_rows1([], []).

generate\_rows1([Book|Books], [tr([td(Book)])|Rows\_]):- generate\_rows1(Books, Rows\_).

task\_3(Publisher, Books) :-

findall(Book,

(book(Book, \_, Editions, \_),

(

length(Editions, 5), % Если размер Editions равен 5

nth1(1, Edition, Pib),

nth1(3, Edition, Year)

;

member(Edition, Editions), % В остальных случаях

nth1(1, Edition, Pib),

nth1(3, Edition, Year)

),

Pib == Publisher,

Year > 2014),

Books).

task\_3\_page(\_Request):-

reply\_html\_page(

title('Введите издательство'),

[form(

[action=location\_by\_id(task\_3\_page\_V), method(post)],

[

table([

tr([th('Введите издательство:'), td(input([name(izdadelstvo)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Искать'])))

] )

]

)]

).

task\_3\_page\_V(Request):-

http\_parameters(

Request,

[izdadelstvo(I, [])]

),

task\_3(I, Books),

generate\_rows1(Books, Rows\_),

ins(Rows\_, tr(

[

th('Книги')

]

), Rows\_with\_Headers\_),

reply\_html\_page(

title('Книги, изданные в заданном издательстве за последние десять лет'),

[h3('Книги, изданные в заданном издательстве за последние десять лет:'),

table(

[border(2)],

Rows\_with\_Headers\_

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/')], 'На главный экран'))

)]

).

%4. Найти все книги заданного автора

task\_4(Author, Books) :-

findall(Book,

(book(Book, Authors, \_, \_),

member(Author, Authors)),

Books).

task\_4\_page(\_Request):-

reply\_html\_page(

title('Введите автора'),

[form(

[action=location\_by\_id(task\_4\_page\_V), method(post)],

[

table([

tr([th('Введите автора:'), td(input([name(avtor)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Искать'])))

] )

]

)]

).

task\_4\_page\_V(Request):-

http\_parameters(

Request,

[avtor(I, [])]

),

task\_4(I, Books),

generate\_rows1(Books, Rows\_),

ins(Rows\_, tr(

[

th('Книги')

]

), Rows\_with\_Headers\_),

reply\_html\_page(

title('Все книги заданного автора'),

[h3('Все книги заданного автора:'),

table(

[border(2)],

Rows\_with\_Headers\_

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/')], 'На главный экран'))

)]

).

%5. Найти все книги, цена которых превышает заданную сумму

task\_5(Price, Books) :-

findall(Book,

(book(Book, \_, Editions, \_),

(

length(Editions, 5), % Если размер Editions равен 5

nth1(5, Editions, Cost) % Выбираем пятый элемент

;

member(Edition, Editions), % В остальных случаях

nth1(5, Edition, Cost) % Выбираем пятый элемент

),

Cost > Price), % Сравниваем с Price

Books).

task\_5\_page(\_Request):-

reply\_html\_page(

title('Все книги, цена которых превышает заданную сумму'),

[form(

[action=location\_by\_id(task\_5\_page\_V), method(post)],

[

table([

tr([th('Введите цену:'), td(input([name(cena), type(number)]))]),

tr(td([colspan(2), align(right)], input([type=submit, value='Искать'])))

] )

]

)]

).

task\_5\_page\_V(Request):-

http\_parameters(

Request,

[cena(S, [])]

),

% Преобразование S в число

atom\_number(S, SNum),

task\_5(SNum, Books),

generate\_rows1(Books, Rows\_),

ins(Rows\_, tr(

[

th('Книги')

]

), Rows\_with\_Headers\_),

reply\_html\_page(

title('Книги, цена которых превышает заданную сумму ~w' - S),

[h3('Книги, цена которых превышает заданную сумму ~w :' - S),

table(

[border(2)],

Rows\_with\_Headers\_

),

form(

[style('display: inline-block')],

p(button([type(submit), formaction('/')], 'На главный экран'))

)]

).