Министерство науки и высшего образования Российской Федерации
Южно-Уральский государственный университет
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

004.4(07)
Д304

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Методические указания по выполнению курсовых работ

Челябинск
Издательский центр ЮУрГУ
2021

УДК 004.434(075.8)
Д304

Одобрено

учебно-методической комиссией
института естественных и точных наук

Рецензент
Е.М. Сартасов

Д304 **Функциональное и логическое программирование:** методические указания по выполнению курсовых работ / сост. А.К. Демидов. – Челябинск: Издательский центр ЮУрГУ, 2021. – 31 с.

В методических указаниях рассматриваются этапы разработки курсовой работы по дисциплине «Функциональное и логическое программирование», приводятся критерии оценки, требования к оформлению и содержанию пояснительной записки и презентации.

Методические указания содержат перечень тем для курсовых работ, рекомендации по реализации, тестированию и отладке программы, порядок защиты курсовой работы, структура доклада и презентации. В приложениях приводятся примеры оформления основных частей пояснительной записки.

Методические указания предназначены для студентов направления 01.03.02 «Прикладная математика и информатика».

УДК 004.434(075.8)

© Издательский центр ЮУрГУ, 2021

1 ЭТАПЫ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Курсовая работа по дисциплине «Функциональное и логическое программирование» представляет собой учебно-практическое исследование, предназначенное для систематизации, углубления и закрепления знаний, полученных студентом в процессе изучения конкретной дисциплины в соответствии с учебным планом.

Перечень этапов, распределение времени и баллов показаны в таблице 1.

Таблица 1 – Продолжительность и трудоемкость этапов

Наименование этапа	Продолжительность	Часы	Баллы	Результат работы
Постановка задачи	2 недели	3	5	Раздел 1 Постановка задачи
Проектирование и реализация программы	3 недели	12	15	Раздел 2 Проектирование и реализация, исходный код
Тестирование программы, улучшение и исправление ошибок	3 недели	6	10	Раздел 3 Тестирование программы, исполняемый код
Оформление программной документации и отчета по курсовой работе	2 недели	6	5	Пояснительная записка
Защита курсовой работы	2 недели	3	15	Презентация к докладу
ВСЕГО	12 недель	30	50	

2 ТЕМЫ ДЛЯ КУРСОВЫХ РАБОТ

Для курсовой работы предлагается три группы тем:

– применение функциональный подход при разработке обычных программ на языке Kotlin (ФП);

– применение логического программирования для решения специализированных задач на языке Prolog (ЛП);

– реализация экспертной системы (ЭС).

Функциональная парадигма в разработке программ становится все более популярной, все современные языки являются мультипарадигменными и позволяют использовать императивную, объектно-ориентированную и функциональную парадигму при разработке разных частей одной программы. На уровне реализации алгоритмов императивная парадигма всё больше вытесняется функциональной, и задачей курсовой работе по этой группе тем

является максимальное использование функционального подхода при написании кода. Функциональный подход позволяет упростить написание кода, уменьшить количество ошибок за счет большей понятности кода и использования неизменяемых структур данных. Язык Kotlin сразу разрабатывался как функциональный язык, поэтому его синтаксис помогает применять функциональную парадигму. Язык Kotlin в настоящее время является основным при разработке приложений для Android.

Две остальные группы тем демонстрируют преимущества логического программирования при решении специальных задач и для разработки экспертных систем с помощью специализированной среды CLIPS и оболочек для экспертных систем на языке Prolog. Логическое программирование является одним из основных видов декларативного программирования и широко используется при разработке систем искусственного интеллекта. Среда CLIPS показывает, как можно сочетать логический (представление знаний с помощью правил) и функциональный подходы.

В таблице 2 представлен перечень тем курсовых работ с краткими пояснениями.

Таблица 2 – Темы курсовых работ

№	Темы курсовых работ	Группа
1	Интерпретатор Scheme. Разработка интерпретатора Scheme на языке Kotlin	ФП
2	Библиотека XML. Разработка библиотеки для обработки и трансформации XML-документов на языке Kotlin	ФП
3	Функциональный подход при разработке веб-приложения. Разработка простого веб-приложения на языке Kotlin (например, крестики-нолики, memory, сапер)	ФП
4	Функциональный подход при разработке Android-приложения. Разработка простого Android-приложения на языке Kotlin (например, крестики-нолики, memory, сапер)	ФП
5	Функциональный подход при разработке серверного приложения. Разработка простого серверного приложения на языке Kotlin (например, чат)	ФП
6	Комбинаторная логика. Реализовать алгоритм перевода программы в комбинаторную форму и вычисление в комбинаторной логике на языке Prolog	ЛП
7	Мини-SQL. Реализовать подмножество команд SQL для создания и управления внутренней БД на языке Prolog (развитие задания 4 по ЛП)	ЛП
8	Решатель головоломок-1. Написать решатель судоку на языке Prolog	ЛП
9	Решатель головоломок-2. Составить из списка чисел арифметическое выражение, равное заданному числу, на языке Prolog	ЛП

Окончание таблицы 2

№	Темы курсовых работ	Группа
10	Проверка синтаксиса. Написать программу на языке Prolog для проверки синтаксиса для подмножества языка Basic	ЛП
11	Экспертная оболочка. Написать экспертную оболочку на языке Prolog, поддерживающую объяснения (как?, почему?), прямой и обратный вывод, нечеткую логику. Данные хранятся в виде объект-атрибут-значение.	ЭС
12	Экспертная оболочка для диагностики. Написать экспертную оболочку на языке Kotlin или Scheme, загружающую байесовские правила из текстового файла.	ЭС
13	Экспертная система для компьютерного магазина. Реализовать ЭС на языке Prolog или CLIPS, выдающую рекомендации при покупке компьютера в зависимости от используемых программ и назначения.	ЭС
14	Экспертная система для выбора подарка. Реализовать ЭС на языке Prolog или CLIPS, выдающую рекомендации при покупке подарка в зависимости от пола, возраста, характера, привычек, хобби, типа отношений между ними, типа торжества, финансов выделенные на подарок.	ЭС
15	Экспертная система для распознавания животных. Реализовать ЭС на языке Prolog или CLIPS, распознающую 20-30 видов животных.	ЭС
16	Экспертная система для распознавания пород собак (кошек). Реализовать ЭС на языке Prolog или CLIPS, распознающую 20-30 пород собак.	ЭС
17	Экспертная система для распознавания грибов (лекарственных растений). Реализовать ЭС на языке Prolog или CLIPS, распознающую 20-30 видов грибов (растений).	ЭС

3 ПОСТАНОВКА ЗАДАЧИ

На первом этапе разработки необходимо описать предметную область, решаемую задачу, все требования, допущения и ограничения, входные и выходные данные. Постановка задачи выполняется на естественном языке. При необходимости для пояснений используются графические изображения.

Целью постановки задачи является согласование с заказчиком (в курсовой работе – это руководитель) выполняемых программой функций, а также показать, что разработчик правильно понял задачу, учитывает все важные для заказчика требования и ограничения.

При разработке ЭС в постановке задачи должны быть перечислены все классифицируемые объекты (диагнозы). Также может быть выбрана оболочка ЭС. Так как оболочка реализована с помощью конкретного языка программирования, то выбор оболочки фиксирует выбор языка программирования.

Выбор оболочки ЭС определяется особенностями входных данных (их точность, возможное отсутствие информации), необходимость в нечетких выводах, промежуточных классах (диагнозах) или объяснений. В качестве оболочки ЭС можно в курсовой работе после необходимой доработки можно использовать:

– примеры оболочек из приложения М;

– среду CLIPS;

– оболочки из [1].

Примеры постановки задачи можно найти в приложениях И и К.

Критерии оценки этапа для БРС:

– сформулирована тема, используемый язык программирования – 2 балла;

– указаны все функциональные требования (выполняемые функции, входные и выходные данные, список диагнозов для ЭС) – 3 балла.

Итого максимум 5 баллов.

4 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Прежде чем выполнять какой-то алгоритм, нужно иметь объекты, к которым применяются эти операции. Для выполнения курсовой работы достаточно базовых структур данных, определенных в используемом языке (факты и структуры в языках Prolog и CLIPS, списки и объекты на формах и HTML-страницах в языке Kotlin).

При разработке программы на Kotlin необходимо разработать интерфейс, выделить объекты, задать им идентификаторы. В зависимости от типа приложения разработка программы может выполняться в IntelliJ IDEA Community или в Android Studio.

Так как функциональный подход особенно удобен при обработке последовательностей, то темы для курсовых работ предполагают работу с матрицей или последовательностью объектов в интерфейсе.

Если объекты размещаются в интерфейсе с помощью редактора форм, то рекомендуется давать им имена в формате “имя№”, чтобы упростить получение доступа к этим объектам из программы. Также рекомендуется использовать стили или CSS для установки повторяющихся свойств этих объектов.

```
val n=16
var btn=ArrayList<Button>()
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_screen)
  for(i in 1..n) {
    var id=resources.getIdentifier("btn"+i,"id",packageName)
    btn.add(findViewById<View>(id) as Button)
  }
}
```

Вместо создания объектов вручную можно создавать и размещать объекты в форме программно.

```
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_example)
  val layout=findViewById(R.id.btnLayout) as ConstraintLayout
  for(i in 1..n) {
    val b = Button(this)
    btn.add(b)
    b.layoutParams= ConstraintLayout.LayoutParams(
       ConstraintLayout.LayoutParams.WRAP_CONTENT,
       ConstraintLayout.LayoutParams.WRAP_CONTENT)
    b.text = "btn"+i
    b.setOnClickListener(View.OnClickListener {
        // действие при нажатии
    })
    layout.addView(b);
  }
}
```

При написании кода нужно рассматривать все циклы на предмет замены их функциями высшего порядка: map, mapIndexed, forEach, forEachIndexed, filter, filterIndexed, fold, foldRight, find, findLast, groupBy, dropWhile, dropLastWhile, takeWhile, takeLastWhile, all, any, count, partition, reduce, reduceIndexed.

Также рекомендуется рассмотреть возможность частичного применения функций. Для этого необходимо определить вспомогательный класс и сделать перегрузку операции вызова.

```
class Partial<T>
fun <T> partial(): Partial<T> = Partial()
// Для функций с 2 параметрами
operator @Suppress("UNUSED_PARAMETER")
fun <P1, P2, R> ((P1, P2) -> R).invoke(p1: P1,
       p2: Partial<P2> = partial()): (P2) -> R {
    return { p2: P2 -> this(p1, p2) }
}
// Для функций с 3 параметрами
operator @Suppress("UNUSED_PARAMETER")
fun <P1, P2, P3, R> ((P1, P2, P3) -> R).invoke(p1: P1,
      p2: Partial<P2> = partial(), p3: Partial<P3> = partial()):
   (P2, P3) -> R {return { p2: P2, p3: P3 -> this(p1, p2, p3) }
}
operator @Suppress("UNUSED_PARAMETER")
fun <P1, P2, P3, R> ((P1, P2, P3) -> R).invoke(p1: P1, p2: P2,
      p3: Partial<P3> = partial()): (P3) -> R {
    return { p3: P3 -> this(p1, p2, p3) }
}
```

```
val plus={a:Int,b:Int -> a+b} // функция с 2 параметрами
val plus5=plus(5) // подстановка 1-го параметра
plus5(15) // получаем функцию с 1 параметром
```

Для сложных последовательностей вычислений можно воспользоваться конвейером, использующим возможности перегрузки операций в Kotlin.

```
infix inline fun <P, R> P.pipe(t: (P) -> R): R = t(this)
2 pipe mul2 pipe plus5 // конвейер вместо plus5(mul2(2))
2 pipe mul(2) pipe plus(5) // то же с частичным применением
```

При выборе свойств объектов для классификации в ЭС рекомендуется выбирать в первую очередь те свойства, которые можно определить визуально или эмпирически, без точных измерительных инструментов. Например, для классификации химических элементов в качестве свойств можно рассмотреть физическое состояние при комнатной температуре (газ, жидкость или твердое тело) или цвет, но не температуру плавления, валентность или химическую формулу.

Правила ЭС должны содержать минимальный набор условий и не включать те свойства, которые не определяют различия между классифицируемыми объектами. Если существует сходство в свойствах у некоторых классифицируемых объектов, то для уменьшения правил можно выделить вспомогательный класс, в который поместить совпадающий набор свойств.

После формулировки правил на естественном языке их необходимо переписать в форме, подходящей для выбранной оболочки.

Результатом данного этапа является предварительная версия программы (альфа-версия). Проверяется в первую очередь понятность и качество кода, соответствие между реализованными функциями (правилами ЭС) и требованиями в постановке задачи.

Руководителем могут быть указаны пропущенные требования, которые не вошли в постановку задачи или не были точно специфицированы, но текущая их реализация делает интерфейс неудобным для использования. Также руководителем могут быть выявлены ошибки во внешнем поведении программы, не соответствующие поставленной задаче. Эти замечания не влияют на оценку за данный этап, но должны быть исправлены на следующем этапе.

Критерии оценки этапа для БРС:

– определены объекты интерфейса в языке Kotlin или выделены свойства для классификации (диагностики) в ЭС – 5 баллов;

– выполнена реализация основных функций на языке Kotlin или правил ЭС – 5 баллов;

– используются функции высшего порядка в языке Kotlin или набор условий в правилах является минимальным – 5 баллов.

Итого максимум 15 баллов.

5 ТЕСТИРОВАНИЕ ПРОГРАММЫ И ИСПРАВЛЕНИЕ ОШИБОК

На данном этапе сначала исправляются все ошибки, улучшается код и интерфейс программы по замечаниям руководителя.

Затем производится тестирование программы. Тестирование – это процесс исполнения программы с целью обнаружения ошибок. Для этого рассматриваются несколько сценариев работы пользователя, и вы должны получить все возможные результаты программы, все сообщения об ошибочных действиях пользователя, если такие сообщения предусмотрены в вашей программе.

Для ЭС процесс тестирования достаточно очевиден – необходимо проверить возможность получения всех классифицируемых объектов (диагнозов).

Если поведение программы не соответствует ожидаемым результатам, то выполняется процесс отладки – поиск части кода с ошибкой и её исправление.

Критерии оценки этапа для БРС:

– исправлены все замечания по коду, сделанные руководителем на предыдущем и на этом этапе – 5 баллов;

– проверка внешнего поведения программы не выявило ошибок – 5 баллов.

Итого максимум 10 баллов.

6 ОФОРМЛЕНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Параметры страницы: верхнее поле – 20 мм, нижнее поле – 20 мм, левое поле – 25 мм, правое поле – 10 мм. Нумерация страниц работы должна быть сквозной. Первой страницей считается титульный лист. Титульный лист, задание, аннотацию и оглавление включают в общую нумерацию страниц работы, но номера страниц на этих листах не проставляют. Номера страниц размещают в нижнем колонтитуле с выравниванием номера посередине.

Основной текст курсовой работы должен быть набран шрифтом Times New Roman размером 14 пт с одинарным межстрочным интервалом. Отступ первой строки абзаца – 0.7 см. В тексте не допускаются висячие строки (неполные строки в начале страницы). Для листингов программ используется шрифт Courier New, отступ первой строки абзаца – 0 см. Если количество строк кода не превышает пяти, то используется размер шрифта 14 пт, более пяти строк – 12.5 пт (чтобы получалось примерно 50 строк на страницу).

Опечатки, описки и графические неточности, обнаруженные в процессе подготовки работы, а также в результате проверки её руководителем допускается исправлять подчисткой или закрашиванием корректирующим карандашом или лентой, с последующим нанесением на том же месте исправлений, близких к компьютерному формату, шариковой или гелиевой ручками черного цвета.

Заголовки структурных элементов (частей) пояснительной записки печатают прописными буквами посередине, а заголовки разделов – прописными буквами с абзацного отступа. На разделы можно делить только основной материал и приложения. Разделы должны иметь порядковую нумерацию 1, 2, 3 и т. д.

Подразделы должны иметь нумерацию в пределах каждого раздела. Номер подраздела включает номер раздела, например, 1.1, 1.2 и т. д. Заголовки подразделов печатают с абзацного отступа с прописной буквы. После номера раздела и подраздела в их названии точка не ставятся. После заголовка раздела в конце страницы должно размещаться не менее трех строк текста.

Все иллюстрации в работе (эскизы, схемы, графики, фотографии) называются рисунками. Название рисунка состоит из слова «Рисунок», его номера (используется сквозная нумерация рисунков в пределах всей работы) и наименования, отделенного символами «пробел» и тире от цифрового обозначения. Наименование пишется под рисунком посредине строки и может включать расшифровку обозначений, использованных в рисунке. На все рисунки в тексте работы должны быть ссылки. Первая ссылка имеет вид «рисунок 1» или «… в соответствии с рисунком 1», а все последующие ссылки на этот рисунок должны иметь вид – «см. рисунок 1».

Таблицы нумеруют арабскими цифрами. Слово «Таблица» и ее номер помешают слева над таблицей без абзацного отступа, например, «Таблица 1». После номера можно написать наименование таблицы, отделяя его символами «пробел» и тире от цифрового обозначения. Если таблица не помещается на странице, её делят на части. Слово «Таблица», ее номер и заголовок (при его наличии) указывают над первой частью таблицы. Над другими частями пишут слова, например, «Продолжение таблицы 1» с указанием ее номера, а на последней странице – «Окончание таблицы 1». Все продолжения и окончание таблицы начинаются с повторения шапки таблицы. Рисунки и таблицы располагаются вслед за первым упоминанием о них в тексте.

Каждое приложение следует начинать с новой страницы. Наверху посередине страницы указывается слово «ПРИЛОЖЕНИЕ» прописными буквами и дается его обозначение заглавными буквами русского алфавита, начиная с А. Строкой ниже записывается тематический заголовок приложения с прописной буквы. Текст каждого приложения, при необходимости, может быть разделен на разделы, которые нумеруют в пределах каждого приложения (А.1, А.2 и т.д.). Аналогично иллюстрации и таблицы в приложениях нумеруют в пределах каждого приложения, например – Рисунок А.3, Таблица Д.2.

7 СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Структурными элементами (частями) курсовой работы являются [2]:

– титульный лист;
– задание на работу;
– аннотация;
– оглавление;
– введение;
– обзор литературы и постановка задачи;
– основной материал по специальной части;
– заключение;

– ***библиографический список;***

– приложения.

Обязательные структурные элементы выделены полужирным шрифтом. Пример ***титульного листа*** приведен в приложении А. Пример ***задания на работу*** приведен в приложении Б.

Аннотацию помещают в пояснительной записке после задания. Аннотация включает:

– характеристику основной темы;

– проблемы объекта (если есть);

– цели (и задачи) работы;

– результаты работы.

Пример оформления аннотации показан в приложении В.

Оглавление состоит из перечня частей, разделов и подразделов работы и включает: введение, наименование всех разделов и подразделов основного материала, заключение, библиографический список и наименование приложений, для каждого из которых указываются номер страниц, с которых начинаются эти элементы курсовой работы. От конца текста до номера страницы дается отточие.

Пример оформления оглавления показан в приложении Г.

Во ***введении*** должна быть раскрыта актуальность темы курсовой работы, приведены цель и задачи работы, объект и предмет работы, а также показана практическая применимость полученных автором результатов. Шаблон введения показан в приложении Д.

Заключение должно содержать:

– краткие выводы по результатам выполнения курсовой работы и оценку полноты решений поставленных в работе задач и достижения цели работы;

– рекомендации по конкретному использованию результатов курсовой работы;

– оценку результативности или эффективности предлагаемого решения.

Пример оформления заключения показан в приложении Е.

Пример оформления библиографического списка показан в приложении Ж.

В ***основной части*** курсовой работы приводятся данные, отражающие сущность, методику и основные результаты выполненной работы. Основная часть должна содержать следующие разделы:

– постановка задачи;

– проектирование и реализация;

– тестирование программы.

Постановка задачи включает краткое описание задачи.

Проектирование и реализация должно содержать:

– пояснения по структурам данных (в том числе объектам интерфейса, используемым в программе), функциям (предикатам);

– примеры функционального подхода или реализации правил ЭС.

Тестирование программы должно содержать протокол взаимодействия с программой для одного примера. В случае разработки приложений на Kotlin нужно привести скриншоты, подтверждающие, что указанные в постановке задачи функции реализованы и выполняются без ошибок.

Примеры оформления основной части показаны в приложениях И и К.

В ***приложения*** включаются материалы, связанные с выполненной курсовой работой, которые по каким-либо причинам не были включены в основную часть. В приложение А включаются исходные тексты программы, разделенные на подразделы по файлам. Пример оформления приложения показан в приложении Л.

На данном этапе оценивается только оформление пояснительной записки, так как постановка задачи и разработка алгоритма были уже оценены на предыдущих этапах. При грубых нарушения правил форматирования, изложенных в разделе 7, пояснительная записка возвращается на доработку.

Критерии оценки этапа для БРС:

– пояснительная записка содержит все разделы и части – 2 балла;

– пояснительная записка оформлена в соответствии с шаблоном и правилами форматирования – 2 балла;

– нет орфографических и синтаксических ошибок – 1 балл.

Итого максимум 5 баллов.

8 ЗАЩИТА КУРСОВОЙ РАБОТЫ

В последнюю неделю семестра проводится защита курсовой работы. Защита проводится перед комиссией, состоящей не менее чем из двух преподавателей.

На защиту студент предоставляет:

1. Задание на курсовую работу (оформляется в начале семестра).
2. Программный продукт.
3. Пояснительную записку, содержащую описание разработки, программную документацию и соответствующие иллюстрации. В приложении приводится исходный код программы.

На защите студент коротко (3–5 минут) докладывает об основных проектных решениях, принятых в процессе разработки, и отвечает на вопросы комиссии.

План презентации:

– титульный слайд;

– постановка задачи – 1 слайд (сократить при необходимости);

– проектирование (интерфейс для приложений или набор свойства и пример правила для ЭС) – 2-3 слайда;

– реализация (примеры функционального подхода в приложениях на Kotlin или правил на Prolog) – 1-3 слайда;

– тестирование (скриншоты результатов работы программы) – 1–3 слайда;

– заключение – 1 слайд.

Каждый слайд, кроме титульного, должен иметь номер в правом нижнем углу.

Оценка «отлично» выставляется за курсовую работу, которая полностью соответствует техническому заданию, работоспособна во всех режимах, пояснительная записка имеет логичное, последовательное изложение материала с соответствующими выводами и обоснованными положениями. При защите студент показывает глубокое знание вопросов темы, свободно оперирует данными разработки, легко отвечает на поставленные вопросы.

Оценка «хорошо» выставляется за курсовую работу, которая полностью соответствует техническому заданию, работоспособна в большинстве режимов, пояснительная записка имеет грамотное, последовательное изложение материала с соответствующими выводами, однако с не вполне обоснованными положениями. При защите студент показывает знание вопросов темы, свободно оперирует данными разработки, без особых затруднений отвечает на поставленные вопросы.

Оценка «удовлетворительно» выставляется за курсовую работу, которая не полностью соответствует техническому заданию, работоспособна только в части режимов, пояснительная записка базируется на теоретическом материале, но имеет поверхностный анализ, есть непоследовательность в изложении материала, представлены необоснованные положения. При защите студент показывает слабое знание вопросов темы, не всегда дает аргументированные ответы на поставленные вопросы.

Оценка «неудовлетворительно» выставляется за курсовую работу, которая не соответствует техническому заданию, не работоспособна, пояснительная записка не отвечает требованиям, изложенным в методических рекомендациях, не содержит выводов и обоснованных положений. При защите студент показывает слабое знание вопросов темы, неуверенность в оперировании данными разработки, не дает аргументированные ответы на вопросы.

Критерии оценки этапа для БРС:

– подготовлена презентация – 3 баллов;

– в устном докладе студент показывает знания о проектных решениях в курсовой работе, свободно оперирует терминами применительно к рассматриваемой задаче – 5 баллов;

– студент может ответить на вопросы членов комиссии – 7 баллов.

Итого максимум 15 баллов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Merritt D. Building Expert Systems in Prolog. [Электронный ресурс] / Amzi! inc., 2000. – 308 с. – Режим доступа: http://www.inf.fu-berlin.de/lehre/SS09/KI /folien/merritt.pdf (дата обращения: 01.04.2021).

2 СТО ЮУрГУ 21–2008 Стандарт организации. Система управления качеством образовательных процессов. Курсовая и выпускная квалификационная работа. Требования к содержанию и оформлению / составители: Т.И. Парубочая, Н.В. Сырейщикова, А.Е. Шевелев, Е.В. Шевелева. – Челябинск: Изд-во ЮУрГУ, 2008. – 55 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

ПРИМЕР ТИТУЛЬНОГО ЛИСТА КУРСОВОЙ РАБОТЫ

Министерство науки и высшего образования РФ
ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)
Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

«Экспертная система для классификации животных»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине «Функциональное и логическое программирование»

ЮУрГУ–01.03.02.2021.313.ПЗ КР

Руководитель,

_____Демидов А.К.

«___» _____ 2021 г.

Автор работы:

Студент группы: ЕТ – 412

_____ Иванов И.И.

«___» _____ 2021 г.

Работа защищена с оценкой

_____

«___» _____2021 г.

Челябинск – 2021

ПРИЛОЖЕНИЕ Б
ПРИМЕР ЗАДАНИЯ НА КУРСОВУЮ РАБОТУ

Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Прикладная математика и программирование»
Направление Прикладная математика и информатика

УТВЕРЖДАЮ
Заведующий кафедрой ПМиП
_____А.А.Замышляева
_____ 2021 г.

ЗАДАНИЕ
на курсовую работу студента
_____*Иванова И.И.*_____
Группа __*ЕТ-412*____

1. Дисциплина _____*Функциональное и логическое программирование*_____
2. Тема работы ____*Экспертная система для классификации животных*_____
_____
3. Срок сдачи студентом законченной работы *11 мая 2021 г.*
4. Перечень вопросов, подлежащих разработке
1) разработка правил экспертной системы (ЭС);
2) реализация ЭС на языке Prolog;
3) тестирование ЭС и исправление ошибок;
4) оформление отчета по курсовой работе;
5) презентация проектных решений для защиты КР

5. Календарный план

Наименование разделов (этапов) курсовой работы	**Срок выполнения разделов (этапов) работы**	**Отметка о выполнении руководителя**
постановка задачи	01.03.2021-15.03.2021	
проектирование и реализация	16.03.2021-31.03.2021	
тестирование	01.04.2021-15.04.2021	
оформление отчета по курсовой работе	16.04.2021-10.05.2021	
защита курсовой работы	11.05.2021-20.05.2021	

Руководитель работы _____ /_____
Студент _____ /_____
(подпись) (расшифровка)

ПРИЛОЖЕНИЕ В
ПРИМЕР ОФОРМЛЕНИЯ АННОТАЦИИ НА КУРСОВУЮ РАБОТУ

АННОТАЦИЯ

Иванова И.И. Экспертная система для классификации животных. – Челябинск: ЮУрГУ, ЕТ-412, 2021. – 13с., 1 ил., библиографический список – 3 наим., 1 прил.

В курсовой работе описывается разработка экспертной системы с помощью логического программирования. Работа содержит результаты анализа предметной области, определения требований к ЭС, проектирования, реализации и тестирования ЭС.

В результате работы была разработана экспертная система, код которой приводится в приложении.

ПРИЛОЖЕНИЕ Г
ПРИМЕР ОГЛАВЛЕНИЯ

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
1 ПОСТАНОВКА ЗАДАЧИ 6
2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 7
3 ТЕСТИРОВАНИЕ ПРОГРАММЫ 11
ЗАКЛЮЧЕНИЕ 14
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 15
ПРИЛОЖЕНИЕ А 16

ПРИЛОЖЕНИЕ Д
ПРИМЕРЫ ОФОРМЛЕНИЯ ВВЕДЕНИЯ

Д.1 Пример 1

ВВЕДЕНИЕ

Актуальность темы. Логическое программирование является одним из основных видов декларативного программирования и широко используется при разработке систем искусственного интеллекта.
Цель работы – разработать экспертную систему для классификации животных.
Задачи работы:
– изучить представление знаний в задачах искусственного интеллекта;
– научиться определять с помощью языка Prolog [1] правила вывода для экспертных систем;
– овладеть методами поиска решения в системах искусственного интеллекта;
– научиться разрабатывать программные средства для систем искусственного интеллекта.
Объект работы – классификация животных
Предмет работы – применение логического программирования и языка программирования Prolog для разработки экспертной системы [2].
Результаты работы можно использовать при создании моделей объектов, определяемых с помощью правила поведения, при решении задач классификации по тематике проводимых научно-исследовательских проектов.

Д.2 Пример 2

ВВЕДЕНИЕ

Актуальность темы. Функциональная парадигма [1] в разработке программ становится все более популярной, все современные языки являются мультипарадигменными и позволяют использовать императивную, объектно-ориентированную и функциональную парадигму при разработке разных частей одной программы. На уровне реализации алгоритмов императивная парадигма всё больше вытесняется функциональной. Функциональный подход позволяет упростить написание кода, уменьшить количество ошибок за счет большей понятности кода и использования неизменяемых структур данных.

Язык Kotlin [4] сразу разрабатывался как функциональный язык, поэтому его синтаксис помогает применять функциональную парадигму. Kotlin может компилировать код в Java Virtual Machine (JVM) и JavaScript, что позволяет добиться высокой производительности и кроссплатформенности скомпилированных программ. Язык Kotlin в настоящее время является основным при разработке приложений для Android.

Цель работы – разработать программу для моделирования подбора кода
Задачи работы:
– изучить синтаксис описания программ в функциональном стиле на языке Kotlin;
– научиться применять функции высшего порядка для анализа и обработки данных;
– научиться использовать в работе концепции функционального программирования.
Объект работы – браузерное приложение для моделирования подбора кода
Предмет работы – применение функциональной парадигмы и языка программирования Kotlin для разработки программы.
Результаты работы можно использовать при разработке алгоритмических и программных решений системного и прикладного программного обеспечения.

ПРИЛОЖЕНИЕ Е
ПРИМЕРЫ ОФОРМЛЕНИЯ ЗАКЛЮЧЕНИЯ РАБОТЫ

Е.1 Пример 1

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выявлены значащие свойства объектов для их классификации и определены правила с использованием. Для интерпретации правил были разработаны вспомогательные предикаты на языке программирования Prolog. Разработанная экспертная система была проверена на контрольных тестах и в код были внесены необходимые исправления. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать для исследования и разработки математических, информационных и имитационных моделей, представления знаний в интеллектуальных системах по тематике проводимых научно-исследовательских проектов.

Е.2 Пример 2

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был определен интерфейс пользователя для приложения, разработан код html-страницы и css-таблицы для элементов интерфейса. Было выполнено проектирование и реализация приложения на языке Kotlin. Разработанное приложение было проверено на контрольных тестах и в код были внесены необходимые исправления. Таким образом, цель работы была достигнута, задачи – решены.

Производительность программ, написанных на Kotlin, почти на порядок лучше чем у программ, написанных на Python, то есть Kotlin лучше подходит обработки данных сложной структуры. При этом исходный код остается достаточно лаконичным и наглядным за счет функционального подхода. За счет статической типизации многие ошибки могут выявлены на этапе компиляции или во время написания кода. Пока у Python есть преимущество перед Kotlin только в большем количестве библиотек для математической обработки данных.

Результаты работы можно использовать при разработке системного и прикладного программного обеспечения для анализа и обработки данных по тематике проводимых научно-исследовательских проектов.

ПРИЛОЖЕНИЕ Ж
ПРИМЕР ОФОРМЛЕНИЯ БИБЛИОГРАФИЧЕСКОГО СПИСКА

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Демидов, А. К. Функциональное и логическое программирование: учебное пособие / А. К. Демидов – Челябинск: Издательство ЮУрГУ, 2000. - 58 с.
2 Merritt D. Building Expert Systems in Prolog. [Электронный ресурс] / Amzi! inc., 2000. – 308 с. – Режим доступа: http://www.inf.fu-berlin.de/lehre/SS09/KI/folien/merritt.pdf (дата обращения: 01.04.2021).
3 Смолин, Д. В. Введение в искусственный интеллект: конспект лекций : учебное пособие / Д. В. Смолин. – 2-е изд., перераб. – Москва : ФИЗМАТЛИТ, 2007. – 264 с. – URL: https://e.lanbook.com/book/2325 (дата обращения: 01.04.2021).
4 Reference – Kotlin Programming Language. – Режим доступа: https://kotlinlang.org/docs/reference/ (дата обращения: 01.04.2021).
5 Жемеров, Д. Kotlin в действии / Д. Жемеров, С. Исакова ; перевод с английского А. Н. Киселев. – Москва : ДМК Пресс, 2018. – 402 с. – URL: https://e.lanbook.com/book/112926 (дата обращения: 01.04.2021).

ПРИЛОЖЕНИЕ И
ПРИМЕР ОСНОВНОЙ ЧАСТИ ДЛЯ ЭКСПЕРТНОЙ СИСТЕМЫ

1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать экспертную систему для классификации следующих животных: гепард, тигр, жираф, зебра, страус, пингвин, буревестник.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Для распознавания выберем следующие свойства: цвет, внешние признаки (наличие копыт, шерсти или перьев, длина ног или шеи), особенности окраски (пятна или полосы). Эти свойства не связаны с поведением или научной классификацией животных (класс-род-семейство) и могут быть определены по фотографии или по результатам наблюдения в природе. Также в экспертной системе предусмотрен вывод и в случае отсутствия информации о некоторых свойствах.

Некоторых животных можно сгруппировать в классы, имеющие общие свойства (птица, млекопитающее и т.д.). Для таких классов введем вспомогательные правила, чтобы не повторять проверку таких общих свойств в каждом правиле.

Например, для гепарда можно сформулировать следующее правило: «Гепард является <млекопитающим>, <хищником>, имеет желто-коричневый цвет и темные пятна».

Для продукционных систем используется несколько стратегий поиска [3]:

– Стратегия глубины. Только что активизированное правило помещается поверх всех правил с таким же приоритетом. Недостаток: опрос пользователя будет выполняться в порядке перечисления правил в ЭС, если правила определены в неудачном порядке, то опрос будет выглядеть хаотичным.

– Стратегия ширины. Только что активизированное правило помещается ниже всех правил с таким же приоритетом. Недостаток: возможны ситуации, когда сначала будут запрошены все факты (даже если в них нет необходимости), а потом сделан вывод.

– LEX стратегия. Активация правила, выполненная более новыми фактами, располагается перед активацией, осуществленной более поздними фактами. Обеспечивает последовательное движение к некоторой гипотезе (или группе), пока она не будет опровергнута.

– MEA стратегия. Осуществляется упорядочение правил по первым фактам. Улучшает стратегию глубины, проверка гипотез будет упорядочена.

– Стратегия упрощения. Среди всех правил с одинаковым приоритетом только что активизированное правило располагается выше всех правил с равной или большей определенностью. Определенность правила задается количеством условий в левой части правил. Более простые правила могут

быстрее привести к результату, меньше фактов нужно запросить у пользователя.

– Стратегия усложнения. Среди всех правил с одинаковым приоритетом только что активизированное правило располагается выше всех правил с равной или большей определенностью.

– Случайная стратегия. Каждой активации назначается случайное число, которое используется для определения местоположения среди активаций с определенным приоритетом.

Будем использовать стратегию поиска в глубину, несмотря на указанный выше недостаток. Так как эта стратегия используется для вывода в языке Prolog, то можно не писать собственный интерпретатор правил и машину вывода, а воспользоваться интерпретатором языка Prolog. Это обеспечить высокую скорость выполнения правил, так как они будут скомпилированы.

Для этого нужно переписать все разработанные правила на язык Prolog. Правило для гепарда на языке Prolog будет выглядеть так:

```
animal_is(гепард):-
    it_is(млекопитающее),
    it_is(хищник),
    + цвет('желто-коричневый'),
    + раскраска('темные пятна').
```

Для проверки условий правила определим вспомогательные предикаты: +, –. Эти предикаты выполняют проверку истинности или ложности указанного факта. Если значение свойства еще не введено пользователем, то вызывается предикат ask.

Для ввода значений свойств от пользователя определим предикаты ask, ask_menu, ith, print_menu.

Предикат ith возвращает i-й элемент списка или „–“, если i=0.

Предикат print_menu печатает список возможных значений свойства.

Предикат ask_menu выполняет печать меню со списком возможных значений для некоторого свойства и ввод ответа пользователя.

Предикат ask может работать в двух режимах: сбор возможных значений свойств и запрос фактов у пользователя.

Запуск на выполнение осуществляется с помощью предиката run. Сначала предикат run выполняет сбор списков возможных значений для свойств, вызывая и применяя все правила ЭС, затем происходит реальный опрос пользователя с использование стратегии глубины.

3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Запускаем ЭС с помощью команды:

```
| ?- run.
```

Выбираем для тестирования животное “жираф” и отвечаем на вопросы, задаваемые ЭС:

```
покрыто
1 - шерстью
2 - перьями
0 - нет в списке/не знаю
1
ест
1 - мясо
2 - траву
0 - нет в списке/не знаю
2
острые зубы
1 - да
2 - нет
0 - нет в списке/не знаю
2
дает молоко
1 - да
2 - нет
0 - нет в списке/не знаю
1
на ногах
1 - когти
2 - копыта
0 - нет в списке/не знаю
2
длинная шея
1 - да
2 - нет
0 - нет в списке/не знаю
1
длинные ноги
1 - да
2 - нет
0 - нет в списке/не знаю
1
раскраска
1 - темные пятна
2 - черные полоски
0 - нет в списке/не знаю
1
Это животное - жираф
```

Экспертная система успешно определила выбранное животное по его свойствам. Аналогично были проверены остальные животные.

ПРИЛОЖЕНИЕ К
ПРИМЕР ОСНОВНОЙ ЧАСТИ ДЛЯ ВЕБ-ПРИЛОЖЕНИЯ

1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать браузерное приложение, которое при старте генерирует 3-разрядное число. Пользователь должен определить это число, изменяя цифры и нажимая кнопку Open. В случае неудачи цифры меняют фон на красный, в случае успеха – на зеленый.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Для взаимодействия html-страницы и программы на Kotlin необходимо задать идентификаторы для всех важных элементов html-страницы. Строка цифр идентификатор идентификатор myrow, кнопка – mybutton (оформить как таблицу?).

Взаимодействие с html-страницей в Kotlin выполняется с помощью пакетов org.w3c.dom и kotlin.browser [3].

Сгенерированный код будет храниться в списке code, список ячеек с цифрами в списке tdlist. При получении коллекции ячеек из строки myrow сразу выполним преобразовании коллекции в список, а элементы списка из Element в HTMLElement, используя функцию высшего порядка map:

```
document.getElementById("myrow")!!.children.asList().map{it as
HTMLElement}
```

Функция main производит инициализацию элементов html-страницы и назначает обработчики событий. Для обработки событий определим две вспомогательные функции, которые будут вызываться в событии click.

Функция incText изменяет состояние текста ячейки с цифрой.

Функция test проверяет соответствие текущего набора цифр и цифр в списке code.

Эти были введены для повышения понятности кода, хотя в Kotlin можно обойтись без них. Код без функции incText:

```
it.addEventListener("click", {e->
  {  it.innerText=((it.innerText.toInt()+1)%10).toString()
     it.style.backgroundColor="white"
  } (e.currentTarget as HTMLElement)})
```

Использование функций map, all и forEach позволило упростить код, повысить его понятность за счет исключения операторов цикла.

3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для тестирования программы заменим случайную генерацию на фиксированный код 123.

После запуска начальное состояние 000, как показано на рисунке 1.

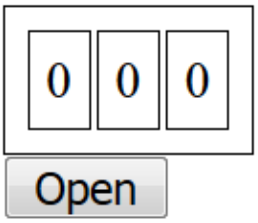

Рисунок 1 – Состояние страницы после запуска

Проверяем нажатие на цифры и переход с 9 на 0 (рисунок 2).

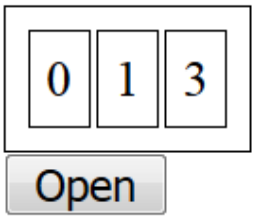

Рисунок 2 – Состояние страницы после изменения цифр

После нажатия кнопки Open фон у цифр становится красным (рисунок 3).

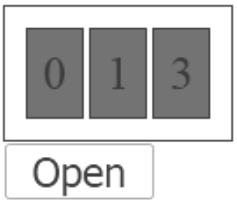

Рисунок 3 – Состояние страницы после нажатия кнопки Open

Установим код 123 и нажмем кнопку Open – фон у цифр становится зеленым (рисунок 4).

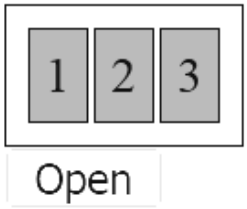

Рисунок 4 – Состояние страницы при выборе корректного кода

Все заданные в постановке задачи функции были проверены, и ошибок не было обнаружено.

ПРИЛОЖЕНИЕ Л
ПРИМЕР ОФОРМЛЕНИЯ ПРИЛОЖЕНИЯ С ИСХОДНЫМ КОДОМ

ПРИЛОЖЕНИЕ А

А.1 Файл index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Break the lock</title>
<style> table, th, td { border: 1px solid black; padding: 5px; }
</style>
</head>
<body>
<table ><tr id="myrow"><td></td><td></td><td></td></tr></table>
<button id="mybutton">Open</button>
<script src="out/production/HelloWorld/lib/kotlin.js"></script>
<script src="out/production/HelloWorld/HelloWorld.js"></script>
</body>
</html>
```

А.2 Файл main.kt

```
import org.w3c.dom.*
import kotlin.browser.document
import kotlin.random
val code=listOf(1,2,3) // List(3){ Random.nextInt(10) }
fun incText(e:HTMLElement) {
    e.innerText=((e.innerText.toInt()+1)%10).toString()
    e.style.backgroundColor="white"
}
val tdlist:List<HTMLElement>=document.getElementById("myrow")!!.
     children.asList().map{it as HTMLElement}
fun test() {
  val color=if((tdlist zip code).
                all({it.first.innerText.toInt()==it.second}))
         "green" else "red"
  tdlist.forEach {it.style.backgroundColor=color}
}
fun main(args: Array<String>) {
  tdlist.forEach { it.innerText="0"
       it.style.backgroundColor="white"
       it.addEventListener("click",
          {e->incText(e.currentTarget as HTMLElement)})
  }
  (document.getElementById("mybutton")!! as HTMLButtonElement).
       addEventListener("click", { test() })
}
```

ПРИЛОЖЕНИЕ М
ПРИМЕРЫ ОБОЛОЧЕК ДЛЯ ЭКСПЕРТНОЙ СИСТЕМЫ

М.1 Пример простой оболочки экспертной системы

```
:-dynamic(mem/2).
:-dynamic(menu/2).
:-dynamic(menu_collect/0).
% пример правил
animal_is(гепард):-
    it_is(хищник),
    + цвет('желто-коричневый'),
    + раскраска('темные пятна').
it_is(хищник):-
    + 'острые зубы',
    + 'на ногах'(когти),
    + 'глаза смотрят вперед'.

+ A :- atom(A)->(mem(A,Y)->Y=да; ask(A,да,Y),!,Y=да);
      A=..[U,V],(mem(U,Y)->Y=V; ask(U,V,Y),!,Y=V).
- A :- atom(A)->(mem(A,Y)->Y=нет; ask(A,нет,Y),!,Y=нет);
      A=..[U,V], (mem(U,Y)->Y\=V; ask(U,-V,Y),!,Y\=V).

ask(_,V,V):-menu_collect,(V=да;V=нет),!.
ask(U,-V,'-'):-menu_collect,
  (\+menu(U,V),assertz(menu(U,V));true),!.
ask(U,V,V):-menu_collect,
  (\+menu(U,V),assertz(menu(U,V));true),!.
ask(U,V,R):-(V=да;V=нет),!,ask_menu(U,[да,нет],R),
  assertz(mem(U,R)).
ask(U,_,R):-findall(W,menu(U,W),L),ask_menu(U,L,R),
  assertz(mem(U,R)).

ith(0,_,'-').
ith(1,[R|_],R).
ith(N,[_|T],R):-N>1,N1 is N-1,ith(N1,T,R).

print_menu([],_).
print_menu([H|T],N):-write(N),write(' - '),write(H),nl,
  N1 is N+1,print_menu(T,N1).

ask_menu(U,L,R):-write(U),nl,print_menu(L,1),
   write('0 - нет в списке/не знаю'),nl,
   flush_output,read_integer(N),ith(N,L,R),!.

run:- (\+ menu(_,_),
   asserta(menu_collect), findall(A,animal_is(A),_),
   retract(menu_collect);true), (animal_is(X)->
   write('Это - '),write(X),nl;
   write('В ЭС нет такого объекта'),nl),retractall(mem(_,_)).
```

М.2 Пример оболочки экспертной системы с возможностью объяснения

```
:- dynamic(факт/2).
:- dynamic(ложь/2).
:- op(980, fx, правило).
:- op(980, fx, почему).
:- op(975, xfx, если).
:- op(950, xfx, то).
:- op(930, fx, не).
:- op(940, xfy, и).
:- op(945, xfy, или).

% пример правил
правило 1 если класс(хищник) и цвет(желто-коричневый)
    и раскраска('темные пятна') то это(гепард).
правило 2 если 'острые зубы'(да) и 'на ногах'(когти) и
    'глаза смотрят вперед'(да) то класс(хищник).

check((не G),N):-!,(check(G,N)->!,fail;true).
check(G1 и G2,N):-!,check(G1,N),check(G2,N).
check(G1 или G2,N):-!,(check(G1,N);check(G2,N)).
check(G,_):-G=..[F,_],D=..[F,_],
   (факт(D,_),!,D=G;ложь(G,_),!,fail).
check(G,_):-G=..[_,_],(правило N если P то G),
   (check(P,N),!,assertz(факт(G,N));assertz(ложь(G,N)),fail).
check(G,N):-G=..[F,_],D=..[F,X],
   findall(P,(правило _ если P то _),R),
   collect(R,F,L),sort(L,L1),!,(L1=[] -> G=..[F,'-'];
  repeat,ask(F,L1),
    catch(read_integer(M),_,(explain(N),get_char(_),fail)),
    M1 is M+1, nth(M1,['-'|L1],X),!,assertz(факт(D,0)),!,D=G).

explain(N):-(правило N если P то G),
  write('сейчас рассматривается правило:'),nl,
  write(правило N если P то G),nl,flush_output,!.

ask(F,L1):-
   write('Введите значение для '),write(F),
   write(' или ? для пояснения:'),nl,
   write('0 - вариант отсутствует'),nl,writelist(L1,1),
   write('Введите номер варианта: '),flush_output,!.

writelist([],_).
writelist([H|T],I):-write(I),write(' - '),write(H),nl,
   J is I+1, writelist(T,J).

collect([],_,[]).
collect([G1|G2],F,L):-
collect(G1,F,L1),collect(G2,F,L2),append(L1,L2,L).
collect(не G1,F,L):-collect(G1,F,L).
```

```
collect(G1 и G2,F,L):-
  collect(G1,F,L1),collect(G2,F,L2),append(L1,L2,L).
collect(G1 или G2,F,L):-
  collect(G1,F,L1),collect(G2,F,L2),append(L1,L2,L).
collect(G,F,[A]):-G=..[F,A],!,nonvar(A).
collect(_,_,[]).

(почему не G):-факт(G,_),!,write('нет, утверждение '),write(G),
   write(' является истинным'),nl;
    G=..[F,_],D=..[F,_],факт(D,0),!,write('факт '),write(D),
    write(' введен пользователем'),nl;
    ложь(G,N),!,(правило N если P то G),
    write(правило N если P то G),nl;
    write('факт '),write(G),write(' не был опровергнут').
(почему G):-ложь(G,_),!,write('нет, утверждение '),write(G),
    write(' является ложным'),nl;
    факт(G,0),!,write('факт '),write(G),
    write(' введен пользователем'),nl;
    факт(G,N),!,(правило N если P то G),
    write(правило N если P то G);
    write('факт '),write(G),write(' не был выведен'),nl.

% :- initialization(main).
main:-retractall(факт(_,_)),retractall(ложь(_,_)),
  check(это(X),0),write(это(X)),nl.
```

ОГЛАВЛЕНИЕ

1 Этапы выполнения курсовой работы 3
2 Темы для курсовых работ 3
3 Постановка задачи 5
4 Проектирование и реализация 6
5 Тестирование программы и исправление ошибок 9
6 Оформление пояснительной записки 9
7 Содержание пояснительной записки 10
8 Защита курсовой работы 12
Библиографический список 13
Приложения
Приложение А 14
Приложение Б 15
Приложение В 16
Приложение Г 17
Приложение Д 18
Приложение Е 20
Приложение Ж 21
Приложение И 22
Приложение К 25
Приложение Л 27
Приложение М 28

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ
ПРОГРАММИРОВАНИЕ

Методические указания
по выполнению курсовых работ

Составитель **Демидов** Андрей Константинович

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 15.06.2021. Формат 60×84 1/16. Печать цифровая.
Усл. печ. л. 1,86. Тираж 30 экз. Заказ 167/290.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.