Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизации обработки информации (АОИ)

В. М. Зюзьков, Н. Ю. Салмина

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Учебное методическое пособие

Томск 2019

Корректор: А. Н. Миронова

Зюзьков В. М., Салмина Н. Ю.

Функциональное и логическое программирование : учебное методическое пособие / В. М. Зюзьков, Н. Ю. Салмина. – Томск : ФДО, ТУСУР, 2019. – 66 с.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящее учебно-методическое пособие разработано на основании и с учетом положений следующих документов:

1) Федерального закона от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»;

2) Порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры, утвержденного приказом Минобрнауки РФ от 05 апреля 2017 г. № 301;

3) ФГОС ВО и основной профессиональной образовательной программы (ОПОП) по направлению подготовки 09.03.04 «Прикладная информатика» (уровень бакалавриата);

4) Положения о проверке самостоятельности выполнения письменных работ бакалавров, специалистов и магистров в ТУСУРе (утв. директором Департамента образования ТУСУР П. Е. Трояном 26.05.2016).

В данном пособии:

– рассмотрен процесс подготовки и выполнения текстовых работ (контрольной, курсовой и лабораторных работ) по дисциплине;

– представлена информация для обучающихся о требованиях к составлению отчетов по контрольной и лабораторным работам;

– рассмотрены требования к подготовке материала, написанию и защите (рецензированию) курсовой работы.

В пособии представлена программа изучения разделов дисциплины «Функциональное и логическое программирование», изложено описание инсталляций интерпретаторов языков Лисп и Пролог, приведены варианты заданий для выполнения контрольной, курсовой и лабораторных работ.

Выбор варианта текстовых работ

Выбор варианта текстовых работ (контрольной, лабораторных, курсовой) осуществляется по общим правилам с использованием следующей формулы:

$$V = (N \times K) \text{ div } 100,$$

где V – искомый номер варианта,

N – общее количество вариантов,

div – целочисленное деление,

при $V = 0$ выбирается максимальный вариант,

K – код варианта, указанный в личном кабинете студента.

Требования к оформлению отчета по контрольной и лабораторным работам

Текстовые работы должны быть выполнены согласно требованиям образовательного стандарта вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Введен приказом ректора от 03.12.2013 № 14103. Режим доступа: https://regulations.tusur.ru/documents/70.

Структура отчета по лабораторной работе:

– титульный лист (приложение А);
– цели и задачи работы;
– основная часть (расчетная часть);
– выводы;
– сокращения, обозначения, термины и определения;
– список используемых источников;
– приложения (при необходимости).

Составленные и отлаженные программы (обязательно с комментариями) студент по мере освоения разделов дисциплины «Функциональное программирование» оформляет в виде отчета и отправляет для проверки преподавателю.

I ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

1 Программа раздела «Функциональное программирование»

Язык Лисп. Основы программирования. Элементарные понятия языка Лисп. Атомы и списки как основные объекты языка Лисп. Внутреннее представление списков. Понятие функции. Программа на языке Лисп. Вычислимые выражения. Базовые функции языка. Лямбда-выражения и определение новых функций.

Рекурсивные функции. Понятие рекурсии. Как работает рекурсивная функция. Правила записи рекурсивной функции. Рекурсия с несколькими терминальными или рекурсивными ветвями. Вспомогательные функции над списками.

Технология программирования на языке Лисп. Передача параметров. Глобальные и локальные переменные. Диалоговый режим работы. Функции ввода-вывода. Разрушающие функции. Функционалы. Циклы и блочные функции. Циклические предложения. Массивы. Свойства символов. Ассоциативные списки. Работа с файлами. Определение входных и выходных потоков. Чтение символов из файла.

Модели представления знаний. Фреймы. Понятие и состав фрейма. Пример построения фреймовой структуры на Лиспе. Семантические сети. Представление знаний с помощью семантических сетей. Пример построения семантической сети с помощью Лиспа.

2 Инсталляция XLisp

Интерпретатор с языка XLisp представлен в виде архива XLisp. XLisp работает в среде Windows, для его установки на компьютере достаточно распаковать архив XLisp (http://www.xlisp.org/). Запустите файл XLisp.exe из каталога XLisp\bin и войти в интерпретатор. Лисп выдает приглашение ">" и ждет ввода.

Программы можно набирать в окне интерпретатора, но это неудобно. Лучше воспользоваться каким-либо внешним текстовым редактором (XLisp не имеет собственного редактора). Например, можно воспользоваться стандартной программой Notepad («блокнот»). Набранные программы рекомендуется сохранять в виде текстовых файлов с расширением .lsp (можно использовать расширение .txt) в рабочем каталоге.

По умолчанию рабочим каталогом будет XLisp\bin, но можно его поменять. Для этого рекомендуется создать ярлык для XLisp.exe и изменить рабочий каталог в свойствах этого ярлыка.

Находясь в среде интерпретатора, программу можно загрузить при выборе пункта меню File | Load Lisp source…

Необходимо учитывать следующую особенность работы в Лиспе с блокнотом. После набора текста программы рекомендуется перейти курсором на новую строчку, иначе XLisp при загрузке файла с программой выдаст ошибку «неожиданный конец файла».

3 Лабораторная работа «Разработка программы с использованием языка Лисп»

Цель работы – получить практические навыки в решении задач по функциональному программированию с использованием языка Лисп.

Задание

1. Задание состоит из трех задач, в которых необходимо составить программы на Лиспе.

В *первой задаче* требуется применение простой рекурсии. При составлении программ (если не оговорено противное) можно использовать все встроенные функции Лиспа. Отладку программ можно осуществлять с помощью функции трассировки (trace <имя функции>), для отключения трассировки функции используйте (untrace <имя функции>).

Во *второй и третьей задачах* для программирования требуется использовать локальные или вспомогательные функции.

В *третьей задаче* требуется использовать функционалы. При составлении программ (если не оговорено противное) можно использовать все встроенные функции языка Лисп. Тексты всех программ, если вы мыслите в духе функционального программирования, буквально состоят из нескольких строчек.

Примеры решения задач

1. Напишите функцию, зависящую от двух аргументов x и u, удаляющую все вхождения x в список u на всех уровнях.

Решение задачи 1

```
(defun f (x u)
        (cond ((null u) nil)
                (t (let ((c (car u)) (r (cdr u)))
```

```
                    (cond ((and (atom c) (equal x c)) (f x r ))
                          ((atom c) (cons c (f x r)))
                          ( t (cons (f x c) (f x r))))
               )))
)
```

2. Напишите функцию, удаляющую повторные вхождения элементов в список, например, (a b c d a) -> (a b c d).

Решение задачи 2

```
(defun f (x)
    (labels
        (( f1 (y z)
            (cond ((null y) z)
                ((member (car y) z) (f1 (cdr y) z))
                ( t (f1 (cdr y) (append z (list (car y))))))))
    (f1 x nil))
)
```

3. Напишите функцию, строящую список всех подмножеств данного множества.

Решение задачи 3

```
(defun f (l)
        (cond ((null l) (list l))
                (t (append (mapcar (lambda (x) (cons (car l) x)) (f (cdr l)))
                        (f (cdr l)) )))
)
```

Варианты заданий

Вариант 1

1. Последовательность чисел Фибоначчи 1, 1, 2, 3, 5, 8, 13… строится по следующему закону: первые два числа – единицы; любое следующее число есть сумма двух предыдущих f(n) = f(n – 1) + f(n – 2). Напишите функцию (f n f1 f2) с накапливающимися параметрами f1 и f2, которая вычисляет n-е число Фибоначчи.

2. Определите функцию, зависящую от двух аргументов u и n, которая по данному списку строит список его элементов, встречающихся в нем не менее n раз. Проверьте работу этой функции на примере (a b a c b c a b d a b) для n = 1,2,5,0.

3. Используя функционалы, напишите функцию, которая из данного списка строит список списков его элементов, например, (a b) -> ((a) (b)).

Вариант 2

1. Определите возведение в целую степень (^ x n) через умножение и деление.

2. Напишите функцию, заменяющую Y на число, равное глубине вложения Y в W, например, Y = a, W = ((a b) a (c (a (a d)))) ->((2 b) 1 (c (3 (4 d)))).

3. Напишите функцию, единственным аргументом которой являлся бы список списков, объединяющую все эти списки в один.

Вариант 3

1. Напишите функцию, вычисляющую последний элемент списка.

2. Напишите функцию, которая делает из списка множество, т. е. удаляет все повторяющиеся элементы.

3. Напишите функцию (exists p x), которая проверяет, существует ли элемент списка x, удовлетворяющий предикату p (p – функция или функциональное имя).

Вариант 4

1. Напишите функцию, которая из данного одноуровневого списка строит список списков его элементов, например, (a b) -> ((a) (b)).

2. Напишите функцию, которая сортирует список чисел, используя алгоритм простой вставки.

3. Напишите функцию (all p x), которая проверяет, для всех ли элементов списка x выполняется предикат p (p – функция или функциональное имя).

Вариант 5

1. Определите функцию, зависящую от аргументов u и v, являющихся списками, которая вычисляет список всех элементов, содержащихся либо в u, либо в v, но не одновременно в u и v.

2. Определите функцию (f s), результатом которой является список, получающийся из списка списков s после удаления всех подсписков, содержащих числа.

3. Напишите функцию (filter p x), которая «фильтрует» (создает список) элементы списка x, удовлетворяющие предикату p (p – функция или функциональное имя).

Вариант 6

1. Напишите функцию, аналогичную встроенной функции замены subst в списке s выражения x на y, но производящую взаимную замену x на y, т. е. x->y, y->x.

2. Определите функцию (f s), которая вычисляет список (m1 m2 m3), состоящий из трех наибольших элементов списка s: m1 >= m2 >= m3.

Исходный список содержит не менее трех элементов.

3. Определите функцию (f s n), которая из списка чисел s создает новый список, прибавляя к каждому атому число n. Исходный список не предполагается одноуровневым.

Вариант 7

1. Определите функцию (summa_digits n), результатом которой является сумма цифр натурального числа n.

2. Определите функцию (f s), которая в одноуровневом списке чисел s переставляет все отрицательные элементы в начало списка, например, (f '(4 -8 6 -9 -7)) -> (-8 -9 -7 4 6), меняя знак у каждого атома. Исходный список не предполагается одноуровневым.

3. Определите функцию (f s), которая из списка чисел s создает новый список, меняя знак у каждого атома. Исходный список не предполагается одноуровневым.

II ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

1 Программа раздела «Логическое программирование»

Математические основы логического программирования. Логический язык первого порядка. Формальные теории первого порядка. Определение формальной теории. Исчисление высказываний. Определение теорий первого порядка. Доказательство от противного. Задача об автоматическом доказательстве теорем. Предваренная нормальная форма. Сколемизация. Конъюнктивная нормальная форма. Сведение к дизъюнктам. Правило резолюции для исчисления высказываний. Унификация. Правило резолюции для исчисления предикатов. Алгоритм резолюций. Опровержение методом резолюций.

Введение в Пролог. Хорновская логическая программа. Сеанс работы с интерпретатором Пролога. Пример Пролог-программы: родственные отношения. Рекурсивные правила. Общие принципы поиска ответов на вопросы системой Пролог. Декларативная и процедурная семантика программ. Алгоритм работы интерпретатора Пролога. Синтаксис языка SWI-Prolog. Порядок предложений и целей.

Структуры данных. Предикат унификации. Арифметические выражения. Примеры программ с числами. Дифференцирование. Списки. Синтаксис и семантика списков. Некоторые предикаты для списков. Структуры. Выборка структурированной информации из базы данных. Рекурсивные структуры. Модификация синтаксиса (операторная запись).

Управление повторением в прологе. Отсечение. Определение отсечения. Примеры программ с отсечением. Отрицание как неудача. Трудности с отсечением и отрицанием. Рекурсия.

Внелогические предикаты пролога. Анализ и синтез термов. Проверка типа термов. Создание и декомпозиция термов. Ввод и вывод. Метапрограммирование. Эквивалентность программ и данных. Предположение об открытости мира. Программирование второго порядка. Операции с базой данных.

2 Инсталляция SWI-Prolog

Интерпретатор с языка Пролог представлен в виде упакованного файла pl.arj. SWI-Prolog работает в среде Windows; для его установки на компьютере достаточно распаковать файл pl.arj вместе со всеми хранящимися в этом файле каталогами. Запустите файл pl.exe из каталога pl\bin и вы войдете в интерпретатор. Пролог выдает приглашение "?-" и ждет вашего ввода. Ссылка для установки программы: https://www.swi-prolog.org/

Чтобы набирать программы, надо воспользоваться каким-либо внешним текстовым редактором, т. к. SWI-Prolog не имеет собственного редактора. Например, вы можете пользоваться стандартной программой Notepad («Блокнот»). Набранные программы сохраняйте в виде текстовых файлов с расширением .pl (но можно и расширение .txt) в рабочем каталоге. По умолчанию рабочим каталогом будет pl\bin, но вы можете его изменить: для этого создайте ярлык для pl.exe и измените рабочий каталог в свойствах этого ярлыка. Чтобы загрузить программу из файла c:\pl\work\name.pl, находясь в среде интерпретатора, вы должны в ответ на приглашение "?-" набрать имя файла в квадратных скобках:

? - [name]

если установлен рабочий каталог - pl\work.

Или набрать имя файла с полным путем к нему:

? - ['c:\pl\work\name.pl'].

если установлен другой рабочий каталог. Отметим следующую особенность работы в SWI-Prolog с блокнотом. После набора текста программы обязательно перейдите курсором на новую строчку, иначе SWI-Prolog при загрузке файла с программой выдаст ошибку «неожиданный конец файла».

6 Контрольная работа «Разработка программы для написания простых предикатов с использованием языка Пролог»

Задание состоит из решения двух задач, в которых требуется составить программы на Прологе для написания простых предикатов. При составлении программ (если не оговорено противное) можно использовать все встроенные предикаты Пролога. Тексты всех программ получаются небольшие.

SWI-Prolog не имеет стандартного help'a для Windows, для этого используется предикат help. Вызов help(<имя предиката>) выдает на экран информацию об этом предикате. Вызов help(7) выдает на экран список всех встроенных предикатов с комментариями. Текстовый файл руководства по SWI-Prolog: - pl\library\manual. Отладку предикатов можно осуществлять с помощью предиката трассировки trace(<имя предиката>), отключение трассировки предиката: - trace(<имя предиката>, - all).

Пример задания с решением

Задача 1

Условие

Постройте предикат position_max(+L, -M, -N), который в списке L находит максимальное значение M и порядковый номер N этого значения.

Решение

Будем использовать метод накапливающего параметра. Для этого введем вспомогательный предикат position_max(+List, +I, +M0, +N0, -M, -N), где I равен номеру рассматриваемого элемента в исходном списке, M0 – текущее значение максимума, N0 – позиция текущего максимума.

```
position_max([X|T],M,N):-
    position_max(T,1,X,1,M,N).
position_max([],_,M,N,M,N).

position_max([A|T],I,X,_,M,N):-
    A>X,
    K is I+1,
    position_max(T,K,A,K,M,N).

position_max([A|T],I,X,Y,M,N):-
    A=<X,
    K is I+1,
    position_max(T,K,X,Y,M,N).

?- position_max([3,2,5,1,4,3],M,N).
M = 5
N = 3
Yes
```

Задача 2

Условие

Определите умножение целых чисел через сложение и вычитание.

Решение

Определим предикат p(+X,+Y,?R), где X и Y – сомножители, R – произведение.

Делаем рекурсию по второму аргументу предиката, используя рекурсивное определение X*Y=X*(Y-1)+X.

```
p(X,0,0).

p(X,Y,R):- Y>0,
           Y1 is Y-1,
           p(X,Y1,R1),
           R is R1+X.

p(X,Y,R):-  Y<0,
            Y1 is -Y,
            p(X,Y1,R).
```

Варианты заданий

Вариант 1

1. Определите возведение в целую степень через умножение и деление.

2. Напишите предикат p(+L, -N), истинный тогда и только тогда, когда N – предпоследний элемент списка L, имеющего не менее двух элементов.

Вариант 2

1. Напишите предикат p(+X, +N, -L), истинный тогда и только тогда, когда L – список из N раз повторенных элементов X.

2. Напишите предикат p(+L, -S), истинный тогда и только тогда, когда S – список списков элементов списка L. Например, p([a, b, c],[[a], [b], [c]]) – истина.

Вариант 3

1. Напишите предикат p(+L, -S), истинный тогда и только тогда, когда L – список списков, а S – список, объединяющий все эти списки в один.

2. Напишите предикат p(+L, -S), истинный тогда и только тогда, когда список S есть циклическая перестановка элементов списка L, например, p([f, g, h, j], [g, h, j, f]) – истина.

Вариант 4

1. Напишите предикат p(+X, +N, +V, -L), истинный тогда и только тогда, когда список L получается после добавления X на N-е место в список V.

2. Напишите предикат p(+N, +V, -L), истинный тогда и только тогда, когда список L получается после удаления N-го элемента из списка V.

Вариант 5

1. Напишите предикат p(+V, -L), истинный тогда и только тогда, когда список L получается после удаления всех повторных вхождений элементов в список V, например, p([a, b, c, d, d, a], [a, b, c, d]) – истина.

2. Напишите предикат p(+V, -L), истинный тогда и только тогда, когда список L получается после удаления из списка V всех элементов, стоящих на четных местах, например, p([1, 2, 3, 4, 5, 6], [1, 3, 5]) – истина.

Вариант 6

1. Напишите предикат p(+V, +X, -L), истинный тогда и только тогда, когда список L получается из списка V после удаления всех вхождений X на всех уровнях, например, p([1, [2, 3, [1]], [3, 1]], 1, [[2, 3, []], [3]]) – истина.

2. Напишите обобщение предиката member, когда ищется элемент на всех уровнях в списке.

Вариант 7

1. Определите предикат p(+U, +V, -L), истинный тогда и только тогда, когда список L есть список всех элементов списка U, не содержащихся в списке V.

2. Определите предикат p(+U, +V, -L), истинный тогда и только тогда, когда L – список всех элементов, содержащихся либо в списке U, либо в списке V, но не одновременно в U и V.

Вариант 8

1. Определите предикат p(+V, -L), истинный тогда и только тогда, когда L – список всех элементов списка V, встречающихся в нем более одного раза.

2. Напишите предикат subst(+V, +X, +Y, -L), истинный тогда и только тогда, когда список L получается после замены всех вхождений элемента X в списке V на элемент Y.

Вариант 9

1. Напишите предикат p(+V, -L), истинный тогда и только тогда, когда список L получается из списка V после удаления всех повторяющихся элементов, т. е. из списка получается множество.

2. Напишите предикат exists(+P, +L), который проверяет, существует ли элемент списка L, удовлетворяющий предикату P.

Вариант 10

1. Напишите предикат all(+P, +L), который проверяет, для всех ли элементов списка L выполняется предикат P.

2. Напишите предикат filter(+V, +P, -L), истинный тогда и только тогда, когда список L есть список всех элементов из списка V, удовлетворяющих предикату P («фильтрация» списка).

Вариант 11

1. Используя предикаты "родитель"(Родитель, Отпрыск), "женщина"(Человек), "мужчина"(Человек) и "супруги"(Жена, Муж), определите отношения "теща", "шурин" и "зять".

2. Башня из кубиков может быть описана совокупностью фактов вида "на"(Кубик1, Кубик2), которые истинны, если Кубик1 поставлен на Кубик2. Определите предикат "выше"(Кубик1, Кубик2), который истинен, если Кубик1 расположен на башне выше, чем Кубик2. (Указание: "выше" является транзитивным замыканием отношения "на".)

Вариант 12

1. Напишите предикат gcd(+A,+B,-D), истинный тогда и только тогда, когда D – наибольший общий делитель двух целых положительных чисел A и B.

2. Напишите программу для отношения double(+List, -ListList), в котором каждый элемент списка List удваивается в списке ListList. Например, double([1,2,3],[1,1,2,2,3,2]) выполнено.

Вариант 13

1. Напишите новую версию предиката length(+L, -N), в котором при подсчете количества элементов списка не учитывается пустой список. К примеру, для списка [a,b,c,d,e] новая версия процедуры должна сообщить, что длина списка равна пяти, а для списка [a,[],c,d,[]] эта процедура должна давать длину, равную трем.

2. Пусть имеется список структур "client": [client(a,29,3),client(b,29,6),client(c,40,2)].

Первым аргументом каждой структуры служит имя клиента, вторым – суточный тариф, а третьим – количество дней, на которое взята автомашина.

Напишите правило, позволяющее вычислить итоговую сумму оплаты, объединяющую выплаты всех клиентов, данные о которых содержатся в списке.

Вариант 14

1. Опишите процедуру для предиката *расщепить/4*, которая берет список целых чисел L1 и целое число N и выдает списки L2 и L3, такие, что числа из исходного списка, меньшие, чем N, помещаются в список L2, а остальные – в список L3.

2. Напишите предикат для вычисления чисел Фибоначчи, используя метод накапливающего параметра.

Вариант 15

1. Напишите предикат digits(+N, -L), истинный тогда и только тогда, когда L – список цифр натурального числа N.

2. Напишите предикат summa_digits(+N, -S), истинный тогда и только тогда, когда S – сумма цифр натурального числа N.

7 Лабораторная работа «Разработка программ с использованием языка Пролог»

Цель работы: освоить построение простых и более сложных программ с помощью языка Пролог.

Задание

Лабораторная работа включает выполнение заданий из двух блоков:

I. Решение двух задач, в которых требуется *написать простые программы на Прологе*. При составлении программ (если не оговорено противное) можно использовать все встроенные предикаты Пролога.

II. Решение двух задач, в которых требуется *написать более сложные программы на Прологе* (как правило, требуется определить несколько предикатов). При составлении программы (если не оговорено противное) можно использовать все встроенные предикаты Пролога.

Пример решения задач из блока I

Задача 1

Условие

Сортировка списка простым обменом (по возрастанию).

Решение

Заданный список L сортируется по следующему алгоритму:

– если L содержит два соседних элемента, нарушающих требуемое упорядочение, то эти элементы меняются местами, после чего к полученному списку применяется этот же алгоритм;

– если в L не встречается ни одной неупорядоченной пары соседних элементов, то список L отсортирован и тем самым является искомым списком.

Предикат ord(+L) проверяет, является ли список L отсортированным.

```
ord([]).
ord([_]).
ord([X,Y|T]):-
      X =< Y,
      ord([Y|T]).

change(L,L):-
      ord(L),!.
change(L,S):-
      append(L1,[X,Y|L2],L),
      X>Y,!,
      append(L1,[Y,X|L2],Z),
      change(Z,S).
```

Отсечения в данном случае ликвидируют многочисленные правильные ответы.

Задача 2

Условие

Пусть бинарное дерево задается рекурсивной структурой tree(<левое поддерево>,<правое поддерево>) и пустое дерево задано термом nil. Составьте программу subtree(+S, +T), определяющую, является ли S поддеревом T.

Решение

Для решения используется очевидная рекурсия.

```
subtree(X, X).
subtree(X, tree(L,_)):-
          subtree(X,L).
subtree(X,tree(_,R)):-
  subtree(X,R).
```

Пример решения задач из блока II

Задача 1

Условие

Определим операторы:

```
:- op( 100, fy, ~).
:- op( 110, xfy, &).
:- op( 120, xfy, v).
```

Булева формула есть терм, определяемый следующим образом: константы true и false – булевы формулы; если X и Y – булевы формулы, то и X v Y, X & Y, ~X – булевы формулы, здесь v и & – бинарные инфиксные операторы дизъюнкции и конъюнкции, а ~ – унарный оператор отрицания.

Напишите программу, распознающую логические формулы в конъюнктивной нормальной форме, т. е. формулы, являющиеся конъюнкцией дизъюнкций литералов, где литерал – атомарная формула или ее отрицание.

Решение

Из определения операций видно, что конъюнкция и дизъюнкция являются право ассоциативными, т. е. запрос

```
? - X & Y & Z = A & B
```

приводит к унификации X = A, Y & Z = B.

Определим два вспомогательных предиката literal(X) и dis(X), которые проверяют? является ли X литералом и элементарной дизъюнкцией соответственно.

```
literal(true).
literal(false).
literal(~ X):-
        literal(X).
```

```
dis(X):-
        literal(X).
dis(X v Y):-
        literal(X),
        dis(Y).

con(X):- dis(X).
con(X & Y):-
         dis(X),
         con(Y).

?- con( true & (~ false v true v false) & ~ true).
Yes
```

Задача 2

Условие

Напишите предикат p(+N, -L), истинный тогда и только тогда, когда список L содержит все последовательности (списки) из N нулей, единиц и двоек, в которых никакая цифра не повторяется два раза подряд (нет куска вида XX).

Решение

```
p(1,[[0],[1],[2]]).
p(N,R):- N>1,
      N1 is N-1,
      p(N1,R1),
      t(R1,R).

t([],[]).
t([X|T],Z):-
      t(T,R1),
      s(X,R),
      append(R,R1,Z).

s([0|T],[[1,0|T],[2,0|T]]).
s([1|T],[[0,2|T],[1,2|T]]).
s([2|T],[[0,2|T],[1,2|T]]).
```

Варианты заданий

Блок I

Вариант 1

1. Напишите предикат, аналогичный предикату subst, но производящий взаимную замену X на Y, т. е. X->Y, Y->X.

2. Напишите предикат, который определяет, является ли данное натуральное число простым.

Воспользуйтесь более общей задачей:
ispr(N, M) – "Число N не делится ни на одно число, большее или равное M и меньшее N".

Имеем ispr(N, M) истинно, во-первых, если N = M, и, во-вторых, если истинно ispr(N,M+1) и N не делится на M.

Вариант 2

1. Сортировка списка простой вставкой (по возрастанию).

2. Сортировка списка простым выбором (по возрастанию).

Вариант 3

1. Напишите предикат p(+L, -N), истинный тогда и только тогда, когда N – количество различных элементов списка L.

2. Напишите предикат p(+X, +Y, -Z), истинный тогда и только тогда, когда Z есть "пересечение" списков X и Y, т. е. список, содержащий их общие элементы, причем кратность каждого элемента в списке Z равняется минимуму из его кратностей в списках X и Y.

Вариант 4

1. Запрограммируйте предикат p(+A,+B), распознающий, можно ли получить список элементов A из списка элементов B посредством вычеркивания некоторых элементов.

Алгоритм: Если A – пустой список, то ответом будет «да». В противном случае нужно посмотреть, не пуст ли список B. Если это так, то ответом будет «нет». Иначе нужно сравнить первый элемент списка A с первым элементом списка B. Если они совпадают, то надо снова применить тот же алгоритм к остатку списка A и остатку списка B. В противном случае нужно снова применить тот же алгоритм к исходному списку A и остатку списка B.

2. Напишите предикат p(+X, +Y, +L), истинный тогда и только тогда, когда X и Y являются соседними элементами списка L.

Вариант 5

1. Определите отношение sum_tree(+TreeOfInteger, -Sum), выполненное, если число Sum равно сумме целых чисел, являющихся вершинами дерева TreeOfInteger.

2. Определим операторы:

:- op(100, fy, ~).

:- op(110, xfy, &).

:- op(120, xfy, v).

Булева формула есть терм, определяемый следующим образом: константы true и false – булевы формулы; если X и Y – булевы формулы, то и X v Y, X & Y, ~X – булевы формулы, здесь v и & – бинарные инфиксные операторы дизъюнкции и конъюнкции, а ~ – унарный оператор отрицания. Напишите предикат p(+T), определяющий, является ли данный терм T булевой формулой.

Вариант 6

1. Встроенный предикат functor(+Term, ?Functor, ?Arity) определяет для заданного составного терма Term его функтор Functor и местность Arity. Встроенный предикат arg(+N, +Term, ?Value) определяет для целого числа N и заданного составного терма Term его N-й аргумент Value. Определите

предикаты functor1 и arg1 – аналоги предикатов functor и arg через предикат univ (=..)

2. Напишите предикат range(?M, ?N, ?L), истинный тогда и только тогда, когда L – список целых чисел, расположенных между M и N включительно (предикат должен допускать различное использование, когда не менее двух из трех аргументов конкретизованы). Указание: используйте предикаты var(+X) и nonvar(+X).

Вариант 7

1. Напишите вариант программы plus(?X, ?Y, ?Z), пригодный для сложения, вычитания и разбиения чисел на слагаемые. Указание: используйте для порождения чисел встроенный предикат between(+Low, +High, ?Value), который порождает все целые числа от нижней границы Low до верхней границы High.

2. Напишите программу вычисления целочисленного квадратного корня из натурального числа N, определяемого как число I, такое, что I*I ≤ N, но (I+1)*(I+1) > N. Используйте определение предиката *between/3* для генерирования последовательности натуральных чисел с помощью механизма возвратов.

Блок II

Вариант 1

1. Напишите предикат p(+N, +K, -L), истинный тогда и только тогда, когда L – список всех последовательностей (списков) длины K из чисел 1,2,...,N.

2. Напишите предикат p(+N, -L), истинный тогда и только тогда, когда список L содержит все последовательности (списки) из N нулей и единиц, в которых никакая цифра не повторяется три раза подряд (нет куска вида XXX).

Вариант 2

1. Определите отношение ordered(+Tree), выполненное, если дерево Tree является упорядоченным деревом целых чисел, т. е. число, стоящее в любой вершине дерева, больше любого элемента в левом поддереве и меньше любого элемента в правом поддереве. Указание: можно использовать вспомогательные предикаты ordered_left(+X, +Tree) и ordered_right(+X, +Tree), которые проверяют, что X меньше (больше) всех чисел в вершинах левого (правого) поддерева дерева Tree и дерево Tree – упорядочено.

2. Определим операторы:

```
:- op( 100, fy, ~).
:- op( 110, xfy, &).
:- op( 120, xfy, v).
```

Булева формула есть терм, определяемый следующим образом: константы true и false – булевы формулы; если X и Y – булевы формулы, то и X v Y, X & Y, ~X – булевы формулы, здесь v и & – бинарные инфиксные операторы дизъюнкции и конъюнкции, а ~ – унарный оператор отрицания.

Напишите программу, распознающую логические формулы в дизъюнктивной нормальной форме, т. е. формулы, являющиеся дизъюнкцией конъюнкций литералов, где литерал – атомарная формула или ее отрицание.

Вариант 3

1. Определим операторы:

```
:- op( 100, fy, ~).
:- op( 110, xfy, &).
:- op( 120, xfy, v).
```

Булева формула есть терм, определяемый следующим образом: константы true и false – булевы формулы; если X и Y – булевы формулы, то и X v Y, X & Y, ~X – булевы формулы, здесь v и & – бинарные инфиксные операторы дизъюнкции и конъюнкции, а ~ – унарный оператор отрицания.

2. Напишите программу, задающую отношение negation_inward(+F1, -F2), которое выполнено, если логическая формула F2 получается из логической формулы F1 внесением всех операторов отрицания внутрь конъюнкций и дизъюнкций.

Вариант 4

1. Определите предикат occurances(+Sub,+Term,-N), истинный, если число N равно числу вхождений подтерма Sub в терм Term. Предполагается, что терм Term не содержит переменных.

2. Разработайте программу «Советник по транспорту». Выберите либо сеть, состоящую из городов, либо транспортную сеть маршрутов поездов или автобусов в пределах одного города. Вы должны информировать систему о том, откуда и куда вы собираетесь добраться, а система должна выдавать рекомендации о том, какими поездами, автобусами, самолетами вам следует воспользоваться, чтобы добраться до пункта назначения.

Вариант 5

1. Напишите предикат p(+S, -L), который переводит предложение S, представленное строкой, в список атомов L. Например, p('gfrtyre hjnki <> pi 876 h', [gfrtyre, hjnki, '<>', pi, 876,h]) выполнено. Указание: воспользуйтесь предикатом *name/2*.

2. Множественное число большинства английских существительных получается путем добавления буквы *-s* к форме единственного числа. Но если существительное заканчивается буквой *-y*, следующей за согласной, множественное число образуется путем замены буквы *-y* на сочетание *-ies*; если же существительное заканчивается буквой *-o*, следующей за согласной, множественное число образуется путем добавления сочетания *-es*. Напишите утверждения для предиката *множественное_число/2*, которые задают все эти правила. Указание: воспользуйтесь предикатом *name/2*.

Вариант 6

1. Простейшая система кодирования сообщений заключается в замене каждой буквы сообщения на букву, находящуюся на N-й по отношению к ней позиции в алфавите. Например, для N=2 буква *a* заменяется на *c*, буква *y* на *a* и т. д. Зная, что коды ASCII букв от *a* до *z* изменяются от 97 до 122, напишите процедуру для предиката *шифратор/3*, который берет шифруемое слово и целое число и выдает слово, представляющее шифр данного слова, полученный с помощью указанного метода. Указание: воспользуйтесь предикатом *name/2*.

2. Напишите предикат *предшествует/2*, который берет два атома в качестве своих аргументов и успешно согласуется, если первый из них в лексикографическом порядке предшествует второму. Указание: воспользуйтесь предикатом *name/2*.

Вариант 7

1. Одним из примеров использования предиката *name/2* может служить генерация новых атомов для представления вновь вводимых объектов, например, abc1, abc2, abc3 и т. д. Эти имена характеризуются тем, что все они состоят из корня, определяющего тип именуемого объекта, и целочисленного суффикса для различения объектов одного типа. Напишите программу: новое_имя(+X, -Y). Последовательность имен создается с помощью возвратов. Указание: воспользуйтесь предикатом *int_to_atom(+N,-X)*, который конвертирует натуральное число N в атом X.

2. Построить программу «сжать», назначение которой – преобразование английских слов в их «звуковой» код. Этот процесс предусматривает «сжатие» примерно одинаково звучащих слов в одинаковый их код – своего рода аббревиатуру этих слов. Слова «сжимаются» в соответствии со следующими правилами:

– первая буква слова сохраняется;

– все последующие за ней гласные, а также буквы *h, w, y* удаляются;

– сдвоенные буквы заменяются одиночными;

– закодированное слово состоит не более чем из четырех букв, остальные буквы удаляются.

Примеры: сжать(barrington, brng) и сжать(llewellyn, ln) – выполнено.

Указание: воспользуйтесь предикатом *name/2*.

8 Курсовая работа

8.1 Общие требования

Цели и задачи курсовой работы

Курсовая работа выполняется в сроки, определенные рабочим учебным графиком и рабочим учебным планом.

Курсовая работа является завершающим этапом в изучении дисциплины «Функциональное и логическое программирование» и выполняется с целью закрепления умений и навыков обучающегося по разработке и моделированию интеллектуальных систем с помощью языков функционального и логического программирования.

Задачи курсовой работы:

– закрепление знаний и овладение навыками использования понятийного аппарата функционального и логического программирования;

– приобретение навыков и освоение методов программирования достаточно сложных задач на языках функционального и логического программирования;

– подготовка к выполнению выпускной квалификационной работы.

В процессе выполнения курсовой работы студент должен приобрести и закрепить навыки:

– самостоятельного сбора и обобщения теоретического и практического материала по использованию технологий функционального и логического программирования;

– анализа поставленной задачи;

– работы с языками программирования: разработки алгоритмов, представления данных для решения поставленных задач, разработки модели различных классов систем с применением языков функционального и логического программирования, осуществления разработки программного обеспечения на языках Лисп и Пролог;

– работы с научно-технической литературой прикладного характера, использования библиографического поиска при анализе возможных вариантов решений по программированию интеллектуальных систем;

– использования основных методов и процессов разработки программного обеспечения;

– владения математическим аппаратом, применяемым в функциональном и логическом программировании;

– владения языками Лисп и Пролог для построения моделей искусственного интеллекта;

– оформления курсовой работы.

Выполнение курсовой работы способствует закреплению профессиональной компетенции по владению навыками использования различных технологий разработки программного обеспечения (ПК-3).

Общие требования к курсовой работе

Работа должна быть выполнена самостоятельно с применением языков программирования Лисп и Пролог. Программа должна представлять собой законченный продукт, который может использовать сторонний пользователь.

Материал, на котором строится подготовка и написание курсовой работы, должен быть точным, достоверным, обоснованным и опираться на результаты проведенного исследования. В курсовой работе должна наблюдаться внутренняя логическая связь, последовательность изложения.

При написании работы рекомендуется придерживаться научного стиля. Использование научного стиля изложения предполагает точность, ясность и краткость. Форма подачи материала – безличный монолог. Не употребляются местоимения первого лица единственного и множественного числа.

Курсовая работа должна быть отформатирована и написана грамотно. Следует соблюдать единообразие в применении терминов, сокращений слов и условных обозначений.

После того как работа будет написана, ее необходимо еще раз внимательно прочитать с целью устранения имеющихся опечаток, орфографических, пунктуационных и стилистических ошибок, повторов, неточностей и других недостатков.

Задание на курсовую работу

Курсовая работа выполняется в соответствии с заданием, выданным руководителем курсовой работы.

Задание на курсовую работу заполняется обучающимся самостоятельно, на основании описания предложенной тематики курсовых работ (п. 8.2).

Для некоторых курсовых работ обучающимся потребуются некоторые знания из теории графов. В качестве литературы можно использовать любой учебник по дискретной математике, содержащий теорию графов в небольшом объеме [11–13].

Для обучающихся с применением дистанционных образовательных технологий тема курсовой работы выбирается согласно варианту, задание оформляется на специальном бланке (приложение В).

Функции руководителя курсовой работы

Основными функциями руководителя курсовой работы являются:

– консультирование по вопросам содержания и последовательности выполнения курсовой работы;

– оказание помощи студенту в подборе необходимой литературы;

– контроль хода выполнения курсовой работы;

– подготовка письменных замечаний на курсовую работу;

– допуск студента к защите (рецензированию) курсовой работы.

Рекомендации по выполнению курсовой работы

Независимо от того, на каком языке требуется выполнить курсовую работу, с использованием языка Лисп или Пролог, обучающийся предоставляет граф в виде двух списков:

– списка вершин,

– списка ребер.

Каждое ребро, в свою очередь, есть список из двух вершин.

Рекомендуется использование двух алгоритмов для курсовых работ с графами.

Алгоритм поиска в глубину в графе для реализации на Лиспе

Функция (depth V,E,x,y,p,end) выдает путь (список вершин):

V – список вершин графа;

E – список ребер;

x – стартовая (начальная) вершина, при рекурсивном вызове depth, x – текущая вершина, откуда ведется поиск пути;

y – список вершин – соседей вершины x;

p – накапливаемый путь (накапливающий параметр), в начале поиска – пустой список, вершины накапливаются в обратном пройденному порядке;

end – предикат (функциональный аргумент), которому должна удовлетворять целевая (конечная) вершина искомого пути.

If x- целевая вершина ,
then получаем результат , добавляя к пути p вершину x, else
if список y вершин-соседей пуст then ответ = nil else
if первая вершина в списке y принадлежит пройденному пути p
then вызываем рекурсивно функцию depth для хвоста списка y else
if первая вершина в списке y не принадлежит пройденному пути p

then вызываем рекурсивно функцию, накапливая параметр p и меняя параметры x и y

else вызываем рекурсивно функцию depth для хвоста списка y

Алгоритм поиска в глубину для Пролога

Описание необходимых предикатов:

next(+X, ?Y) – выдает истину тогда и только тогда, когда X и Y – смежные вершины;

path(+Start, -Path) вычисляет путь Path (список вершин в порядке, обратном пройденному) из начальной вершины Start в целевую вершину, удовлетворяющую некоторому предикату end;

depth(+P, +X, -Path) – предикат, вычисляющий путь Path из стартовой вершины в целевую, P – накаливающийся параметр, содержащий уже пройденный путь, прежде чем попасть в вершину X.

Предикат path вычисляется с помощью правила

```
path(Start, Path):-
    depth([], Start, Path).
```

Алгоритм для вычисления depth(P, X, Path):

1) если X – целевая вершина, то результатом является путь P, к которому добавлена вершина X;

2) в противном случае находим соседнюю вершину Y для X, которая не входит в пройденный путь P, добавляем вершину X к пути P и рекурсивно вызываем depth.

8.2 Варианты тем для курсовых работ

1. Упрощение электрических цепей

Цепь состоит из компонентов только трех видов: резисторов, емкостей и индуктивностей. Преобразовать цепь в более простую, используя упрощающие правила при параллельном и последовательном соединении компонент одного вида. Треугольники преобразовывать в звезды. Язык программирования: SWI-prolog.

Используйте следующее представление предметной области. Структуры r(Номинал), l(Номинал) и c(Номинал) изображают соответственно резистор, индуктивность и емкость с номиналами. Вся цепь представляется в базе данных фактами вида

comp(<метка элемента>,
<элемент: резистор, индуктивность или емкость>,
<список узлов элемента>).

Упрощение цепи сводится к изменению базы данных.

2. Программа для алгебраических вычислений

Объекты, с которыми обучающемуся придется работать, – это многочлены от нескольких переменных, представленных в символьном виде с вещественными коэффициентами. Многочлены должны изображаться как арифметические выражения, так умножение изображается знаком ‘*’, а возведение в степень – знаком ‘^’. Язык реализации – SWI-Prolog.

Для манипуляций с многочленами нужны некоторые предикаты, чтобы пользователь мог получать ответы на вопросы, на которые не удается ответить с помощью традиционных языков программирования. Для этого обучающемуся понадобится обозначать многочлены идентификаторами и хранить в базе данных пару (имя многочлена, сам многочлен). Предикаты выполняют некоторые операции над своими операндами и помещают результат в качестве значения некоторого имени многочлена.

Список предикатов:

1. Ввести многочлен и записать его под некоторым именем.

2. Образовать алгебраическую сумму (разность, произведение) двух многочленов и записать полученный многочлен под некоторым именем.

3. Возвести данный многочлен в целую степень и результат записать под некоторым именем.

4. Заменить каждое вхождение некоторой переменной в многочлене на данный многочлен и результат записать под некоторым именем.

5. Вычислить производную многочлена по переменной и результат записать под некоторым именем.

6. Напечатать данный многочлен.

Многочлен представлять в виде суммы членов, включающих только операции умножения и возведения в степень. В каждом таком одночлене все константы перемножены и образуют числовой коэффициент (первый сомножитель), переменные упорядочены по алфавиту и все степени одной переменной объединены так, что каждая переменная встречается лишь один раз. Следует приводить подобные члены, т. е. объединять одночлены, имеющие одинаковые наборы переменных и степеней, с соответствующим изменением коэффициентов.

Рекомендуемая литература: [14, 15].

Указания. Некоторые фрагменты программы:

```
% Полиномы хранятся в виде фактов
% pol(+Name, -<список одночленов>).

% Ввод полинома: enter(+Name)

enter(X):-
   write('Введите полином с именем '),write_ln(X),
   enter(X,[],_).

enter(X,S,P):-
```

```
    write_ln('Введите очередной одночлен или end'),
    read(T),
    ((T=end,place(X,S,P));
     (T \= end,
      enter(X,[T|S],P))).

% привести подобные во введенном полиноме, упорядочить члены
% загрузить в базу данных

place(X,S,P):-
    merge(S,[],P),
    retractall(pol(X,_)),
    assert(pol(X,P)).

% сложение полиномов, представленных списками одночленов
merge([],L,L).
merge([X|L1],L2,L):-
    insert(X,L2,L3),
    merge(L1,L3,L).

insert(X,[],[X]).
insert(X,[Y|T],[Z|T]):-
    equal(X,Y,Z),Z \= 0,!.
insert(X,[Y|T],T):-
    equal(X,Y,0),!.
insert(X,[Y|T],[X,Y|T]):-
    low(X,Y).
insert(X,[Y|T],[Y|T1]):-
    not(low(X,Y)),
    insert(X,T,T1).

/* equal(X,Y,Z) – из двух мономов X, Y при приведении подобных получаем Z
low(X,Y) – моном X предшествует моному Y */

% приведение полинома к более "читабельному" виду
canon([],0).
canon([X],X).
canon([X,Y],Y+X).
canon([X,Y|T],Z+Y+X):-
    length(T,M),M>0,
    canon(T,Z).
```

```
% показать полином
show(X):-
   pol(X,P),
   canon(P,P1),
   write('Полином с именем '),write_ln(X),write_ln(P1).

% сложение полиномов
plus_pol(X,Y,Z):-
   pol(X,P1),
   pol(Y,P2),
   merge(P1,P2,P),
   retractall(pol(Z,_)),
   assert(pol(Z,P)),
   show(Z).
```

Протокол:

```
?- enter(a).
Введите полином с именем a
Введите очередной одночлен или end
-8*x^5.
Введите очередной одночлен или end
9.
Введите очередной одночлен или end
10*x^5.
Введите очередной одночлен или end
x^3.
Введите очередной одночлен или end
-7*x^3.
Введите очередной одночлен или end
end.
Yes

?- show(a).
Полином с именем a
2 * x ^ 5 + -6 * x ^ 3 + 9
Yes

?-plus_pol(a,a,b).
Полином с именем b
4 * x ^ 5 + -12 * x ^ 3 + 18
Yes
```

3. Игра «Суммируйте до 20»

Два игрока по очереди называют какое-либо число в интервале от 1 до 3 включительно. Названные числа суммируются, выигрывает тот игрок, сумма чисел после хода которого равна 20. Напишите программу на SWI-Prolog, выигрывающую данную игру на стороне компьютера.

Указания. Используйте запоминающие функции [3]. Обучающемуся рекомендуется написать программу в таком виде, чтобы ее можно было легко модернизировать для решения более общей задачи.

Более общая задача: перед началом игры компьютер спрашивает: «Какой список допустимых ходов (какие числа можно называть) и какая предельная (выигрышная) сумма?».

4. Задача Прима – Краскала («жадный» алгоритм) на Прологе

Дана плоская страна и в ней n городов. Нужно соединить все города телефонной связью так, чтобы общая длина телефонных линий была минимальной.

Уточнение задачи. В задаче речь идет о телефонной связи, т. е. подразумевается *транзитивность* связи: если i-й город связан с j-м, а j-й с k-м, то i-й связан с k-м. Подразумевается также, что телефонные линии могут разветвляться только на телефонной станции, а не в чистом поле. Наконец, требование минимальности (вместе с транзитивностью) означает, что в искомом решении не будет циклов. В терминах теории графов задача Прима – Краскала выглядит следующим образом:

Дан граф с n вершинами; заданы длины ребер. Найти остовное дерево минимальной длины.

Как известно, дерево с n вершинами имеет n-1 ребер. Оказывается, каждое ребро надо выбирать жадно (лишь бы ни возникали циклы).

Алгоритм Прима – Краскала

(краткое описание)

В цикле n-1 раз делай:
"выбрать самое короткое еще не выбранное ребро при условии, что оно не образует цикл с уже выбранными".

Выбранные таким образом ребра образуют искомое остовное дерево. Напишите программу для решения задачи Прима – Краскала на SWI-Prologe.

5. Задача Прима – Краскала («жадный» алгоритм) на Лиспе

Решите задачу Прима – Краскала (см. предыдущую тему) на языке XLisp.

6. Упрощение арифметических выражений

Назовем арифметическим выражением терм, при конструировании которого используются только атомы, числа, скобки и знаки арифметических операций. Обучающемуся требуется написать программу для упрощения арифметических выражений на SWI-Prologe.

В целом задача упрощения выражений является достаточно сложной и в каком-то смысле неконкретизованной, т. к. единого верного решения для этой задачи нет. Если арифметическое выражение имеет несколько вариантов более простого представления, то какой из них выбрать в качестве решения? Это зависит от того, для каких целей необходимо упрощение.

Задачу упрощения выражения поставим следующим образом. Необходимо найти эквивалентное выражение, форма записи которого является более короткой, чем форма записи исходного выражения. Для упрощения выражений рекомендуется использовать различные рекурсивные «правила переписывания», каждое из которых «упрощает» какое-нибудь подвыражение в исходном выражении. Правила переписывания должны соответствовать обычным математическим преобразованиям, как-то: приведению подобных и т. п., и представляются правилами Пролога (см. программу «дифференцирование выражений» из [3]).

Программа, разработанная обучающимся, должна быть некоторым компромиссом между желательной простотой написания и той сложностью, которой от нее требует поставленная задача.

Рекомендуется к использованию источник [16].

7. Определение связности графа на Прологе

Напишите программу на SWI-Prologe, определяющую, является ли данный неориентированный граф связным.

Указание: запрограммируйте предварительно предикат path(+X,+Y), проверяющий, существует ли путь из вершины X в вершину Y.

8. Определение связности графа на Лиспе

Напишите программу на языке XLisp, определяющую, является ли данный неориентированный граф связным.

Указание: запрограммируйте предварительно предикат (path X Y), проверяющий, существует ли путь из вершины X в вершину Y.

9. Определение эйлерова пути на Прологе

Напишите программу на SWI-Prologe, определяющую эйлеров путь, начинающийся с заданной вершины в неориентированном графе. Путь называется эйлеровым, если проходит через все ребра графа по одному разу. Теорема Эйлера утверждает, что такой путь всегда существует, если количество вершин в графе с нечетной степенью равно 0 или 2. Степень вершины – это количество ребер, которые инцидентны данной вершине. Если количество вершин с нечетной степенью равно 2, то эйлеров путь всегда начинается в одной из таких вершин.

10. Определение эйлерова пути на Лиспе

Напишите программу на языке XLisp, определяющую эйлеров путь, начинающийся с заданной вершины в неориентированном графе (см. предыдущую тему).

11. Определение компонент связности на Прологе

Напишите программу на языке SWI-Prolog, определяющую компоненты связности данного неориентированного графа. Каждая компонента связности – список вершин, следовательно, решением задачи должен быть список списков.

Указание: запрограммируйте предварительно предикат path(+X,+Y), проверяющий, существует ли путь из вершины X в вершину Y.

12. Определение компонент связности на Лиспе

Напишите программу на языке XLisp, определяющую компоненты связности данного неориентированного графа. Каждая компонента связности – список вершин, следовательно, решением задачи должен быть список списков.

Указание: запрограммируйте предварительно предикат (path X Y), проверяющий, существует ли путь из вершины X в вершину Y.

8.3 Требования к структуре и содержанию курсовой работы

Структура курсовой работы:

– титульный лист;
– задание на курсовую работу;
– оглавление (содержание);
– введение;
– основная часть;
– заключение;
– сокращения, обозначения, термины и определения;
– список использованных источников;
– приложения.

Титульный лист

На титульном листе отражаются:

– названия министерства, учебного заведения и кафедры;

– тема исследования. Тема исследования оформляется прописными буквами;

– вид работы – курсовая работа;

– сведения о студенте и руководителе курсовой работы: фамилия, имя, отчество и номер группы студента, выполнившего работу; ученая степень и должность руководителя по курсовой работе;

– дата предоставления работы;

– год написания работы.

Пример оформления титульного листа представлен в приложение Б.

Задание

Курсовая работа выполняется в соответствии с заданием, выданным руководителем курсовой работы. Задание оформляется на специальном бланке (приложение В). Форма задания определяется кафедрой автоматизации обработки информации (АОИ). Формулировка темы курсовой работы в задании должна соответствовать её формулировке в перечне тем курсовых работ. Допускается изменение темы курсовой работы после согласования с руководителем курсовой работы. Задание на курсовую работу заполняется студентом, согласовывается с руководителем курсовой работы и утверждается заведующим кафедрой АОИ, после утверждения задания вносить в него изменения и дополнения не разрешается.

Задание содержит конкретное название темы, необходимые исходные данные, перечень основных литературных источников, перечень графического материала, перечень разделов (глав) текстовой части проекта (работы). В задании указывается дата выдачи задания и представления работы

к защите (рецензированию), задание подписывается студентом и руководителем проекта (работы) и утверждается заведующим кафедрой.

Бланк задания на курсовую работу представлен в приложении В.

Оглавление (содержание)

Оглавление включает перечень основных частей работы с указанием страниц, на которых их помещают. Оглавление должно отражать все материалы, представляемые при подготовке курсовой работы.

Вместо слова «Оглавление» допускается использовать наименование «Содержание». Слово «Оглавление» записывают в виде заголовка, симметрично тексту, с прописной буквы, без номера раздела.

В оглавлении перечисляются заголовки разделов, подразделов, список литературы, каждое приложение и указывают номера страниц, на которых они начинаются.

Заголовки в оглавлении должны точно повторять заголовки в тексте курсовой работы. Последнее слово заголовка соединяют отточием с соответствующим ему номером страницы в правом столбце оглавления.

Пример оформления оглавления приведен в приложении Г.

Введение

В разделе «Введение» представляют цель работы, область исследования и (или) область применения исследуемого объекта.

Во введении приводится:

– формулировка задачи;

– определение цели;

– описание исходных данных (информация об используемой программе).

Объем введения – 1–2 страницы. Заголовок «Введение» записывают симметрично тексту с прописной буквы и, как правило, ставят перед ним номер раздела, например: «1 Введение».

Основная часть

В связи с программно-исследовательским характером курсовой работы структура основной части должна включать два основных раздела (главы):

– раздел, содержащий информационные основы разрабатываемой темы, где даны краткие описания известных функций, параметров и характеристик исследуемой программы (пакета), полученные из литературных источников, включая Интернет, обоснование требований на исследования неизвестных функций, параметров и характеристик программы (пакета);

– практическую часть, в которой содержится план исследований, указаны основные этапы исследований, выполнена обработка, анализ и формулировка полученных результатов в виде описания полученных параметров, характеристик и исследованных функций программы (пакета).

В основной части должно быть решение поставленной задачи, в частности:

1) анализ задачи;
2) обоснование выбора алгоритма;
3) обоснование выбора структур данных;
4) описание алгоритма;
5) обоснование набора тестов.

Анализ задачи

Разработка алгоритма представляет собой задачу на построение. Поэтому, как обычно в таких случаях (можно, например, вспомнить о методе решения геометрических задач на построение), необходим этап анализа задачи. Он позволяет установить, что является входом и выходом будущего алгоритма, выделить основные необходимые отношения между входными и выходными объектами и их компонентами, выделить подцели, которые

нужно достичь для решения задачи, и как следствие этого, выработать подход к построению алгоритма. Результатом этапа анализа задачи должна быть спецификация алгоритма, т. е. формулировка в самом общем виде того, что (в рамках выбранного подхода) должен делать алгоритм, чтобы переработать входные данные в выходные.

Описание алгоритма

Прежде всего нужно иметь в виду, что такое описание предназначено не для машины, а для человека. Другими словами, речь идет не о программе, а о некотором тексте (т. е. о словесном описании), по которому можно получить представление об общей структуре разрабатываемого алгоритма, о смысле его отдельных шагов и их логической взаимосвязи. Сохранение достаточно высокого уровня описания алгоритма также облегчает его обоснование. Поэтому шаги алгоритма должны описываться в терминах тех объектов и отношений между ними, о которых идет речь в формулировке задачи. Например, для «геометрической» задачи шаги алгоритма следует описывать как действия над точками, прямыми и т. п., как проверки свойств типа принадлежности трех точек одной прямой и т. п. Но не должно быть работы с кодами этих объектов, например, с матрицей координат точек некоторого множества.

При программировании на Прологе описание предикатов должно заключаться в указании, для каких отношений между сущностями (объектами предметной области) они введены. Какие аргументы предиката являются входными, а какие выходными? При программировании на Лиспе для описания функций должно быть указано, что функция вычисляет, а не как она это делает.

Нужно подобрать набор тестов, достаточный для демонстрации работы программы и ее реакции на экстремальные ситуации и неправильное обращение.

Заключение

Заключение должно содержать краткие выводы о проделанной работе, практическое приложение, перспективы использования результатов разработанного проекта. В заключении обучающемуся следует указать, чему он научился на примере этой задачи (на этот вопрос легко ответить, если сформулировать его в виде: «Что я в следующий раз сделаю иначе?»).

Сокращения, обозначения, термины и определения

При многократном упоминании устойчивых словосочетаний в тексте работы следует использовать аббревиатуры или сокращения. При первом упоминании должно быть приведено полное название с указанием в скобках сокращенного названия или аббревиатуры, например: «фильтр нижних частот (ФНЧ)»; «амплитудная модуляция (АМ)», при последующих упоминаниях следует употреблять сокращенное название или аббревиатуру.

Расшифровку аббревиатур и сокращений, установленных государственными стандартами (ГОСТ 2.316, ГОСТ 7.12), допускается не приводить.

Если в работе используется значительное количество (более пяти) сокращений, обозначений и (или) нестандартных терминов, соответствующие пояснения рекомендуется выполнять в виде специального раздела «Сокращения, обозначения, термины и определения».

Наличие специального раздела не исключает расшифровку сокращения или обозначения после первого упоминания в тексте. Раздел «Сокращения, обозначения, термины и определения» оформляют на отдельном листе, помещают его после заключения и указывают в оглавлении работы. Запись сокращений, обозначений, терминов приводят, как правило, в алфавитном порядке. Каждое сокращение, обозначение, термин указывают на новой строке, с абзацного отступа. Через знак «тире» записывают необходимую расшифровку, определение и/или пояснение и завершают строку точкой с запятой, а последнюю строку – точкой.

Список использованных источников

В список использованных источников входят те источники литературы, на которые есть ссылки в курсовой работе.

Приложения

В качестве приложений к курсовой работе помещают листинги программ и результаты их работы.

Требования к программе:

1) программа должна иметь простую и понятную структуру;

2) в программе должны быть прокомментированы используемые структуры данных,

3) для каждой функции каждой функции должно быть указано, что она делает, что является входными данными и результатом;

4) должен быть прокомментирован используемый алгоритм.

На все приложения в тексте работы должны быть даны ссылки. Приложения располагают в курсовой работе и обозначают в порядке ссылок на них в тексте.

Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением букв Ё, З, Й, О, Ч, Ь, Ы, Ъ.

Каждое приложение в работе следует начинать с нового листа (страницы) с указанием наверху посередине страницы слова «Приложение» и его обозначения, а под ним в скобках – «обязательное» (если его выполнение предусмотрено заданием) или «справочное». Приложение должно иметь заголовок, который записывают симметрично тексту с прописной буквы отдельной строкой.

Курсовая работа выполняется в строгом соответствии правилам, изложенным в Положении по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе.

8.4 Требования к оформлению курсовой работы

Требования к оформлению курсовой работы (проекта) представлены в образовательном стандарте вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Режим доступа: https://regulations.tusur.ru/documents/70

Курсовая работа должна включать текстовый, графический и другой иллюстративный материал. Объем курсовой работы – 20–25 страниц печатного текста с приложением таблиц, схем, программ, форм и других материалов. В курсовой работе важно формулировать свои выводы и комментарии.

Формат листа:

– размер А4 (210 х 297 мм);

– поля: левое – 30 мм; правое – 10 мм; верхнее – 20 мм; нижнее – 20 мм;

– абзацный отступ должен быть одинаковым по тексту отчета – 1,25 см;

– выравнивание основного текста – по ширине;

– нумерация страниц по центру сверху. Титульный лист считается первой страницей, но не нумеруется;

– интервал между абзацами не увеличен (до и после – 0).

Шрифт:

– стиль: Times New Roman;

– текст работы должен быть напечатан с полуторным междустрочным интервалом и размером шрифта либо 12, либо 14 пунктов.

Заголовки:

– основная часть может состоять из 2–3 разделов, каждый из которых может выключать несколько подразделов;

– подразделы должны иметь нумерацию в пределах каждого раздела;

– каждый раздел (1, 2, 3) начинается с нового листа. Подразделы и пункты на новый лист переносить не следует;

– наименования разделов, подразделов выравниваются по центру;

– если заголовок состоит из двух и более строк, то между ними межстрочный интервал 1, а слова переносят целиком;

– заголовок раздела (1, 2, 3) оформляется **строчными буквами полужирного начертания**;

– заголовок подраздела (1.1, 1.2, 2.1, 2.2 и т. п.) оформляется **строчными буквами полужирного начертания**;

– задание и список использованных источников не нумеруются. Эти заголовки выравниваются по центру, оформляются **строчными буквами полужирного начертания**;

– между двумя заголовками – одна пустая строка;

– между заголовком и текстом – 2 пустые строки.

Пример

2 Разработка простой программы с использованием языка Пролог

↕____*одна пустая строка*

2.1 Общая характеристика программы

↕____*две пустые строки*

Далее одинаково по всей работе размещается текст шрифтом 14 размера через полуторный интервал с абзацного отступа.

Тттттттттттттттттттт. Пппп.

После текста перед следующим заголовком _ ДВЕ ПУСТЫЕ СТРОКИ.

↕____*две пустые строки*

2.2 Система найма, приема и адаптации работников организации

Рисунки:

– все иллюстрации (схемы, диаграммы и т. п.) именуются рисунками;

– на графиках необходимо указывать единицы измерения по осям;

– на все рисунки по тексту перед объектом должны быть ссылки в формате: *«на рисунке 1.1 изображена процедура совершенствования…»*;

– если рисунок заимствован, то ссылка ставится в тексте, а не в названии рисунка: «на рисунке 1.1 [4] изображено»;

– название рисунка размещается внизу по центру листа в формате: Рисунок 1.1 – Название;

– рисунки нумеруются в пределах раздела (приложения) арабскими цифрами. Например, «Рисунок 2.3» (третий рисунок второго раздела, «Рисунок В.1» (первый рисунок приложения В).

Таблицы:

– на все таблицы должны быть ссылки в тексте. Таблицу следует располагать в работе непосредственно после абзаца где она упоминается впервые, или на следующем листе (странице), а при необходимости в приложении к работе*;*

– если данные заимствованы, то указывается в основном тексте ссылка на источник;

– название таблицы должно отражать содержание, быть точным и кратким, его помещают после номера таблицы, через знак «тире» с прописной буквы;

– точка после порядкового номера и в конце названия таблицы не ставится;

– допускается использование шрифта на размер меньше, чем основной текст, межстрочный интервал – одинарный;

– при переносе таблицы на следующую страницу «шапка» таблицы повторяется, а справа перед таблицей помещаются слова «Продолжение таблицы 2.4» либо «Окончание таблицы 2.4». Название таблицы не повторяется.

Требования к оформлению текста

По всему тексту следует у*брать режим автоматических переносов*.

При подготовке отчета необходимо обратить внимание на стиль и язык изложения, постараться обеспечить лаконичность и четкость формулировок, точность определений, разнообразие употребляемых слов, литературную форму выражения мысли. Стиль написания отчета – *безличный монолог*, т. е. от третьего лица. *Не рекомендуется употреблять* форму первого и второго лица местоимений единственного числа.

Во всём отчете должно быть достигнуто единообразие терминов, обозначений и условных сокращений.

Оформление списка использованных источников

В список использованных источников включаются лишь те источники, на которые есть ссылки по тексту работы. Список формируется в порядке упоминания источников в тексте работы. При отсылке к источнику в тексте работы, после упоминания о нем, проставляется номер в квадратных скобках, под которым он значится в списке источников.

Внимание! В таблице 8.1 приведены требования к оформлению элементов библиографического описания источников. Необходимо строго придерживаться изложенных требований.

Краткая схема библиографического описания (описание состоит из обязательных элементов) схематично может быть представлена так:

Заголовок описания. Основное заглавие: сведения, относящиеся к заглавию / Сведения об ответственности. – Сведения об издании. – Место издания, дата издания. – Объем.

Таблица 8.1 – Обязательные элементы библиографического описания

Структурный элемент	**Требования к оформлению**	**Пример**
Заголовок описания	Заголовок применяют при составлении записи на произведение одного, двух, трех авторов. Заголовок описания может включать имя лица (имя лица – условно применяемое понятие, включающее фамилию, инициалы или имя и отчество, псевдоним, личное имя или прозвище в качестве фамилии), наименование организации, унифицированное заглавие произведения, обозначение документа, географическое название, иные сведения	Кириллов, В. И.
Основное заглавие	Основным заглавием является заглавие книги или статьи, а сведением, относящимся к заглавию, – пояснение жанра, типа издания, например, сборник статей, учебное пособие и т. п. Если книга или статья размещена в электронной библиотечной системе или в сети Интернет далее в квадратных скобках указывается [Электронный ресурс]	Логика : учеб. пособие Логика [Электронный ресурс] : учеб. пособие
Сведения об ответственности	Это сведения о соавторах, переводчиках, редакторах и/или о той организации, которая принимает на себя ответственность за данную публикацию. Первым сведениям об ответственности предшествует знак *косая черта (слеш) (/)*. Имя и отчество автора(ов) указываются вначале, затем указывается фамилия. Последующие группы сведений отделяют друг от друга знаком «точка с запятой». Однородные сведения внутри группы отделяют запятыми. При переходе к следующему элементу библиографического списка ставятся знаки «точка» и «тире»	*Для книги одного автора*: / В. И. Кириллов. – / В. И. Кириллов ; Моск. гос. юрид. академия. – / В. И. Кириллов, А. П. Садохин, И. В. Петрова. –
Сведения об издании	Сведения об издании включают качественную и количественную характеристику документа. Сведения об издании отделяются от следующего библиографического элемента знаками «точка» и «тире»	– 2-е изд., перераб. и доп. –
Место издания, дата издания	Место издания – наименование города. Москва, Санкт-Петербург, Ростов-на-Дону сокращаются (М., СПб., Ростов н/Д), все остальные города пишутся полностью (Новосибирск, Киев). После наименования города ставится знак «двоеточие», затем пробел и указывается название издательства. После запятой приводится год издания. Место издания и дата издания отделяются от следующего библиографического элемента знаками «точка» и «тире»	М. : Проспект, 2009.
Объем	Объем – это количество страниц или диапазон страниц, на которых опубликована статья в журнале или сборнике. После указания объема ставится точка	233 с.
Источник	Если указывается учебник или статья, размещенные в электронной библиотечной системе или сети Интернет, в конце указывается URL (режим доступа), ссылка и в круглых скобках дата обращения	https://biblio-online.ru/viewer/logika-431777# (дата обращения: 18.01.2019)
Пример: Кириллов, В. И. Логика [Электронный ресурс] : учеб. пособие / В. И. Кириллов ; Моск. гос. юрид. академия. – 6-е изд., перераб. и доп. – М. : Проспект, 2009. – 233 с. – Режим доступа: https://biblio-online.ru/viewer/logika-431777# (дата обращения: 12.01.2019).		

Примеры оформления библиографических данных

Нормативный правовой акт из справочно-правовой системы «Консультант Плюс»

О предоставлении государственных гарантий субъектов Российской Федерации и муниципальных гарантий по кредитам либо облигационным займам [Электронный ресурс] : Постановление Правительства РФ от 30 декабря 2012 г. № 1487. – Режим доступа: Доступ из справ.-правовой системы «КонсультантПлюс».

Ресурс удаленного доступа (из сети Интернет)

Текущие показатели состояния Российского валютного рынка [Электронный ресурс] // Сайт Московской межбанковской валютной биржи. – Режим доступа: https://www.moex.com/ru/markets/currency/ (дата обращения: 16.01.2019).

Статья из научного журнала (одного автора)

Ефимов, А. А. Организация фирмы-посредника по оказанию услуг на рынке программных продуктов / А. А. Ефимов // Вестник ИНЖЭКОНа. – Сер. Экономика. – 2010. – Вып. 3(38). – С. 383–387.

Книга одного автора

Афонасова, М. А. Бизнес-планирование : учеб. пособие / М. А. Афонасова. – Томск : Томск. гос. ун-т систем упр. и радиоэлектроники, 2012. – 108 с.

Книга двух авторов

Рохмистров, М. С. Социология предпринимательства [Электронный ресурс] : учеб. пособие для академического бакалавриата / М. С. Рохмистров, С. Н. Рохмистров. – М. : Юрайт, 2018. – 245 с. – Режим доступа: https://biblio-online.ru/viewer/sociologiya-predprinimatelstva-441531# (дата обращения: 18.01.2019).

Книга более трех авторов

Боброва, О. С. Основы бизнеса [Электронный ресурс] : учебник и практикум для академического бакалавриата / О. С. Боброва, С. И. Цыбуков, И. А. Бобров [и др.]. – М. : Юрайт, 2019. – 330 с. – Режим доступа: https://biblio-online.ru/viewer/osnovy-biznesa-433141# (дата обращения: 01.02.2019).

Оформление перечислений

Особенностью отчетов по практикам является наличие большого количества перечислений, классификаций, основных понятий. Необходимо обращать внимание на их грамотное оформление.

Перечисление может быть нумерованным либо маркированным. *Нумерованный список* применяется в обязательном порядке, если в обобщающей части содержится *количественное числительное* (в примере это слово «три»).

Пример

Выделяют три уровня представления информации:

1) физическое;

2) концептуальное;

3) внешнее.

В *маркированном списке* перечислений в качестве маркера рекомендуется использовать либо «точку», либо «тире». Выбранный знак должен быть одинаковым по всей работе. Все элементы перечисления в целом должны грамматически подчиняться *вводному (обобщающему)* предложению, которое предшествует перечислению.

Пример

Информатика включает в себя следующие дисциплины:

– математическую логику;

– комбинаторику;

– теорию графов и т. д.

Вводное предложение (обобщающая часть предложения) при перечислении не должно заканчиваться предлогами «что», «на», «для», «в» и союзами «как», «при», «чтобы».

Неправильно	**Правильно**
Комплекс недвижимости подразделяется на: • данные адресного плана; • данные дежурного плана; • реестр объектов социальной инфраструктуры.	Комплекс недвижимости содержит следующие данные: • данные адресного плана; • данные дежурного плана; • реестр объектов социальной инфраструктуры.

Остальные требования к оформлению текста, в том числе формул, списка используемых источников и т. п., представлены в требованиях образовательного стандарта вуза ОС ТУСУР 02–2013. Работы студенческие по направлениям подготовки и специальностям гуманитарного профиля. Общие требования и правила оформления (https://regulations.tusur.ru/documents/71).

8.5 Защита (рецензирование) курсовой работы

Для подведения итогов – защиты (рецензирования) курсовой работы обучающемуся необходимо разместить в электронном курсе данной дисциплины следующие электронные документы:

– курсовую работу;

– файл с готовой программой.

Критерии оценивания курсовой работы

Основными критериями оценивания, по которым руководитель курсовой работы проводит рецензирование, являются:

– работа с источниками литературы при подготовке курсовой работы;

– логика изложения материала;

– полнота раскрытия темы;

– уровень разработки программы;

– оформление курсовой работы.

Для обучающихся с применением дистанционных образовательных технологий защита курсовой работы проводится в два этапа. На первом этапе курсовая работа проверяется руководителем курсовой работы и составляется рецензия, в которой отмечаются недостатки, требующие исправления. Рецензия на курсовую работу размещается в электронном курсе дисциплины.

После исправления недостатков наступает второй этап, на котором руководитель высылает студенту от 3 до 5 дополнительных вопросов по тематике курсовой работы. Итоговая оценка учитывает качество выполнения курсовой работы, правильность и полноту ответов на дополнительные вопросы.

После получения итоговой оценки обучающемуся необходимо загрузить исправленный вариант курсовой работы в электронный курс для оформления портфолио студента.

ЛИТЕРАТУРА

I Функциональное программирование

1. Салмина, Н. Ю. Функциональное программирование и интеллектуальные системы : учеб. пособие / Н. Ю. Салмина. – Томск : ФДО, ТУСУР, 2016. – 100 с.

2. Зюзьков, В. М. Функциональное программирование : учеб. пособие / В. М. Зюзьков. – Томск : Томский межвузовский центр дистанционного образования, 2005. – 140 с.

3. Зюзьков, В. М. Логическое и функциональное программирование: учеб.-метод. пособие / В. М. Зюзьков. – Томск : Томский межвузовский центр дистанционного образования, 2000. – 72 с.

4. Образовательный стандарт вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Введен приказом ректора от 03.12.2013 № 14103 [Электронный ресурс]. – Режим доступа: https://regulations.tusur.ru/documents/70 (дата обращения: 07.06.2019).

II Логическое программирование

5. Зюзьков, В. М. Логическое программирование : учеб. пособие / В. М. Зюзьков. – Томск : ТМЦДО, 2005. – 145 с.

6. Братко, И. Программирование на языке Пролог для искусственного интеллекта / И. Братко. – М. : Мир, 1990. – 560 с.

7. Стерлинг, Л. Искусство программирования на языке Пролог / Э. Шапиро, Л. Стерлинг. – М. : Мир, 1990. – 235 с.

8. Малпас, Дж. Реляционный язык Пролог и его применение / Дж. Малпас. – М. : Наука, 1990. – 464 с.

9. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: пер. с фр. / А. Тейз, П. Грибомон, Ж. Луи и др. – М. : Мир, 1990. – 432 с.

10. Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. – М. : Юрайт, 2019. – 397 с.

11. Цуканова, Н. И. Теория и практика логического программирования на языке Visual Prolog 7 : учеб. пособие для вузов [Электронный ресурс] / Т. А. Дмитриева, Н. И. Цуканова. – М. : Горячая линия – Телеком, 2015. – 232 с. – Режим доступа: https://e.lanbook.com/reader/book/111113/#1 (дата обращения: 19.06.2019).

12. Нефедов, В. Н. Курс дискретной математики / В. Н. Нефедов, В. А. Осипова. – М. : Изд-во МАИ, 1992. – 264 с.

13. Дискретная математика : учеб. пособие для вузов / Д. С. Ананичев, [и др.] ; под науч. ред. А. Н. Сесекина. – М. : Юрайт, 2019.

14. Жигалова, Е. Ф. Дискретная математика : учеб. пособие / Е. Ф. Жигалова. – Томск : ТУСУР, 2014. – 98 с.

15. Уэзерелл, Ч. Этюды для программистов / Ч. Уэзерелл. ; пер. с англ. – М. : Мир, 1982. – С. 114–120.

16. Лорьер, Ж.-Л. Системы искусственного интеллекта / Ж.-Л. Лорьер. – М. : Мир, 1991. – С. 129–131.

17. Положение по организации выполнения и защиты курсовых проектов и курсовых работ в ТУСУРе [Электронный ресурс]. – Режим доступа: https://regulations.tusur.ru/documents/63 (дата обращения: 19.06.2019).

ПРИЛОЖЕНИЕ А

(справочное)

Пример оформления титульного листа отчета по контрольной и лабораторным работам

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ТЕМА РАБОТЫ ПРОПИСНЫМИ БУКВАМИ

Отчет по лабораторной/контрольной работе

по дисциплине «Функциональное и логическое программирование»

Студент гр. _____
_____ /_____/
(подпись) *(Ф. И. О.)*
«____»_____ 20__ г.

Преподаватель
канд. техн. наук,
доцент кафедры АОИ
_____ / Н. Ю. Салмина /
(подпись) *(И. О. Фамилия)*
«___»_____ 20__ г.

Томск 20__

ПРИЛОЖЕНИЕ Б

(справочное)

Пример оформления титульного листа курсовой работы

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ТЕМА РАБОТЫ ПРОПИСНЫМИ БУКВАМИ

Курсовая работа
по дисциплине «Функциональное и логическое программирование»

Студент гр. _____
(номер группы)

_____ _____
(подпись) *(И. О. Фамилия)*

«___» _____ 20___ г.
(дата)

Руководитель

_____
(должность, ученая степень)

_____ _____
(подпись) *(И. О. Фамилия)*

Томск 20__

ПРИЛОЖЕНИЕ В

(справочное)

Форма задания на курсовую работу

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Заведующий кафедрой АОИ
_____
(звание, ученая степень)
_____ /_____/
(подпись) *(И. О. Фамилия)*
«_____»_____20___ г.
(дата)

ЗАДАНИЕ
на курсовую работу (проект) по дисциплине
«Функциональное и логическое программирование»

студенту _____

группа _____ факультет _____

1. Тема курсовой работы _____
_____
2. Срок сдачи студентом законченного работы «____» _____20____г.
3. Исходные данные к курсовой работе__ _____
_____
4. Содержание курсовой работы (перечень подлежащих разработке вопросов_____
_____
5. Перечень графического материала _____
6. Дата выдачи задания «____» _____ 20___г.

Руководитель
_____ _____ /_____ /
(должность) *(подпись)* *(И. О. Фамилия)*

Задание принял к исполнению
«____» _____ 20 ___ г. /_____ /
(дата) *(подпись студента)*

ПРИЛОЖЕНИЕ Г

(справочное)

Пример оформления оглавления

Оглавление