Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)

Кафедра моделирования и системного анализа

Панов С.А.

ТЕОРИЯ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам

Томск
2015

Панов С.А. Теория и технологии программирования / Методические указания к лабораторным работам – Томск: Томский государственный университет систем управления и радиоэлектроники, Кафедра моделирования и системного анализа, 2015. – 28 с.

Содержание

Лабораторная работа № 1

Методика составления технического задания

Цель работы: на основе приведенных правил составить техническое задание на программный продукт.

1 ОБЩИЕ СВЕДЕНИЯ

Данный раздел должен содержать информацию, предоставляющую общие сведения о проекте.

1.1Формулировка задания

Краткая формулировка задания на разработку.

1.1 Цели, достигаемые разработкой.

Например: "Автоматизация бизнес-процедур, выполняемых при заключении и исполнении долгосрочных договоров..."

1.2Категории пользователей.

Категории пользователей, на которых должен быть ориентирован результат разработки (директор, менеджер, бухгалтер и т.д.).

Области применения (универсальное решение, отражение специфики отраслей, регионов, отдельных организаций). Дополнительно указываются ограничения на применение, обусловленные спецификой предметной области, либо проектными решениями, например: "... предназначено только для организаций розничной торговли, ведущих количественно-стоимостной учет".

1.3 Наименование организации-заказчика.

Наименование организации-заказчика (заказчиков), город, Ф.И.О., телефоны и адреса электронной почты ответственных представителей.

1.4Основание для проведения работ.

Основание для проведения работ: приказ, распоряжение, план разработки версии, договор с заказчиком.

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Данный раздел должен содержать информацию, достаточную для понимания разработчиками сути, специфики и объема работ по техническому заданию.

В случаях, когда разработке ТЗ предшествовала стадия анализа, и имеется документ "Описание предметной области", содержащий необходимую информацию, он включается в состав приложений к ТЗ со ссылкой в данном разделе. При необходимости здесь могут быть даны пояснения и дополнения. Дублирование информации, содержащейся в указанных документах, не производится.

2.1 Описание (схемы) бизнес-процессов.

Описываются состав и алгоритмы выполнения бизнес-процессов предметной области. Приводятся источники формирования первичных документов, общие правила обработки, формирования выходных документов. Формы входных и выходных документов с примерами заполнения приводятся в приложении. Указывается количество и периодичность поступления первичных документов и формирования отчетов, другие характеристики (для конкретных заказчиков, либо типичных случаев).

Схемы бизнес-процессов включаются по согласованию с разработчиком технического проекта (или исполнителем, если ТП не разрабатывается).

При необходимости могут включаться описания (схемы) организационной структуры конкретных предприятий, перечни функциональных обязанностей.

Рекомендуется пояснять описания и схемы конкретными примерами (включая числовые), вынося их в приложения.

2.2 Состав данных и алгоритмы обработки информации

По каждому документу или потоку данных дается перечень учитываемых реквизитов (показателей). Приводится описание процедур обработки, взаимосвязь различных документов (текстовое или в виде блок-схем).

2.3 Недостатки существующих проектных решений

Краткое описание недостатков или указание на отсутствие проектных решений, необходимых для реализации описанных процессов в существующих аналогах или существующей версии (номер).

2.4 Текущий уровень автоматизации

Раздел при необходимости включается только для заказных доработок и содержит сведения, поясняющие требований к разработке (например, техническая и программная оснащенность, наличие программ-аналогов и взаимодействующих систем).

3 ТРЕБОВАНИЯ К РАЗРАБОТКЕ

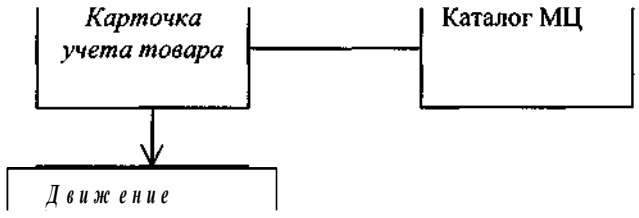
3.1 Информационная модель

Текстовое описание и схемы реализуемых бизнес-объектов (сущностей), их атрибутов, состояний и взаимосвязей.

Приводится схема, отображающая новые объекты, их взаимосвязи между собой и с объектами, существующими в системе.

.

Пример:



Приводится перечисление наиболее существенных для дальнейшей реализации атрибутов объектов с указанием возможных

значений.

Примечание: раздел является обязательным и не подлежит исключению.

3.2 Структура меню модуля

В виде схемы приводится развернутая структура меню модуля. Для нового модуля изображается целиком (кроме меню администратора и общесистемного), для дорабатываемого - только те фрагменты меню, в которые вносятся изменения. При этом новые функции выделяются жирным шрифтом.

Пример:

Документы

|Карточка учета товара

|Накладная на внутреннее перемещение

…

|Накладная на реализацию

| **Акт на списание**

|Накладная на возврат

|от покупателя

| поставщику

|на оптовый склад

3.3 Функциональные требования

Здесь приводится перечень функций (режимов выполнения), с указанием назначения и краткой характеристикой.

Описание может быть выполнено в виде таблицы. Примерная форма перечня (допускается расширение путем добавления колонок - например, ограничения, примечания и т.д.):

Раздел меню, наименование	Назначение, ссылка на описание бизнес-процесса	Использует т объекты (входные данные)	Оперирует с объектами (выходные данные)
Меню “Документы”			
Приходные накладные	Ввод и редактирование накладных приход от поставщика непосредственно в розницу (см. раздел 2.XX, рис. N, операция M)	Каталог подразделений Каталог МОЛ Каталог контрагентов Каталог КУТ	Создает приходы на заданный разрез хранения
Акт на списание	Ввод и редактирование (см. раздел 2.XX, рис. N, операция M)	Каталог подразделений Каталог МОЛ Записи по Приходам	Записи по расходам выбранных приходов

Ограничения по работе с системой, предусмотренные данной функциональностью, например, "...необходимо ведение партионного учета МЦ".

Примечание: разделы 3.2 и 3.3. при описании новых модулей или функций считается обязательным, исключение не допускается. Если предметом ТЗ является одиночная функция, допускается вместо структуры меню указывать только путь (способ активизации), вместо табличного описания приводится текстовое.

3.4 Требования к информационному обеспечению

Требования по составу и структуре входных данных, включая:

- данные, вводимые пользователем при работе с функциями, перечисленными в разделе "Функциональные требования";
- данные, хранящиеся в БД системы

Требования по составу и структуре выходных данных.

Требования по реализации печатных форм первичных и отчетных документов.

Требования по организации обмена данными между компонентами (модулями) системы.

Требования по срокам хранения данных.

Требования по организации обмена данными с другими программами (включая экспорт-импорт данных).

Требования по использованию общегосударственных (отраслевых) классификаторов, справочников, перечней.

Требования по изменению (дополнению) общесистемных справочников и классификаторов.

Требования по изменению (дополнению) параметров настройки. Требования по изменению (дополнению) других таблиц БД. Требования по использованию конкретных СУБД. Требования по обеспечению конвертации.

Примечание: при необходимости последовательность

перечисленных выше требований может быть изменена.

3.5 Требования к пользовательскому интерфейсу

Приводятся требования к пользовательскому интерфейсу, не оговоренные действующим стандартом, либо детализирующие его положения.

3.6 Требования к алгоритмам

Краткие описания алгоритмов, которые должны быть изменены или специально разработаны в обеспечение функциональных требований.

3.7 Прочие требования

Требования по изменению (дополнению) общесистемных и сервисных функций.

Требования по обеспечению быстродействия.

Требования по обеспечению надежности.

Требования по защите информации.

Требования по совместимости с имеющимися у заказчика аппаратными и программными средствами (только для заказных доработок)

Другие требования.

Примечание: в разделе 3.7 не следует приводить требования, которые уже реализованы в системе, или описаны в других разделах.

4 ПОРЯДОК КОНТРОЛЯ И ОБЕСПЕЧЕНИЯ КАЧЕСТВА

4.1 Экспертиза

Требуется (не требуется) разработка макета на стадии "Технический проект" на уровне (меню, основных интерфейсов и т.д.).

Требуется (не требуется) проведение экспертизы с участием

представителей (наименования служб и подразделений, внешних организаций):

- технического проекта
- макета

4.2 Тестирование

Тестирование проводить в соответствии с требованиями (наименование документа).

Требуется (не требуется) участие в тестировании представителей (наименования служб и подразделений, внешних организаций) на этапе (внутреннего, внешнего) тестирования.

4.3 Опытная эксплуатация

Требуется (не требуется) проведение опытной эксплуатации в перечисленных далее внешних организациях:

1. Организация 1
2. Организация 2
3. ...

Вместо наименований могут указываться также типы организаций.

5 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

5.1 Требования к справочной подсистеме

Приводятся требования к справочной подсистеме, не оговоренные действующим стандартом, либо детализирующие его положения.

5.2 Требования к документации пользователя

Требования:

- по разработке отдельной книги документации;
- дополнения существующих книг (новые разделы, либо

добавление в существующие разделы).

Дополнительно могут указываться примерный объем и структура документации.

Примечание: могут включаться требования по разработке или представлению иных видов документации, включая проектную.

Приложения

N	Наименование, (источник, дата публикации)	Стр.	Файл
1. 2. 3.			

Перечень документов, прилагаемых к ТЗ, включая копии законодательных (нормативных) актов, форм первичных и отчетных документов (и т.д.)

Графа "Файл" заполняется, если приложение передается в виде отдельного файла. Допускается также его включение в виде объекта средствами MS Word, использование гиперссылок.

Варианты

1. Разработка БД ‘Записная книжка’.
2. Разработка БД ‘Личная библиотека’.
3. Разработка БД ‘Программа телепередач’ [с функцией напоминания].
4. Разработка БД ‘Экзаменационная сессия’.
5. Разработка БД ‘Успеваемость студентов в группе’.
6. Разработка БД ‘Книга личных контактов’.
7. Разработка БД ‘Кадровое агентство’
8. Разработка БД ‘Учет кадров на предприятии (в ВУЗе)’

9. Разработка БД ‘Кадровое агентство’
10. Разработка БД ‘Склад продуктов (строительных товаров, …)’

Лабораторная работа № 2

Изучение методик проектирования

Цель: на основании ТЗ составленного в лабораторной работе №1 определить:

- Анализ существующего уровня автоматизации.
- Составляется список программного обеспечения, используемого в компании, и приводятся данные об использовании этих пакетов в каждом из подразделений организации.
- Общие требования к ИС
- Формулируются общие требования к функциональности разрабатываемой системы.
- Формы документов
- Устанавливаются перечень и структура документов, которые должны формироваться системой.
- Описание системы учета
- Описание системы учета включает в себя следующие документы:
- Учетная политика компании
- План счетов и используемых аналитик
- Список типовых хозяйственных операций и их отражение в проводках
- Описание справочников
- По каждому справочнику, проектируемому в системе, дается описание необходимой иерархической структуры.
- Организационная диаграмма
- Организационная диаграмма используется для отражения организационной структуры подразделений предприятия и их зон ответственности.
- Описание состава автоматизируемых бизнес-процессов
- Все бизнес-процессы компании должны быть перечислены в общем

списке и каждый должен иметь свой уникальный номер.

- Диаграммы прецедентов
- Для выделения автоматизируемых бизнес-процессов и их основных исполнителей используются диаграммы прецедентов.
- Физическая диаграмма
- Физическая диаграмма служит для того, чтобы описать взаимодействие организации на верхнем уровне с внешними контрагентами.
- Описания бизнес-процессов (книга бизнес-процессов).

Лабораторная работа № 3

Тестирования программного продукта

Цель: научиться выявлять ошибки в программном обеспечении

Вопросы

1. Перечислите и кратко охарактеризуйте различные способы контроля качества ПО.
2. Дайте определение тестирования и кратко прокомментируйте его.
3. Что означает в контексте тестирования ожидаемое поведение программы?
4. Что входит в искусственные, специально заданные условия воздействия на систему, которые имеются в виду в определении тестирования?
5. В чем важность концепции теста?
6. В чем преимущества автоматического тестирования перед "ручным"?
7. В чем трудности автоматического тестирования?
8. Приведите свои собственные примеры проблем с интерфейсами к тестируемым системам.
9. Приведите примеры того, как прогон тестов может влиять на поведение системы.
10. В чем смыл факторизации входных значений при тестировании?
11. Расскажите о разных вариантах организации команды тестировщиков.
12. Перечислите и кратко охарактеризуете виды тестирования.

Задания

1. Как на ваш взгляд концепция тестирования системы методом черного ящика связан с типичной организацией команды тестировщиков и ее взаимодействия с командой разработчиков? В ответе учтите различные стратегии, принятые в разных известных вам методологиях разработки

ПО. Ответ нарисуйте в виде карты памяти.

2. Нарисуйте в виде карты памяти взаимодействие между различными видами тестирования.
3. Нарисуйте пример жизненного цикла ошибки.

Лабораторная работа № 4

Изучение функциональных языков программирования

Цель: познакомится с языками программирования LISP и Prolog.

PROLOG

Задание 1

Вариант 0

Написать программу для реверса списка. Например: список [1, 2, 3] преобразуется в

список [3, 2, 1].

Вариант 1

Написать программу для получения значения n-го элемента списка. Например: в

списке [three, one, two] второй элемент равен one.

Вариант 2

Написать программу для удаления из списка всех элементов, равных 0. Например:

список [1, 0, 2, 0, 0, 3] преобразуется в список [1, 2, 3].

Вариант 3

Написать программу для циклического сдвига списка вправо на заданное число

элементов. Например: список [6, 5, 4, 3, 2, 1], циклически сдвинутый вправо на 2

элемента, преобразуется в список [2, 1, 6, 5, 4, 3].

Вариант 4

Написать программу для удаления из списка 2-ого, 4-ого и т.д. элементов. Например:

список [6, 5, 4, 3, 2, 1] преобразуется в список [6, 4, 2].

Вариант 5

Написать программу для замены в списке всех элементы, равные 0, на -1. Например:

список [1, 0, 0] преобразуется в список [1, -1, -1].

Вариант 6

Написать программу для перевода списка арабских чисел (от 1 до 10) в список

римских чисел. Например: список [1, 2, 3] преобразуется в список [“I”, “II”, “III”].

Вариант 7

Написать программу для подсчета количества определенных элементов в списке.

Например: в списке [1, 2, 1, 3, 1] три единицы.

Вариант 8

Написать программу для подсчета количества элементов списка без какого-либо

указываемого элемента. Например: в списке [1, 2, 1, 3, 1] два элемента без учета

единиц.

Вариант 9

Написать программу для подсчета количества элементов списка, значения которых

лежат в определенном диапазоне. Например: в списке [10, 20, 10, 30, 15] два элемента,

значения которых больше 10 и меньше 30.

Задание 2

Вариант 0

Написать программу для нахождения среднего арифметического листьевых вершин

бинарного дерева.

Вариант 1

Написать программу для проверки упорядоченности бинарного дерева.

Вариант 2

Вывести бинарное дерево на экран в виде дерева.

Вариант 3

Написать программу для вычисления глубины бинарного дерева (глубина пустого

дерева равна 0, глубина одноузлового дерева равна 1).

Вариант 4

Написать программу для подсчета количества листьевых вершин дерева, значения

которых лежат в определенном диапазоне.

Вариант 5

Написать программу для преобразования дерева в список.

Вариант 6

Написать программу для нахождения среднего арифметического отрицательных узлов

дерева.

Вариант 7

Написать программу для подсчета количества вершин бинарного дерева, значения

которых не равны 0.

Вариант 8

Написать программу для нахождения среднего арифметического положительных

узлов дерева.

Вариант 9

Написать программу для подсчета количества вершин бинарного

дерева, значения

которых равны 0.

Методические указания:

Списки

Prolog позволяет определить рекурсивные типы данных. Примерами рекурсивных

типов данных служат списки и деревья.

Список – это объект данных, содержащий конечное число других объектов

(элементов списка). Список, содержащий числа 1, 2 и 3, записывается следующим образом:

[1, 2, 3]

Для объявления списка используется следующее описание домена:

DOMAINS

integerlist=integer*

Список является рекурсивным составным объектом, он состоит из двух частей:

− головы списка – первого элемента списка;

− хвоста списка – списка_____, включающего все последующие элементы.

[1| [2, 3]]

голова списка 1

хвост списка [2, 3]

Так как список имеет рекурсивную составную структуру, для работы со списками

используется рекурсия.

Пример: печать элементов списка.

DOMAINS

integerlist=integer*

PREDICATES

printlist (integerlist)

CLAUSES

printlist ([]):- !. %для пустого списка ничего не делать

printlist ([H|T]):- write (H), nl, printlist (T). %для непустого списка отделить голову,

%напечатать ее, продолжить печать для

%хвоста списка

LISP

Задание 1

Вариант 0

Определить рекурсивную функцию, возвращающую значение n-го члена ряда

Фибоначчи: f(0)=0, f(1)=1, f(n)=f(n-1)+f(n-2).

Вариант 1

Определить рекурсивную функцию для удаления последнего элемента списка.

Вариант 2

Определить рекурсивную функцию, возвращающую произведение двух целых

положительных чисел (использовать суммирование).

Вариант 3

Определить рекурсивную функцию, возвращающую последний элемент списка.

Вариант 4

Определить рекурсивную функцию, возвращающую значение суммы ряда целых

четных чисел от 2 до n.

Вариант 5

Определить рекурсивную функцию, возвращающую список, из которого удалены 2-

ой, 4-ый и т.д. элементы.

Вариант 6

Определить рекурсивную функцию, возвращающую количество элементов в списке

без какого-либо указываемого элемента.

Вариант 7

Определить рекурсивную функцию, возвращающую количество определенных

элементов в списке.

Вариант 8

Определить рекурсивную функцию для циклического сдвига списка вправо на один

элемент.

Вариант 9

Определить рекурсивную функцию, возвращающую список, из которого удалены 1-

ой, 3-ый и т.д. элементы.

Методические указания

Символ – имя, состоящее из букв, цифр и специальных символов, и, возможно,

представляющее некоторый лисповский объект (то есть, может иметь некоторое значение).

Число – может быть целым или вещественным.

Символы t и nil – обозначают логическое значение истина и ложь, соответственно.

Атомы – это символы и числа.

Список – это основной тип данных в Lisp. Списки заключаются в круглые скобки,

элементы списка разделяются пробелами.

Пример списка: (1 2 (3 4) 4 5)

Список, в котором нет ни одного элемента, называется пустым списком и

обозначается () или nil.

В виде списков могут быть записаны как данные, так и программы. Например, список

(+ 2 3), в зависимости от контекста, может рассматриваться или как вызов функции

суммирования, или как список, состоящий из трех элементов.

Функция quote

‘(arg) или (quote arg)

Назначение: блокирует вычисление выражения.

Пример:

>(+ 2 3)

5

>‘(+ 2 3) ; или >(quote (+ 2 3))

(+ 2 3)

Базовые функции

В Lisp для построения, разбора и анализа списков существуют очень простые базовые

функции. Всего их пять – **car**, **cdr**, **cons**, **atom** и **eq**.

Функция car

(car list)

Назначение: выделяет первый элемент списка – голову списка.

Пример:

>(car ‘(one two three))

one

Функция cdr

(cdr list)

Назначение: выделяет хвост списка (как список).

Пример:

>(car ‘(one two three))

(two three)

Функция cons

(cons head tail)

Назначение: строит новый список из переданных ей в качестве аргументов головы и

хвоста.

Пример:

>(cons ‘one ‘(two three))

(one two three)

Функция atom

(atom arg)

Назначение: проверяет, является ли аргумент атомом.

Пример:

>(atom ‘(1 2 3))

NIL

>(atom ‘one)

T

>(atom (car (one two three)))

T

Функция eq

(eq symbol1 symbol2)

Назначение: проверяет тождественность двух символов.

Пример:

>(eq ‘one ‘one))

T

>(eq ‘one ‘two))

NIL

Управляющие структуры

Примером управляющей структуры в языке может служить предложение cond.

Предложение cond

(cond (predicate1 form1) (predicate2 form2) … (predicateN formN))

Назначение: выполнение ветвления.

Описание действия: выражения **predicate**, исполняющие роль предикатов,

вычисляются последовательно слева направо до тех пор, пока не встретится выражение,

значением которого является истина, далее вычисляется результирующее выражение **form**,

соответствующее этому предикату, и полученное значение возвращается в качестве значения

всего предложения **cond**. Если истинного предиката нет, то значением **cond** будет nil.

Рекомендуется в качестве последнего предиката использовать символ t, и соответствующее

ему результирующее выражение будет вычисляться всегда в тех случаях, когда ни одно

другое условие не выполняется.

Пример: функция deftype, определяющая тип выражения (пустой список, атом или

список).

```
>(defun deftype(arg)
(cond ((null arg) ‘emptylist) ((atom arg) ‘atom) (t ‘list)))
DEFTYPE
>(deftype ())
EMPTYLIST
>(deftype 'abc)
ATOM
>(deftype '(a b c))
LIST
```

Простая рекурсия

Функция является рекурсивной, если в ее определении содержится вызов этой же

функции. Рекурсия является простой, если вызов функции встречается в некоторой ветви

лишь один раз. Простой рекурсии в процедурном программировании соответствует

обыкновенный цикл. Например, задача нахождения значения факториала n! сводится к

нахождению значения факториала (n-1)! и умножения найденного значения на n.

Пример: нахождение значения факториала n!.

```
>(defun factorial (n)
(cond
((= n 0) 1) ;факториал 0! равен 1
(t (* (factorial (- n 1)) n)) ;факториал n! равен (n-1)!*n
)
)
>(factorial 3)
6
```

Трассировка программы

Для отладки программы можно использовать возможности трассировки. Трассировка

позволяет проследить процесс нахождения решения.

Для того, чтобы включить трассировку, можно воспользоваться следующей

директивой:

(trace function)

Назначение: включает режим трассировки.

Пример:

>(trace factorial)

(FACTORIAL)

> (factorial 3)

Entering: FACTORIAL, Argument list: (3)

Entering: FACTORIAL, Argument list: (2)

Entering: FACTORIAL, Argument list: (1)

Entering: FACTORIAL, Argument list: (0)

Exiting: FACTORIAL, Value: 1

Exiting: FACTORIAL, Value: 1

Exiting: FACTORIAL, Value: 2

Exiting: FACTORIAL, Value: 6

6

(untrace function)

Назначение: отключает режим трассировки.

Пример:

> (untrace factorial)

NIL__