МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет математики, информационных и авиационных технологий
Кафедра телекоммуникационных технологий и сетей

Е.Г. Чекал, А.А. Чичев

НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические рекомендации
для самостоятельной работы студентов направлений
09.03.02 «Информационные системы и технологии»
(бакалавриат)

Ульяновск
2019

УДК 683.03(075)
ББК 32.965я7
Ч-78

Методические рекомендации рекомендованы
к введению в образовательный процесс решением Ученого совета
факультета математики, информационных и авиационных технологий
Ульяновского государственного университета
(протокол № 2/19 от 19.03.2019)

Чекал Е.Г.

Ч-78 **Надежность информационных систем.** Методические рекомендации для самостоятельной работы студентов / Е.Г. Чекал , А.А. Чичев. – Ульяновск : УлГУ, 2019. – 40 с.

Методические рекомендации составлены в соответствии с программой дисциплины «Надежность информационных систем» и предусматривают подготовку по направлению 09.03.02 «Информационные системы и технологии» (бакалавриат) очной и заочной форм обучения.

В методических рекомендациях дается информация о дисциплине: цели, задачи, компетенции, особенности процесса изучения, основные виды и формы самостоятельной работы студентов, основные виды и формы контроля, списки рекомендуемой литературы, необходимого программного обеспечения, информационно-справочных систем.

Приводятся рекомендации к самостоятельной работе студентов при изучении теоретического материала, подготовке докладов, выполнению лабораторных работ.

Методические рекомендации могут использоваться студентами родственных специальностей и направлений.

УДК683.03(075)
ББК 32.965я7

ОГЛАВЛЕНИЕ

1. Общие положения
1.1. Информация о дисциплине

Целью изучения дисциплины является получение студентами теоретических знаний и практических навыков в области надежности информационных систем, позволяющих применять современные методы расчета и обеспечения надежности аппаратных и программных средств, при проектировании и сопровождении информационных систем различного назначения.

Задачи, решаемые в процессе изучения дисциплины, направлены на овладение студентами методами и современными инструментальными средствами исследования оценки надежности информационных систем, а также основами разработки средств обнаружения, локализации и восстановления отказавших программных средств.

Дисциплина изучается на лекциях, практических занятиях и в ходе самостоятельной работы студентов.

На лекциях студенты приобретают теоретические знания по основам теории надежности, традиционным методам анализа и расчета надежности аппаратных средств, элементам теории восстановления, особенностям обеспечения надежности программных средств, методам обеспечения и повышения надежности информационных систем.

На практических и лабораторных занятиях студенты приобретают умения и навыки комплексного использования методов оценки, обеспечения и повышения надежности аппаратных и программных средств, получают практические навыки по расчету показателей надежности и построению математических моделей информационных систем, осваивают приемы работы по обнаружению, локализации и восстановлению отказавших элементов.

В ходе самостоятельной работы студенты выполняют проработку теоретического материала по конспектам лекций и рекомендованной литературе, выполняют индивидуальные задания.

Дисциплина изучается в 7 семестре. Промежуточный контроль проводится в форме опросов, оценки докладов и защиты лабораторных работ, итоговый контроль проводится в форме экзамена.

В результате освоения дисциплины должны быть сформированы следующие компетенции:

Код и наименование реализуемой компетенции	Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций
ПК-9 Способен поддерживать работоспособность информационных систем и технологий в заданных функциональных характеристиках и соответствии критериям качества	Знать: - основные понятия теории надежности информационных систем; - о проблемах и основных направлениях развития методов и средств повышения надежности аппаратных и программных средств; - основные факторы, определяющие надежность функционирования информационных систем Уметь: - организовать работы по обнаружению, локализации и восстановлению отказавших элементов; - выполнять кодирование, тестирование, отладку и документирование программного обеспечения; - анализировать информацию о надежности информационных систем;

	Владеть: - навыками обнаружения, локализации и восстановления отказавших программных элементов - навыками расчета показателей надежности информационных систем; - технической терминологией теории надежности информационных систем
ПК-12 Способен оценивать надежность и качество функционирования информационных систем и технологий	Знать: - основные правовые документы, касающиеся ответственности за случайные или преднамеренные действия по снижению надежности информационных систем; - о взаимосвязи надежности, качества и безопасности информационных систем; - характеристики и показатели надежности информационных систем; - методы обеспечения и повышения надежности информационных систем. Уметь: - выполнять формализацию требований к разрабатываемой информационной системе с точки зрения надежности; - использовать математические модели надежности информационных систем; - разрабатывать программы обеспечения надежности ИС - рассчитывать и анализировать показатели надежности информационных систем; - распознавать случайные или преднамеренные действия, направленные на снижение надежности информационных систем; Владеть: - навыками комплексного использования методов оценки, обеспечения и повышения надежности информационных систем;
ПК-13 Способен осуществлять сертификацию ИТ-проекта по стандартам качества	Знать: - о взаимосвязи надежности, качества и безопасности информационных систем; Уметь: - рассчитывать и анализировать показатели надежности информационных систем; Владеть: - навыками расчета показателей надежности информационных систем;

ПК-15 Способен проводить расчет экономической эффективности информационных систем и технологий	Знать: - о влиянии надежности на экономическую эффективность информационных систем и на решение бизнес-задач в целом; - математические модели надежности информационных систем; - методы обеспечения и повышения надежности информационных систем. Уметь: - рассчитывать и анализировать показатели надежности, влияющих на экономическую эффективность информационных систем и технологий; Владеть: - навыками расчета показателей надежности, влияющих на экономическую эффективность информационных систем и технологий;

Особенностями процесса изучения данной дисциплины, в виду ее сложности, являются:
- интерактивный характер проведения лекций;
- разбор сложных вопросов программирования на практических занятиях, изучение дополнительных тем и заслушивание докладов;
- выполнение лабораторных работ по программированию и планированию вне лаборатории.

1.2. Основные виды и формы СРС

Основными ***видами*** СРС по дисциплине «Надежность информационных систем» являются:
– самостоятельное изучение теоретического материала по конспектам лекций и рекомендованной литературе;
– самостоятельное выполнение лабораторных работ.

Инициативная самостоятельная работа с целью реализации студентом собственных учебных и научных интересов, например, участие в олимпиадах, семинарах, конференциях и т.п. - данными рекомендациями не рассматривается.

Основные ***формы*** СРС по дисциплине «Надежность информационных систем» включают:
– подготовку докладов с презентациями, программными примерами и тезисами докладов;
– выполнение в лаборатории по инструкциям лабораторных работ и подготовку отчетов;
- выполнение вне лаборатории лабораторных работ и подготовку отчетов.

1.3. Основные виды и формы контроля СРС

Основные виды и формы контроля СРС по дисциплине «Надежность информационных систем» включают:
- устный опрос;
– оценку докладов;
– защиту лабораторных работ;
- экзамен.

1.4. Список рекомендуемой литературы для СРС

основная

1. Майерс Г. Надежность программного обеспечения. Пер. с англ. Ю.Ю. Галимова; Под ред. В.Ш. Кауфмана. М.:Мир, 1980. - 360 с.

2. Липаев, В. В. Надежность и функциональная безопасность комплексов программ реального времени (для магистров) / В. В. Липаев. — Саратов : Вузовское образование, 2015. — 207 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/27295.html (дата обращения: 05.11.2019). — Режим доступа: для авторизир. пользователей

3. Основы теории надежности информационных систем: учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: ИД «Форум»: ИНФРА_М, 2015. - 256с

4. Богатырев, В. А. Информационные системы и технологии. Теория надежности : учебное пособие для бакалавриата и магистратуры / В. А. Богатырев. — Москва : Издательство Юрайт, 2019. — 318 с. — (Бакалавр и магистр. Модуль). — ISBN 978-5-534-00475-5. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/433723

5. Василенко Н. В. Модели оценки надежности программного обеспечения / Н. В. Василенко, В. А. Макаров // Вестник Новгородского гос. ун-та. – 2004. – № 28. – С. 126–132.

6. Котляров В. П. Основы тестирования программного обеспечения : учеб. пособие / В. П. Котляров, Т. В. Коликова. – М. : БИНОМ. Лаб. знаний : Интернет-Ун-т информ. технологий, 2006. – 285 с.

дополнительная

1. Шураков Виктор Владимирович. Надежность программного обеспечения систем обработки данных. М.:Финансы и статистика, 1987. - 272 с.

2. Надежность автоматизированных систем управления. Автомян И.О. и др.; Под ред. Я.А. Хетагурова. М.: Высшая школа, 1979. - 287 с.

3. Синицын С.В. Верификация программного обеспечения: учебное пособие / Синицын С.В., Н.Ю. Налютин. - Интернет университет информационных технологий. - М. : Интернет-Университет Инф. Технологий : БИНОМ : Лаборатория знаний, 2008. - 304 с. : ил. - (Основы информационных технологий). - ISBN 5-9556-0033-7 (в пер.).— Текст : электронный // https://www.intuit.ru/studies/courses/1040/209/info (дата обращения: 03.11.2019). — Режим доступа: произвольный

учебно-методическая

1. Чекал Елена Георгиевна. Надежность информационных систем : учеб. пособие . Ч. 1 / Чекал Елена Георгиевна, А. А. Чичев; УлГУ, ФМиИТ. - Ульяновск : УлГУ, 2012. - Загл. с экрана; Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 2,79 МБ). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/811

2. Чичев Александр Алексеевич. Операционные системы : учеб.-метод. Пособие.

Ч. 1 : Работа с операционной системой / А. А. Чичев, Е. Г. Чекал; УлГУ, Фак. математики, информ. и авиац. технологий, Каф. информ. технологий. - Ульяновск : УлГУ, 2015. - Загл. с титул. экрана; Электрон. версия печ. публикации. - Электрон. текстовые дан. (1 файл : 1,87 Мб). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/302

Ч. 4 : Работа с операционной системой / А. А. Чичев, Е. Г. Чекал; УлГУ, Фак. математики, информ. и авиац. технологий, Каф. информ. технологий. - Ульяновск : УлГУ, 2019. - Загл. с экрана. - Электрон. текстовые дан. (1 файл : 2,63 Мб). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/4302

3. Чичев Александр Алексеевич. Администрирование информационных систем : учеб.-метод. пособие. Ч. 1 : Общие вопросы / А. А. Чичев, Е. Г. Чекал; УлГУ, ФМИАТ, Каф. информ. технологий. - Ульяновск : УлГУ, 2018. - Загл. с экрана. - Электрон. текстовые дан. (1 файл : 2,12 Мб). - Текст : электронный. http://lib.ulsu.ru/MegaPro/Download/MObject/1377

4. Смагин А.А.Рекомендации поразработке и оформлению рефератов, курсовых, выпускных квалификационных и дипломных работ: учебное пособие /А. А. Смагин, Ю.Д.

Украинцев,А.А. Булаев.–Ульяновск : УлГУ -2019. С. 47.
https://ulsu.ru/media/uploads/mail%40bulalex.ru/2020/01/15/%D0%A0%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BД%D0%B4%D0%B0%D1%86%D0%B8%D0%B8%20%D0%BF%D0%BE%20%D0%BA%D1%83%D1%80%D1%81%20%D0%B8%20%D0%B4%D0%B8%D0%BF%D0%BB.%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%BC.pdf

1.5. Программное обеспечение для СРС

Необходимое программное обеспечение для СРС по данной дисицплине:

1. ОС ALTLinux (open source),
2. IDE NetBeans, Intellij IDEA (open source),
3. СУБД MariaDB, PostgreSQL (open source),
4. Libre Office (open source)

ОС ALTLinux устанавливается с ftp-сервера. ISO-образ версии ОС ALTLinux Kdesktop 7.0.5 копируется со страницы

http://ftp.altlinux.ru/pub/distributions/ALTLinux/p7/images/kdesktop/

Для 32-разрядных машин выбирается

altlinux-7.0.0-kdesktop-i586-ru-install-dvd5.iso

Для 64-разрядных машин выбирается

altlinux-7.0.5-kdesktop-x86_64-ru-install-dvd5.iso

В случае установки более новых версий необходимо будет самостоятельно доустановить компилятор для C/C++, среду Qt Creator и СУБД MariaDB.

1.6. Информационно-справочные системы и базы данных для СРС

1.6.1. Электронно-библиотечные системы:

А.6.1.1. IPRbooks : электронно-библиотечная система : сайт / группа компаний Ай Пи Ар Медиа. - Саратов, [2019]. – URL: http://www.iprbookshop.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

А.6.1.2. ЮРАЙТ : электронно-библиотечная система : сайт / ООО Электронное издательство ЮРАЙТ. – Москва, [2019]. - URL: https://www.biblio-online.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

А.6.1.3. Консультант студента : электронно-библиотечная система : сайт / ООО Политехресурс. – Москва, [2019]. – URL: http://www.studentlibrary.ru/catalogue/switch_kit/x2019-128.html. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

А.6.1.4. Лань : электронно-библиотечная система : сайт / ООО ЭБС Лань. – Санкт-Петербург, [2019]. – URL: http://www.studentlibrary.ru/pages/catalogue.html https://e.lanbook.com. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

А.6.1.5. Znanium.com : электронно-библиотечная система : сайт / ООО Знаниум. - Москва, [2019]. - URL: http://www.studentlibrary.ru/pages/catalogue.html http://znanium.com. – Режим доступа : для зарегистрир. пользователей. - Текст : электронный.

А.6.1.6. ИНТУИТ **[**Электронный ресурс**]** Интернет университет информационных технологий / - Электрон. дан. – Москва, [2019]. - URL : https://www.intuit.ru - Режим доступа: для всех пользователей. - Текст : электронный.

1.6.2. Электронно-правовые системы:

А.6.2.1. КонсультантПлюс [Электронный ресурс]: справочная правовая система. /ООО «Консультант Плюс» - Электрон. дан. - Москва : КонсультантПлюс, [2020].

1.6.3. Базы данных периодических изданий:

А.6.3.1. База данных периодических изданий : электронные журналы / ООО ИВИС. - Москва, [2020]. – URL: https://dlib.eastview.com/browse/udb/12. – Режим доступа : для авториз. пользователей. – Текст : электронный.

А.6.3.2. eLIBRARY.RU: научная электронная библиотека : сайт / ООО Научная Электронная Библиотека. – Москва, [2020]. – URL: http://elibrary.ru. – Режим доступа : для авториз. пользователей. – Текст : электронный

А.6.3.3. «Grebennikon» : электронная библиотека / ИД Гребенников. – Москва, [2020]. – URL: https://id2.action-media.ru/Personal/Products. – Режим доступа : для авториз. пользователей. – Текст : электронный.

1.6.4. Национальная электронная библиотека : электронная библиотека : федеральная государственная информационная система : сайт / Министерство культуры РФ ; РГБ. – Москва, [2020]. – URL:http://www.studentlibrary.ru/pages/catalogue.html https://нэб.рф. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

1.6.5. SMART Imagebase // EBSCOhost : [портал]. – URL: https://ebsco.smartimagebase.com/?TOKEN=EBSCO-1a2ff8c55aa76d8229047223a7d6dc9c&custid=s6895741. – Режим доступа : для авториз. пользователей. – Изображение : электронные.

1.6.6. Федеральные информационно-образовательные порталы:

А.6.6.1. Единое окно доступа к образовательным ресурсам : федеральный портал / учредитель ФГАОУ ДПО ЦРГОП и ИТ. – URL: http://window.edu.ru/. – Текст : электронный.

А.6.6.2. Российское образование : федеральный портал / учредитель ФГАОУ ДПО ЦРГОП и ИТ. – URL: http://www.edu.ru. – Текст : электронный.

1.6.7. Образовательные ресурсы УлГУ:

А.6.7.1. Электронная библиотека УлГУ : модуль АБИС Мега-ПРО / ООО «Дата Экспресс». – URL: http://lib.ulsu.ru/MegaPro/Web. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

А.6.7.2. Образовательный портал УлГУ. – URL: http://edu.ulsu.ru. – Режим доступа : для зарегистр. пользователей. – Текст : электронный

2. Рекомендации по изучению теоретического материала

2.1. Основные понятия теории надежности

Основные вопросы темы:

1. Надежность системы, аппаратная, программная, функциональная и эксплуатационная надежность.
2. Безотказность, ремонтопригодность, долговечность и сохраняемость.
3. Независимый, внезапный, постепенный и перемежающийся отказ (сбой).
4. Наработка до отказа, наработка между отказами.
5. Факторы влияющие на надежность при проектировании, в процессе изготовления эксплуатации.
6. Пути повышения надежности

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.2. Необходимые сведения из теории вероятности и математической статистики

Основные вопросы темы:

1. Схема независимых испытаний Бернулли, случайные величины, закон распределения дискретной случайной величины, плотности распределения непрерывной случайной величины, функция распределения непрерывной случайной величины.
2. Критерии согласия проверки гипотез

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.3. Классификация и расчет показателей надежности ИС

Основные вопросы темы:

1. Определение надежности элемента системы, работающего до первого отказа.
2. Вероятность отказа до определенного момента.
3. Плотность распределения отказов.
4. Вероятность безотказной работы до определенного момента.
5. Функция интенсивности отказов.
6. Примеры наиболее часто используемых законов распределения времени безотказной работы ИС

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.4. Надежность взаимосвязанных элементов системы

Основные вопросы темы:

1. Последовательное и параллельное соединение элементов системы.
2. Мостиковые структуры.
3. Преобразование треугольного соединения элементов в звезду и наоборот

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.5. Классификация методов резервирования. Расчет надежности системы с резервированием

Основные вопросы темы:

1. Постоянное резервирование: общее, поэлементное.
2. Резервирование замещением: нагруженный, облегченный, ненагруженный резерв.
3. Поэлементное резервирование замещением.
4. Резервирование с дробной кратностью и постоянно включенным резервом

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.6. Надежность восстанавливаемых систем

Основные вопросы темы:

1. Восстанавливаемые и не восстанавливаемые системы.
2. Вероятностные модели отказов.
3. Надежность систем с восстановлением.
4. Надежность систем без восстановления

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.7. Качество программных средств ИС

Основные вопросы темы:

1. Определение требований к ПС.
2. Понятие качества ПС.
3. Спецификация качества ПС.
4. Функциональная спецификация.
5. Методы спецификации семантики функций

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.8. Факторы, определяющие надежность ПС. Показатели надежности ПС

Основные вопросы темы:

1. Общие принципы обеспечения надежности ПС.
2. Обеспечение завершенности, точности, автономности, устойчивости ПС.
3. Обеспечение защищенности ПС: защита от сбоев аппаратуры, от влияния "чужой" программы, защита от влияния "своей" программы, от ошибок оператора, защита от НСД, защита от защиты.
4. Показатели надежности ПС.
5. Математические модели надежности комплексов программ

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.9. Методы проектирования надежного ПС

Основные вопросы темы:

1. Понятие архитектуры ПС. Основные классы архитектур ПС.
2. Архитектурные функции. Контроль архитектуры ПС.
3. Основные характеристики программного модуля.
4. Современные методы разработки структуры программы.
5. Контроль структуры программы)

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.10. Методы создания надежного ПС

Основные вопросы темы:

1. Порядок разработки программного модуля.
2. Структурное программирование.
3. Пошаговая детализация и понятие о псевдокоде.
4. Оптимизация программ.
5. Контроль программного модуля

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]

[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.11. Методы тестирования и отладки надежного ПС

Основные вопросы темы:

1. Понятие обоснования программ.
2. Тестирование ПС. Организация тестирования.
3. Тестирование архитектуры, качества, документации, требований к ПС.
4. Принципы и виды отладки ПС. Организация отладки ПС.
5. Автономная отладка. Комплексная отладка

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.12. Надежность оператора

Основные вопросы темы:

1. Основные понятия. Виды отказов оператора.
2. Методы прогнозирования надежности оператора.
3. Методы повышения надежности оператора

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.13. Надежность сетей передачи данных

Основные вопросы темы:

1. Процедуры повышенной надежности доставки данных.
2. Логико-вероятностный расчет надежности передачи информации в типовых сетевых структурах.
3. Методы расчета показателей надежности двухполюсной сети связи

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.14. Методы обеспечения надежности ИС на этапах жизненного цикла

Основные вопросы темы:

1. Проектирование информационных систем и надежность.
2. Предварительный анализ надежности.
3. Детальная разработка технологической и эксплуатационной документации.
4. Ресурсы необходимые для обеспечения надежности.

5. Средства встроенного контроля процесса функционирования информационных систем.

6. Особенности обеспечения надежности функционирования информационных систем на этапе разработки.

7. Организация службы эксплуатации.

8. Планирование профилактического обслуживания системы и предотвращения ее износа

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

2.15. Взаимосвязь надежности, безопасности и экономической эффективности ИС

Основные вопросы темы:

1. Основные понятия и факторы, определяющие безопасность информационных систем.

2. Ресурсы для обеспечения функциональной безопасности ИС.

3. Разработка требований к функциональной безопасности ИС.

4. Основные понятия и определения экономической эффективности информационных систем.

5. Взаимосвязь надежности и функциональной безопасности ИС.

6. Влияние надежности ИС на экономическую эффективность ИС

Рекомендации по изучению темы:

[1, Майерс Г.]
[2, Липаев В.В.]
[3. Симонов В.Л.]
[4. Богатырев В.А.]

3. Рекомендации к выполнению лабораторных работ

3.1. Подготовка программной среды

Для выполнения лабораторных работ необходимо установить ОС ALTLinux Kdesktop 7.0.5 и среды разработки (см. п.1.5).

3.2. Лабораторная работа № 1

Тема: Расчет показателей надежности ИС

Цель: Научиться расчитывать показатели надежности ИС

Варианты:

1. Информационная система состоит из трех устройств, причем отказ любого из них приводит к отказу всей системы. Интенсивность отказов первого устройства равна $\lambda_1 = 0{,}16 \cdot 10^{-3}\ 1/час$, второго $\lambda_2 = 0{,}23 \cdot 10^{-4}\ 1/час$, третьего $\lambda_3 = 0{,}06 \cdot 10^{-6}\ 1/час$. Найти вероятность безотказной работы системы в течении 100 часов, среднее время безотказной работы системы.

2. Информационная система состоит из трех устройств, причем отказ любого из них приводит к отказу всей системы. Среднее время безотказной работы устройств соответственно равно: 160час, 320 час, 600 час. Для устройств справедлив экспоненциальный закон надежности. Найти среднее время безотказной работы системы.

3. Информационная система состоит из двух устройств, причем работоспособность любого из них гарантирует работу всей системы. Вероятности безотказной работы каждого из них в течении 100 часов соответственно равны: 0,95 и 0,97. Справедлив экспоненциальный закон надежности. Найти среднее время безотказной работы системы.

4. Вероятность безотказной работы одного устройства системы равно 0.997. Найти вероятность безотказной работы и вероятность отказа системы, состоящей из 10 таких устройств, причем работоспособность любого из них гарантирует работу всей системы.

5. Интенсивность отказов структурированной кабельной системы ИС имеет вид:

$$\lambda(t) = k(1 - \exp(-kt))/(1 - 0.5\exp(-kt))$$

Требуется определить вероятность безотказной работы, плотность распределения отказов.

6. В результате анализа протокола отказов структурированной кабельной системы ИС установлено, что вероятность безотказной работы выражается формулой

$$P(t) = 3e^{-\lambda t} - 3e^{-2\lambda t} + e^{-3\lambda t}$$

Требуется определить интенсивность отказов, наработку на отказ.

7. Время безотказной работы устройства подчиняется закону Вейбулла с параметрами $\lambda = 0{,}16 \cdot 10^{-4}\ 1/час$, k=1.5, t=100 час. Требуется определить вероятность безотказной работы, интенсивность отказов.

8. Время безотказной работы структурированной кабельной системы ИС подчиняется экспоненциальному закону с параметром $\lambda = 0{,}2 \cdot 10^{-5}\ 1/час$. Время работы 20000час. Требуется определить вероятность отказа, интенсивность отказов, наработку до первого отказа при 800, 1000, 2000 часов.

9. 7. Время безотказной работы устройства подчиняется закону Рэлея с параметром $d = 10час$. Требуется определить вероятность безотказной работы, интенсивность отказов, наработку на отказ, плотность распределения отказов для t=1000 час.

10. Вероятность безотказной работы устройства в течение 120 часов равна 0.9. Предполагается, что справедлив экспоненциальный закон надежности. Требуется определить вероятность безотказной работы, интенсивность отказов, наработку на отказ, плотность распределения отказов.

11. Вероятность безотказной работы системы в течение 1000 часов равна 0.95. Система состоит из 120 равнонадежных элементов, причем отказ любого из них приводит к отказу всей системы. Найти вероятность безотказной работы одного элемента.

12. Система состоит из 127 устройств, средняя интенсивность отказов которых $\lambda cp = 0{,}32 \cdot 10^{-6}\ 1/час$. Причем отказ любого из них приводит к отказу всей системы. Найти вероятность безотказной работы системы для 500 часов.

13. В результате анализа протокола отказов структурированной кабельной системы ИС установлено, что вероятность безотказной работы выражается формулой

$$P(t) = e^{-2\lambda t} + 4e^{-5\lambda t} + e^{-\lambda t}$$

Требуется определить наработку на отказ, плотность распределения отказов

14. . Интенсивность отказов структурированной кабельной системы ИС имеет вид:

$$\lambda(t) = 2(1 - \exp(-2t))/(1 - 0.8\exp(-2t))$$

Требуется определить вероятность безотказной работы, наработку на отказ.

15. При проведении испытаний устройства получены следующие данные:

для t = 200часов $\lambda = 0{,}4 \cdot 10^{-4}\ 1/час$,

для t = 400часов $\lambda = 0{,}8 \cdot 10^{-4}\ 1/час$,

для t = 600часов $\lambda = 1{,}2 \cdot 10^{-4}\ 1/час$,

для t = 800часов $\lambda = 1{,}6 \cdot 10^{-4}\ 1/час$,

для t = 1000часов $\lambda = 2{,}0 \cdot 10^{-4}\ 1/час$,

Найти вероятность безотказной работы, интенсивность отказов, наработку на отказ, плотность распределения отказов для t = 1000часов.

16. Время безотказной работы устройства подчиняется закону Вейбулла с параметрами $\lambda = 0{,}4 \cdot 10^{-4}\ 1/час$, k=1.2, t=100 час. Требуется определить наработку на отказ, плотность распределения отказов.

17. Система состоит из 12 устройств, средняя интенсивность отказов которых $\lambda ср = 0{,}1 \cdot 10^{-6}\ 1/час$. Причем работоспособность любого из них гарантирует работу всей системы. Найти вероятность безотказной работы системы для 200 часов.

18. Вероятность безотказной работы устройства в течение 100 часов равна 0.89. Предполагается, что справедлив закон надежности Рэлея. Требуется определить вероятность безотказной работы, интенсивность отказов, наработку на отказ, плотность распределения отказов.

19. Вероятность безотказной работы одной ПЭВМ системы равно 0.997. В системе есть один сервер, двукратно зарезервированный. Найти вероятность безотказной работы и вероятность отказа системы, состоящей из 10 таких устройств, причем работоспособность любого из них гарантирует работу всей системы.

20. Информационная система имеет файл-серверную архитектуру. Сервер трехкратно зарезервирован. Вероятность безотказной работы одной ПЭВМ системы равно 0.9, сервера 0.98. Найти вероятность безотказной работы системы, имеющей четыре автоматизированных рабочих места.

21. Информационная система имеет клиент-серверную трехзвенную архитектуру. Сервера двукратно зарезервированы. Вероятность безотказной работы одной ПЭВМ системы равно 0.9, серверов 0.98. Найти вероятность безотказной работы системы, имеющей 15 автоматизированных рабочих места.

22. Время безотказной работы ТС подчиняется закону Рэлея с параметром d=10 час. Определить вероятность безотказной работы, интенсивность отказов, наработку на отказ для Т=1000 часов.

Порядок сдачи лабораторной работы:

В отчёте, созданном в текстовом процессоре Libre Office Writer, должно быть:

а) задание на лабораторную работу;

б) подробное решение.

Титульный лист отчета приведен в приложении 1.

По требованию преподавателя дать ответы на вопросы по работе

Срок сдачи лабораторной — до

4.3. Лабораторная работа 2

Тема: Расчет надежности системы с резервированием
Цель: Научиться рассчитывать показатели надежности ТС при резервировании

Варианты:

По заданной структурной схеме надежности ТС ИС в соответствии с вариантом задания рассчитать вероятность безотказной работы до резервирования и после резервирования при $T=10^6$ часов. Резервировать выделенные элементы.

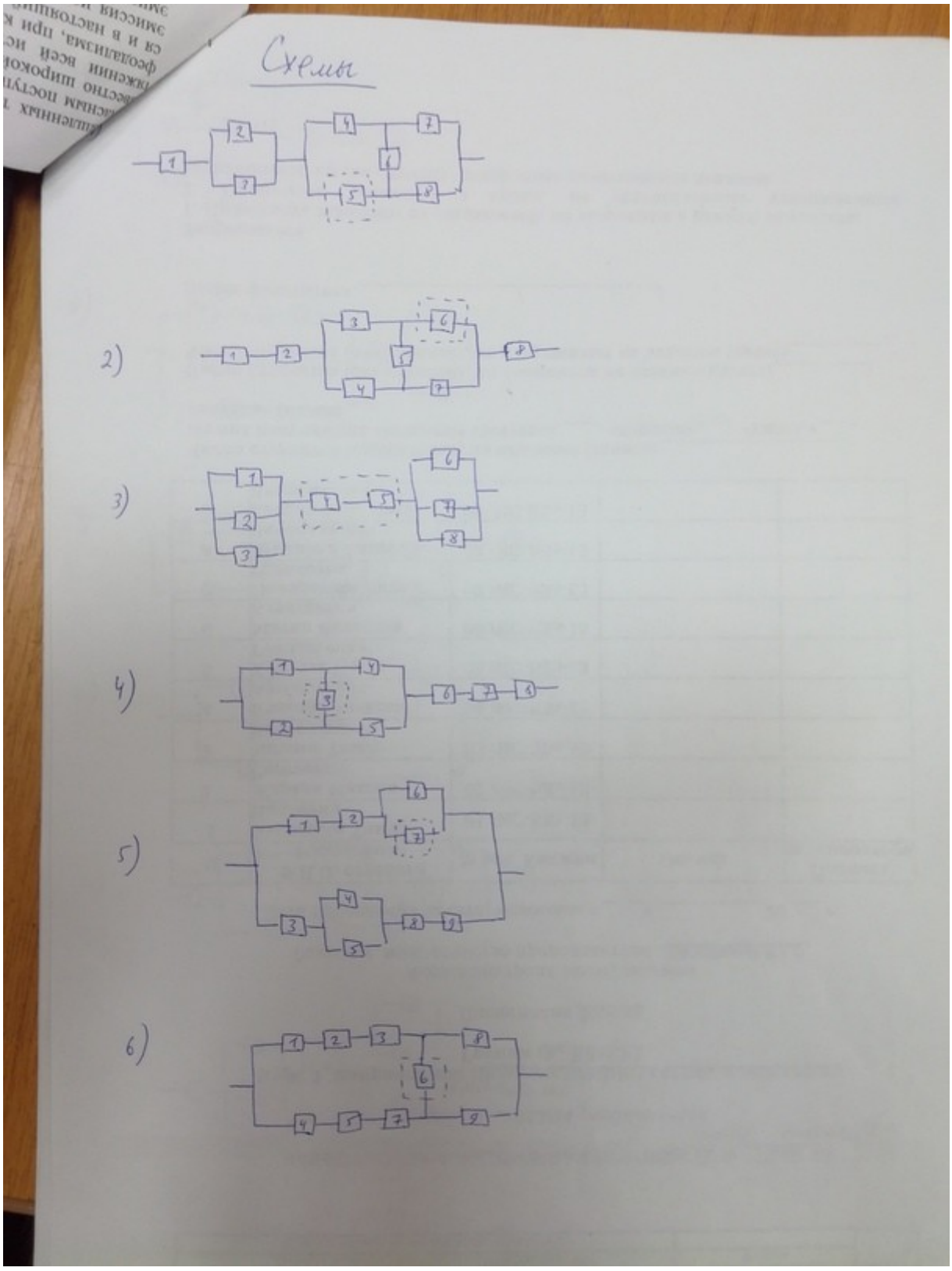
Сравнить рассчитанные значения. Предполагается, что все элементы системы работают в нормальном режиме эксплуатации (интенсивность отказов постоянна). Резервирование осуществлять идентичными по надежности резервными элементами. Переключатели при резервировании считаются идеальными. значения интенсивности отказов элементов приведены в таблице.

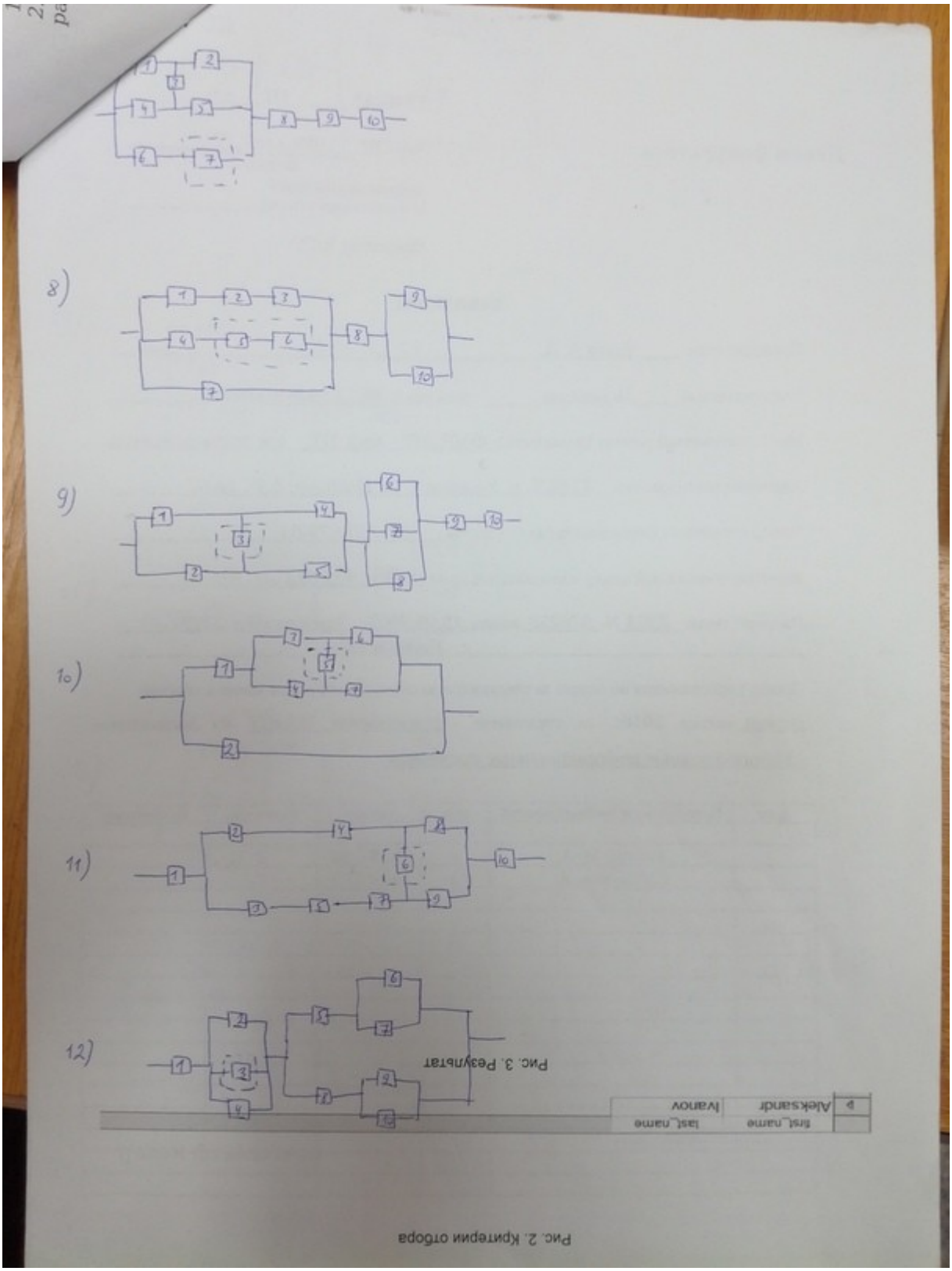
Таблица

Численные значения параметров к заданию

№ вар.	Интенсивность отказов элементов, λ ($\lambda \cdot 10^{-6}$ 1/час)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.1	1.0				0.5	1.0						0.1		
2	0.2	0.5								1.0					0.1
3	0.1	1.0				2.0		1.0				5.0			0.2
4	0.05	1.0					0.5				0.2				0.02
5	0.01						0.5				1.0				
6	0.01											0.05		0.1	-
7	0.05	0.5				0.05			0.005		0.1	0.2		0.1	-
8	0.1	0.5			0.2			0.01			0.5			0.1	-
9	0.03	0.5			0.2			1.0			0.03			0.1	-
10	0.1	0.5				1.0		0.5				1.0		0.1	-
11	0.05	0.2		0.5									0.2		0.1
12	0.02	0.1		1.0					2.0				0.1		0.05
13	0.01	0.2				0.1		1.0			0.5			0.1	-
14	0.01	0.1		10.0				0.2		10.0			0.5		-
15	0.01	1.0		5.0					0.2		5.0			0.1	-
16	0.1	1.0		2.0	1.0		5.0			3.0			1.0		0.05

17	0.1	5.0	1.0	5.0	10.0	5.0	1.0	0.2						
18	0.01	1.0											0.1	-
19	0.1	5.0	0.5	5.0	1.0	3.0	1.0	5.0	0.5	5.0				
20	0.1	10.0		20.0				10.0						
21	0.1	1.0		0.5		2.0	0.5	0.2	1.0					
22	1.0			0.2	0.5	1.0	0.5	1.0	1.0	0.1				
23	0.5	0.2	1.0	0.5	1.0	0.5	1.0	0.2	0.5	1.0	0.2			
24	1.0	2.0		4.0	2.0	4.0	5.0	1.0						
25	0.5	10.0		0.5	5.0	0.8	5.0	1.0	5.0					
26	1.0	2.0	3.0	5.0	2.0	5.0	1.0							
27	5.0	10.0		15.0	10.0	10.0	15.0	10.0						
28	1.0	2.0		5.0	2.0	1.0								
29	5.0	20.0		50.0	30.0	1.0								
30	2.0	1.0	2.0	1.0	5.0	2.0	5.0	2.0	1.0	2.0	1.0	2.0	1.0	
31	2.0	1.0	2.0	1.0	5.0	2.0	5.0	2.0	1.0	2.0	1.0	2.0	1.0	
32	5.0	2.0		5.0	1.0	2.0	3.0	1.0						
33	1.0	2.0	3.0	4.0	2.0	3.0	5.5	0.2	0.5					
34	6.0	3.0	6.0	3.0	6.0	20.0	10.0							
35	1.0	2.0	1.0	2.0	1.0	5.0								
36	2.0			1.0	0.6									
37	10.0			30.0	5.0	2.0								
38	3.0	2.0		1.0	2.0	3.0	2.0							
39	8.0		3.0		5.0	2.0								
40	2.0		5.0	8.0	2.0	5.0	8.0							
№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
вар.	Интенсивность отказов элементов, λ , ($\lambda \cdot 10^{-6}$ 1/час)														


Схемы
2)
3)
4)
5)
6)



Порядок сдачи лабораторной работы:

В отчёте, созданном в текстовом процессоре Libre Office Writer, должно быть подробное решение с обоснованием выводов.
Титульный лист отчета приведен в приложении 1.
По требованию преподавателя дать ответы на вопросы по работе

Срок сдачи лабораторной — до

4.4. Лабораторная работа 3

Тема: Автоматизированное тестирование веб-сайтов в сети Интернет
Цель: Познакомиться с системой автоматизированнооого тестирования веб-сайтов

Задание

Разработать скрипт, позволяющий открыть определенный веб-сайт. зарегистрироваться на нем. Зайти под созданной учетной записью. Осуществить правильные и неправильные действия.

Порядок сдачи лабораторной работы:

В отчёте, созданном в текстовом процессоре Libre Office Writer, должен быть написан скрипт, который выполняется в лаборатории.
Титульный лист отчета приведен в приложении 1.
По требованию преподавателя дать ответы на вопросы по работе

Срок сдачи лабораторной — до

4.5. Лабораторная работа 4

Тема: Разработка тест-монитора в Qt Creator
Цель: Сформировать навыки разработки тест-монитора

Задание

Разработать программу (тест-монитор), осуществляющую:
1)
- генерацию тестов, числом не менее 100, в том числе случайным образом.;
- инициацию тестируемой программы как независимого процесса;
- передачу на вход тестируемой программы тестов;
- прием выходных данных;
- создание протокола тестирования в виде текстового файла;
2)
-запись протокола тестирования в СУБД MySQL;
- выбор информации о проведенных процессах тестирования для расчета показателей надежности;
- расчет показателей надежности тестируемой программы по одной из указанных в вариантах методик;
3)
- серию процессов тестирования;
- периодический выбор информации о проведенных процессах тестирования для расчета показателей надежности;
- расчет показателей надежности тестируемой программы по одной из указанных в вариантах методик;
- выдачу таблицу с информацией о номере серии испытаний и значению показателей надежности.

Тест-монитор разрабатываются в среде **IDE Qt Creator на языке C++ с использованием классов библиотеки Qt либо на JAVA в IDE NetBeans или Intellij IDEA.**

Варианты тестируемых программ сортировки:

1. Бинарная сортировка
2. Быстрая сортировка
3. Метод двоичного включения
4. Метод прямого включения
5. Метод прямого выбора
6. Метод пузырька
7. Сортировка перечислением
8. Сортировка слиянием
9. Сортировка Шелла
10. Шейкерная сортировка

Варианты методик расчета надежности:

1.Модель Нельсона-Коркорэна
2. Модель Холстеда
3. Модель Милса
4. Модель Муса
5. Модель Шумана

Варианты заданий лабораторных работ

№	сортировка, методика расчета надежности	№	сортировка, методика расчета надежности	№	сортировка, методика расчета надежности
1	1,4	11	1,2	21	1,3
2	2,3	12	2,1	22	2,2
3	3,2	13	3,4	23	3,1
4	4,1	14	4,3	24	4,4
5	5,4	15	5,2	25	5,3
6	6,3	16	6,1	26	6,2
7	7,2	17	7,4	27	7,1
8	8,1	18	8,3	28	8,4
9	9,4	19	9,2	29	9,3
10	10,3	20	10,1	30	10,2

Порядок сдачи лабораторной работы:

1) Задание сдается по частям. Демонстрация функционирования тест-монитора в лаборатории. Доработка или исправление по указанию преподавателя.

2) В отчёте должно быть

а) задание;

б) исходные тексты тест-монитора и тестируемой программы.

3) В электронном виде сдаётся исполняемый файл + исходники.

В отчёте, созданном в текстовом процессоре Libre Office Writer, должно быть программное средство.

Титульный лист отчета приведен в приложении 1.

По требованию преподавателя повторить работу в лаборатории и дать ответы на вопросы по работе

Срок сдачи лабораторной — до

4.6. Лабораторная работа 5

Тема: Разработка программ создания ЭЦП/проверки ЭЦП.
Цель: Сформировать навыки разработки программ электронной подписи

Задание

Написать консольную программу на языке С в IDE Qt Creator.

Составить руководство оператора для этой программы.

Общие требования:

- все создаваемые типы, поля, события должны сопровождаться комментариями;

- деление на модули обязательно;

- разделение модулей на заголовочные файлы и файлы реализаций обязательно;

- стандартную библиотеку C++ использовать запрещено;

- программа выполняется в консольном режиме;

- для всех лабораторных работ требуется создание диалогового интерфейса, реализующего выполнение операций по выбору пользователя;

- изменять вариант без согласования с преподавателем запрещено.

Дополнительная информация

1. Алгоритм программы создания ЭЦП.

1.1. Требования.

1.1.1. Программа должна быть написана на языке С в Linux. Для разработки может использоваться любой дисприбутив Linux, но программа должна быть работоспособна в AltLinux.

1.1.2. В качестве сред разработки может быть использовано:

a) IDE Eclipse.

1.1.3. Дабы не углубляться в криптографию, в программе предлагается использовать достаточно простой алгоритм формирования ЭЦП.

1.1.4. Программа должна обрабатывать документы (документ — это файл) объёмом до 8 (восьми) мегабайт.

1.1.5. Интерфейс с пользователем - текстовый.

1.1.6. Итак, нам потребуется зона памяти, в которую будем считывать подписываемый документ и с которой будем работать. Опишем её следующим образом:

```
union ecp_tip
  { unsigned long int ecpl [2097152];
    char ecpc [8388608];
  };
```

1.2. Диалог программы. Вводим:

- ФИО полностью в переменные strf, stri, strf;

- ключ в поле keyc следующей структуры

union key_tip

```
{ unsigned long int keyl;
  char keyc[4];
};
```

- имя подписываемого файла в переменную namefile.

Ключ вводим как четыре символа, причём при вводе ключа желательна проверка, что символы вводятся разные.

1.3. Открываем файл с документом и считываем его в ecpc потоком, то есть, вместе с символами «конец строки», «перевод каретки», прочими символами форматирования и вообще любыми символами, которые есть в файле. То есть, читать, пока не конец файла, но не более 8 мегабайт. При чтении подсчитываем количество прочтённых байт, запоминаем в size_ecpc и проверяем, что их не более 8 млн. штуков.

1.4. В конец ecpc добавляем ФИО:

```
strcat (ecpc, '\n');
strcat (ecpc, strf);  // фамилия
strcat (ecpc, '  ');
strcat (ecpc, stri);  // имя
strcat (ecpc, '  ');
strcat (ecpc, stro); // отчество
```

1.5. В конец ecpc добавляем дату и время подписания документа:

```
strcat (ecpc, '\n');
strcat (ecpc, date(...));              //на самом деле, здесь не так немножко
strcat (ecpc, '\n');
```

1.6. Вычисляем, сколько полных unsigned long int помещается в ecpc. Запоминаем это число n. Определяем остаток от деления size_ecpc/n. Если остаток не равен 0, тогда последний неполный unsigned long int дополняем нулями (двоичными!). Получаем n+1 полных long'ов.

1.7. Вычисляем hash-функцию следующим образом (это пример, варианты приведены ниже):

$$hash = \sum_{i=1}^{n+1} ecpl[i]*i$$

1.8. Складываем побитово два unsigned числа:

```
ecp = hash | keyl;
```

1.9. Итого, мы получили ЭЦП. Добавляем полученную ecp к файлу подписываемого документа:

```
strcat (ecpc, ecp);
strcat (ecpc, '\n');
```

1.10. Сохраняем ecpc в файл с именем: namefile-ecp.«тип_файла_тот-же».

2. Алгоритм программы проверки ЭЦП.

2.1. Аналогичен п. 1.1.

2.2. Диалог программы. Вводим:

- ключ в переменную

union key_tip
{ unsigned long int keyl;
char keyc[4];
};

- имя полученного от адресата файла в переменную namefile.

2.3. Аналогичен п. 1.3.

2.4. Удаляем из ecpc последний перевод строки и цифровую подпись ecp и сохраняем её.

2.5. Вычисляем n+1 по п. 1.6.

2.6. Вычисляем hash-функцию по п. 1.7.

2.7. Складываем побитово два unsigned числа:

ecp1 = hash | keyl;

2.8. Сравниваем ecp и ecp1.

Равно? Тогда «Документ неизменён.».

Иначе «Документ исправлен!».

Варианты hash-функций:

Используемая таблица

```
static const unsigned char sTable[256] =
{
  0xa3,0xd7,0x09,0x83,0xf8,0x48,0xf6,0xf4,0xb3,0x21,0x15,0x78,0x99,0xb1,0xaf,0xf9,
  0xe7,0x2d,0x4d,0x8a,0xce,0x4c,0xca,0x2e,0x52,0x95,0xd9,0x1e,0x4e,0x38,0x44,0x28,
  0x0a,0xdf,0x02,0xa0,0x17,0xf1,0x60,0x68,0x12,0xb7,0x7a,0xc3,0xe9,0xfa,0x3d,0x53,
  0x96,0x84,0x6b,0xba,0xf2,0x63,0x9a,0x19,0x7c,0xae,0xe5,0xf5,0xf7,0x16,0x6a,0xa2,
  0x39,0xb6,0x7b,0x0f,0xc1,0x93,0x81,0x1b,0xee,0xb4,0x1a,0xea,0xd0,0x91,0x2f,0xb8,
  0x55,0xb9,0xda,0x85,0x3f,0x41,0xbf,0xe0,0x5a,0x58,0x80,0x5f,0x66,0x0b,0xd8,0x90,
  0x35,0xd5,0xc0,0xa7,0x33,0x06,0x65,0x69,0x45,0x00,0x94,0x56,0x6d,0x98,0x9b,0x76,
  0x97,0xfc,0xb2,0xc2,0xb0,0xfe,0xdb,0x20,0xe1,0xeb,0xd6,0xe4,0xdd,0x47,0x4a,0x1d,
  0x42,0xed,0x9e,0x6e,0x49,0x3c,0xcd,0x43,0x27,0xd2,0x07,0xd4,0xde,0xc7,0x67,0x18,
  0x89,0xcb,0x30,0x1f,0x8d,0xc6,0x8f,0xaa,0xc8,0x74,0xdc,0xc9,0x5d,0x5c,0x31,0xa4,
  0x70,0x88,0x61,0x2c,0x9f,0x0d,0x2b,0x87,0x50,0x82,0x54,0x64,0x26,0x7d,0x03,0x40,
  0x34,0x4b,0x1c,0x73,0xd1,0xc4,0xfd,0x3b,0xcc,0xfb,0x7f,0xab,0xe6,0x3e,0x5b,0xa5,
  0xad,0x04,0x23,0x9c,0x14,0x51,0x22,0xf0,0x29,0x79,0x71,0x7e,0xff,0x8c,0x0e,0xe2,
  0x0c,0xef,0xbc,0x72,0x75,0x6f,0x37,0xa1,0xec,0xd3,0x8e,0x62,0x8b,0x86,0x10,0xe8,
  0x08,0x77,0x11,0xbe,0x92,0x4f,0x24,0xc5,0x32,0x36,0x9d,0xcf,0xf3,0xa6,0xbb,0xac,
  0x5e,0x6c,0xa9,0x13,0x57,0x25,0xb5,0xe3,0xbd,0xa8,0x3a,0x01,0x05,0x59,0x2a,0x46
};
```

1. MaPrime2c

В качестве подсистемы рассеивания здесь используется таблица замены (подстановки), которая инициирует массивное обращение к памяти. На аппаратных решениях это может быть выполнено весьма быстро, но программные реализации могут "проседать". Конструктивной причиной различия в производительности является еще и побайтовое сжатие, в отличие от 32-разрядного во многих из наиболее распространенных. Собственно этот момент и имеет две стороны - хорошее распределение и пропорциональное снижение производительности.

Если брать на вскидку актуальный список простых хеш функций общего назначения, где обычно присутствует и MaPrime2c, то мы увидим последнюю как-раз таких ближе к концу списка. По качеству распределения она будет одной из первых, по скорости - из последних. Почему так? Всё потому, что сейчас акцент ставится именно на скорости, соответственно и оптимизация смотрит именно в этом направлении

```
#define PRIME_MULT 1717
unsigned int maPrime2cHash (unsigned char *str, unsigned int len)
{
  unsigned int hash = len, i;

  for (i = 0; i != len; i++, str++)
    {
      hash ^= sTable[( *str + i) & 255];
      hash = hash * PRIME_MULT;
    }
  return hash;
}
```

2. MaFastPrime1

Указанный далее алгоритм сочетает в себе элементы конструкции MaPrime2c и новомодный базис некриптографических хешей - обработку блоками в 4 байта. Как видно, подход остался прежним, за исключением табличной замены, которая теперь отсутствует. Хвост, то есть не кратные 4 байтам данные, сжимаются отдельно. В этом случае применяется классический подход. Что мы имеем в результате? Кто пан, а кто пропал - покажут тесты, но пока можно сказать одно. Алгоритм конструктивно быстрее, особенно на блоках кратных 32-м битам. На остальных наборах данных замедление не будет принципиально ощутимо, поскольку блоки будут невелики - не более трех байт.

```
unsigned int maFastPrime1Hash(char *str, unsigned int len)
{
	unsigned int hash = len, i = 0, k;
	long rem = len;
	unsigned char trail;
	const unsigned char * data = (const unsigned char *)str;

	while (rem >= 4)
	   {
		k = *(unsigned int*)data;
		k += i++;
		hash ^= k;
		hash *= 171717;
		data += 4;
		rem -= 4;
	}
	while (rem >= 0) {
		trail = *(unsigned char*)data;
		trail += i++;
		hash ^= trail;
		hash *= 171717;
		data++;
		rem--;
	}
	return hash;
}
```

3. MaRushPrime1

Как известно, 32-х разрядная модификация классического алгоритма MaPrime2c сочетает значительно увеличенную производительность (ввиду перехода на 4-х байтные блоки) и традиционную прозрачную структуру, основанную на простых математических операциях. Сам алгоритм не только не усложнился, но и даже "похудел" ввиду отсутствия таблицы подстановок. Безусловно, переход на 32-х разрядную обработку не мог сказаться на числе коллизий в итоге, за низкое количество которых мы так любим MaPrime2c. Будем считать это маленьким,

необходимым злом. Но можно ли еще его ускорить саму функцию? Попробуем. Рассмотренный ранее FastPrime1c имеет одно особенность - там применяется побайтовый просмотр для некратных данных в конце строки. Что если мы будем использовать все тот же блок, забитый пустыми значениям до кратного? Скажется ли это на результате? Безусловно. В плане качества распределения FastPrime1 будет лучше, но, вероятно, масштаб новоприобретенных коллизий не будет столь велик?

```
unsigned int maRushPrime1Hash(char *str, unsigned int len)
{
	unsigned int hash = len, i = 0, k;
	long rem = len;

	const unsigned char * data = (const unsigned char *)str;

	while (rem >= 4) {
		k = *(unsigned int*)data;
		k += i++;
		hash ^= k;
		hash *= 171717;
		data += 4;
		rem -= 4;
	}
	switch (rem) {
	case 3:
		k = (unsigned int)(data[0]) | (unsigned int)(data[1] << 8) |
			(unsigned int)(data[2] << 16);
		k += i++;
		hash ^= k;
		hash *= 171717;
		break;

	case 2:
		k = (unsigned int)(data[0]) | (unsigned int)(data[1] << 8);
		k += i++;
		hash ^= k;
		hash *= 171717;
		break;

	case 1:
		k = (unsigned int)(data[0]);
		k += i++;
		hash ^= k;
```

```
            hash *= 171717;
            break;
        }
        return hash;
}
```

4. MaHash7

Что отличает эту функцию? Правильно, бросается в глаза таблица замены (подстановки). Зачем она в таком простом алгоритме? Прежде всего затем, что таблица замены - это сложная функция, которую просто описали несколько иным способом. При этом такую функцию очень легко реализовать как программно, так и аппаратно, скорость запроса к таблице высока в обоих случаях. У функции есть параметр - число итераций. Статистически, лучшим параметром будет одна итерация для большинства текстов. Хотя для оригинальной функции оптимальными будут три итерации. Сама таблица подстановки идентична таковой в криптоалгоритме Skipjack. Результаты не лучшие, но конструкция имеет право на жизнь.

```
#define LROT14(x) (((x) << 14) | ((x) >> 18 ))
#define ITERATIONS 1

unsigned int MaHash7 (unsigned char *str, unsigned int len)
{
  unsigned int hash = len, i;

  for (i = 0; i != len * ITERATIONS; i++)
    {
     str += (i % len);
      hash = LROT13 (hash + ((hash << 8) ^ (hash >> 12))) - sTable[(*str + i) & 255];
    }
  return hash;
}
```

5. MaHash9

Другие константы сдвигов и циклический сдвиг влево на 14 бит, дополнительная операция побитового НЕ. Как результат - немного лучше показатели в ряде тестов, но в итоге изменения себя не оправдали. В итоге - далеко не первое место, так же - твердый середнячок.

```
#define LROT14(x) (((x) << 14) | ((x) >> 18 ))


unsigned int MaHash9 (unsigned char *str, unsigned int len)
{
   unsigned int hash = len, i;
   for (i = 0; i != len; i++, str++)
   {
      hash = LROT14( ~hash + ((hash << 6) ^ (hash >> 11))) - sTable[(*str + i) & 255];
   }
   return hash;
}
```

6. MaHash2

По-сути, это два алгоритма MaHash для параллельного подсчета двух 32-разрядных значений. На каждом этапе результат вычисления оригинальной и видоизмененной функции с обратным циклическим сдвигом вправо взаимозаменяются. Результатом хэш-функции будет сумма подфункций по модулю 2 (операция XOR). Такая конструкция позволила улучшить результаты алгоритма в плане числа коллизий. Достоинства алгоритма - отсутствие операций деления, умножения (наиболее медленных на современных неспециализированных процессорах) и возможность распараллеливания. С неба звезд не хватает, но показывает сравнительно хорошие результаты - восьмое место в рейтинге с небольшим отставанием от ближайших соперников.

```
#define LROT(x) (((x) << 11) | ((x) >> 21 ))
#define RROT(x) (((x) << 21) | ((x) >> 11 ))

unsigned int MaHash2 (unsigned char *str, unsigned int len)
{
   unsigned int t, hash1 = len, hash2 = len, i;
   for (i = 0; i != len; i++, str++)
   {
      hash1 += sTable[(*str + i) & 255];
      hash1 = LROT(hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      hash2 += sTable[(*str + i) & 255];
      hash2 = RROT(hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
      t = hash1;
      hash1 = hash2;
      hash2 = t;
   }
   return hash1 ^ hash2;
}
```

7. MaHash8

Изменились сдвиги, циклический сдвиг увеличился не несколько разрядов. На смену взаимозамены результатов подфункций теперь они “собираются” из младшего родного слова и старшего слова альтернативной функции. Показал в среднем лучшие среди участников результаты.

```
#define LROT14(x) (((x) << 14) | ((x) >> 18 ))
#define RROT14(x) (((x) << 18) | ((x) >> 14 ))
unsigned int MaHash8 (unsigned char *str, unsigned int len)
{
   unsigned int sh1, sh2, hash1 = len, hash2 = len, i;
   for (i = 0; i != len; i++, str++)
   {
      hash1 += sTable[(*str + i) & 255];
      hash1 = LROT14(hash1 + ((hash1 << 6) ^ (hash1 >> 11)));
      hash2 += sTable[(*str + i) & 255];
      hash2 = RROT14(hash2 + ((hash2 << 6) ^ (hash2 >> 11)));
      sh1 = hash1;
      sh2 = hash2;
      hash1 = ((sh1 >> 16) & 0xffff) | (sh2 & 0xffff) << 16;
      hash2 = ((sh2 >> 16) & 0xffff) | (sh1 & 0xffff) << 16;
   }
   return hash1 ^ hash2;
}
```

8. MaHash4

Да, здесь опять классические сдвиги и обычный циклический сдвиг. Только вместо взаимозамены опять “сборка” значений. Эта функция, как и прошлая, стабильно показывает очень хорошие результы, не имеет “провалов”. Не даром заняла второе место в общем зачете и победила в сокращенном тесте.

```
#define LROT(x) (((x) << 11) | ((x) >> 21 ))

unsigned int MaHash4 (unsigned char *str, unsigned int len)
{
  unsigned int sh1, sh2, hash1 = len, hash2 = len, i;
  for (i = 0; i != len; i++, str++)
    {
      hash1 += sTable[(*str + i) & 255];
      hash1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      hash2 += sTable[(*str + i) & 255];
      hash2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
      sh1 = hash1;
      sh2 = hash2;
      hash1 = ((sh1 >> 16) & 0xffff) | (sh2 & 0xffff) << 16;
      hash2 = ((sh2 >> 16) & 0xffff) | (sh1 & 0xffff) << 16;
    }
  return hash1 ^ hash2;
}
```

9. MaHash5

Добавлено дополнительное преобразование в конце шага. Результаты стабильны для обоих тестов, хотя в сравнении с прошлой функцией усложнение, которое немного улучшило результат первого теста, не оправдано. Как результат, пятое место в зачете и фактическая несостоятельность.

```
#define LROT(x) (((x) << 11) | ((x) >> 21 ))

unsigned int MaHash5 (unsigned char *str, unsigned int len)
{
  unsigned int sh1, sh2, hash1 = len, hash2 = len, i;
  for (i = 0; i != len; i++, str++)
    {
      hash1 += sTable[(*str + i) & 255];
      hash1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      hash2 += sTable[(*str + i) & 255];
      hash2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
      sh1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      sh2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));

      hash1 = ((sh1 >> 16) & 0xffff) | (sh2 & 0xffff) << 16;
      hash2 = ((sh2 >> 16) & 0xffff) | (sh1 & 0xffff) << 16;
    }
  return hash1 ^ hash2;
}
```

10. MaHash11

Это MaHash2 с другими константами сдвигов. Причем, теперь для каждой подфункции теперь применяется свое значение циклического сдвига. Показала себя очень хорошо, без резких счачков коллизий в сложных наборах. Третье место в нашем рейтинге.

```
#define LROT12(x) (((x) << 12) | ((x) >> 20 ))
#define RROT13(x) (((x) << 19) | ((x) >> 13 ))

unsigned int MaHash11 (unsigned char *str, unsigned int len)
{
   unsigned int t, hash1 = len, hash2 = len, i;
   for (i = 0; i != len; i++, str++)
   {
      hash1 += sTable[(*str + i) & 255];
      hash1 = LROT12(hash1 + ((hash1 << 6) ^ (hash1 >> 14)));
      hash2 += sTable[(*str + i) & 255];
      hash2 = RROT13(hash2 + ((hash2 << 6) ^ (hash2 >> 14)));
      t = hash1;
      hash1 = hash2;
      hash2 = t;
   }
   return hash1 ^ hash2;
}
```

11. MaHash10

Здесь уже нет двух параллельных подфункций, хотя и вычисляются два 32-разядных значения на каждом шаге. Как и впрошлой функции, циклические сдвиги различны для каждого вычисления. Как оказалась, такая конструкция имеет небольшие всплески коллизий и от классической пока отсказываться спешить не следует. Из всего семейства показала худший результат, хотя и опередила некоторых именитых соперников.

```
#define LROT14(x) (((x) << 14) | ((x) >> 18 ))
#define RROT14(x) (((x) << 18) | ((x) >> 14 ))

unsigned int MaHash10 (unsigned char *str, unsigned int len)
{
   unsigned int hash1 = len, hash2 = len, i, value;
   for (i = 0; i != len; i++, str++)
   {
      value = sTable[(*str + i) & 255];
      hash1 =  LROT14(hash2 + ((hash2 << 6) ^ (hash2 >> 11)));
      hash1 += value;
      hash2 =  RROT15(hash1 + ((hash1 << 6) ^ (hash1 >> 11)));
      hash2 += value;
   }
   return hash1 ^ hash2;
}
```

12. MaHash3

Как показали тесты, функция - одна из самых удачных в семействе. Дополнительная интерация немного снизила число коллизий на самых сложных тестах.

```
#define LROT(x) (((x) << 11) | ((x) >> 21 ))
#define RROT(x) (((x) << 21) | ((x) >> 11 ))

unsigned int
MaHash3 (unsigned char *str, unsigned int len)
{
  unsigned int t, hash1 = len, hash2 = len, i;
  unsigned char index;

  for (i = 0; i != len; i++, str++)
    {
      index = (*str + i) & 255;
      hash1 += sTable[index];
      hash1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      hash2 += sTable[(sTable[index] + 1) & 255];
      hash2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
      t = hash1;
      hash1 = hash2;
      hash2 = t;
    }
  hash1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
  hash1 += sTable[len];
  hash2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
  hash2 += sTable[len];

  return hash1 ^ hash2;
}
```

13. MaBKDR

У BKDR есть небольшой недостаток, он хорош на строках одной длины - на больших объемах текстов различной длины число коллизий увеличивается. Это легко исправить, просто добавляем счетчик символа на каждом шаге. Такая модификация, как показали исследования, значительно улучшит статистические показатели, снизив число столкновений. В то же время никаких деградаций функции по результатам тестов мной замечено не было. Значит, модификация имеет право на жизнь и такой вариант предпочтительнее оригинальной функции. И, что важно, добавлена лишь одна операция, которая выполняется быстро на современных процессорах.

```
unsigned int maBKDRHash (unsigned char *str, unsigned int len)
{
  unsigned int seed = 131313;   /* 31 131 1313 13131 131313 etc.. */
  unsigned int hash = 0;
  unsigned int i = 0;

  for (i = 0; i < len; str++, i++)
    {
      hash = (hash * seed) + *str + /* maBKDR modification here*/ i;
    }
  return hash;
}
```

14. MaPrime

Существующие простые хэш-функции на простых числах немного деградируют в сложных случаях, судя по моим тестам. Такие случаи - это наборы текстов, имеющие совпадающие участки. Более точно - это потому, что алгоритмы не предусмотрели четкой взаимосвязи между символом и его позицией в строке. Наш алгоритм очень прост, основан на операции умножения на простое число. Используется таблица подстановки, так же что и в других моих функциях. Как вариант, может использоваться упрощенный вариант без таблицы. Преимущества - элементарная и понятная структура, простота реализации и хорошие результаты во всех тестах.

```
#define PRIME_MULT 0x1FAF
#define START_PRIME 0x3A8F05C5
#define USE_SBOX

unsigned int maPrimeHash (unsigned char *buf, unsigned int len)
{
   unsigned int hval = START_PRIME, i;


   for (i = 0; i != len; i++, buf++)
   {
#ifdef USE_SBOX
      hval ^= sTable[(*buf +  i) & 255];
#else
      hval += ((unsigned int)*buf)  ^  i;
#endif
      hval *= PRIME_MULT;
   }
   return hval;
}
```

15. MaPrime2d

Как показали автоматизированные тесты, функция показывает более стабильные результаты, нежели оригинальная. Да и дополнения не так сильно отражаются на производительности. Второй момент - это возможность параметризации, т.е. тонкой натройки под определенный набор данных. Существенный параметр здесь - это константа циклического сдвига. В тесте использован сдвиг на два бита вправо, что обеспечило усредненные результаты по тестам вцелом, однако использование некоторых других значений может быть более приемлимым в зависимости от набора текстов. К примеру - один или три бита.

```
unsigned int maPrime2dHash (unsigned char *str, unsigned int len)
{
  unsigned int hash = 0, i;
  unsigned int rotate = 2;
  unsigned int seed = 0x1A4E41U;

  for (i = 0; i != len; i++, str++)
    {
      hash += sTable[(*str + i) & 255];
      hash = (hash << (32 - rotate) ) | (hash >> rotate);
      hash = (hash + i ) * seed;
    }
  return (hash + len) * seed;
}
```

16. MaHash8v64

Это - двойник алгоритма MaHash8, только теперь функция возвращает 64-разрядный хэш, тоесть оба 32-разрядных значения, не сжимая из в одно. Сам алгоритм полностью 32-разрядный, скорость его не уменьшилась. Такую манимуляцию можно провести со всеми алгоритмами семейства, в которых структура - это две подфункции. Достоинства алгоритма - высокая скорость, равная алгоритму с 32-разрядным хэшем, простота реализации и очень хорошие показатели по числу коллизий. Функция, безусловно, лидер в нашем забеге, хотя участвует вне конкурса - ведь хэш в два раза больше, чем у других участников. Для увеличения лавинного эффекта можно использовать простую модификацию с дополнительным преобразованием финального 64-битного значения.

```
#define LROT14(x) (((x) << 14) | ((x) >> 18 ))
#define RROT14(x) (((x) << 18) | ((x) >> 14 ))
#ifdef HIAVAL
#define LROT64(x) (((x) << 29) | ((x) >> 34 ))
#endif

unsigned long long MaHash8v64 (unsigned char *str, unsigned int len)
{
  unsigned int sh1, sh2, hash1 = len, hash2 = len, i;
  unsigned long long digest;

  for (i = 0; i != len; i++, str++)
    {
      hash1 += sTable[( *str + i) & 255];
      hash1 = LROT14 (hash1 + ((hash1 << 6) ^ (hash1 >> 11)));
      hash2 += sTable[( *str + i) & 255];
      hash2 = RROT14 (hash2 + ((hash2 << 6) ^ (hash2 >> 11)));
      sh1 = hash1;
      sh2 = hash2;
      hash1 = ((sh1 >> 16) & 0xffff) | ((sh2 & 0xffff) << 16);
      hash2 = ((sh2 >> 16) & 0xffff) | ((sh1 & 0xffff) << 16);
    }

#ifdef HIAVAL
  digest = (((unsigned long long) hash1) << 32)  | ((unsigned long long) hash2 );
  digest ^= LROT64(digest + ((digest << 13) ^ (digest >> 23)));
#else
  digest = (((unsigned long long) hash2) << 32)  | ((unsigned long long) hash1 );
#endif
```

```
  return digest;
}
```

17. MaHash4v64

Это модификация существующего алгоритма, аналогично вышерассмотренной 64-разрядной функции. Также, как и в прошлом варинте, можно усилить лавинный эффект путем добавления дополнительного преобразования. Это немного замедлит алгоритм, поскольку операции выполняются над 64-разрядными беззнаковыми числами, но функция будет выглядеть интереснее.

```
#define LROT(x) (((x) << 11) | ((x) >> 21 ))
#define RROT(x) (((x) << 21) | ((x) >> 11 ))
#ifdef HIAVAL
#define LROT64(x) (((x) << 29) | ((x) >> 34 ))
#endif

unsigned long long MaHash4v64 (unsigned char *str, unsigned int len)
{
  unsigned int sh1, sh2, hash1 = len, hash2 = len, i;
  unsigned long long digest;

  for (i = 0; i != len; i++, str++)
    {

      hash1 += sTable[( *str + i) & 255];
      hash1 = LROT (hash1 + ((hash1 << 6) ^ (hash1 >> 8)));
      hash2 += sTable[( *str + i) & 255];
      hash2 = RROT (hash2 + ((hash2 << 6) ^ (hash2 >> 8)));
      sh1 = hash1;
      sh2 = hash2;
      hash1 = ((sh1 >> 16) & 0xffff) | ((sh2 & 0xffff) << 16);
      hash2 = ((sh2 >> 16) & 0xffff) | ((sh1 & 0xffff) << 16);
    }

#ifdef HIAVAL
  digest = (((unsigned long long) hash1) << 32)  | ((unsigned long long) hash2 );
  digest ^= LROT64(digest + ((digest << 13) ^ (digest >> 23)));
#else
  digest = (((unsigned long long) hash2) << 32)  | ((unsigned long long) hash1 );
#endif
  return digest;

}
```

18. MaHashMR1

Белая ворона? Более чем. Здесь мы имеет саму необычную структуру: четыре “вращающихся” 8-битных регистра накопления, которые модифицируются с помощью двойной табличной замены. На каждом шаге для накопления входного текста выбирается следующий регистр, затем все биты всех регистров циклически сдвигаются на один бит. Функция может иметь любое кол-во итераций. По всей видимости, оптимальным их числом будет три.

```
#define LROT1(x) (((x) << 1) | ((x) >> 31 ))
#define MR_ITERATIONS 3

union type_regs
  {
    unsigned long d32;
    unsigned char d8[4];
  };
unsigned long MaHashMR1 (unsigned char *str, unsigned int len)
{
  unsigned int  i, s = 0;
  unsigned char val;
  union type_regs regs;

  regs.d32 = len;
  for (i = 0; i != len * MR_ITERATIONS; i++)
    {
      val = *(str + (i % len));
      regs.d8[s] =  sTable[(sTable[(val + i) & 255] + regs.d8[s]) & 255];
      s = (s + 1) % 4;
      regs.d32 = LROT1(regs.d32);
    }
  return regs.d32;
}
```

19. MurmurHash2AM

Как можно улучшить функцию, котороя итак уже близка к оптимальной? Легко. Надо добавить счетчик. Счетчик блока. Ведь счетчик вообще - дело хорошее. Он помогает нам для двух одинаковых блоков вычислить разные значения. Это не совсем то, что даст хорошая табличная подстановка, но уже избавит от множества проблем. Мы не получим таких всплесков на сложных наборах, как исходный вариант. При этом общие результаты пострадать не должны. Итак, перед вами моя модификация MurmurHash2AM. M- потому, что модификация. AM, как ни странно - тоже имеет смысл - имя автора модификации.

```
#define mmix(h,k) { k *= m; k ^= k >> r; k *= m; h *= m; h ^= k; }
unsigned int MurmurHash2AM ( char * key, unsigned int len)
{
  const unsigned int m = 0x5bd1e995;
  const int r = 24;
  unsigned int l = len, h = 0, i = 1;
  const unsigned char * data = (const unsigned char *)key;

  while (len >= 4)
    {
      unsigned int k = *(unsigned int*)data + i++;
      mmix(h,k);
      data += 4;
      len -= 4;
    }
  unsigned int t = 0;
  switch (len)
    {
    case 3:      t ^= data[2] << 16;
    case 2:      t ^= data[1] << 8;
    case 1:      t ^= data[0];
    };

  mmix(h,t);
  mmix(h,l);
  h ^= h >> 13;
  h *= m;
  h ^= h >> 15;
  return h;
}
```

20. GoulburnHash

```
const unsigned int g_table0[256] =
{
  4143812366UL,2806512183UL,4212398656UL,393834663UL,
  3943187971UL,847901099UL,3746904015UL,2990585247UL,
  4243977488UL,4075301976UL,2737181671UL,2429701352UL,
  4196558752UL,3152011060UL,1432515895UL,204108242UL,
  1180540305UL,922583281UL,1734842702UL,1453807349UL,
  507756934UL,1553886700UL,2005976083UL,3346025117UL,
  97642817UL,2510760451UL,4103916440UL,3222467334UL,
  1312447049UL,522841194UL,3955607179UL,3028936967UL,
  2763655970UL,3033075496UL,1935362065UL,512912210UL,
  2660383701UL,1652921526UL,260485165UL,141882627UL,
  2895806269UL,804034013UL,1356707616UL,3942447612UL,
  2875374199UL,81028672UL,1055595160UL,2755907176UL,
  2880512448UL,1232977841UL,3719796487UL,2940441976UL,
  3739585976UL,168332576UL,1318372270UL,3173546601UL,
  3992298512UL,3785690335UL,3667530757UL,3101895251UL,
  2789438017UL,3213463724UL,3067100319UL,2554433152UL,
  794184286UL,2599814956UL,1251486151UL,4214997752UL,
  690900134UL,323888098UL,1537487787UL,1155362310UL,
  1826165850UL,2358083425UL,2957662097UL,2514517438UL,
  1828367703UL,3847031274UL,2308450901UL,955547506UL,
  1037823031UL,2922505570UL,2544914051UL,2572931499UL,
  442837508UL,1873354958UL,2004376537UL,25413657UL,
  3560636876UL,1768043132UL,2870782748UL,1031556958UL,
  715180405UL,201079975UL,4116730284UL,2748714587UL,
  1091411202UL,33354499UL,1931487277UL,1039106939UL,
  3327011403UL,396608379UL,3447523131UL,301432924UL,
  3180185526UL,1780290520UL,3909968679UL,2398211959UL,
  3704875308UL,66082280UL,601805180UL,3226323057UL,
  3284786200UL,2282257088UL,700775591UL,3528928994UL,
  1601645543UL,120115228UL,568698020UL,178214456UL,
  41846783UL,897656032UL,3309570546UL,2624714322UL,
  2542948622UL,1168171675UL,2460933760UL,93808223UL,
  2384991231UL,4268721795UL,4001720080UL,1516739672UL,
  4111847489UL,810915309UL,1238071781UL,935043360UL,
  2020231594UL,37717498UL,3603218947UL,1534593867UL,
  2819275526UL,1965883441UL,674162751UL,128087286UL,
```

```
  4138356188UL,543626850UL,1355906380UL,3565721429UL,
  1142978716UL,1614752605UL,1624389156UL,3363454971UL,
  2029311310UL,2249603714UL,3448236784UL,1764058505UL,
  2198836711UL,3481576182UL,3168665556UL,3834682664UL,
  1979945243UL,3456525349UL,2721891322UL,1099639387UL,
  1528675965UL,3069012165UL,1807951214UL,1901014398UL,
  2805656341UL,3321210152UL,2317543573UL,1015607418UL,
  178584554UL,4020226276UL,492648819UL,97778844UL,
  4134244261UL,1389599433UL,331211243UL,3769684011UL,
  2036127367UL,3174548433UL,3241354897UL,2570869934UL,
  3071842004UL,1972073698UL,48467379UL,1015444026UL,
  3126762609UL,1104264591UL,3096375666UL,1380392409UL,
  684368280UL,1493310388UL,2109527660UL,3034364089UL,
  3168522906UL,3042350939UL,3696929834UL,3410250713UL,
  3726870750UL,3357455860UL,1816295563UL,2678332086UL,
  26178399UL,614899533UL,2248041911UL,1431155883UL,
  1184971826UL,3711847923UL,2744489682UL,168580352UL,
  694400736UL,2659092308UL,811197288UL,1093111228UL,
  824677015UL,2041709752UL,1650020171UL,2344240270UL,
  3773698958UL,3393428365UL,3498636527UL,556541408UL,
  1883820721UL,3249806350UL,3635420446UL,1661145756UL,
  3087642385UL,1620143845UL,3852949019UL,1054565053UL,
  3574021829UL,2466085457UL,2078148836UL,460565767UL,
  4097474724UL,1381665351UL,1652238922UL,2200252397UL,
  3726797486UL,4001080204UL,259576503UL,567653141UL,
  325219513UL,1227314237UL,3191441965UL,1433728871UL,
  4198425173UL,2908977223UL,3757065246UL,294312130UL,
  4136006097UL,3409363054UL,2112383431UL,1177366649UL
};

const unsigned int g_table1[128] =
{
  826524031UL,360568984UL,3001046685UL,1511935255UL,
  1287825396UL,3167385669UL,1488463483UL,4077470910UL,
  1360843071UL,986771770UL,2307292828UL,3845679814UL,
  1429883439UL,1990257475UL,4087625806UL,1700033651UL,
  1388994450UL,935547107UL,3237786789UL,644530675UL,
  2274037095UL,888755779UL,3020158166UL,2136355264UL,
  2558959443UL,1751931693UL,2325730565UL,3029134627UL,
  668542860UL,2140243729UL,2384660990UL,666440934UL,
```

```
 842610975UL,1563602260UL,1429103271UL,899918690UL,
 3441536151UL,4078621296UL,1527765522UL,4191433361UL,
 222526771UL,309447417UL,2035245353UL,3730203536UL,
 3330019758UL,876252573UL,2545027471UL,453932528UL,
 282738293UL,1826993794UL,1569532013UL,543681326UL,
 3097574376UL,2336551794UL,1563241416UL,1127019882UL,
 3088670038UL,2766122176UL,3706267663UL,1110947226UL,
 2608363541UL,3166834418UL,1310161541UL,755904436UL,
 2922000163UL,3815555181UL,1578365408UL,3137960721UL,
 3254556244UL,4287631844UL,750375141UL,1481489491UL,
 1903967768UL,3684774106UL,765971482UL,3225162750UL,
 2946561128UL,1920278401UL,1803486497UL,4166913456UL,
 1855615192UL,1934651772UL,1736560291UL,2101779280UL,
 3560837687UL,3004438879UL,804667617UL,2969326308UL,
 3118017313UL,3090405800UL,566615197UL,2451279063UL,
 4029572038UL,2612593078UL,3831703462UL,914594646UL,
 2873305199UL,2860901605UL,3296630085UL,1273702937UL,
 2852911938UL,1003268745UL,1387783190UL,159227777UL,
 2211994285UL,28095103UL,3659848176UL,3976935977UL,
 3301276082UL,2641346573UL,651238838UL,2264520966UL,
 1484747269UL,3016251036UL,3857206301UL,91952846UL,
 1662449304UL,2028491746UL,1613452911UL,2409055848UL,
 1453868667UL,4146146473UL,1646176015UL,3769580099UL,
 3171524988UL,2980516679UL,828895558UL,3384493282UL
};
typedef unsigned long long DIGEST_TYPE;

DIGEST_TYPE GoulburnHash( char *cp, unsigned int len)
{
  register unsigned int h = 0, u;

  for ( u=0; u<len; ++u )
    {
      h += g_table0[ (unsigned char) cp[u] ];
      h ^= (h << 3) ^ (h >> 29);
      h += g_table1[ h >> 25 ];
      h ^= (h << 14) ^ (h >> 18);
      h += 1783936964UL;
    }
  return h;}
```

21. OneAtTimeHash

```
typedef unsigned long long DIGEST_TYPE;

DIGEST_TYPE OneAtTimeHash (char *key, unsigned int key_len)
{
  unsigned int hash = 0;
  size_t i;

  for (i = 0; i < key_len; i++)
    {
      hash += (unsigned char) key[i];
      hash += (hash << 10);
      hash ^= (hash >> 6);
    }
  hash += (hash << 3);
  hash ^= (hash >> 11);
  hash += (hash << 15);
  return hash;
}
```

Порядок сдачи лабораторной работы:

В отчёте, созданном в текстовом процессоре Libre Office Writer, должно быть программное средство.
Титульный лист отчета приведен в приложении 1.
По требованию преподавателя повторить работу в лаборатории и дать ответы на вопросы по работе

Срок сдачи лабораторной — до

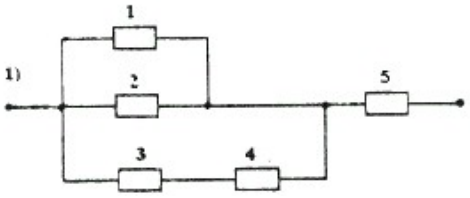
4. Комплект тестов

Тест 1

1. Как определяется вероятность безотказной работы информационной системы, если известны вероятности безотказной работы аппаратного, программного, эргономического, информационного обеспечения?

2. В программу было внесено s=30 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=15 собственных ошибок программы и v= 5 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

3. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов.



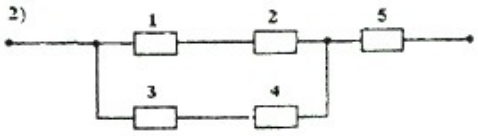
4. В течение некоторого периода времени производилось наблюдение за работой радиолокационной станции. За весь период наблюдения было зарегистрировано 15 отказов. До начала наблюдения станция проработала 258 час, к концу наблюдения наработка станции составила 1233 час. Требуется определить среднюю наработку на отказ $t_{ср}$.

Тест 2

1. Назовите основные количественные показатели надежности аппаратного обеспечения информационной системы?

2. В программу было внесено s=20 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=10 собственных ошибок программы и v= 2 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

3. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов.



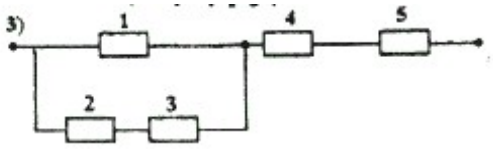
4. Производилось наблюдение за работой трех однотипных изделий. За период наблюдения было зафиксировано по первому экземпляру аппаратуры 6 отказов, по второму и третьему – 11 и 8 отказов соответственно. Наработка первого экземпляра составила 181 час, второго – 329 и третьего – 245 час. Требуется определить наработку аппаратуры на отказ.

Тест 3

1. Назовите основные количественные показатели надежности программного обеспечения информационной системы?

2. В течение суток осуществлялся прогон программы. Из 300 тестов были признаны неправильными результаты 15 тестов. Какова вероятность отказа программы?

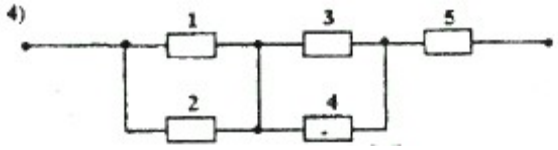
3. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов.



4. На испытание поставлено N =1000 невосстанавливаемых элементов. Число отказов фиксировалось в каждом интервале времени испытаний Δt = 500 час. На первом интервале отказало 145, на втором - 86, на третьем - 77 элементов. Требуется определить вероятность безотказной работы, частоту и интенсивность отказов, среднюю наработку до первого отказа элементов.

Тест 4

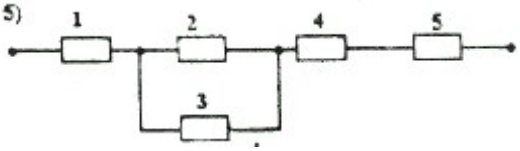
1. Что такое стохастическое тестирование программных средств?
2. Утверждается, что в программе нет ошибок. Вносится 4 ошибки. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.
3. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов.



4. На испытание поставлено 6 однотипных изделий. Получены следующие значения t_i (t_i - время безотказной работы i- го изделия) : t_1 =280 час; t_2 = 350 час; t_3 =400 час; t_4=320 час; t_5 =380 час; t_6 =330 час. Определить статистическую оценку среднего времени безотказной работы изделия.

Тест 5

1. Назовите основные количественные показатели надежности информационного обеспечения информационной системы
2. Утверждается, что в программе нет ошибок. Вносится 19 ошибки. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.
3. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов.



4. В результате наблюдения за 5 образцами радиоэлектронного оборудования получены данные до первого отказа всех 5 образцов: t_1 =470 час; t_2 = 590 час; t_3

=425 час; t_4=620 час; t_5 =880 час. Необходимо найти среднюю наработку до первого отказа.

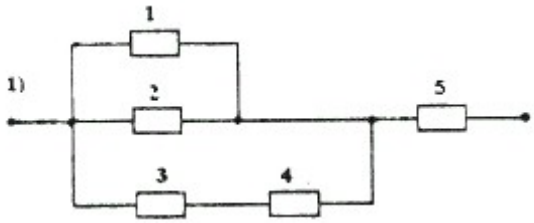
Тест 6

1. Назовите наиболее часто используемые модели надежности аппаратных средств

2. Допустим , что на испытание поставлено 1000 однотипных электронных ламп типа 6Ж4. За первые 3000 час отказало 80 ламп. За интервал времени 3000—4000 час отказало еще 50 ламп. Требуется определить частоту и интенсивность отказов ламп в промежутке времени 3000—4000 час.

3. В программу было внесено s=30 ошибок. Тестирование продолжалось, пока не были найдены все внесенные ошибки. Оказалось, что дополнительно были найдены n=20 собственных ошибок программы. Какова вероятность гипотезы, предполагающей, что программа имела первоначально не более 40 ошибок.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и пятый элемент двукратно резервирован.



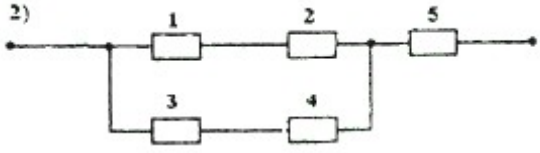
Тест 7

1.Назовите наиболее часто используемые модели надежности программных средств

2. На испытание поставлено 500 однотипных мониторов. За 3000 час отказало 8 мониторов. Требуется определить вероятность безотказной работы и вероятность отказа мониторов в течение 3000 час.

3. Утверждается, что в программе нет ошибок. Вносится 32 ошибки. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и первый элемент двукратно зарезервирован.



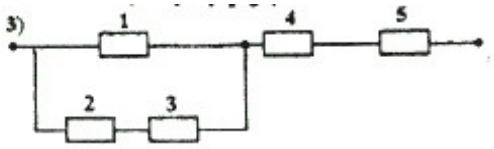
Тест 8

1. Какие тесты нужно создать для проверки работы оператора цикла?

2. На испытание поставлено 100 однотипных изделий. За 4000 час отказало 50 изделий. За интервал времени 4000—4100 час отказало еще 20 изделий. Требуется определить частоту и интенсивность отказов изделий в промежутке времени 4000—4100 час .

3. В течение суток осуществлялся прогон программы. Из 400 тестов были признаны неправильными результаты 22 тестов. Какова вероятность отказа программы?

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и первый элемент двукратно зарезервирован.



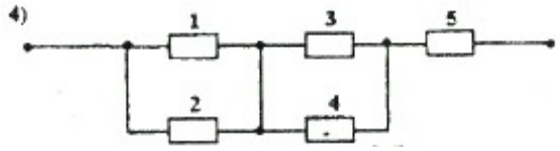
Тест 9

1. Какие тесты нужно создать для проверки работы условного оператора?

2. В течение 1 000 час из 10 гироскопов отказало 2. За интервал времени 1000—1100 час отказал еще один гироскоп. Требуется найти частоту и интенсивность отказов гироскопов в промежутке времени 1000—1100 час.

3. В программу было внесено s=45 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=10 собственных ошибок программы и v=5 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и второй элемент двукратно зарезервирован.



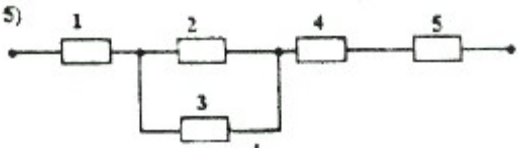
Тест 10

1. Какие тесты нужно создать для проверки принадлежности параметра X отрезку [a,b] ?

2. На испытание поставлено 400 резисторов. За время наработки 10000 час отказало 4 резистора. За последующие 1000 час отказал еще 1 резистор. Определить частоту и интенсивность отказов резисторов в промежутке времени 10000 –11000 час .

3. В программу было внесено s=60 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=20 собственных ошибок программы и v=15 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и четвертый элемент двукратно зарезервирован.



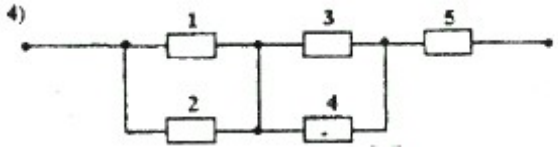
Тест 11

1. При каком способе резервирования аппаратных средств надежность аппаратных средств будет ниже?

2. В течение 3000 час из 10 жестких дисков ЭВМ отказал 1. За интервал времени 3000—3100 час отказал еще один диск. Требуется найти частоту и интенсивность отказов жестких дисков ЭВМ в промежутке времени 3000—3100 час.

3. В программу было внесено s=50 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=20 собственных ошибок программы и v= 10 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и второй элемент двукратно зарезервирован.



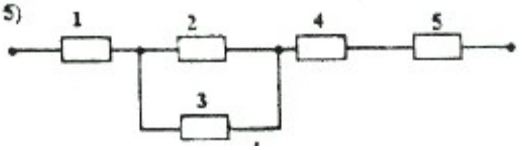
Тест 12

1. Назовите три способа резервирования аппаратных средств информационной системы

2. На испытание поставлено 400 резисторов. За время наработки 10000 час отказало 4 резистора. За последующие 1000 час отказал еще 1 резистор. Определить частоту и интенсивность отказов резисторов в промежутке времени 10000 –11000 час .

3. Утверждается, что в программе нет ошибок. Вносится 15 ошибок. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и первый элемент двукратно зарезервирован.



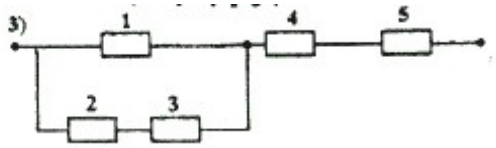
Тест 13

1. Назовите три периода работы аппаратных средств информационной системы, которые учитываются в теории надежности

2. В течение некоторого периода времени производилось наблюдение за работой радиолокационной станции. За весь период наблюдения было зарегистрировано 15 отказов. До начала наблюдения станция проработала 258 час, к концу наблюдения наработка станции составила 1233 час. Требуется определить среднюю наработку на отказ $t_{ср}$.

3. Утверждается, что в программе нет ошибок. Вносится 10 ошибок. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и первый элемент трехкратно зарезервирован.



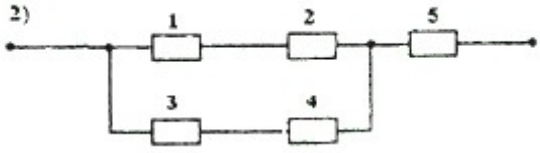
Тест 14

1. Какие тесты нужно создать для проверки принадлежности параметра X множеству значений {X1,..,Xn}?

2. Производилось наблюдение за работой трех однотипных изделий. За период наблюдения было зафиксировано по первому экземпляру аппаратуры 6 отказов, по второму и третьему – 11 и 8 отказов соответственно. Наработка первого экземпляра составила 181 час, второго – 329 и третьего – 245 час. Требуется определить наработку аппаратуры на отказ.

3. В программу было внесено s=30 ошибок. Тестирование продолжалось, пока не были найдены все внесенные ошибки. Оказалось, что дополнительно были найдены n=40 собственных ошибок программы. Какова вероятность гипотезы, предполагающей, что программа имеет не более 60 ошибок.

4. Рассчитать вероятность безотказной работы системы со следующей структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и пятый элемент трехкратно зарезервирован.



Тест 15

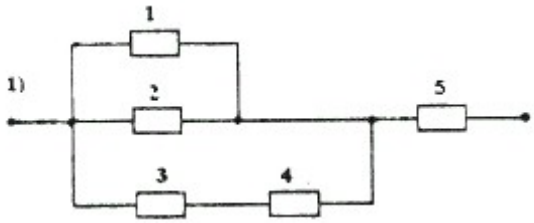
1. Что такое структурное тестирование программных средств?

2. На испытание поставлено N =1000 невосстанавливаемых элементов. Число отказов фиксировалось в каждом интервале времени испытаний Δt = 500 час. На первом интервале отказало 145, на втором - 86, на третьем - 77 элементов. Требуется определить вероятность безотказной работы, частоту и интенсивность отказов, среднюю наработку до первого отказа элементов.

3. В результате прогона программы на 100 тестах, в 25 признаны ошибочными результаты. Какова вероятность безотказной работы программы?

4. Рассчитать вероятность безотказной работы системы со следующей

структурной схемой соединения элементов, если известны вероятности безотказной работы ее элементов и четвертый элемент трехкратно зарезервирован.



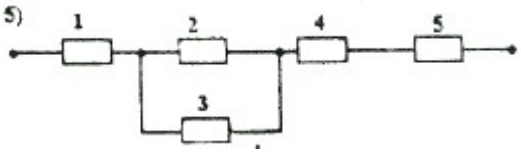
Тест 16

1. Назовите несколько моделей надежности программных средств

2. На испытание поставлено 6 однотипных изделий. Получены следующие значения t_i (t_i - время безотказной работы i- го изделия) : t_1 =280 час; t_2 = 350 час; t_3 =400 час; t_4=320 час; t_5 =380 час; t_6 =330 час. Определить статистическую оценку среднего времени безотказной работы изделия.

3. Утверждается, что в программе нет ошибок. Вносится 40 ошибок. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе были ошибки.

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов.



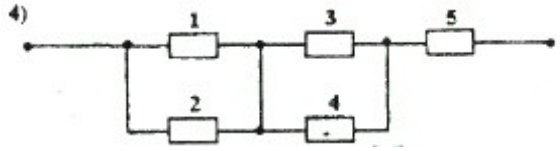
Тест 17

1. Как повысить надежность хранения информации?

2. В результате наблюдения за 5 образцами радиоэлектронного оборудования получены данные до первого отказа всех 5 образцов: t_1 =470 час; t_2 = 590 час; t_3 =425 час; t_4=620 час; t_5 =880 час. Необходимо найти среднюю наработку до первого отказа.

3. В результате прогона программы на 200 тестах, в 32 признаны ошибочными результаты. Какова вероятность безотказной работы программы?

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов.



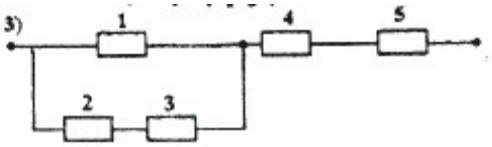
Тест 18

1. Как вычислить вероятность безотказной работы оператора?

2. Утверждается, что в программе нет ошибок. Вносится 10 ошибок. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе не было ошибок.

3. В течение некоторого периода времени производилось наблюдение за работой радиолокационной станции. За весь период наблюдения было зарегистрировано 23 отказа. До начала наблюдения станция проработала 348 час, к концу наблюдения наработка станции составила 1512 час. Требуется определить среднюю наработку на отказ $t_{ср}$.

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов.



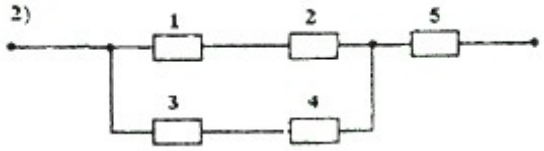
Тест 19

1. Как с точки зрения надежности следует подключать в локальной сети двухкратно зарезервированный сервер и пять рабочих станций?

2. В программу было внесено s=30 ошибок. Тестирование продолжалось, пока не были найдены все внесенные ошибки. Оказалось, что дополнительно были найдены n=40 собственных ошибок программы. Какова вероятность гипотезы, предполагающей, что программа имела первоначально не более 70 ошибок.

3. Производилось наблюдение за работой трех однотипных изделий. За период наблюдения было зафиксировано по первому экземпляру аппаратуры 8 отказов, по второму и третьему – 18 и 7 отказов соответственно. Наработка первого экземпляра составила 234 час, второго – 427 и третьего – 356 час. Требуется определить среднюю наработку аппаратуры на отказ.

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов.



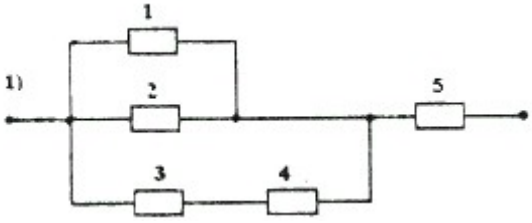
Тест 20

1. Какова должна быть кратность резервирования программных средств, чтобы решение о правильности результата принималось по правилу 4 из 6?

2. В программу было внесено s=60 ошибок, после чего разрешено начать тестирование. Причем, при тестировании было обнаружено n=15 собственных ошибок программы и v=46 внесенных ошибок. Оценить первоначальное количество ошибок программы, используя статистическую модель Миллса.

3. На испытание поставлено N =5000 невосстанавливаемых элементов. Число отказов фиксировалось в каждом интервале времени испытаний Δt = 500 час. На первом интервале отказало 230, на втором - 68, на третьем - 82 элементов. Требуется определить вероятность безотказной работы, частоту и интенсивность отказов, среднюю наработку до первого отказа элементов.

4. Рассчитать вероятность отказа работы системы со следующей структурнойсхемой соединения элементов, если известны вероятности отказа работы ее элементов.



Тест 21

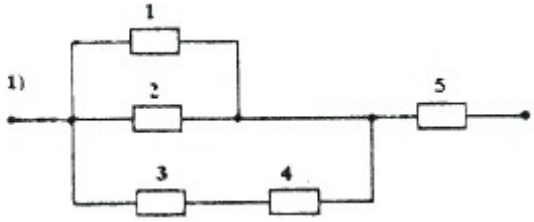
1. Как осуществляется резервирование программных средств?

2. Утверждается, что в программе нет ошибок. Вносится 45 ошибок. При тестировании все внесенные ошибки были обнаружены, кроме собственных ошибок программы. Определить по модели Миллса вероятность того, что в программе были ошибки.

3. На испытание поставлено 4 однотипных изделий. Получены следующие значения t_i (t_i - время безотказной работы i- го изделия) : t_1 =820 час; t_2 = 530 час; t_3 =830 час; t_4=230 час. Определить статистическую оценку среднего времени

безотказной работы изделия.

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов и пятый элемент двукратно зарезервирован.



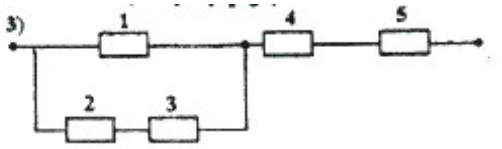
Тест 22

1. Что такое функциональное тестирование программных средств?

2. В течение суток осуществлялся прогон программы. Из 500 тестов были признаны неправильными результаты 45 тестов. Какова вероятность отказа программы?

3. В результате наблюдения за 3 образцами радиоэлектронного оборудования получены данные до первого отказа всех 3 образцов: t_1 =740 час; t_2 = 950 час; t_3 =524 час. Необходимо найти среднюю наработку до первого отказа.

4. Рассчитать вероятность отказа работы системы со следующей структурной схемой соединения элементов, если известны вероятности отказа работы ее элементов и четвертый элемент двукратно зарезервирован.



5. Рекомендации по подготовке к экзамену
5.1. Вопросы к экзамену

Раздел 1. Расчет надежности технических средств ИС

Тема 1. Основные понятия теории надежности (*Надежность системы, аппаратная, программная, функциональная и эксплуатационная надежность. Безотказность, ремонтопригодность, долговечность и сохраняемость. Независимый, внезапный, постепенный и перемежающийся отказ (сбой). Наработка до отказа, наработка между отказами. Факторы влияющие на надежность при проектировании, в процессе изготовления эксплуатации. Пути повышения надежности*)

Тема 2. Необходимые сведения из теории вероятности и математической статистики (*Схема независимых испытаний Бернулли, случайные величины, закон распределения дискретной случайной величины, плотности распределения непрерывной случайной величины, функция распределения непрерывной случайной величины. Критерии согласия проверки гипотез*)

Тема 3. Классификация и расчет показателей надежности ИС (*Определение надежности элемента системы, работающего до первого отказа. Вероятность отказа до определенного момента. Плотность распределения отказов. Вероятность безотказной работы до определенного момента. Функция интенсивности отказов. Примеры наиболее часто используемых законов распределения времени безотказной работы ИС*)

Тема 4. Надежность взаимосвязанных элементов системы (*Последовательное и параллельное соединение элементов системы. Мостиковые структуры. Преобразование треугольного соединения элементов в звезду и наоборот*)

Тема 5-6. Классификация методов резервирования. Расчет надежности системы с резервированием (*Постоянное резервирование: общее, поэлементное. Резервирование замещением: нагруженный, облегченный, ненагруженный резерв. Поэлементное резервирование замещением. Резервирование с дробной кратностью и постоянно включенным резервом*)

Тема 7. Надежность восстанавливаемых систем (*Восстанавливаемые и не восстанавливаемые системы. Вероятностные модели отказов. Надежность систем с восстановлением. Надежность систем без восстановления*)

Раздел 2. Надежность программных средств ИС

Тема 1. Качество программных средств ИС (*Определение требований к ПС. Понятие качества ПС. Спецификация качества ПС. Функциональная спецификация. Методы спецификации семантики функций*)

Тема 2. Факторы, определяющие надежность ПС. Показатели надежности ПС (*Общие принципы обеспечения надежности ПС. Обеспечение завершенности, точности, автономности, устойчивости ПС. Обеспечение защищенности ПС: защита от сбоев аппаратуры, от влияния "чужой" программы, защита от влияния "своей" программы, от ошибок оператора, защита от НСД, защита от защиты. Показатели надежности ПС. Математические модели надежности комплексов программ*)

Тема 3. Методы проектирования надежного ПС (*Понятие архитектуры ПС. Основные классы архитектур ПС. Архитектурные функции. Контроль архитектуры ПС. Основные характеристики программного модуля. Современные методы разработки структуры программы. Контроль структуры программы*)

Тема 4. Методы создания надежного ПС (*Порядок разработки программного модуля. Структурное программирование. Пошаговая детализация и понятие о псевдокоде. Оптимизация программ. Контроль программного модуля.*)

Тема 5. Методы тестирования и отладки надежного ПС (*Понятие обоснования программ. Тестирование ПС. Организация тестирования. Тестирование архитектуры, качества, документации, требований к ПС. Принципы и виды отладки ПС. Организация отладки ПС. Автономная отладка. Комплексная отладка*)

Раздел 3. Надежность информационных систем

Тема 1. Надежность оператора (*Основные понятия. Виды отказов оператора. Методы прогнозирования надежности оператора. Методы повышения надежности оператора*)

Тема 2. Надежность сетей передачи данных *(Процедуры повышенной надежности доставки данных. Логико-вероятностный расчет надежности передачи информации в*

типовых сетевых структурах. Методы расчета показателей надежности двухполюсной сети связи)

Тема 3-4. Методы обеспечения надежности ИС на этапах жизненного цикла (*Проектирование информационных систем и надежность. Предварительный анализ надежности. Детальная разработка технологической и эксплуатационной документации. Ресурсы необходимые для обеспечения надежности. Средства встроенного контроля процесса функционирования информационных систем. Особенности обеспечения надежности функционирования информационных систем на этапе разработки. Организация службы эксплуатации. Планирование профилактического обслуживания системы и предотвращения ее износа*)

Тема 5. Взаимосвязь надежности, безопасности и экономической эффективности информационных систем (*Основные понятия и факторы, определяющие безопасность информационных систем. Ресурсы для обеспечения функциональной безопасности ИС. Разработка требований к функциональной безопасности ИС. Основные понятия и определения экономической эффективности информационных систем. Взаимосвязь надежности и функциональной безопасности ИС. Влияние надежности ИС на экономическую эффективность ИС*)

5.2. Допуск к экзамену

К экзамену допускаются студенты, выполнившие и защитившие все задания.

Приложение 1

Министерство науки и высшего образования РФ ФГБОУ ВПО “Ульяновский государственный университет”	Форма	
Ф-Титульный лист лабораторной работы		

Министерства науки и высшего образования РФ
Ульяновский государственный университет

Факультет Математики, информационных и авиационных технологий
Кафедра Телекоммуникационные технологии и сети

ЛАБОРАТОРНАЯ РАБОТА
по дисциплине

(название дисциплины)

(название темы)

Направление бакалавриата Информационные системы и технологии. 09.03.02

Работу выполнил студент _____ _____ _____
(группа) (подпись, дата) (Ф.И.О.)

Научный руководитель _____ _____ _____
(должность) (подпись, дата) (Ф.И.О.)

_____
(оценка)

У Л Ь Я Н О В С К
20__ г.