Министерство образования и науки Российской Федерации
Федеральное государственное учреждение высшего образования
Московский государственный университет технологий и управления
(первый казачий университет)
Университетский колледж информационных технологий

Специальность 09.02.03 Программирование в компьютерных системах
ПМ1 Разработка программных модулей программного обеспечения компьютерных сетей
Сборник практических заданий
для проведения
УП1.1 учебной практики на базе прикладного и системного программирования

Москва
2017

Содержание

1 Введение **3**

2 Первая часть учебной практики **5**
2.1 Введение (для студентов) . . . 5
2.2 Стандарты кодирования . . . 5
2.3 Задание на практику . . . 6

3 Вторая часть учебной практики **12**
3.1 Введение (для студентов) . . . 12
3.2 Методика оценивания . . . 12
3.3 Практическая работа №1. Работа с файловой системой в Linux (12 часов) . . . 12
3.4 Практическая работа №2. Работа с файловой системой в Windows (4 часа) . . . 14
3.5 Практическая работа №3. Рекурсия (6 часов) . . . 14
3.6 Практическая работа №4. Работа с параметрами командной строки (2 часа) . . . 14
3.7 Практическая работа №5. Создание новых процессов/потоков (14 часов) . . . 16
3.8 Практическая работа №6. Создание новых нитей выполнения в WPF (6 часов) . . . 16
3.9 Практическая работа №7. Динамическое программирование (4 часа) . . . 17
3.10 Практическая работа №8. Использование каналов в Linux (10 часов) . . . 19
3.11 Практическая работа №9. Разделяемая память и семафоры в Linux (8 часов) . . . 19
3.12 Практическая работа №10. Разделяемая память, критические секции, каналы в Windows (6 часов) 19

4 Отчет **20**
4.1 Структура отчета . . . 20
4.2 Титульный лист . . . 20
4.3 Оформление отчета . . . 20
4.3.1 Требования машинописи . . . 21
4.4 Введение . . . 21
4.5 Основная часть . . . 21
4.5.1 Разработка спецификаций различных компонент (ПК1.1) . . . 21
4.5.2 Осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля (ПК1.2) . . . 21
4.5.3 Выполнение отладки программных модулей с использованием специализированных программных средств (ПК1.3) . . . 21
4.5.4 Выполнение тестирования программных модулей (ПК1.4) . . . 21
4.5.5 Осуществление оптимизации программного кода модуля (ПК1.5) . . . 22
4.5.6 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций (ПК1.6) . . . 22
4.6 Заключение . . . 22

Приложение А. Титульный лист **23**

1 Введение

«Учебная практика направлена на направлена на формирование у обучающихся умений, приобретение первоначального практического опыта и реализуется в рамках профессиональных модулей ОПОП СПО по основным видам профессиональной деятельности для последующего освоения ими общих и профессиональных компетенций по избранной специальности[1]».

Данная учебная практика входит в профессиональный модуль «ПМ1 Разработка программных модулей программного обеспечения компьютерных сетей» и направлена на выполнение следующих требований к студенту[2]:

- иметь практический опыт:
  - разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования;
  - разработки кода программного продукта на основе готовой спецификации на уровне модуля;
  - использования инструментальных средств на этапе отладки программного продукта;
  - проведения тестирования программного продукта по определенному сценарию;
- уметь:
  - осуществлять разработку кода программного модуля на современных языках программирования;
  - создавать программу по разработанному алгоритму как отдельный модуль;
  - выполнять отладку и тестирование программы на уровне модуля;
  - оформлять документацию на программные средства;
  - использовать инструментальные средства для автоматизации оформления документации;
- знать
  - основные этапы разработки программного обеспечения;
  - основные принципы технологии структурного и объектно-ориентированного программирования;
  - основные принципы отладки и тестирования программных продуктов;
  - методы и средства разработки технической документации.

Результатом освоения профессионального модуля должно быть формирование следующих компетенций:

- выполнять разработку спецификаций отдельных компонент;
- осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля;
- выполнять отладку программных модулей с использованием специализированных программных средств;
- выполнять тестирование программных модулей;
- осуществлять оптимизацию программного кода модуля;
- разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

Кроме того, изучение профессионального модуля должно способствовать формированию следующих общих компетенций:

- понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес;
- организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество;
- принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность;

[1] ст. 6 положения о практике обучающихся, осваивающих основные профессиональные образовательные программы среднего профессионального образования, утвержденного приказом Министерства образования и науки Российской Федерации (Минобрнауки России) от 18 апреля 2013 г. N 291

[2] Федеральный государственный стандарт среднего специального образования по специальности 09.02.03 Программирование в компьютерных системах, утвержденный приказом Министерства образования и науки Российской Федерации от 28 июля 2014 г. №804

- осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития;
- использование информационно-коммуникационных технологий в профессиональной деятельности;
- работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями;
- брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий;
- самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации;
- ориентироваться в условиях частой смены технологий в профессиональной деятельности;

Содержательно учебная практика состоит из двух частей:

1. первая часть (72 часа) базируется на курсе «Основы программирования», в ней производится отработка всех перечисленных выше компетенций, знаний, умений и навыков;
2. вторая часть (72 часа) базируется на МДК.01.01 «Системное программирование» и предназначена для дополнительного практического освоения тем этого курса: языка программирования C, а также программного интерфейса, предлагаемого современными операционными системами для решения задач системного программирования.

2 Первая часть учебной практики

2.1 Введение (для студентов)

В ходе первой части учебной практики Вам предстоит написать достаточно большую программу, попутно осуществляя отладку, тестирование, оптимизацию, документирование. В ходе работы, скорее всего, вам придется многократно просмотреть и переписать исходный код своего программного продукта: так работают все начинающие программисты. Тематика заданий выбрана специально с целью подготовки к предметам следующих курсов.

Оценка по результатам практики зависит от следующих параметров:

- уровня сложности (от 0 до 2);
- доли выполнения задания;
- качества исходного кода (требования указаны ниже);
- качества сопутствующей документации;
- архитектуры программного продукта (высоко оценивается объектно-ориентированная реализация), учитывается разделение на модули, предназначенные для ввода-вывода и на поддержание модели базы данных;
- посещения занятий;
- систематичности и регулярности работы над заданиями учебной практики;
- заполнения дневника, отчета.

На зачетное занятие необходимо предоставить отчет по результатам учебной практики, дневник учебной практики и в электронном виде материалы, подтверждающие выполнение заданий.

Обратите внимание на то, что использование каких-либо сторонних библиотек, а также библиотек, не изученных на занятиях, без согласования с преподавателем запрещено (так как это может сильно изменить уровень сложности задания, а также понизить уровень проработанности Вами исходного кода).

По умолчанию предполагается, что Вы пишете программы на языке Паскаль (Free Pascal) в среде Lazarus, работающую в консольном режиме.

По желанию (но это очень серьезная ответственность) можно программу написать на Ассемблере, C, C++, Rust, D, Go.

В идеале вы должны разработать отдельные модули для хранения информации (и проверки корректности данных) – модель, отдельные модули для ввода/вывода информации в файл(-ы), отдельные модули для организации интерфейса. Модель должна быть выполнено независимой от всего остального, остальные модули используют модель, но модель не использует остальные модули.

2.2 Стандарты кодирования

Ниже перечислены те требования к исходному коду, которые будут проверяться преподавателем в течение семестра.

1. (a) в начале каждого файла должен быть комментарий, содержащий сведения об авторе работы, номере задания и варианта, краткой формулировки задания (или его части);
   (b) к неочевидным действиям должны быть указаны комментарии (проверяется методом экспертной оценки лицом, осуществляющим проверку);
   (c) для каждой подпрограммы должен быть указан комментарий, содержащий полное описание ее работы, описание всех аргументов и результатов (этот комментарий должен быть достаточен для возможности использовать подпрограмму в других программах (без изучения собственно текста подпрограммы);
   (d) для каждой глобальной переменной должно быть указано ее назначения;
   (e) во всей программе должен соблюдаться единый стиль оформления; для языка Паскаль требования следующие:
      - для всех переменных должны быть использованы «говорящие» (англоязычные!) названия, при этом названия должны соответствовать стилю Camel: например, countOfTables);
      - для всех подпрограмм должны быть использованы говорящие (англоязычные!) названия (в случае наличия первое слово – глагол, например, showTable);

- на одной строке должен быть написан один оператор (кроме случая использования вложенных операторов без begin-end);
- количество пробелов перед строкой программы должно соответствовать уровню вложенности (по два пробела на уровень вложенности);
- слова begin и end, соответствующие друг другу, должны располагаться строго с одной и той же позиции по вертикали;
- количество строк в подпрограмме и в самой программе (между begin и end) должно быть не более 25 строк;
- три и более сходных по назначению подпрограммы должны выделяться в отдельный модуль;

(f) в подпрограммах должны отсутствовать случаи использования глобальных переменных;

(g) в подпрограммах, предназначенных для вычислений не должно быть ввода-вывода; в подпрограммах, предназначенных для ввода-вывода, не должно быть вычислений, кроме тех, что нужны для красивого/корректного ввода-вывода;

(h) в программе не должны использоваться операторы goto, break, continue; процедуры halt и exit (и их аналоги);

(i) вместо явно указанных значений чисел, в тексте программы должны использоваться константы.

2. Общие требования к надежности

(a) при работе с файлами должны проверяться все ошибки ввода/вывода;

(b) при вводе с клавиатуры должна проверяться корректность входных данных (в том числе пользователь может вместо числа ввести строку, число может превышать интервал представления типами данных языка);

(c) подпрограммы и модули при условии соблюдения всех требований, описанных в комментариях, должны правильно выполняться.

3. Общие требования к эргономике

(a) При запуске программы должно сообщаться ее назначение;

(b) При вводе должна выводиться подсказка;

(c) Должен быть реализован консольный интерфейс, напоминающий оконный (например, как в FAR): ввод/вывод должен осуществлять в окошки или таблицы, любые действия пользователя не должны портит интерфейс; создание консольного интерфейса должно проводиться с помощью базовых средств языка или самых простых библиотек (позволяющих менять цвет фона, символа, перемещать курсор, получать символ с клавиатуры);

4. Требования к оптимальности

(a) Программа и подпрограммы должны быть оптимальны по быстродействию и/или памяти (в случае, если это не требует очень больших усилий).

(b) В программе и модулях не должно быть набора более двух операторов, который повторяется (в том числе с шаблонными изменениями) в разных местах программы.

2.3 Задание на практику

Вам необходимо поэтапно разработать программу, позволяющую работать с базой данных, согласно вашему заданию. 0, 1 и 2 уровень соответствуют при прочих равных оценкам 3, 4 и 5.

Предполагается, что чтение и запись в файл производится в случае, когда пользователь выбрал соответствующий пункт меню (в остальное время содержимое базы данных находится в оперативной памяти); при этом имя файла пользователь вводит с клавиатуры.

Собственно разработка программы состоит из следующих этапов:

1. Создание прототипа базы данных (количество хранимых объектов от 0 до 100), реализующий следующие функции:

- **Добавление информации о новом объекте (в удобном интерфейсе[3], который нельзя испортить, каждое отдельное поле вводится либо в поле таблицы, либо в окошке)**
- **Удобный просмотр списка объектов (необходимо учесть, что их может быть больше 25)**

[3]интерфейс можно улучшать по сравнению с предложенным вариантом

- **Удаление информации об объекте (необходимо предусмотреть удобный выбор объекта для удаления – с помощью выбора в таблице)**
- **Удаление информации об объекте по значению первого поля (значение вводится таким образом, чтобы нельзя было испортить интерфейс, в окошке)**
- **Изменение информации об объекте (необходимо предусмотреть удобный выбор объекта для изменения – с помощью выбора в таблице) и удобный ввод (прямо в таблице или в окошках – по одному окошку на поле)**
- **Изменение информации об объекте по значению первого поля (значение вводится таким образом, чтобы нельзя было испортить интерфейс, в окошке)**
- **Нахождение записи по значению первого поля**
- **Сортировка по значению одного из полей первой таблицы (выбираемому пользователем).**

2. Внедрение функций для работы с файлами
   - чтение/запись в бестиповый файл (с достаточно оптимальным форматом хранения, должны быть применены все оптимизации способа хранения в случае, если это не требует применения методов компрессии).
3. **Внедрение другого способа хранения информации в программе: вместо массива используются списки.**
4. Рефакторинг программы в стиле ООП (как именно - согласуется индивидуально).

Общие требования к первой программе: создание экранного меню, управляемого с помощью клавиш-стрелок, при этом enter - активизация пункта меню, ESC - выход. Пункт меню должен подсвечиваться изменением цвета фона. В случае ввода информации в окошко должно контролироваться, что пользователь не ввел слишком много символов (информация должна оставаться в рамках окошка). Интерфейс должен быть оконным в том смысле, что информация входят в окошки, которые рисуются в текстовом режиме (с использованием псевдографики). По согласованию с преподавателем Вы можете реализовывать пользовательский интерфейс и другим способом в случае, если степень удобства не уменьшается. В задании предполагается, что работа происходит не прямо внутри файлов, то есть необходимо использовать функции чтения/записи из пользовательского меню, чтобы информация записалась в файл.

Кроме того, необходимо разработать следующую часть программной документации:

- Блок-схемы алгоритмов одного из модулей (не менее трех блок-схем, выбирать следует самые сложные алгоритмы)
- Описание тестовых случаев в виде таблиц со строками: предусловия, выполняемые действия, постусловия, ожидаемые результаты, результат тестирования.

Результаты учебной практики оформляются в виде **отчета**, содержимое которого указано ниже в соответствующем разделе.

Задания 0-2 уровней

Далее перечислены индивидуальные задания. Работа с первой таблицей реализуется студентами при выполнении на 0 уровне; с первой и второй – на 1 уровне; со всеми таблицами – на 2 уровне.

Студент может предложить свой вариант реализуемой им базы данных и согласовать его в течении первых двух пар семестра.

1 База данных студентов группы.

Таблица №1 (студентов). Поля: фамилия, имя, отчество, пол, возраст.

Таблица №2 (оценок). Поля: предмет, оценка, дата, описание оценки.

Таблица №3 (предметов). Поля: название, количество часов, , тип отчетности (зачет, экзамен, контрольная работа). При реализации таблицы №3 в таблице №2 указывается не название предмета, а ссылка на строку в таблице №3.

Поиск информации осуществляется по ФИО

2 База данных расходов семьи.

Таблица №1 (расходов). Поля: товар, стоимость, количество, дата.

Таблица №2 (товаров). Поля: название, единица измерения (кг, упаковка, единица), производитель. При использовании этой таблицы в таблица №1 вместо ввода названия товара выбирает товар из таблицы №2.

Таблица №3 (производителей). Поля: название и адрес. При использовании этой таблице вместо названия производителя в таблице №2 выбирается производитель из таблицы №3.

Сортировка осуществляется по дате расхода, поиск по названию товара.

3 База данных загрузки аудиторий.

Таблица №1 (загрузки аудиторий): дата и время начала, дата и время конца, аудитория, преподаватель.

Таблица №2 (аудиторий): название, количество мест, наличие проектора. При использовании этой таблицы в таблице №1 вместо ввода названия аудитории осуществляется выбор строки из таблицы №2.

Таблица №3 (преподавателей): Фамилия, имя, отчество, пол. При использовании этой таблицы в таблице №1 вмето ввода ФИО преподавателя осуществляется выбор строки из таблицы №3.

4 База данных учета доходов и расходов предпринимателя.

Таблица №1 (доходов/расходов): дата, тип операции (доход/расход), объем операции, описание, корреспондент.

Таблица №2 (корреспондентов): ФИО или название, адрес, телефон. При использовании таблицы №2 в таблице №1 вместо ввода названия корреспондента он выбирается из таблицы №2.

Таблица №3 (договоров): корреспондент, название, описание, дата начала действия, дата окончания действия.

5 База данных велоклуба.

Таблица №1 (членов велоклуба): ФИО, тип велосипеда (MTB и др.), стаж участия в велоклубе.

Таблица №2 (покатушек): название, количество км, дата, длительность.

Таблица №3 (участия в покатушках) – использование таблицы обозначает возможность указания информации об участии членов велоклуба в конкретных покатушках.

6 База данных рейсов авиакомпании.

Таблица №1 (рейсов): дата и время вылета, аэропорт вылета, аэропорт прилета, дата и время прилета, марка самолета.

Таблица №2 (аэропортов): код, город, название. Использование таблицы №2 обозначает, что в таблице №1 вместо ввода названия аэропортов осуществляется выбор аэропортов из таблицы №1.

Таблица №3 (стоимостей билетов для рейса): рейс, код кресла, стоимость.

7 База данных автобусных маршрутов.

Таблица №1 (маршрутов): номер маршрута, номер парка, времена начала и окончания движения, длина маршрута в км.

Таблица №2 (остановок): название, координаты GPS (можно хранить в виде строки).

Таблица №3 соответствия маршрутов и остановок.

8 База данных электричек.

Таблица №1 (рейсов): вокзал, номер поезда, количество вагонов, тип (экспресс/обычный/спутник).

Таблица №2 (остановок): название, адрес.

Таблица №3 (остановок рейса): к рейсу можно добавить остановки и указать время, когда на остановку прибывает и время, когда отправляется с нее электричка.

9 База данных товаров Интернет-магазина.

Таблица №1 (товаров): название товара, категория, цена товара, описание товара.

Таблица №2 (категорий): название категории. При использовании таблицы №2 в таблице №1 вместо ввода названия категории осуществляется выбор категории из таблицы №2.

Таблица №3 (остатки товара): товар, дата и время, количество оставшихся товаров.

10 База заказов Интернет-магазина.

Таблица №1 (заказов): ФИО заказчика, стоимость заказа, скидка (в процентах), адрес доставки.

Таблица №2 (строк заказа): заказ, товар, количество, цена товара с учетом количества. При реализации таблицы №2 в таблице №1 стоимость заказа рассчитывается автоматически.

Таблица №3 (товаров): название товара, описание товара, цена товара. При реализации таблицы №3 в таблице №2 вместо ввода названия товара выбирается товар из таблицы №3, а цена товара рассчитывается автоматически.

11 База данных выборов.

Таблица №1 (результатов): участок, кандидат, количество голосов.

Таблица №2 (участковых комиссий): адрес, количество зарегистрированных жителей, телефон, размер поселения. В случае реализации таблицы №2 в таблице №1 вместо ввода номера участка осуществляется его выбор из таблицы №2.

Таблица №3 (кандидатов): ФИО, годовой доход, пол, возраст. В случае реализации таблицы №3 в таблице №1 вместо указания ФИО кандидата осуществляется его выбор из таблицы №3.

12 База данных практических работ.

Таблица №1 (выполнения работ): практическая работа, студент, номер варианта, номер уровня, дата сдачи, оценка.

Таблица №2 (практических работ): название, количество часов. В случае реализации таблицы №2 в таблице №1 вместо указания названия практической работы осуществляется выбор ее из таблицы №2.

Таблица №3 (студентов): ФИО, оценка за прошлый семестр. В случае реализации таблицы №3 в таблице №1 вместо указания ФИО студента осуществляется выбор студента из таблицы №3.

13 База данных операторов и телеканалов.

Таблица №1 (операторов): Название, тип (спутник, кабель, Интернет), охват (кол-во миллионов домохозяйств), минимальная стоимость подписки.

Таблица №2 (телеканалов): название, охват (кол-во миллионов домохозяйств), краткое описание.

Таблица №3 предназначена для задания соответствия операторам наборов телеканалов.

14 База данных тарифных планов оператора.

Таблица №1 (телеканалов): название, тип вещания (обычный/HD), флаг общедоступности.

Таблица №2 (тарифных планов): название, стоимость.

Таблица №3 задает соответствие каждому тарифному плану набора телеканалов.

15 База данных незаконно огороженных берегов.

Таблица №1 (ограждений): водный объект, регион, GPS-координаты, длина недоступного участка берега, дата фиксации нарушения.

Таблица №2 (водных объектов): название, тип (озеро, река, водохранилище), площадь бассейна, длина. В случае реализации таблицы №2 в таблице №1 вместо ввода названия водного объекта он выбирается из таблицы №2.

Таблица №3 (регионов): название, номер, адрес правительства региона. В случае реализации таблицы №3 в таблице №1 вместо ввода названия региона он выбирается из таблицы №3.

16 База данных временного прекращения движения в метро.

Таблица №1 (прекращений движения): дата и время начала прекращения движения, дата и время окончания прекращения движения, станция, станция (от какой до какой станции прекращено движение).

Таблица №2 (станций): название станции, линия, GPS-координаты станции, пассажиропоток в сутки. В случае, если реализуется таблица №2, в таблице №1 вместо ввода названия станции она выбирается из таблицы №2.

Таблица №3 (линий): название линии, ее длина в километрах, год открытия первой станции. В случае, если реализуется таблица №3, в таблице №2 вместо ввода названия линии она выбирается из таблицы №3.

17 База данных проката фильмов.

Таблица №1 (сеансов): дата, время, кинотеатр, фильм, номер зала, тип сеанса (3D/Imax/обычный).

Таблица №2 (кинотеатров): название, количество залов, адрес, телефон. В случае реализации таблицы №2 в таблице №1 указывается не название кинотеатра, а осуществляется выбор кинотеатра из таблицы №1

Таблица №3 (фильмов): название, длина фильма, страна. В случае реализации таблицы №3 в таблице №1 вместо ввода названия фильма осуществляется его выбор из таблицы №3.

18 База данных эвакуированных автомобилей:

Таблица №1 (актов эвакуации): улица, автостоянка, GPS-координаты, тип нарушения (стоянка на проезжей части в месте запрета, стояна на тротуаре, стоянка на газоне), номер автомобиля, тип автомобиля (легковой/грузовой малой тонажности/грузовой большой тонажности).

Таблица №2 (улиц): название, длина. В случае реализации таблицы №2 в таблице №1 вместо ввода названия улицы осуществляется ее выбор из таблицы №2.

Таблица №3 (штрафных автостоянок): название, адрес, телефон. В случае реализации таблицы №3 в таблице №1 вместо ввода названия автостоянки осуществляется ее выбор из таблицы №3.

19 База данных средних специальных учебных учреждений.

Таблица №1 (учреждений): название, адрес, тип подчинения (федеральный/региональный), год основания, номер лицензии, номер аккредитации, дата окончания действия аккредитации.

Таблица №2 (контрольных цифр приема): учреждение, специальность, количество бюджетных мест, количество коммерческих мест.

Таблица №3 (специальностей): номер, название, длительность обучения (в случае, если оно начинается после 9 класса). В случае реализации таблицы №3 в таблице №2 вместо ввода названия специальности осуществляется ее выбор из таблицы №3.

20 База данных поселков.

Таблица №1 (поселков): название, девелопер, площадь, количество жителей.

Таблица №2 (домов): поселок, номер дома, площадь дома, количество этажей, тип дома.

Таблица №3 (девелоперов): название девелопера, годовой доход, адрес. В случае реализации таблицы №3 в таблице №1 вместо ввода названия девелопера осуществляется его выбор из таблицы №3.

21 База данных сухопутной военной техники.

Таблица №1 (техники): название, модель, разработчик, предприятие, стоимость, тип.

Таблица №2 (разработчиков): ФИО, год рождения, пол. В случае реализации таблицы №2 в таблице №1 вместо ввода ФИО разработчика осуществляется его выбор из таблицы №2.

Таблица №3 (предприятий): название, место нахождения, количество работников, директор. В случае реализации таблицы №3 в таблице №1 вместо ввода названия предприятия осуществляется его выбор из таблицы №3.

22 База данных деревьев в городе.

Таблица №1 (деревьев): GPS-координаты, вид дерева, округ, год посадки.

Таблица №2 (видов деревьев): название вида, средняя высота, признак вечной зелености. В случае реализации таблицы №2 в таблице №1 осуществляется не ввод названия вида дерева, а его выбор из таблицы №2.

Таблица №3 (округов): название, площадь, количество жителей. В случае реализации таблицы №3 в таблице №1 вместо ввода названия округа осуществляется выбор округа из таблицы №3.

23 База данных футбольных матчей.

Таблица №1 (матчей): дата, команда, команда, счет, место проведения.

Таблица №2 (команд): название, город, страна. В случае реализации таблицы №2 вместо ввода названий команд в таблице №1 осуществляется выбор команд из таблицы №2.

Таблица №3 (футболистов): команда, ФИО, зарплата в год, возраст.

24 База данных обращений жителей.

Таблица №1 (заявлений): дата, время, объект, заявитель, содержание обращения (до 255 символов), дата ответа, ответ на обращение (до 255 символов).

Таблица №2 (заявителей): ФИО, адрес, телефон. В случае реализации таблицы №2 в таблице №1 вместо ввода ФИО заявителя осуществляется его выбор из таблицы №2.

Таблица №3 (объектов): название, адрес, GPS-координаты. В случае реализации таблицы №3 в таблице №1 вместо ввода названия объекта осуществляется его выбор из таблицы №3.

25 База данных студентов колледжа.

Таблица №1 (студентов): ФИО, группа, признак бюджетности, стипендия (нет/обычная/повышенная), флаг наличия социальной стипендии, дата рождения.

Таблица №2 (групп): название, номер специальности, отделение. В случае реализации таблицы №2 в таблице №1 вместо ввода названия группы осуществляется выбор группы из таблицы №2.

Таблица №3 (отделений): название, ФИО заведующего, номер комнаты, где оно расположено. В случае реализации таблицы №3 в таблице №2 осуществляется выбор отделения из таблицы №3, а не ввод его названия.

26 База данных паевых инвестиционных фондов (ПИФ).

Таблица №1 (ПИФ): название, тип (интервальный, открытый, закрытый), стоимость пая на момент открытия, среднее изменение стоимости пая за год, управляющая компания.

Таблица №2 (управляющий компаний): название, дата регистрации. В случае реализации таблицы №2 в таблице №1 вместо ввода названия управляющей компании осуществвляется ее выбор из таблицы №2.

Таблица №3 (котировок): ПИФ, дата, стоимость пая.

27 База данных посылок для таможенной службы.

Таблица №1 (посылок): дата и время таможенного оформления посылки, отправитель, получатель, стоимость груза, вес посылки.

Таблица №2 (получателей): ФИО/название, тип (юридическое лицо/физическое лицо), адрес. В случае реализации таблицы №2 вместо ввода названия получателя осуществляется его выбор из таблицы №2.

Таблица №3 (отправитель): ФИО/название, тип (юридическое лицо/физическое лицо), адрес. В случае реализации таблицы №3 вместо ввода названия отправителя осуществляется его выбор из таблицы №3.

28 База данных занятости врачей в поликлинике.

Таблица №1 (занятости времени): дата, время начала интервала, время окончания интервала, тип приема (первичный/повторный), пациент, врач.

Таблица №2 (врачей): ФИО, специальность, номер кабинета. В случае реализации таблицы №2 в таблице №1 вместо ввода ФИО врача осуществляется его выбор из таблицы №2.

Таблица №3 (пациентов): ФИО, адрес, номер полиса, дата окончания срока действия полиса. В случае реализации таблицы №3 в таблице №1 вместо ввода ФИО пациента осуществляется его выбор из таблицы №3.

29 База данных запросов по исправлению ошибок в программе.

Таблица №1 (запросов): название, описание, дата регистрации запроса, дата завершения обработки запроса, ответственный программист, отправитель запроса.

Таблица №2 (программистов): ФИО, заработная плата. В случае реализации таблицы №2 в таблице №1 вместо ввода ФИО программиста осуществляется его выбор в таблице №2.

Таблица №3 (отправителей): ФИО, дата регистрации, версия продукта. В случае реализации таблицы №3 в таблице №1 вместо ввода отправителя осуществляется его выбор из таблицы №3.

30 База данных митингов.

Таблица №1 (митингов): дата, время, адрес, количество участников заявленное, количество участников по факту (может не указываться), список заявителей, флаг разрешения.

Таблица №2 (заявителей): ФИО, флаг наличия административных правонарушений. В случае реализации таблицы №2 вместо ввода списка заявителей в таблице №1 осуществляется их выбор в таблице №2.

Таблица №3 (правонарушений): митинг, ФИО нарушителя, нормативный акт, статья и пункт нормативного акта, флаг осуждения судом.

31 База данных новостей.

Таблица №1 (новостей): название, содержание, дата и время появления, агентство.

Таблица №2 (агенств): названием, e-mail адресом, телефоном.

Таблица №3 (связей новостей) с помощью таблицы указываются связи между новостями, позволяющие группировать их и выводить похожие новости в потенциально возможном будущем программном продукте.

3 Вторая часть учебной практики

3.1 Введение (для студентов)

Выполнение заданий второй части учебной практики базируется на изученном материале в МДК «Системное программирование», курсе «Операционные системы», курсе «Основы программирования» и МДК «Прикладное программирование». Все задания, кроме одного, выполняются на языке C или C++ и компилируются в зависимости от операционной системы в трансляторе clang или среде Microsoft Visual Studio.

Часть заданий выполняется в среде операционной системы семейства Linux (или в bash – подсистеме Linux в Windows), другие задания – в среде Windows.

Описание всех функций, можно найти на сайтах: `http://opennet.ru/` и `http://ru.cppreference.com/`, `http://msdn.microsoft.com`.

В заданиях запрещено вызывать команды shell (bash) или использовать другие способы их выполнения, кроме использования функций Posix/WinAPI или близких к нимс (exec, кроме явно указанных случаев, в частности, запрещен).

3.2 Методика оценивания

Зачёт ставится при условии выполнения 8 практических работ. Оценка зависит от сроков сдачи и количества сданных работ. Оценка за каждую работу зависит от срока сдачи работы. Работа не должна содержать ошибок (в том числе должны проверяться все возможные ошибки выполнения вызываемых функций).

Все студенты выполняют задания одного уровня сложности.

Оценка считается путем суммирования оценок за каждую работу и деления полученного числа на 10, а далее округления по правилам математики.

Зачет выставляется в случае предотавления отчета, дневника и приложений к нему.

3.3 Практическая работа №1. Работа с файловой системой в Linux (12 часов)

1. Цель работы: повторение языка C/C++; закрепление навыков работы со средой Linux; закрепление/ознакомление с функциями, предназначенными для работы с каталогами.

2. Порядок выполнения:

   (a) Осуществите повторение основ языка C (операторы и операции, описание и объявление функций, работа со строками, структурами, массивами);

   (b) Осуществите ознакомление со средой Linux или эмулятора Linux;

   (c) Изучите функции rename (stdio.h), opendir, closedir, readdir, scandir (dirent.h), unlink (unistd.h), stat (sys/stat.h); переменную errno, функцию strerror.

   (d) Напишите программу в соответствии с вашим вариантом.

   Все ошибки необходимо обрабатывать, способ обработки оставляется на усмотрение студента: в некоторых случаях надо вывести диагностическое сообщение и завершить работу программы; в некоторых случаях – вывести сообщение, но продолжить выполнение.

   Номера работ выбираются в соответствии с номером в журнале третьего курса.

   Все файлы, с которыми происходит работа, лежат в текущем каталоге.

   1 Удалить первый символ у имен обычных файлов (regular file) текущего каталога в случае, если в имени ровно 2 гласных латинских буквы.

   2 Удалить обычные файлы (regular file), имена которых читаются слева направо и справа налево одинаково

   3 Перед каждым именем файла (regular file) вставить его номер (1, 2, 3 и т. д.). Номер выставляется в соответствии с тем, в каком порядке выдает результаты функция readdir (или FindFirstFile, FindNextFile).

4 Удалить обычные файлы (regular file), имена которых имеют в своем составе ровно 2 согласных латинских буквы.

5 Во всех обычных файлах (regular file), имена которых состоят только из латинских букв, поменять порядок букв на противоположный.

6 Удалить обычные файлы (regular file), у которых расширение и основная часть имени совпадают.

7 У обычных файлов (regular file), у которых имя является целым неотрицательным числом прибавить к этому числу 1000. Предполагается, что изначально файлы имеют названия, числовое значение которых не превышает 999.

8 Удалить из имен обычных файлов (regular file) все цифры.

9 Удалить обычные файлы (regular file), длина которых совпадает с названием файла (в случае, если название файла – это целое неотрицательное число).

10 Переименовать обычные файлы (regular file), название которых содержит символ `!` в имя, совпадающее с его длиной.

11 К имени обычных файлов (regular file), содержащим символ подчеркивания, приписать после этого подчеркивания длину файла.

12 К имени обычных файлов (regular file), имеющих нечетную длину, приписать символ `!`, если он отсутствует.

13 Удалить обычные файлы (regular file), имя которых содержит символ `~` в случае, если длина их равна 0.

14 н Удалить обычные файлы (regular file), имя которых является целым неотрицательным числом и превышает размер файла.

15 Удалить у всех имен обычных файлов (regular file) знаки `~`, размещенные в конце имени.

16 Удалить все обычные файлы (regular file), у которых права доступа владельца включают возможность их запуска.

17 Заменить пробелы в именах всех обычных файлов (regular file) на знаки подчеркивания.

18 Удалить из имен обычных файлов (regular file) гласные буквы латинского алфавита.

19 Переставить символы в основной части всех имен обычных файлов (не в расширении) в обратном порядке.

20 Все обычные файлы, имеющие нулевую длину переименовать в имя, имеющее формат: `zero<номер>`, где `<номер>` – это числа попорядку, начиная с 1.

21 Удалить обычные файлы, которые в имени имеют хотя бы два символа, не являющиеся латинской буквой или цифрой.

22 В именах обычных файлов заменить символ `$` на длину файла.

23 Переименовать обычные файлы, к которым есть доступ только на чтение для владельца путем добавления к нему расширения `.read`

24 Переименовать обычные файлы, имена которых – целые неотрицательные числа на название того же числа в 16-ой системе счисления.

25 Дописать к именам обычных (регулярных) файлов (перед расширением, если оно есть) его длину в байтах.

26 Удалить из имен обычных (регулярных) файлов все гласные латинские буквы.

27 Все натуральные числа, находящиеся в именах обычных (регулярных) файлов перевернуть (записать цифры в обратном порядке)

28 Удалить все обычные (регулярные) файлы, не имеющие расширение, или имеющие расширение более 3 символов.

29 Удалить файлы, в которых в имени есть фрагмент `Del`

30 Переименовать файлы, имеющие в имени файла фрагмент `.to.` на последовательность символов, стоящую после этого фрагмента

31 Переименовать файл, в котором имя состоит из четного числа символов, в файл, в котором половинки этого имени переставлены местами

3. Содержание отчета:

(a) Исходный код написанного программного продукта

(b) Ответы на контрольные вопросы

4. Контрольные вопросы:

   (a) Как осуществляется перебор файлов в каталоге?

   (b) Как осуществляется проверка типа файла?

   (c) Как осуществляется переименование или удаление файла?

   (d) Как осуществляется проверка ошибок и получение информации о них?

3.4 Практическая работа №2. Работа с файловой системой в Windows (4 часа)

1. Цель работы: ознакомление с фукциями, предназначенными для работы с каталогами (WinAPI).

2. Порядок выполнения:

   (a) Ознакомьтесь/повторите функции FindFirstFile, FindNextFile, DeleteFile, MoveFile, DeleteFile, GetLastError, FormatMessage (windows.h)

   (b) Выполните задание практической работы №1 с использованием Visual Studio/C++/консольное приложение и функций WinAPI. Так как в Windows автоматически файлам назначает признак готовности к архивированию, обычными файлами следует считать и подготовленные к архивированию файлы. Кроме того, вместо текущего каталога можно использовать удобный для Вас каталог, который Вы явно укажете в исходном тексте программы.

3. Содержание отчета:

   (a) Исходный код написанного программного продукта

   (b) Ответы на контрольные вопросы

4. Контрольные вопросы:

   (a) Как осуществляется перебор файлов в каталоге?

   (b) Как осуществляется проверка типа файла?

   (c) Как осуществляется переименование или удаление файла?

   (d) Как осуществляется проверка ошибок и получение информации о них?

3.5 Практическая работа №3. Рекурсия (6 часов)

1. Цель работы: дополнительный практический опыт использования рекурсии

2. Порядок выполнения: Выполните задания практических работ №1 и №2 с учетом того, что файлы ищутся не только в текущем каталоге, но и во всех его подкаталогах (любой степени вложенности). Обратите внимание на то, что существуют фиктивные каталоги `.` и `..` в каждом каталоге (их надо игнорировать, иначе произойдет зацикливание).

3. Содержание отчета:

   (a) Исходный код написанных программных продуктов

   (b) Ответы на контрольные вопросы

4. Контрольные вопросы:

   (a) Как осуществляется перебор файлов в каталоге и во всех вложенных в него?

3.6 Практическая работа №4. Работа с параметрами командной строки (2 часа)

1. Цель работы: изучение способа обработки аргументов из командной строки

2. Порядок выполнения.

   (a) Изучите/повторите назначение аргументов функции main, традиционно называемых argc и argv.

   (b) Напишите программу, работающую в среде Linux и осуществляющую обработку параметров командной строки согласно своему варианту. При отсутствии аргументов программа должна выдавать инструкцию, при неверном числе аргументов – информацию об ошибке.

1 Напишите программу mycp, имеющую два аргумента и копирующая файл, имя которого стоит в первом аргумент, в файл, имя которого стоит во втором аргументе. Пример вызова: ./mycp test1 test2

2 Напишите программу myrm, имеющую один аргумент и удаляющую файл, имя которого задано. Пример вызова: ./myrm test.txt

3 Напишите программу mycat, имеющую один аргумент и выводящую содержимое файла на экран. Пример вызова: ./mycat test.txt

4 Напишите программу mytail, имеющую два аргумента: имя файла и число n – и выводящую содержимое последних n строк файла. Пример вызова: ./mytail test.txt 10

5 Напишите программу myhead, имеющую два аргумента: имя файла и число n – и выводящую содержимое первых n строк файла. Пример вызова: ./myhead test.txt 10

6 Напишите программу mymkdir, имеющую один аргумент и создающую каталог. Пример вызова: ./mymkdir dir1

7 Напишите программу mynl, имеющую один аргумент и выводящую содержимое файла с указанием номеров строк. Пример вызова: ./mynl test.txt

8 Напишите программу mywc, имеющую один аргумент и выводящую количество строк, слов и символов в файле.

9 Напишите программу mymv, имеющую два аргумента и перемещающая файл, имя которого стоит в первом аргумент, в файл, имя которого стоит во втором аргументе. Пример вызова: ./mymv test1 test2

10 Напишите программу exists, имеющую один аргумент – имя файла, и выводящую на экран 1, если файл существует и 0 – если нет. Пример вызова: ./exists test.txt

11 Напишите программу np2n, имеющую два аргумента – имена файлов, которая в первом файле последовательность байтов 13 и 10 заменяет на байт 10 и результат пишет во второй файл. Пример вызова: ./np2n test.txt test2.txt

12 Напишите программу n2np, имеющую два аргумента – имена файлов, которая в первом файле байт 10 заменяет на байт 13 и 10 и результат пишет во второй файл. Пример вызова: ./n2np test.txt test2.txt

13 Напишите программу dec2hex, имеющую один аргумент – десятичное натуральное число и выводящее соответствующее 16-ое число. Пример вызова: ./dec2hex 125

14 Напишите программу hex2dec, имеющую один аргумент – 16-ое натуральное число и выводящее соответствующее 10-ое число. Пример вызова: ./hex2dec F2F

15 Напишите программу isDir, имеющую один аргумент – имя файла и выводящая 1, если файл - каталог и 0 в противном случае. Пример вызова: ./isDir directory

16 Напишите программу zero, имеющую один аргумент – имя файла и заменяющую указанный файл на файл нулевой длины. Пример вызова: ./zero test.dat

17 Напишите программу exchange, имеющую два аргумента – имена файлов и меняющаю эти файлы местами. Пример вызова: ./exchange text1.txt file1.txt

18 Напишите программу cat2, имеющую три аргумента – имена файлов, которая содержимое первых двух файлов записывает подряд в третий файл. Пример вызова: ./cat2 file1.dat file2.dat result.dat

19 Напишите программу rev, имеющую один аргумент – имя файла, которая выводит содержимое файла, в котором каждую строку переворачивает наоборот (располагает символы в обратном порядке). Пример вызова: ./rev test.txt

20 Напишите программу sort, имеющую один аргумент – имя файла, которая выводит содержимое файла в алфавитном порядке (сортирует строки). Пример вызова: ./sort test.txt

21 Напишите программу uniq, имеющую один аргумент – имя файла, которая выводит все строки файла, исключая повторяющиеся строки (в любом порядке). Пример вызова: ./uniq test.txt

22 Напишите программу mysubst, имеющую три аргумента: строку s1, строку s2 и имя файла, которая выводит содержимое файла, заменяя подстроки s1 на s2. Пример вызова: ./mysubst Миша Мишутка test.txt

23 Напишите программу mytruncate, имеющую три аргумента: число l и название файла, которая данный файл обрезает, уменьшая его размер до l. Пример вызова: ./mytruncate 10 test.txt

24 Напишите программу mydd, имеющую три аргумента: число l и название двух файлов, которая копирует из первого файла во второй l байт. Пример вызова: ./mydd 10 test.txt test2.txt

25 Напишите программу myunexpand, имеющую два аргумента – имена двух файлов, которая в данном первом файле заменяет несколько пробелов, оканчивающихся на позициях, пропорциональных 8, на символ табуляции и выводит результат во второй файл. Пример вызова: ./myunexpand test.txt test2.txt

26 Напишите программу myexpand, имеющую два аргумента – имена двух файлов, которая в данном первом файле заменяет табуляции на пробелы, оканчивающиеся на позиции, пропорциональной 8, и выводит результат во второй файл. Пример вызова: ./myexpand test.txt test2.txt

27 Напишите программы `diff`, имеющую два агумента – имена двух файлов, которая сравнивает два файла на предмет их полного совпадения и выводит на экран: `true` или `false`.

28 Напишите программы `diff_ci`, имеющую два агумента – имена двух файлов, которая сравнивает два файла на предмет их полного совпадения за исключением возможной разницы регистров букв латинского алфавита и выводит на экран: `true` или `false`.

29 Напишите программы `diff_is`, имеющую два агумента – имена двух файлов, которая сравнивает два файла на предмет их полного совпадения за исключением возможной разницы количества пробелов между другими участками текста и выводит на экран: `true` или `false`.

30 Напишите программы `mytr`, имеющую три агумента – имя файла и два символа, которая в файле заменяет первый символ на второй или `false`.

31 Напишите программы `mytr_c`, имеющую три агумента – имя файла и два символа, которая в файле заменяет первый символ на второй, а второй – на первый. или `false`.

3. Содержание отчета:
   (a) Исходный код написанного программного продукта
   (b) Ответы на контрольные вопросы
4. Контрольные вопросы:
   (a) Как осуществляется работа с аргументами функции main?

3.7 Практическая работа №5. Создание новых процессов/потоков (14 часов)

1. Цель работы: изучение способа создания новых процессов.
2. Порядок выполнения:
   (a) Изучите функции fork, waitpid.
   (b) Реализуйте задание 3-ей практической работы таким образом, чтобы каждый каталог обрабатывался отдельным процессом (в среде Linux).
   (c) Изучите функции CreateThread, WaitForSingleObject, CloseHandle.
   (d) Реализуйте задание 3-ей практической работы таким образом, чтобы каждый каталог обрабатывался отдельным потоком (в среде Windows).
3. Содержание отчета:
   (a) Исходный код написанного программного продукта
   (b) Ответы на контрольные вопросы
4. Контрольные вопросы:
   (a) Как осуществляется работа с функциями fork и waitpid?
   (b) Как осуществляется работа с функциями CreateThread, WaitForSingleObject, CloseHandle.

3.8 Практическая работа №6. Создание новых нитей выполнения в WPF (6 часов)

1. Цель работы: изучение способа создания новых потоков в среде WPF.
2. Порядок выполнения:
   (a) Изучите классы Task и свойство Dispatcher.
   (b) В практической работе предмета «Прикладное программирование», осуществляющей работу с файлами, ListView и/или TreeView осуществите чтени файла в отдельном потоке (а ее результат корректно отображался после выполнения).

3. Содержание отчета:
   (a) Исходный код написанного программного продукта
   (b) Ответы на контрольные вопросы
4. Контрольные вопросы:
   (a) Как осуществляется работа с потоками в среде .NET?
   (b) Как передать информацию между потоками в среде .NET?

3.9 Практическая работа №7. Динамическое программирование (4 часа)

1. Цель работы: изучить методику решения задачи динамического программирования на простейшем примере.
2. Порядок выполнения:
   (a) Реализуйте задание в соответствии со своим вариантом в среде Linux (с обязательным использованием рекурсии и возвратом из функции результата, соответствующему каталогу, имя которого было передано в нее).

   1 Найти файл (выведите на экран его имя), имеющий наибольшую длину среди файлов, расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   2 Найти файл (выведите на экран его имя), имеющий наименьшую длину среди файлов, расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   3 Найти файл (выведите на экран его имя), имеющий наибольшую длину имени среди файлов, расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   4 Найти файл (выведите на экран его имя), имеющий наименьшую длину имени среди файлов, расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   5 Найти файл (выведите на экран его имя), имеющий наиболее позднее время последнего обращения в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   6 Найти файл (выведите на экран его имя), имеющий наиболее раннее время последнего обращения в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

   7 Найти количество файлов, имеющих размер больше 1Кб в текущем каталоге и его подкаталогах любой степени вложенности.

   8 Найти количество файлов, имеющих время последнего обращения более года назад в текущем каталоге и его подкаталогах любой степени вложенности.

   9 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число файлов, оканчивающихся на знак `~`. Если таких каталогов несколько, то выведите любой из них.

   10 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число файлов, имеющих нулевой размер. Если таких каталогов несколько, то выведите любой из них.

   11 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число файлов, имеющих расширение `.txt`. Если таких каталогов несколько, то выведите любой из них.

   12 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий суммарный наибольший размер файлов. Если таких каталогов несколько, то выведите любой из них.

   13 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий суммарный наименьший размер файлов. Если таких каталогов несколько, то выведите любой из них.

14 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число подкаталогов. Если таких каталогов несколько, то выведите любой из них.

15 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число файлов. Если таких каталогов несколько, то выведите любой из них.

16 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наименьшее число файлов. Если таких каталогов несколько, то выведите любой из них.

17 Найти каталог (выведите на экран его имя), имеющий наиболее позднее время последнего обращения в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

18 Найти каталог (выведите на экран его имя), имеющий наиболее раннее время последнего обращения в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

19 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наибольшее число файлов, доступных для чтения (см. `S_IRUSR`). Если таких каталогов несколько, то выведите любой из них.

20 Найти каталог (выведите его имя на экран) в текущем каталоге и его подкаталогах любой степени вложенности, имеющий наименьшее число файлов, доступных для чтения (см. `S_IRUSR`). Если таких каталогов несколько, то выведите любой из них.

21 Найти файл (выведите на экран его имя), имеющий наибольшее расширение среди файлов, расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

22 Найти файл (выведите на экран его имя), имеющий наименьшее расширение среди файлов среди имеющих расширение, расположенный в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

23 Найти файл (выведите на экран его имя), имеющий наибольшее число цифр в имени, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

24 Найти файл (выведите на экран его имя), имеющий наименьшее ненулевое число цифр в имени, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

25 Найти пустой каталог (выведите на экран его имя), имеющий наибольшую степень вложенности, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких каталогов несколько, то найдите любой из них.

26 Найти обычный (регулярный) файл, в названии которого наибольшее число цифр, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

27 Найти обычный (регулярный) файл, в названии которого наименьшее число цифр, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

28 Найти обычный (регулярный) файл, в названии которого использовано наименьшее число символов (при подсчете совпадающие символы учитываются один раз), среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

29 Найти обычный (регулярный) файл, в названии которого использовано наибольшее число символов (при подсчете совпадающие символы учитываются один раз), среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

30 Найти обычный (регулярный) файл, имя которого лексикографически наименьшее, среди расположенных в текущем каталоге или его подкаталоге любой степени вложенности. Если таких файлов несколько, то найдите любой из них.

3. Содержание отчета:

(a) Исходный код написанного программного продукта

(b) Ответы на контрольные вопросы

4. Контрольные вопросы:

(a) Как осуществляется разбиение вашей задачи на подзадачи?

3.10 Практическая работа №8. Использование каналов в Linux (10 часов)

1. Цель работы: изучение каналов в Linux
2. Порядок выполнения:

(a) Изучите функции pipe, close, read, write;

(b) Напишите программу практической работы №7, но каждый каталог должен обрабатываться отдельным потоком, а передача результата должна осуществляться через pipe.

3. Содержание отчета:

(a) Исходный код написанного программного продукта

(b) Ответы на контрольные вопросы

4. Контрольные вопросы:

(a) Опишите архитектуру вашего программного продукта.

3.11 Практическая работа №9. Разделяемая память и семафоры в Linux (8 часов)

1. Цель работы: изучение семафоров и разделяемой памяти
2. Порядок выполнения:

(a) Изучите/повторите следующие функции: `sem_open`, `sem_close`, `sem_post`, `sem_wait`, `sem_unlink`; `shm_open`, `ftruncate`, `mmap`, `munmap`, `shm_unlink`;

(b) Напишите программу практической работы №8 так, чтобы количество порождаемых процессов не превышало 5, и каждый из них, обрабатывая каталог, мог передать информацию о найденных файлах и быть готовым сканировать следующий каталог. Межпроцессное взаимодействие организуйте с помощью разделяемой памяти, семафоров и каналов. Искомый результат должен храниться в разделяемой области памяти.

3. Содержание отчета:

(a) Исходный код написанного программного продукта

(b) Ответы на контрольные вопросы

4. Контрольные вопросы:

(a) Опишите назначение разделяемой памяти и семафоров в вашей задаче.

3.12 Практическая работа №10. Разделяемая память, критические секции, каналы в Windows (6 часов)

1. Цель работы: изучение критических секций в WinAPI
2. Порядок выполнения:

(a) Изучите функции: InitializeCriticalSection, EnterCriticalSection, LeaveCriticalSection, DeleteCriticalSection, CreatePipe, ReadFile, WriteFile.

(b) Напишите программу практической работы №9 в среде Windows так, чтобы каждый каталог обрабатывался отдельным потоком, используя критические секции для регулирование межпотокового взаимодействия и каналы для передачи информации между потоками в тех случаях, когда это оптимальнее использования разделяемых переменных.

3. Содержание отчета:

(a) Исходный код написанного программного продукта

(b) Ответы на контрольные вопросы

4. Контрольные вопросы:

(a) Опишите архитектуру вашего программного продукта

4 Отчет

Отчет по учебной практике представляет собой документ, отражающий всю работу, проделанную студентом в ходе прохождения практики. Целью составления отчета является с одной стороны облегчение общей оценки практики руководителем практики, с другой стороны позволяет сформировать целостный образ о проделанной работе у студента.

4.1 Структура отчета

Отчет состоит из следующих элементов:

1. Титульный лист
2. Содержание
3. Введение
4. Основная часть
   (a) Разработка спецификаций различных компонент (ПК1.1)
   (b) Осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля (ПК1.2)
   (c) Выполнение отладки программных модулей с использованием специализированных программных средств (ПК1.3)
   (d) Выполнение тестирования программных модулей (ПК1.4)
   (e) Осуществление оптимизации программного кода модуля (ПК1.5)
   (f) Разработка компонентов проектной и технической документации с использованием графических языков спецификаций (ПК1.6)
5. Заключение
6. Приложения

Оформление отчета производится в соответствии с оформлением данного документа.

4.2 Титульный лист

Пример титульного листа приведен в приложении А.

4.3 Оформление отчета

Отчет оформляется на листах формата А4 (допускается его выполнение и рукописным способом). Поля: слева – 30 мм; сверху, снизу – 20 мм; справа – 10 мм.

Основные параметры: шрифт — Times New Roman (или аналогичный по характеристикам), размер — 14, полуторный интервал, выравнивание по ширине, первая строка абзаца оформляется с отступом 1,5 см. Заголовки выравниваются по центру. Заголовки верхнего уровня располагаются на новых страницах, после них отступ 18 пт, используется полужирное начертание и все символы в них – заглавные. Заголовки второго уровня имеют те же характеристики, но отступы после них – 12 пт и располагаться они могут в любой части страницы. При этом следующие строки не должны печататься на следующей странице, а заголовок не может разрываться.

Все страницы (кроме титульного листа и задания) должны быть пронумерованы (номер указывается в правом верхнем углу). В нумерации титульный лист учитывается. Расстояние от номера страницы до текста документа должно составлять не менее 10 мм.

Рисунки имеют сквозную нумерацию и подписываются снизу, например «Рисунок 1» (без кавычек); рисунки, размещенные в приложениях имеют нумерацию в пределах приложения и подписываются в соответствии с примером: «Рисунок А.1», где А — обозначение приложения, а 1 — номер рисунка. После номера рисунка указывается название рисунка.

Аналогично, таблицы имеют сквозную нумерацию, подписываются сверху с выравниванием вправо («Таблица 1 Ошибки»). На все таблицы должны быть ссылки из текста отчета; обычно, таблица имеет нарисованные рамки, ограничивающие как саму таблицу, так и ячейки.

Таблицы и рисунки должны иметь сквозную нумерацию и названия, в основном тексте на каждый рисунок и таблицу должна быть ссылка. При необходимости можно нумеровать так же некоторые формулы.

Перед списком обязательно должно быть пояснение. Ненумерованный список должен оформляться с использованием тире.

Далее приводятся требования, являющиеся выдержками из государственных стандартов или общепринятых правил. Приведенные требования регламентируют те ситуации, что наиболее часто встречаются при выполнении работ. В остальных случаях Вы можете руководствоваться государственными стандартами ЕСКД (единая система конструкторской документации), правилами, принятыми при выполнении других работ в колледже и в других учебных заведениях.

4.3.1 Требования машинописи

Текст должен быть оформлен в соответствии с основными требованиями машинописи:

- перед основными знаками препинания (кроме тире) пробел не ставится, после — ставится всегда; тире и дефисы должны отличаться друг от друга (тире длиннее, дефис короче);
- дробная и целая часть числа отделяются друг от друга запятой (кроме исходного текста программы);
- сноски оформляются способом, который по умолчанию применяется в редакторах Microsoft Word или Libreoffice Writer, но в конце сносок устанавливается закрывающаяся скобка (как в данном документе).
- не допускается использование двух и более пробелов вместо одного (в редакторах можно отследить выполнение этого требования путем включения просмотра невидимых символов)

4.4 Введение

щВо введении студент излагает постановку задачи (краткую формулировку заданий), цель и задачи учебной практики (изложение ведется на основе анализа введения сборника практических заданий).

4.5 Основная часть

В основной части студент излагает обзор всей проделанной работы.

4.5.1 Разработка спецификаций различных компонент (ПК1.1)

В разделе «Разработка спецификаций различных компонент (ПК1.1)» студент описывает[4] как он осуществлял разработку спецификаций, в частности

- приводит ссылку на приложение, где размещена спецификация модулей (часть interface)

4.5.2 Осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля (ПК1.2)

В разделе «Осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля (ПК1.2)» студент описывает свою работу над кодом, в частности, приводит ссылку на приложения, содержащие исходные коды разработанных программных продуктов. Исходный код приводится шрифтом, размер которого не меньше 8pt.

Кроме того, в этом разделе студент проводит обзор разработанных программных продуктов, включая копии экрана с целью прояснить достоинства разработанного им программного обеспечения.

4.5.3 Выполнение отладки программных модулей с использованием специализированных программных средств (ПК1.3)

В разделе «Выполнение отладки программных модулей с использованием специализированных программных средств (ПК1.3)» студент описывает, как он проводил отладку своей программы. В частности, приводит примеры наиболее содержательных (для него) ошибок и способов их устранения и использованные им функции, предоставляемый средой разработки.

4.5.4 Выполнение тестирования программных модулей (ПК1.4)

В разделе «Выполнение тестирования программных модулей (ПК1.4)» студент описывает, как он проводил тестирование. В частности, приводит ссылку на приложение, содержащее описание тестовых случаев

[4] конкретное содержание и способ изложения выбирает студент самостоятельно. Это замечание относится ко всему наполнению отчета

4.5.5 Осуществление оптимизации программного кода модуля (ПК1.5)

В разделе «Осуществление оптимизации программного кода модуля (ПК1.5)» студент описывает те оптимизации, что он проводил во время написания программных продуктов, приводя показательные примеры таких оптимизаций.

4.5.6 Разработка компонентов проектной и технической документации с использованием графических языков спецификаций (ПК1.6)

В разделе «Разработка компонентов проектной и технической документации с использованием графических языков спецификаций (ПК1.6)» студент приводит блок-схемы[5] для наиболее сложных алгоритмов, содержащихся в разработанных им модулях.

4.6 Заключение

В заключении студент описывает достигнутые им результаты на основе введения: он показывает, что все цели и задачи, приведенные во введении достигнуты.

Кроме того, студент описывает **личный** взгляд на результаты практики: чему он научился, в чем изменился, какие сделал личностные и профессиональные выводы и так далее.

[5]указанные блок-схемы не должны повторять программный код, так как они предназначены для чтения человеком

Приложение А. Титульный лист

Министерство образования и науки Российской Федерации
ФГБОУ ВО «МГУТУ им. К. Г. Разумовского (ПКУ)»
Университетский колледж информационных технологий

Отчет по учебной практике
ПМ.01 «Разработка программных модулей программного обеспечения компьютерных сетей»,
студента группы П-203, обучающегося специальности
09.02.03 «Программирование в компьютерных системах»

Выполнил студент: Иванов И. И.
Преподаватель: Глускер А. И.

Москва
2020