МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных технологий и компьютерных систем

И З У Ч Е Н И Е
СОВРЕМЕННЫХ ТЕХНОЛОГИЙ УПРАВЛЕНИЯ РАЗРАБОТКОЙ ПРОГРАММНЫХ ПРОЕКТОВ

Методические указания
по выполнению заданий производственной
(эксплуатационной) практики
для студентов дневной формы обучения направлений
09.03.01 – Информатика и вычислительная техника
и 09.03.04 – Программная инженерия

Севастополь
СевГУ
2023

УДК 004.42(076.5)
ББК 32.973-018я73
И395

Р е ц е н з е н т :
В. Ю. Карлусов - канд. техн. наук, доцент кафедры информационных систем

Составители: Т. В. Волкова, Е. С. Владимирова

И395 Изучение современных технологий управления разработкой программных проектов : методические указания по выполнению заданий производственной (эксплуатационной) практики для студентов дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия / Севастопольский государственный университет», Институт информационных технологий, кафедра информационных технологий и компьютерных систем ; сост. : Т. В. Волкова, Е. С. Владимирова. – Севастополь : СевГУ, 2023. – 28 с. – Текст : электронный.

Целью методических указаний является оказание помощи студентам при выполнении заданий в ходе производственной практики. Набор заданий, предусмотренных к выполнению, позволит студентам получить практические навыки в использовании систем контроля версий и систем удаленного хранения репозиториев при организации командной работы над программным проектом, а также в оформлении презентаций и технических отчетов о проделанной работе.

Предназначены для студентов второго курса дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия.

УДК 004.42(076.5)
ББК 32.973-018я73

Методические указания рассмотрены и рекомендованы к изданию на заседании кафедры «Информационные технологии и компьютерные системы», протокол № 2 от 14 сентября 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель проектно-технологической практики – расширить теоретические и практические знания, полученные в процессе обучения по дисциплинам «Программирование. Языки высокого уровня», «Алгоритмизация вычислительных процессов», «Объектно-ориентированное программирование», получить навыки командной работы при разработке программного обеспечения, получить начальные навыки в разработке бизнес-планов и технических заданий на оснащение отделов, лабораторий, офисов компьютерным и сетевым оборудованием, освоить возможности пакета Microsoft Office по разработке презентаций, оформлению научно-технических отчетов по результатам выполненной работы.

Основными задачами проектно-технологической практики являются:

- углубление и закрепление полученных теоретических знаний;
- ознакомление на практике с методологиями управления проектами разработки программного обеспечения, такими как Scrum, Kanban, WaterFall;
- получение навыков управления компьютером с помощью работы в командной строке;
- практическое применение интерактивной среды разработки IDE для улучшения процесса написания программного кода;
- изучение возможностей систем сборки проектов, на примере Gradle;
- получение навыков в использовании системы контроля версии (типа Git) для хранения, отслеживания и внесения изменений в программный проект;
- получение навыков совместной работы над проектом через Github/Gitlab/Bitbucket/Azure (по выбору команды студентов);
- изучение возможностей Pull Requests / Merge Requests, как способов командной работы с 1 кодом;
- получение навыков тестирования программ с помощью unit-тестов (на примере Unit-5 для Java);
- автоматический запуск тестов в CI/CD в выбранной платформе совместной работы;
- получение практических навыков разработки программного проекта;
- получение навыков взаимодействия в командной работе над учебным программным проектом;
- получение навыков в подготовке презентаций и технических отчетов средствами Microsoft Office.

Учебная технологическая практика базируется на следующих учебных дисциплинах:

- Информатика;
- Программирование. Языки высокого уровня;
- Алгоритмизация вычислительных процессов;
- Объектно-ориентированное программирование.

Изучение этих дисциплин позволяет студентам, в результате успешного усвоения теоретических курсов, иметь знания, умения и компетенции для освоения программы учебной практики.

Содержание практики.

1) Подготовительный этап включает: организационное собрание, заполнение документов на практику, изучение теоретического материала по основам командной работы над программным проектом, разбиение студентов на команды и распределение ролей в команде, получение индивидуального задания.

2) Этап разработки и отладки программ в рамках распределенной работы над программным проектом: выполнение индивидуальных заданий и мероприятия для осуществления контроля за ходом разработки.

3) Заключительный семинар посвящен обмену опытом командной работы над проектом с использованием различных методологий.

4) Защита отчета по ознакомительной практике производится на комиссии кафедры, как правило, в последние два дня практики (не позднее установленного срока). Комиссия, после сообщения студента о результатах практики, вопросов и обсуждения, объявляет оценку.

Студенты при прохождении практики обязаны в конце дня записывать выполненную работу в дневнике-графике.

1. ВЫПОЛНЕНИЕ ОБЗОРА ЗАДАЧ, ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ, ТЕХНОЛОГИЙ УПРАВЛЕНИЯ КОМАНДНОЙ РАЗРАБОТКОЙ ПРОГРАММНЫХ ПРОЕКТОВ НА СОВРЕМЕННЫХ ПРЕДПРИЯТИЯХ IT-ОТРАСЛИ

В соответствующем разделе отчета по практике рекомендуется привести краткое описание задач и современных технологий программирования, реализуемых предприятием, на котором студент проходит практику. Более подробно должны быть описаны направления, в которых принимал участие студент. Студенты, проходящие практику в университете могут описать интересующие их направления (например, задачи, технологии и средства разработки (фреймворки) современного web-программирования).

В данном разделе рекомендуется также дать краткое описание возможностей систем контроля версий и систем управления репозиториями (более подробно Git, GitLab), а также современных технологий управления командной разработкой программных проектов (например, Scrum).

2. ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОГО ЗАДАНИЯ 1: «УПРАВЛЕНИЕ КОМПЬЮТЕРОМ С ПОМОЩЬЮ КОМАНДНОЙ СТРОКИ И ФАЙЛОВОГО МЕНЕДЖЕРА FAR MANAGER»

Варианты и порядок выполнения практического задания 1 описаны в [1]. В разделе отчета по практике, посвященного выполнению задания 1, должны присутствовать приведенные ниже подразделы.

2.1. Цель работы.

2.2. Постановка задачи (с данными варианта задания).

2.3. Этапы выполнения и результаты работы. Здесь должен быть приведен отчет о выполнении каждого пункта задания со скриншотами (там, где это требуется).

2.4. Вывод.

3. ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОГО ЗАДАНИЯ 2: «КОМАНДНАЯ РАБОТА НАД ПРОГРАММНЫМ ПРОЕКТОМ»

Перед выполнением практического задания 2 рекомендуется ознакомиться с теоретическими сведениями, приведенными в [5-8], а также выполнить следующие действия:

1) установить сервис VPN (Windscribe (https://windscribe.com/download) или Proton (https://protonvpn.com/ru/) и включить его;
2) зарегистрироваться на сервере GitLab;
3) установить Git (https://git-scm.com/downloads).
4) установить пакет разработчика и IDE с поддержкой Git (например, jdk-11.0.10.9-hotspot и IntelliJ IDEA https://www.jetbrains.com/ru-ru/idea/download/#section=windows (версия Community или бесплатная 30-дневная версия Ultimate или полная бесплатная версия для студентов).

Примечание 1. – В начале работы нужно открыть GitBash в папке, которую вы собираетесь сделать локальным репозиторием (перейти в папку, сделать правый щелчок мыши, в контекстном меню выбрать пункт Git Bash Here), затем выполнить команды, указав свой username и email (рекомендуется указать username и email, которые вы использовали при регистрации на GitLab):

git config --global user.name "My Name"

git config --global user.email myEmail@example.com

Указанные команды выполняются 1 раз после установки Git на локальный компьютер, настройки будут действовать для всех проектов.

После этого нужно выполнить команду:

git init

для инициализации репозитория (в текущей папке появится папка .git. Чтобы ее увидеть включите режим отображения скрытых папок).

Краткое описание остальных команд Git дано в [6].

Примечание 2. – Если вы планируете загружать проект (непустой) из локального репозитория в пустой проект на GitLab, не рекомендуется инициализировать создаваемый для этого проект на GitLab файлом readme (очевидно, чтобы не создавалась ветка main).

В разделе отчета по практике, посвященного выполнению задания 2, должны присутствовать приведенные ниже подразделы.

3.1. Постановка задачи разработки программного обеспечения

Студенты, проходящие практику на предприятиях, описывают задачу, которую им поставил руководитель практики от профильной организации.

Студенты, проходящие практику в СевНТУ могут взять для описания один из проектов, разработанных ими ранее [2-3] (проект должен включать не менее трех классов), либо (по желанию) выбрать для разработки один из проектов, описанных в приложениях А-В и привести соответствующую постановку задачи. Учтите, что проекты, которые приведены в приложении В и описаны в [4] очень полезны для получения опыта в программировании, но требуют для своей реализации достаточно много времени (за период практики реализовать их маловероятно).

3.2. Структура проекта и описание разработанного ПО

Студенты, проходящие практику на предприятиях, могут привести общую структуру проекта в рамках решаемой задачи, подробно описав ту подсистему, в разработке которой они принимали участие, а также ПО, которое они разработали.

Студенты, которые проходят практику в университете, должны привести структуру проекта и описание классов.

В конце раздела должна быть отсылка к приложению: «Текст программы приведен в Приложении А».

3.3. Организация процесса командной работы над проектом

В данном разделе необходимо привести таблицу распределения ролей и заданий в команде (пример – таблица 3.1) и таблицу с описанием процесса разработки (пример – таблица 3.2). Не забывайте сделать отсылки ко всем таблицам и рисункам в тексте отчета по практике. В основе примера – проект IntellectMatr, описанный в лабораторной работе №8 [3].

Таблица 3.1 – Распределение ролей и заданий в команде

№ п/п	ФИО	Роль	Задания
1.	Иванов Александр Петрович	team leader	1) Разработка первой версии проекта IntellectMatr (без собственных исключений). 2) Внесение изменений в класс IO: добавление выброса и обработки собственных исключений в методы ввода и вывода матрицы). 3) Разработка Unit-тестов для класса IntellectMatr.
2.	Петрова Ольга Николаевна	developer	1) Разработка класса MyException. 2) Внесение изменений в класс Intellect Matr: добавление методов нахождения номера первого столбца матрицы, содержащего отрицательные элементы и метода удаления всех столбцов, содержащих отрицательные элементы.

Таблица 3.2 – Этапы командной работы над проектом

1.	Иванов Александр Петрович	Создание начальной версии Java-Gradle-проекта IntellectMatr – локального репозитория под контролем Git с ветками master и developer (1 commit в ветке master, не менее 2 commit в ветке developer), загрузка на GitLab (если планируется загружать готовый проект из локального репозитория,, не рекомендуется инициализировать созданный для этого проект на GitLab файлом readme), приглашение участников в проект. В ветке master – первая версия (готовый работающий код: сборка, запуск и проверка в IntelliJ Idea). Все остальные изменения производятся в ветке developer)
2.	Петрова Ольга Николаевна	Клонирование репозитория с GitLab в локальный репозиторий. Добавление класса MyException в ветке developer. Отправка ветки в глобальный репозиторий.
3.	Иванов Александр Петрович	Скачивание изменений из глобального репозитория в локальный (слияние веток developer глобального и локального репозитория), внесение изменений в класс IO (добавление выброса и обработки собственных исключений класса MyException). Передача изменений из ветки develop в глобальный репозиторий.
4.	Петрова Ольга Николаевна	Скачивание изменений из глобального репозитория в локальный (слияние веток developer глобального и локального репозитория), изменение класса Intellect Matr (добавление методов нахождения номера первого столбца матрицы, содержащего отрицательные элементы и метода удаления всех столбцов, содержащих отрицательные элементы). Передача изменений из ветки develop в глобальный репозиторий.
5.	Иванов Александр Петрович	Скачивание изменений из глобального репозитория в локальный (слияние веток developer глобального и локального репозитория), разработка Unit-тестов для класса IntellectMatr. Проверка кода в ветке developer (сборка и запуск в IntelliJ Idea). Передача изменений из ветки develop в глобальный репозиторий. Выполнение запроса на слияние веток master и develop в ветку master.

Далее в этом пункте своего отчета по практике каждый член команды приводит описание последовательности своих действий в командной строке GitBash (рекомендуется) или использования возможностей IDE по интеграции с Git, которые в конечном итоге привели к решению заявленных в таблице 2.1 задач. Описание должно сопровождаться соответствующими скриншотами окон GitBash, IntelliJ Idea, GitLab, оформленными в виде рисунков с отсылкой к ним в тексте отчета. В частности, после каждого выполненного commit (фиксации изменений в текущей ветке) должен быть приведен скриншот вывода команды git log.

В качестве заключительного скриншота в данном пункте рекомендуется привести изображение окна GitLab с историей проекта.

Для экономии ресурсов принтера при выполнении скриншотов рекомендуется выбирать дизайн «черным по белому» окон GitBash, IntelliJ Idea или преобразовать цвет окон в приложении Fotoshop.

Примечание. Работа над проектом может быть распределена на б*о*льшее, чем показано в таблицах 3.1 и 3.2, число членов команды. Если студент выполняет второе задание один, то он должен смоделировать работу команды, состоящей из двух человек (завести 2 профиля на GitLab и два локальных репозитория).

3.4. Отладка и тестирование программы

Если при разработке программы не использовалось модульное (unit) тестирование, то в этом пункте можно указать, какие средства отладки использовались (например, отладчик, встроенный в IDE) и констатировать: анализ результатов запуска программы в среде разработки на тестовых примерах показывает корректность выполнения всех функций, предложенных для разработки.

Если юнит-тестирование использовалось, то нужно привести краткое описание технологии юнит-тестирования, необходимую программную поддержку (например, возможности Unit5) и разработанные юнит-тесты с их описанием.

3.5. Результаты выполнения программы

В данном пункте нужно привести скриншот (скриншоты) выполнения основных функций программы. Оформить в виде рисунков с отсылкой к ним в тексте отчета.

4. СОДЕРЖАНИЕ ОТЧЕТА ПО ПРАКТИКЕ

Отчет по практике (не менее 20 страниц) включает в себя структурные единицы, которые отображены в таблице 4.1:

В разделе «ВВЕДЕНИЕ» описываются цели и задачи практики (перечислены в разделе «Введение» данных методических указаний), указываются компетенции, которые должен освоить студент в процессе прохождения практики (перечислены в дневнике практики). Далее нужно дать краткий перечень технологий, с которыми студент ознакомился во время практики и краткое содержание разделов отчета по практике (1-2 предложения по каждому разделу).

Содержание разделов 1, 2 и 3 приведено выше (разделы 1-3 данных методических указаний).

Таблица 4.1 – Структурные единицы отчета по практике

Титульный лист
Индивидуальное задание на практику
Содержание
Введение
1. Обзор задач, технологий программирования, технологий управления командной разработкой программных проектов на современных предприятиях IT-отрасли
2. Практическое задание 1. Управление компьютером с помощью командной строки и файлового менеджера Far manager
3. Практическое задание 2. Командная работа над программным проектом
Заключение
Библиографический список
Приложение А

Раздел «ЗАКЛЮЧЕНИЕ» констатирует решение всех задач, описанных в разделах 1 (выполнен обзор …), 2 и 3, содержит обобщение практических результатов, изложенных в отчете.

Раздел «БИБЛИОГРАФИЧЕСКИЙ СПИСОК» отражает все источники, которые использовались при выполнении индивидуального задания и написании отчета по практике). На каждый источник, приведенный в списке должна быть ссылка в тексте отчета.

В разделе «ПРИЛОЖЕНИЕ А» приводится текст разработанного программного обеспечения.

Отчет должен быть оформлен в строгом соответствии с требованиями [9], приведенными в приложении Г к данным методическим указаниям.

5. ТИПОВЫЕ ЗАДАНИЯ ДЛЯ РАЗРАБОТКИ ПРОГРАММНЫХ ПРОЕКТОВ

Как говорилось выше, для выполнения практического задания 2 студент может использовать любой из разработанных им ранее проектов, содержащих не менее трех классов, или разработать новый проект из приведенного ниже перечня.

5.1. Тип 1 – программные проекты средней сложности (мини-проекты)

Перечисленные ниже мини-проекты могут выполняться как командой, состоящей из нескольких студентов, так и одним студентом (индивидуально).

Калькулятор. Реализовать простейший калькулятор, который дает возможность пользователю складывать, умножать, вычитать, делить числа, находить их квадратный корень.

Генератор паролей. Создать генератор случайных паролей для своих друзей и семьи, чтобы обезопасить их учётные записи!

Игра-викторина. Приложения-викторины задают пользователям серию вопросов и дают им возможность ответить. Затем викторина даёт пользователю результаты. Реализовать программу-викторину на любую интересную тему.

Игра "Угадай число". В данном мини-проекте необходимо реализовать игру, в которой компьютер пытается угадать целое число от 1 до 1000. Пользователь загадывает число, а программа задает ему вопросы из разряда Да или Нет в попытках угадать число. Если программа угадает число, она спрашивает у пользователя, то ли число она угадала. Если да - то «ураaa!». Если нет, программа показывает грустный смайлик и предлагает новую игру.

Программа-шифровщик. В данном мини-проекте необходимо реализовать программу, которая позволяет как шифровать сообщение, так и дешифровать сообщение. Предполагается, что обладая знанием ключа/ключей, студент может зашифровать сообщение, а другой студент - расшифровать, и наоборот.

Описание вышеперечисленных мини-проектов детализировано в Приложении А.

Интеллектуальная матрица. Создать программу, реализующую объект «Числовая матрица», способный выполнять заданные математические операции над инкапсулированной матрицей. Матрица должна вводиться из текстового файла и выводиться в текстовый файл. При обнаружении неправильного формата матрицы должны выбрасываться и обрабатываться соответствующие пользовательские исключения.

Варианты заданий для мини-проекта «Интеллектуальная матрица» приведены в Приложении Б.

5.2. Тип 2 – программные проекты повышенной сложности

Задание по практике предполагает разработку программной системы, моделирующей работу простого компьютера. Программная модель состоит из трех частей, которые должны выполняться тремя студентами в команде. Первая часть состоит в разработке программы-ассемблера, которая будет переводить программу в мнемокодах в машинные коды. Вторая часть – создание программы-симулятора компьютера, которая должна выполнять машинную программу, состоящую из команд, входящих в систему команд данного компьютера. Третья часть – разработка графического интерфейса, позволяющего управлять программами, созданными в первой и второй частях, современным способом. Варианты заданий на практику отличаются набором команд (системой команд компьютера), структурой, размером и разрядностью оперативной памяти, структурой процессора (приложение В). Пример разработки подобного проекта приведен в [4].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. «Работа с файлами и папками с помощью приложения «Командная строка» и файлового менеджера Far-manager»: методические указания к выполнению практического задания по дисциплине «Производственная (эксплуатационная) практика» для студентов дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия / Сост. Т.В. Волкова, Е.С.Владимирова – Севастополь: СевГУ, 2022.– 37 с.

2. «Использование коллекций и карт отображений при разработке java-программ»: методические указания по выполнению заданий учебной практики для студентов дневной формы обучения направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия / Сост. Т.В. Волкова, Е.С. Владимирова. – Севастополь: СевГУ, 2020. – 101 с.

3. Программирование. Языки высокого уровня. Методические указания к лабораторным работам по дисциплине «Программирование. Языки высокого уровня» для студентов направлений 09.03.01 – Информатика и вычислительная техника и 09.03.04 – Программная инженерия дневной формы обучения. [В 3 частях]. Часть 3. Использование классов библиотеки Java / Министерство образования и науки Российской Федерации, ФГАОУ ВО «Севастопольский государственный университет», Институт информационных технологий и управления в технических системах, кафедра информационных технологий и компьютерных систем ; сост. Т. В. Волкова, Е. С. Владимирова. – Севастополь : [Изд-во СевГУ], 2020. – 96 с.

4. «Разработка программных моделей вычислительных систем»: методические указания к выполнению курсового проекта по дисциплине "Программирование" / Сост. А.К.Васильченко, – Севастополь: Изд-во СевНТУ, 2010.–64 с.

5. Git-local-branching-on-the-cheap: учебник по Git. – Электронный ресурс. – Режим доступа: https://git-scm.com/book/ru/

6. Работа с Git. – Электронный ресурс. – Режим доступа: https://vk.com/@miffoul-rabota-s-git

7. Как пользоваться GitLab. – Электронный ресурс. – Режим доступа: https://losst.ru/kak-polzovatsya-gitlab

8. Getting Started with Gradle. – Электронный ресурс. – Режим доступа: https://www.jetbrains.com/help/idea/getting-started-with-gradle.html

ПРИЛОЖЕНИЕ А

Детализация описания мини-проектов, представленных в п. 5.1

1. Калькулятор

Реализовать простейший калькулятор, который дает возможность пользователю складывать, умножать, вычитать, делить числа, находить их квадратный корень.

КАК пользователь
КОГДА я ввожу два числа
ТОГДА я увижу сумму двух чисел

КАК пользователь
КОГДА я ввожу два числа
ТОГДА я могу увидеть разность двух чисел

КАК пользователь
КОГДА я ввожу два числа
ТОГДА я увижу произведение двух чисел

КАК пользователь
КОГДА я ввожу два числа
И первое число является числителем
И второе является знаменателем и *не равно* нулю
ТОГДА я увижу частное двух чисел

КАК пользователь
КОГДА я ввожу два числа
И первое число является числителем
И второе является знаменателем и *равно* нулю
ТОГДА я должен увидеть ошибку

---

2. Генератор паролей

Создайте генератор случайных паролей для своих друзей и семьи, чтобы обезопасить их учётные записи!

Реализовать простейший генератор паролей, который дает возможность пользователю получить пароль, удовлетворяющий его требованиям.

КАК пользователь
КОГДА я открываю программу
И указываю количество обычных символов
ТОГДА я вижу сгенерированный пароль
И этот пароль содержит выбранное количество обычных символов

КАК пользователь
КОГДА я открываю программу
И указываю количество специальных символов
ТОГДА я вижу сгенерированный пароль
И этот пароль содержит выбранное количество специальных символов

КАК пользователь
КОГДА я открываю программу
И указываю количество букв в верхнем регистре
ТОГДА я вижу сгенерированный пароль
И этот пароль содержит выбранное количество букв в верхнем регистре

КАК пользователь
КОГДА я открываю программу
И указываю различные критерии(разное число обычных, специальных, в верхнем регистре символов и цифр)
ТОГДА я вижу сгенерированный пароль
И этот пароль должен удовлетворять *всем* требованиям

---

3. Игра-викторина

Приложения-викторины задают пользователям серию вопросов и дают им возможность ответить. Затем викторина даёт пользователю результаты. Реализовать программу-викторину на любую интересную вам тему. Да, можно про Наруто. Да, можно не про аниме. На любую тему :3

КАК пользователь
КОГДА я открываю программу
ТОГДА программа *предлагает мне сыграть в игру (с)* и ответить на 10 вопросов

КАК пользователь
КОГДА я не соглашаюсь сыграть в игру
ТОГДА программа благодарит за участие и завершается

КАК пользователь
КОГДА я соглашаюсь сыграть в игру
ТОГДА программа задает мне вопрос и дает возможность ответить

КАК пользователь
КОГДА я даю ответ на вопрос
ТОГДА программа задает мне следующий вопрос

КАК пользователь
КОГДА я даю ответ на последний (десятый) вопрос
ТОГДА программа показывает результаты(сколько очков заработано
И программа предлагает сыграть снова

---

4. Игра "Угадай число"

В данном мини-проекте вам необходимо реализовать игру, в которой компьютер пытается угадать целое число от 1 до 1000. Пользователь загадывает число, а программа задает ему вопросы из разряда *Да* или *Нет* в попытках угадать число. Если программа угадает число, она спрашивает у пользователя, это ли число она угадала. Если да - то урааа! Если нет - программа показывает грустный смайлик и предлагает новую игру. *Предполагается, что пользователь играет и отвечает на вопросы честно, а не как всегда*.

КАК пользователь
КОГДА я открываю программу
ТОГДА программа *предлагает мне сыграть в игру (с)*

КАК пользователь
КОГДА я не соглашаюсь сыграть в игру
ТОГДА программа благодарит за участие и завершается

КАК пользователь
КОГДА я соглашаюсь сыграть в игру
ТОГДА программа предлагает мне загадать число от 1 до 1000
И после этого начинает задавать вопросы из-разряда *Да/Нет*

КАК пользователь
КОГДА я даю ответ на вопрос *Да*
ТОГДА программа задает мне следующий вопрос

КАК пользователь
КОГДА я даю ответ на вопрос *Нет*
ТОГДА программа задает мне следующий вопрос

КАК пользователь
КОГДА я даю ответ на вопрос, который не является *Да* или *Нет*
ТОГДА программа просит ввести ответ снова, так как *у нее лапки*

КАК пользователь
КОГДА я даю ответ на вопрос
И программа предполагает что угадала число
ТОГДА программа спрашивает - такое ли число было загадано
И предлагает дать ответ на вопрос *Да* или *Нет*

КАК пользователь
КОГДА я даю ответ на вопрос
И программа угадала число
ТОГДА программа показывает результаты - как быстро она угадала, сколько шагов и какое число было загадано
И программа предлагает сыграть снова

КАК пользователь
КОГДА я даю ответ на вопрос
И программа не число
ТОГДА программа показывает грустный смайлик
И программа предлагает сыграть снова

---

5. Программа-шифровщик

В данном мини-проекте вам необходимо реализовать программу, которая позволяет как шифровать сообщение, так и дешифровать сообщение. Предполагается, что обладая знанием ключа/ключей, вы можете зашифровать сообщение, а ваш друг - расшифровать, и наоборот.

КАК пользователь
КОГДА я открываю программу
ТОГДА программа предлагает либо зашифровать текст, либо дешифровать текст

КАК пользователь
КОГДА я ввожу текст для шифрования
И я ввожу ключ для шифрования
ТОГДА программа показывает зашифрованный текст

КАК пользователь
КОГДА я ввожу текст для дешифрования
И я ввожу ключ для дешифрования
ТОГДА программа показывает расшифрованный текст

Приложение Б

Варианты заданий для мини-проекта «Интеллектуальная матрица» (поля и методы класса IntellectMatr), представленного в п. 5.1

Вариант 1.
Разработать класс «Интеллектуальная матрица» (IntellectMatr).
Поля: 1) ссылка на прямоугольную (квадратную) матрицу (public), тип элементов матрицы – float.
Методы:
1) конструктор (параметр – ссылка на матрицу);
2) процедура вывода матрицы в окно терминала (без параметров);
3) функция, возвращающая значение суммы элементов главной диагонали матрицы (без параметров);
4) функция, возвращающая значение определителя матрицы (без параметров);
5) функция, возвращающая адрес вектора (одномерного массива), i-ый элемент которого равен количеству отрицательных элементов в i-ой строке матрицы (без параметров);
6) функция, возвращающая адрес транспонированной матрицы (без параметров).

Разработать класс MatrDemo, содержащий статические методы:
1) процедура вывода вектора (одномерного массива) в окно терминала (параметр – ссылка на вектор);
2) метод main, демонстрирующий применение методов класса «Интеллектуальная матрица» для двух матриц различной размерности.

Вариант 2. Разработать класс «Интеллектуальная матрица» (IntellectMatr).
Поля: 1) ссылка на прямоугольную (квадратную) матрицу (public), тип элементов матрицы – double.
Методы:
1) конструктор (параметр – ссылка на матрицу);
2) процедура вывода матрицы в окно терминала (без параметров);
3) функция, возвращающая значение суммы элементов под главной диагональю матрицы (без параметров);
4) функция, возвращающая значение определителя матрицы (без параметров);
5) функция, возвращающая адрес вектора (одномерного массива), i-ый элемент которого равен сумме положительных элементов i-ой строки матрицы (без параметров);
6) функция, возвращающая адрес матрицы-произведения инкапсулированной матрицы на другую матрицу (параметр – ссылка на матрицу).

Разработать класс MatrDemo, содержащий статические методы:
1) процедура вывода вектора (одномерного массива) в окно терминала (параметр – ссылка на вектор);
2) метод main, демонстрирующий применение методов класса «Интеллектуальная матрица» для двух матриц различной размерности.

Вариант 3. Разработать класс «Интеллектуальная матрица» (IntellectMatr).
Поля: 1) ссылка на прямоугольную (квадратную) матрицу (public), тип элементов матрицы – int.
Методы:
1) конструктор (параметр – ссылка на матрицу);
2) процедура вывода матрицы в окно терминала (без параметров);
3) функция, возвращающая значение произведения элементов над главной диагональю матрицы (без параметров);

4) функция, возвращающая значение определителя матрицы (без параметров);
5) функция, возвращающая адрес вектора (одномерного массива), i-ый элемент которого равен произведению положительных элементов i-го столбца матрицы (без параметров);
6) функция, возвращающая адрес транспонированной матрицы (без параметров).

Разработать класс MatrDemo, содержащий статические методы:
1) процедура вывода вектора (одномерного массива) в окно терминала (параметр – ссылка на вектор);
2) метод main, демонстрирующий применение методов класса «Интеллектуальная матрица» для двух матриц различной размерности.

Вариант 4. Разработать класс «Интеллектуальная матрица» (IntellectMatr).
Поля: 1) ссылка на прямоугольную (квадратную) матрицу (public), тип элементов матрицы – double.
Методы:
1) конструктор (параметр – ссылка на матрицу);
2) процедура вывода матрицы в окно терминала (без параметров);
3) функция, возвращающая значение максимального элемента под второй (не главной) диагональю матрицы (без параметров);
4) функция, возвращающая значение определителя матрицы (без параметров);
5) функция, возвращающая адрес вектора (одномерного массива), i-ый элемент которого равен числу положительных элементов в i-ом столбце матрицы (без параметров);
6) функция, возвращающая адрес матрицы-произведения инкапсулированной матрицы на другую матрицу (параметр – ссылка на матрицу).

Разработать класс MatrDemo, содержащий статические методы:
1) процедура вывода вектора (одномерного массива) в окно терминала (параметр – ссылка на вектор);
2) метод main, демонстрирующий применение методов класса «Интеллектуальная матрица» для двух матриц различной размерности.

Вариант 5. Разработать класс «Интеллектуальная матрица» (IntellectMatr).
Поля: 1) ссылка на прямоугольную (квадратную) матрицу (public), тип элементов матрицы – int.
Методы:
1) конструктор (параметр – ссылка на матрицу);
2) процедура вывода матрицы в окно терминала (без параметров);
3) функция, возвращающая значение минимального элемента над второй (не главной) диагональю матрицы (без параметров);
4) функция, возвращающая значение определителя матрицы (без параметров);
5) функция, возвращающая адрес вектора (одномерного массива), i-ый элемент которого равен единице, если в i-ом столбце матрицы есть положительные элементы, и нулю, в противном случае;
6) функция, возвращающая адрес транспонированной матрицы (без параметров).

Разработать класс MatrDemo, содержащий статические методы:
1) процедура вывода вектора (одномерного массива) в окно терминала (параметр – ссылка на вектор);
2) метод main, демонстрирующий применение методов класса «Интеллектуальная матрица» для двух матриц различной размерности.

ПРИЛОЖЕНИЕ В

Варианты задания, описанного в п. 5.2 − Описание структуры процессора, системы команд и способов кодирования

Структура 1 (рисунок В.1). Описание и система команд (таблица В.1)

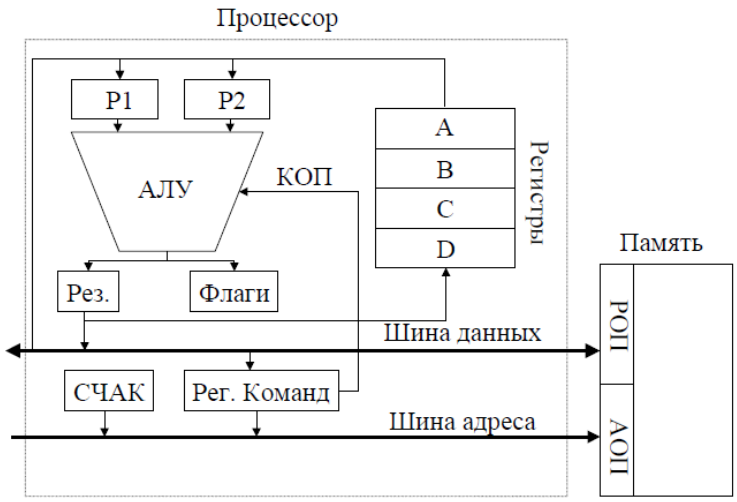

Рисунок В.1 – Структура 1

Объем оперативной памяти − 256 байт.

Длина команд − один или два байта. В первом байте биты 0..3 являются кодом операции, а следующие интерпретируются в зависимости от принадлежности команд к той или иной группе:

- для арифметических и логических команд содержат номера РОН (0-A, 1-B, 2-C, 3-D), над которыми производятся операции;
- для команд пересылки информации − номер регистра, с которым осуществляется операция;
- не используются для команд передачи управления и управления процессом.

Второй байт используется для задания адреса памяти, где находится операнд, адреса перехода для команд передачи управления или константы.

Таблица В.1 – Система команд для структуры 1

Код	Мнемокод и операнды	Описание
0000	ADD регистр1,регистр2	регистр1=регистр1+регистр2
0001	ADD регистр, память	регистр=регистр+память
0010	AND регистр1,регистр2	регистр1=регистр1&регистр2
0011	OR регистр1,регистр2	регистр1=регистр1 | регистр2
0100	NOT регистр	инверсия битов регистра
0101	MOV регистр,константа	занести константу в регистр
0110	LOAD регистр, адрес	загрузка данных в регистр из памяти
0111	STORE регистр, адрес	запись данных из регистра в память
1000	JMP адрес	безусловный переход
1001	JZ адрес	переход, если 0
1010	JO адрес	переход, если переполнение
1011	JL адрес	переход, если меньше
1100	JNZC регистр, адрес	переход, если не ноль, декремент регистра
1101	CLF	установка регистра флагов в 0
1110	CALL адрес	переход к подпрограмме, адрес возврата в регистре D
1111	RET	возврат из подпрограммы

Структура 2 (рисунок В.2). Описание и система команд (таблица В.2)

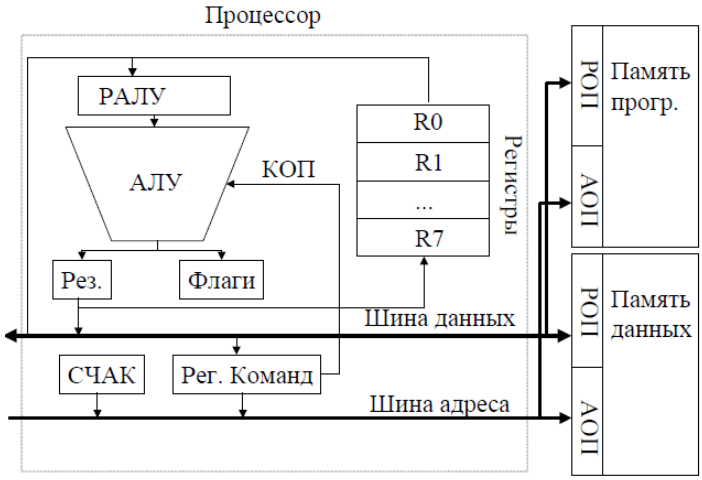

Рисунок В.2 – Структура 2

Объем оперативной памяти данных − 256 байтов, объем оперативной памяти программ − 256 байтов. Данные и программы хранятся раздельно.

Таблица В.2 – Система команд для структуры 2

Код	Мнемокод и операнды	Описание
00000	ADD РОН	рез=РОН+рез
00001	ADD адрес	рез=память[адрес]+рез
00010	ADC РОН	рез=РОН+рез+флаг переполнения
00011	SUB РОН	рез= рез-РОН
00100	SUB адрес	рез=рез-память[адрес]
00101	SBB РОН	рез=рез-РОН-флаг переполнения
00110	AND РОН	рез=РОН&рез
00111	AND адрес	рез=память[адрес]&рез
01000	OR РОН	рез=РОН | рез.
01001	OR адрес	рез=память[адрес] | рез
01010	XOR РОН	рез=РОН⊗рез.
01011	XOR адрес	рез=память[адрес]⊗рез
01100	MUL РОН	рез=рез*РОН
01101	DIV РОН	рез=рез/РОН (деление нацело, остаток – в РОН+1)
01110	LDR РОН	рез=РОН
01111	SVR РОН	РОН=рез
10000	CALL РОН,адрес	переход к подпрограмме; адрес возврата сохраняется в РОНе;
10001	RET РОН	возврат из подпрограммы; адрес возврата находится в РОНе
10010	INC РОН	увеличение РОНа на 1
10011	DEC РОН	уменьшение РОНа на 1
10100	MOV РОН,константа	занести в РОН константу
10101	MOV РОН,адрес	РОН= память[адрес]
10110	MOV память,РОН	память[адрес]=РОН
10111	LOAD РОН	рез=память[адрес=РОН]
11000	SAVE РОН	память[адрес=РОН]=рез
11001	JMP адрес	безусловный переход
11010	JZ адрес	переход, если 0
11011	JNZ адрес	переход, если не 0
11110	HALT	останов
11111	CLR	установка регистра результата в 0

Команды имеют длину один или два байта. Первый байт определяет код команды в соответствии с таблицей команд (бит 0-4), а также номер регистра-операнда (биты 5-7), указывающего на один из РОНов процессора.

Второй байт определяет адрес операнда в памяти, или задает константу.

Арифметические и логические операции выполняются над операндом, находящимся в памяти или регистре, и результатом предыдущей операции, который находится в регистре результатов. Флаг переполнения учитывается как единица младшего разряда, если используется в арифметических командах.

Структура 3 (рисунок В.3). Описание и система команд (таблица В.3)

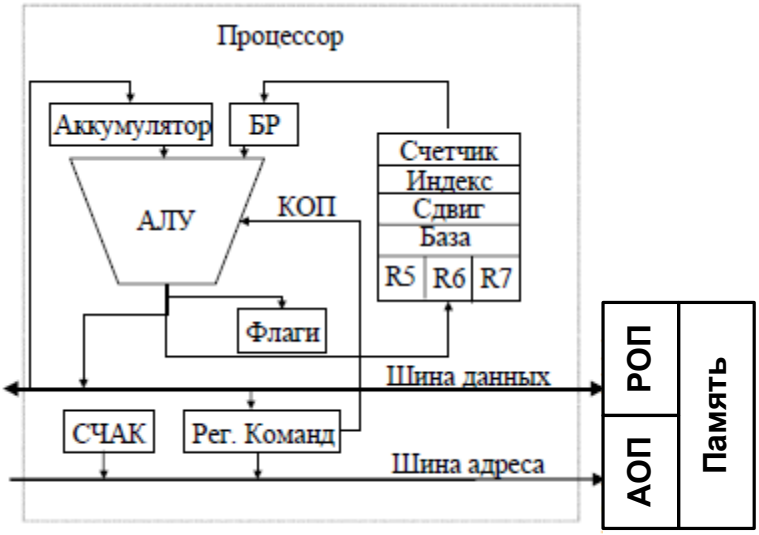

Рисунок В.3 – Структура 3

Объем оперативной памяти − 256 байт. Микропроцессор имеет 8 регистров, 5 из них − специального назначения:
РОН0 (АС). аккумулятор (в него заносятся результаты выполнения всех операций);
РОН 1 (CNT) . счетчик (для операций увеличения (уменьшения) на 1);
РОН 2 (SH) . сдвиговый регистр (для выполнения команд сдвига);
РОН 3 (BASE) . регистр базы (содержит базовый адрес операнда в ОП);
РОН 4 (IND) . регистр индекса (для формирования смещения операнда в ОП).

Таблица В.3 – Система команд для структуры 3

Код	Мнемокод	Описание
0000	ADD регистр	аккумулятор= аккумулятор + регистр
0001	ADC регистр	аккумулятор= аккумулятор + регистр + флаг переполнения
0010	SUB регистр	аккумулятор= аккумулятор - регистр
0011	SBB регистр	аккумулятор= аккумулятор - регистр - флаг переполнения
0100	AND регистр	аккумулятор=аккумулятор & регистр
0101	OR регистр	аккумулятор=аккумулятор | регистр
0110	NOT	аккумулятор=не аккумулятор
0111	MOV регистр, константа	регистр=константа
1000	STOS	память[BASE+IND]= аккумулятор IND=IND+1
1000	LODS*	Аккумулятор =память[BASE+IND], IND=IND+1
1001	OUTR регистр, адрес	память[адрес]= регистр
1001	INR* регистр, адрес	регистр= память[адрес]
1010	LOOP адрес	переход по адресу, если CNT>0, CNT=CNT-1
1011	JNZ адрес	переход, если не 0
1100	JB адрес	переход, если результат меньше 0
1101	SWAP регистр	Обмен данными между аккумулятором и регистром
1110	SHR	Сдвиг вправо регистра SH на число разрядов в регистре CNT
1111	SHL	Сдвиг влево регистра SH на число разрядов в регистре CNT

Коды команд:
Команды имеют длину один или два байта. Биты 0-3 первого байта определяют код команды в соответствии с таблицей команд. Номер регистра хранится в битах 4-6 первого байта. Бит 7 указывает направление перемещения данных: 0 означает перемещение данных *в* оперативную память, в остальных случаях он равен 1. Второй байт определяет константу или адреса памяти. Арифметические и логические операции выполняются над операндами, находящимися в аккумуляторе и (или) регистре.

Структура 4 (рисунок В.4). Описание и система команд (таблица В.4)

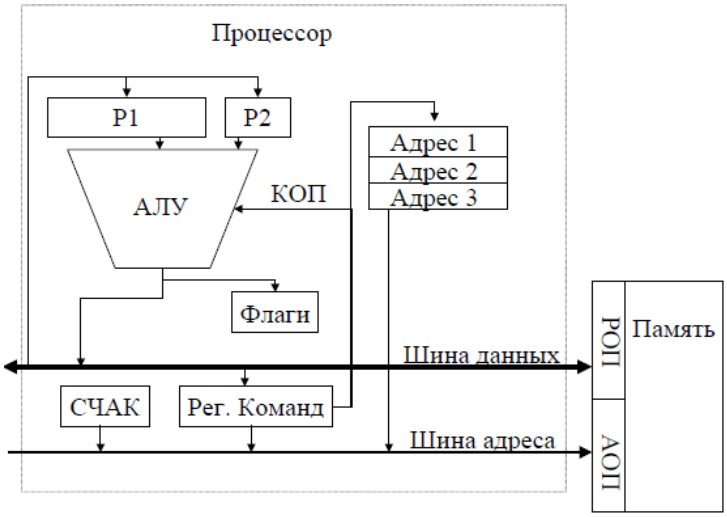

Рисунок В.4 – Структура 4

Объем оперативной памяти . 2048 байт (8 страниц по 256 байт).

Регистр СЧАК имеет 12 битов, но старший бит не используется.

Микропроцессор имеет три адресных регистра. Регистры АДРЕС1 (А1) и АДРЕС2 (А2) содержат адреса операндов команд, регистр АДРЕС3 (А3) . адрес результата операции. Эти регистры имеют по 8 битов и определяют смещение в диапазоне 0-255 внутри страницы. Номер страницы (целое число от 0 до 7) задан в битах 5-7 регистра команд.

Команды имеют длину один или два байта. Биты 0-4 первого байта определяют код команды в соответствии с таблицей команд. Биты 5-7 первого байта задают номер страницы ОП, в которой располагаются данные. Второй байт определяет константу.

Таблица В.4 – Система команд для структуры 4

Код	Мнемокод	Описание
00000	MOVA1 константа	A1=константа
00001	MOVA2 константа	A2=константа
00010	MOVA3 константа	A3=константа
00100	ADD страница*	Память[A3]= Память[A1]+ Память[A2]
00101	ADC страница	Память[A3]= Память[A1]+ Память[A2]+флаг переп.
00110	SUB страница	Память[A3]= Память[A1]- Память[A2]
00111	SBB страница	Память[A3]= Память[A1]- Память[AC2] -флаг переп.
01000	MUL страница	Память[A3]= Память[A1]* Память[A2]
01001	DIV страница	Память[A3]= Память[A1]/ Память[A2]
01010	MOD страница	Память[A3]= Память[A1]% Память[A2]
01011	ABS страница	Память[A3]= | Память[A1] |**
01100	AND страница	Память[A3]=Память[A1]& Память[A2]
01101	OR страница	Память[A3]= Память[A1] | Память[A2]
01110	XOR страница	Память[A3]= Память[A1] ^ Память[A2]
01111	NOT страница	Память[A3]=not Память[A1]
10000	JMP страница	Переход по адресу A3
10001	JB страница	Переход по адресу A3, если результат меньше 0
10010	JNZ страница	Переход по адресу A3, если результат не равен 0
10101	MOVS	Память[A3]=Память[A1], A3+=1, A1+=1
10101	CMPS	Сравнение данных Память[A1] и Память[A3]***

** отрицательные числа представляются в дополнительном коде. Для нахождения модуля необхо-
димо выполнить перевод в прямой код и обнулить старший бит

*** выполняется аналогично операции вычитания, но результат операции не заносится в память

Структура 5 (рисунок В.5). Описание и система команд (таблица В.5)

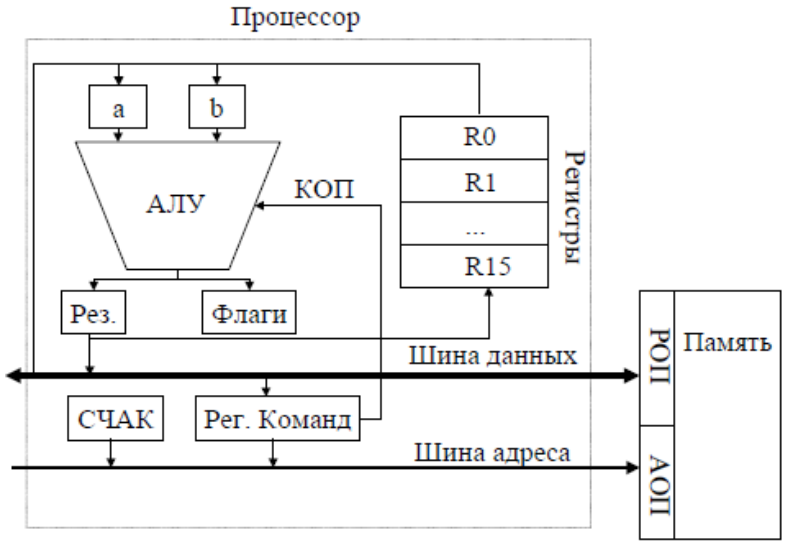

Рисунок В.5 – Структура 5

Объем оперативной памяти . 256 байт. Буферные регистры a и b предназначены дляхранения операндов операций. Рабочий регистр Рез используется как регистр результата, полученного при обработке данных в АЛУ.

Коды команд:

Команды имеют длину один или два байта. Первый байт определяет код команды в соответствии с таблицей команд (биты 0-3). Следующие 4 бита предназначены для указания номера регистра, коммутируемого к буферному регистру a.

Второй байт используется для задания константы или адреса памяти, где находится операнд. Если в качестве второго операнда используется регистр, то его номер хранится в младших 4-х битах второго байта команды, а старшие 4 бита не используются.

Таблица В.5 – Система команд для структуры 5

Код	Мнемокод	Описание
0000	ADNAB регистр,регистр	$\bar{a}+b$
0001	ADANB регистр,регистр	$a+\bar{b}$
0010	ADAB регистр,регистр	$a+b$
0011	ANBONAB регистр,регистр	$a\bar{b} \vee \bar{a}b$
0100	ABONANB регистр,регистр	$ab \vee \overline{ab}$
0101	NAB регистр,регистр	$\overline{ab}$
0110	ANB регистр,регистр	$a\bar{b}$
0111	JZ адрес	Переход, если результат равен 0
1001	JL адрес	Переход, если результат меньше 0
1010	JMP адрес	Безусловный переход
1100	MOVC регистр, константа	Регистр:=Константа
1101	MOVR регистр, адрес	Регистр:= Память[адрес]
1110	MOVM регистр, адрес	Память[адрес]:=регистр
1111	CMP регистр, регистр	Сравнение двух регистров. Установка флагов.

ПРИЛОЖЕНИЕ Г

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ТЕХНИЧЕСКОГО ОТЧЕТА ПО ПРАКТИКЕ

1. Оформление отчета по практике должно соответствовать требованиям к текстовым учебным документам соответствующих ГОСТов [Электронный фонд правовой и нормативно-технической документации. http://docs.cntd.ru].
2. Текстовая часть отчета по практике выполняется с использованием печатающих и графических устройств на одной стороне листа белой бумаги формата А4 с параметрами: кегль - 14; шрифт - TimesNewRoman, обычный; цвет шрифта - черный; поля, не менее:

верхнее - 20 мм; левое-30 мм;
нижнее - 20 мм; правое - 15 мм.

Параметры абзаца:
выравнивание по ширине;
уровень – основной текст;
отступ слева – 0;
отступ справа – 0;
интервал перед – 0;
интервал после – 0;
первая строка – отступ, значение – 1 см;
междустрочный интервал − полуторный;

3. Автоматическая расстановка переносов в основном тексте (исключая титульный лист и заголовки пунктов, подпунктов, рисунков и таблиц).

4. Названия разделов с одинарной нумерацией (1, 2, …) и без нумерации (ВВЕДЕНИЕ, ЗАКЛЮЧЕНИЕ, БИБЛИОГРАФИЧЕСКИЙ СПИСОК, ПРИЛОЖЕНИЕ А) выполняются прописными буквами. В отчете требуется выделить эти разделы полужирным шрифтом и применить к ним выравнивание по центру. Такие пункты располагаются с новой страницы. При этом после пункта пропускается одна строка. Точка в конце названия пункта не ставится.

5.Названия пунктов с двойной и тройной нумерацией выполняют строчными буквами (первая – прописная), начинают с абзаца, до и после названия пункта пропускают одну строку, точка в конце названия не ставится. Можно выделить эти пункты полужирным шрифтом. Нельзя, чтобы название пункта осталось на одной странице, а текст пункта начинался со следующей страницы.

6. При использовании маркированного списка, в качестве маркера можно применять только тире. Маркированный список начинается после предложения с двоеточием (перечисление), каждый пункт – с маленькой буквы, после пункта – точка с запятой (не может быть двух предложений в одном пункте). Нельзя, чтобы предложение с двоеточием осталось на одной

странице, а пункты списка полностью были перенесены на следующую страницу. В одноуровневом нумерованном списке рекомендуется использовать формат номера со скобкой. В таком списке может присутствовать несколько предложений в одном пункте (если только этот список не используется в качестве аналога маркированного списка).

7.Требования к оформлению таблиц. Название таблицы (Таблица номер – Название) приводится над таблицей и выравнивается по левому краю таблицы. Нумерация таблиц осуществляется в рамках пункта с одинарной нумерацией, как это сделано в методических указаниях (Таблица 2.1 – Варианты заданий). Либо можно использовать сквозную нумерацию таблиц в рамках всего документа. Точка в конце названия таблицы не ставится. На таблицу обязательно нужно сделать ссылку в тексте отчета (например, «Варианты заданий приведены в таблице 2.1.»). При этом соответствующая таблица должна быть помещена на той же или на следующей странице. Если вся таблица не помещается на текущей страницы, то часть таблицы (дублируя заголовок) можно перенести на следующую страницу. Вместо названия таблицы перед следующей частью таблицы приводится фраза «Продолжение таблицы номер» (например, Продолжение таблицы 2.1). Выравнивается фраза по левому краю таблицы.

8.Требования к оформлению рисунков. Название рисунка (Рисунок номер – Название) приводится под рисунком (подрисуночная подпись). Точка в конце названия не ставится. Нумерация рисунков производится подобно нумерации таблиц. Подрисуночная подпись выравнивается по центру рисунка. На рисунок, как и на таблицу, обязательно должна быть ссылка в тексте отчета (например, «Результаты выполнения программы изображены на рисунке 2.1.»). Сокращения слова «рисунок» ни в подрисуночной подписи, ни в ссылке на рисунок не допускаются. Рисунок должен быть приведен на той же странице, где находится ссылка на него, или на следующей.

9.Рисунок, как и таблицу, можно расположить на нескольких страницах. Тогда на первой странице в качестве подрисуночной подписи приводится название рисунка (Рисунок номер – Название), а на остальных страницах – фраза продолжения (Продолжение рисунка номер).

10.Исходя из вышесказанного, необходимо отметить, что у таблицы и рисунка номер может состоять из одной позиции (если применяется сквозная нумерация в рамках отчета) или из двух позиций, разделенных точкой (если используется нумерация в рамках раздела – более удобный способ). **Рисунков и таблиц с тройной и т. д. нумерацией быть не может!**

11.В библиографическом списке должно быть не менее трех источников, записанных в соответствии с ГОСТ. На каждый из источников должна быть ссылка в тексте отчета в квадратных скобках. Источники приводятся в библиографическом списке в алфавитном порядке или в порядке ссылок на них в тексте отчета.

12.Чертежи **(если они предусмотрены заданием по практике**) выполняются по ГОСТу в стандартном формате (А1, А2, А3, А4). Масштабирование рамок не допускается. Все элементы чертежа должны быть

свободно читаемыми (не мельчить!). В тексте отчета должна быть ссылка на каждый чертеж по его шифру (год, номер группы, номер варианта, номер чертежа, разделенные точками). Если чертеж большой, можно сделать его многолистовым, располагая части чертежа (например, схемы алгоритма) на отдельных листах формата А4 с использованием соответствующих штампов (большого на первом листе чертежа и усеченных на следующих листах) и блоков межстраничного перехода.

13.Рекомендуемый минимальный объем отчета по практике – 20 листов. Страницы отчета нумеруют арабскими цифрами, с соблюдением сквозной нумерации по всему тексту. Номер проставляется в правом верхнем углу без точки в конце номера.

14. Тексты программ, приводимые в отчете по практике, рекомендуется форматировать следующим образом: Arial, 12, полужирный, выравнивание по левому краю, без абзацев, черный текст на белом фоне, комментарии выделяются курсивом. Делать скриншоты текстов программ и вставлять в отчет в виде рисунков не допускается.

15. Требования по содержанию отчета по практике. Отчет должен содержать все разделы и подразделы, перечисленные в документе «СОДЕРЖАНИЕ ОТЧЕТА ПО ПРАКТИКЕ».

16. Нумерация страниц отчета считается с титульного листа. Лист задания (двухсторонний документ) считается за 2 страницы. Затем идет лист СОДЕРЖАНИЕ (приведенное на нем содержание отчета должно быть получено методом автооглавления, т.е. страницы, указанные в содержании, должны соответствовать страницам в тексте, на которых расположены указанные разделы и пункты). На титульном листе, листе задания и листе содержания страницы не ставятся. Таким образом, раздел ВВЕДЕНИЕ начинается со страницы с номером 5. Номера страниц приводятся в верхнем правом углу листа (на всех листах, кроме листа задания печать односторонняя).

СТАНДАРТЫ

1. ГОСТ 2.109-73. Единая система конструкторской документации (ЕСКД). Основные требования к чертежам (с Изменениями N 1-11). М. : Стандартинформ, 2011. — Текст : непосредственный.

2. ЕСПД. ГОСТ 19.701−90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992-01-01. – М : Государственный стандарт Союза ССР. — Текст : непосредственный.

3. ГОСТ Р 7.0.5—2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления : национальный стандарт Российской Федерации : утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 28 апреля 2008 г. N 95-ст. : дата введения 1 января 2009 года / разработан: Федеральным государственным

учреждением "Российская книжная палата" Федерального агентства по печати и массовым коммуникациям. — М. : Стандартинформ, 2008. — 23 с. — Текст : непосредственный

И З У Ч Е Н И Е СОВРЕМЕННЫХ ТЕХНОЛОГИЙ УПРАВЛЕНИЯ РАЗРАБОТКОЙ ПРОГРАММНЫХ ПРОЕКТОВ

Методические указания
по выполнению заданий производственной
(эксплуатационной) практики

Составители: **Волкова** Татьяна Викторовна,
Владимирова Елена Сергеевна

В авторской редакции

Изд. № 66/2023. Объем 1,75 п.л. Усл. печ. л. 1,63. Уч.-изд. л. 1,715.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»