

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ

по дисциплине

«Практикум по применению программного обеспечения»

для студентов направления подготовки

13.03.02 Электроэнергетика и электротехника

Направленность профиль "Цифровые технологии в
электроэнергетика"

Ставрополь, 2026

СОДЕРЖАНИЕ

Введение.....	3
Лабораторная работа №1. «Единая система конструкторской документации. Общие требования к текстовым документам».....	4
Лабораторная работа №2. Применение на практике возможностей современных текстовых редакторов для оформления текстовых документов, согласно ЕСКД	30
Лабораторная работа №3. Применение на практике возможностей текстовых редакторов для автоматизации оформления текстовых документов	51
Лабораторная работа №4. Первичное знакомство с векторными редакторами.....	67
Лабораторная работа №5. Применение на практике возможностей векторных редакторов для формирования электрических схем	96
Лабораторная работа №6. Первичное знакомство с языком программирования Python.....	124
Лабораторная работа №7. Python: списки, циклы, функции	144
Лабораторная работа №8. Работа с файлами, визуализация данных с помощью языка Python	165

Введение

Настоящее учебное пособие предназначено для проведения лабораторного практикума по дисциплине «Практикум по применению программного обеспечения» студентов, обучающихся по направлению 13.03.02 «Электроэнергетика и электротехника».

По современным требованиям инженер-электрик должен свободно применять компьютеры в своей производственной деятельности. В связи с этим цель освоения дисциплины заключается в формировании общепрофессиональных компетенций через приобретение студентами знаний об автоматизации выполнения конструкторско-проектной документации с помощью современных программных средств вычислительной техники.

Непрерывная подготовка студентов в области применения ЭВМ создает возможность широкого использования ЭВМ при изучении технологических дисциплин, выполнении курсовых и дипломных проектов, в научно-исследовательских работах.

Условно весь практикум можно разделить на три части. В первой части студенты знакомятся с единой системой конструкторской документации (ЕСКД), изучают возможности текстовых редакторов для создания документов согласно ГОСТ 2.105-95, приобретают навыки для автоматизации процесса оформления текстовых документов.

Во второй части, студенты изучают основные правила оформления чертежей согласно правилам ЕСКД, приобретают основные навыки работы с простейшим векторным редактором.

Третья часть посвящена знакомству с основами языка программирования Python, являющимся мощным современным инструментом анализа и визуализации данных.

Лабораторная работа №1. «Единая система конструкторской документации. Общие требования к текстовым документам»

Цель работы: Изучить назначение, структуру и основные понятия единой системы конструкторской документации (ЕСКД).

Теоретическая часть

1. Краткие сведения о единой системе конструкторской документации

Единая система конструкторской документации (ЕСКД) – комплекс государственных стандартов, устанавливающих взаимосвязанные правила, требования и нормы по разработке, оформлению и обращению конструкторской документации, разрабатываемой и применяемой на всех стадиях жизненного цикла изделия (при проектировании, разработке, изготовлении, контроле, приёмке, эксплуатации, ремонте, утилизации).

Единая система конструкторской документации (ЕСКД) была разработана в 1968 г. Основной организацией по разработке ЕСКД от Госстандарта является ВНИИНМАШ (с 01.01.1991 г. ВНИИН-МАШ был переименован во Всесоюзный научно-исследовательский институт экспертизы общесоюзных стандартов (ВНИИЭС)). В создании ЕСКД участвуют научно-исследовательские и конструкторские организации промышленности.

Необходимость разработки ЕСКД определялась постоянным увеличением потока конструкторской документации. В связи с тем, что в народном хозяйстве были недостаточно стандартизованы правила оформления конструкторской документации, а следовательно, невозможно было обеспечить единство разработки, оформления и обращения

документации, в отраслях промышленности начали создаваться ведомственные системы чертежного хозяйства.

Многообразие различных по форме, содержанию и оформлению документов являлось серьезным препятствием в деле создания сложной техники, особенно в условиях постоянно расширяющейся кооперации предприятий. Достаточно сказать, что для переоформления конструкторской документации на изделие средней сложности (порядка 1 000 оригинальных деталей) при передаче ее из конструкторского бюро на завод, при различии в системах чертежного хозяйства, требовалось 500 часов высококвалифицированного труда инженерно-технических работников.

В то же время, в конструкторском бюро средней мощности за год разрабатывается около 150 тыс. чертежей.

Целью разработки системы ЕСКД явились:

- обеспечение единства правил выполнения и оформления конструкторской документации, способствующее одинаковому ее пониманию на различных предприятиях и в организациях разных отраслей промышленности и, тем самым, обеспечение возможности организации производства изделий на одном предприятии по технической документации другого без дополнительной переработки;

- сокращение объема конструкторской документации и упрощение правил оформления текстовых документов и схем, а также устранение дублирующих документов;

- создание правил индексации и обозначения технической документации с целью сокращения времени ее поиска;

- обеспечение возможности привязки и использования системы в условиях применения автоматизированной системы управления и т.д.

В стандартах ЕСКД сохранилась, по возможности, преемственность правил и положений действовавшей ранее системы чертежного хозяйства, а также обеспечена согласованность правил оформления графических

документов (чертежей и схем) с рекомендациями международных организаций по стандартизации - ИСО и МЭК.

Единая система конструкторской документации - комплекс государственных стандартов, устанавливающих взаимосвязанные правила и положения по порядку разработки, оформления и обращения конструкторской документации, разрабатываемой и применяемой организациями и предприятиями.

Весь комплекс ЕСКД делится на 10 групп:

0 - общие положения;

1 - основные положения;

2 - классификация и кодирование изделий в конструкторских документах;

3 - общие правила выполнения чертежей;

4 - правила построения чертежей изделий машиностроения и приборостроения;

5 - правила обращения конструкторских документов (учет, хранение, дублирование, внесение изменений);

6 - правила выполнения эксплуатационной и ремонтной документации;

7 - правила выполнения схем;

8 - правила выполнения документов строительных и судостроения;

9 - прочие стандарты (резервная группа).

Стандартом ЕСКД 2.001-93 устанавливается область распространения нормативных материалов, которая включает в себя:

– все виды конструкторских документов;

– учетно-регистрационную документацию и документацию по внесению изменений в конструкторские документы;

– нормативно-техническую и технологическую документацию в той части, которая не регламентируется специальными стандартами;

– терминологию и основные положения по отработке конструкторских документов.

В соответствии с ГОСТ 2.101-68 дано понятие изделия, под которым понимается любой предмет или набор предметов производства, подлежащих изготовлению на предприятии.

ГОСТ 2.101-68 установил следующие виды изделий: детали; сборочные единицы; комплексы; комплекты.

Все изделия разделены на неспецифицированные (детали) - не имеющие составных частей и специфицированные (сборочные единицы, комплексы, комплекты) - состоящие из двух и более составных частей, причем составной частью может быть любое изделие - от детали до комплекта.

Каждое из этих специфицированных изделий, входящих в комплекс, служит для выполнения одной или нескольких основных функций, установленных для всего комплекса, например цех-автомат, буровая установка. В комплекс, кроме изделий, выполняющих основные функции, могут входить детали, сборочные единицы и комплекты, предназначенные для выполнения вспомогательных функций, например комплекс запасных частей.

Комплектом, в соответствии со стандартом, называют два и более изделия, не соединенных на предприятии-изготовителе сборочными операциями и представляющих набор изделий, имеющих общее эксплуатационное назначение вспомогательного характера, например комплект запасных частей или измерительной аппаратуры.

ГОСТ 2.102-68 устанавливает виды и комплектность конструкторских документов.

ГОСТ 2.103-68 четко определил стадии разработки конструкторской документации и содержание этапов работ на каждой из стадий.

Третья группа стандартов ЕСКД содержит общие правила выполнения чертежей и определяет форматы и их обозначения, масштабы, изображения, виды, разрезы и т.п., в том числе порядок нанесения на чертежах размеров и

предельных отклонений, отклонений формы, расположения поверхностей и шероховатости поверхностей и т.д.

Одно из основных назначений стандартов ЕСКД - ускорение и упрощение проектных конструкторских работ. На решение этой задачи направлены стандарты четвертой группы, определяющие правила выполнения чертежей основных машиностроительных деталей (пружин, зубчатых и червячных соединений, шлицевых и шпоночных соединений, подшипников и т.д.), и в том числе их упрощенных изображений, и седьмой группы, предусматривающие правила упрощения и условные обозначения при вычерчивании электрических, пневматических, гидравлических и т. п. схем и их элементов.

Шестая группа стандартов ЕСКД определяет номенклатуру эксплуатационных и ремонтных документов и правила их выполнения, предусмотрев, в частности, обязательные ремонтные документы (соответственно для среднего и капитального ремонта): руководство по среднему (капитальному) ремонту (литера РС или РК), нормы расхода запасных частей (ЗС или ЗК), нормы расхода материалов (МС, МК), ведомость документов для ремонта (ВР). Необходимость составления остальных ремонтных документов, а также эксплуатационных документов устанавливается разработчиками по согласованию с заказчиками.

Стандарты ЕСКД распространяются на изделия машиностроения и приборостроения. Область распространения отдельных стандартов расширена, что оговорено во введении к ним.

2. Требования к оформлению текстовых документов

Требования к оформлению текстовых документов изложены в следующих стандартах:

– ГОСТ 2.105–95 Общие требования к текстовым документам;

– ГОСТ 2.004-88: ЕСКД Общие требования к выполнению конструкторских и технологических документов на печатающих и графических устройствах вывода ЭВМ;

– ГОСТ 28388-89 Система обработки информации. Документы на магнитных носителях данных;

– ГОСТ 7.32-2001 Отчет о научно-исследовательской работе. Структура и правила оформления;

– ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

Ниже приведены основные положения из упомянутых выше стандартов.

2.1 Общие требования к оформлению текстового документа

Текст печатается на одной стороне листа белой бумаги формата А4. Основное требование к тексту – единообразие. Это касается шрифта, абзацных отступов, выравнивания текста, нумерации пунктов и частей и т.д.

Цвет шрифта – черный. Размер шрифта (кегель): для текста – 12, 14, для формул – 14, 16, для таблиц – 10, 12 или 14. Тип шрифта - Times New Roman. Междустрочный интервал – полуторный. Выравнивание текста – по всей ширине страницы.

Размеры полей: правое - 15 мм, верхнее и нижнее - 20 мм, левое - 30 мм.

Абзацный отступ не менее 1,25 см.

Страницы нумеруются арабскими цифрами (нумерация сквозная по всему тексту). Номер страницы ставится в нижнем правом углу или в центре нижней части листа без точки. Титульный лист включается в общую нумерацию, номер на нем не ставится.

Знаки препинания (точка, запятая, двоеточие, точка с запятой, многоточие, восклицательный и вопросительный знаки) от предшествующих слов не должны быть отбиты.

Следует отличать при наборе два знака: тире и дефис. Дефис – короткая черточка (-) используется, в основном, при наборе сложных слов, при переносах слов и от предшествующих и последующих слов не должен быть отбит (например: архитектурно-строительный).

Тире – длинная черточка (–). Это пунктуационный знак. Чаще всего тире отбивается справа и слева пробелом. Между числительными, выраженными числами, в значении «от и до» (10–15) тире используется без отбивки; между числительными, набранными прописью, тире отбивается пробелом (десять – пятнадцать).

Тире и дефис никогда не переносятся в следующую строку.

Кавычки и скобки применяются одного начертания по всему тексту, они не отбиваются от заключенных в них слов. Знаки препинания от них не должны быть отбиты (например: 1). «1».).

Знаки сноски – цифры или звездочки – в основном тексте должны быть отбиты от слова, к которому они относятся, но не от точки или запятой (например: См. текст *; материалы курса.*).

Знаки номера (№) и параграфа (§) применяют только с относящимися к ним числами и отбиваются от них пробелом (например: № 2, § 11).

Сокращенные выражения (и т.д., и т.п., т.е.) набираются без пробела.

Знак процента (%) применяется только с относящимся к нему числом, от которого отбивка не делается (например: 32%, 0,5%).

Знаки градуса (°), минуты ('), секунды (") и терции (""') от предыдущих чисел не должны быть отбиты (например: 8"12'; –25°). Если за этими знаками следует сокращенное обозначение шкалы, то оно должно быть отбито вместе со знаком градуса (25 °C).

Фамилии, названия организаций, фирм, названия изделий и другие имена собственные должны приводиться на языке оригинала. Допускается

приводить названия организаций в переводе на русский язык с добавлением (при первом упоминании) оригинального названия.

Все сокращения должны быть расшифрованы, за исключением небольшого числа общеупотребительных.

2.2 Оформление содержания

Заголовок СОДЕРЖАНИЕ пишется заглавными буквами посередине строки.

Содержание включает введение, наименование всех глав (разделов), параграфов (подразделов), пунктов (пунктов подразделов), заключение, список использованных источников и наименование приложений с указанием номеров страниц, с которых начинаются эти элементы.

Наименования, включенные в содержание, записывают строчными буквами, начиная с прописной буквы.

2.3 Оформление заголовков

Заголовки разделов располагают в середине строки без точки в конце и печатают прописными (большими) буквами (СОДЕРЖАНИЕ, ВВЕДЕНИЕ и т.д.) без подчеркивания. Названия заголовков допускается выделять жирным шрифтом.

Расстояние между заголовком и текстом должно быть равно 7-10 мм (12 пт).

Разделы нумеруют, и могут делить на параграфы, которые в свою очередь могут делиться на более мелкие разделы.

Номер параграфа состоит из номеров раздела и параграфа в разделе, разделенных точкой. В конце номера точка не ставится. Аналогичным образом нумеруются и пункты в параграфе (например: 2.4.2 Анализ результатов).

Заголовки параграфов, пунктов и подпунктов следует печатать с абзацного отступа с прописной буквы без точки в конце, не подчеркивая. Если заголовок состоит из двух предложений, их разделяют точкой. Переносы слов в заголовках и подзаголовках не допускаются. Если заголовок состоит из двух самостоятельных предложений, между ними ставят точку, а в конце точку опускают. Если такой заголовок не уместится в одну строку, его разбивают так, чтобы точка попадала внутрь строки, а не заканчивала ее. Заголовки и подзаголовки не следует подчеркивать, а также выделять другим цветом. Не разрешается оставлять заголовок (подзаголовок) в нижней части страницы, помещая текст на следующей.

Каждый раздел текстового документа рекомендуется начинать с нового листа (страницы).

2.4 Правила переноса

Не допускаются неблагозвучные переносы, искажающие смысл.

Не допускается перенос сокращений, набираемых прописными буквами (ГОСТ), прописными с отдельными строчными (КЗоТ) и с цифрами (ФА 1000).

Не должны быть разделены переносом числа (исключение составляют те, что соединены знаком тире 1985–1990).

Не допускается размещение в разных строках чисел и их наименований (250 кг), знаков номера и параграфа и относящихся к ним чисел (№ 25).

Сокращенные выражения при переносе не разделяются (т.е., т.д.)

При переносе не должны быть оторваны инициалы от фамилий и инициалы друг от друга (А.А. Иванов).

2.5 Оформление перечисления

Перед каждым перечислением следует ставить дефис или, при необходимости ссылки в тексте на одно из перечислений, строчную букву (за исключением ё, з, й, о, ч, ь, ы, ъ).

Для дальнейшей детализации перечислений необходимо использовать арабские цифры, после которых ставится скобка.

а) _____

б) _____

1) _____

2) _____

в) _____

2.6 Оформление таблиц

Таблицы набираются единообразно по всему тексту. На все таблицы в тексте должны быть ссылки. Таблица должна располагаться непосредственно после текста, в котором она упоминается впервые, или на следующей странице. От текста таблица отбивается сверху и снизу пустой строкой.

Допускается помещать таблицу вдоль длинной стороны листа документа.

Все таблицы нумеруются. Нумерация - сквозная, либо в пределах раздела. В последнем случае номер таблицы состоит из номера раздела и порядкового номера внутри раздела, разделенных точкой (например: Таблица 1.2).

Название таблицы помещают над таблицей слева, без абзацного отступа в одну строку с ее номером через тире. Точка в конце названия не ставится.

Пример:

Таблица 3 - Коэффициенты отражения и пропускания световой волны тонкой металлической пленкой

Металл	Nk	n	ТЕ - волна		ТМ – волна	
			Толщина в λ (при R=T)	Фаза в π (при R=T)	Толщина в λ (при R=T)	Фаза в π (при R=T)
Свинец	3.48	2.01	0.013	0.18	0.027	0.47
Платина	4.26	2.06	0.010	0.19	0.018	0.22

Выше и ниже таблицы оставляют интервал 12 пт. Таблица должна быть обязательно выровнена по центру страницы. Интервал между строк одинарный.

При переносе таблицы на следующую страницу название помещают только над первой частью, при этом нижнюю горизонтальную черту, ограничивающую первую часть таблицы, не проводят. Над другими частями таблицы слева пишут слово «Продолжение» и указывают номер таблицы (например: Продолжение таблицы 1).

Таблицу с большим количеством столбцов допускается делить на части и помещать одну часть под другой в пределах одной страницы. Если строки и столбцы таблицы выходят за формат страницы, то в первом случае в каждой части таблицы повторяется головка, во втором случае - боковик. При делении таблицы на части допускается ее головку или боковик заменять соответственно номером столбцов и строк. При этом нумеруют арабскими цифрами столбцы и (или) строки первой части таблицы.

Заголовки столбцов и строк таблицы следует писать с прописной буквы в единственном числе, а подзаголовки столбцов - со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков столбцов и строк точки не ставят. Разделять заголовки и подзаголовки боковых столбцов диагональными линиями не допускается.

Заголовки столбцов, как правило, записывают параллельно строкам таблицы, но при необходимости допускается их перпендикулярное расположение.

Горизонтальные и вертикальные линии, разграничивающие строки таблицы, допускается не проводить, если их отсутствие не затрудняет пользование таблицей. Но головка таблицы должна быть отделена линией от остальной части таблицы.

Если все показатели, приведенные в графах таблицы, выражены в одной и той же единице физической величины, то ее обозначение необходимо помещать над таблицей справа.

Если в большинстве граф таблицы приведены показатели, выраженные в одних и тех же единицах физических величин (например, в миллиметрах, вольтах), но имеются графы с показателями, выраженными в других единицах физических величин, то над таблицей следует писать наименование преобладающего показателя и обозначение его физической величины, например, «Размеры в миллиметрах», «Напряжение в вольтах», а в подзаголовках остальных граф приводить наименование показателей и (или) обозначения других единиц физических величин.

Для сокращения текста заголовков и подзаголовков граф отдельные понятия заменяют буквенными обозначениями, если они пояснены в тексте или приведены на иллюстрациях, например D – диаметр, H – высота, L – длина.

Показатели с одним и тем же буквенным обозначением группируют последовательно в порядке возрастания индексов.

Ограничительные слова «более», «не более», «менее», «не менее» и т.п. должны быть помещены в одной строке или графе таблицы с наименованием соответствующего показателя после обозначения его единицы физической величины, если они относятся ко всей строке или графе.

Если в графе таблицы помещены значения одной и той же физической величины, то обозначение единицы физической величины указывают в заголовке (подзаголовке) этой графы.

Если числовые значения величин в графах таблицы выражены в разных единицах физической величины, их обозначения указывают в подзаголовке каждой графы.

Обозначения, приведенные в заголовках граф таблицы, должны быть пояснены в тексте или графическом материале документа.

2.7 Оформление чисел

Числа должны быть отбиты от относящихся к ним знаков и наименований на один пробел (25 000 т). Числа с буквами в обозначениях не отбиваются (3а, 15в). Цифры и буквы, разделенные точками, набирают без отбивки (1.3.14а).

Числа делят пробелами на группы (по три цифры) справа налево, начиная с 5-значных (35 786, 52 632, 8 201 793).

Не разбиваются цифры, обозначающие номер в марке машин и механизмов, в нормативных документах (например: ГОСТ 16598-70).

Для обозначения крупных чисел (тысяч, миллионов, миллиардов) употребляются сочетания цифр с сокращением (20 млн, 12 млрд, 35 тыс.). Это распространяется и на сочетания крупных чисел с обозначением единиц физических величин, денежных единиц и т.п. (20 млн км; 200 млрд кВт ч).

В сокращениях млн, млрд точка не ставится. Точка ставится в сокращении тысяча (тыс.).

Дробные числа делятся пробелами на группы по три, но после запятой в обратном направлении по сравнению с целыми, т.е. слева направо (например: 25,123 137; 9,465 4 м).

Существительное после дробного числа согласуется с его дробной частью и ставится в род. падеже ед. ч. (например: 0,56 литра, 10⁴ тонны).

Простые дроби от целой части не отбиваются.

Употребляют многоточие, тире, предлог *от* перед первым числом и *до* перед вторым (например: 5...10 м; 5–10 м; длиной от 5 до 10 м).

При цифровой форме крупных чисел необходимо сохранять нули в числе нижнего предела (высота 15 000–20 000 м; длина 6 000 000–8 000 000 км).

При буквенно-цифровой форме чисел допустимо опускать в числе нижнего предела обозначение тыс., млн, млрд (например: высота 20–30 тыс. м).

Исключение составляют взаимосвязанные относительные числа (например: Это составляет 60–80% всей массы груза. Остальные – 40–20%).

При написании порядковых числительных используют следующие падежные окончания:

а) однобуквенное, если последней букве числительного предшествует гласный: 5-й (пятый, пятой), 5-я (пятая); 5-е; 5-м; 5-х.

б) двухбуквенным, если последней букве числительного предшествует согласный: 5-го (пятого); 5-му; 5-ми.

Принято обозначать арабскими цифрами без наращивания падежных окончаний номера томов, глав, страниц, иллюстраций, таблиц, приложений и т.п., если родовое слово (том, глава и т.д.) предшествует им (например: в томе 6, но в 6-м томе; в главе 5, но в 5-й главе).

Римскими цифрами без наращивания падежных окончаний набирают:

а) номера съездов, конференций, конгрессов и т.д. (XXVI съезд);

б) обозначения веков (XX век);

в) номера международных объединений (III Интернационал);

г) номера выборных органов (VI Государственная дума);

д) обозначение кварталов года (III–IV кварталы);

е) части или разделы книг и т.п.

В научной и деловой литературе при использовании сложных слов с числительным и прилагательным *процентный* принята форма из числительного в цифровой форме, дефиса и падежного окончания -ный, -ного, -ному и т.д. (10%-ный раствор). Может использоваться форма с наращиванием одно- и двухбуквенного окончания (15%-й раствор; 20%-го раствора и т.д.).

2.8 Оформление формул и физических элементов

Набор математических формул должен быть по всему тексту единообразным: по шрифту, по способу расположения (в красную строку, в левый край, с заданным отступом от левого края), по применению индексов и т.д.

Группы однотипных формул выравниваются вертикально по основному знаку математических соотношений (обычно, по знаку равенства (=)).

$$5 + 5 = 10$$

$$15 + 1 = 26$$

Математические знаки действий (+, −, × и др.) и соотношений (= и др.) должны быть отбиты от смежных символов или чисел пробелами.

Однако математические знаки перед числами в значении степени увеличения, положительной или отрицательной величины от чисел не отбиваются (+5, −20).

Математические сокращения отбиваются от смежных элементов формул ($2 \sin a$, $\operatorname{tg} a$). Если сокращения имеют показатель степени, то отбивка относится за показатель ($2 \sin^2 a$).

Символы, обозначающие функции (f), приращения (Δ), дифференциалы (D) и др., от следующих за ними обозначений функций или аргументов, в том числе и заключенных в скобки, не отбиваются ($f(x)$, $2dx dy$).

Сокращенные обозначения физических величин и единиц измерения должны быть отбиты от символов и цифр, к которым они относятся (15 кг, 22 Дж).

Индексы и показатели степени должны быть по размеру заметно меньше символов и чисел в формуле (10^2 , $K_{пр}$), от них не отбиваются.

Формулы следует выделять из текста в отдельную строку. Над и под каждой формулой или уравнением нужно оставить по пустой строке. Если формула не умещается в одну строку, то оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (-), умножения (х), деления (:), или других математических знаков, причем этот знак в начале следующей строки повторяют. При переносе формулы на знаке, символизирующем операцию умножения, применяют знак «х».

Если нужны пояснения к символам и коэффициентам, то они приводятся сразу под формулой в той же последовательности, в которой они идут в формуле. Первая строка пояснения должна начинаться со слова «где» без двоеточия после него.

Пример:

$$I=U/R,$$

где U — напряжение на участке цепи, В;

R — сопротивление участка цепи, Ом.

Все формулы нумеруются. Обычно нумерация сквозная. Номер проставляется арабскими цифрами в круглых скобках в крайнем правом положении на строке.

$$A = a:b \tag{1}$$

Допускается нумерация формул в пределах раздела. В этом случае номер формулы состоит из номера раздела и порядкового номера внутри раздела, разделенных точкой (например: (1.4)).

При ссылке на какую-либо формулу ее номер ставят точно в той же графической форме, что и после формулы (например: в формуле (1.1); уравнение (3.2)).

Формула включается в предложение как его равноправный элемент, поэтому в конце формул и в тексте перед ними знаки препинания ставят в соответствии с правилами пунктуации. Формулы, следующие одна за другой и не разделенные текстом, отделяют запятой или точкой с запятой непосредственно за формулами до их номера:

$$5 + 5 = 10; 2 + 2 = 4; \text{ либо } 5 + 5 = 10, 2 + 2 = 4$$

В формулах в первую очередь используются круглые скобки, во вторую – квадратные, в третью – фигурные: (), [], { }. Они не отбиваются от элементов формулы: $(1 + 3)$. Если за скобкой следует индекс, то он набирается без отбивки от скобки, а отбивка переносится за индекс (например: $(a + c)^2$).

Если открывающиеся или закрывающиеся скобки следуют одна за другой, то их разделяют пробелом (например: (тире (черточка)).

Точка может использоваться как основной знак умножения:

а) перед числовым сомножителем: $35 \quad 0,18 \cdot 5,2; a \cdot 5;$

б) для выделения какого-либо множителя: $2 \cdot 3 \quad x y \cdot z;$

в) для записи скалярного произведения векторов: $a \cdot b;$

г) между аргументом тригонометрической функции и буквенным обозначением: $a \cdot \sin x \cdot b \cos y$ и в некоторых др. случаях.

Знак не ставят:

а) перед буквенным символом: $3ac, ac;$

б) перед скобками и после них: $4(a + b)(a + c)$ и в некоторых др. случаях.

Знак деления в виде косой черты (/) набирается в формулах, набранных без выделения формулы отдельной строкой (например: ...в уравнении $3/5$...). Если формула набрана отдельной строкой, знак деления следует обозначать горизонтальной чертой (—), проходящей посередине

строки (в дроби). Все математические знаки и знаки препинания равняются по этой строке.

Косой крест (x) употребляют как знак умножения

а) при указании размеров: площадь 4,5x3 м (без отбивки);

б) для записи векторного произведения: $a \times c$ (с отбивкой);

в) при переносе формулы с одной строки на другую при знаке умножения.

2.9 Оформление иллюстраций

На все рисунки в тексте должны быть даны ссылки. Рисунки должны располагаться непосредственно после текста, в котором они упоминаются впервые, или на следующей странице. Рисунки нумеруются арабскими цифрами, при этом нумерация сквозная, но допускается нумеровать и в пределах раздела (главы). В последнем случае, номер рисунка состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой (например: Рисунок 1.1).

Подпись к рисунку располагается под ним посередине строки. Слово «Рисунок» пишется полностью. В этом случае подпись должна выглядеть так: Рисунок 2 - Структура фирмы

Точка в конце названия не ставится.

При ссылках на иллюстрации следует писать «... в соответствии с рисунком 2» при сквозной нумерации и «... в соответствии с рисунком 1.2» при нумерации в пределах раздела.

2.10 Оформление сноски

Если необходимо пояснить отдельные данные, приведенные в документе, то эти данные следует обозначать надстрочными знаками сноски.

Сноски в тексте располагают с абзацного отступа в конце страницы, на которой они обозначены, и отделяют от текста короткой тонкой горизонтальной линией с левой стороны.

Знак сноски ставят непосредственно после того слова, числа, символа, предложения, к которому дается пояснение, и перед текстом пояснения.

Знак сноски выполняют арабскими цифрами со скобкой и помещают на уровне верхнего обреза шрифта.

Пример: «... печатающее устройство ¹⁾ ...». Нумерация сносок отдельная для каждой страницы.

Допускается вместо цифр выполнять сноски звездочками: *. Применять более четырех звездочек не рекомендуется.

2.11 Оформление ссылки

Библиографические ссылки употребляют:

- при цитировании;
- при заимствовании положений, формул, таблиц, иллюстраций;
- при необходимости отсылки к другому изданию, где более полно изложен вопрос;
- при анализе в тексте опубликованных работ.

Есть два вида библиографических ссылок: внутритекстовые и подстрочные.

Внутритекстовые ссылки размещаются непосредственно в строке после текста, к которому относятся. Оформляются в квадратных скобках с указанием номера в списке литературы, например, [31]. Ссылки на несколько конкретных работ автора могут быть даны, например, в форме [12-17, 19].

При цитировании, а также в случаях, требующих указания конкретной страницы источника, в скобках дополнительно указывается страница [12. С. 7] или страницы "от - до" [19. С. 7 – 9].

Пример:

Когда частица пролетает вблизи ядра, на неё действует кулоновская сила отталкивания [14, С.51].

Подстрочные ссылки размещаются внизу страницы, под строками основного текста, имеют сквозную нумерацию по всему документу.

Пример:

Литературовед Левидов А.М. писал о том, что "не количество прочитанных книг служат показателем культуры чтения, а качество их потребления".¹

¹ Левидов А.М. Литература и действительность / А.М. Левидов. - М.: Академия, 1987. - С. 409.

2.12 Оформление списка литературы

Список литературы должен называться «Список использованных источников».

Сведения об источниках должны включать: фамилию, инициалы автора, название источника, ответственность, место издания, издательство, год издания, количество страниц.

Фамилию автора указывают в именительном падеже. Наименование места издания необходимо приводить полностью в именительном падеже, допускается сокращенное название двух городов: Москва (М.), Санкт-Петербург (СПб).

Для статей указываются фамилия и инициалы автора, название статьи, название журнала, год издания, номер журнала, номер страницы.

Государственного стандарта по оформлению списка литературы нет, но существует общепринятая практика. Чаще всего используют алфавитное расположение источников или расположение по типам документов. Алфавитное расположение источников означает, что выдерживается строгий алфавит заголовков библиографического описания (авторов или заглавий).

При расположении по типам документов материал в списке литературы располагается сначала по типам изданий: книги, статьи, официальные документы, стандарты и т. д., а внутри раздела - по алфавиту (автор или заглавие).

В этом случае порядок расположения источников может быть следующим:

- нормативные акты;
- книги;
- печатная периодика;
- источники на электронных носителях локального доступа;
- источники на электронных носителях удаленного доступа (т.е. интернет-источники).

В каждом разделе сначала идут источники на русском языке, а потом - на иностранных языках (так же в алфавитном порядке).

Нормативные акты располагаются в следующем порядке:

- международные акты, ратифицированные Россией, причем сначала идут документы ООН;
- Конституция России;
- кодексы;
- федеральные законы;
- указы Президента России;
- постановления Правительства России;
- приказы, письма и пр. указания отдельных федеральных министерств и ведомств;
- законы субъектов России;
- распоряжения губернаторов;
- распоряжения областных (республиканских) правительств;
- судебная практика (т.е. постановления Верховного и прочих судов России);
- законодательные акты, утратившие силу.

Федеральные законы следует записывать в формате: Федеральный закон от [дата] № [номер] «[название]» // [официальный источник публикации, год, номер, статья]

Законы располагаются не по алфавиту, а по дате принятия (подписания Президентом России) - впереди более старые.

Если при написании работы использовался законодательный сборник или издание отдельного закона, в список литературы все равно следует записать закон (приказ и т.п.) с указанием официального источника публикации. Для федеральных актов такими источниками являются: «Собрание законодательства Российской Федерации», «Российская газета», «Собрание актов Президента и Правительства Российской Федерации» и др.

Под электронными ресурсами подразумеваются как данные из Интернета, так и данные на CD, дискетах и т.п. Все такого рода данные считаются опубликованными.

Примеры оформления источников:

Справочные издания

1. Большая Российская энциклопедия. В 30 т. Т. 4. Большой Кавказ – Великий канал / отв. ред. С.Л. Кравец. – М.: Науч. изд-во «Большая Российская энциклопедия», 2006. – 751 с.

2. Сборник материалов открытой научно-практической конференции «Современные проблемы менеджмента» / ТюмГАСУ. – Тюмень: Печатник, 2005. – 395 с.

Один автор

3. Крысько В.Г. Социальная психология: учебник / В.Г. Крысько // 2-е изд. – СПб.: Питер, 2007. – 432 с.

Два и более авторов

4. Никитина Е.С. Семиотика. Курс лекций: учеб. пособие / Е.С. Никитина, Н.Н. Алексеева. – М.: Академический проект; Трикста, 2006. – 528 с.

Статья в периодическом издании

5. Егорова П.Д., Минтусов И.Л. Портрет делового человека // Проблемы теории и практики управления. — 1992. — № 6. — С. 3—17.

Статья из энциклопедии и словаря

6. Бирюков Б.В. Моделирование // БСЭ. — 3-е изд. — М., 1974. — Т. 16. — С. 393 —395.

Стандарт

7. ГОСТ Р 52652-2006. Информационно-коммуникационные технологии в образовании. - Введ. 2006-12-27. - М.: Стандартинформ, 2007. - 3 с.

Газета

8. Учительская газета: независимое педагогич. изд. - М., 1924 - Выходит еженедельно.

Ресурсы локального доступа

9. 5555 шедевров мировой живописи [Электронный ресурс]. - М.: ООО "ДиректМедиа Паблишинг", 2003. - 1 эл. опт. диск (CD-ROM): цв. - (Электронная библиотека).

Ресурсы удаленного доступа

10. А.А.Говоров, Т.Г.Куприянова; Московский гос. ун-т печати, Лаборатория компьютеризации ФИДиКТ. - Электрон. текстовые дан. и граф. дан. - М.: Издательство МГУП "Мир книги", 1998. — 348 с.: цв. - Режим доступа: <http://www.hi-edu.ru/e-books/HB/>, свободный. - Электрон. версия печ. публикации.

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с

диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Откройте архив «ЕСКД.zip», изучите структуру и основные документы, входящие в единую систему конструкторской документации.
2. Сформируйте краткий конспект о назначении и структуре ЕСКД.
3. Подробно изучите ГОСТ 2.105–95 «ОБЩИЕ ТРЕБОВАНИЯ К ТЕКСТОВЫМ ДОКУМЕНТАМ».

4. Выпишите основные требования к оформлению текстовых документов.

5. Сформируйте краткую памятку по оформлению текстовых документов в виде таблицы:

Таблица 1 – Форма таблицы

Показатель (параметр)	Значение по стандарту ЕСКД
...	...

При составлении опираться на вопросы:

- Оформление полей в текстовых документах, отступы;
- Разделение текста на разделы и подразделы;
- Правильное оформление заголовков;
- Запись и оформление формул, таблиц и рисунков в документе;
- Нумерация формул, таблиц и рисунков;
- Система сокращений и допущений;
- др.

Например:

Таблица 2 – Пример заполнения

Показатель (параметр)	Значение по стандарту ЕСКД
Стиль шрифта	Times New Roman
Размер шрифта	12,14
...	...

6. Найдите ошибки оформления в выданном преподавателем текстовом документе.

7. Ответьте на поставленные контрольные вопросы.

Содержание отчета

Отчет должен содержать:

- краткий конспект о назначении и структуре ЕСКД;
- основные требования к оформлению текстовых документов
- памятку по оформлению текстовых документов в виде таблицы (таблица 1);
- список ошибок оформления в выданном преподавателем текстовом документе.

Контрольные вопросы:

1. С какой целью была введена ЕСКД?
2. Какие функции выполняет ЕСКД?
3. На какие группы делится комплекс ЕСКД?
4. Какова область распространения стандарта ЕСКД?
5. Какие основные правила оформления текстовых документов?

Лабораторная работа №2. Применение на практике возможностей современных текстовых редакторов для оформления текстовых документов, согласно ЕСКД

Цель работы: Научиться применять возможности современных текстовых редакторов для создания текстовых документов согласно ГОСТ 2.105-95 ЕСКД.

Теоретическая часть

Текстовый процессор Word 2010 входит в состав семейства программных продуктов Microsoft Office 2010 и предназначен для работы под управлением операционной системы Windows. Word 2010 один из лучших текстовых редакторов современности позволяющий создавать множество различных видов документов: служебные записки, отчеты, информационные листки, брошюры, проводить вычисления в таблицах, создавать и редактировать рисунки, использовать фигурный текст, настраиваемые шаблоны для типовых документов и многое другое.

Word 2010 обладает универсальными средствами, позволяющими оформлять документацию в соответствии с требованиями ЕСКД.

Рассмотрим основные из них.

1 Настройка полей

Поля документа – это расстояние между краями страницы и текстом. Соответственно, чем больше поле, тем меньше текста помещается на странице, и наоборот. В любом документе есть правое, левое, верхнее и нижнее поля.

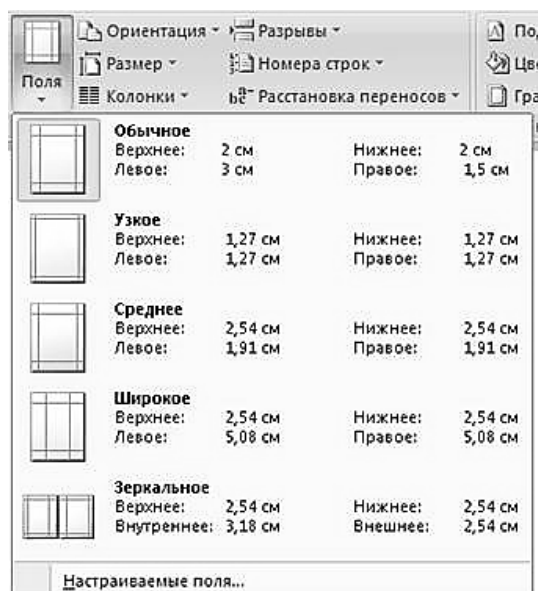


Рисунок 2.1 – Меню «Поля»

На полях могут располагаться колонтитулы, примечания, графические объекты. Обычно размер полей устанавливают в пределах от 1 до 3 см, широкие поля могут достигать и 5 см. В меню Поля, которое появляется после нажатия на одноименную кнопку, можно выбрать один из наиболее распространенных вариантов полей документа (рисунок 2.1). Если ни один из этих вариантов вам не подходит, выберите команду «Настраиваемые поля» и задайте размеры полей вручную. Если задать требуемые размеры полей в области «Поля», то можно посмотреть, как будет выглядеть документ с установленными вами параметрами на образце (рисунок 2.2).

Поля страниц (так же, как отступы и выступы) можно быстро задавать с помощью горизонтальной (для правого и левого полей) и вертикальной (для верхнего и нижнего полей) линеек. Размер поля показывают деления на серой части линейки (рисунок 2.3).

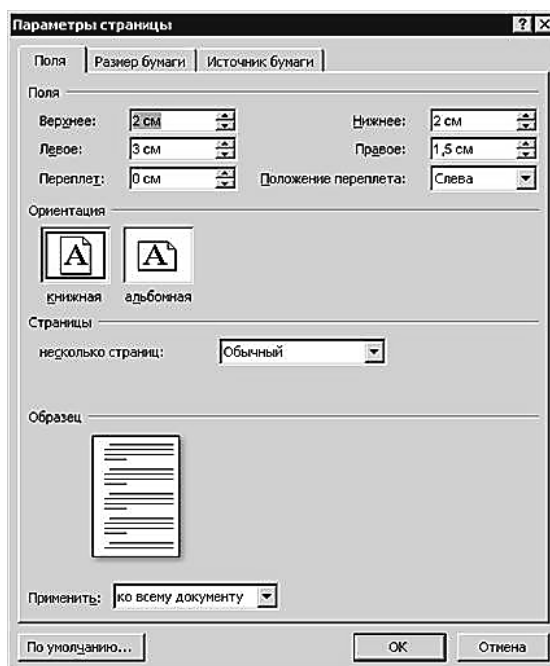


Рисунок 2.2 – Диалоговое окно «Параметры страницы», вкладка «Поля»



Рисунок 2.3 – Отображение размеров полей на горизонтальной и вертикальной линейках

2. Настройки шрифтов

Чтобы изменить шрифт по умолчанию в настройках, необходимо на вкладке «Главная» в разделе «Шрифт» нажать на маленькую стрелку (рисунок 2.4) в правом углу.

В диалоговом окне «Шрифт» выберите тот шрифт, который хотите использовать в качестве шрифта по умолчанию. Также можете выбрать его начертание и размер.

В данном случае мы выбрали «Times New Roman – Обычный – 12». После того как сделали свой выбор, жмем кнопку в левом углу – «По умолчанию». Будет предложено подтвердить это действие.

В Word 2007 просто соглашаемся. А в Word 2010, будет предоставлена возможность задать эти настройки либо только для этого документа, либо для всех документов. Выбираем «Все документы основанные на нормальном

шаблоне .dotm» и жмем ОК.

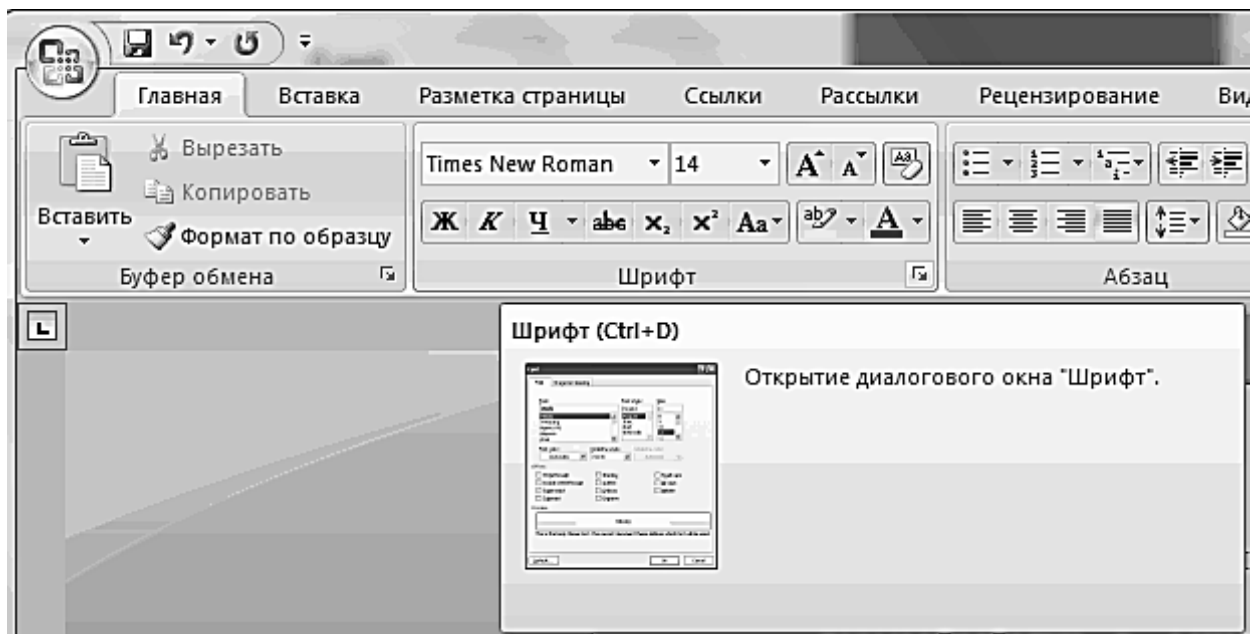


Рисунок 2.4 – Раздел «Шрифт»

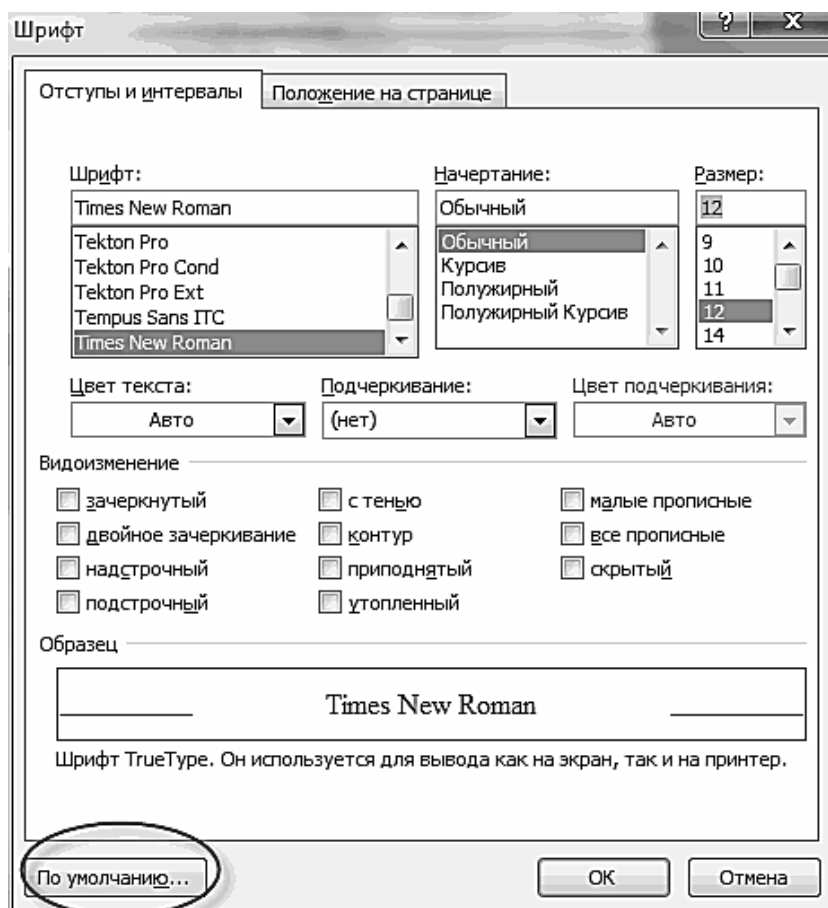


Рисунок 2.5 – Окно настроек шрифта

Теперь, в следующий раз, когда вы создадите новый документ,

шрифтом *по умолчанию*, будет выбранный вами шрифт.

3. Режим отображения непечатаемых знаков

Функция отображения непечатаемых символов важная часть программы MS Word при работе со сложно форматированным текстом.

Эти символы отображают места нажатий таких клавиш как пробел, Tab, Enter и так далее. Естественно они и получили название непечатаемые, так как при печати на бумаге они не отображаются.

Чтобы включить или отключить эту функцию необходимо на панели инструментов программы Word на вкладке «Главная» в поле «Абзац» нажать на инструмент (иконку) указанный на рисунке 2.6.

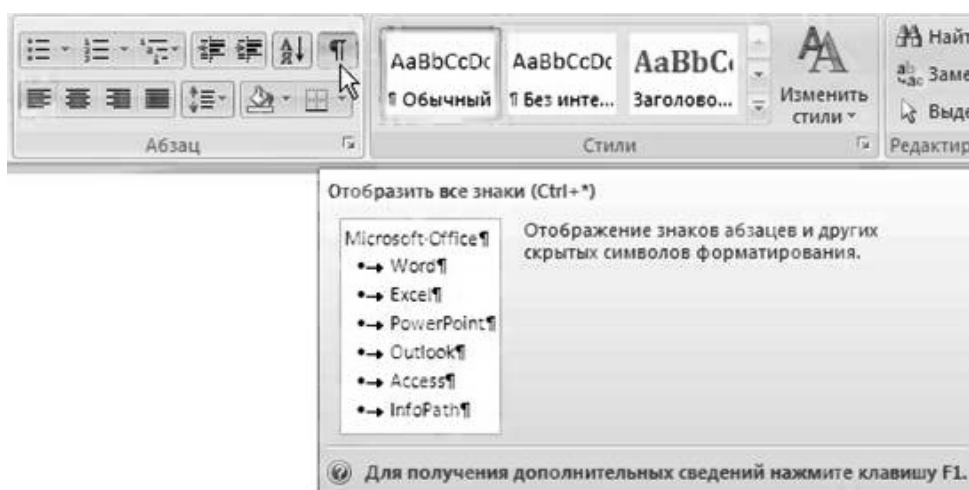


Рисунок 2.6 – Отображение непечатаемых знаков

После чего, если набран текст, сразу появятся точки между словами, знаки абзаца и прочие символы в тексте, которые ранее не отображались.

Режим отображения данных знаков необходим для удобства более правильного форматирования текста, так как создание сложных по структуре текстов с использованием различных элементов требует довольно сложного форматирования текста, таблиц, что невозможно без отображения непечатаемых символов.

При оформлении текстовых документов согласно стандартам ЕСКД рекомендуется всегда начинать работать с включенным отображением

непечатаемых знаков. Список наиболее часто встречаемых непечатаемых символов в программе MS Word представлен в таблице 2.1.

Таблица 2.1 – Список некоторых непечатаемых символов в MS Word

Непечатаемый символ	Название	Какое сочетание клавиш нажать для ввода.
·	пробел (точка между словами)	Нажать пробел
¶	Конец абзаца	Нажать Enter
↵	Разрыв строки	Сочетание Shift+Enter
□	Ячейка таблицы	Автоматически появляется в каждой ячейке
¬	Мягкий перенос	Сочетание клавиш CTRL+ДЕФИС.
°	Неразрывный пробел	CTRL+ Shift+ПРОБЕЛ

Также часто встречаются разрыв страницы, колонки, раздела

4. Режим отображения границ текста

Помимо режима отображения непечатаемых знаков, может быть полезен режим отображения границ текста, который позволяет отобразить пунктирную рамку вокруг доступной для размещения текста области лист, с учетом заданных полей документа. Для включения этого режима необходимо выполнить команду меню «Файл → Параметры → Дополнительно → Показывать содержимое документа → Показывать границы текста».

5. Режимы просмотра документа

Текстовый редактор MS Word имеет 5 режимов просмотра документов, которые можно поменять на вкладке «Вид» (рисунок 2.7).

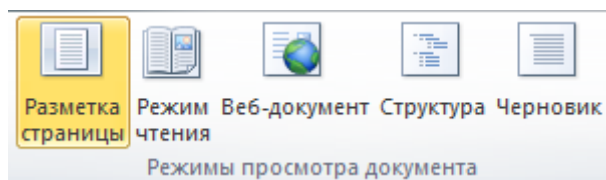


Рисунок 2.7 – Режимы просмотра документа

Режим «Разметка страницы» – это способ отображения рабочей области документа, при котором представлены измерительные линейки, а

сам документ разбит на страницы разделенные полями. Это наиболее универсальный режим для работы – доступны все инструменты форматирования, и можно добавлять в документ таблицы и различные графические объекты.

«Режим чтения» предназначен для максимального удобства чтения документа. Присутствие кнопок команд минимизировано, а рабочая область занимает максимум пространства.

Режим «Веб-документа» необходим для работы с документами, предназначенными для публикации во всемирной сети интернет.

Режим «Структура» позволяет выделять в документе структурные элементы (заголовки разделов, подразделов, параграфов, таблиц и рисунков) и применять к ним единые правила оформления. Кроме того, применение этого режима в процессе работы над документом позволяет автоматизировать некоторые рутинные операции, такие как автоматическая генерация оглавления или содержания. Более подробно этот режим будет рассмотрен в следующей лабораторной работе.

Режим «Черновик» предназначен для набора текста, без использования форматирования и применения различных объектов (рисунков, таблиц). Здесь отсутствует вертикальная линейка, границы и поля документа. Разрыв страницы показан точечной горизонтальной линией. Минимализм данного режима позволяет лучше сосредоточиться на наборе текста, и к тому же является наиболее экономным для ресурсов компьютера, что особенно актуально для бюджетных ПК.

Как можно заметить из представленного выше описания, при создании и оформлении текстовых документов согласно стандартам ЕСКД интерес представляют только два из пяти режимов просмотра редактора MS Word.

6. Форматирование абзацев

Абзац – это структурная составляющая текста, обычно связанная логически и выделенная графически, например, в виде отступов. Когда вы

набираете текст, каждый раз нажимая клавишу «Enter», вы создаете новый абзац. Настройка вида абзаца определяет его формат, который отвечает за размещение текста на странице.

Настройка (форматирование) абзаца, подразумевает:

- расстояние справа или слева от абзаца;
- междустрочный интервал абзаца;
- выравнивание строк абзаца относительно левого и правого поля;
- отступ или выступ первой строки абзаца;
- интервал перед абзацем и после него;
- разбивка абзаца на страницы.

В первую очередь, приступая к настройке абзаца, необходимо уметь его выделять. Для этого сделайте следующее:

1 способ:

Трижды щелкните левой кнопкой мыши по любому слову абзаца;

2 способ:

Используйте комбинации клавиш:

<Ctrl+Shift+(стрелка вверх)> – выделение от места расположения курсора ввода текста до начала абзаца;

<Ctrl+Shift+(стрелка вниз)> – выделение от места расположения курсора ввода текста до конца абзаца.

6.1 Выравнивание текста

По умолчанию текст абзаца выравнивается по левому краю. Однако стандарты ЕСКД требуют выравнивание по всей ширине страницы.

Для того чтобы задать нужный способ выравнивания строк абзаца, воспользуйтесь одним из нижеперечисленных способов:

1 способ:

Перейдите на вкладку «Главная», а затем выделите необходимые абзацы (очередность не имеет значения).

В группе «Абзац» щелкните одну из четырех доступных кнопок

выравнивания:

- «По левому краю» – выравнивание строк абзаца по левому краю;
- «По центру» – выравнивание строк абзаца по центру (пропорционально равноудалены от правого и левого края);
- «По правому краю» – выравнивание строк абзаца по правому краю;
- «По ширине» – выравнивание строк абзаца одновременно и по левому и по правому полю. При этом текст визуально «растянется» (заполнит пространство строк), а между словами появится дополнительное расстояние.

2 способ:

Выделите нужные абзацы, а затем используйте сочетания «быстрых» клавиш:

- <Ctrl+L> – выравнивание по левому краю;
- <Ctrl+E> – выравнивание по центру;
- <Ctrl+R> – выравнивание по правому краю;
- <Ctrl+J> – выравнивание по ширине.

6.2 Отступ первой строки абзаца

Наиболее часто применяется при форматировании абзацев установка отступа первой строки. Настроить отступы можно визуально (на глаз) или точно (задав определенное значение в окне «Абзац»). Визуальная настройка предполагает использование горизонтальной линейки, когда вы можете быстро менять размер отступа, руководствуясь ощущением нужного размера «на глаз». Точная настройка позволяет задать значение отступа в единицах измерения, используемых в документе – стоит использовать ее тогда, когда вы знаете размер величины отступа.

6.2.1 Визуальная настройка отступа первой строки

Чтобы задать отступ первой строки абзаца, сделайте следующее:

1 способ:

- выделите нужные абзацы;

– на горизонтальной линейке подведите курсор к значку «Отступ первой строки»;

– щелкните и удерживайте левую кнопку мыши, а затем перетащите значок по линейке до нужного места.

2 способ:

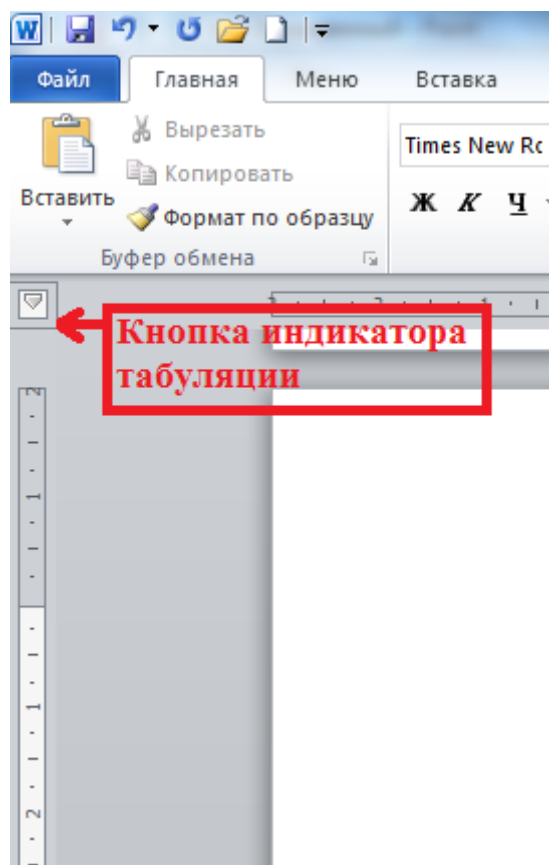


Рисунок 2.8 – Индикатор табуляции

– щелкайте по кнопке индикатора табуляции до появления значка «Отступ первой строки» (рисунок 2.8);

– щелкните по горизонтальной линейке в том месте, где должен находиться отступ первой строки.

6.2.2 Точная настройка отступа первой строки

1 способ:

– выделите нужные абзацы;

– щелкните правой кнопкой мыши по выделенному фрагменту;

– в контекстном меню выберите пункт «Абзац» (вы перейдете в окно

«Абзац»);

– на вкладке «Отступы и интервалы» в группе «Отступ» в графе «Первая строка» задайте нужное значение отступа (по умолчанию 1,25 см).

2 способ:

В окне открытого документа перейдите к вкладке «Главная»;

Выделите нужные абзацы;

В группе «Абзац» щелкните по кнопке «Абзац»;

Далее в окне «Абзац» действуйте так же, как и в первом случае.

6.3 Разбивка абзацев по страницам

При работе с большими документами, бывает нужно задать пропорциональное размещение абзацев текста на страницах. Например, вы пишете документ с большими абзацами, и вам надо, чтобы каждый новый абзац начинался с новой страницы. Для разбивки абзацев по страницам сделайте следующее:

1. Выделите необходимые абзацы;
2. Щелкните по выделенному фрагменту правой кнопкой мыши и в появившемся контекстном меню, выберите пункт «Абзац»;
3. В окне «Абзац» перейдите на вкладку «Положение на странице»;
4. В группе «Разбивка на страницы» (рисунок 2.9) выберите нужные пункты:

– «Запрет висячих строк» – запрет перехода одной строки на следующую страницу или первой строки на предыдущую;

– «Не отрывать от следующего» – запрет перехода следующего абзаца на следующую страницу;

– «Не разрывать абзац» – запрет деления абзаца на две страницы;

– «С новой страницы» – вставка разрыва страницы перед абзацем (каждый следующий абзац начинается на новой странице).

6.4 Междустрочные интервалы

Междустрочный интервал очень часто используется при форматировании документов, а наиболее распространенные его значения – одинарный и полуторный. Сам интервал определяется на основе базовых линий строк, которые проходят по центру букв (при этом учитываются их верхние и нижние элементы, типа надстрочного/подстрочного знаков). Так, одинарный интервал, соответствует величине шрифта текста, и является интервалом «по умолчанию» в Word 2010. Полуторный интервал используется в большинстве деловых документов, в курсовых и рефератах.

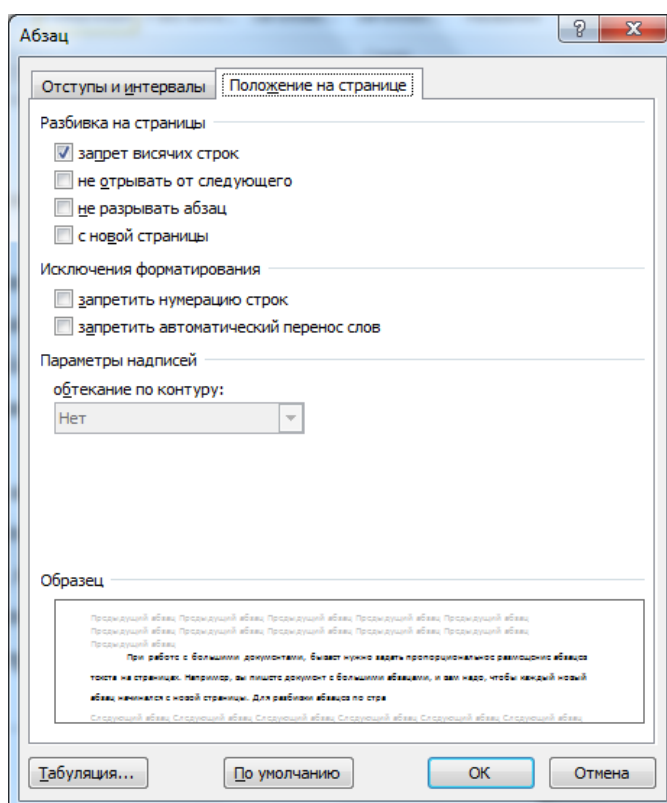


Рисунок 2.9 – Разбивка абзаца при помощи средств вкладки «Положение на странице»

Если в строке имеются знаки большого размера, графика или формулы, междустрочный интервал для этой строки автоматически увеличивается. Чтобы в пределах абзаца интервалы между всеми строками были одинаковы, необходимо задать точное значение междустрочного интервала, достаточное для того, чтобы самые крупные знаки или графические объекты полностью отобразились на строке. Если некоторые знаки или графические объекты отображаются не полностью, междустрочный интервал необходимо

увеличить.

Для того чтобы задать междустрочный интервал, сделайте следующее:

1 способ:

1) выделите нужные абзацы и щелкните правой кнопкой мыши по выделенному участку;

2) в контекстном меню выберите пункт «Абзац»;

3) в окне «Абзац», вкладки «Отступы и интервалы» в группе «Интервал» в графе «Междустрочный» выберите из списка нужный:

– «Одинарный» – интервал соответствует размеру шрифта текста абзаца;

– «1,5 строки» – интервал равен полуторному размеру шрифта текста абзаца;

– «Двойной» – интервал равен двойному размеру шрифта текста абзаца;

– «Минимум» – интервал не меньше заданного в пунктах размера;

– «Точно» – интервал точно соответствует заданному в пунктах размеру (то есть если шрифт равен 12пт, то интервал будет точно таким);

– «Множитель» – множитель для интервала, равного размеру шрифта текста абзаца (чем больше множитель, тем больше интервал).

2 способ:

На панели инструментов нажмите кнопку «Интервал» и выберите нужный интервал из выпадающего списка (рисунок 2.10).



Рисунок 2.10 – Настройка интервалов

3 способ:

1) выделите нужные абзацы;

2) используя сочетания клавиш, задайте необходимый интервал:

- Ctrl+1 – одинарный интервал;
- Ctrl+5 – полуторный интервал;
- Ctrl+2 – двойной интервал.

6.5 Настройка интервалов между абзацами

1 способ:

- выделите нужные абзацы;
- щелкните по выделенному фрагменту правой кнопкой мыши и в появившемся контекстном меню, выберите пункт «Абзац»;
- в окне «Абзац» вкладки «Отступы и интервалы» в группе «Интервал» в пунктах «Перед» (сверху абзаца) и «После» (после абзаца) задайте при помощи регуляторов нужное значение. Учтите что каждый щелчок по ним, соответствует шагу кратному 6пт (пунктов). Чтобы ввести произвольное значение, используйте область ввода данных.

2 способ:

- выделите нужные абзацы и перейдите к вкладке «Разметка страницы»;
- в группе «Абзац» в графе «Интервал» задайте нужное значение.

7 Разрывы страниц

Зачастую возникает необходимость начать новую страницу, в то время как предыдущая еще не заполнена полностью текстом. Например, в книге так начинается новая глава или перед заголовком в документе необходимо принудительно перейти к новой странице

Чтобы начать новую страницу в редакторе Word есть специальная опция – «Разрывы». При добавлении текста Word автоматически вставляет «мягкий» (изменяемый) перенос, иначе называемый «разрывом страницы». По мере редактирования текста этот разрыв меняет свое положение относительно текста, оставаясь в конце текущей страницы. В обычном

режиме просмотра мягкий разрыв отображается пунктирной линией, в разметке страниц или в предварительном просмотре вы видите страницы «как есть».

«Жесткий» (принудительный) разрыв страницы можно вставить в любом месте, при этом последующий текст будет всегда начинаться с новой страницы. В обычном режиме просмотра и при просмотре непечатаемых символов для остальных режимов можно увидеть жесткий разрыв страницы в виде пунктирной линии со словами «Разрыв страницы».

Иногда возникает необходимость использовать различные параметры форматирования для разных страниц документа (например, один из листов документа должен иметь альбомную ориентацию). В этом случае документ необходимо разбить на разделы. Каждый раздел можно будет форматировать совершенно независимо от других разделов.

Последовательность действий для вставки разрывов:

1. Откройте документ.
2. Поставьте курсор перед каким-либо словом.
3. В группе «Страницы» вкладки «Вставка» нажмите кнопку «Разрывы страниц»- «Страница» или комбинацию клавиш «Ctrl + Enter».
4. В документ будет вставлен разрыв страницы, он представляет непечатаемый знак, который можно увидеть в режиме отображения непечатаемых знаков.
5. Для удаления разрыва страницы удалите непечатаемый знак «Разрыв страницы».
6. Установите курсор в конце документа во вкладке «Вставка» в группе «Параметры страницы» нажмите кнопку «Разрывы» и в появившемся меню выберите команду «Разрывы разделов» - «Следующая страница»
7. Переместите место вставки за маркер первого раздела, например, командой «Ctrl+End» - в конец документа.

Для ручного разделения таблицы необходимо выделить таблицу, открыть справа раздел «Конструктор», затем - «Макет» - «Разделить».

8 Нумерация страниц, удаление номера с титульного листа

Если документ состоит более чем из двух страниц, то перед распечаткой желательно пронумеровать страницы. Благодаря этому в нем будет гораздо легче ориентироваться. Для некоторых типов документов (например, для научных работ) нумерация необходима. Номер страницы можно разместить в верхнем или нижнем полях, по краю или по центру. Для нумерации страниц служит кнопка «Номер страницы» (лента «Вставка», панель «Колонтитулы»). Чтобы пронумеровать страницы, сделайте следующее.

1. Перейдите на вкладку Вставка и нажмите кнопку «Номер страницы» в группе «Колонтитулы» (рисунок 2.11).

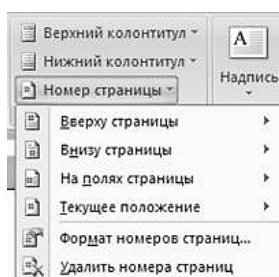


Рисунок 2.11 – Меню «Номера страниц»

2. Выберите положение номера на странице, используя доступные варианты в подменю «Вверху страницы», «Внизу страницы» и «На полях страницы». Microsoft Word предлагает самые разные варианты оформления номеров страниц.

При нажатии кнопки «Номер страницы» открывается окно «Формат номера страницы», где можно установить тип нумерации - продолжающаяся (сквозная) или локальная (для раздела), а также задать формат вывода номеров страниц (вместо арабских цифр можно выбрать римские, буквы алфавита и др.).

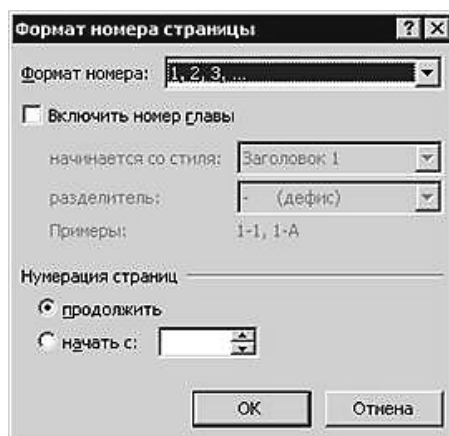


Рисунок 2.12 – Выбор формата номера страницы

При необходимости можно в окне «Формат номеров страниц» выбрать форматы нумерации и снять «номер на первой странице».

9 Вставка формул

Формулы могут быть вставлены двумя способами. Первый – с помощью команды «Вставка → Формула». После этого появляется фрейм для вставки формулы, в котором можно что-то напечатать. Самое интересное, что этот фрейм «Место для формулы», если нажать на стрелочку с правого края можно сделать встроенным и он будет встроен в строку текста, а можно так и оставить между строками текста.

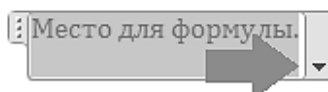


Рисунок 2.13 – Вставка формул

Выделяем фрейм «Место для формулы» и на закладке «Работа с формулами: Конструктор» начинаем вводить математические символы, либо сразу выбираем формулу, где вместо пустых ячеек или готовых значений вводим свои.



Рисунок 2.14 – Работа с формулами

Этот способ ввода формул появился MS Word относительно недавно, поэтому в случае необходимости сохранения документа в старом формате *.doc (например, для возможности открытия в MS Word 2003) формулы, набранные таким способом, будут преобразованы в рисунок и не доступны для редактирования. В связи с этим, рекомендуется использовать второй способ ввода формул через вставку объекта «Microsoft Equation 3.0» (Вкладка «Вставка», кнопка «объект»).

10 Изменение ориентации страницы

Microsoft Word предлагает различные варианты разметки страницы и параметры форматирования, которые влияют на то, как содержимое документа отображается на странице. Вы можете настроить ориентацию страницы, размер страницы и поля, в зависимости от того, как вы хотите, чтобы выглядел документ.

Чтобы изменить ориентацию страницы:

1. Выберите вкладку Разметка страницы.
2. Кликните по команде Ориентация в группе Параметры страницы.
3. Для изменения ориентации кликните по одному из вариантов: Книжная или Альбомная.

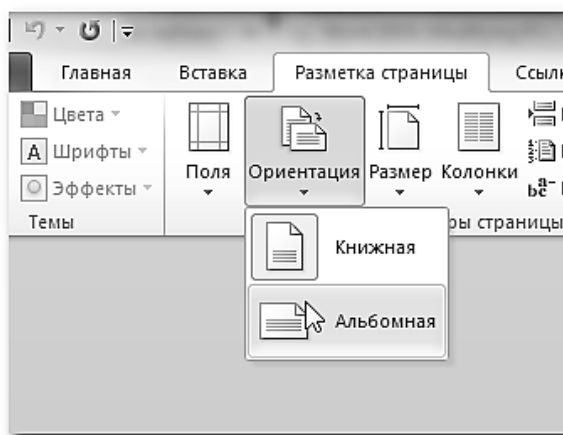


Рисунок 2.15 – Изменение ориентации страницы

Альбомная означает, что страница будет ориентирована горизонтально, а Книжная – вертикально.

Чтобы изменить размер страницы:

1. Выберите вкладку Разметка страницы.
2. Кликните по команде Размер, появится выпадающее меню. Текущий размер страницы выделен.
3. Кликните по нужному вам размеру. Размер страницы документа изменится.

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает

с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Используя возможности текстового редактора Microsoft Word, исправьте в соответствии с вашим вариантом документ согласно требованиями ЕСКД к оформлению текстовых документов.

2. В качестве приложения добавьте к исправленному документу заполненную в результате выполнения прошлой лабораторной работы таблицу-памятку по оформлению текстовых документов. Таблица должна располагаться на листе с альбомной ориентацией.

3. В качестве второго приложения добавьте введенные двумя способами указанные преподавателем 5-7 формул.

Содержание отчета

Исправленный документ согласно задания.

Контрольные вопросы

1. Где находится кнопка отображения непечатаемых знаков?
2. Что такое «мягкий» и «жесткий» разрывы страниц? Как их вставлять?
3. Как удалить номер с первой страницы документа?
4. Какими способами можно разорвать таблицу в документе?
5. Как поменять ориентацию части страниц документа?
6. Какие режимы просмотра документа вы знаете? Каково назначение каждого из них? Какие режимы применяются при создании и оформлении текстовых документов согласно стандартам ЕСКД?
7. Что такое режим отображения непечатаемых знаков? Какие непечатаемые знаки вы знаете? Как включить режим отображения границ текста?
8. Какие способы ввода формул вы знаете?

Лабораторная работа №3. Применение на практике возможностей текстовых редакторов для автоматизации оформления текстовых документов

Цель работы: Научиться применять возможности текстовых редакторов для автоматизации создания текстовых документов согласно ГОСТ 2.105-95 ЕСКД.

Теоретическая часть

Текстовый редактор Word 2010 обладает рядом возможностей, которые позволяют в значительной степени облегчить и автоматизировать процесс оформления текстовых документов согласно стандарту ГОСТ 2.105-95 ЕСКД. К ним относятся: автоматическая нумерация разделов и подразделов, автоматическая нумерация таблиц, рисунков и формул, автоматический перенос таблиц, автоматическая генерация содержания, автоматическая расстановка отступов между заголовками и так далее. Рассмотрим подробнее каждый из них.

1 Структура документа

Очень часто, работая с большими документами, пользователи делают не структурное форматирование (с помощью стилей), а визуальное (например, заголовки набирают большим размером шрифта и делают их полужирными). Такой подход имеет свои преимущества и недостатки. К преимуществам можно отнести простоту, логичность, быстроту форматирования небольших текстов. Однако, в случае работы с большими документами (от 30 страниц и более), перечисленные достоинства превращаются в недостатки. Предположим, текст имеет 4 раздела, а в каждом разделе есть по 4 подраздела. Итого получается 4 заголовка разделов и 16 заголовков подразделов. Допустим, при оформлении заголовков пользователь забыл, что от текста их необходимо отделять расстоянием 7-

10 мм. В случае визуального оформления необходимо отыскать в тексте все 20 заголовков и назначить им необходимый отступ. Эта процедура займет определенное время. В больших документах количество заголовков может приближаться к сотне, что еще больше затрудняет процедуру форматирования. Структурное же форматирование позволяет изменять оформление сразу всех однотипных элементов. Суть его заключается в том, чтобы не осуществлять непосредственное форматирование текста, а разбить его на определенные структурные группы (заголовки рисунков, таблиц, разделов, подразделов и так далее), после чего задать стили оформления каждой из этих групп. Такой подход также имеет свои достоинства и недостатки. К недостаткам можно отнести дополнительные затраты времени на формирование структурных групп и менее логичный подход к форматированию текста. Если документ создается «с нуля», как правило, перечисленные недостатки не носят ярко выраженный характер. Однако в случае редактирования уже готового документа, оформленного по принципу визуального форматирования, может потребоваться значительное время на его переделывание под структурное форматирование. В связи с этим, при создании нового документа рекомендуется сразу использовать структурное форматирование, а при редактировании уже существующих, созданных по принципу визуального форматирования, во многих ситуациях целесообразнее не тратить время на структурирование документов.

Рассмотрим процесс создание структуры на примере нового документа. Предположим, документ имеет 4 раздела, в каждом из которых по 4 подраздела. Создадим новый документ и на отдельных строчках введем названия всех заголовков:

Введение

Раздел 1

Подраздел 1.1

Подраздел 1.2

Подраздел 1.3

Подраздел 1.4

Раздел 2

.....

Раздел 3

.....

Заключение

Список использованных источников

Перейдем в режим структуры документа. Для этого необходимо выполнить команду меню «Вид → Структура». Выделим поочередно каждый из этих заголовков и с помощью панели инструментов «Работа со структурой» (рисунок 3.1) назначим каждому заголовку свой уровень (первый уровень – для заголовков разделов, второй уровень – для заголовков подразделов)

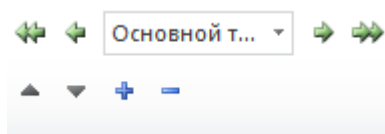


Рисунок 3.1 – Панель инструментов «Работа со структурой»

При назначении уровней можно заметить, что одновременно меняется форматирование текста (размер, тип начертания и так далее). Это связано с тем, что каждому элементу структуры автоматически назначается определенный стиль из активного набора стилей.

2 Стили оформления

Стили представляют собой набор атрибутов форматирования, то есть могут включать гарнитуру, начертание и размер шрифта, выравнивание и др. К выделенному фрагменту текста все атрибуты форматирования стиля применяются одновременно. В этом заключается первое удобство использования стилей. Второе состоит в том, что стили позволяют, задав

один раз необходимые параметры, пользоваться ими все время, даже в разных документах.

В Word есть большое количество заготовленных стилей, к которым наиболее часто обращаются пользователи программы. Со стилями работают все, иногда того не зная. Даже если вы не задавали никакой стиль, программа по умолчанию идентифицирует его как Обычный.

Для применения стилей используется группа Стили на вкладке Главная на ленте.

Различают стили абзаца, символа, таблицы и списка. Это означает, что тот или иной стиль может быть применен к абзацу, к произвольному участку текста, списку или таблице соответственно.

Стили группируются в наборы стилей – совокупность стилей для каждого элемента структуры документа, который однозначно определяют его внешний вид и позволяют «одним щелчком мыши» изменить оформление текста под необходимые правила.

В текстовом редакторе MS Word есть несколько предустановленных наборов стилей. Для их просмотра необходимо на вкладке «Главная» нажать кнопку «Изменить стили → Набор стилей». К стандартным наборам стилей относятся: «Газетная бумага», «Изысканные», «Простые», «Рукописные», «Современные» и другие. К сожалению, ни один из стандартных наборов стилей не соответствует стандартам ЕСКД. Но к счастью, Word позволяет создавать свои собственные наборы стилей. Для этого нужно применить любой из стандартных наборов стилей (например, Word 2003), и выполнить команду «Изменить стили → Набор стилей → Сохранить как набор экспресс-стилей». Появится диалоговое окно, в котором нужно ввести имя сохраняемого стиля. После сохранения, в наборе стилей появится соответствующий введенному имени, новый набор стилей. Активируйте его и нажмите на кнопку «Стили» в правом нижнем углу соответствующей группы. Появится окно редактирования стилей, изображенное на рисунке 3.2.

По умолчанию в списке стилей отображаются не все стили, а только те,

которые используются в текущем документе. При необходимости, можно включить отображение всех стилей, если нажать на кнопку «Параметры» и выбрать все отображаемые стили (рисунок 3.3).

Выделите в тексте документа заголовок раздела, соответствующий ему стиль «Заголовок 1» станет активным. Через контекстное меню стиля выберите пункт «Изменить». Откроется окно редактирования стиля, изображенное на рисунке 3.4.

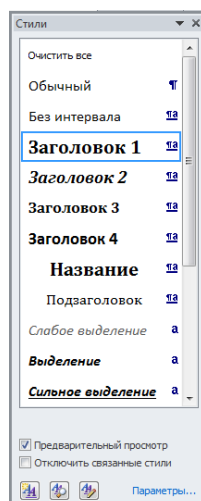


Рисунок 3.2 – Список стилей выбранного набора стилей

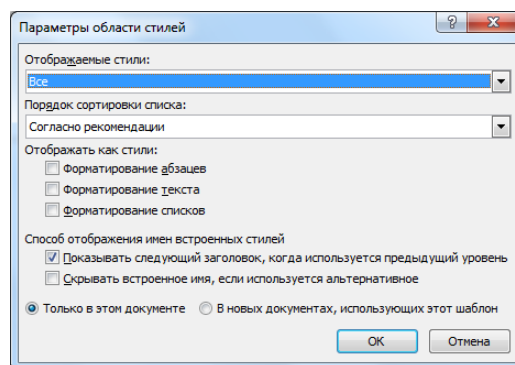


Рисунок 3.3 – Параметры отображения стилей

Здесь можно задать все правила оформления заголовков первого уровня, включая размер, выравнивание текста, отступ от текста, запрет отрыва заголовков от следующего за ним текста, необходимость начинать каждый новый раздел с новой страницы и так далее.

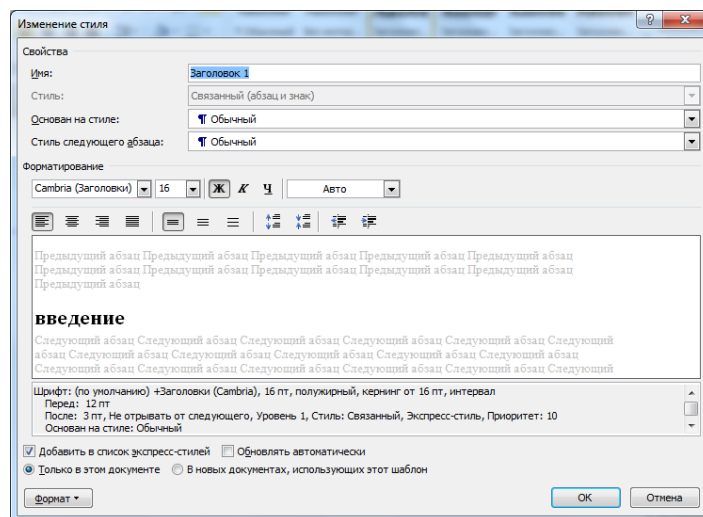


Рисунок 3.4 – Окно редактирования стиля

Аналогичным образом настраиваются правила оформления подзаголовков (заголовки второго уровня), основного текста (стиль «Обычный») и так далее.

3 Автоматическая нумерация разделов и подразделов

Для автоматической нумерации разделов и подразделов необходимо воспользоваться многоуровневым списком. Возможны два подхода: создать свой новый многоуровневый список, либо воспользоваться уже готовым списком из библиотеки списков и отредактировать его в соответствии с требованиями ЕСКД. Наиболее соответствующий требованиям ЕСКД список представлен на рисунке 3.5.

Его необходимо применить к любому из заголовков документа. После этого все заголовки будут пронумерованы. Однако обратите внимание, что между цифрой и названием раздела стоит символ табуляции, а не пробела.

Чтобы это исправить, необходимо через контекстное меню списка выбрать команду «Изменить отступы в списке». Откроется диалоговое окно, изображенное на рисунке 3.6.

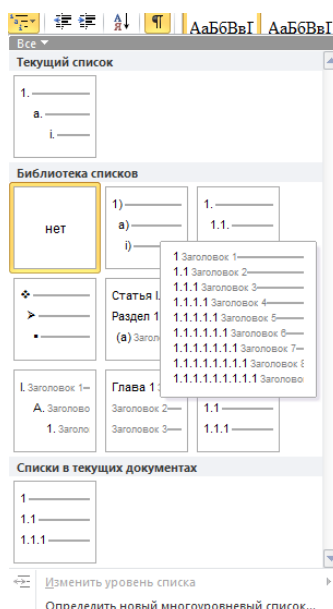


Рисунок 3.5 – Многоуровневый список автоматической нумерации разделов из библиотеки списков MS Word

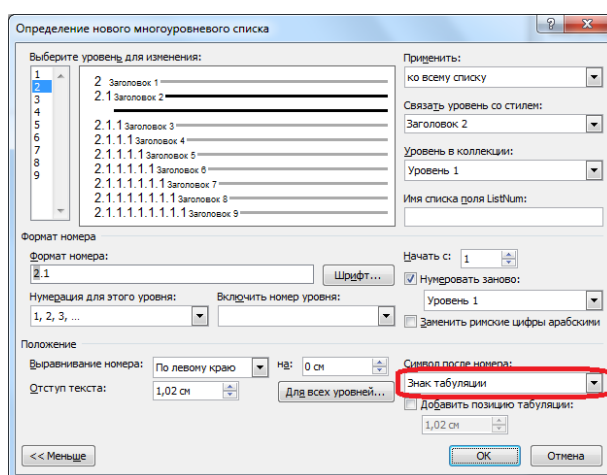


Рисунок 3.6 – Диалоговое окно редактирования списка

В нем необходимо поменять знак табуляции, на пробел, а еще лучше, вообще убрать символ после номер и ввести неразрывный пробел после цифры в поле «Формат номера». Кроме того, для некоторых заголовков (введение, заключение и др.) необходимо отключить нумерацию.

4 Автоматическая генерация содержания

Применение структурного форматирования позволяет также задействовать мощные средства MS Word по автоматизации создания содержания документа. Для этого необходимо перейти на вкладку «Ссылки»,

и выполнить команду меню «Оглавление → Оглавление». Откроется диалоговое окно, изображенное на рисунке 3.7. В нем можно задать максимальный уровень вложенности подзаголовков, которые будут вынесены в содержание, а также изменить шрифт каждого уровня (через кнопку «Изменить»). В дальнейшем, шрифт всегда можно отредактировать через список стилей (стили с именами «Оглавление 1», «Оглавление 2» и так далее).

В процессе работы с документом, названия разделов и подразделов могут измениться, поэтому в MS Word предусмотрена возможность автоматического обновления оглавления. Для этого необходимо щелкнуть правой кнопкой по сгенерированному оглавлению и в открывшемся контекстном меню выбрать пункт «Обновить поле». При этом возможны два варианта: обновить оглавление целиком, либо же обновить только страницы оглавления.

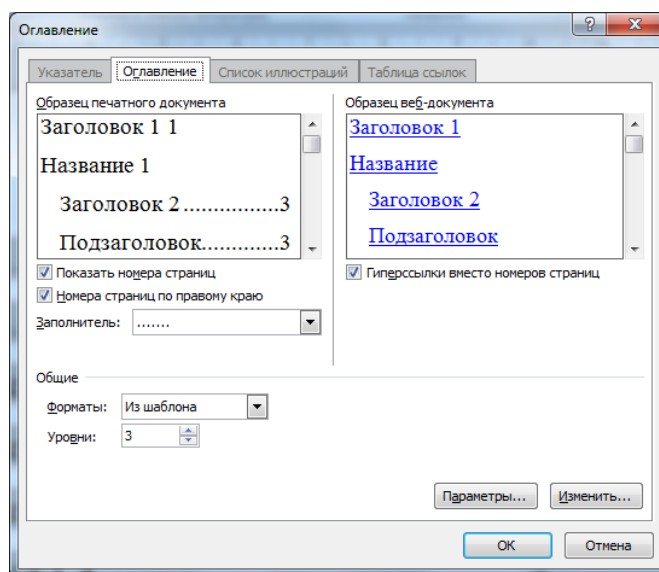


Рисунок 3.7 – Диалоговое окно создания оглавления

5 Автоматическая нумерация рисунков

Для автоматической нумерации рисунков необходимо установить курсор под рисунком и выполнить команду меню «Ссылки → Вставить название». Откроется диалоговое окно, изображенное на рисунке 3.8, в котором можно задать способ нумерации рисунков (сквозной или в пределах

раздела).

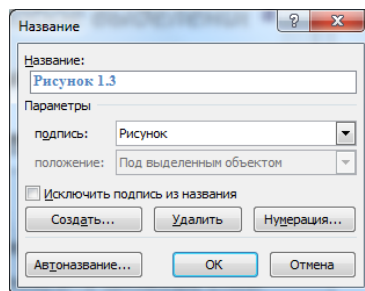


Рисунок 3.8 – Диалоговое окно автоматической нумерации рисунков

После этого необходимо отредактировать соответствующий стиль подписи рисунков.

6 Автоматическая нумерация таблиц

Таблицы нумеруются аналогичным образом. Однако после нумерации Word автоматически присваивает им такой же стиль, как и для подписи рисунков. Согласно стандартам ЕСКД выравнивание подписей рисунков и таблиц разное. Поэтому для таблиц необходимо создать дополнительный стиль. Для этого в окне списка стилей, нужно нажать на кнопку «Создать стиль», ввести имя «Название таблиц», в поле «Основан на стиле» выбрать стиль, соответствующий подписям рисунков и изменить выравнивание текста.

7 Автоматическая нумерация формул

Формулы нумеруются похожим образом. Однако, во-первых, необходимо поставить галочку «Исключить подпись из названия», а, во-вторых, аналогично таблицам, необходимо создать отдельный стиль для подписей формул. К сожалению, скобки в номерах формул придется ставить вручную. Однако можно настроить выравнивание по правому краю с помощью раздела «Табуляция» стиля (рисунок 3.9).

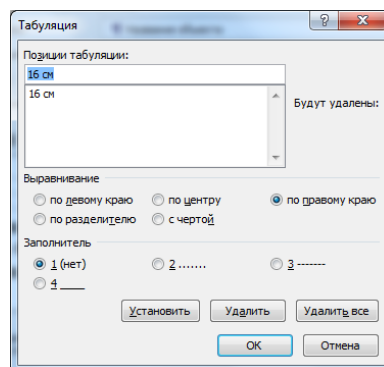


Рисунок 3.9 – Задание отступов табуляции для выравнивания формул

9 Перекрестные ссылки и закладки

По требованиям ЕСКД, на все таблицы, рисунки и формулы должны быть ссылки в тексте. В случае автоматической нумерации невозможно заранее предугадать, какой номер будет у рисунка, таблицы и формулы. Поэтому необходимо, чтобы номер вычислялся и вставлялся автоматически. В редакторе MS Word есть стандартные средства, позволяющие выполнять такие действия. Такой инструмент называется «Перекрестные ссылки». Установите курсор в нужную позицию в тексте и нажмите кнопку «Вставка → Перекрестная ссылка». Откроется диалоговое окно, изображенное на рисунке 3.10.

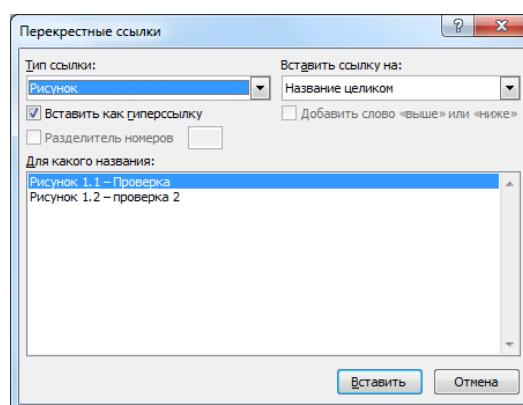


Рисунок 3.10 – Вставка перекрестных ссылок

Как видно из рисунка, в списке содержатся все автоматически пронумерованные объекты (рисунки, таблицы, формулы). Однако при попытке вставить ссылку на них, в текст вставляется надпись целиком (вместе со словом Рисунок и его названием). Аналогичным образом обстоят

дела с таблицами и формулами. Остается лишь догадываться, почему разработчики MS Word реализовали именно такой алгоритм. Но к счастью, его можно обойти с помощью инструмента «Закладки». Для вставки закладки, нужно выделить интересующий фрагмент текста (например, номер рисунка) и нажать на кнопку «Вставка → Закладка». В открывшемся диалоговом окне нужно ввести какое-нибудь осмысленное название рисунка без точек и пробелов. После этого, она появится в списке перекрестных ссылок, если выбрать тип ссылки «Закладки». Для удобства, можно включить режим отображения закладок в настройках программы через меню «Файл → Параметры → Дополнительно → Показывать содержимое документа → Показывать закладки». В этом случае, все закладки в тексте будут выделены полупрозрачными серыми квадратными скобками.

10 Обновление всех полей

Как было показано выше, MS Word умеет автоматически нумеровать рисунки, формулы и таблицы. Однако в процессе работы с документом, может возникнуть ситуация, когда часть рисунков или формул были удалены. В этом случае, необходимо заново произвести нумерацию. Для этого необходимо выделить интересующий кусок текста и нажать на клавишу F9. Если необходимо обновить нумерацию по всему документу, сначала нужно выделить весь текст (команда Ctrl+A), а затем нажать на клавишу F9.

11 Автоматический перенос таблиц

При работе с таблицами в текстовом процессоре MS Word иногда возникает необходимость повторять определенную строку как заголовок на каждой странице. Если это первая строка таблицы либо несколько первых строк, то делается это достаточно просто, для этого при работе с таблице, во вкладке «Макет», необходимо установить галочку «Повторить строки заголовков»

Однако, иногда возникает необходимость повторять, в качестве заголовка, не первые строки, а только вторую, третью и т.д. (например, рисунок 3.11)

Список научных трудов Иванова Ивана Ивановича

№ п/п	Название	Характер работы	Выходные данные	Объем, стр.	Соавторы
1	2	3	4	5	6
I. Научные труды, опубликованные до защиты кандидатской диссертации					
1.	Влияние гармонических колебаний на поведение домашних животных (статья)	печатн.	Проблемы домашних животных : сб. научных трудов / МГУ. – Москва, 2013. – С. 45-65	20	-
2.	---				
3.					
4.					

Рисунок 3.11 – Пример таблицы для переноса

В данном случае необходимо воспроизвести вторую строку при каждом переносе таблицы на следующую страницу. Если повторение второй строки организовать вместе с первой, то проблем нет никаких.

Однако если вторую строку необходимо повторять, а первую нет – тогда проблема, поскольку MS Word такой возможности не предоставляет, опция повторения строки просто-напросто неактивна.

К сожалению, просто выставить повторение лишь второй строки не получится. Разработчики не предусмотрели такой возможности, поэтому придется проявить фантазию и воспользоваться инструментарием, который предоставляет пользователю MS Word.

Наиболее простой способ заключается в том, чтобы создать не одну таблицу, а две: первая таблица будет состоять лишь из одной строки – это строка заголовка, а вторая – основная, в качестве заголовков будет иметь строку с нумерацией (рисунок 3.12).

И следующим шагом, который приходит на ум – удаление строки, которая разделяет две таблицы, но «умный» Word при этом преобразует две таблицы в одну и, соответственно, мы приходим к тому, с чего начинали, т.е. одной таблице, в которой невозможно повторить вторую строку в одиночку. Интересный момент заключается в том, что даже если предварительно

позаботиться о том чтобы первую строку основной таблицы сделать повторяющейся, а склеить таблицы после, то повторяющаяся строка также удалиться.

Поэтому, чтобы достичь поставленной цели придется прибегнуть к определенной хитрости, а именно не удалять строку, а сделать ее очень маленькой в один пункт и, дабы еще больше уменьшить расстояние выставить междустрочный интервал минимальным. Если планируется сделать несколько таблиц, то для размера шрифта и междустрочного интервала следует сделать свой стиль.

О том, что с таблицей что-то не так выдает лишь более толстая полоса, отделяющая верхушку таблицы от основной части, для исправления этого, просто сделаем невидимой нижнюю границу верхней таблицы.

№ п/п	Название	Характер работы	Выходные данные	Объем, стр.	Совторы
1	2	3	4	5	6
I. Научные труды, опубликованные до защиты кандидатской диссертации					
1.	Влияние гармонических колебаний на поведение домашних животных (статья)	печати	Проблемы домашних животных. сб. научных трудов / МГУ. – Москва, 2013. – С. 45-65	20	-
2.	...				
3.					
...					
31.					
32.					
33.					
34.					

1	2	3	4	5	6
35.					
36.					

Рисунок 3.12 – Пример разделения таблицы на две части

Теперь вообще ничего не выдает то, что вместо одной таблицы, на самом деле, две. Естественно таким же образом можно отделить не одну строку сверху, а несколько.

К сожалению, возможностей для автоматической вставки фразы

«Продолжение таблицы» MS Word также не предусматривает. Однако, согласно документу «Изменение № 1 ГОСТ 2.105-95 Единая система конструкторской документации. Общие требования к текстовым документам», принятому Межгосударственным советом по стандартизации, метрологии и сертификации по переписке (протокол № 23 от 28.02.2006), при подготовке текстовых документов с использованием программных средств надпись «Продолжение таблицы» допускается не указывать. Кроме того, изменения коснулись и нижней горизонтальной линии, ограничивающей таблицу при переносе:

«Если в конце страницы таблица прерывается и ее продолжение будет на следующей странице, в первой части таблицы нижнюю горизонтальную линию, ограничивающую таблицу, допускается не проводить».

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических

розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Создайте новый документ со структурой, соответствующей вашему заданию по вариантам из предыдущей лабораторной работы.

2. Попробуйте применить различные наборы стилей форматирования, посмотрите как изменится оформление документа. Убедитесь, что стандартные наборы стилей не соответствуют требованиям ЕСКД к оформлению текстовых документов.

3. Создайте свой набор стилей, настройте стили в соответствии со стандартами ЕСКД.

4. Вставьте в документ несколько формул и сделайте их автоматическую нумерацию.

5. Вставьте в документ несколько рисунков и сделайте их автоматическую нумерацию.

6. Вставьте в документ несколько таблиц и сделайте их автоматическую нумерацию.

7. Сделайте ссылки в тексте, на формулы, рисунки и таблицы, пронумерованные автоматически.

8. Настройте автоматический перенос таблиц.

9. Оформите текстовый документ, соответствующий вашему варианту из предыдущей лабораторной работы.

10. Сохраните созданный набор стилей, для его повторного применения на домашнем компьютере.

11. Ознакомьтесь с документом «Изменение №1 к ГОСТ 2.105-95».

Содержание отчета

Оформленный документ согласно задания.

Контрольные вопросы:

1. Какие подходы к форматированию текста вам известны? В чем преимущества и недостатки каждого из них?
2. Что такое набор стилей? Для чего они применяются? Какие стандартные наборы стилей существуют? В чем их недостаток?
3. Как создать свой набор стилей?
4. Как отредактировать стили из набора стилей?
5. Как создать новый стиль в наборе стилей?
6. Как выполнить нумерацию разделов и подразделов?
7. Как автоматически составить содержание документа?
8. Как выполнить автоматическую нумерацию таблиц, формул и рисунков в документе? В чем преимущества и недостатки возможностей текстовых редакторов по выполнению автоматической нумерации формул, таблиц и рисунков?
9. Как в документе создать закладку? Каким способом создать перекрестную ссылку на закладку? Как включить режим отображения закладок? Для чего он нужен?
10. Как обновить оглавление и нумерацию таблиц, формул и рисунков?
11. Какие важные поправки были внесены документом «Изменение №1 к ГОСТ 2.105-95»

Лабораторная работа №4. Первичное знакомство с векторными редакторами

Цель работы: изучение основных правил оформления чертежей, приобретение основных навыков работы с простейшими графическими инструментами векторных редакторов.

Теоретическая часть

1 Основные правила выполнения чертежей

Перечень стандартов ЕСКД, регламентирующих правила выполнения чертежей следующий:

1. Основные положения.
 - Общие положения: ГОСТ- 2.001-70.
 - Виды изделий: ГОСТ- 2.101-68.
 - Стадии разработки: ГОСТ- 2.103-68.
 - Основные надписи: ГОСТ- 2.104-68.
 - Основные требования к текстовым документам: ГОСТ- 2.105-68.
 - Основные требования к рабочим чертежам: ГОСТ-2.107-68.
 - Спецификация: ГОСТ- 2.108-68.
 - Правила выполнения чертежей деталей, сборочных, общих видов, габаритных и монтажных: ГОСТ- 2.109-68.
 - Технические условия, Правила построения, изложения и оформления: ГОСТ- 2.114-70.
2. Общие правила выполнения чертежей.
 - Форматы: ГОСТ- 2.301-68.
 - Масштабы: ГОСТ- 2.302-68.
 - Линии: ГОСТ- 2.303-68.
 - Шрифты чертежные: ГОСТ- 2.304-81.
 - Изображения - виды, разрезы, сечения: ГОСТ- 2.305-68.

- Обозначение графических материалов и правила их нанесения на чертежах: ГОСТ- 2.306-68.
- Нанесение и указание размеров и предельных отклонений: ГОСТ- 2.307-68, ГОСТ- 2.308-68.
- Нанесение на чертежах обозначений шероховатости поверхностей: ГОСТ- 2789-73.
- Нанесение на чертежах обозначений покрытий, термической и других видов обработки: ГОСТ- 2.310-68.
- Изображение резьбы: ГОСТ- 2.311-68.
- Условное изображение и обозначение швов сварных соединений: ГОСТ- 2.312-68.
- Условное изображение и обозначение швов неразъемных соединений: ГОСТ- 2.313-68.
- Изображения упрощенные и условные крепежных деталей : ГОСТ- 2.315-68.
- Правила нанесения на чертежах надписей, технических требований и таблиц: ГОСТ- 2.316-68.
- Аксонометрические проекции: ГОСТ- 2.317-69.
- Правила выполнения чертежей различных изделий (пружин, зубчатых колес, реек, червяков, звездочек цепных передач, подшипников и т.п.) ГОСТ- 2.401-68, ГОСТ- 2.421-70.
- Правила выполнения схем и условные графические обозначения (общие требования, схемы электрические, кинематические, машины электрические и их элементы, гидравлические насосы и двигатели, трубопроводы и их элементы) ГОСТ- 2.701-68 ГОСТ- 2.786-70.

1.1 Форматы

Каждый чертеж должен быть выполнен на листе определенных размеров, который называется форматом. ГОСТ 2.301-68* устанавливает пять основных форматов для чертежей и других конструкторских

документов: A0, A1, A2, A3, A4. Площадь формата A0 равна $\sim 1 \text{ м}^2$. Другие основные форматы могут быть получены последовательным делением формата A0 на две равные части параллельно меньшей стороне соответствующего формата (рисунок 4.1).

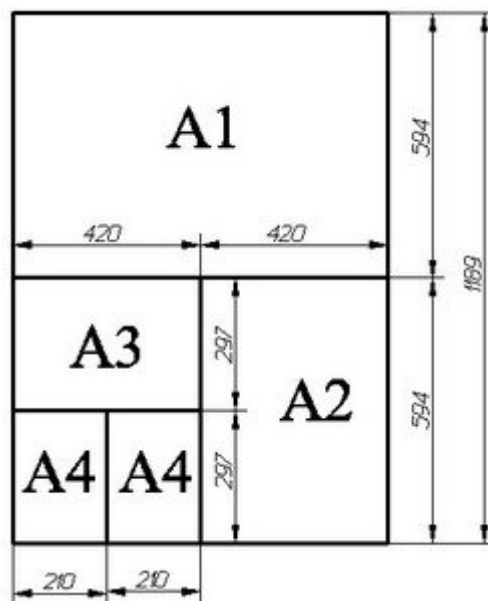


Рисунок 4.1 – Схема деления форматов

Размеры сторон основных форматов приведены в таблице 4.1.

Таблица 4.1 – Форматы листов

Обозначение формата	Размеры сторон формата, мм
A0	841 × 1189
A1	594 × 841
A2	420 × 594
A3	297 × 420
A4	210 × 297

При необходимости допускается применять формат A5, с размерами сторон 148×210 мм. Также допускается применение дополнительных форматов образуемых увеличением коротких сторон основных форматов на величину кратную их размерам.

В технике все линейные измерения производят в мм и единицы измерения не указывают, в том числе на чертежах. При наличии других единиц – их обозначения указывают.

На чертежи наносится рамка (обрамляющая линия), которую проводят сплошной толстой основной линией. Обрамляющая линия проводится вдоль левой стороны формата на расстоянии 20 мм от внешней рамки (поле для подшивки), а вдоль остальных сторон – на расстоянии 5 мм (рисунок 4.2).

Формат А4 располагается только вертикально. Остальные основные форматы можно располагать и вертикально, и горизонтально.

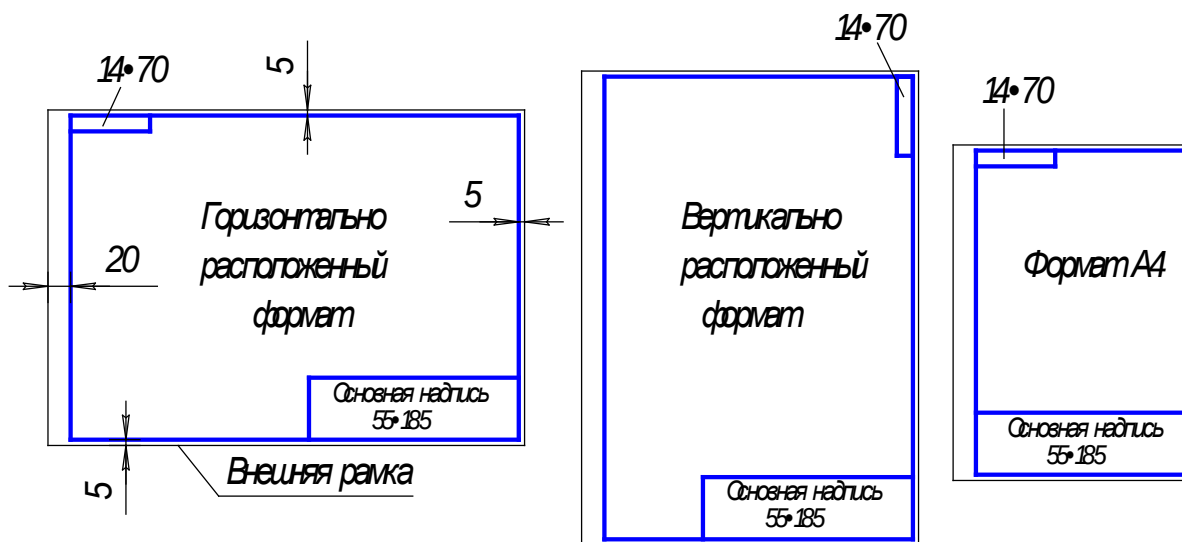


Рисунок 4.2 – Расположение рамки на разных форматах

1.2 Основные надписи

Каждый чертеж должен иметь основную надпись, которая располагается в правом нижнем углу чертежа: на формате А4 вдоль короткой стороны, а на форматах больше А4 может располагаться как вдоль длинной стороны, так и вдоль короткой стороны формата.

ГОСТ 2.104-68* устанавливает форму, размеры, порядок заполнения основных надписей и дополнительных граф к ним в конструкторских документах:

на чертежах и схемах – форма 1 (рисунок 4.3);

на текстовых документах – форма 2 и 2а (рисунок 4.4).

Форма 1

Dimensions: 7, 10, 23, 15, 10, 55, 5, 185, 15, 17, 18, 20, 18.

Fields: (2), (1), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12).

Labels: Изм., Лист, Документа, Подпись, Дата, Разработ, Проверил, Т.контр., Н.контр., Утвердил, Литера, Масса, Масштаб, Лист, Листов.

Рисунок 4.3 – Основная надпись для чертежей и схем

Форма 2

Dimensions: 40, 5, 185.

Fields: (2), (1), (3), (4), (11), (12), (7), (8), (9), (10).

Labels: Изм., Лист, Документа, Подпись, Дата, Разработ, Проверил, Т.контр., Н.контр., Утвердил, Литера, Лист, Листов.

Форма 2а

Dimensions: 15, 5, 185, 10, 7.

Fields: (2), (1), (3), (4), (11), (12), (7), (8), (9), (10).

Labels: Изм., Лист, Документа, Подпись, Дата, Разработ, Проверил, Т.контр., Н.контр., Утвердил, Литера, Лист, Листов.

В учебных заведениях заполняют следующие графы (графы обозначены числами в скобках):

графа 1 – наименование изделия, изображенного на чертеже. Вначале пишут имя существительное, затем определения;

графа 2 – обозначение (номер) чертежа по ГОСТ 2.201-80;

графа 3 – обозначение материала детали (графу заполняют только на чертежах деталей);

графа 4 – литера, присвоенная документу (литера «У» – для учебных чертежей);

графа 5 – масса изделия в килограммах;
графа 6 – масштаб изображения;
графа 7 – наименование учебного заведения (ТГУ) и группы;
графа 8 – фамилии студента и преподавателя;
графа 9 – подписи студента и преподавателя;
графа 10 – дата подписания чертежа;
графа 11 – порядковый номер листа;
графа 12 – общее количество листов документа.

Рисунок 4.4 – Основная надпись для текстовых документов

В графе с размерами 14×70 записывают то же обозначение чертежа, что и в графе 2, только повернутое на 180° для горизонтальных форматов и форматов А4, и на 90° для вертикальных форматов.

1.3 Масштабы

Масштабом называется отношение линейных размеров изображения предмета на чертеже к его действительным размерам.

Предпочтительно выполнять чертежи так, чтобы размеры изображения и самого предмета были равны, т.е. в масштабе 1:1. Однако, в зависимости от величины и сложности предмета, а также от вида чертежа часто приходится размеры изображения увеличивать или уменьшать по сравнению с истинными. В этих случаях прибегают к построению изображения в масштабе.

Согласно ГОСТ 2.302 -68* установлены следующие масштабы:

- натуральная величина – 1:1;
- масштабы уменьшения – 1:2; 1:2,5; 1:4; 1:5; 1:10; 1:15; 1:20; 1:25; 1:40; 1:50; 1:75; 1:100; 1:200; 1:400; 1:500; 1:800; 1:1000;
- масштабы увеличения – 2:1; 2,5:1; 4:1; 5:1; 10:1; 20:1; 40:1; 50:1; 100:1.

При проектировании генеральных планов крупных объектов допускается применение масштабов 1:2000; 1:5000; 1:10000; 1:20000; 1:25000; 1:50000.

При выборе масштаба следует руководствоваться, прежде всего, удобством пользования чертежом.

Масштаб, указываемый в графе, имеющей заголовок «Масштаб» (в основной надписи, в таблицах), обозначают по типу 1:1; 1:2; 2:1 и т. д.

Масштаб изображения, отличающийся от указанного в основной надписи, указывают в скобках (без буквы М) рядом с обозначением изображения.

Например: А (2:1); Б – Б (2:1).

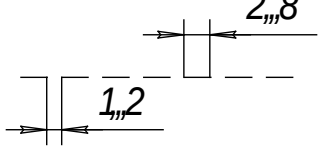
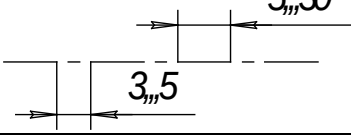
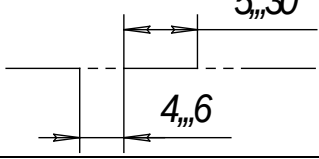
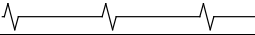
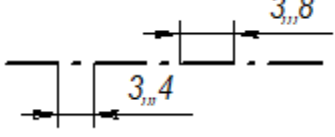
1.4 Линии

ГОСТ 2.303 - 68* устанавливает начертания и основные назначения линий на чертежах всех отраслей промышленности и строительства (таблица 4.2). При выполнении чертежей применяют линии различной толщины и начертания. Каждая из них имеет свое назначение:

1. Сплошная толстая основная линия. Такую линию применяют для изображения видимых контуров предметов, рамки и граф основной надписи чертежа. Ее толщину (s) выбирают в пределах от 0,5 до 1,4 мм в зависимости от величины и сложности изображений, от формата чертежа.

2. Штриховая линия. Она применяется для изображения невидимых контуров предмета. Штриховая линия состоит из отдельных штрихов (черточек) приблизительно одинаковой длины. Длину каждого штриха выбирают от 2 до 8 мм в зависимости от величины изображения. Расстояние между штрихами в линии должно быть от 1 до 2 мм, но приблизительно одинаковое на всем чертеже. Толщина штриховой линии берется от $s/3$ до $s/2$.

Таблица 4.2 – Линии чертежа

№ п/п	Наименование и начертание	Толщина линий по отношению к основной линии	Основное назначение
1.	Сплошная толстая основная (в дальнейшем основная) 	S(0,5...1,4)	Линии видимого контура; линии перехода видимые; линии контура сечения (вынесенного и входящего в состав разреза)
2.	Сплошная тонкая 	S/3...S/2 (0,4...0,7)	Линии контура наложенного сечения; линии размерные, выносные; линии штриховки; линии-выноски, полки линий выносок; линии перехода воображаемые; линии для изображения пограничных деталей (обстановка); линии ограничения выносных элементов.
3.	Сплошная волнистая 	S/3...S/2	Линии обрыва; линии разграничения вида и разреза.
4.	Штриховая 	S/3...S/2	Линии невидимого контура; линии перехода невидимого контура.
5.	Штрих-пунктирная тонкая 	S/3...S/2	Линии осевые и центровые; линии сечений, являющиеся осями симметрии для наложенных или вынесенных сечений.
6.	Разомкнутая 	S...1,5S	Линии сечения
7.	Штрих-пунктирная тонкая с двумя точками 	S/3...S/2	Линии сгиба на развертках; линии для изображения частей изделий в крайних или промежуточных положениях; линии для изображения развертки, совмещенной с видом.
8.	Сплошная тонкая с изломами 	S/3...S/2	Длинные линии обрыва
9.	Штрих-пунктирная утолщенная 	S/2...2/3S	Линии, обозначающие поверхности, подлежащие термообработке или покрытию; линии для изображения элементов, расположенных перед секущей плоскостью («наложенная проекция»)

3. Штрих пунктирная тонкая линия. Если изображение симметрично, как, то на нем проводят ось симметрии. Для этой цели используют штрихпунктирную тонкую линию. Эта линия делит изображение на две одинаковые части. Она состоит из длинных тонких штрихов (длина их выбирается от 5 до 30 мм) и точек между ними. Вместо точек допускается чертить коротенькие штрихи – протяжки – длиной 1-2 мм. Расстояние между длинными штрихами от 3 до 5 мм. Толщина такой линии от $s/3$ до $s/2$.

Штрихпунктирную тонкую линию используют и для указания осей вращения, центра дуг окружностей. При этом положение центра должно определяться пересечением штрихов, а, а не точкой.

4. Сплошная тонкая линия. Толщина ее от $s/3$ до $s/2$. Она используется для проведения выносных и размерных линий.

5. Штрихпунктирная с двумя точками тонкая линия. При построении разверток используют штрихпунктирную с двумя точками тонкую линию для указания линии сгиба.

6. Сплошная волнистая линия. Ее используют в основном как линию обрыва в тех случаях, когда изображение дано на чертеже не полностью. Толщина такой линии от $s/3$ до $s/2$.

При выполнении учебных чертежей надо учитывать, что от правильного применения линий по их назначению, правильного выбора их толщин, качественного выполнения штриховых и штрих–пунктирных линий в большой мере зависит удобство пользования чертежом.

1.4 Шрифты чертежные

Все надписи на чертежах следует выполнять шрифтами, установленными ГОСТ 2.304 – 81* «Шрифты чертежные».

Шрифты различают по типам и размерам.

Размер шрифта ***h*** определяется высотой прописных (заглавных) букв в миллиметрах, измеряемой перпендикулярно к основанию строки.

Установлены следующие размеры шрифта: (1,8); 2,5; 3,5; 5; 7; 10; 14; 20; 28; 40. Применение шрифта размером 1,8 не рекомендуется.

Стандартом установлены два типа шрифта: А и Б. Тип шрифта определяет кратность толщины d линии букв размеру шрифта: для типа А: $d=(1/14)h$, для типа Б: $d=(1/10)h$. Шрифты могут быть выполнены без наклона или с наклоном около 75 градусов к основанию строки.

Толщина линии шрифта d определяется в зависимости от типа и высоты шрифта.

Ширина g буквы определяется по отношению к размеру шрифта h , например: $g=(6/10)h$, или по отношению к толщине линии шрифта d , например: $g=6d$. Шрифты в ГОСТ 2.304 – 81* выполнены по вспомогательной сетке, образованной вспомогательными линиями, в которую вписываются буквы. Шаг вспомогательных линий сетки определяется в зависимости от толщины линий шрифта d . Построение шрифта на вспомогательной сетке показано на рисунке 4.5.

Параметры шрифтов типа Б (до размера 20) приведены в таблице 4.3.

Таблица 4.3 – Параметры шрифтов

Параметры шрифта	Обозначение	Относительный размер	Размеры, мм						
Размер шрифта высота прописных букв	h	$10/10h$	2,5	3,5	5	7	10	14	20
Высота строчных букв		$7/10h$	1,8	2,5	3,5	5	7	10	14
Расстояние между буквами		$2/10h$	0,5	0,7	1,0	1,4	2,0	2,8	4,0
Минимальный шаг строк		$17/10h$	4,3	6,0	8,5	12,0	17,0	24,0	34,0
Минимальное расстояние между словами		$6/10h$	1,5	2,1	3,0	4,2	6,0	8,4	12,0
Толщина линий шрифта		$1/10h$	0,25	0,35	0,5	0,7	1,0	1,4	2,0
Ширина прописных: букв: Г,Е,З,С; А,Д,М,Х,Ц,Ы,Ю; Ж,Ф,Ш,Ъ; Щ;		$5/10h$ $7/10h$ $8/10h$	1,3 1,8 2	1,8 2,5 2,8	2,5 3,5 4	3,5 5 5,6	5,0 7 8	7,0 10 11,2	10 14 16

Б,В,И,К,Л,Н,О,П,Р,Т, У,Ч,Ь,Э,Я		9/10h	2,2	3,2	4,5	6,3	9	12,6	18
Ширина строчных букв:		6/10h	1,5	2,1	3	4,2	6	8,4	12
э,с									
а,м,ц,ъ,ы,ю		4/10h	1	1,4	2	2,8	4	5,6	8
ж,т,ф,ш		6/10h	1,5	2,1	3	4,2	6	8,4	12
щ		7/10h	1,8	2,5	3,5	5	7	10	14
б,в,г,д,е,и,к,л,н,о,п,р,у,		8/10h	2	2,8	4	5,6	8	11,2	16
х,ч,ь,э,я		5/10h	1,3	1,8	2,5	3,5	5	7	10
Ширина цифр:									
1		3/10h	0,7	1,1	1,5	2,1	3	4,2	6
4		6/10h	1,5	2,1	3	4,2	6	8,4	12
2,3,5,6,7,8,9,0		5/10h	1,3	1,8	2,5	3,5	5	7	10

А Б В Г Д Е Ж З И Й К

Л М Н О П Р С Т У Ф h

Х Ц Ч Щ Ъ Ы Ь Э Ю

Я а б в г д е ж з и й к

л м н о п р с т у ф х ц $3/10h$

ч щ ъ ы ь э ю я $4/10h$

1 2 3 4 5 6 7 8 9 0 75°

□ ∅ R ⊥ ≤ ≧

Построения геометрические

Рисунок 4.5 – Построение шрифта на вспомогательной сетке

2 Основные сведения о векторных редакторах

MS Visio 2010 – мощный графический редактор, предназначенный для быстрого и эффективного создания векторных графических изображений любой сложности. С помощью встроенных шаблонов, трафаретов и стандартных модулей предоставляется возможность создавать как простейшие схемы, так и очень сложные чертежи и организационные диаграммы.

MS Visio 2010 – это приложение для создания диаграмм и схем, помогающее визуализировать, исследовать и распространять сложные данные. В Visio сложные для понимания таблицы и текст можно преобразовать в наглядные доступные схемы.

MS Visio 2010 позволяет сохранять созданные файлы в различных графических форматах, таких как GIF (Graphic Interchange Format), JPEG (Joint Photographic Expert Group), чертеж Auto CAD и т.п.

Visio 2010 и более ранние версии Microsoft Visio поддерживают просмотр и сохранение диаграмм в форматах VSD и VDX. VSD является собственным бинарным файловым форматом, который используется во всех предыдущих версиях Visio. VDX является хорошо задокументированным XML "DatadiagramML" форматом. Начиная с версии Visio 2013, сохранение в формате VDX больше не поддерживается в пользу новых VSDX и VSDM файловых форматов. Visio 2010 и более ранние версии Microsoft Visio используют VSD формат как формат по умолчанию, Visio 2013 использует VSDX формат по умолчанию.

На вход редактора может поступать информация в самых разнообразных форматах. Вот некоторые из них:

Файловые форматы:

VSD – диаграмма или схема,

VSS – фигура,

VST – шаблон,

VDX – диаграмма в формате XML,
VSX – фигура XML,
VSS – фигура,
VST – шаблон,
VDX – диаграмма в формате XML,
VSX – фигура XML,
VTX – шаблон XML,
VSL – надстройка.
VSDX – OPC/XML диаграмма,
VSDM – OPC/XML диаграмма, содержащая макрос

2.1 Создание нового документа

При открытии программы Visio2010 предлагает выбрать тип создаваемого документа, фактически выбор определяет только то, какие шаблоны (наборы готовых элементов) будут предложены при этом для работы. Предположим мы собираемся начертить принципиальную схему, тогда в появившемся окне нажимаем: «Создать – Техника» на экране появятся варианты технических схем, выбираем «Принципиальная электротехническая схема» и нажимаем «Создать».

2.2 Выбор формата бумаги

Если необходимо построить документ определенного формата (например, А3), то рекомендуется выбрать размер бумаги. Для этого нажмите на вкладку Конструктор, нажмите Размер и выберите из списка нужный размер, поскольку для нашего примера формата А3 в списке нет, выбираем «Другие размеры страниц» (рисунок 4.6). Появится окно (рисунок 4.7) в котором необходимо выбрать один из стандартных размеров или задать свой собственный, указав ширину и высоту бумаги.

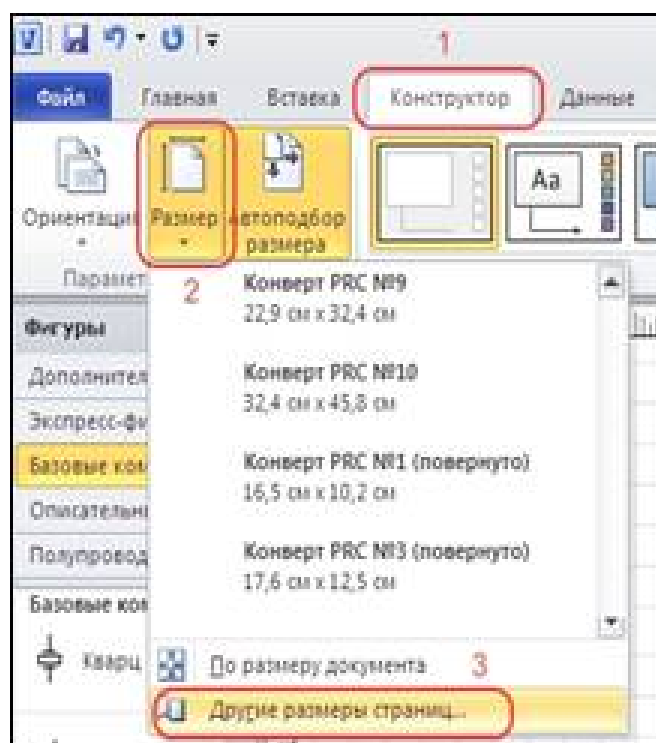


Рисунок 4.6 – Размер страниц

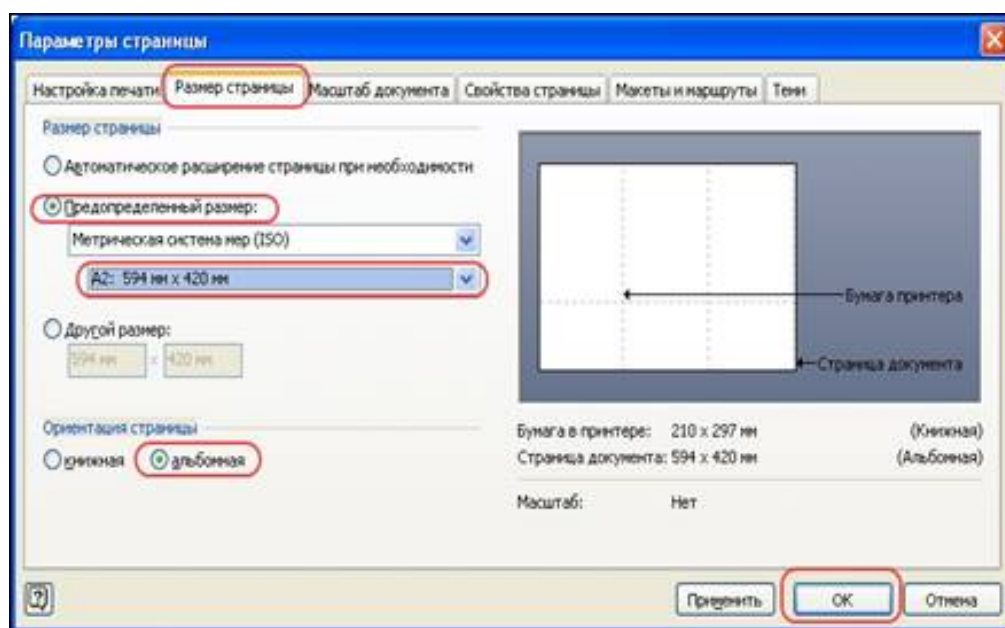


Рисунок 4.7 – Выбор формата страниц

2.3 Сохранение документа

Для сохранения схемы (пока пустой) нажмите «Файл → Сохранить». В появившемся окне выберите диск и папку, в которые необходимо сохранить

схему, введите имя схемы в поле «Имя файла». Нажмите кнопку «Сохранить».

2.4 Назначение кнопок программы

Меню программы имеет несколько вкладок, на рисунке 4.8 поясняется назначение кнопок вкладки «Главная».

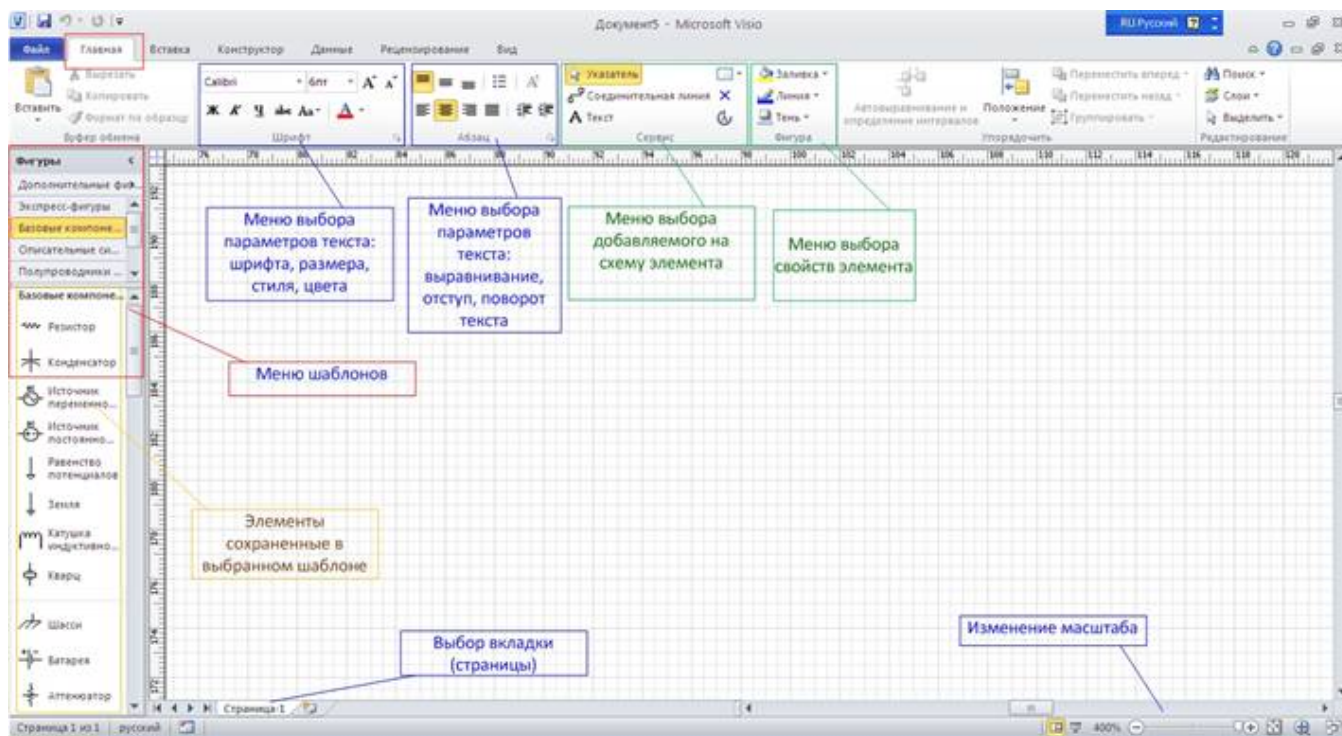


Рисунок 4.8 – Назначение кнопок программы

2.5 Изображение геометрических фигур, вставка текста

Для изображения геометрических фигур существует кнопка «Рисование», которая расположена на вкладке «Главная» в группе «Сервис». Эта панель содержит следующие инструменты (таблица 4.4).

Прежде, чем рисовать какой-нибудь сложный объект, разбейте его мысленно на элементарные составляющие и рисуйте именно их: прямоугольники, окружности или эллипсы, дуги, прямые линии, текст и др.

Рисовать объекты удобно, используя для точного позиционирования инструмент «Размер и положение», вызываемый через вкладку «Вид», меню «Области задач».

Таблица 4.4 – Команды панели рисования

	Rectangle Tool (активизировать инструмент «Прямоугольник», «Горячая клавиша» – [Ctrl+8]);
	Ellipse Tool (активизировать инструмент «Эллипс», «Горячая клавиша» – [Ctrl+9]);
	Line Tool (активизировать инструмент «Линия», «Горячая клавиша» – [Ctrl+6]);
	Arc Tool (активизировать инструмент «Дуга», «Горячая клавиша» – [Ctrl+7]);
	Freeform Tool (активизировать инструмент «Произвольная кривая», «Горячая клавиша» – [Ctrl+5]);
	Pencil Tool (активизировать инструмент «Карандаш», «Горячая клавиша» – [Ctrl+4]);

1.5.1 Прямоугольник (квадрат)

Для изображения на странице документа прямоугольника, нажмите кнопку «Прямоугольник», находящуюся на Панели инструментов «Рисование» (того же эффекта можно достичь при использовании «горячей» клавиши [Ctrl-8]). По этой команде указатель мыши превратится в перекрестие со значком прямоугольника внизу. Перемещая мышь, позиционируйте перекрестие в одну из вершин будущего прямоугольника (при этом текущие координаты перекрестия можно видеть в Строке состояния в виде: X = ... мм, Y = ... мм). Достигнув вершины, нажмите левую клавишу мыши и, удерживая её нажатой, перемещайте указатель мыши по диагонали к противоположной вершине прямоугольника (при этом текущую ширину и высоту будущего прямоугольника можно видеть в Строке состояния в виде надписи: Ширина = ... мм, Высота = ... мм). После отпускания левой клавиши мыши, построение прямоугольника будет завершено. Если при растягивании прямоугольника будет зажата клавиша [Shift], будет формироваться квадрат. Квадрат можно рисовать и без нажатой [Shift], ориентируясь на вспомогательную пунктирную диагональ, которую формирует Visio под углом 45° к линии горизонта, но это будет менее удобно. В любом случае мы увидим на экране пунктирный прямоугольник с зелёными квадратными маркерами выделения в характерных точках прямоугольника, а также рукоятку вращения прямоугольника вокруг центра

вращения (рисунок 4.9). Центр вращения станет видимым после позиционирования указателя мыши на зелёном кружочке, а вокруг кружочка появится круговая стрелка, сигнализирующая о том, что объект готов к вращению. Зажав левую клавишу мыши, можно осуществить вращение прямоугольника, как по часовой стрелке, так и против неё. При этом в Строке состояния можно будет считывать величину угла поворота в градусах. Кстати, центр вращения объекта можно перемещать с помощью мыши в любое место, даже за пределы объекта.

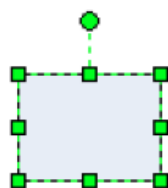


Рисунок 4.9 – Выделенный прямоугольник

Используя угловые маркеры, можно с помощью мыши осуществлять пропорциональные увеличения или уменьшения высоты и ширины прямоугольника одновременно, с помощью же серединных маркеров можно изменять что-то одно: либо высоту, либо ширину прямоугольника.

Visio позволяет осуществлять более сложные изменения выделенной фигуры с помощью кнопки «Карандаш», находящейся на Панели инструментов «Рисование». После нажатия этой кнопки происходит очередное изменение маркеров выделенной фигуры (см. рисунок 4.10).

Между каждой из двух соседних вершин прямоугольника находятся круглые зелёные маркеры – так называемые контрольные точки. При наведении на контрольную точку указателя мыши, последний превращается в крестообразную стрелку – признак захвата маркера. После этого, зажав левую клавишу мыши и перемещая контрольную точку, можно менять кривизну данного отрезка (рисунок 4.11).

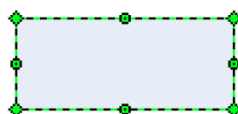


Рисунок 4.10 – Выделение прямоугольника при нажатой кнопке «Карандаш»

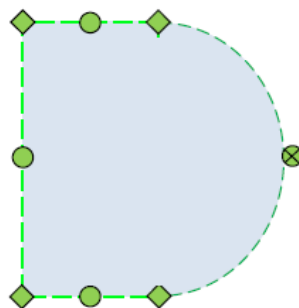


Рисунок 4.11 – Изменение кривизны одной из сторон прямоугольника

У фигуры можно изменять не только кривизну сторон, но и положение вершин. При захвате мышью маркера одной из вершин фигуры (они отмечены зелёными ромбами), указатель мыши трансформируется в крестообразную стрелку. Теперь маркер вершины можно перемещать в новое место. На рисунке 4.12 показана предыдущая фигура, у которой сделан перенос левой верхней вершины (она отмечена розовым цветом).

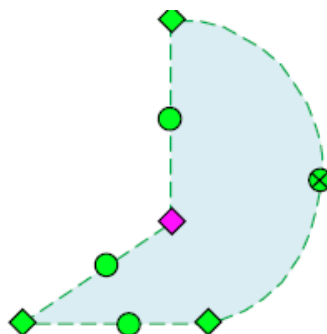


Рисунок 4.12 – Изменение местоположения одной из вершин фигуры

По умолчанию изображаемая фигура заливается белой краской. При необходимости выделенную фигуру можно залить другой краской или удалить заливку вообще (перейти, к так называемому, проволочному каркасу). Для этого следует вызвать меню кнопки «Цвет заливки», щёлкнув мышью по значку «▼», расположенному рядом с кнопкой справа. В открывшемся меню (см. рисунок 4.13) выберите подходящую краску или откажитесь от заливки вообще, щёлкнув по надписи «Нет заливки».

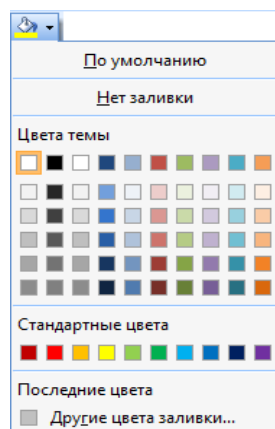


Рисунок 4.13 – Меню кнопки «Цвет заливки»

При необходимости, фигуру можно заштриховать. Для этого щёлкните правой клавишей мыши по контуру штрихуемой фигуры. В открывшемся контекстном меню выбранного объекта щёлкните мышью по пункту «Format ►», а в меню следующего уровня – по пункту «Fill...» (Заливка). Такого же эффекта можно добиться и с помощью строки меню, выбрав там пункты Format – Fill. В любом случае откроется одноимённое окно, рисунок 4.14.

Выполните необходимые настройки (в частности, в списке «Узор» найдите подходящий вариант заливки, а в списке «Цвет узора» нужный цвет штриховки), щёлкните по кнопке <ОК>. Выбранная фигура окажется заштрихованной. Необходимым условием успешной заливки или штриховки фигуры – контур такой фигуры должен быть замкнутым.

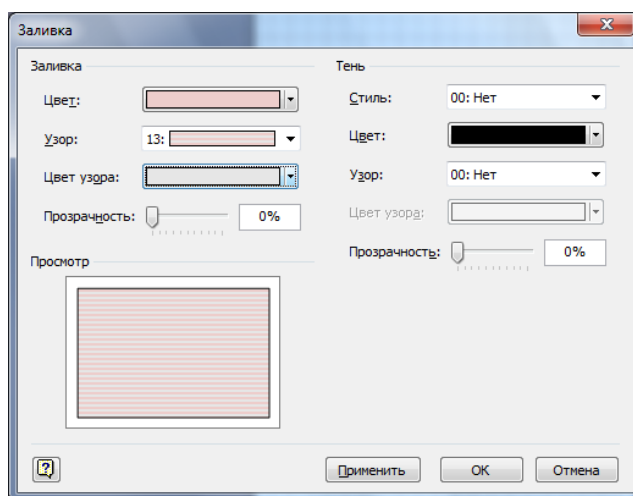


Рисунок 4.14 – Окно «Заливка»

Чтобы изменить цвет или толщину линии контура выбранного объекта, а также её тип (например, вам требуется вместо сплошной линии применить пунктирную), щёлкните правой клавишей мыши по контуру и в контекстном меню выберите пункт «Формат ►», а в меню следующего уровня – пункт «Линия...», либо тот же путь пройти, используя строку меню. В открывшемся окне «Линия» (см. рисунок 4.15) выберите подходящий образец типа линии (Шаблон), толщину и цвет.

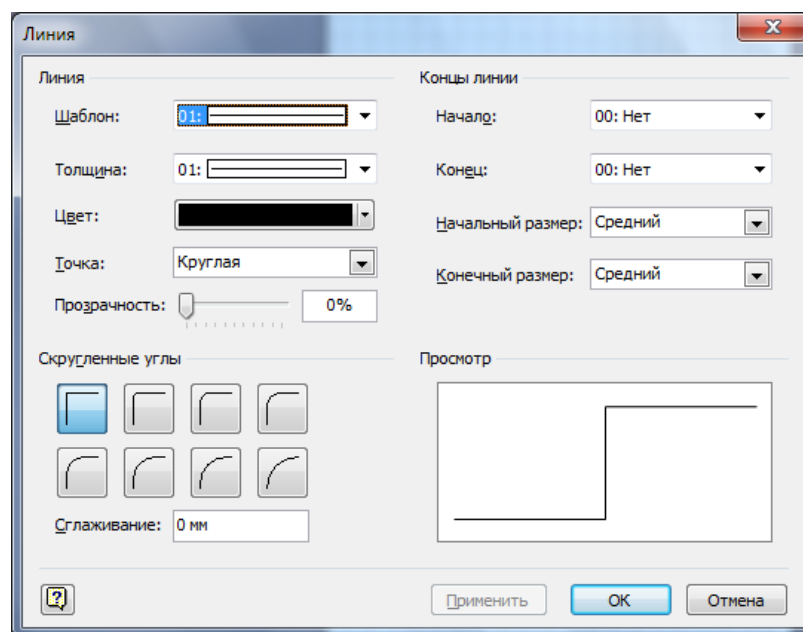


Рисунок 4.15 – Окно «Линия» для настройки параметров выбранной линии

Для снятия выделения любого объекта (объектов) удобно пользоваться клавишей [Esc], а для отказа от выбранного инструмента щёлкните мышью по кнопке <Указатель> или выберите другой подходящий для дальнейшей работы инструмент.

Все приёмы редактирования, только что описанные для фигуры «Прямоугольник», будут справедливы и для других фигур.

1.5.2 Эллипс (окружность)

Для изображения эллипса отыщите на Панели инструментов «Рисование» кнопку <Эллипс> и щёлкните по ней мышью. По этой команде указатель мыши превратится в перекрестие со значком эллипса внизу.

Перемещая мышь, позиционируйте перекрестие в одну из вершин будущего воображаемого прямоугольника, внутри которого будет вписан эллипс. Достигнув вершины, нажмите левую клавишу мыши и, удерживая её нажатой, перемещайте указатель мыши по диагонали к противоположной вершине воображаемого прямоугольника. После отпускания левой клавиши мыши, построение эллипса будет завершено. Если при построении эллипса будет утоплена клавиша [Shift], будет формироваться окружность.

1.5.3 Дуга

Для изображения дуги отыщите на Панели инструментов «Рисование», кнопку <Дуга> и щёлкните по ней мышью. По этой команде указатель мыши превратится в перекрестие со значком дуги внизу. Перемещая мышь, позиционируйте перекрестие в начало дуги. Далее, нажмите левую клавишу мыши и, удерживая её нажатой, перемещайте указатель мыши в конец будущей дуги. После отпускания левой клавиши мыши, построение дуги будет завершено.

Не забывайте, что для редактирования любого выделенного объекта, в том числе и дуги, удобно использовать такой инструмент, как <Карандаш>, находящейся на Панели инструментов «Рисование».

1.5.4 Прямая линия

Для изображения прямой линии нажмите кнопку <Line Tool>, находящуюся на Панели инструментов «Рисование» (выбрать эту кнопку можно и с помощью «горячей» клавиши [Ctrl-6]). По этой команде указатель мыши превратится в перекрестие со значком прямой линии внизу. Перемещая мышь, позиционируйте перекрестие в начало линии. Далее, нажмите левую клавишу мыши и, удерживая её нажатой, перемещайте указатель мыши в конец линии. Если при этом держать утопленной клавишу [Shift], то угол наклона линии будет кратен 45°. После отпускания левой клавиши мыши, построение линии будет завершено.

Для того чтобы оформить один или оба конца выделенной линии в виде стрелки, жирной точки и т.п., следует выбрать в Строке меню пункт «Format», а в выпавшем подменю – пункт «Line...». При этом откроется одноимённое окно.

Установите в разделе «Line ends» (Концы линии) нужную конфигурацию начала и конца линии, соответственно, в списках «Begin» и «End». При необходимости, в разделе «Line» (Линия) можно изменить тип, толщину и цвет линии, соответственно, в списках «Pattern», «Weight» и «Color». После завершения настроек, щёлкните по кнопке <OK>.

1.5.5 Контур, состоящий из ломаных прямых

Нажмите кнопку <Line Tool>. Выведите указатель мыши в начало первого предполагаемого отрезка прямой. Удерживая нажатой клавишу [Ctrl], нажмите левую клавишу мыши и переместите указатель мыши к концу отрезка. Продолжая удерживать клавишу [Ctrl], отпустите левую клавишу мыши и вновь её нажмите. Изобразите второй отрезок. Таким образом изобразите весь контур. Если контур должен быть замкнутым, конец последнего отрезка должен совпасть с началом первого. При необходимости замкнутый контур можно залить.

1.5.6 Контур в форме плавной кривой (сплайна)

Для изображения плавной кривой нажмите на Панели инструментов «Рисование», кнопку <Freeform Tool> (Произвольная кривая). Ту же команду можно подать и с помощью «горячей» клавиши [Ctrl-5]. По этой команде указатель мыши превратится в перекрестие со значком кривой внизу. Перемещая мышь, позиционируйте перекрестие в начало кривой. Далее, нажмите левую клавишу мыши и, удерживая её нажатой, перемещайте указатель мыши по траектории будущей кривой. После отпускания левой клавиши мыши, построение кривой будет завершено (рисунок 4.16). Кривизну начального (1-2) и конечного (7-8) участков кривой можно

изменять, вращая касательные вокруг конечных точек 1 и 8. Можно менять и амплитуду кривых на конечных участках, для этого нужно изменять с помощью мыши размер касательных (приближая или удаляя точку А по отношению к точке 1, а также приближая или удаляя точку В по отношению к точке 8).

Если требуется изобразить замкнутый контур, прорисовку кривой следует выполнять таким образом, чтобы её конечная и начальная точки совпали. Замкнутую кривую можно залить.

Любую полученную фигуру, будь она разомкнута или замкнута, можно отредактировать. Оставаясь в режиме формирования произвольной кривой, можно деформировать контур, используя для этой цели многочисленные узлы, расположенные вдоль всей кривой (см. рисунок 4.16). Ухватите указателем мыши нужный узел (признаком захвата узла будет трансформация указателя в крестообразную стрелку) и, нажав после этого левую клавишу мыши, переместите узел, а вместе с ним и близлежащий участок кривой, в новое место.

При необходимости на кривой можно удалить любой лишний узел. Для этого щёлкните мышью по такому узлу (он окрасится в розовый цвет) и нажмите клавишу [Delete]. Узел исчезнет, а кривая плавно выпрямится, ориентируясь на оставшиеся узлы.

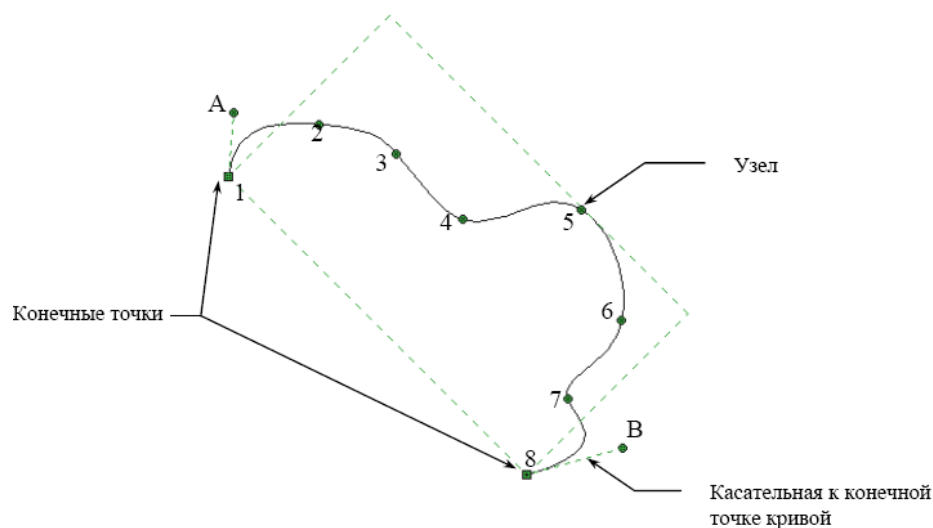


Рисунок 4.16 – Произвольная кривая

1.5.7 Вставка текста

Для вставки текста внутрь того или иного графического объекта (прямоугольника, эллипса и др.) выделите его и сразу приступайте к вводу текста с клавиатуры. При этом текст будет размещаться в специальном текстовом окне, размещённом над графическим объектом. Если длина фразы превысит ширину окна, то произойдёт автоматический перенос избыточного текста на новую строку. Переход к новой строке возможен и с помощью нажатия клавиши [Enter]. Текстовое окно может полностью скрыть графический объект, но после окончания набора текста и нажатия [Esc], содержимое текстового окна будет передано в центр выделенной фигуры.

Для создания поясняющих надписей, заголовков и других текстовых объектов в произвольном месте документа, служат кнопки <Text Tool> (Текст) и <Text Block Tool> (Абзац), находящиеся в одном кнопочном меню на Панели инструментов «Стандартная» («горячие» клавиши [Ctrl-2] и [Ctrl-Shift-4], соответственно). Последняя кнопка (<Text Block Tool>) используется для написания текста, который нужно будет повернуть на определённый угол. При выборе любой из этих кнопок, указатель мыши трансформируется в перекрестие со значком печатной страницы внизу. Растяните с помощью мыши прямоугольник нужных размеров, внутри которого будет располагаться текст. По умолчанию текстовый курсор окажется в центре этого прямоугольника, т.е. вводимый текст будет центрироваться как по горизонтали, так и по вертикали. При необходимости другого форматирования, можно использовать соответствующий инструмент на панели инструментов «Formatting» (Форматирование). Там же можно выбрать и другие параметры шрифта: требуемую гарнитуру шрифта с указанием особенностей его использования (Bold – жирный, Italic – наклонный, Underline – подчёркнутый), размер и цвет шрифта.

Полезно навесить над текстовым окном линейку. Для этого щёлкните правой клавишей мыши внутри текстового прямоугольника, появится

контекстное меню (рисунок 4.17), в нём щёлкните мышью слева от надписи «Text Ruler» (Текстовая линейка).



Рисунок 4.17 – Контекстное меню текстового окна

Перед набором текста, при необходимости, можно выставить параметры абзаца. Для этого служат соответствующие ползунки, размещённые на текстовой линейке (см. рисунок 4.18): для задания отступа первой строки, для задания левого и правого краёв абзаца, для синхронного перемещения двух левых ползунков.

Технология работы с ними и табулостопами такая же, как и в текстовом процессоре MS Word. Набор текста осуществляется так же, как и в Word, т.е. весь текст абзаца следует набирать как одну длинную строку, программа Visio самостоятельно будет осуществлять перевод курсора на новую строку. Нажатие клавиши [Enter] означает переход к новому абзацу. Для завершения набора текста нажмите клавишу [Esc] или нажмите кнопку <Pointer Tool> (Указатель объектов).

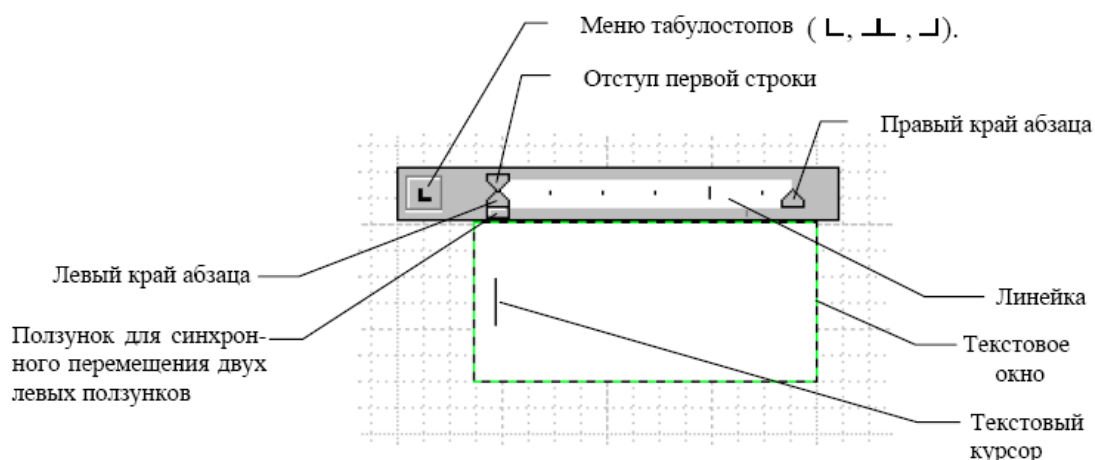


Рисунок 4.18 – Текстовое окно с линейкой

В некоторых случаях при работе с текстом может понадобиться панель инструментов форматирования текста, расположенная на главной вкладке. Например, требуется написать химическую формулу воды – H_2O . Напишите вначале формулу без индексов, т.е. $H2O$. Далее, выделите «2» и щёлкните по кнопке «Подстрочный» на панели «Формат текста» (это кнопка с надписью X_2). При необходимости формирования надстрочных, или верхних индексов (таких, например, как a^2), следует выбрать пункт «Надстрочный» на панели «Формат текста» (это кнопка с надписью X^2).

Для вставки текста внутрь графического объекта Visio (прямоугольника, эллипса и др.), нужно сделать по этому объекту двойной щелчок мышью. При этом внутри объекта появится мигающий текстовый курсор, приглашающий к набору текста.

1.6 Редактирование. Масштабирование

При стыковании линий не всегда удастся точно совместить концы линий, так в нашем случае получился выступ у тылового контакта (рисунок 4.19).

Чтобы исправить рисунок необходимо увеличить масштаб изображения. Максимальное значение бегунка масштаба 400%, но на самом деле можно ввести в поле с клавиатуры большее значение (максимум 1457%).

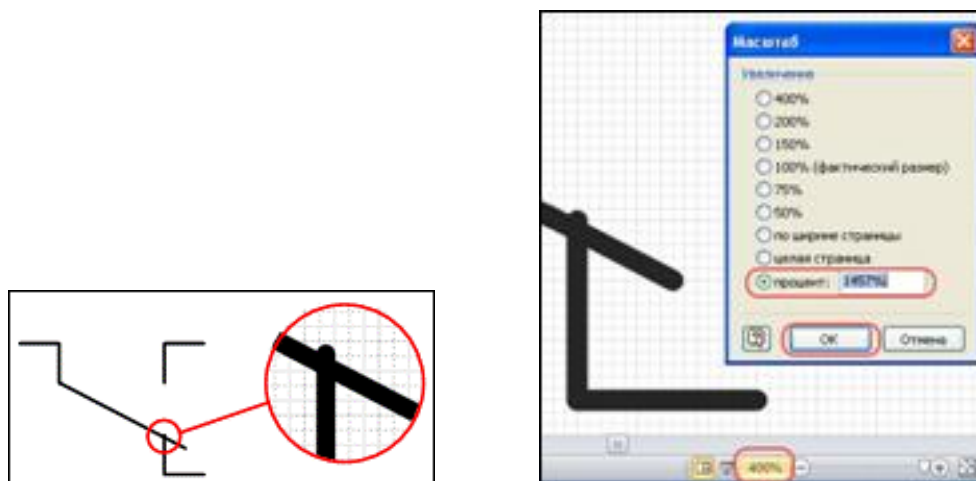


Рисунок 4.19 – Изменение масштаба

Еще проще изменить масштаб, используя колесо мыши при нажатой клавише Ctrl. После увеличения изображения делаем щелчок мыши на линии, которую необходимо изменить и, потянув за точку вниз, делаем высоту линии меньше (рисунок 4.20).

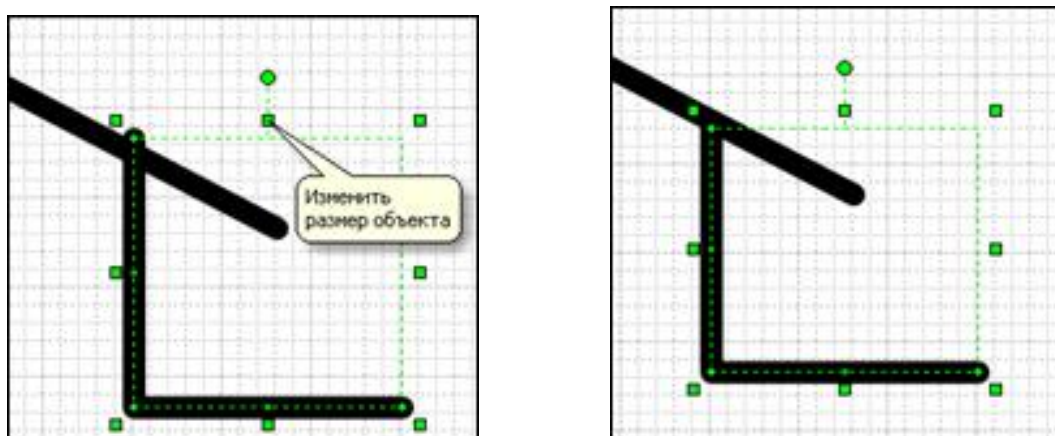


Рисунок 4.20 – Изменение масштаба с помощью колеса мышки

1.7 Сохранение чертежа в другом формате

Может потребоваться сохранить созданный рисунок в другом формате, например для сохранения чертежа в виде картинки (растрового изображения) выделим весь чертеж (Ctrl + A) и нажмем «Файл → Сохранить как». В появившемся окне выберем папку в которую необходимо сохранить картинку, зададим имя файлу, выберем тип файла Формат JPEG и нажмем кнопку «Сохранить». В окне параметров вывода выберем максимальное качество и разрешение для печати, нажмем кнопку ОК.

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с

диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения. При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Задание

1. Откройте и изучите стандарт ГОСТ- 2.104-68 «Основные надписи».
2. Создайте в программе MS Visio два листа: формата А4 и А0. Начертите и заполните на них рамку, основную надпись и дополнительные графы согласно стандартам ЕСКД.
3. Сохраните созданную рамку в векторном и растровом форматах.
4. Создайте рамку к текстовым документам.

Содержание отчета

Созданные документы согласно задания.

Контрольные вопросы

1. К какому классу редакторов относится Visio?
2. Можно ли прямоугольник трансформировать в овал?
3. Как изобразить контур, состоящий из ломанных прямых?
4. Как изобразить контур в форме плавной кривой (сплайна)?
5. Как сделать заливку замкнутой фигуры?
6. Для чего нужно окно «Размер и положение»? Как его открыть?

Лабораторная работа №5. Применение на практике возможностей векторных редакторов для формирования электрических схем

Цель работы: изучение расширенных возможностей векторных редакторов по созданию графических элементов.

Теоретическая часть

1 Расширенные возможности Visio

В предыдущей лабораторной работе были рассмотрены простейшие графические примитивы (линии, прямоугольники, овалы). Однако графический редактор MS Visio обладает широким спектром инструментов, позволяющим из простейших примитивов создавать более сложные графические объекты. К таким инструментам относятся: группировка объектов, объединение, соединение, вычитание объектов и другие. Рассмотрим основные из них.

1.1 Конструктор фигур

Одним из способов создания новых фигур является возможность производить над стандартными фигурами различные операции. Для применения той или иной операции следует выделить необходимые фигуры, открыть вкладку «Разработчик», в группе «конструктор форм» нажать на кнопку «Операции» и в открывшемся меню выбрать одну из команд.

С помощью команды «Объединение» можно соединить несколько фигур в одну. При этом фигуры не обязательно должны пересекаться. Если же фигуры пересекаются, то они объединяются по периметру (рисунок 5.1).

Команда «Объединить» позволяет выполнить комбинирование выделенных фигур, то есть объединить непересекающиеся фрагменты и удалить общие.

Результатом действия команды является одна фигура, состоящая из соответствующих фрагментов исходных фигур (рисунок 5.2).

Команда «Фрагмент» разбивает фигуру на фрагменты по перекрывающимся областям или линиям пересечения. После деления каждый фрагмент становится самостоятельной фигурой (рисунок 5.3).

После выполнения команды «Пересечение» на листе остается фигура, образованная общими фрагментами (рисунок 5.4).

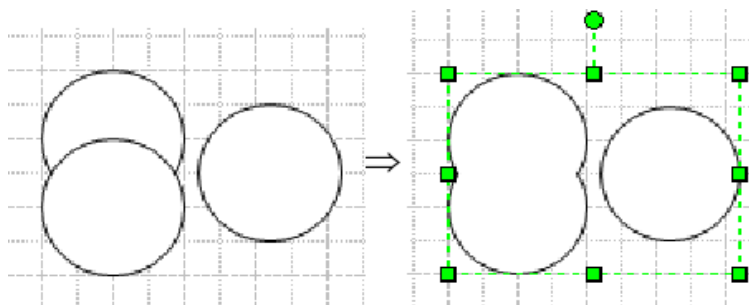


Рисунок 5.1 – Создание новой фигуры путем объединения нескольких фигур

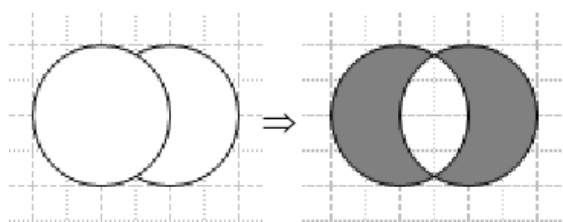


Рисунок 5.2 – Комбинирование фигур с помощью команды «Объединить»

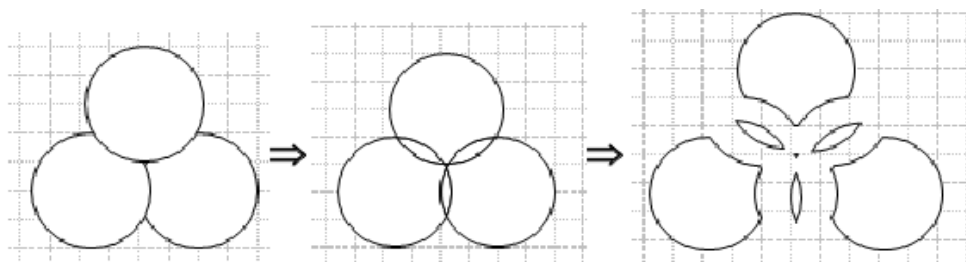


Рисунок 5.3 – Разделение фигур на фрагменты с помощью команды «Фрагмент»

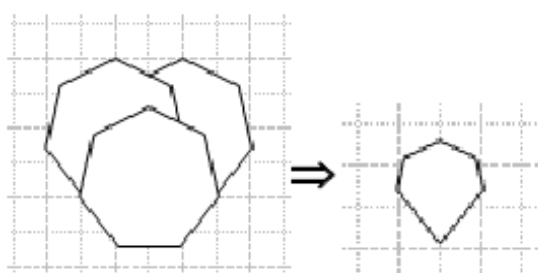


Рисунок 5.4 – Получение фигуры с помощью команды «Пересечение»

В результате применения команды «Вычитание» происходит удаление всех выбранных фигур, кроме нижней левой. Таким образом новую фигуру можно получить, вырезав из одной фигуры другую (рисунок 5.5).

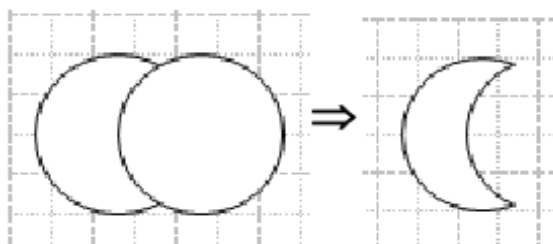


Рисунок 5.5 – Получение фигуры с помощью команды «Вычитание»

В отличие от объединения, при соединении фигур они не сливаются, а объединяются, дополняя друг друга. При этом от фигур, принимающих участие в соединении, остается только одна, состоящая из контурных линий всех фигур (рисунок 5.6). Следует отметить, что при этом сохраняется возможность редактировать соединенные фигуры с помощью инструмента «Карандаш».

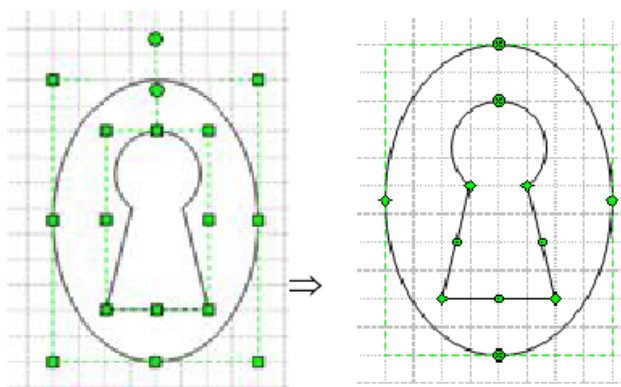


Рисунок 5.6 – Получение одной фигуры из двух путем соединения

С помощью команды «Обрезать» замкнутый контур любой фигуры можно сделать разомкнутым. Также с помощью этой команды можно «отрезать» часть контура, чтобы удалить его или использовать в дальнейшем, например при создании пользовательских фигур (рисунок 5.7).

С помощью команды «Смещение» можно «размножить» выделенную фигуру, создав копии левее и правее ее текущего положения. Расстояние до копий определяется значением, указанным в окне диалога «Смещение». На

рисунке 5.8 показан результат применения данной операции с величиной смещения 2 мм.

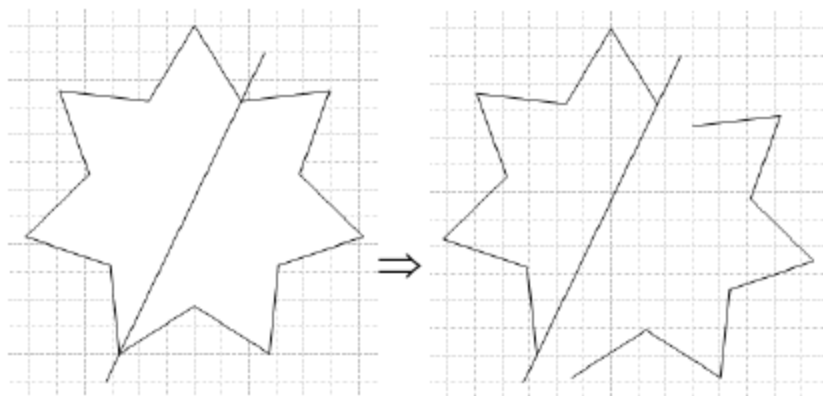


Рисунок 5.7 – Отсечение части контура фигуры с помощью команды «Обрезать»

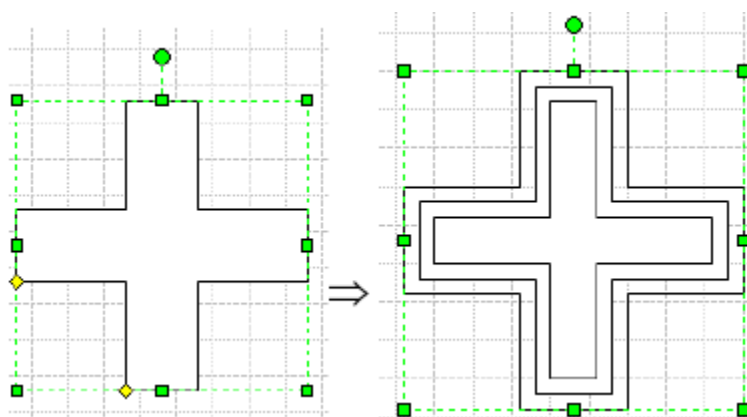


Рисунок 5.8 – Размножение фигуры с помощью команды «Смещение»

1.2 Группировка фигур

Группировка фигур – удобное средство, позволяющее управлять поведением группы фигур. Число фигур, входящих в группу ничем не ограничено. В отличие от объединенных, фигуры, подвергшиеся группировке, сохраняют свои свойства. Сгруппированные фигуры теряют свою самостоятельность, т.е. все фигуры в группе перемещаются, вращаются и изменяют свои размеры одновременно.

Группировка фигур полезна для объединения готовых фрагментов рисунка, которые необходимо защитить от случайных изменений.

Объекты, входящие в группу, по прежнему поддаются

индивидуальному редактированию. При этом необходимо выбрать сначала сам сгруппированный объект, а потом повторно нажать на линию или прямоугольник, который необходимо отредактировать.

Более удобным способом изменения фигур в группе и добавления в нее новых является открытие специального окна редактирования группы. Для этого следует выделить необходимую группу и выбрать через контекстное меню команду «Группировать→ Открыть группу». После этого откроется новое окно. Например для группы объектов на листе рисунка «Документ1» откроется окно редактирования группы «Документ1:Sheet.1<ГРУППА>». Лист редактирования группы и лист соответствующего рисунка интерактивно связаны между собой. Это означает, что все изменения, внесенные на одном листе, сразу отображаются на другом. Таким образом, после внесения необходимых изменений окно листа редактирования можно закрыть – все внесенные изменения будут автоматически сохранены на листе рисунка.

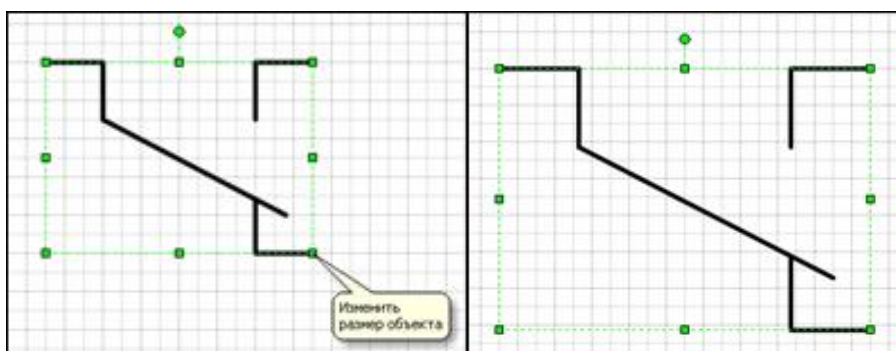


Рисунок 5.9 – Редактирование сгруппированных объектов

Кроме того, объекты, входящие в группу, поддаются также групповому редактированию. При редактировании сгруппированного объекта, если тянуть элемент за середину прямоугольника меняется его размер только по одной оси (X или Y). Если же тянуть за угол изменяется размер сразу по обеим осям (увеличивается размер объекта с соблюдением пропорций) (рисунок 5.9).

Для того чтобы разгруппировать объект необходимо выделить его, нажать правую кнопку и выбрать в меню последовательно «Группировать – Разгруппировать».

1.3 Другие операции над объектами

Чтобы зеркально отразить выделенный элемент в «Главном меню» выберем меню «Положение → Повернуть фигуры → Отразить слева направо» (рисунок 5.10), фигура отразится. В этом же меню можно отразить фигуру по вертикали или повернуть на 90 градусов.

Повернуть объект на определенный угол можно, нажав левую кнопку на точке над объектом (рисунок 5.11) и не отпуская, сместив указатель мыши относительно центра фигуры.

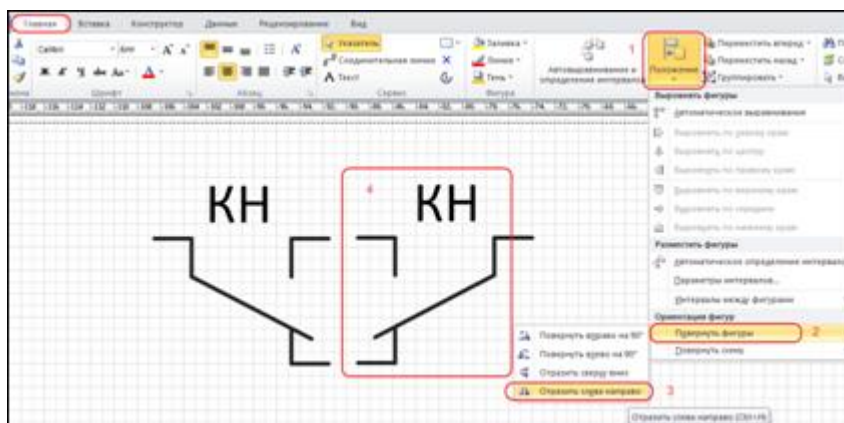


Рисунок 5.10 – Отражение объекта

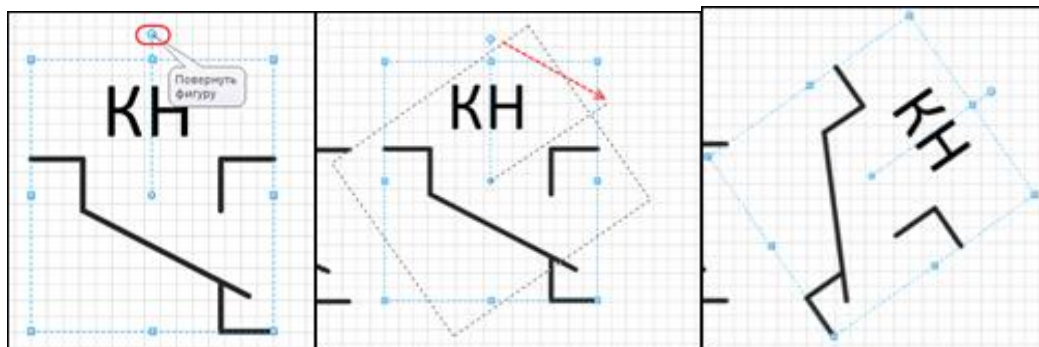


Рисунок 5.11 – Поворот объекта

Для перемещения элемента (группы элементов) по листу, необходимо выделить объект (группу объектов), а затем переместить элемент (группу элементов) мышью в нужное место. Для точного перемещения можно использовать курсорные клавиши клавиатуры. Если при нажатии курсорных клавиш перемещается не объект, а двигается лист, необходимо нажать клавишу «Scroll Lock» (над курсорными клавишами).

Для перемещения одновременно всех элементов чертежа нажмите сочетание клавиш Ctrl+A (английская), выделятся все элементы на листе.

Для задания точных координат объектов при перемещении или точного угла поворота объектов, можно воспользоваться окном «Размер и положение» во вкладке «Вид → Область задач».

Очень часто бывает необходимо создать связанные объекты, при смещении одного из которых, связь между ними не терялась бы. Это возможно только в том случае, если соединяемые фигуры имели точки соединения.

Для добавления на элемент точки соединения, необходимо выбрать инструмент «Точка соединения» в главном меню (рисунок 5.12).

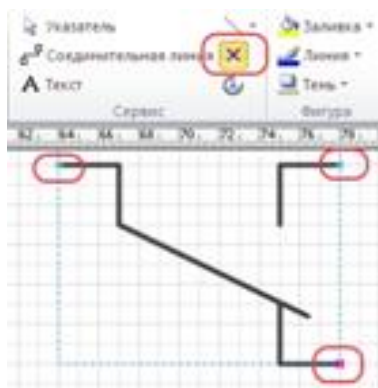


Рисунок 5.12 – Инструмент «Точка соединения»

Наведите курсор мыши на точку фигуры, к которой может выполняться соединение, и, удерживая нажатой кнопку Ctrl, сделайте щелчок левой кнопкой мыши. На изображении появится крестик (рисунок 5.12). Для представленного на рисунке в качестве примера контакта реле это будут три точки: общий, фронтальной и тыловой контакты.

Можно выделить три способа соединения объектов на чертеже между собой:

- 1 способ: удлинить черту одного из элементов;
- 2 способ: использовать фигуру «Линия»;
- 3 способ: использовать элемент «Соединительная линия».

В первом случае необходимо выбрать линию одного из элементов и

удлинить ее до другого элемента. Во втором случае необходимо выбрать инструмент «Линия» и провести линию от одного элемента до другого. Третий способ аналогичен второму, но выбирать нужно инструмент «Соединительная линия». Второй способ от третьего отличается тем, как отреагирует чертеж на сдвиг одного из соединенных элементов.

При наведении курсора на точку соединения элемента будет появляться красный прямоугольник. Если при соединении объектов концы линии или соединителя попадут в такие точки, то элементы не потеряют связи при смещении одного из них – линия самостоятельно удлинится (укоротится) и изменится ее угол, а соединитель изменит длину или на нем появится изгиб под прямым углом. При построении принципиальных схем второй способ соединения предпочтительнее (рисунок 5.13).

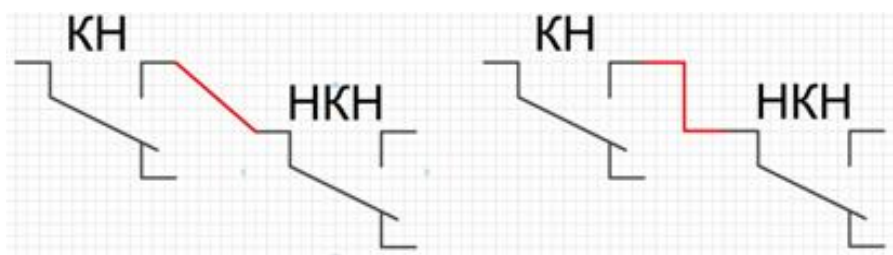


Рисунок 5.13 – Соединение элементов с помощью прямой линии (слева) и с помощью соединительной линии «справа»

2 Слои

Слой является именованной категорией фигур. Слои можно использовать для систематизации связанных фигур на странице документа. Назначая фигуры различным слоям, можно выборочно просматривать, печатать, назначать цвет и блокировать разные категории фигур, а также управлять привязкой и приклеиванием к фигурам в слое.

Например, при составлении макета офиса фигуры стен, дверей и окон должны быть назначены одному слою, фигуры электрических розеток – другому, а фигуры мебели – третьему. Поэтому при работе с фигурами в электрической системе другие слои можно заблокировать, чтобы случайно не

изменить расположение фигур стен или мебели. Каждая страница документа может содержать собственный набор слоев.

2.1 Добавление слоя

Можно добавить новые слои для систематизации пользовательских категорий фигур, а затем назначить фигуры этим слоям.

1. На вкладке Главная в группе Правка нажмите кнопку Слои и выберите пункт «Свойства слоя».

2. В диалоговом окне «Свойства слоя» нажмите кнопку «Создать».

3. Введите имя слоя, а затем нажмите кнопку «ОК».

4. В строке, соответствующей новому слою, установите флажки в каждом столбце для свойств, которые нужно назначить слою, если они еще не выбраны.

При копировании фигуры, назначенной слою, с одной страницы на другую в пределах одного документа или из одного документа в другой слой будет добавлен на новую страницу. Если на странице уже есть слой с таким же именем, фигура будет добавлена к существующему слою.

2.2 Назначение фигуры слою

Фигуру можно назначить нескольким слоям или не назначать вообще. Большинство фигур уже назначены слоям, поэтому при их размещении на страницу автоматически добавляется соответствующий слой.

1. Выделите фигуру.

2. На вкладке «Главная» в группе Правка нажмите кнопку «Слои» и выберите пункт «Назначить слою».

3. В диалоговом окне «Слой» щелкните слой, которому нужно назначить фигуру.

Для назначения фигуры нескольким слоям щелкните каждый слой, удерживая нажатой клавишу CTRL.

2.3 Активация одного или нескольких слоев

Активация слоя является быстрым способом назначения фигур слою при их добавлении на страницу. Если фигура еще не назначена слою, при добавлении она автоматически назначается активному слою.

Например, если требуется добавить фигуры электропроводки в документ с макетом офиса, можно активировать слой электрики. После этого все добавляемые фигуры будут назначаться слою электрики. Если требуется добавить фигуры окон, в качестве активного слоя можно указать слой стен.

Можно указать несколько активных слоев. Фигуры, добавленные на страницу, автоматически назначаются всем активным слоям.

1. На вкладке Главная в группе Правка нажмите кнопку Слои и выберите пункт Свойства слоя.

2. В диалоговом окне Свойства слоя установите флажок в столбце Активный для каждого слоя, который требуется сделать активным.

Слой или слои будут сделаны активными на текущей странице.

Слой, изменение которого заблокировано, активировать нельзя.

2.4 Переименование слоя

1. На вкладке «Главная» в группе «Правка» нажмите кнопку «Слои» и выберите пункт «Свойства слоя».

2. В диалоговом окне «Свойства слоя» выберите слой, который нужно переименовать, и выполните команду «Переименовать».

3. Введите новое имя, а затем два раза нажмите кнопку «ОК».

Слой будет переименован на текущей странице. Фигуры слоя не будут перемещены или изменены.

2.5 Удаление слоя

Для удаления слоя следует:

1. На вкладке «Главная» в группе «Правка» нажмите кнопку «Слои» и

выберите пункт «Свойства слоя».

2. В диалоговом окне «Свойства слоя» проверьте, не назначены ли данному слою фигуры.

3. Если в слое есть фигуры, назначьте их другому слою, выполнив указанные ниже действия.

4. Чтобы закрыть диалоговое окно «Свойства слоя», нажмите кнопку «ОК».

5. Выберите фигуры, которые требуется переназначить.

6. На вкладке «Главная» в группе «Правка» нажмите кнопку «Слои» и выберите команду «Назначить слою».

7. В диалоговом окне «Слой» щелкните слой, которому требуется назначить фигуры.

8. Нажмите кнопку «ОК».

9. В группе «Правка» нажмите кнопку «Слои» и снова выберите пункт «Свойства слоя».

10. В диалоговом окне «Свойства слоя» выберите слой, который требуется удалить, а затем нажмите кнопку «Удалить».

Чтобы удалить все неиспользуемые слои, установите флажок «Удалить неиспользуемые слои» в диалоговом окне «Свойства слоя», а затем нажмите кнопку «ОК».

2.6 Отображение или скрытие слоя

Для отображения или скрытия слоя следует:

1. На вкладке «Главная» в группе «Правка» нажмите кнопку «Слои». Выберите пункт «Свойства слоя».

2. В диалоговом окне «Свойства слоя» снимите или установите флажок в столбце «Видимость» для каждого слоя, который нужно скрыть или отобразить.

3 Наборы элементов (трафареты) MS Visio

Помимо простейших фигур (линии, прямоугольники, эллипсы и др.), «MS Visio» располагает множеством других, более сложных фигур. С целью удобства поиска нужной, все они разбиты по тематическим категориям, а те, в свою очередь, по типам. Для выбора нужной категории необходимо в окне «Фигуры» (вкладка «Вид», меню «Области задач») выбрать меню «Дополнительные фигуры» (рисунок ...) и выбрать из списка интересующую категорию.

Например, можно выбрать категорию «Блок-схема», а в ней – набор элементов «Разные фигуры». Выбранный тип представлен несколькими окнами трафаретов: «Стрелки», «Фоновые рисунки», «Фигуры простой блок-схемы» и «Рамки и заголовки» и другие. Из них лишь одно может быть активным в текущий момент времени, остальные окна свёрнуты. Чтобы раскрыть нужное окно, следует щёлкнуть по его заголовку.

Чтобы перенести нужный трафарет из окна «Фигуры» на страницу документа, следует установить на него указатель мыши и, зажав левую клавишу мыши, «перетащить» объект в нужное место документа.

Размещённый на странице документа трафаретный объект, при необходимости, можно доработать.

Для выбора другой категории трафаретов следует воспользоваться кнопкой «Фигуры», находящейся на панели «Стандартная» (рисунок 5.14).

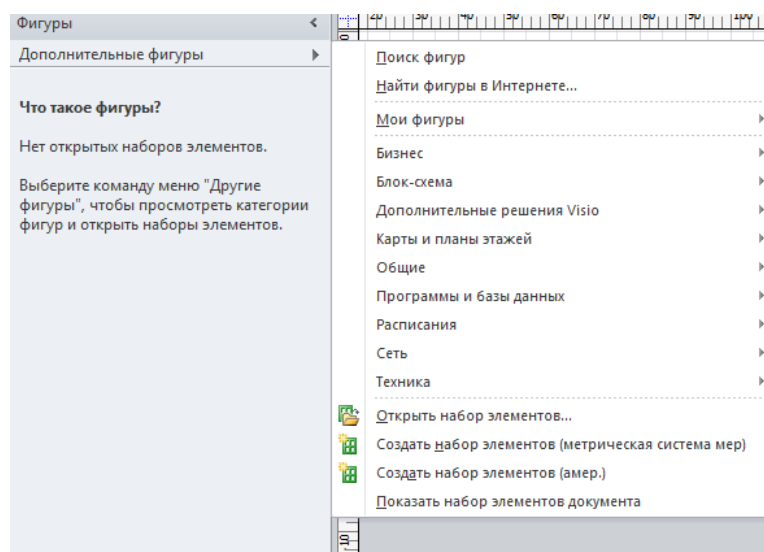


Рисунок 5.14 – Дополнительные фигуры MS Visio

3.1 Создание новых трафаретов

К сожалению, программный комплекс MS Visio не имеет стандартный набор элементов, позволяющий выполнять схемы электрических сетей по стандартам ЕСКД. Однако он позволяет создать свой набор элементов, который в дальнейшем можно многократно использовать.

Предположим, вы решили использовать MS Visio для изображения электрических принципиальных схем (ЭЗ). Для создания соответствующего набора элементов щёлкните по клавише «Дополнительные фигуры» на панели инструментов «Фигуры», в выпавшем меню выберите пункт «Создать набор элементов». При этом откроется новое и пока пустое окно трафаретов «Набор элементов2» (рисунок 5.15). Звёздочка слева на заголовке окна «Набор элементов2» говорит о том, что это окно находится в режиме редактирования.

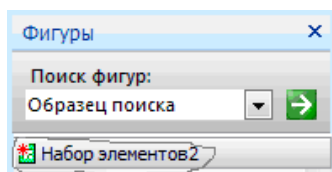


Рисунок 5.15 – Окно трафаретов «Набор элементов2»

Безликое имя «Набор элементов2» совершенно не говорит о содержании окна. Изменим его на более понятное – ЭЗ. Для этого щелчком правой клавиши мыши по заголовку окна «Набор элементов2» вызовите его контекстное меню, в котором выберите пункт «Свойства». При этом откроется одноимённое окно (рисунок 5.16).

В текстовом поле «Название» введите «ЭЗ», щёлкните по кнопке <ОК>. Изобразите справа (на Странице рисунка) резистор – прямоугольник размером 4x10 мм. При задании требуемых размеров, можно ориентироваться на линейки и на подсказку о размерах объекта в Строке состояния.

Итак, на странице рисунка мы сформировали условное графическое обозначение резистора с размерами по ЕСКД. Выделите его и переместите с помощью мыши в пока ещё пустое окно трафаретов «Э3». Там появится пиктограмма резистора с именем «Master.0».

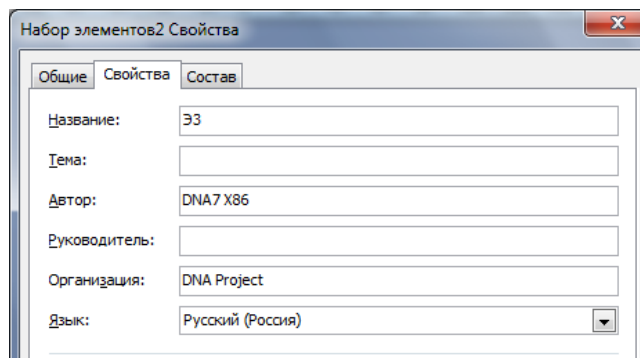


Рисунок 5.16 – Окно «Свойства» трафарета

Это имя не подходит для резистора. Щелчком правой клавиши мыши по пиктограмме резистора вызовите его контекстное меню и выберите в нём пункт «Переименовать образец». В строке «Master.0» появится текстовый курсор, приглашая к редактированию надписи. Введите слово «Резистор», нажмите [Enter]. Произойдёт нужная замена имени трафарета.

Таким образом, можно создать в окне «Э3» столько трафаретов, сколько нужно (рисунок 5.17).

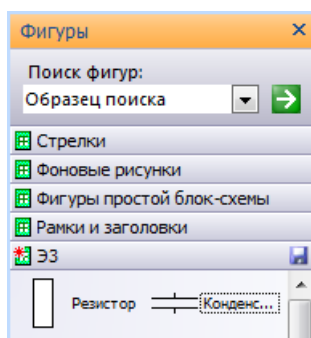


Рисунок 5.17 – Окно трафаретов «Э3»

Чтобы сохранить на диске содержимое окна трафаретов «Э3», щёлкните мышью по значку дискеты, находящегося в правой стороне заголовка окна «Э3» (см. рисунок 5.17). В открывшемся окне «Сохранить набор элементов») укажите имя файла, тип этому файлу Visio 2007 присвоит сам (*.vss). Чтобы снять красную звёздочку со строки заголовка окна

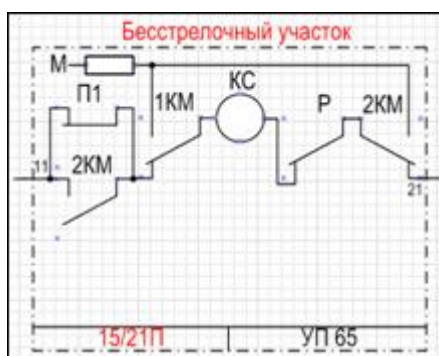
трафаретов, щёлкните по ней мышью. В открывшемся меню щёлкните по пункту «Изменить набор элементов». Окно выйдет из режима редактирования (звёздочка исчезнет).

Теперь, чтобы вы в следующий раз смогли получить доступ к этому окну трафаретов, после запуска Visio 2007 щёлкните по кнопке «Фигуры», находящейся на панели «Стандартная». В открывшемся меню выберите пункт «Мои фигуры» и в меню следующего уровня выберите свою библиотеку.

3.2 Построение больших схем

Используя элементы набора можно составить более сложные схемы, например блоки ЭЦ (электрических цепей), из которых в свою очередь можно создать набор элементов (рисунок 5.18).

Теперь для того чтобы построить какую-нибудь электрическую схему достаточно достать из базы нужные блоки, расставить их в правильном порядке (1,2,3) и соединить выводы блоков между собой (4, 5) (рисунок 5.19). Остается подписать конкретные имена блоков (вместо 15/21П, М17 и т. д.). Для того чтобы сделать построение схем более комфортным можно сделать временные подсказки в блоках используя цвет. Можно пометить цветом имена блоков, чтобы не забыть их заменить.



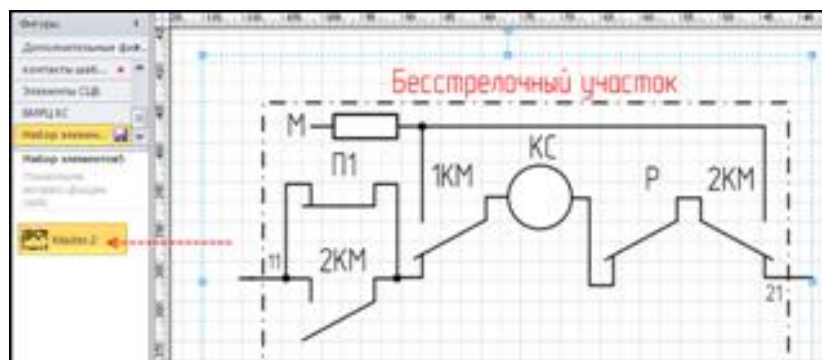


Рисунок 5.18 – Схема из набора элементов

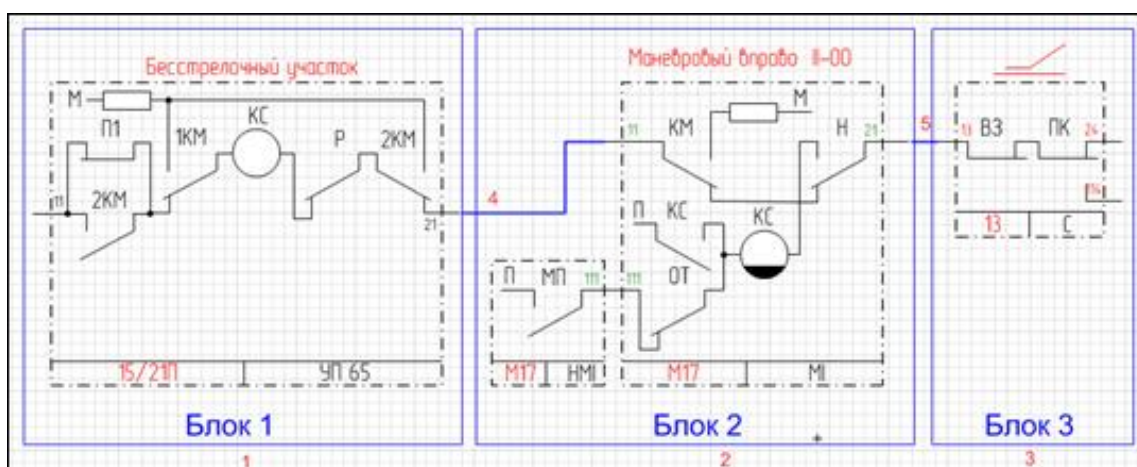


Рисунок 5.19 – Составление схемы из блоков

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения. При защите лабораторной работы студент отвечает

на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Задание

1. Изучите структуру и основные документы, входящие в ЕСКД.
2. Согласно Вашему варианту, создайте трафарет условных графических обозначений по ЕСКД.

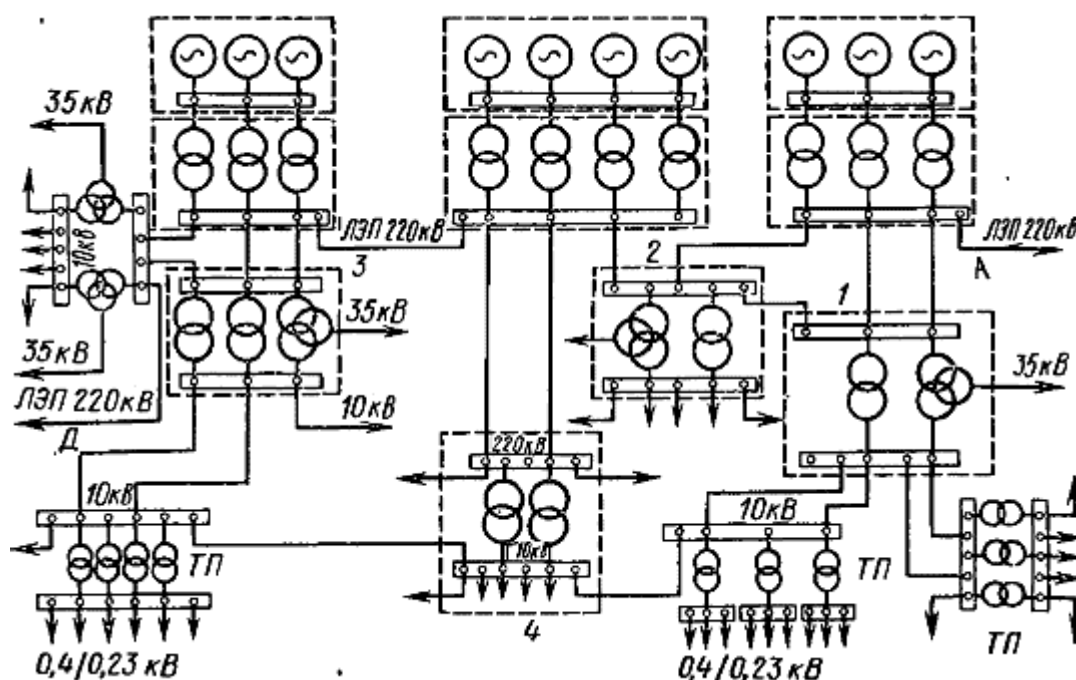
Таблица 5.1 – Варианты заданий

Вариант	ГОСТ	Вариант	ГОСТ
1	ГОСТ 2.722-68 ЕСКД	11	ГОСТ 2.756-76 ЕСКД
2	ГОСТ 2.723-68 ЕСКД	12	ГОСТ 2.757-81 ЕСКД
3	ГОСТ 2.728-74 ЕСКД	13	ГОСТ 2.767-89 ЕСКД
4	ГОСТ 2.767-89 ЕСКД	14	ГОСТ 2.768-90 ЕСКД
5	ГОСТ 2.725-68 ЕСКД	15	ГОСТ 2.736-68 ЕСКД
6	ГОСТ 2.726-68 ЕСКД		
7	ГОСТ 2.727-68 ЕСКД		
8	ГОСТ 2.730-73 ЕСКД		
9	ГОСТ 2.745-68 ЕСКД		

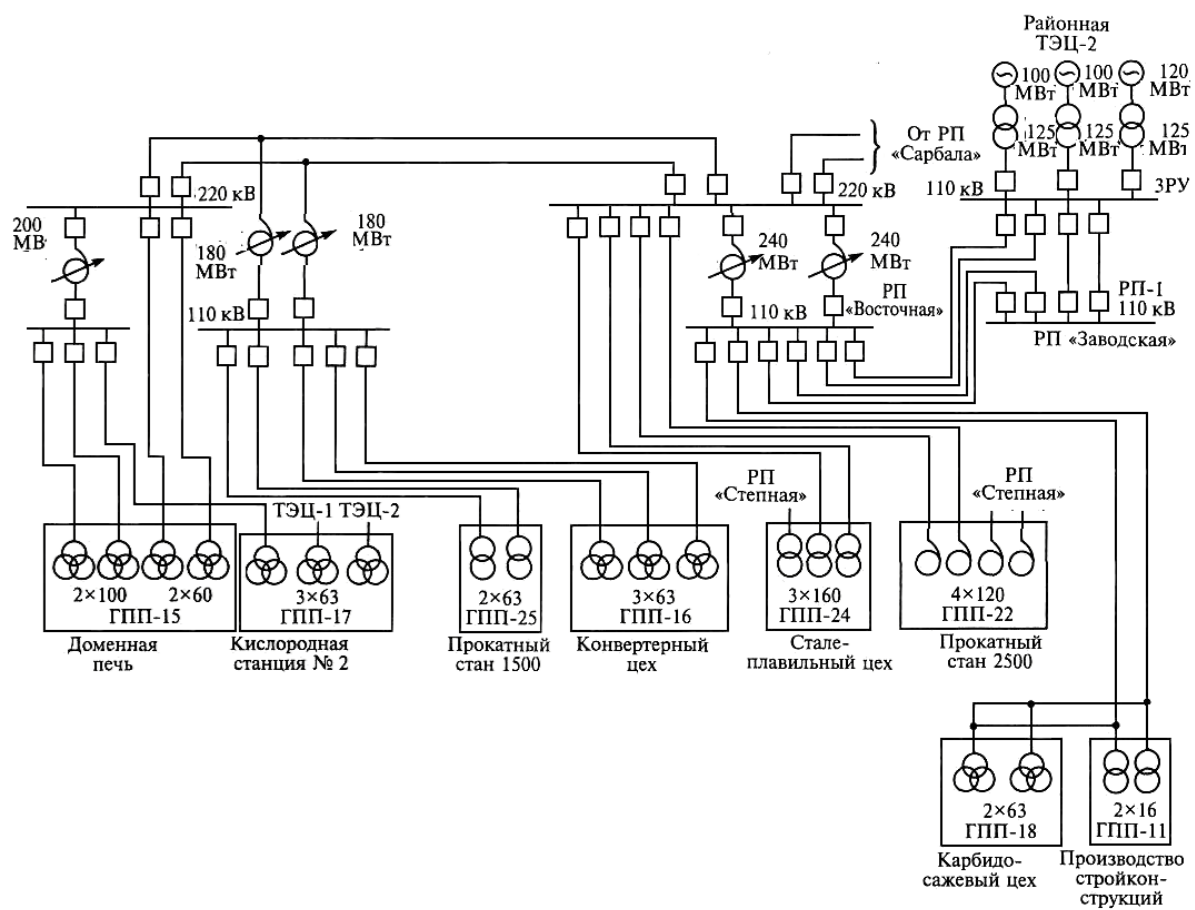
10	ГОСТ 2.752-71 ЕСКД		
----	--------------------	--	--

3. Создайте трафарет «Рамки» и перенесите в него рамки, созданные в предыдущей лабораторной работе для книжной и альбомной ориентации. Сохраните выбранный трафарет. Создайте новый документ, создайте слой «Рамка» и вставьте в него соответствующую рамку из сохраненного трафарета. Заблокируйте слой «Рамка» для предотвращения ее случайного изменения.

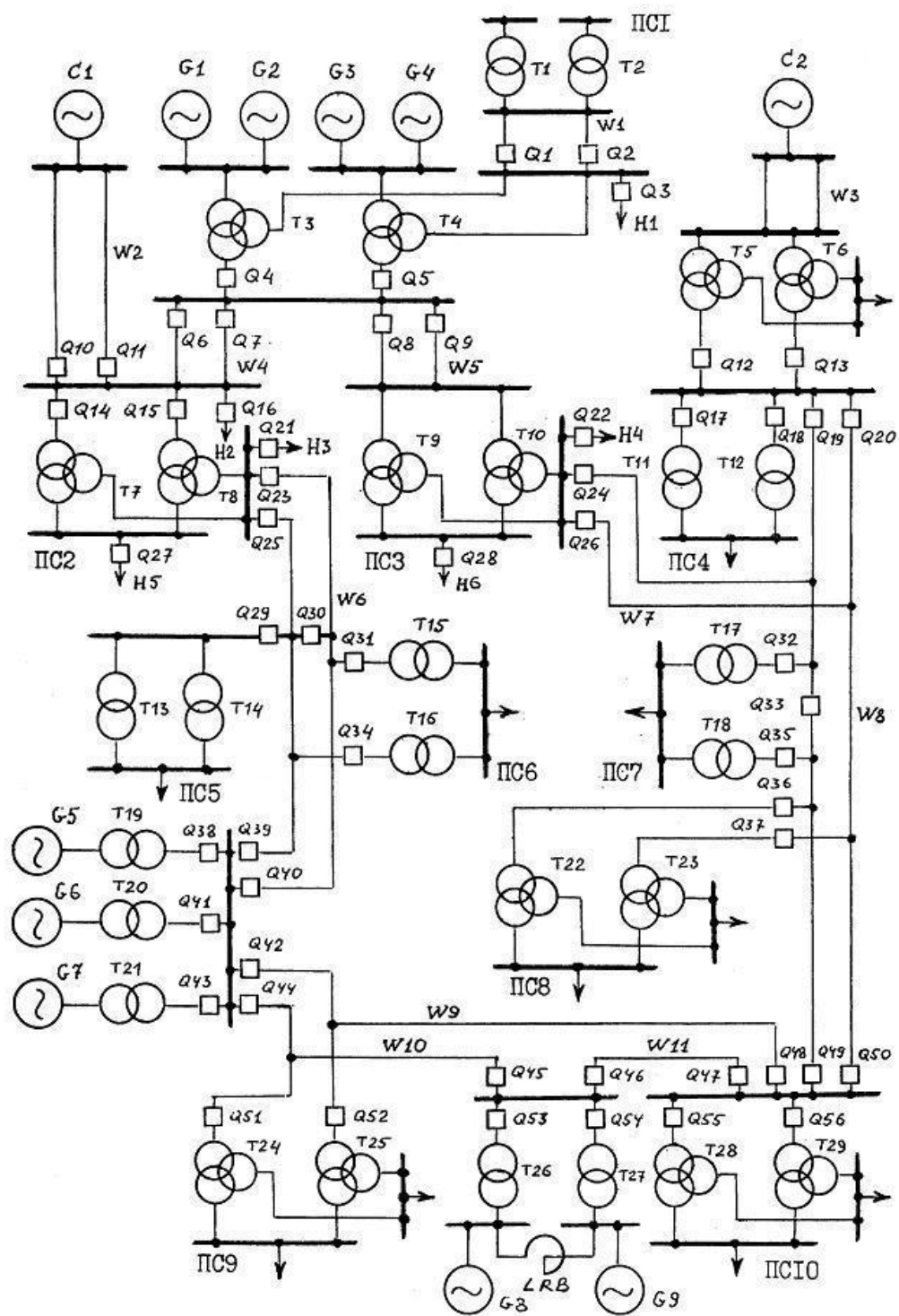
4. Согласно вашему варианту, начертите схему сети. Для удобства используйте слои и трафареты.



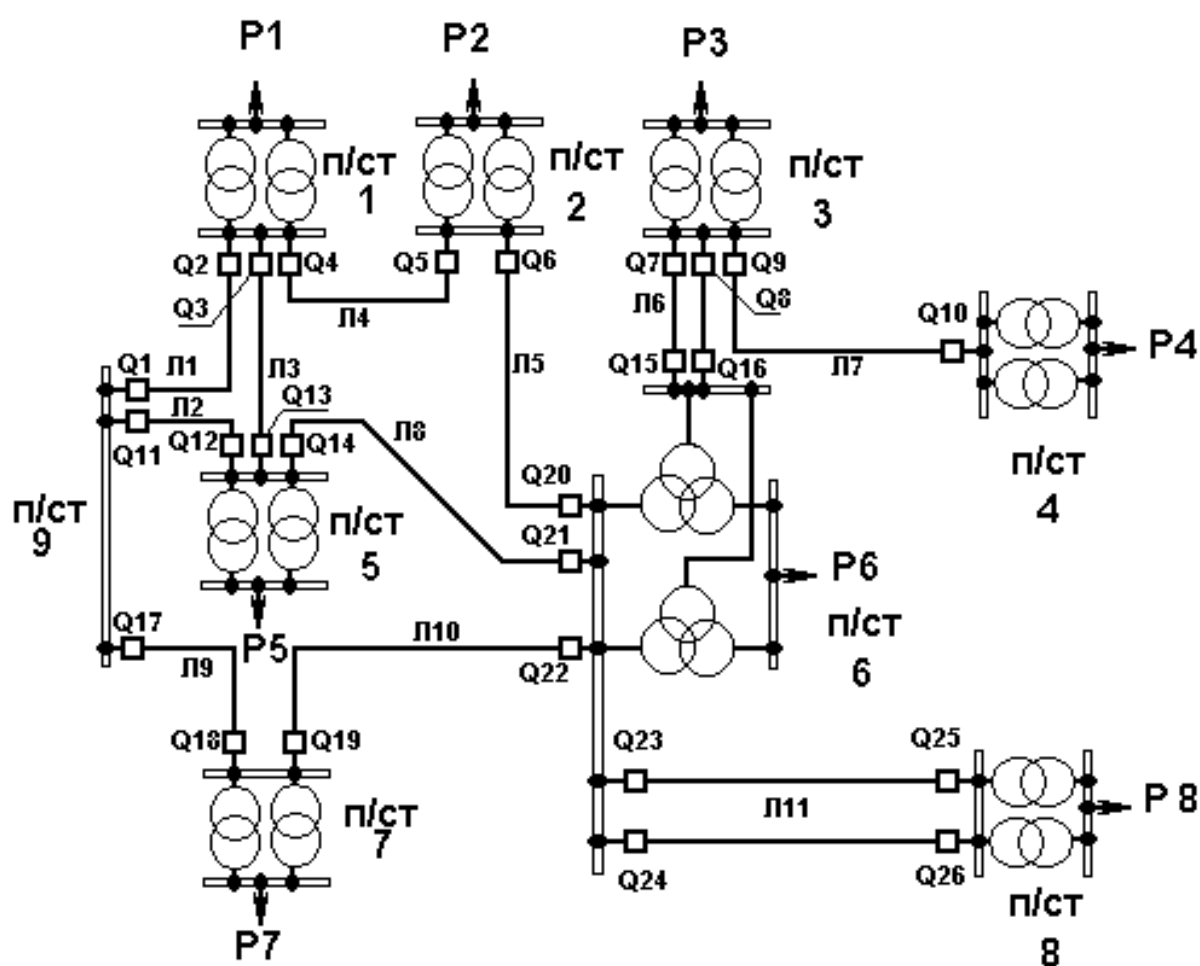
Вариант 1



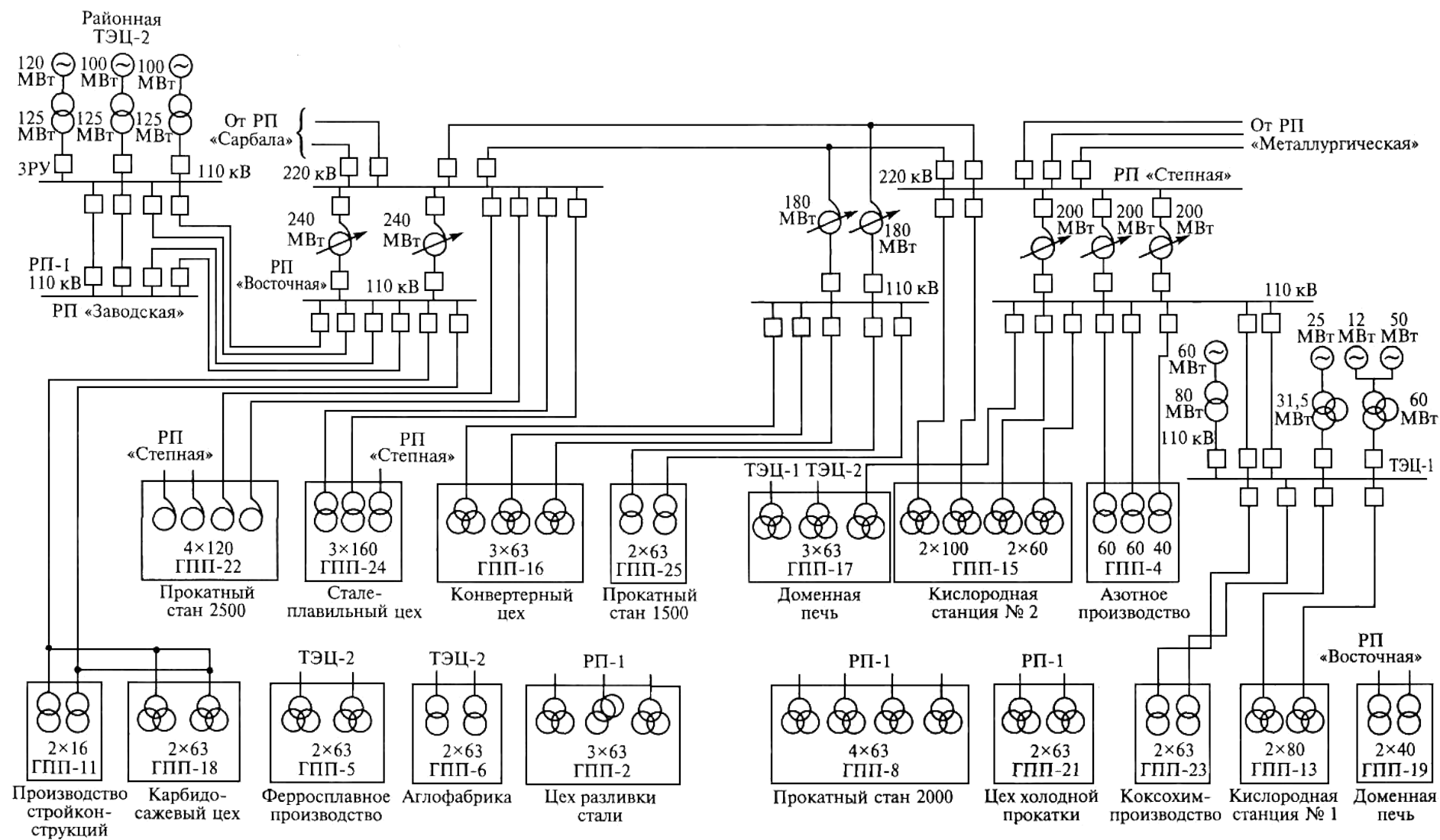
Вариант 2



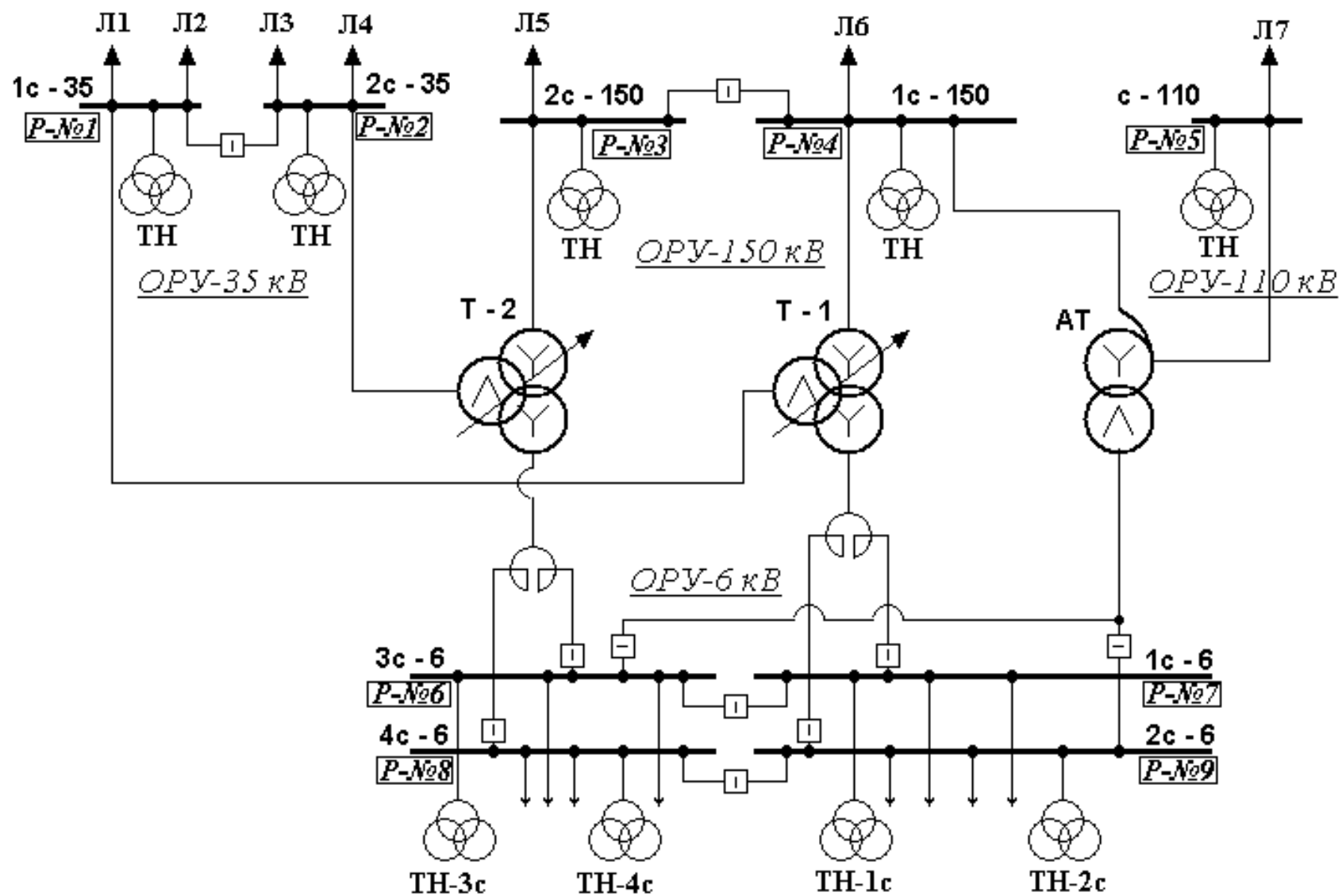
Вариант 3



Вариант 4

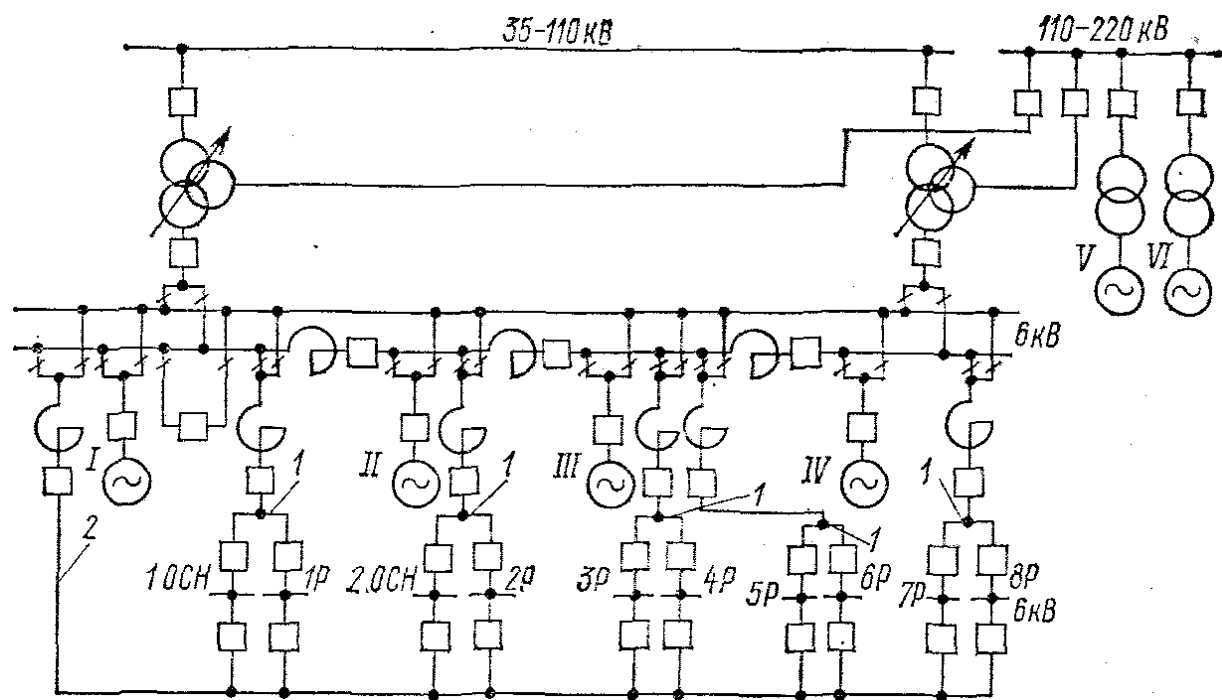


Вариант 5

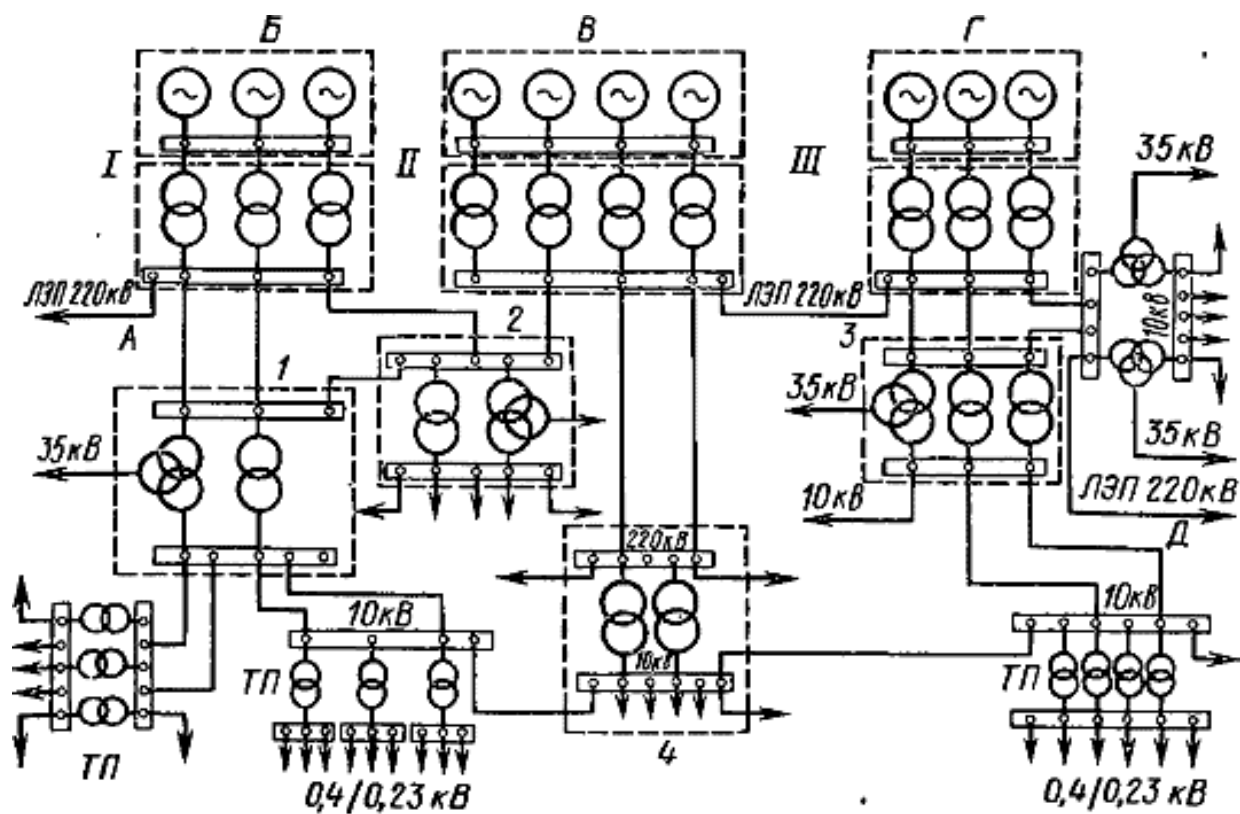


Вариант 6





Вариант 10



Вариант 11

известны? Чем они отличаются?

2. Что такое группировка объектов? Чем группировка отличается от операции объединения? Какие способы редактирования группы существуют?

3. Что такое соединительные точки? Для чего они нужны? Какие способы соединения элементов вам известны?

4. Что такое слои Visio, каково их назначение?

5. Как создать слой и как добавить в него элементы? Может ли один элемент находиться сразу в нескольких слоях?

6. Для чего нужны наборы элементов «MS Visio»? Укажите последовательность команд по созданию собственного набора элементов.

Лабораторная работа №6. Первичное знакомство с языком программирования Python

Цель работы: получить общие сведения о языке Python, научиться объявлять и присваивать переменные, познакомиться с арифметическими и логическими операторами, познакомиться с оператором ветвления, научиться выводить значения переменных на экран и вводить их с клавиатуры.

Теоретическая часть

1 Общие сведения о языке Python

1.1 Краткая история языка

Язык программирования Python является относительно «молодым» языком по сравнению со многими другими современными языками программирования. Python был задуман в еще в далеких 80-х программистом из Голландии Гвидо ван Россумом. Создавать его Гвидо начал в 1989 году как замену языку ABC, предназначенного для обучения студентов программированию. Он создавался на энтузиазме, без бюджета и поддержки. Работал Гвидо ван Россум по вечерам и в выходные дни. В результате за несколько недель был написан интерпретатор Python. Многие идеи были взяты из других языков программирования таких как C и ABC. В итоге из этого проекта вырос Python. Сегодня Python входит в тройку самых популярных языков мира. Гвидо ван Россум был поклонником комедийного телешоу «Летающий цирк Монти Пайтона» и в честь этого английского сериала, популярного в 1970 – х годах дал название своему языку программирования Python. Этот сериал принес еще одно очень популярное сегодня слово «спам». Оно обозначает навязчивую рекламу, с которой знаком каждый.

Публикация исходного текста Python произошла в 1991 году. Язык был относительно несложным. В нем было мало функций и интерпретатор «interpreter» тоже был небольшого размера. Всего было 3 версии питона:

Первая версия Python 1.0 увидела свет в январе 1994 года и развитие ее релизов продолжалось до 2000 года. Последним релизом стал Python 1.6. С 2000 по 2008 год развивалась его вторая версия. В это время проект был помещен на SourceForge. Это платформа, на которой расположены проекты с открытым кодом и был дан импульс к развитию языка сообществом программистов. В 2008 году вышла третья версия, которая развивается и в настоящее время. Поддержка второй версии постепенно прекращается и все большее число программистов пользуется третьей версией Python. На сегодняшний день последней версией питона является Python 3.10. Третья версия языка осуществляет поддержку второй не полностью.

1.2 Общие сведения о языке

Для того чтобы понять, что такое Python нужно разобраться зачем вообще нужны языки программирования. Они нужны для взаимодействия человека и компьютера. Между собой люди общаются на естественном языке. Компьютеры общаются между собой на естественном для машин языке – машинном языке, языке нулей и единиц. Для того, чтобы компьютер и человек понимал друг друга нужен промежуточный язык, который, с одной стороны, был бы удобен для понимания человеку, а с другой стороны, мог бы быть переведен с помощью специальной «программы-переводчика» на язык, понятный компьютеру. Таких языков со времен изобретения компьютера было придумано огромное множество. Все они имеют что-то общее, и каждый из них обладает своими особенностями. Одну и ту же задачу можно решить на разных языках программирования. Точно так же, как один и тот же предмет или действие можно описать на разных человеческих языках. Но одна и та же задача на разных языках программирования будет занимать разное количество строк, и выполняться за разное время. Это не говорит о

том, что какой-то из языков лучше, а какой-то хуже. Просто каждый язык изначально создается для решения конкретных задач и имеет в себе более широкий набор встроенных механизмов для их решения, по сравнению с другими языками, изначально не создаваемых для решения этих задач.

Python может быть полезен вам в вашей дальнейшей профессиональной деятельности для решения следующих задач:

- обработка статистических и экспериментальных данных;
- выполнение сложных математических операций;
- визуализация данных и результатов их обработки с помощью графиков, диаграмм, гистограмм и так далее;
- применение технологии искусственного интеллекта для решения поставленных профессиональных задач;
- работа с базами данных и технологией больших данных (BigData).

Все перечисленные задачи и многие другие можно решить с помощью встроенных и сторонних библиотек языка Python, поскольку Python является относительно простым высокоуровневым скриптовым языком, который служит для создания различных сценариев.

«Высокоуровневый» означает, что для программирования на Python, не требуется знания машинных кодов, команд для компьютера. Программирование при помощи машинных кодов увеличивает скорость работы программы, но очень трудоемко и требует хорошего знания архитектуры компьютера. Одним из таких низкоуровневых языков является ассемблер. Чаще всего низкоуровневое программирование используется в промышленности для специализированных контроллеров, где нет большого объема памяти. В отличие от ассемблера, Python не работает с машинными кодами. При этом программы выполняются более медленно, но они более понятны для человека. Учитывая быстродействие современных компьютеров, во многих случаях это преимущество простоты кода языка Python перевешивает его недостаток в скорости выполнения.

Под скриптовым языком программирования можно понимать то, что выполнение программы происходит сверху вниз построчно. Существует два принципиально разных типа «программ-переводчиков»: компиляторы и интерпретаторы.

Компиляторы переводят текст программы на язык, понятный компьютеру, сразу целиком. В итоге получается бинарный исполняемый файл, который можно скопировать и затем запустить на любом компьютере. Интерпретаторы переводят текст программы на язык, понятный компьютеру «на лету», построчно, и компьютер сразу же исполняет полученную команду. Поэтому, для того, чтобы программа, написанная на языке Python могла выполняться на компьютере, необходимо чтобы на нем был установлен интерпретатор языка Python.

1.3 Установка интерпретатора языка

Скачать самую последнюю версию интерпретатора языка Python можно совершенно бесплатно на официальном сайте проекта: <https://www.python.org>.

При установке рекомендуется установить галочки «Install launcher for all users» и «Add Python to PATH».



Рисунок 6.1 – Установка интерпретатора языка Python

После установки, запустить интерпретатор языка Python можно через меню «Пуск→Python→IDLE». Откроется интегрированная среда разработки IDLE (Integrated Development Environment), изображенная на рисунке 6.2.

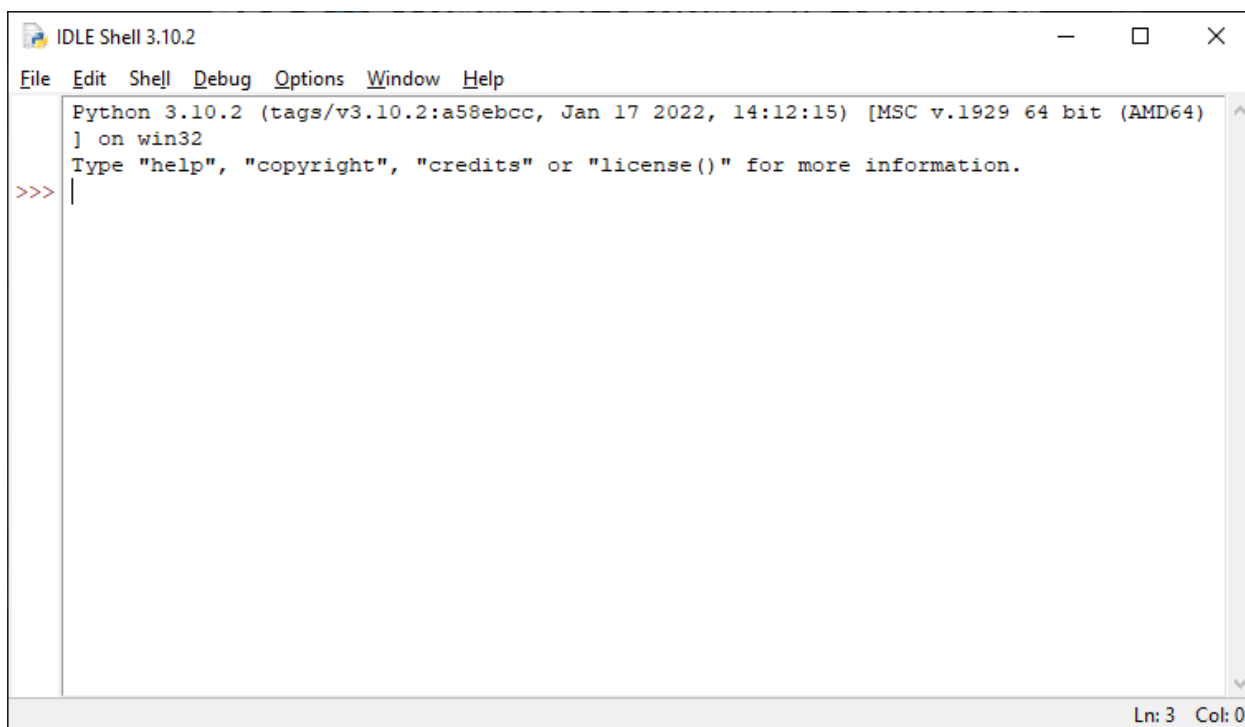


Рисунок 6.2 – Установка интерпретатора языка Python

1.4 Режимы работы Python

Интегрированная среда разработки IDLE поддерживает 2 режима работы:

1. Интерактивный режим. В этом режиме команды выполняются сразу после их вызова. Результат выполнения команд сразу выводится на экран. Этот режим работы хорошо подходит для написания простых программ и тестирования работы фрагментов кода. Интерактивный режим имеет следующие взаимосвязанные недостатки:

- программный код, который вводится, нигде не сохраняется. Чтобы повторно запустить один и тот же программный код, его нужно снова ввести;
- программы, которые введены в интерактивном режиме, после выполнения интерпретатором Python исчезают. Чтобы выполнить их

повторно, нужно их снова набрать или использовать операции копирования, что неэффективно.

2. Программный режим. В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью. Предварительно текст программы нужно сохранить. Программа на языке Python носит имя «сценарий» или «скрипт». Если система находится в интерактивном режиме, то программный режим можно вызвать с помощью команды «File→New File». Откроется текстовый редактор, изображенный на рисунке 6.3.

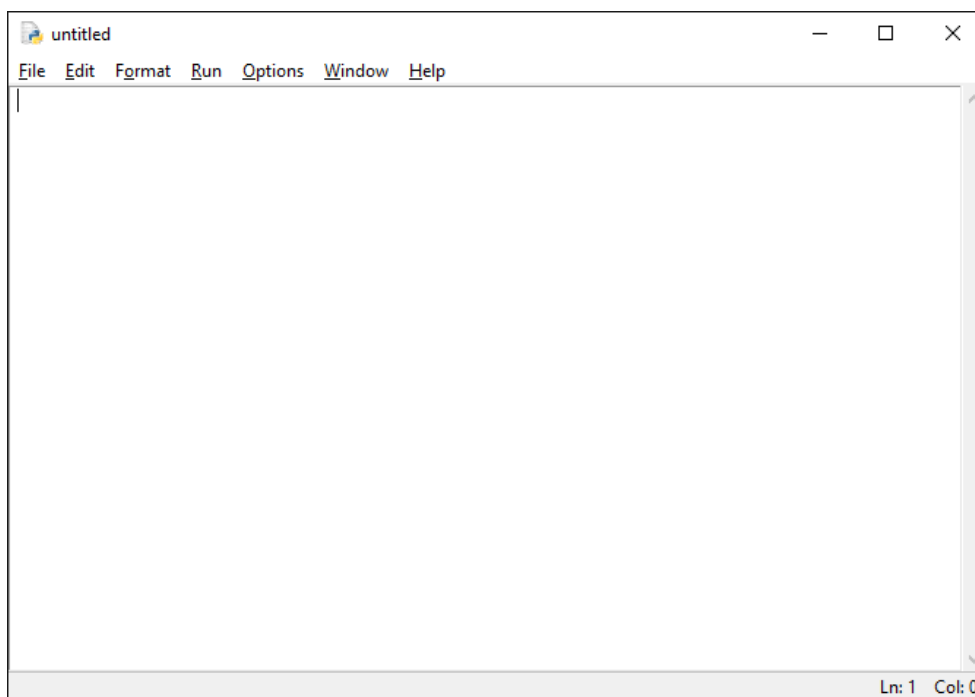


Рисунок 6.3 – Установка интерпретатора языка Python

После того, как текст программы будет введен, необходимо выполнить команду меню «Run→Run Module». Программа попросит выбрать путь для сохранения текста скрипта, после чего результаты выполнения программы будут отображены в окне среды IDLE.

2 Первая программа на языке Python

Исторически так сложилось, что самая первая программа на любом языке программирования называется «Hello world!». Суть этой программы заключается в том, чтобы вывести на экран какую-нибудь простейшую

фразу. Давайте напишем свою первую программу на языке Python. Для этого в среде IDLE (неважно в каком режиме, интерактивном или программном) необходимо написать следующую команду:

```
print("Hello world!")
```

И запустить ее на исполнение (в интерактивном режиме просто нажать Enter, в программном режиме выполнить команду меню «Run→Run Module»). На экране появится соответствующая фраза (рисунок 6.4).

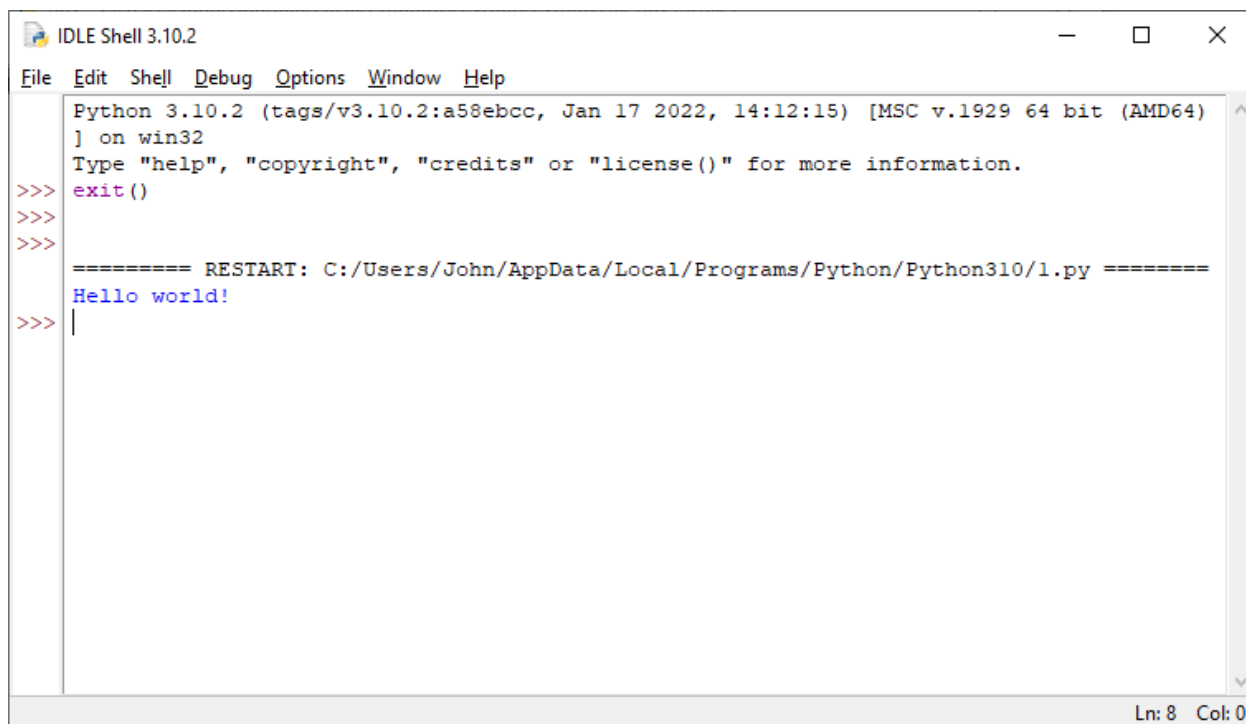


Рисунок 6.4 – Установка интерпретатора языка Python

Давайте разберемся, что означает эта команда. Ключевое слово «print» является названием функции. Функция – это заверченный кусочек программы, который выполняет определенные действия и, что самое важное, может использоваться многократно. В данном случае функция «print» выполняет вывод информации на экран. Вся программа строится на использовании уже готовых функций, входящих в состав Python, функций, написанных другими программистами и подключенных к вашей программе в виде модулей, либо же пользовательских функций, которые вы пишете сами. Функция может не иметь аргументов, или иметь один и более аргумент. Аргумент – это как бы начальные условия, параметры, описывающие

особенности выполнения функции. Аргументы функции заключаются в круглые скобки после названия самой функции и перечисляются через запятую, если их несколько. Если функция не имеет аргументов, в круглых скобках ничего не пишется. В данном случае аргументов функции является фраза «Hello world!». Она заключается в кавычки (двойные или одинарные), так как является строкой.

Ценность языков программирования заключается не в том, что они позволяют вывести на экран заранее определенные фразы, а в том, что позволяют производить вычисления и расчеты и на их основе уже принимать решение, что выводить на экран.

Для этого в любом языке программирования существуют переменные. Переменные – это своего рода ячейки памяти, в которых хранятся определенные значения и над которыми можно производить вычисления (сложение, умножение и так далее). Изначально значения в эти ячейки попадают различными способами:

- вводятся с клавиатуры;
- считываются из файла;
- считываются из баз данных;
- считываются из интернета;
- поступают от измерительных данных;
- заранее однозначно задаются программистом.

Переменная характеризуется именем, типом данных и значением.

Имя – это уникальный идентификатор, по которому однозначно можно обратиться к ячейке памяти, не перепутав ее с какой-либо другой ячейкой. Тип указывает, что же собственно говоря хранится в ячейке памяти. Типы данных можно условно разделить на основные и составные. Основные, в свою очередь, делятся на простейшие и расширенные. К простейшим типам данных относятся:

- логические (bool, могут принимать значение True либо False);
- целочисленные (int, для хранения целых чисел);

- вещественные (float, для хранения дробных чисел);
- комплексные (complex, для хранения комплексных чисел, состоящих из вещественной и мнимой части);
- строковые (str, для хранения строковых данных).

На основе этих простейших типов строятся более сложные, расширенные типы данных (списки, массивы, множества, словари), а также любые другие, составные типы данных.

Особенностью языка Python является то, что в любой переменной в любой момент времени может храниться любой тип данных. Например:

```
a=True  
a=1  
a=0.1  
a='Hello'
```

В данном случае, а – имя переменной. Имя переменной должно начинаться с буквы или символа подчеркивания. Имя переменной не может начинаться с цифры. Имя переменной может содержать только буквенно-цифровые символы и символы подчеркивания (A – z, 0–9 и _). Имена переменных чувствительны к регистру (age, Age и AGE - это три разные переменные). После знака равенства указывается значение, которые мы заносим в переменную. В первой строке мы заносим значение типа «bool», затем целочисленное, затем вещественное и в конце – строковое. Узнать, какого типа данные сейчас хранятся в переменной можно с помощью функции type(). Например:

```
print("Hello world!")  
a=True  
print(type(a))  
a=1  
print(type(a))  
a=0.2  
print(type(a))  
a='Hello'  
print(type(a))
```

В результате получим:

```

>>> <class 'str'>
===== RESTART: C:/Users/John/AppData/Local/Programs/Python/Python310/1.py =====
Hello world!
<class 'bool'>
<class 'int'>
<class 'float'>
<class 'str'>
>>>

```

Над переменными можно выполнять различные операции. Операции над переменным осуществляются с помощью операторов.

Оператор в Python — это символ, который выполняет операцию над одним или несколькими операндами. Операндом выступает переменная или значение, над которыми проводится операция.

Операторы Python бывают 7 типов:

- арифметические операторы;
- операторы сравнения;
- операторы присваивания;
- логические операторы;
- операторы принадлежности;
- операторы тождественности;
- битовые операторы.

Познакомимся с некоторыми из них:

2.1 Операторы языка Python

2.1.1 Арифметические операторы Python

Сложение (+): складывает значение по обе стороны оператора.

```

>>> 3+4
7

```

Вычитание (-): вычитает значение правой стороны из значения в левой.

```

>>> 3-4
-1

```

Умножение (*): перемножает значения с обеих сторон оператора.

```

>>> 3*4
12

```

Деление (/): делит значение левой стороны на значение правой. Тип данных результата деления — число с плавающей точкой.

```
>>> 3/4
0.75
```

Возведение в степень ():** возводит первое число в степень второго.

```
>>> 3**4
81
```

Деление без остатка (//): выполняет деление и возвращает целочисленное значение частного, убирая цифры после десятичной точки.

```
>>> 4//3
1
>>> 10//3
3
```

Деление по модулю (остаток от деления) (%): выполняет деление и возвращает значение остатка.

```
>>> 3%4
3
>>> 4%3
1
>>> 10%3
1
>>> 10.5%3
1.5
```

2.1.2 Операторы сравнения

Операторы сравнения в Python проводят сравнение операндов. Они сообщают, является ли один из них больше второго, меньше, равным или то и то.

Меньше (<): этот оператор проверяет, является ли значение слева меньше, чем правое.

```
>>> 4<3
False
```

Больше (>): проверяет, является ли значение слева больше правого.

```
>>> 4>3
True
```

Меньше или равно (<=): проверяет, является ли левая часть меньше или равной правой.

```
>>> 7<=7
```

```
True
```

Больше или равно (>=): проверяет, является ли левая часть больше или равной правой.

```
>>> 0>=0
True
```

Равно (==): этот оператор проверяет, равно ли значение слева правому.

```
>>> 3==3.0
True
>>> 1==True
True
>>> 7==True
False
>>> 0==False
True
>>> 0.5==True
False
```

Не равно (!=)

Проверяет, не равно ли значение слева правому.

Когда условие выполнено, возвращается `True`. В противном случае – `False`. Это возвращаемое значение можно использовать в последующих инструкциях и выражениях.

Пример:

```
>>> 1!=1.0
False
```

2.1.3 Операторы присваивания

Оператор присваивания присваивает значение переменной. Он может манипулировать значением до присваивания. Есть 8 операторов присваивания: 1 простой и 7 с использованием арифметических операторов.

Присваивание (=): присваивает значение справа левой части. Стоит обратить внимание, что `==` используется для сравнения, а `=` — для присваивания.

```
>>> a = 7
>>> print(a)
7
```

Сложение и присваивание (+=): суммирует значение обеих сторон и присваивает его выражению слева. `a += 10` — это то же самое, что и `a = a + 10`. То же касается и все остальных операторов присваивания.

```
>>> a += 2
>>> print(a)
9
```

Вычитание и присваивание (-=): вычитает значение справа из левого и присваивает его выражению слева.

```
>>> a -= 2
>>> print(a)
7
```

Деление и присваивание (/=): делит значение слева на правое. Затем присваивает его выражению слева.

```
>>> a /= 7
>>> print(a)
1.0
```

Умножение и присваивание (*=): перемножает значения обеих сторон. Затем присваивает правое левому.

```
>>> a *= 8
>>> print(a)
8.0
```

Деление по модулю и присваивание (%=): выполняет деление по модулю для обеих частей. Результат присваивает левой части.

```
>>> a %= 3
>>> print(a)
2.0
```

Возведение в степень и присваивание (=):** выполняет возведение левой части в степень значения правой части. Затем присваивает значение левой части.

```
>>> a **= 5
>>> print(a)
32.0
```

Деление с остатком и присваивание (//=): выполняет деление с остатком и присваивает результат левой части.

```
>>> a //= 3
>>> print(a)
10.0
```

2.1.4 Логические операторы Python

Это союзы, которые позволяют объединять по несколько условий. В Python есть всего три оператора: `and` (и), `or` (или) и `not` (не).

И (`and`): если условия с двух сторон оператора `and` истинны, тогда все выражение целиком считается истинным.

```
>>> a = 7 > 7 and 2 > -1
>>> print(a)
False
```

Или (`or`): выражение ложно, если оба операнда с двух сторон ложные. Если хотя бы одно из них истинное, то и все выражение истинно.

```
>>> a = 7 > 7 or 2 > -1
>>> print(a)
True
```

Не (`not`)

Этот оператор инвертирует булевы значения выражения. `True` превращается в `False` и наоборот. В примере внизу булево значение `0` – `False`. Поэтому оно превращается в `True`.

```
>>> a = not(0)
>>> print(a)
True
```

2.1.5 Оператор ветвления

Еще одним важным оператором является оператор ветвления. Оператор ветвления(условия) – оператор управления, осуществляющий выполнение одного и только одного составного оператора из двух возможных в зависимости от условия. Предположим, нам нужно сравнить два числа `A` и `B` и вывести на экран наибольшее из них. В этом случае нам поможет оператор ветвления, который выглядит следующим образом:

```
if выражение:
    инструкция_1
    инструкция_2
    ...
    инструкция_n
```

После оператора *if* записывается выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором.

Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо логическое *True*. После выражения нужно поставить двоеточие “:”.

ВАЖНО: блок кода, который необходимо выполнить, в случае истинности выражения, должен быть смещен относительно оператора *if* на один или несколько пробелов. Количество пробелов, на которое смещены команды, не имеет значения: 1,2,4,10,20 и так далее. Важно чтобы все команды, которые должны выполняться, были смещены на одинаковое количество пробелов, то есть располагались как бы в столбик:

```
if a>5:
    print("hello")
    print("hello")
    print("hello")
    print("hello")
```

Для удобства восприятия, принято смещать блок команд на 4 пробела. Именно столько пробелов интегрированная среда разработки добавит сама, если нажать на клавишу «Tab».

Примеры:

```
if 1:
    print("hello 1")
```

Напечатает: *hello 1*

```
a = 3
if a == 3:
    print("hello 2")
```

Напечатает: *hello 2*

```
a = 3
if a > 1:
    print("hello 3")
```

Напечатает: *hello 3*

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция *if – else*.

```
if выражение:
    инструкция_1
    инструкция_2
```

```
...
инструкция_n
else:
    инструкция_a
    инструкция_b
...
инструкция_x
```

Примеры.

```
a = 3
if a > 2:
    print("H")
else:
    print("L")
```

Напечатает: *H*

```
a = 1
if a > 2:
    print("H")
else:
    print("L")
```

Напечатает: *L*

Для реализации выбора из нескольких альтернатив можно использовать конструкцию *if – elif – else*.

```
if выражение_1:
    инструкции_(блок_1)
elif выражение_2:
    инструкции_(блок_2)
elif выражение_3:
    инструкции_(блок_3)
else:
    инструкции_(блок_4)
```

Пример.

```
a = input("введите число:")
a = int(a)
if a < 0:
    print("Neg")
elif a == 0:
    print("Zero")
else:
    print("Pos")
```

В данном примере мы встречаем две новые функции: `input` и `int`. Функция `input` предназначена для ввода данных с клавиатуры. У нее есть один аргумент – подсказка пользователю, которая будет выведена на экран. Результатом выполнения функции является строка, которую пользователь введет с клавиатуры. Чтобы понять, для чего нужна функция `int` давайте

вернемся типам данных языка Python. В языке Python существуют следующие базовые типы данных: логический, целочисленный, вещественный, комплексный и строковый. Важно то, что в зависимости от того, какой тип данных хранится в переменной будет зависеть какие операции можно будет выполнить с этой переменной. Например, над числами можно выполнять арифметические операции: складывать, вычитать, возводить в квадрат. Сложно себе представить, как можно возвести в квадрат строку, например. В то же самое время, например, сложение строк является довольно очевидной операцией: мы просто объединяем эти две строки. Таким образом, в зависимости от того, какого типа данные хранятся в переменной будет зависеть результат операций над этими переменными. Например:

```
a=5
b=6
c="5"
d="6"
print(a+b)
print(c+d)
```

На экран будет выведено две строчки: 11 и 56. 11 потому что a и b являются числами и при сложении будут использованы правила сложения чисел, а 56, потому что переменные c и d являются строками и при сложении будут использоваться правила сложения строк, то есть к концу первой строки будет добавлена вторая строка. Результатом выполнения функции `input` всегда является строка. Для того, чтобы сказать программе, что нам нужна не строка, а число, мы вызываем функцию `int()`. Если нам нужно будет вещественное число, мы используем функцию `float()`, если комплексное, то функцию `complex()` и так далее. Аналогичным образом, если нам нужно преобразовать число в строку, мы используем функцию `str()`.

Еще один интересный момент: чтобы сократить количество строк в программе, мы можем использовать в качестве аргумента функции другую функцию. Например, вместо

```
a = input("введите число:")
a = int(a)
```

можно сразу написать:

<pre>a = int(input("введите число:"))</pre>

Результат будет один и тот же

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Ознакомьте с теоретическим материалом.
2. Ответьте на контрольные вопросы, приведенные в конце работы.
3. Выполните индивидуальное задание, согласно вашему номеру варианта (Таблица 6.1).

Таблица 6.1 – Задания для самостоятельного выполнения:

№ варианта	Задачи для выполнения
1	9,6,10
2	10,7,4
3	7,5,2
4	6,8,9
5	6,7,2
6	7,10,1
7	9,10,1
8	10,8,2
9	8,9,2
10	5,6,7
11	9,5,6
12	5,7,2
13	7,5,1
14	3,9,4
15	6,7,8

Задача №1. Создайте программу решения квадратного уравнения. Коэффициенты А,В,С квадратного уравнения пользователь должен вводить с клавиатуры, корни уравнения необходимо вывести на экран.

Задача №2. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо найти их сумму и вывести на экран.

Задача №3. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо найти их произведений и вывести на экран.

Задача №4. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо посчитать, сколько чисел оказалось больше 10 и вывести на экран это количество.

Задача №5. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо посчитать количество четных и нечетных чисел и вывести на экран это количество.

Задача №6. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо вывести на экран сообщение, показывающее, каких чисел оказалось больше: четных или нечетных.

Задача №7. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо посчитать произведение четных и сумму нечетных чисел и вывести на экран полученные значения.

Задача №8. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо найти сумму квадратов этих чисел и вывести на экран.

Задача №9. Напишите следующую программу: пользователь вводит с клавиатуры длину и ширину прямоугольника. Необходимо посчитать и вывести на экран: периметр прямоугольника, его площадь и длину диагонали.

Задача №10. Напишите следующую программу: пользователь вводит с клавиатуры 5 чисел. Необходимо четные числа сложить по правилам сложения чисел, а нечетные числа сложить по правилам сложения строк.

Контрольные вопросы:

1. Почему язык Python получил такое название?
2. Назовите основные особенности языка Python.
3. Для каких задач может быть полезен язык Python?
4. Назовите режимы работы среда разработки IDLE. В чем их особенности?
5. Что такое функция в языке? Что такое аргумент функции?
6. Что такое переменная? Чем характеризуется переменная?
7. Что такое оператор? Какие операторы вы знаете?
8. Как выводить на экран значения переменных?
9. Как ввести с клавиатуры значения переменных?
10. Опишите синтаксис и назначение оператора ветвления. Приведите пример его использования.
11. Для чего нужны функции `int()`, `float()`, `complex()`, `str()`?

Лабораторная работа №7. Python: списки, циклы, функции

Цель работы: научиться работать со списками, циклами, создавать собственные функции

Теоретическая часть

1 Списки

На прошлом занятии были рассмотрены основные типы данных языка Python: целочисленные, вещественные, строковые и логические. Однако, помимо перечисленных, в языке Python есть поддержка более сложных типов данных. Если у вас есть список предметов (например, список марок авто), то хранение автомобилей в отдельных переменных может выглядеть так:

```
car1 = "Ford"  
car2 = "Volvo"  
car3 = "BMW"
```

Для каждой новой марки нужно выделять переменную с новым именем. Это очень неудобно, ведь если их будет не 3, а, например, 3000, нужно будет создавать 3000 новых переменных. Кроме того, работать с такими переменными тоже будет неудобно. Например, если нужно будет проверить, есть ли среди перечисленных марок какая-то определенная, нужно будет написать 3000 команд сравнения. К счастью, существуют более простые подходы к написанию программы. Для таких случаев существует специальный тип переменных, который называется «список». Список – это специальная переменная, которая может содержать более чем одно значение. Для создания списка необходимо использовать квадратные скобки. Например:

```
a=[]
```

Эта команда создаст пустой список. При создании списка можно сразу же поместить в него значения, через запятую. Например:

```
a=[1, 2, 3, 10]
```

Вывести на экран все элементы списка можно с помощью уже известной нам команды print:

```
a=[1, 2, 3, 10]
print(a)
```

1.1 Обращение к элементам списка

Каждый элемент в списке имеет свой индекс. Индекс – это порядковый номер элемента в списке. Индекс начинается с нуля.



Рисунок 7.1 – Индексация элементов в списке

Для доступа к элементу списка по индексу используются квадратные скобки:

```
a=[1, 2, 3, 10]
print(a[2])
```

выведет на экран число 3, так по индексу 2 располагается именно это число.

Существует также отрицательный индекс. Отрицательными индексами называют расположение элементов в списке справа налево. При этом нумерация начинается с -1.



Рисунок 7.2 – Индексация элементов в списке с конца

Отрицательным индексом удобно пользоваться, когда необходимо обратиться к последнему в списке элементу, не высчитывая его номер. Любой конечный элемент будет с индексом, равным -1.

Если же нужно получить из списка не один, а сразу несколько элементов, используются так называемые срезы. Срезы (slices) — это подмножества элементов списка. У них есть свой собственный синтаксис. Записывается срез так же, как обращение к элементу, используя индекс. Пример:

```
elements[START:STOP:STEP]
```

В этом случае берётся срез от номера start (включительно) до stop (не включая его), а step — это шаг. По умолчанию start и stop равны 0, step равен 1.

```
elements = [0.1, 0.2, 1, 2, 3, 4, 0.3, 0.4]
int_elements=elements[2:6] #выборка со 2 по 6-й элементы
print(int_elements)
```

На экран будут выведены элементы с индексами 2,3,4 и 5, то есть, в данном случае, только целые числа.

1.2 Комментарии в программе

В последнем примере встречается новый символ – решетка #. Этот символ используется для написания комментариев к программе. Все, что идет после этого символа и до конца строки программа не воспринимает. Это всего лишь подсказка, которую программист пишет сам себе или тем, кто будет читать эту программу в дальнейшем. Символ # удобно использовать, если нужно временно отключить выполнение какой либо команды. В этом случае можно полностью удалить эту команду. Но тогда нужно будет ее снова писать, когда она нам понадобится. Другой выход – написать символ # в начале строки с этой командой и она не будет выполняться. Например:

```
car1 = "Ford"
car2 = "Volvo"
car3 = "BMW"
print(car1)
#print(car2)
#print(car3)
```

Выведет на экран только слово «Ford», так как вывод на экран двух остальных переменных «закомментирован».

Символ # называется однострочным комментарием, поскольку действует только до конца строки. Существует также и многострочный комментарий. Она начинается с трех кавычек (одинарных, или двойных, не имеет значение) и заканчивается тремя такими же кавычками. Многострочные комментарии используется в случаях, когда нужно написать много комментариев и не хочется в начале каждой строки писать символ #, либо если нужно временно отключить сразу несколько строк программы.

```
car1 = "Ford"
car2 = "Volvo"
car3 = "BMW"
'''print(car1)
print(car2)
print(car3)'''
```

Не выведет на экран ничего, поскольку команды вывода на экран закомментированы тройными кавычками.

1.3 Добавление элементов в список

В списках доступно добавление, изменение, удаление элементов. Рассмотрим каждый способ изменения элементов на примерах.

Для того чтобы добавить новый элемент в список, используется `list.append(x)`, где `list` — список, `x` — нужное значение.

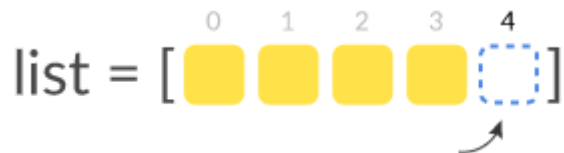


Рисунок 7.3 – Добавление элемента в конец списка

```
>>> elements = [1, 2, 3, 'word']
>>> elements.append('meow')
>>> print(elements)
[1, 2, 3, 'word', 'meow']
```

Для простого примера, рассмотрим создание списка с нуля с помощью метода `append()`:

```
>>> elements = []
>>> elements.append(1)
>>> elements.append('word')
>>> elements.append('meow')
>>> print(elements)
[1, 'word', 'meow']
```

Функция `append()` добавляет элемент в конец списка. Но если нужно добавить элемент в середину списка, необходимо использовать метод `insert`. Метод `insert` имеет следующий синтаксис:

`list.insert(i, x)`, где `list` – список, `i` – позиция, `x` – нужное значение.

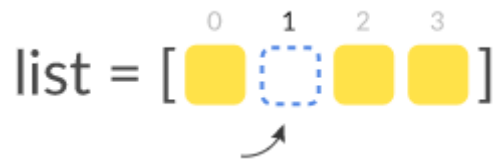


Рисунок 7.4 – Добавление элемента в середину списка

Пример:

```
>>> elements = [1, 2, 4]
>>> print(elements)
[1, 2, 4]

>>> elements.insert(2, 3)
>>> print(elements)
[1, 2, 3, 4]
```

Изменение элементов списка происходит следующим образом: нужно выбрать элемент по индексу (порядковому номеру элемента) и присвоить новое значение.



Рисунок 7.5 – Замена элемента в списке

```
>>> elements = [2, 4, 6]
>>> elements[2] = 8

>>> print(elements)
[2, 4, 8]
```

В примере выше мы заменили 6 на 8. Не забывайте, что счёт начинается с нуля, и в данном списке цифра 6 это 2-й элемент.

Для удаление из списка используют функция `list.pop(i)`, где `list` – список, `i` – индекс (позиция) элемента в списке. Если не указывать индекс, то будет удаляться последний элемент из списка.



1.4 Объединение списков

Списки в Python можно объединять с помощью оператора + или метода extend .



Рисунок 7.7 – Объединение двух списков

Выглядит это так:

```
>>> a = [1, 3, 5]
>>> b = [1, 2, 4, 6]
>>> print(a + b)
[1, 3, 5, 1, 2, 4, 6]

>>> hello = ["h", "e", "l", "l", "o"]
>>> world = ["w", "o", "r", "l", "d"]
>>> hello.extend(world)    # extends не возвращает новый
список, а дополняет текущий
>>> print(hello)
['h', 'e', 'l', 'l', 'o', 'w', 'o', 'r', 'l', 'd']
```

У списков есть еще несколько полезных функций:

list.clear() – удаляет все элементы списка, не имеет аргументов;

list.count(x) – посчитает количество элементов x в списке;

len(list) – посчитает количество всех элементов в списке;

list.remove(x) – удалить из элемент x (удаляет только один элемент, который встретится первый раз);

list.index(x) – вернет позицию первого найденного элемента x в списке;

list.copy() – служит для копирования списков;

list.reverse() – меняет порядок элементов в списке на противоположный;

list.sort() – сортирует список.

2 Циклы

Выполнение программ, написанных на любом языке программирования, по умолчанию является последовательным. Иногда нам может понадобиться изменить выполнение программы. Выполнение определенного кода может потребоваться повторить несколько раз.

Для этого в языках программирования предусмотрены различные типы циклов, которые способны повторять определенный код несколько раз. Чтобы понять принцип работы оператора цикла, рассмотрим следующую схему.

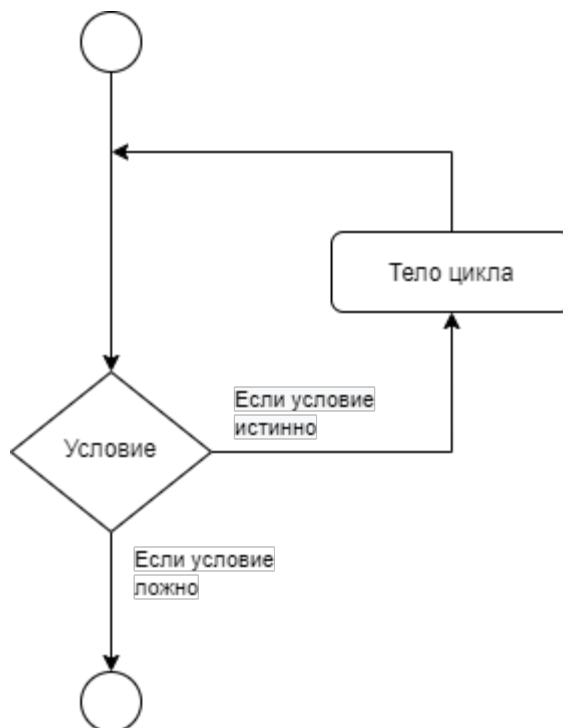


Рисунок 7.8 – Циклы с постусловием

Циклы упрощают сложные задачи до простых. Он позволяет нам изменить поток программы таким образом, что вместо того, чтобы писать один и тот же код снова и снова, мы можем повторять его конечное число раз. Например, если нам нужно вывести первые 10 натуральных чисел, то вместо того, чтобы использовать оператор print 10 раз, мы можем вывести их внутри цикла, который выполняется до 10 итераций.

В Python преимущества циклов, как и в других языках программирования, заключаются в следующем:

- Это обеспечивает возможность повторного использования кода.
- Используя циклы, нам не нужно писать один и тот же код снова и снова.
- С помощью циклов мы можем перебирать элементы сложных структур данных (списков и др.).

В Python существуют следующие два типа циклических выражений:

- Цикл `while`;
- Цикл `for`.

2.1 Циклы `while`

Цикл `while` позволяет выполнять часть кода до тех пор, пока заданное условие не станет ложным. Он также известен как цикл с предварительной проверкой условия.

Его можно рассматривать как повторяющийся оператор `if`. Когда мы не знаем количество итераций, цикл `while` является наиболее эффективным.

Синтаксис приведен ниже.

```
while expression:  
    statements
```

Здесь утверждения могут быть одним утверждением или группой утверждений. Выражение должно быть любым допустимым выражением Python, приводящим к `true` или `false`. Истиной является любое ненулевое значение, а ложью - 0.



Рисунок 7.9 – Циклы с предусловием

Рассмотрим следующий пример цикла while в Python:

```

money = 10
while money > 0:
    msg="у нас есть "+str(money)+" рублей. Можно отдохнуть"
    print (msg)
    money -= 1
print ("Денег больше нет. Пора работать")
  
```

Запустив код, вы увидите следующий результат:

```

--
у нас есть 10 рублей. Можно отдохнуть
у нас есть 9 рублей. Можно отдохнуть
у нас есть 8 рублей. Можно отдохнуть
у нас есть 7 рублей. Можно отдохнуть
у нас есть 6 рублей. Можно отдохнуть
у нас есть 5 рублей. Можно отдохнуть
у нас есть 4 рублей. Можно отдохнуть
у нас есть 3 рублей. Можно отдохнуть
у нас есть 2 рублей. Можно отдохнуть
у нас есть 1 рублей. Можно отдохнуть
Денег больше нет. Пора работать
  
```

Как вы видите, цикл выполнен 10 раз. На каждой итерации цикла мы формировали сообщение, отображая текущее значения переменной money, а

затем уменьшали его на 1. Как только переменная money стала равна нулю, условие продолжения цикла перестало выполняться и он завершился.

2.2 Операторы управления циклом

Мы можем изменить обычную последовательность выполнения цикла while с помощью оператора управления циклом. Когда выполнение цикла while завершается, все автоматические объекты, определенные в этой области видимости, уничтожаются. Python предлагает следующие управляющие операторы для использования в цикле while.

Оператор **continue** - Когда встречается оператор continue, управление переходит в начало цикла. Иными словами, с помощью оператора **continue** можно пропустить итерацию цикла Например:

```
# Вывод всех букв кроме 'a' и 't'
i = 0
str1 = 'javatpoint'

while i < len(str1):
    if str1[i] == 'a' or str1[i] == 't':
        i += 1
        continue
    print('Текущая буква :', str1[i])
    i += 1
```

Вывод:

```
Текущая буква: j
Текущая буква: v
Текущая буква: p
Текущая буква: o
Текущая буква: i
Текущая буква: n
```

Оператор **break**. Оператор **break** выводит управление из цикла, иными словами он досрочно завершает цикл.

Пример:

```
# Вывод всех букв до тех пор, пока не встретится буква 't'.
i = 0
str1 = 'javatpoint'
while i < len(str1):
    if str1[i] == 't':
        i += 1
        break
```

```
break
print('Текущая буква :', str1[i])
i += 1
```

Вывод:

```
Текущая буква : j
Текущая буква: a
Текущая буква: v
Текущая буква: a
```

2.3 Цикл for в Python

Цикл **for** в Python используется для многократного повторения операторов или части программы. Он часто используется для обхода сложных структур данных, например, списков. Синтаксис цикла for в python приведен ниже.

```
for [элемент] in [последовательность]:
    [список операций]
```

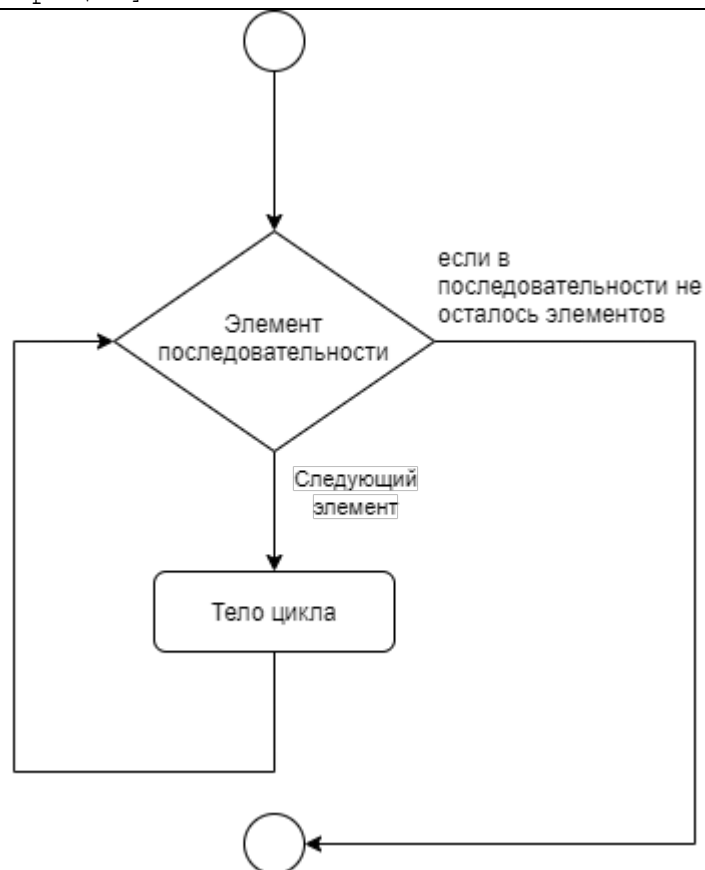


Рисунок 7.10 – Циклы с повторением для элементов последовательности

Цикл for в Python используется для работы с последовательностями или если число итераций известно заранее. Синтаксис выглядит следующим образом:

```
for [элемент] in [последовательность]:  
    [список операций]
```

Пример: Итерация строки с помощью цикла for

```
str = "Python"  
for i in str:  
    print(i)
```

Вывод:

```
P  
y  
t  
h  
o  
n
```

Пример: Программа для печати таблицы умножения заданного числа.

```
list = [1,2,3,4,5,6,7,8,9,10]  
n = 5  
for i in list:  
    c = n*i  
    print(c)
```

Вывод:

```
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```

Пример: Программа для печати суммы заданного списка.

```
list = [10,30,23,43,65,12]  
sum = 0  
for i in list:  
    sum = sum+i  
print("The sum is:",sum)
```

Вывод:

```
The sum is: 183
```

Цикл for удобно использовать совместно с функцией range. Функция range() используется для генерации последовательности чисел. Если мы

передадим `range(10)`, она сгенерирует числа от 0 до 9. Синтаксис функции `range()` приведен ниже.

```
range(start, stop, step size)
```

Start – означает начало итерации.

Stop – означает, что цикл будет повторяться до `stop-1`. `range(1,5)` будет генерировать числа от 1 до 4 итераций. Это необязательный параметр.

Размер шага используется для пропуска определенных чисел в итерации. Его использование необязательно. По умолчанию размер шага равен 1. Это необязательно.

Пример: Программа для печати чисел по порядку.

```
for i in range(10):  
    print(i)
```

Вывод:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

Пример: Программа для печати таблицы умножения заданного числа.

```
n = int(input("Enter the number "))  
for i in range(1,11):  
    c = n*i  
    print(n, "*", i, "=", c)
```

Вывод:

```
Enter the number 10  
10 * 1 = 10  
10 * 2 = 20  
10 * 3 = 30  
10 * 4 = 40  
10 * 5 = 50  
10 * 6 = 60  
10 * 7 = 70  
10 * 8 = 80  
10 * 9 = 90  
10 * 10 = 100
```

Пример: Программа для печати четного числа с использованием размера шага в range().

```
n = int(input("Enter the number "))
for i in range(2,n,2):
    print(i)
```

Вывод:

```
Enter the number 20
2
4
6
8
10
12
14
16
18
```

Мы также можем использовать функцию range() с последовательностью чисел. Функция len() сочетается с функцией range(), которая выполняет итерацию по последовательности с использованием индексации.

Пример:

```
list = ['Peter', 'Joseph', 'Ricky', 'Devansh']
for i in range(len(list)):
    print("Hello", list[i])
```

Вывод:

```
Hello Peter
Hello Joseph
Hello Ricky
Hello Devansh
```

В циклах for могут точно также применяться операторы break и continue для досрочного завершения цикла или для пропуска одной из итераций цикла.

2.4 Вложенные циклы

Python позволяет нам вложить любое количество циклов for или while внутрь друг друга. Внутренний цикл выполняется n раз за каждую итерацию внешнего цикла. Синтаксис приведен ниже.

```
for iterating_var1 in sequence: #outer loop
    for iterating_var2 in sequence: #inner loop
        #block of statements
#Other statements
```

Пример: Вложенный цикл for

```
rows = int(input("Введите количество строк:"))
# Внешний цикл напечатает нужно количество строк
for i in range(0,rows+1):
    # Внутренний цикл напечатает нужное количество звездочек в
    # каждой строке
    for j in range(i):
        print("*",end = ' ')
    print()
```

Вывод:

```
Enter the rows:5
*
**
***
****
*****
```

Пример: Программа для печати пирамиды чисел.

```
rows = int(input("Enter the rows"))
for i in range(0,rows+1):
    for j in range(i):
        print(i,end = ' ')
    print()
```

Вывод:

```
1
22
333
4444
55555
```

В приведенных примерах встречается ранее не рассмотренный дополнительный аргумент функции print – «end». Он позволяет поменять символ окончания строки. По умолчанию символом окончания является символ '\n', что означает закончить строку и начать новую. В данном случае мы меняем его на пустой символ: end = ". У функции print есть еще один дополнительный аргумент: sep. В случае, если для вывода на экран в функцию print передается несколько переменных, он позволяет задавать разделительный символ между этими элементами. По умолчанию sep равен пробелу.

3 Функции

Как было сказано ранее, функция – это заверченный кусочек программы, который выполняет определенные действия и может использоваться многократно. Заверченный означает, что у него есть начало (определенные входные параметры, или по другому начальные условия) и конец (есть определенный результат). До сих пор мы использовали уже готовые функции языка Python, которые за нас написали другие программисты. Однако невозможно написать заранее функции на все случаи жизни, поэтому рано или поздно придется создавать свои собственные функции, решающие специфические задачи. Ценность функций заключается в том, что они могут быть использованы многократно. Например, если мы написали функцию решения квадратного уравнения, то мы можем ее использовать, чтобы решить, 1,10,20, 100, 1000, 1000000 квадратных уравнений, используя при этом команду не «копировать/вставить», а просто указывая в коде название нашей функции и передавая ей нужные аргументы.

Для создания функции существует ключевое слово «def». Далее необходимо указать имя функции и в круглых скобках перечислить ее аргументы. Затем идет знак двоеточия и блок команд, которые выполняет функция. Если у функции есть какой-то результат, который необходимо использовать далее, пишется ключевое слово «return» и переменная, в которой хранится этот результат. Синтаксис создания функции следующий:

```
def my_function(parameters):  
    function_block  
    return expression
```

Например, функция, осуществляющая сложение, будет выглядеть следующим образом:

```
def my_sum(a, b):  
    res=a+b  
    return res
```

Важным является то, что описать функцию нам нужно до того, как мы будем ее использовать. Это связано с тем, что, как было сказано ранее, программа выполняется построчно, одна за другой. Поэтому, если в тексте программы

встречается вызов функции, которая не была описана ранее, возникает ошибка. Например:

```
def my_sum(a,b):  
    res=a+b  
    return res  
  
print(my_sum(1,2))
```

В результате на экране выведется число 3. Но если мы напишем следующим образом:

```
print(my_sum(1,2))  
  
def my_sum(a,b):  
    res=a+b  
    return res
```

то программа выведет сообщение об ошибке.

Функция может и не иметь ключевого слова return, если мы в дальнейшем не собираемся использовать результаты вычислений, полученных в этой функции. Например:

```
def print_my_sum(a,b):  
    res=a+b  
    print(res)  
  
print_my_sum(1,2)
```

В данном случае на экране также будет выведено число 3. Здесь мы не вызывали функция print из основной части нашей программы, так как функция print вызывается внутри print_my_sum. При создании своих функций нам необходимо самим продумывать логику работы нашей программы. Внутри самой функции мы можем использовать все описанные ранее команды: мы можем создавать переменные, списки, использовать операторы ветвления и циклы, вызывать другие функции и так далее. Также мы не ограничены в количестве и типе аргументов.

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство

для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Ознакомьте с теоретическим материалом.
2. Ответьте на контрольные вопросы, приведенные в конце работа.
3. Выполните индивидуальное задание, согласно вашему номеру варианта (Таблица 7.1).

Таблица 7.1 – Задания для самостоятельного выполнения:

№ варианта	Задачи для выполнения
1	1,9,6
2	1,10,7
3	1,7,5

4	1,6,8
5	1,6,7
6	1,7,10
7	1,9,10
8	1,10,8
9	1,8,9
10	1,5,6
11	1,9,5
12	1,5,7
13	1,7,5
14	1,3,9
15	1,6,7

Задача №1. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно, но известно, что ввод должен закончиться, как только пользователь введет символ конца ввода. Заранее этот символ неизвестен, пользователь должен ввести его сам. Напишите функцию, которая будет иметь один аргумент – символ конца ввода, создавать список введенных пользователем чисел и возвращать его, как только пользователь введет символ конца ввода.

Задача №2. Создайте функцию для решения квадратного уравнения. Используйте комментарии и список. Используйте функцию ввода, созданную вами в задаче №1

Задача №3. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет число 0. Напишите функцию вычисления арифметической суммы введенных чисел. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1.

Задача №4. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет символ 's'. Напишите функцию вычисления произведения введенных чисел. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1

Задача №5. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь

введет слово 'break'. Напишите функцию, которая подсчитывает, сколько чисел оказалось больше 10. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1

Задача №6. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет слово 'stop'. Необходимо создать функцию, которая будет считать количество четных и нечетных чисел. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1.

Задача №7. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет число 0. Напишите функцию, которая будет считать произведение четных и сумму нечетных чисел. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1.

Задача №8. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет число 0. Необходимо создать функцию, которая будет считать сумму квадратов этих чисел.

Задача №9. Напишите следующие функции:

- вычисление периметра прямоугольника;
- вычисление площади прямоугольника;
- вычисление длины диагонали прямоугольника.

Задача №10. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно. Ввод заканчивается, как только пользователь введет число 0. Необходимо четные числа сложить по правилам сложения чисел, а нечетные числа сложить по правилам сложения строк. Используйте комментарии, циклы, списки. Используйте функцию ввода, созданную вами в задаче №1.

Контрольные вопросы:

1. Что такое список в языке Python?

2. Как создать список?
3. Какие операции со списками вы знаете?
4. Как получить доступ к отдельным элементам списка?
5. Что такое комментарии в программе? Какие способы создания комментариев вы знаете?
6. Что такое циклы? Какие способы задания циклов вы знаете?
7. Какие операторы управления циклом вы знаете?
8. Какие дополнительные аргументы функции print вы знаете?
9. Что такое функция в языке Python? Для чего они нужны?
10. Как создать свою функцию?

Лабораторная работа №8. Работа с файлами, визуализация данных с помощью языка Python

Цель: Научиться считывать данные из файлов и сохранять результаты вычислений в файлы. Получить общие представления о принципах визуализации данных с помощью средств языка Python.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1 Словари и кортежи

На прошлом занятии был рассмотрен такой полезный тип данных как список. Помимо списков в языке Python существуют еще несколько удобных встроенных типов данных. К ним относятся словари и кортежи.

Словари (dict) – это одна из наиболее часто используемых структур данных, позволяющая хранить объекты, для доступа к которым используется ключ. Данные в словаре хранятся в формате ключ – значение. Если вспомнить такую структуру как список, то доступ к его элементам осуществляется по индексу, который представляет собой целое число. Причем индексы создаются автоматически, независимо от нас. В словаре аналогом индекса является ключ, при этом ответственность за его создание ложится на программиста.

Пустой словарь можно создать, используя функцию dict(), либо просто указав пустые фигурные скобки.

```
>>> d1 = dict()
>>> print(type(d1))
<class 'dict'>
>>> d2 = {}
>>> print(type(d2))
<class 'dict'>
```

Если необходимо создать словарь с заранее подготовленным набором данных, то можно использовать один из перечисленных выше подходов, но с перечислением групп ключ-значение.

```
>>> d1 = dict(Ivan="manager", Mark="worker")
>>> print(d1)
{'Mark': 'worker', 'Ivan': 'manager'}
```

```
>>> d2 = {"A1": "123", "A2": "456"}
>>> print(d2)
{'A2': '456', 'A1': '123'}
```

Чтобы добавить элемент в словарь нужно указать новый ключ и значение.

```
>>> d1 = {"Russia": "Moscow", "USA": "Washington"}
>>> d1["China"] = "Beijing"
>>> print(d1)
{'Russia': 'Moscow', 'China': 'Beijing', 'USA': 'Washington'}
```

Для удаления элемента из словаря можно воспользоваться командой `del`, либо функцией словаря `pop()`. Пример:

```
>>> d2 = {"A1": "123", "A2": "456"}
>>> del d2["A1"]
>>> print(d2)
{'A2': '456'}
```

```
>>> d = {"A1": "123", "A2": "456"}
>>> d.pop("A1")
'123'
>>> print(d)
{'A2': '456'}
```

Как видно из примера, функция `pop` помимо удаления еще и в качестве результата выполнения возвращает значение удаленного элемента. В некоторых ситуациях это бывает полезно.

Доступ к элементу словаря, осуществляется как же как доступ к элементу списка, только в качестве индекса указывается ключ.

```
>>> d1 = {"Russia": "Moscow", "USA": "Washington"}
>>> d1["Russia"]
'Moscow'
```

Проверить наличие ключа в словаре можно с помощью оператора `in`:

```
>>> d2 = {"A1": "123", "A2": "456"}
>>> "A1" in d2
True
>>> "A3" in d2
False
```

У словарей доступен следующий набор дополнительных функций:

– **`clear()`**. Удаляет все элементы словаря. Пример:

```
>>> d2 = {"A1": "123", "A2": "456"}
>>> print(d2)
{'A2': '456', 'A1': '123'}
```

```
>>> d2.clear()  
>>> print(d2)  
{}
```

– **get(key)**. Возвращает значение из словаря по ключу key. Пример:

```
>>> d = {"A1": "123", "A2": "456"}  
>>> d.get("A1")  
'123'
```

– **keys()**. Возвращает все ключи словаря.

```
>>> d = {"A1": "123", "A2": "456"}  
>>> d.keys()  
dict_keys(['A2', 'A1'])
```

Функция keys() может быть полезна, когда нужно со всеми ключами словаря провести какие-то действия (самый простой вариант – вывести на экран). В это случае функция keys() может быть использована в цикле

```
d = {"A1": "123", "A2": "456"}  
for k in d.keys():  
    print(k)
```

На экране будут выведены строки «A1» и «A2».

– **values()**. Функция возвращает значения элементов словаря:

```
>>> d = {"A1": "123", "A2": "456"}  
>>> d.values()  
dict_values(['456', '123'])
```

Аналогично с функцией keys(), функция values() будет полезна, когда нужно произвести какие-то действия со всеми значениями словаря через цикл:

```
d = {"A1": "123", "A2": "456"}  
for k in d.values():  
    print(k)
```

В результате выполнения на экран будет выведены значения «123» и «456».

– **items()**. Возвращает элементы словаря (ключ, значение). По сути является комбинацией двух предыдущих функций. Поэтому будет полезна, когда через цикл необходимо выполнить какие-то действия со всеми парами «ключ» - «значение» словаря. Например:

```
d = {"A1": "123", "A2": "456"}  
for it in d.items():  
    print(it[0], "->", it[1])
```

На экран будет выведен следующий текст:

```
A1 -> 123
```

```
A2 -> 456
```

Обратите внимание, что для вывода ключа мы обращаемся к элементу по индексу 0, а для вывод значения – по индексу 1.

В качестве ключа могут выступать практически любые типы данных, за некоторым исключением. Например, целые и дробные числа, строки, логические переменные могут выступать в качестве ключа, а вот списки в качестве ключа выступать не могут. Дело в том, что считается, что список в любой момент времени может измениться (в него могут быть добавлены новые, удалены старые элементы и так далее). Чтобы исправить эту ситуацию, в языке Python существует очень похожий на списки тип данных, который называется кортежем.

Кортеж – последовательность элементов, которые разделены между собой запятой и заключены в скобки, это неизменяемый упорядоченный тип данных. Грубо говоря, кортеж - это список, который нельзя изменить. То есть, в кортеже есть только права на чтение. Поэтому элементы кортежа должна добавляться в него непосредственно в момент его создания. Создаются кортежи почти точно так же, как и списки, только скобки ставятся не квадратные, а круглые.

Например:

```
T = ('login', 'password')
```

Точно также, как строка может быть преобразована в число, а число в строку, ранее созданный список может быть преобразован в кортеж, а кортеж может быть преобразован в список. Для создания кортежа из списка существует функция `tuple()`, а для создания списка из кортежа существует функция `list()`:

```
sp=['a','b','c']  
tp=(1,2,3)  
sp2=list(tp)  
tp2=tuple(sp)
```

Для доступа к элементам кортежа точно также используются целочисленные индексы.

2 Работа с файлами

До сих пор для ввода и вывода данных мы использовали экран и клавиатуру. Но довольно часто получать данные приходится не с клавиатуры, а из файлов. Для открытия файла используется функция `open()`, которая возвращает файловый объект. Наиболее часто используемый вид данной функции выглядит так

<code>open(имя_файла, режим_доступа)</code>

Имя файла может быть относительным (относительно исполняемой программы) или абсолютным (полный путь к файлу с указанием буквы диска).

Для указания режима доступа используются следующие символы:

‘r’ – открыть файл для чтения;

‘w’ – открыть файл для записи;

‘x’ – открыть файл с целью создания, если файл существует, то вызов функции `open` завершится с ошибкой;

‘a’ – открыть файл для записи, при этом новые данные будут добавлены в конец файла, без удаления существующих;

‘b’ – бинарный режим;

‘t’ – текстовый режим;

‘+’ – открывает файл для обновления.

По умолчанию файл открывается на чтение в текстовом режиме.

У файлового объекта есть следующие атрибуты.

file.closed – возвращает `true` если файл закрыт и `false` в противном случае;

file.mode – возвращает режим доступа к файлу, при этом файл должен быть открыт;

file.name – имя файла.

<pre>>>> f = open("test.txt", "r") >>> print("file.closed: " + str(f.closed)) file.closed: False >>> print("file.mode: " + f.mode) file.mode: r >>> print("file.name: " + f.name)</pre>

```
file.name: test.txt
```

Для закрытия файла используется метод `close()`.

Чтение данных из файла осуществляется с помощью методов `read(размер)` и `readline()`.

Метод `read(размер)` считывает из файла определенное количество символов, переданное в качестве аргумента. Если использовать этот метод без аргументов, то будет считан весь файл.

```
>>> f = open("test.txt", "r")
>>> f.read()
'1 2 3 4 5\nWork with file\n'
>>> f.close()
```

В качестве аргумента метода можно передать количество символом, которое нужно считать.

```
>>> f = open("test.txt", "r")
>>> f.read(5)
'1 2 3'
>>> f.close()
```

Метод `readline()` позволяет считать строку из открытого файла.

```
>>> f = open("test.txt", "r")
>>> f.readline()
'1 2 3 4 5\n'
>>> f.close()
```

Построчное считывание можно организовать с помощью оператора `for`.

```
>>> f = open("test.txt", "r")
>>> for line in f:
...     print(line)
...
1 2 3 4 5
Work with file
>>> f.close()
```

Для записи данных файл используется метод `write(строка)`, при успешной записи он вернет количество записанных символов.

```
>>> f = open("test.txt", "a")
>>> f.write("Test string")
11
>>> f.close()
```

2.1 Текстовые файлы с разделителями

Довольно часто используются так называемые текстовые файлы с разделителями. В каждой строке такого файла записан не один параметр, а несколько, отделенные друг от друга пробелом, либо другим символом, например, ';' или ','. При чтении таких файлов данные для дальнейшей обработки удобно преобразовать в список или словарь. Для этого могут быть полезны функции строковые функции `split()` и `strip()`.

Функция `split` сканирует всю строку и разделяет ее в случае нахождения разделителя.

В строке должен быть как минимум один разделитель. Им может выступать в том числе и символ пробела. Пробел – разделитель по умолчанию. Если параметр не задать, то разделение будет выполнено именно по символу пробела.

Синтаксис функции следующий:

```
string.split(separator*, maxsplit*)
```

Параметр `separator` — необязательный, но он позволяет задать разделитель вручную.

Параметр `maxsplit` определяет максимальное количество разделений. Значение по умолчанию – -1, будут выполнены все разделения.

Если не передать параметр разделителя, то `.split()` выполнит разделение по пробелу.

```
my_st = "Пример строки Python"  
print(my_st.split())
```

Код вернет: ['Пример', 'строки', 'Python'].

Разделителем может выступать и любой другой код, например, запятая (","). Этот код вернет список строк, которые изначально были окружены запятыми.

```
my_st = "Например, строка Python"  
print(my_st.split(", "))
```

Вывод: ['Например', ' строка Python']. Результатом является список подстрок, разделенных по запятым в исходной строке.

Еще одна проблема при чтении данных из файлов заключается в том, что при чтении строк из файлов через цикл, в конце каждой строки также считывается символ конца строки ('\n'). При дальнейшей работе с переменными этот символ может мешать. Для его исключения существует строковый метод strip().

Метод strip() возвращает копию строки, удаляя как начальные, так и конечные символы (в зависимости от переданного строкового аргумента).

Метод удаляет символы как слева, так и справа в зависимости от аргумента (строка, определяющая набор символов, которые необходимо удалить).

Синтаксис метода:

```
string.strip([chars])
```

chars (необязательно) – строка, определяющая набор символов, которые нужно удалить из левой и правой части строки. Если аргумент chars не указан, все начальные и конечные пробелы удаляются из строки.

Помимо метода strip() существует еще две аналогичных функции: lstrip() и rstrip(). Функция lstrip() удаляет элементы только с начала строки, а функция rstrip() удаляет элементы только с конца строки.

Учитывая все вышесказанное, следующий пример демонстрирует чтение данных из текстового файла с разделителями в словарь.

```
f = open("D:\\test.txt", "r")
dc={}
for line in f:
    line=line.strip()
    ls=line.split(';')
    dc[ls[0]]=ls[1]
f.close()
print(dc)
```

3 Модули и пакеты

Модули и пакеты значительно упрощают работу программиста. Функции, объекты и константы, которыми приходится часто пользоваться можно упаковать в модуль, и, в дальнейшем, загружать его в свои программы при необходимости.

Под модулем в Python понимается файл с расширением .py. Модули предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т.п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы. Стоит заметить, что модули могут быть написаны не только на языке Python, но и на других языках (например C).

Самый простой способ импортировать модуль в Python это воспользоваться конструкцией:

```
import имя_модуля
```

Импорт и использование модуля math, который содержит математические функции, будет выглядеть вот так.

```
>>> import math
>>> math.factorial(5)
120
```

За один раз можно импортировать сразу несколько модулей, для этого их нужно перечислить через запятую после слова import:

```
import имя_модуля1, имя_модуля2
```

```
>>> import math, datetime
>>> math.cos(math.pi/4)
0.707106781186547
>>> datetime.date(2017, 3, 21)
datetime.date(2017, 3, 21)
```

Если вы хотите задать псевдоним для модуля в вашей программе, можно воспользоваться вот таким синтаксисом:

```
import имя_модуля as новое_имя
```

```
>>> import math as m
>>> m.sin(m.pi/3)
0.866025403784438
```

Используя любой из вышеперечисленных подходов, при вызове функции из импортированного модуля, вам всегда придется указывать имя модуля (или псевдоним). Для того, чтобы этого избежать делайте импорт через конструкцию from ... import...

```
from имя_модуля import имя_объекта
```

```
>>> from math import cos
>>> cos(3.14)
0.999998731727539
```

При этом импортируется только конкретный объект (в нашем примере: функция `cos`), остальные функции недоступны, даже если при их вызове указать имя модуля.

Для импортирования нескольких функций из модуля, можно перечислить их имена через запятую.

```
from имя_модуля import имя_объекта1, имя_объекта2
```

```
>>> from math import cos, sin, pi
>>> cos(pi/3)
0.5000000000000000
>>> sin(pi/3)
0.866025403784438
```

Импортируемому объекту можно задать псевдоним.

```
from имя_модуля import имя_объекта as псевдоним_объекта
```

```
>>> from math import factorial as f
>>> f(4)
24
```

Если необходимо импортировать все функции, классы и т.п. из модуля, то воспользуйтесь следующей формой оператора `from ... import ...`

```
from имя_модуля import *
```

```
>>> from math import *
>>> cos(pi/2)
6.123233995736766e-17
>>> sin(pi/4)
0.707106781186547
>>> factorial(6)
720
```

3.1 Пакеты в Python

Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл `__init__.py`. Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку).

Для импортирования пакетов используется тот же синтаксис, что и для работы с модулями.

Рассмотрим следующую структуру пакета:

```
fincalc
|-- __init__.py
|-- simper.py
|-- compper.py
|-- annuity.py
```

Рисунок 8.1 – Структура типового пакета

Пакет `fincalc` содержит в себе модули для работы с простыми процентами (`simper.py`), сложными процентами (`compper.py`) и аннуитетами (`annuity.py`).

Для использования функции из модуля работы с простыми процентами, можно использовать один из следующих вариантов:

```
import fincalc.simper
fv = fincalc.simper.fv(pv, i, n)
import fincalc.simper as sp
fv = sp.fv(pv, i, n)
from fincalc import simper
fv = simper.fv(pv, i, n)
```

4 Визуализация данных

Язык Python обрел свою популярность в том числе благодаря множеству удобных библиотек для визуализации данных. Одной из наиболее распространенных таких библиотек является библиотека `Matplotlib`. Для ее установки необходимо через командную строку операционной системы выполнить следующую команду:

```
pip install matplotlib
```

При выполнении этой команды будет автоматически скачана последняя версия программы и записана в нужный каталог системы. Далее необходимо импортировать функции для работы с графикой. Библиотека Matplotlib имеет несколько модулей. Практически все задачи, связанные с построением графиков, можно решить, используя возможности, которые предоставляет модуль pyplot. Для его подключения в программу его необходимо импортировать, используя соответствующую команду:

```
import matplotlib.pyplot as plt
```

Здесь мы задаем псевдоним plt, который позволяет сократить объем кода и сделать его более читаемым. Вместо слова plt вы можете придумать любой другой псевдоним.

Библиотека Matplotlib имеет огромное количество функций и их параметров. Рассмотреть все возможности библиотеки в рамках одного занятия не предоставляется возможным. Целью настоящей лабораторной работы является получение общих представлений об устройстве библиотеки и ее основных функциях, достаточных для дальнейшего ее самостоятельного изучения с помощью официальной документации.

Рассмотрим основные термины и понятия, касающиеся изображения графика, с которыми вам необходимо будет познакомиться, для того, чтобы в дальнейшем у вас не было трудностей при работе с документацией по библиотеке matplotlib.

Корневым элементом при построения графиков в системе Matplotlib является Фигура (Figure). Все, что нарисовано на рисунке является элементами фигуры. Рассмотрим ее составляющие более подробно.

4.1 График

На рисунке представлены два графика – линейный и точечный. Matplotlib предоставляет огромное количество различных настроек, которые можно использовать для того, чтобы придать графику вид, который вам

нужен: цвет, толщина и тип линии, стиль линии и многое другое, все это вы можете найти в документации к библиотеке.

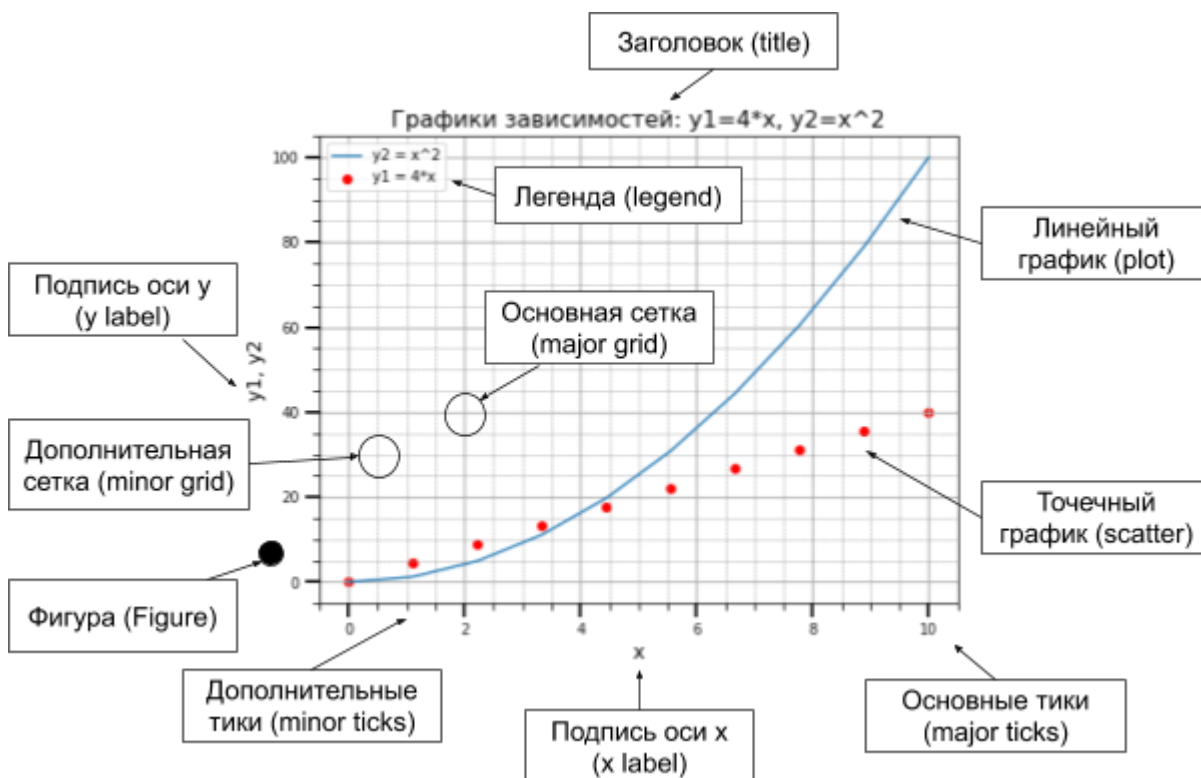


Рисунок 8.2 – Основные графические элементы библиотеки Matplotlib

4.2 Оси

Вторым, после непосредственно самого графика, по важности элементом фигуры являются оси. Для каждой оси можно задать метку (подпись), основные (major) и дополнительные (minor) тики, их подписи, размер и толщину, также можно задать диапазоны по каждой из осей.

4.3 Сетка и легенда

Следующими элементами фигуры, которые значительно повышают информативность графика являются сетка и легенда. Сетка также может быть основной (major) и дополнительной (minor). Каждому типу сетки можно задавать цвет, толщину линии и тип. Для отображения сетки и легенды используются соответствующие команды.

Основным элементом изображения, которое строит pyplot является Фигура (Figure), на нее накладываются один или более графиков, осей, надписей и т.п. Для построения графика используется команда plot(). В самом минимальном варианте можно ее использовать без параметров:

```
import matplotlib.pyplot as plt
plt.plot()
plt.show()
```

Первая функция (plot) строит график, вторая функция (show) отображает его на экране.

В результате выполнения кода будет выведено пустое поле:

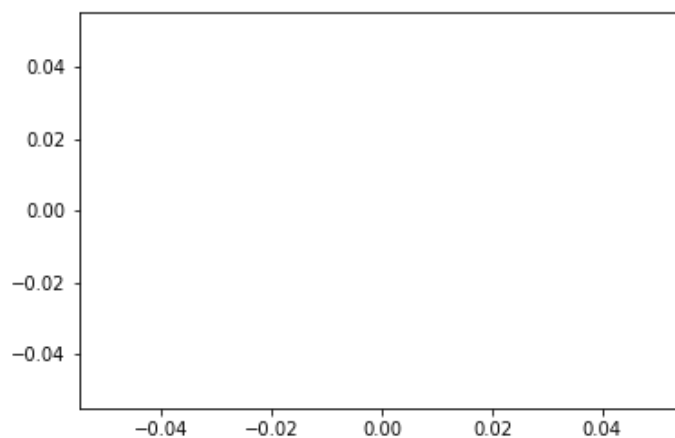


Рисунок 8.3 – Построение пустого графика

Если в качестве параметра функции plot() передать список, то значения из этого списка будут отложены по оси ординат (ось y), а по оси абсцисс (ось x) будут отложены индексы элементов массива:

```
plt.plot([1, 7, 3, 5, 11, 1])
```

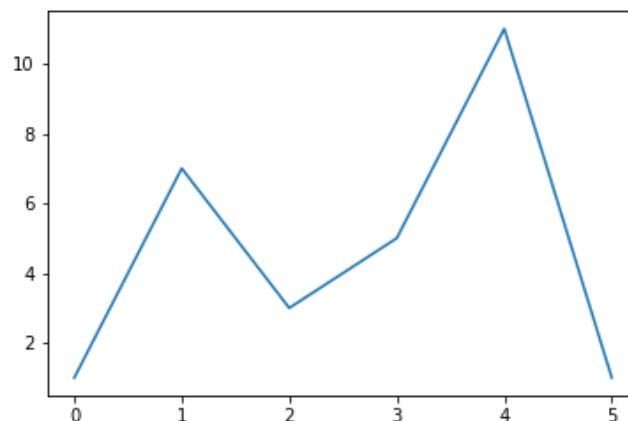


Рисунок 8.4 – Построение графика на основе списка

Для того, чтобы задать значения по осям x и y необходимо в plot() передать два списка:

```
plt.plot([1, 5, 10, 15, 20], [1, 7, 3, 5, 11])
```

В результате получим следующий график:

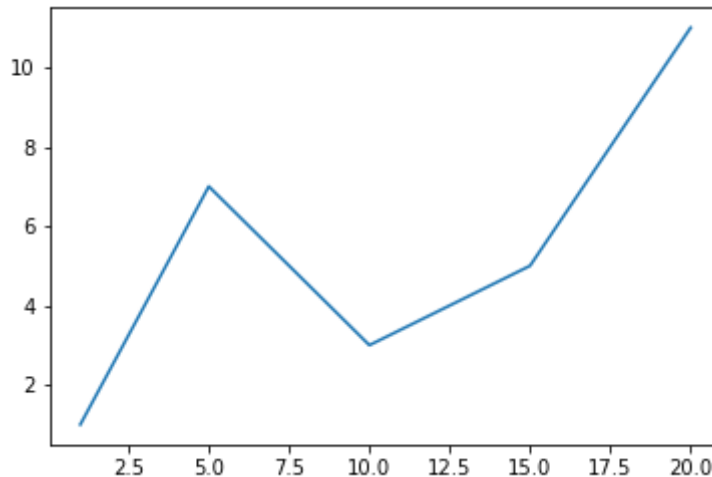


Рисунок 8.5 – Построение графика на основе двух списков

Таким образом, если нам нужно вывести какие-то данные, очевидно, нам нужно предварительно их подготовить. Например, если мы хотим построить график синусоиды, можно использовать рассмотренные ранее циклы и словари:

```
import math
import matplotlib.pyplot as plt
a=dict()
for i in range(0, 360):
    a[i]=math.sin(i/180*math.pi)

plt.plot(a.keys(),a.values())
plt.show()
```

В результате будет создан следующий график:

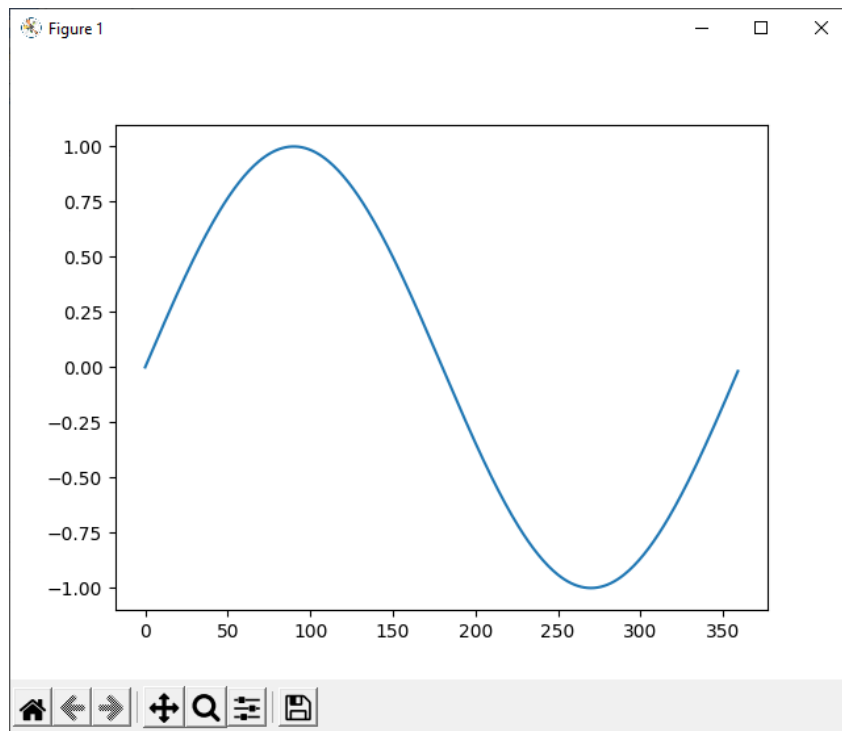


Рисунок 8.6 – Пример построение графика синусоиды

Здесь встречается не рассмотренная ранее библиотека `math`. Она содержит множество наиболее часто используемых математических функций, в том числе необходимую нам тригонометрическую функцию синуса. Список всех доступных функций библиотеки `math` можно найти в документации на официальном сайте проекта Python.

4.4 Текстовые надписи на графике

Наиболее часто используемые текстовые надписи на графике это:

- наименование осей;
- наименование самого графика;
- текстовое примечание на поле с графиком;
- легенда.

Рассмотрим кратко данные элементы.

Для задания подписи оси `x` используется функция `xlabel()`, оси `y` – `ylabel()`. Разберемся с аргументами данных функций.

Аргументов у этих функций довольно много и они позволяют достаточно тонко настроить внешний вид надписей. В рамках настоящей работы мы рассмотрим только основные из них. Полный список всегда можно посмотреть в официальной документации. Для функций `xlabel()/ylabel()` основными являются следующие аргументы:

- Текст подписи: **str** (первый аргумент);
- Расстояние между областью графика, включающую оси, и меткой:

labelpad (численное значение, по умолчанию отсутствует).

Функции `xlabel()/ylabel()` принимают в качестве аргументов параметры класса `matplotlib.text.Text`, некоторые из них нам могут пригодиться:

- Размер шрифта: `fontsize` или `size` – число либо значение из списка: {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}.
- Стиль шрифта: `fontstyle` – значение из списка: {'normal', 'italic', 'oblique'};

Толщина шрифта: `fontweight` – число в диапазоне от 0 до 1000 либо значение из списка: {'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}.

Цвет шрифта – `color`: один из доступных способов определения цвета (см. Цвет линии в официальной документации).

Пример использования:

```
plt.xlabel('Day', fontsize=15, color='blue')
```

4.5 Заголовок графика

Для задания заголовка графика используется функция `title()`:

```
plt.title('Chart price', fontsize=17)
```

Из параметров отметим следующие:

- Текст заголовка (первый параметр, `label`, тип - `str`).
- Выравнивание заголовка: `loc`, значение из набора {'center', 'left', 'right'}

Для функции `title()` также доступны параметры конструктора класса `matplotlib.text.Text`, часть из которых рассмотрены в описании аргументов функций `xlabel()` / `ylabel()`.

4.6 Текстовое примечание

За размещение текста на поле графика отвечает функция `text()`, которой вначале передаются координаты позиции надписи, после этого – текст самой надписи.

```
plt.text(1, 1, 'type: Steel')
```

4.7 Легенда

Легенда будет размещена на графике, если вызвать функцию `legend()`. Функция позволяет настроить расположение легенды, размер шрифта и многие другие параметры. В рамках настоящей работы мы не будем рассматривать аргументы функции, при желании вы можете изучить их самостоятельно.

Давайте немного наведем красоту на наших графиках, используя рассмотренные функции:

```
import math
import matplotlib.pyplot as plt
a=dict()

for i in range(0, 360):
    a[i]=math.sin(i/180*math.pi)

plt.title("y = sin(x)") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки

plt.plot(a.keys(),a.values())
plt.legend(['Синус'])
plt.show()
```

Результат работы кода:

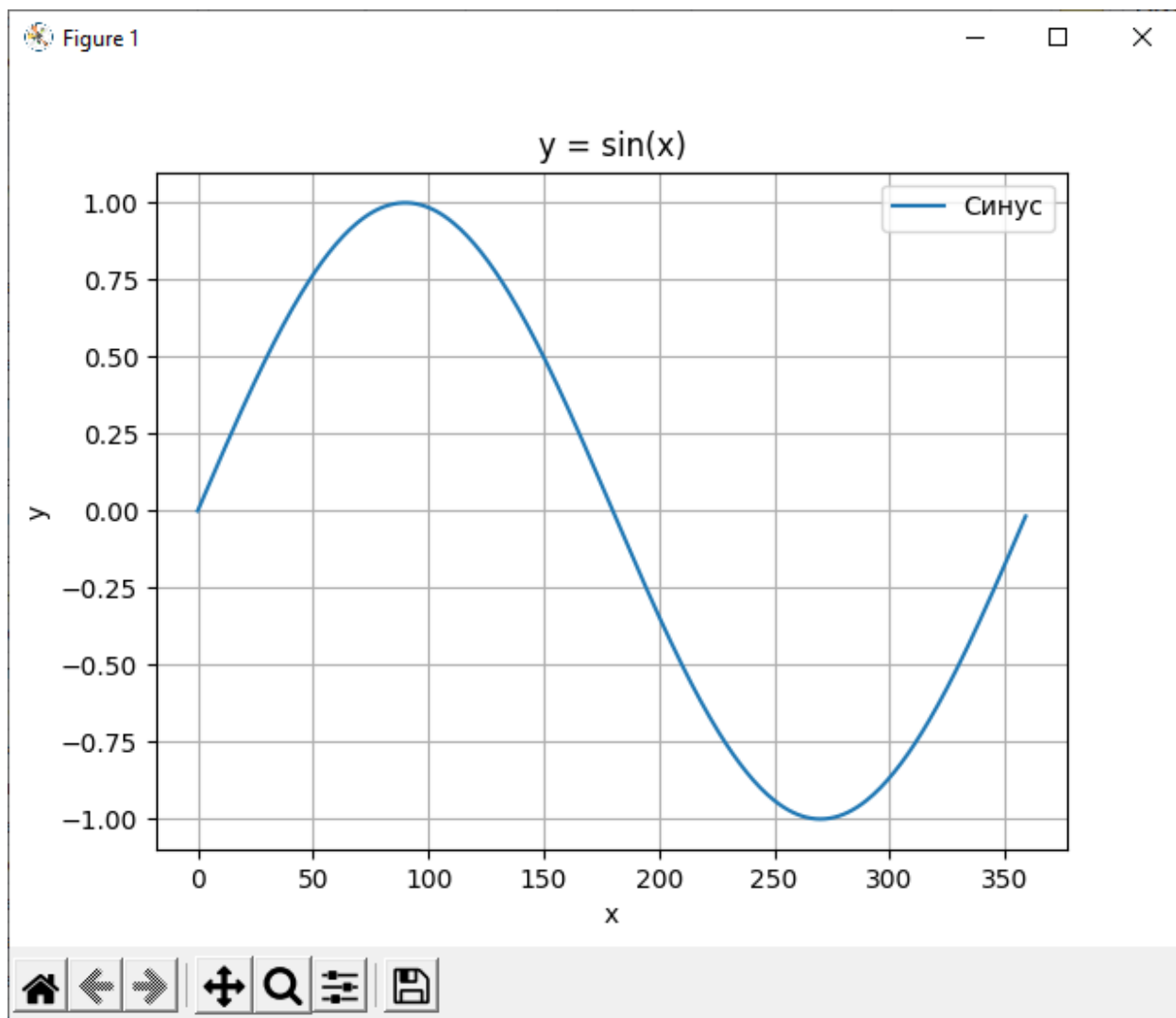


Рисунок 8.7 – Настройка параметров графика

Помимо рассмотренных функций, здесь мы использовали еще одну функцию `grid`. Она позволяет отобразить сетку на графике. Функция позволяет настроить цвет сетки, стиль линии, их толщину, шаг сетки, отображения по осям и так далее.

Например,

```
plt.grid(color='r', linestyle='--', linewidth=2)
```

нарисует следующую сетку:

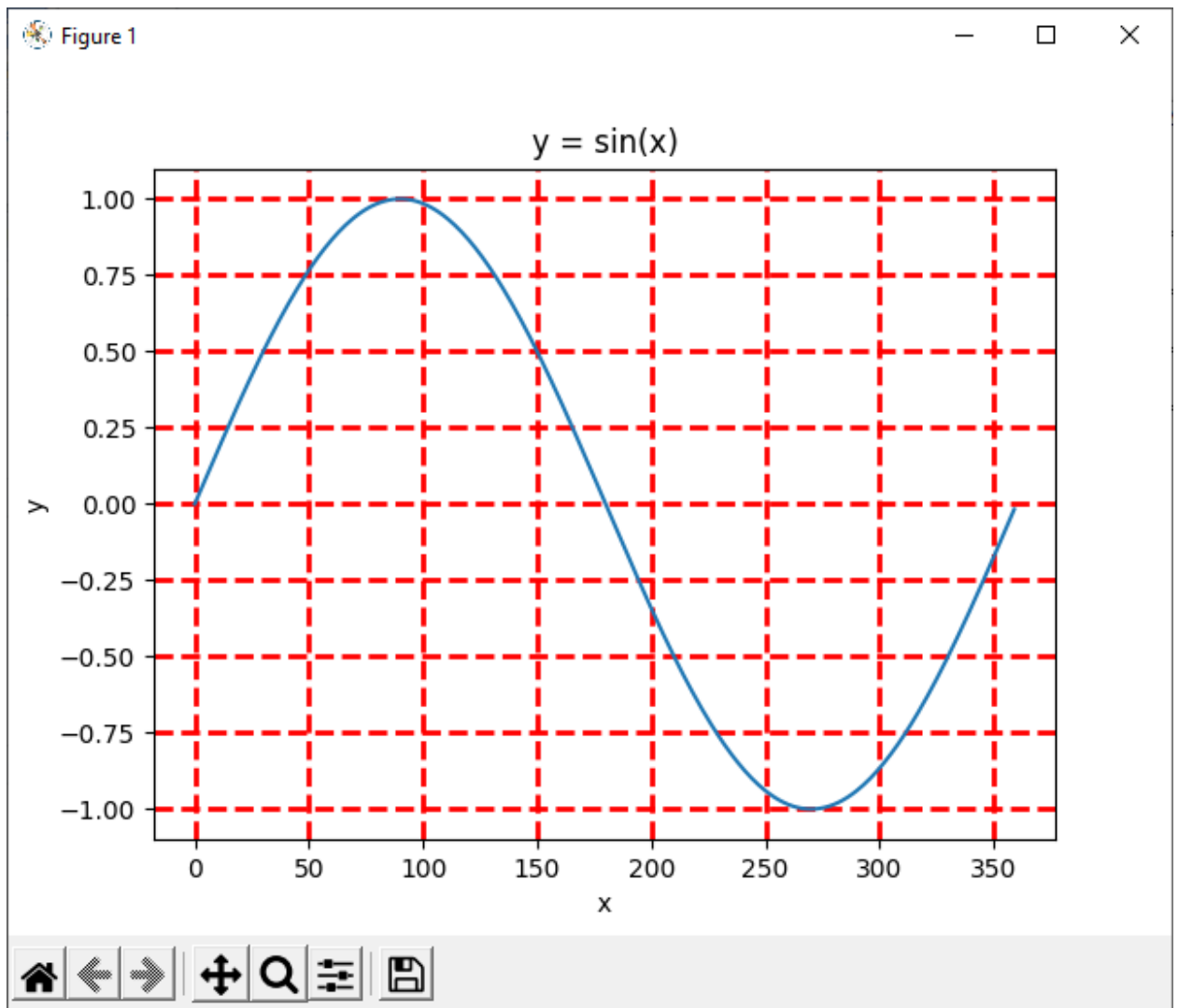


Рисунок 8.8 – Настройка параметров сетки

Список всех параметров функции `grid` и описание множества других полезных функции пакета `matplotlib.pyplot` можно найти в документации на официальном сайте библиотеки `matplotlib`.

Кстати функция `plot` тоже позволяет настраивать цвет, толщину и стиль линии с помощью этих же параметров (`color`, `linestyle`, `linewidth`).

Если нам нужно построить два графика на одном листе, мы просто вызываем второй раз функцию `plot`.

```
import math
import matplotlib.pyplot as plt
a=dict()
b=dict()
for i in range(0, 360):
    a[i]=math.sin(i/180*math.pi)
    b[i]=math.cos(i/180*math.pi)
```

```
plt.title("y = sin(x)") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid(color='r', linestyle='--', linewidth=2) #
# включение отображение сетки

plt.plot(a.keys(),a.values(),linestyle = '--',color = (0, 0,
0, 1),linewidth = 12)
plt.plot(b.keys(),b.values())
plt.show()
```

В результате выполнению получим следующий график:

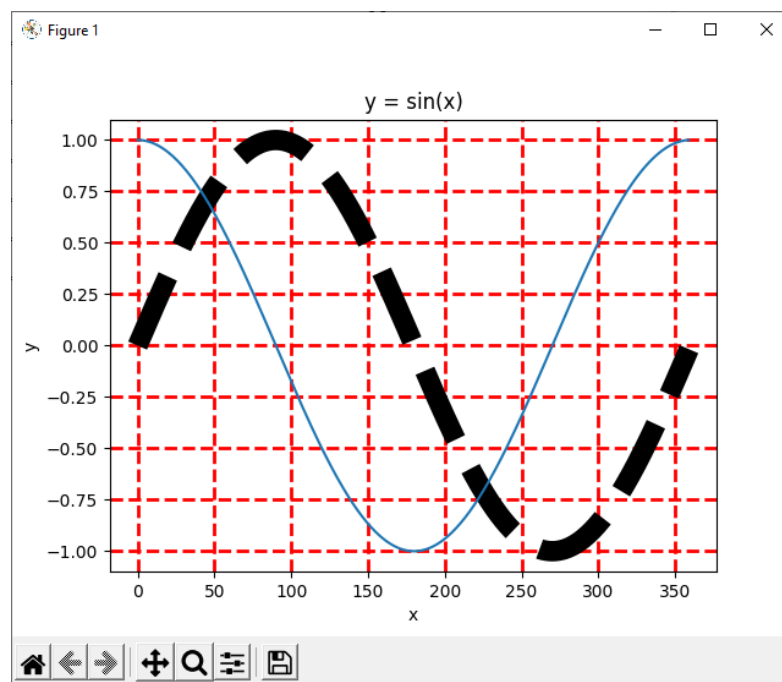


Рисунок 8.9 – Отображение двух графиков в одной рабочей области

Библиотека также позволяет отобразить два графика не накладывая их друг на друга.

```
import math
import matplotlib.pyplot as plt
a=dict()
b=dict()
for i in range(0, 360):
    a[i]=math.sin(i/180*math.pi)
    b[i]=math.cos(i/180*math.pi)

plt.figure(figsize=(9, 9))
plt.subplot(2, 1, 1)
plt.title("y = sin(x)") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
```

```
plt.grid(color='r', linestyle='--', linewidth=2) #
включение отображение сетки
plt.plot(b.keys(),b.values())
plt.subplot(2, 1, 2)
plt.title("y = cos(x)") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid(color='r', linestyle='--', linewidth=2) #
включение отображение сетки
plt.plot(a.keys(),a.values(),linestyle = '--',color = (0, 0,
0, 1),linewidth = 12)
plt.show()
```

Результатом выполнения кода будет следующий график:

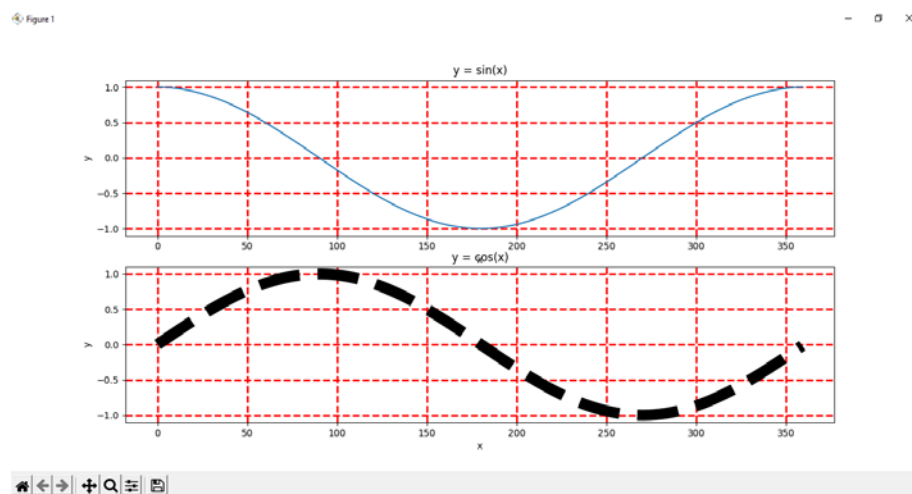


Рисунок 8.10 – Отображение графиков в разных рабочих областях

Здесь мы воспользовались новыми функциями:

`figure()` – функция для задания глобальных параметров отображения графиков. В нее, в качестве аргумента, мы передаем кортеж, определяющий размер общего поля (размер указывается в дюймах; при необходимости указания в сантиметрах, необходимо разделить нужные размеры на 2.54).

`subplot()` – функция для задания местоположения поля с графиком. Существует несколько способов задания областей для вывода через функцию `subplot()` мы воспользовались следующим: первый аргумент – количество строк, второй – столбцов в формируемом поле, третий – индекс (номер поля, считаем сверху вниз, слева направо).

4.8 Тип графика

До этого момента мы работали только с линейными графиками, функция `plot()` позволяет задать тип графика: линейный либо точечный, при этом для точечного графика можно указать соответствующий маркер. Приведем пару примеров:

```
plt.plot(x, y, 'ro')
```

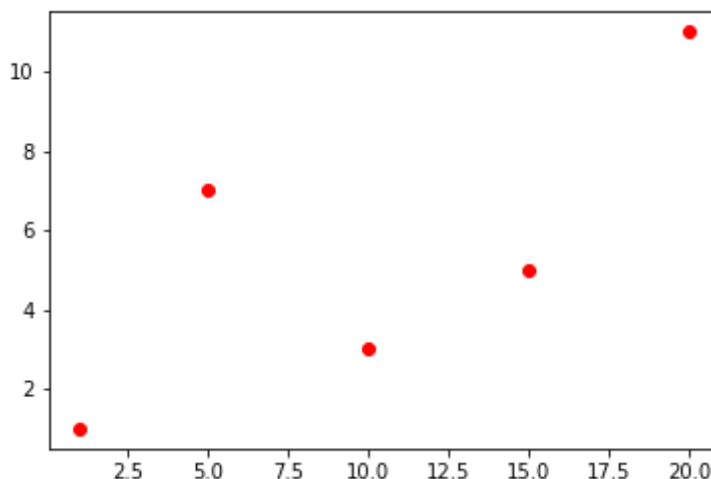


Рисунок 8.11 – Точечный график с круговым маркером

```
plt.plot(x, y, 'bx')
```

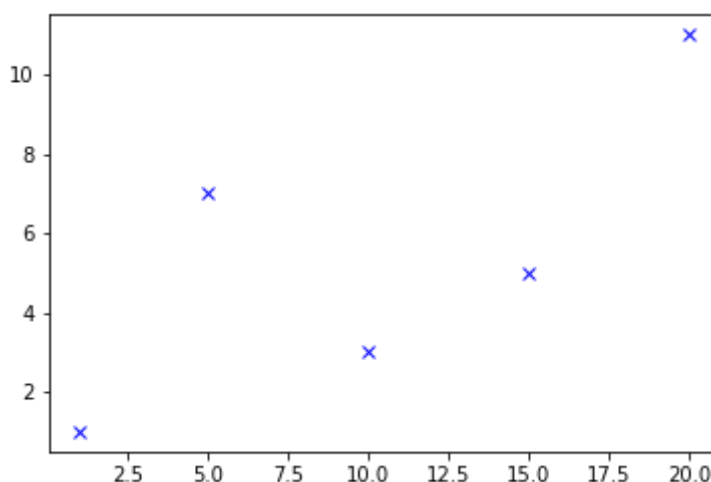


Рисунок 8.12 – Точечный график с маркером типа x

Первая буква задает цвет маркера ('b' – синий; 'g' – зеленый, 'r' – красный, 'c' – бирюзовый; 'm' – фиолетовый (пурпурный); 'y' – желтый; 'k' –

– черный; ‘w’ – белый), вторая буква задает тип маркера: ‘.’ – точка, ‘,’ – пиксель; ‘o’ – окружность; ‘v’ – треугольник, направленный вниз; ‘^’ – треугольник, направленный вверх; ‘<’ – треугольник, направленный влево; ‘>’ – треугольник, направленный вправо; ‘s’ – квадрат; ‘p’ – пятиугольник; ‘*’ – звезда; ‘h’ – шестиугольник; ‘+’ – плюс; ‘x’ – X-образный маркер; ‘D’ – ромб; ‘|’ – вертикальная линия; ‘_’ – горизонтальная линия. Размер маркера можно менять.

4.9 Столбчатые и круговые диаграммы

Помимо линейных графиков, библиотека Matplotlib позволяет также строить столбчатые и линейные диаграммы. Для их построения используются функции:

bar() – для построения вертикальной диаграммы;

barh() – для построения горизонтальной диаграммы;

pie() – для построения круговых диаграмм.

Примеры:

```
import matplotlib.pyplot as plt

fruits=["apple","peach","orange","bannana","melon"]
counts = [34, 25, 43, 31, 17]

plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
plt.show()
```

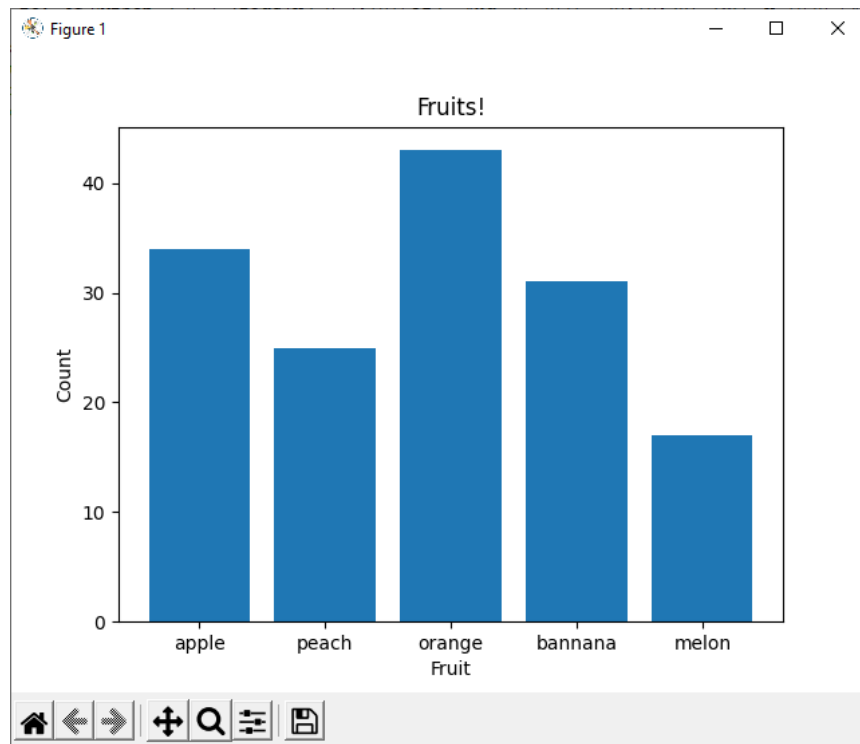


Рисунок 8.13 – Столбчатая диаграмма

```
import matplotlib.pyplot as plt
fruits=["apple","peach","orange","bannana","melon"]
counts = [34, 25, 43, 31, 17]
plt.barh(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
plt.show()
```

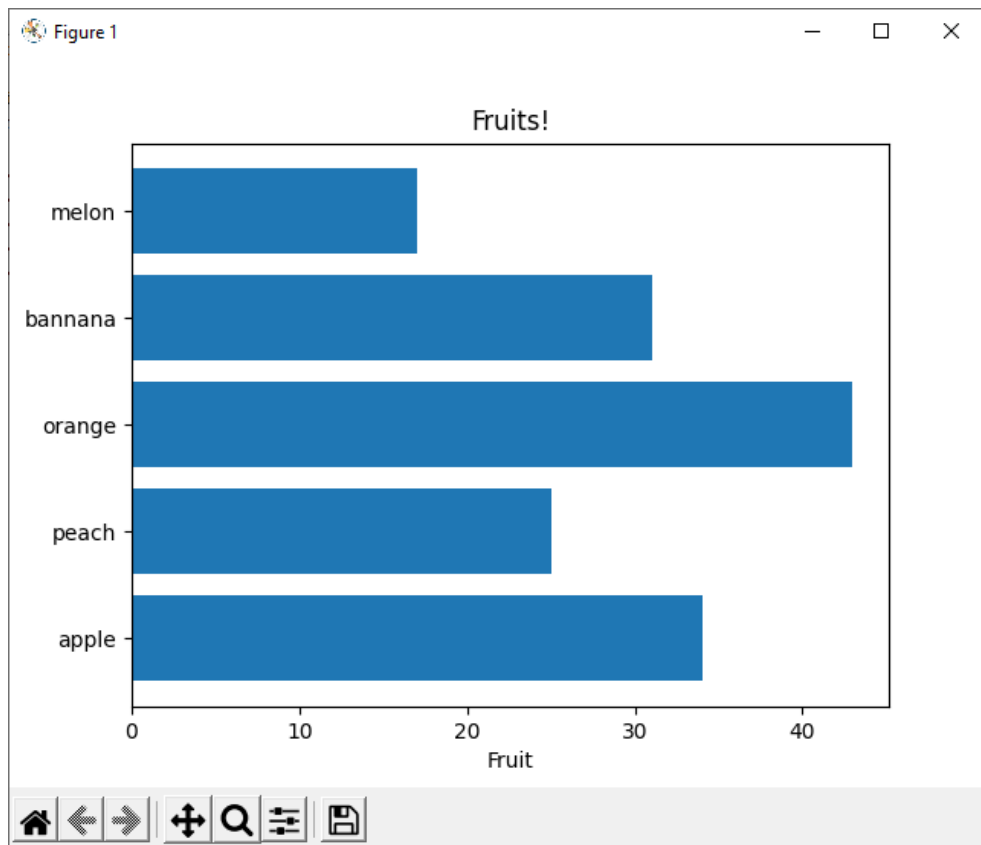


Рисунок 8.14 – Вертикальная столбчатая диаграмма

```
import matplotlib.pyplot as plt
fruits=["apple","peach","orange","bannana","melon"]
counts = [34, 25, 43, 31, 17]
plt.pie(counts, labels=fruits)
plt.title("Fruits!")
plt.show()
```

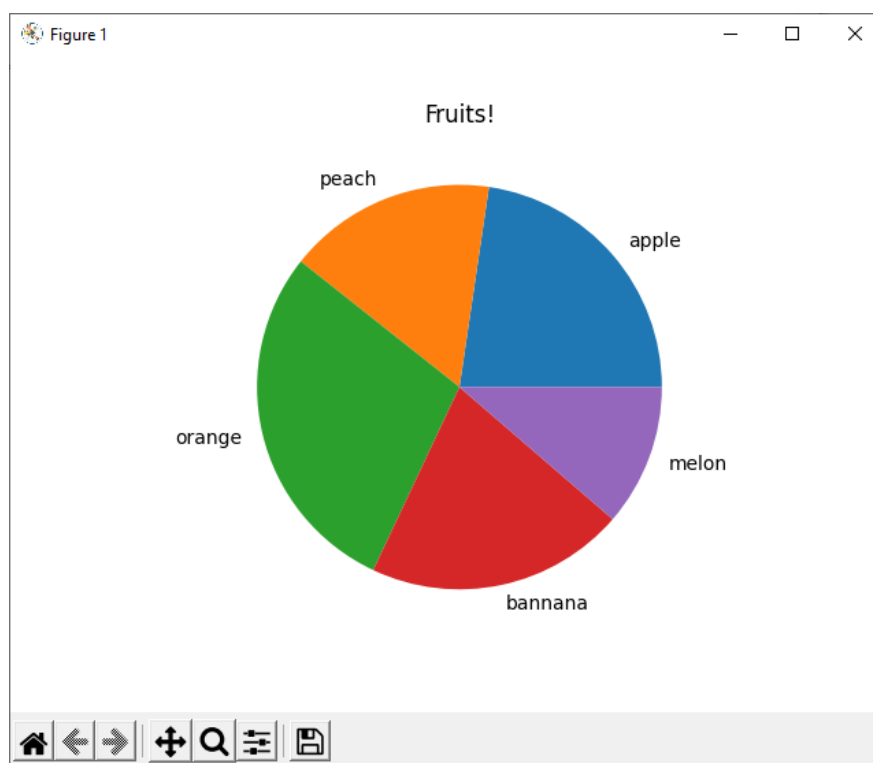


Рисунок 7.15 – Круговая столбчатая диаграмма

Также как и линейные графики, диаграммы имеют множество настроек. Рассмотреть их все в рамках одной лабораторной работы не предоставляется возможным, но вы можете это сделать самостоятельно, используя официальную документацию библиотеки Matplotlib.

Оборудование и материалы

Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel Pentium с тактовой частотой не менее 800 МГц, оперативная память – не менее 64 Мбайт, свободное дисковое пространство – не менее 250 Мбайт, устройство для чтения компакт-дисков, монитор типа Super VGA (число цветов – 256) с диагональю не менее 15". Программное обеспечение – операционная система WINDOWS 2000 / XP / Vista / 7 и выше.

Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для пользователей персональных компьютеров: самостоятельно не производить ремонт персонального компьютера,

установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору); соблюдать правила техники безопасности при работе с электрооборудованием; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Защита работы

Перед выполнением лабораторной работы каждый студент получает индивидуальное задание. Защита лабораторной работы происходит только после его выполнения (индивидуального задания). При защите лабораторной работы студент отвечает на контрольные вопросы, приведенные в конце, и поясняет выполненное индивидуальное задание. Ход защиты лабораторной работы контролируется преподавателем.

Задание

1. Ознакомьте с теоретическим материалом.
2. Ответьте на контрольные вопросы, приведенные в конце работа.
3. Выполните индивидуальное задание, согласно вашему номеру варианта (Таблица 8.1).

Таблица 8.1 – Задания для самостоятельного выполнения:

№ варианта	Задачи для выполнения
1	9,5,6,11
2	6,8,2,11
3	3,5,7,11
4	4,1,6,11
5	7,4,2,11
6	5,6,8,11
7	1,8,6,11
8	5,7,9,11
9	4,6,5,11
10	3,2,10,11
11	2,4,9,11
12	6,8,2,11
13	2,1,5,11
14	1,8,6,11
15	5,3,7,11

Задача №1. Визуализируйте график функции $y=x^2$.

Задача №2. Визуализируйте график функции $y=\sqrt{x}$.

Задача №3. Визуализируйте график функции $y=\ln(x)$.

Задача №4. Визуализируйте график функции $y=\sin(x)$.

Задача №5. Визуализируйте график функции $y=\cos(x)$.

Задача №6. Визуализируйте график функции $y=\operatorname{tg}(x)$.

Задача №7. Визуализируйте график функции $y=\operatorname{ctg}(x)$.

Задача №8. С помощью блокнота создайте текстовый документ, в котором будет содержаться информация о среднесуточной температуре воздуха за 10 дней. Создайте программу, которая прочитает данные из этого файла и отобразит их на графике.

Задача №9. Визуализируйте популярность марок автомобилей на российском рынке с помощью круговой диаграммы.

Задача №10. Визуализируйте численность населения России за последние 10 лет с помощью столбчатой диаграммы.

Задача №11. Пользователь вводит с клавиатуры несколько чисел. Количество заранее неизвестно, но известно, что ввод должен закончиться, как только пользователь введет символ конца ввода. Заранее этот символ неизвестен, пользователь должен ввести его сам. Используя функцию ввода с клавиатуры, написанную при выполнении предыдущей лабораторной работы, визуализируйте числа, введенные пользователем с клавиатуры, и сохраните их в текстовый файл.

Контрольные вопросы:

1. Что такое словари?
2. Что такое кортежи?
3. В чем отличие словарей, списков и кортежей?
4. Что делает функция `split()`? Какие у нее аргументы? Приведите пример ее использования.
5. Что делает функция `strip()`? Какие у нее аргументы? Приведите пример ее использования.

6. В чем отличие функций `strip()`, `lstrip()` и `rstrip()`?
7. Что такое пакеты и модули в языке Python? Как их установить и использовать в своей программе?
8. Для чего нужна библиотека Matplotlib?
9. Расскажите об основных терминах и понятиях библиотеки Matplotlib.
10. Какие основные типы графиков и диаграмм можно построить с помощью библиотеки Matplotlib?
11. Какая функция используется для построения линейных графиков? Какие у нее есть параметры?
12. Какая функция используется для построения точечных графиков? Какие у нее есть параметры?
13. Какая функция используется для построения столбчатых диаграмм?
14. Какая функция используется для построения круговых диаграмм?
15. Как отобразить два и более графиков на одном листе?

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации самостоятельной работы студента

по дисциплине

«Практикум по применению программного обеспечения»

Направление подготовки 13.03.02 «Электроэнергетика и электротехника»
Направленность профиль "Цифровые технологии в электроэнергетика"

Ставрополь, 2026

СОДЕРЖАНИЕ

Введение.....	3
1 Общая характеристика самостоятельной работы студента при изучении дисциплины.....	4
2 План-график выполнения самостоятельной работы	6
3 Контрольные точки и виды отчетности по ним.....	7
4 Методические рекомендации по изучению теоретического материала.....	8
5 Методические указания по подготовке к лабораторным занятиям	15
Список рекомендуемой литературы.....	16

Введение

Самостоятельная работа является внеаудиторной и предназначена для самостоятельного изучения определенных тем дисциплины по материалам, рекомендованным преподавателем, а также в порядке подготовки к лабораторным занятиям.

Цель дисциплины «Практикум по применению программного обеспечения» – формирование согласно ООП ВО набора общепрофессиональных и профессиональных компетенций будущего бакалавра по направлению 13.03.02 «Электроэнергетика и электротехника».

Для достижения этой цели в процессе изучения дисциплины решаются следующие задачи: ознакомление с основными понятиями и терминами изучаемой дисциплины, которыми будущий специалист будет оперировать в своей практической деятельности; освоение базовых принципов работы с современными системами автоматизированного проектирования.

1 Общая характеристика самостоятельной работы студента при изучении дисциплины

Основная задача высшего образования заключается в формировании творческой личности специалиста, способного к саморазвитию, самообразованию, инновационной деятельности. Решение этой задачи вряд ли возможно только путем передачи знаний в готовом виде от преподавателя к студенту. Для решения этой задачи в учебные планы всех специальностей включена самостоятельная работа, составляющая 50% учебного времени студента.

В широком смысле под самостоятельной работой студентов следует понимать совокупность всей самостоятельной деятельности студентов, как в учебной аудитории, так и вне ее, в контакте с преподавателем и в его отсутствие.

В учебном процессе выделяют два вида самостоятельной работы: аудиторная; внеаудиторная.

Аудиторная самостоятельная работа по дисциплине выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию. Внеаудиторная самостоятельная работа выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Содержание внеаудиторной самостоятельной работы определяется в соответствии с рекомендуемыми видами заданий согласно примерной и рабочей программ учебной дисциплины. Видами заданий для внеаудиторной самостоятельной работы по дисциплине «Практикум по применению программного обеспечения» являются:

– для овладения знаниями: чтение текста (учебника, первоисточника, дополнительной литературы), составление плана текста, графическое изображение структуры текста, конспектирование текста, выписки из текста, работа со словарями и справочниками, ознакомление с нормативными документами, учебно-исследовательская работа, использование аудио- и видеозаписей, компьютерной техники и Интернета и др.;

– для закрепления и систематизации знаний: обработка текста, повторная работа над учебным материалом (учебника, первоисточника, дополнительной литературы, аудио и видеозаписей, составление плана, составление таблиц для систематизации учебного материала, ответ на контрольные вопросы, заполнение рабочей тетради, аналитическая обработка текста (аннотирование, рецензирование, реферирование, конспект-анализ и др);

– для закрепления навыков – подготовка к лабораторным работам, оформление отчетов по лабораторным работам, выполнение дополнительных заданий, выданных преподавателем.

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений студентов. Контроль результатов внеаудиторной самостоятельной работы студентов может осуществляться в пределах времени, отведенного на обязательные учебные занятия по дисциплине и внеаудиторную самостоятельную работу студентов по дисциплине, может проходить в письменной, устной или смешанной форме.

Высшей ступенью обучения является приобретение навыков самостоятельной творческой работы, а творческому применению знаний нужно учиться параллельно с их приобретением. Задачами самостоятельной работы студентов по дисциплине «Практикум по применению программного обеспечения» являются:

– формирования готовности студентов к поиску, обработке и применению информации для решения профессиональных задач;

– развитие познавательных способностей и активности студентов, творческой инициативы, самостоятельности, ответственности и организованности;

– формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;

– выработка навыков эффективной самостоятельной профессиональной деятельности.

2 План-график выполнения самостоятельной работы

Общий объём самостоятельной работы в рамках изучения дисциплины «Практикум по применению программного обеспечения» составляет 76 академических часа.

В структуре самостоятельной работы предусмотрено самостоятельное изучение теоретического материала, подготовка к лабораторным занятиям.

В рамках самостоятельной работы предусмотрена подготовка студента к проведению и защите лабораторных работ. В рамках изучения дисциплины студент должен выполнить следующие лабораторные работы:

- Единая система конструкторской документации. Общие требования к текстовым документам;
- Применение на практике возможностей современных текстовых редакторов для оформления текстовых документов, согласно ЕСКД
- Применение на практике возможностей текстовых редакторов для автоматизации оформления текстовых документов;
- Первичное знакомство с векторными редакторами
- Применение на практике возможностей векторных редакторов для формирования электрических схем;
- Первичное знакомство с языком программирования Python
- Python: списки, циклы, функции;
- Работа с файлами, визуализация данных с помощью языка Python.

3 Контрольные точки и виды отчетности по ним

Успеваемость студентов по дисциплине оценивается в ходе текущего и промежуточного контроля.

Текущий контроль осуществляется в течение семестра на лабораторных занятиях.

В начале семестра студенческая группа получает график контрольных мероприятий с указанием форм и сроков их проведения, с указанием баллов за каждое контрольное мероприятие.

Проверка подготовки к лабораторным занятиям проводится в форме защиты отчетов по данным виду занятия.

Защита отчета проходит в форме доклада студента по выполненной работе и ответов на вопросы преподавателя.

Максимальное количество баллов студент получает, если оформление отчета соответствует установленным требованиям, студент может правильно ответить на контрольные вопросы, а отчет полностью раскрывает суть работы. Основанием для снижения оценки являются:

- неполное выполнение задания по лабораторной работе;
- некорректность результатов и выводов;
- затруднение в ответах на контрольные вопросы по лабораторной работе;

4 Методические рекомендации по изучению теоретического материала

Большая часть самостоятельной работы студента состоит в изучении литературы. Одна из задач студента – научиться самостоятельно работать с книгой, а это требует определенных затрат энергии и времени. Самостоятельная работа с книгой может быть успешной, если текст не только прочитан, но и законспектирован. Существует несколько форм записей, но любая форма записи не даст нужного результата, если не будет пробуждать мысли того, кто ее ведет, если отсутствует активная работа ума и формирование своих выводов из прочитанного.

Выбор формы записи зависит от индивидуальных особенностей человека, его образованности и опыта. При этом не меньшую роль играет назначение записей, то есть то, какие задачи ставит перед собой человек (для самообразования, для выступления на семинаре, для использования в будущем).

Введение записей мобилизует наряду со зрительной памятью, также и моторную память. Кроме того, у человека, систематически ведущего записи изучаемой литературы, создается свой фонд материалов для быстрого повторения и мобилизации накопленных знаний.

Все записи должны быть убористыми и компактными. Интервалы между строками должны быть достаточными, чтобы вписывать дополнения. Рекомендуется вести записи ручкой, а карандашом или ручкой другого цвета пользоваться для отметок и выделений при последующей работе. Полезно также датировать записи.

Записи могут носить различный характер: план, выписки, тезисы, аннотирование, конспектирование.

1. План - наиболее краткая формой записи. Это перечень вопросов, рассматриваемых в книге или статье. План обычно раскрывает структуру произведения, логику автора, способствует лучшей ориентации в содержании.

Так, составленным планом можно воспользоваться, чтобы вспомнить прочитанное или быстро отыскать в книге нужное место. Представление об основных пунктах плана дает оглавление книги, поэтому во многих случаях наименования глав и разделов можно использовать в качестве пунктов.

Составление плана приучает логически мыслить, вырабатывать умение сжато и последовательно излагать суть вопроса в письменной и устной форме.

Существует два способа составления плана: работа над ним по ходу чтения и составление плана после ознакомления с произведением. При этом план получается более последовательным и стройным.

Трудность составления плана состоит в том, что надо выяснить для себя, прежде всего, построение изучаемого текста, ход мыслей автора и лишь затем изложить содержание работы кратко и ясно. При всей своей краткости план дает представление о содержании прочитанного. Следует учесть, что форма плана не исключает цитирования отдельных мест и обобщений. Различают простой и развернутый план. В отличие от простого плана развернутый план не только содержит перечисление вопросов, но и раскрывает основные идеи произведения, может включать выдержки из него, схемы, таблицы. Планом, особенно развернутым, необходимо пользоваться при написании выступления или статьи.

В целом развернутый план дает гораздо большее представление о произведении, его основных идеях, задачах, которые в нем решаются. Он может включать положения, замечания, собственные мысли студента.

Важно знать, что составление планов помогает вырабатывать способность к отвлеченному, абстрактному мышлению, но наибольшую пользу составление плана даст подготовленным лицам, которые бывают достаточно лишь взглянуть на перечень основных вопросов, чтобы воспроизвести содержание прочитанного.

2. Тезисы – более сложная и совершенная форма записи, чем составление плана.

Это сжатое изложение основных мыслей прочитанного произведения или подготовляемого выступления. Особенностью тезисов является их утвердительный характер.

В них сосредотачивается самое главное, только выводы и обобщения, в них меньше доказательств, иллюстрации и пояснений. Тезисы не должны повторять дословно текст, но в ряде мест могут быть близки к нему, воспроизводя некоторые характерные выражения автора, важные для

понимания хода его мыслей. Составление тезисов помогает глубже понять основные идеи произведения, выделить главное в нем; приучают сжато, точно и четко сформулировать свои мысли, повышает культуру речи и письма. При составлении тезисов учитывают следующее. Прежде всего, если произведение небольшое, необходимо внимательно изучать его в целом, если большое – изучать по главам и разделам. Затем, когда будут ясны основные идеи, кратко и последовательно излагать их в виде пунктов.

Различают простые и сложные, развернутые тезисы. Если записывают только утверждение чего – либо, такой тезис называют простым, а сложным тезисом будет выражение главной мысли, содержащее, кроме утверждения, еще и краткое ее доказательство.

Часто тезисы формулируются самим автором как выводы и обобщения в заключении книги или разделах книги. Нередко тезисы выделяются в тексте другим шрифтом.

Рекомендуется делать тезисные записи своими словами, причем можно записывать один абзац за другим, учитывая смысловую связь между ними. Но в большинстве случаев следует составлять сводный тезис, сложный по форме. При этом объединяется несколько утверждений, тесно связанных между собой.

Тезисы по содержанию очень близки к **конспекту**, но конспект носит более описательный характер, и его положения не столь категоричны, как в тезисах. Кроме того, конспект представляет собой более полную форму записи.

Следует отметить, что различие между формами записей условно, но в любой форме запись – важнейшая часть самостоятельной работы с книгой.

3. Выписки. Это записи текста из книги: теоретических положений, статистических данных, имеющих по мнению читателя важное значение.

Достоинство выписок состоит в точности воспроизведения текста книги, удобстве пользования записями при последующей работе, в накоплении обобщений и фактического материала. Выписки полезны для повторения, освежения в памяти прочитанного, для быстрой мобилизации своих знаний, когда необходимо в короткий срок вспомнить материал. Выписки выделяют из текста самое главное и тем самым помогают глубже понять его. Без них трудно

обойтись при подготовке доклада, реферата, выступления. Выписки следует рассматривать как составную часть тезисов и конспектов.

Выписывать текст можно и по ходу чтения и после его завершения. В последнем случае надо замечать места, которые потом будут выписаны. Необходимо каждую выписку снабжать ссылкой на источник с указанием соответствующей страницы. Это нужно, чтобы в последствии можно было быстро найти в книге соответствующее место. Целесообразно выписывать из текста только такие места, в которых содержится самое главное, суть вопроса. Выписки должны быть ориентированы на изучение произведения в целом, а не отдельных мест, поскольку положения, вырванные из общего контекста, понимаются нередко совсем не так, как этого хотел автор. Иначе говоря, отдельно взятые, лишенные пояснений выдержки могут быть не поняты или поняты неправильно.

Выписки бывают дословные (цитаты) и «свободные», когда мысли автора излагаются своими словами. Следует учесть, что большие отрывки, которые трудно цитировать, целесообразнее в краткой форме переложить своими словами, но «яркие» и важные места лучше выписывать дословно. Каждую цитату следует заключать в кавычки. Если ее берут из середины предложения, то после вводных кавычек ставят три точки. Ставят их и в конце цитаты, если из предложения опущены последние слова.

Следует знать, что какого-либо единого метода выписок, годного для всех случаев, не существует, поскольку у каждого человека свои особенности мышления и восприятия, свой подход к теме. Все это влияет на содержание и характер выписок.

Выписки рекомендуется хранить в картотеке, конвертах или папках, на которых следует обозначить общую тему.

4. Аннотация – еще одна форма записи, являющаяся кратким обобщением содержания книги. Ею удобно пользоваться, если имеется намерение вернуться к изучаемому произведению. Аннотация может быть необходима и для того, чтобы не забыть о нем.

Для составления аннотации надо сначала полностью прочитать и глубоко продумать произведение. При всей своей краткости аннотация может содержать отдельные фрагменты авторского текста, а не только оценку книги или статьи.

5. Резюме очень близко к аннотации. Это запись, являющаяся краткой оценкой прочитанного материала. Различие между ними состоит в том, что аннотация сжато характеризует произведение в целом, а резюме концентрирует внимание на его выводах, главных итогах.

6. Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspicere», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – неперемutable правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответствии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становится малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и

доказательств, вплоть до фактического материала. Иначе говоря, конспект – это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

5 Методические указания по подготовке к лабораторным занятиям

В рамках самостоятельной работы предусмотрена подготовка студента к проведению и защите лабораторных работ. Лабораторные занятия позволяют интегрировать теоретические знания и формировать практические умения и навыки студентов в процессе учебной деятельности.

Цели лабораторных занятий по дисциплине «Практикум по применению программного обеспечения»:

- 1) закрепление теоретического материала путем систематического контроля за самостоятельной работой студентов;
- 2) формирование умений использования теоретических знаний в процессе выполнения лабораторных работ;
- 3) развитие аналитического мышления путем обобщения результатов лабораторных работ;
- 4) формирование навыков оформления результатов лабораторных работ в виде таблиц, графиков, выводов.

На лабораторных занятиях осуществляются следующие формы работ со студентами: индивидуальная (оценка знаний, выполненных тестовых заданий, проверка рабочих тетрадей); групповая (выполнение заданий малыми группами по 2-4 человека); фронтальная (подведение итогов выполнения лабораторных работ, подведение итогов выполнения теста).

Список рекомендуемой литературы

Основная литература:

1. Северенс, Ч. Введение в программирование на Python / Ч. Северенс. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с. : схем., ил. - <http://biblioclub.ru/>, экземпляров неограничено

2. Балджы, А. С. Математика на Python : учебно-методическое пособие / А.С. Балджы, М.Б. Хрипунова, И.А. Александрова ; Финансовый университет при Правительстве Российской Федерации, 1, Элементы линейной алгебры и аналитической геометрии. - Москва : Прометей, 2018. - 76 с. : табл. - <http://biblioclub.ru/>. - Библиогр. в кн. - ISBN 978-5-907003-86-6, экземпляров неограничено.

3. Борзунов, С. В. Алгебра и геометрия с примерами на Python Электронный ресурс / Борзунов С. В., Кургалин С. Д. : учебное пособие для вузов. - Санкт-Петербург : Лань, 2020. - 444 с. - ISBN 978-5-8114-5489-1, экземпляров неограничено

Дополнительная литература:

1. Персова, М. Г. Современные компьютерные технологии Электронный ресурс : Конспект лекций / М. Г. Персова, Ю. Г. Соловейчик, П. А. Домников. - Новосибирск : Новосибирский государственный технический университет, 2014. - 80 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-7782-2427-8

2. Современные компьютерные технологии : учебное пособие / Р.Г. Хисматов, Р.Г. Сафин, Д.В. Тунцев, Н.Ф. Тимербаев ; Министерство образования и науки России ; Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Казанский национальный исследовательский технологический университет». - Казань : Издательство КНИТУ, 2014. - 83 с. : схем. - <http://biblioclub.ru/>. - Библиогр. в кн. - ISBN 978-5-7882-1559-4

Методические материалы:

1. Методические указания к выполнению практических работ по дисциплине «Практикум по применению программного обеспечения» для студентов направления подготовки 13.03.02 «Электроэнергетика и электротехника», СКФУ, 2025 / сост. Зеленский Е.Г.

2. Методические указания по организации самостоятельной работы студента по дисциплине «Практикум по применению программного обеспечения» для студентов направления подготовки 13.03.02 «Электроэнергетика и электротехника», СКФУ, 2025 / сост. Зеленский Е.Г.

Электронные ресурсы:

1. <http://catalog.ncfu.ru> – электронные каталоги Ассоциации электронных библиотек учебных заведений и организаций СКФО.

2. <http://elibrary.ru> – Научная электронная библиотека eLIBRARY.RU

3. <https://biblioclub.ru/> ЭБС «Университетская библиотека онлайн»

4. www.iprbookshop.ru – Электронно-библиотечная система IPRbooks