МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ

по дисциплине

## «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ПРОГРАММИРОВАНИЕ»

Направление подготовки 09.03.02 Информационные системы и технологии
Направленность (профиль) Прикладное программирование и интернет-технологии

Ставрополь
2026

# СОДЕРЖАНИЕ

**Практическая работа №1. Инфраструктура разработки: Git, GitHub и Markdown**

**Практическая работа №2. Статический веб: Основы HTML5**

**Практическая работа №3. Статический веб: Основы CSS3**

**Практическая работа №4. Документация как код: Система верстки Typst**

**Практическая работа №5. Основы Python: Типы данных и ввод-вывод**

**Практическая работа №6. Управляющие конструкции в Python**

**Практическая работа №7. Коллекции и структуры данных**

**Практическая работа №8. Функциональное программирование и модули**

**Практическая работа №9. Работа с файлами и автоматизация**

**Практическая работа №10. Введение в C++ и настройка окружения**

**Практическая работа №11. Управляющие конструкции и логика**

**Практическая работа №12. Функции и модульное программирование**

**Практическая работа №13. Массивы и работа с памятью**

**Практическая работа №14. Структуры данных и основы ООП**

**Практическая работа №15. Наследование и полиморфизм**

**Практическая работа №16. Шаблоны и стандартная библиотека контейнеров (STL)**

**Практическая работа №17. Потоки данных и обработка исключений**

# ПРАКТИЧЕСКАЯ РАБОТА №1. ИНФРАСТРУКТУРА РАЗРАБОТКИ: GIT, GITHUB И MARKDOWN

## Цели и задачи

Цель практической работы: Освоение базовых инструментов контроля версий, настройка профессионального рабочего окружения в VS Code и изучение стандартов текстовой разметки для документации.

Основные задачи:

– Установить Git и настроить глобальные параметры пользователя: user.name и user.email;

– Авторизоваться в GitHub через VS Code и создать удаленный репозиторий для курса;

– Изучить синтаксис разметки Markdown: заголовки, списки, ссылки, блоки кода и таблицы.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для

практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork1.md

***Порядок выполнения работы:***

1. Работа выполняется локально в редакторе VS Code с использованием встроенного терминала (PowerShell или Bash).

2. Для синхронизации кода используется Git.

3. Проект должен быть опубликован в публичном репозитории на GitHub.

4. Изменения фиксируются атомарными коммитами с описанием на английском языке.

5. Оформление текстовых файлов выполняется строго по спецификации CommonMark (Markdown).

### Задание 1. Настройка окружения и инициализация.

*Подготовка локальной папки проекта и установление связи с облачным хранилищем.*

– Создать рабочую директорию и инициализировать в ней Git-репозиторий командой git init.

– Связать локальный репозиторий с удаленным (git remote add origin).

– Создать файл .gitignore и добавить в него исключения для служебных папок .vscode и временных файлов.

**Задание 2. Документирование проекта (README).**

*Создание 'лица' репозитория с использованием всех основных возможностей markdown*.

– Создать файл README.md с заголовком первого уровня (название дисциплины).

– Добавить раздел 'О себе' со списком навыков (маркированный список) и ссылкой на личный профиль.

– Вставить блок кода с примером любой команды терминала и оформить таблицу с планом обучения на семестр.

**Задание 3. Рабочий цикл Git (Workflow).**

*Отработка фиксации изменений и отправки их на сервер*.

– Добавить созданные файлы в индекс (git add) и сделать первый коммит (git commit).

– Внести изменение в README.md (например, добавить цитату), зафиксировать его вторым коммитом.

– Выполнить команду git push и убедиться, что все файлы и история коммитов отображаются на странице GitHub.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между локальным и удаленным репозиторием?

2. Зачем нужно выполнять команду git add перед коммитом?

3. Какую информацию принято указывать в сообщении к коммиту (commit message)?

4. Как в Markdown сделать текст полужирным или курсивом?

5. Что такое 'stage' (индекс) в терминах Git?

6. Как проверить текущий статус файлов в репозитории через терминал?

7. Для чего нужен файл .gitignore и можно ли его изменять в процессе работы?

8. Как вставить изображение в документ Markdown, чтобы оно отображалось на GitHub?

9. Каким образом можно просмотреть историю всех совершенных изменений в проекте?

# ПРАКТИЧЕСКАЯ РАБОТА №2. СТАТИЧЕСКИЙ ВЕБ: ОСНОВЫ HTML5

## Цели и задачи

Цель практической работы: Изучение принципов построения гипертекстовых документов, освоение семантической разметки и создание каркаса персонального веб-сайта.

Основные задачи:

– Изучить базовую структуру HTML-документа: doctype, html, head, body;

– Освоить работу с текстовыми тегами, списками, таблицами и формами;

– Научиться подключать внешние ресурсы (изображения, иконки) и создавать гиперссылки.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork2.md

***Порядок выполнения работы:***

1. Разработка ведется в VS Code.
2. Для предварительного просмотра используется расширение Live Server.
3. Структура проекта должна содержать файл index.html и папку assets для медиафайлов.
4. Код должен соответствовать стандартам W3C.
5. Особое внимание уделяется использованию семантических тегов вместо универсальных контейнеров div.

**Задание 1. Семантический каркас страницы.**

*Создание логической структуры сайта-портфолио с использованием современных стандартов разметки*.

– Реализовать заголовочную часть сайта (header) с логотипом и навигационным меню (nav).

– Разделить основное содержимое (main) на тематические секции (section) и статьи (article).

– Оформить подвал сайта (footer) с контактной информацией и авторскими правами.

**Задание 2. Контент и работа с данными.**

*Наполнение страницы текстовой и графической информацией,*

*оформление структурированных списков.*

– Создать блок 'Мои проекты', используя маркированные и нумерованные списки.

– Разместить таблицу 'Учебный план', содержащую столбцы с названием дисциплины, часами и формой контроля.

– Вставить портретное изображение, настроив атрибуты alt и title для доступности.

**Задание 3. Интерактивные элементы и формы.**

*Реализация механизмов взаимодействия с пользователем и внешних переходов.*

– Создать форму обратной связи с полями для ввода имени (text), почты (email) и многострочного сообщения (textarea).

– Добавить кнопку отправки формы (submit) и чекбокс для согласия на обработку данных.

– Реализовать систему якорей для быстрой навигации между секциями одной страницы.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем заключается разница между парными и одиночными тегами?

2. Какую роль играет мета-тег charset='UTF-8' в секции head?

3. Почему использование семантических тегов (main, section) важно для SEO и экранных дикторов?

4. В чем отличие абсолютной ссылки от относительной?

5. Какой атрибут позволяет открыть ссылку в новой вкладке браузера?

6. Для чего нужен атрибут alt у тега img?

7. Как создать нумерованный список, начинающийся не с единицы?

8. Какую функцию выполняет тег label в формах и как он связывается с полем ввода?

9. Что такое валидация HTML-кода и как её проверить?

# ПРАКТИЧЕСКАЯ РАБОТА №3. СТАТИЧЕСКИЙ ВЕБ: ОСНОВЫ CSS3

## Цели и задачи

Цель практической работы: Освоение каскадных таблиц стилей, изучение блочной модели (Box Model) и методов позиционирования элементов для создания визуально привлекательного интерфейса.

Основные задачи:

– Изучить способы подключения стилей и приоритеты селекторов (тег, класс, id);

– Освоить управление типографикой: шрифты, размеры, межстрочные интервалы;

– Изучить параметры блочной модели: padding, border, margin и свойство box-sizing.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork3.md

***Порядок выполнения работы:***

1. Стили описываются в отдельном файле style.css и подключаются через тег <link> в секции head.

2. Для подбора цветов рекомендуется использовать Adobe Color или аналоги.

3. Работа с макетом строится на принципах Flexbox.

4. Финальный результат должен корректно отображаться в современных браузерах и быть опубликован через GitHub Pages.

**Задание 1. Типографика и цветовая схема.**

*Настройка шрифтового оформления и базовой палитры сайта*.

– Подключить нестандартный шрифт через сервис Google Fonts и применить его к основному тексту.

– Определить цветовую переменную (CSS variables) для основного акцентного цвета и применить её к заголовкам.

– Настроить стилевое оформление ссылок, убрав стандартное подчеркивание и добавив изменение цвета при наведении (:hover).

**Задание 2. Блочная модель и декоративные элементы.**

*Работа с пространством вокруг контента и визуальными эффектами границ.*

– Добавить внутренние и внешние отступы для всех секций сайта, чтобы контент не прижимался к краям экрана.

– Реализовать скругление углов (border-radius) и легкую тень (box-shadow) для карточек проектов.

– Настроить фоновое изображение или градиент для верхней части страницы (header).

**Задание 3. Компоновка элементов с Flexbox.**

*Создание гибкой структуры навигации и сетки контента*.

– Превратить список навигации в горизонтальное меню, используя display: flex и justify-content.

– Реализовать двухколоночный макет для секции 'О себе' (текст слева, фото справа) через Flexbox.

– Выровнять элементы формы обратной связи по вертикальной оси, обеспечив одинаковые отступы между полями.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. Что означает понятие 'каскадность' в CSS?

2. В чем разница между селекторами .class и #id с точки зрения приоритета?

3. Из каких компонентов состоит блочная модель (Box Model)?

4. Как свойство box-sizing: border-box влияет на расчет ширины элемента?

5. В каких единицах измерения лучше задавать размер шрифта (px, em, rem) и почему?

6. Какую задачу решает свойство flex-direction?

7. Как центрировать блок по горизонтали и вертикали внутри flex-контейнера?

8. Для чего нужны псевдоклассы, такие как :hover или :nth-child()?

9. Как проверить CSS-свойства элемента в инструментах разработчика браузера (F12)?

# ПРАКТИЧЕСКАЯ РАБОТА №4. ДОКУМЕНТАЦИЯ КАК КОД: СИСТЕМА ВЕРСТКИ TYPST

## Цели и задачи

Цель практической работы: Изучение альтернативных методов подготовки технических документов, освоение синтаксиса Typst и автоматизация генерации PDF-отчетов.

Основные задачи:

– Установить расширение Typst в VS Code и создать базовый файл проекта;

– Изучить правила разметки текста, вставки математических формул и таблиц;

– Освоить механизмы управления стилями документа через функции set и show.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork4.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code с установленным расширением Typst LSP.

2. Компиляция документа происходит в фоновом режиме (watch mode).

3. Структура документа должна включать заголовочную часть, оглавление и основной контент.

4. Итоговый PDF-файл и исходный код (.typ) сохраняются в репозиторий проекта.

**Задание 1. Структура технического отчета.**

*Создание каркаса документа с автоматической нумерацией и оглавлением.*

– Сформировать титульную страницу с указанием дисциплины, темы работы и автора.

– Добавить автоматическое оглавление с помощью команды #outline().

– Разбить документ на разделы (Heading), используя символ '=' и настроить сквозную нумерацию страниц.

**Задание 2. Математика и визуализация данных.**

*Оформление сложного научного контента и графических материалов.*

– Написать блок математических формул (включая дроби, греческие

символы и индексы), используя синтаксис '$ ... $'.

– Вставить изображение (схему из предыдущих работ по вебу) с подписью и настроить его обтекание текстом.

– Реализовать таблицу сравнения HTML и Typst, используя функцию #table() с настройкой границ и заливки шапки.

**Задание 3. Стилизация и автоматизация.**

*Настройка глобального внешнего вида документа через функции управления.*

– Применить функцию #set text() для установки шрифта (например, Linux Libertine) и размера кегля.

– Использовать функцию #show для изменения стиля всех заголовков первого уровня (например, сделать их синими).

– Оформить листинг кода из работы по HTML с подсветкой синтаксиса, используя блок '```html ... ```'.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем основные преимущества Typst перед классическим Microsoft Word и LaTeX?

2. Каким образом в Typst осуществляется компиляция исходного кода в PDF?

3. Как вставить в текст гиперссылку или перекрестную ссылку на другой раздел документа?

4. В чем разница между инлайновым ($x+y$) и блочным ($$x+y$$) написанием формул?

5. Какую роль играют функции set и show в архитектуре Typst-документа?

6. Как добавить нумерацию строк в блоках с программным кодом?

7. Как вставить специальный символ (например, эмодзи или символ бесконечности)?

8. Можно ли использовать переменные в Typst для хранения повторяющихся данных?

9. Как настроить поля (margins) страницы документа?

# ПРАКТИЧЕСКАЯ РАБОТА №5. ОСНОВЫ PYTHON: ТИПЫ ДАННЫХ И ВВОД-ВЫВОД

## Цели и задачи

Цель практической работы: Настройка среды разработки для языка Python, изучение базового синтаксиса и освоение принципов взаимодействия программы с пользователем через консоль.

Основные задачи:

– Проверить корректность установки интерпретатора Python и настроить выбор версии в VS Code;

– Изучить основные типы данных: целые числа (int), числа с плавающей точкой (float) и строки (str);

– Освоить функции преобразования типов данных и форматированный вывод.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork5.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code с установленным расширением Python.
2. Код пишется в файлах с расширением .py.
3. Запуск скриптов осуществляется через терминал командой python filename.py.
4. Для вычислений используются стандартные арифметические операторы.
5. Все переменные должны иметь осмысленные названия на английском языке (snake_case).

**Задание 1. Интерактивный калькулятор.**

*Создание программы для выполнения базовых арифметических операций с данными, вводимыми пользователем*.

– Реализовать ввод двух чисел с помощью функции input() и их преобразование в тип float.

– Вычислить сумму, разность, произведение и результат возведения первого числа в степень второго.

– Вывести результаты в консоль, используя f-строки для наглядного оформления ответа.

**Задание 2. Работа со строковыми данными.**

*Освоение методов манипуляции текстом и базового форматирования*.

– Написать скрипт, который запрашивает имя, фамилию и возраст пользователя.

– Реализовать вывод приветствия, где имя будет приведено к верхнему регистру (UPPERCASE), а фамилия — к формату заголовка (Title Case).

– Рассчитать год рождения пользователя на основе текущей даты и вывести итоговую карточку профиля в рамке из символов '*'.

**Задание 3. Математическое моделирование.**

*Применение встроенных функций для решения простых геометрических задач.*

– Написать программу для расчета гипотенузы прямоугольного треугольника по двум катетам (теорема Пифагора).

– Использовать функцию abs() для нахождения расстояния между двумя точками на числовой оси.

– Реализовать округление итогового результата до двух знаков после запятой с помощью функции round().

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем заключается особенность интерпретируемых языков программирования?

2. Какую роль играет виртуальное окружение (venv) в проектах на Python?

3. Почему функция input() всегда возвращает данные строкового типа (str)?

4. В чем разница между операторами деления '/' и '//'?

5. Как работает оператор деления по модулю '%' и где он может быть полезен?

6. Что такое динамическая типизация и как она проявляется в Python?

7. Каким образом можно вывести текст, содержащий как кавычки, так и апострофы?

8. Для чего нужны f-строки и чем они удобнее конкатенации строк через '+'?

9. Как в VS Code запустить скрипт в режиме отладки (Debug)?

# ПРАКТИЧЕСКАЯ РАБОТА №6. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ В PYTHON

## Цели и задачи

Цель практической работы: Освоение алгоритмической логики через использование условных операторов и реализацию повторяющихся вычислений с помощью циклов.

Основные задачи:

– Изучить синтаксис условного оператора if-elif-else;

– Освоить итерацию по диапазонам чисел с использованием функции range();

– Научиться прерывать (break) и продолжать (continue) выполнение циклов.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork6.md

***Порядок выполнения работы:***

1. Код пишется в VS Code.

2. Для проверки условий используются логические операторы (and, or, not).

3. Циклы реализуются через конструкции for и while.

4. Особое внимание уделяется соблюдению синтаксических отступов (4 пробела), которые определяют вложенность блоков кода в Python.

**Задание 1. Логика принятия решений.**

*Создание скриптов с разветвленной структурой в зависимости от входных данных.*

– Написать программу для классификации введенного числа (положительное, отрицательное или ноль).

– Реализовать проверку пароля: если введенная строка совпадает с эталоном — доступ разрешен, иначе — ошибка.

– Создать калькулятор индекса массы тела (ИМТ) с выводом текстовой интерпретации результата (недостаток, норма, избыток).

**Задание 2. Циклические вычисления.**

*Автоматизация повторяющихся действий и работа с числовыми последовательностями.*

– С помощью цикла for вывести таблицу умножения для заданного пользователем числа.

– Реализовать вычисление факториала числа n, введенного с клавиатуры.

– Написать цикл, который суммирует все четные числа в диапазоне от 1 до 100.

**Задание 3. Игровые алгоритмы и обработка исключений.**

*Создание интерактивного взаимодействия с неопределенным количеством итераций.*

– Реализовать игру 'Угадай число': программа загадывает число, а пользователь пытается его отгадать, получая подсказки 'больше' или 'меньше'.

– Использовать цикл while для создания меню программы, которое закрывается только при вводе команды 'exit'.

– Добавить проверку корректности ввода (try-except), чтобы программа не завершалась с ошибкой при вводе текста вместо числа.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. Какую роль в Python играют отступы и что произойдет при их нарушении?

2. В чем разница между операторами сравнения '==' и 'is'?

3. Как работает оператор elif и в каких случаях он предпочтительнее вложенных if?

4. В чем ключевое отличие цикла for от цикла while?

5. Зачем нужна функция range(start, stop, step) и какие параметры в ней обязательны?

6. Как создать бесконечный цикл и как его принудительно остановить?

7. Каким образом логические операторы and и or влияют на результат условия?

8. Для чего используются ключевые слова break и continue внутри цикла?

9. Как проверить, входит ли символ или подстрока в состав строки?

# ПРАКТИЧЕСКАЯ РАБОТА №7. КОЛЛЕКЦИИ И СТРУКТУРЫ ДАННЫХ

## Цели и задачи

Цель практической работы: Изучение способов организации, хранения и манипулирования наборами данных с использованием списков, кортежей и словарей.

Основные задачи:

– Изучить синтаксис создания списков (list), кортежей (tuple) и словарей (dict);

– Освоить методы добавления, удаления и поиска элементов в коллекциях;

– Научиться использовать срезы для извлечения подмножеств данных из последовательностей.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork7.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.

2. Акцент делается на выборе подходящей структуры данных для конкретной задачи.

3. Для обхода коллекций используются циклы for.

4. При модификации списков применяются встроенные методы (append, pop, sort).

5. Итоговый код должен демонстрировать навыки индексации и срезов (slices).

**Задание 1. Динамические списки и агрегация.**

*Создание и обработка упорядоченных наборов числовых и строковых данных.*

– Создать пустой список покупок и реализовать добавление в него 5 элементов через input() в цикле.

– Найти максимальное, минимальное значения и среднее арифметическое в списке случайных чисел.

– Отсортировать список строк по алфавиту и вывести элементы с 2-го по 4-й, используя срезы [1:4].

**Задание 2. Словари и ассоциативные массивы.**

*Организация данных в формате 'ключ-значение' для хранения*

*структурированной информации.*

– Создать словарь 'Студент', содержащий поля: имя, фамилия, возраст и список изученных технологий.

– Реализовать поиск значения по ключу, введенному пользователем, с обработкой ситуации отсутствия ключа (метод .get()).

– Добавить в словарь новую пару 'город проживания' и обновить возраст на единицу.

**Задание 3. Комплексные структуры данных.**

*Работа со списками словарей и вложенными коллекциями.*

– Создать список из трех словарей, каждый из которых описывает отдельный ИТ-проект (название, язык, статус).

– С помощью цикла вывести названия всех проектов, написанных на языке 'Python'.

– Преобразовать список названий проектов в неизменяемый кортеж и вывести его длину.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем главное отличие списка (list) от кортежа (tuple)?

2. Что такое 'изменяемый' (mutable) и 'неизменяемый' (immutable) типы данных?

3. Как проверить наличие ключа в словаре, не вызывая ошибку KeyError?

4. Каким образом работает отрицательная индексация (например, list[-1])?

5. Что возвращает срез списка [::-1]?

6. Можно ли использовать список в качестве ключа словаря? Почему?

7. В чем разница между методами .append() и .extend() у списков?

8. Как удалить элемент из словаря по его ключу?

9. Для чего нужна функция zip() при работе с двумя списками?

# ПРАКТИЧЕСКАЯ РАБОТА №8. ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И МОДУЛИ

## Цели и задачи

Цель практической работы: Освоение принципов декомпозиции кода, повторного использования алгоритмов через функции и изучение механизмов импорта стандартных библиотек.

Основные задачи:

– Изучить синтаксис объявления функций и передачи позиционных и именованных аргументов;

– Освоить области видимости переменных (локальные и глобальные);

– Научиться подключать и использовать стандартные модули Python (math, random, datetime).

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork8.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Функции описываются с использованием ключевого слова def.
3. Аргументы передаются по значению.
4. Возврат результата осуществляется через return.
5. Для каждого логически завершенного действия создается отдельная функция.
6. Импорт внешних модулей производится в начале файла.

**Задание 1. Создание пользовательских функций.**

*Разработка библиотеки функций для выполнения типовых математических и текстовых операций.*

– Написать функцию, которая принимает список чисел и возвращает только четные из них.

– Реализовать функцию проверки палиндрома (слово, которое читается одинаково в обоих направлениях).

– Создать функцию для расчета стоимости заказа с учетом налога и скидки, задаваемых в качестве аргументов по умолчанию.

**Задание 2. Модульность и стандартная библиотека.**

*Использование готовых решений из экосистемы python для решения прикладных задач.*

– Использовать модуль math для вычисления площади круга и гипотенузы треугольника по заданным параметрам.

– С помощью модуля random реализовать генератор случайных паролей заданной длины, состоящих из букв и цифр.

– Использовать модуль datetime для вычисления количества дней, прошедших с начала текущего года до сегодняшнего числа.

**Задание 3. Декомпозиция и рефакторинг кода.**

*Разделение сложной программы на независимые функциональные блоки*.

– Переписать калькулятор из предыдущих работ, выделив каждую арифметическую операцию в отдельную функцию.

– Реализовать 'главную функцию' main(), которая управляет логикой программы и вызывает другие функции.

– Организовать передачу результата выполнения одной функции (например, расчета суммы) в качестве аргумента для другой (например, форматированного вывода).

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем преимущество использования функций перед написанием кода «в поток»?

2. Что такое 'docstring' и как правильно документировать функции в Python?

3. Какую роль играет оператор return и что произойдет, если его не указать?

4. Чем локальная переменная внутри функции отличается от глобальной переменной?

5. В чем разница между параметрами функции и её аргументами?

6. Как импортировать только конкретную функцию из модуля (например, sqrt из math)?

7. Для чего используется конструкция if __name__ == '__main__':?

8. Можно ли передать функцию в качестве аргумента другой функции?

9. Что такое анонимные функции (lambda) и в каких случаях их удобно использовать?

# ПРАКТИЧЕСКАЯ РАБОТА №9. РАБОТА С ФАЙЛАМИ И АВТОМАТИЗАЦИЯ

## Цели и задачи

Цель практической работы: Освоение методов чтения и записи данных в файлы, изучение механизмов обработки исключений и выполнение итогового мини-проекта с последующей синхронизацией на GitHub.

Основные задачи:

– Изучить режимы открытия файлов: чтение ('r'), запись ('w') и добавление ('a');

– Освоить конструкцию try-except для обработки ошибок отсутствия файла или некорректных данных;

– Научиться использовать стандартные методы строк (split, strip) для парсинга содержимого текстовых файлов.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork9.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Для открытия файлов используется менеджер контекста with.
3. Данные считываются построчно или целиком.
4. Для предотвращения аварийного завершения программы применяются блоки try-except.
5. Финальный скрипт и обработанные файлы фиксируются в репозитории с итоговым тегом или описанием завершения курса.

**Задание 1. Чтение и анализ текстовых данных.**

*Создание программы для извлечения полезной информации из внешнего источника.*

– Подготовить текстовый файл input.txt с произвольным набором чисел и строк.

– Написать функцию, которая считывает файл и подсчитывает общее количество слов в нем.

– Реализовать вычисление суммы всех числовых значений, найденных в файле, пропуская текстовые фрагменты.

**Задание 2. Генерация отчетов и запись в файл.**

*Автоматическое создание структурированных документов на основе вычислений скрипта.*

– Реализовать скрипт, запрашивающий у пользователя данные о его выполненных работах за семестр.

– Сформировать итоговую строку отчета и записать её в файл report.txt, не затирая предыдущие записи.

– Добавить в начало файла метку времени записи, используя модуль datetime.

**Задание 3. Итоговый мини-проект: Парсер и логгер.**

*Комплексная задача на обработку ошибок и синхронизацию всего рабочего процесса.*

– Написать программу, которая запрашивает имя файла у пользователя и выводит его содержимое.

– Оформить обработку ошибки FileNotFoundError, чтобы программа вежливо сообщала об отсутствии файла вместо вылета.

– Зафиксировать финальную версию кода коммитом 'Final project: File processing complete' и выполнить push в GitHub-репозиторий.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. Почему использование оператора with предпочтительнее ручного вызова метода .close()?

2. В чем разница между методами .read(), .readline() и .readlines()?

3. Что произойдет с существующим файлом, если открыть его в режиме 'w'?

4. Какую роль играет блок except и можно ли использовать несколько таких блоков?

5. Как проверить существование файла на диске перед его открытием, не используя try-except?

6. Для чего нужен блок finally в конструкции обработки исключений?

7. Каким образом можно записать список строк в файл за одну операцию?

8. Как кодировка файла (например, utf-8) влияет на корректность чтения данных в Python?

9. Как объединить результаты всех 9 лабораторных работ в одном GitHub-репозитории?

# ПРАКТИЧЕСКАЯ РАБОТА №10. ВВЕДЕНИЕ В C++ И НАСТРОЙКА ОКРУЖЕНИЯ

## Цели и задачи

Цель практической работы: Освоение инструментов компиляции, настройка среды разработки VS Code для работы с языком C++ и изучение базовой структуры программы.

Основные задачи:

– Установить компилятор (MinGW-w64 для Windows или встроенный Clang/GCC для macOS/Linux);

– Установить расширение 'C/C++' от Microsoft в VS Code и настроить конфигурацию сборщика;

– Изучить базовую структуру файла .cpp: директивы препроцессора, функцию main и пространство имен std.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork10.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Для компиляции используется набор инструментов GCC (g++) или Clang.
3. Отладка производится с помощью встроенного дебаггера GDB/LLDB.
4. Весь код должен соответствовать стандарту C++17 или выше.
5. Каждый этап работы фиксируется коммитом в Git.
6. Отчет оформляется в формате Markdown.

**Задание 1. Настройка компилятора и проверка версии.**

*Подготовка терминала и проверка работоспособности инструментария сборки*.

– Выполнить команду g++ --version в терминале и убедиться в наличии компилятора в системном пути.

– Создать текстовый файл hello.cpp и написать в нем минимальный код вывода 'Hello, World!'.

– Скомпилировать файл вручную через терминал командой g++ hello.cpp -o hello и запустить полученный исполняемый файл.

**Задание 2. Автоматизация сборки в VS Code.**

*Настройка конфигурационных файлов редактора для быстрой компиляции по горячим клавишам*.

– Настроить файл tasks.json для автоматического вызова компилятора с флагами предупреждений (-Wall).

– Создать файл launch.json для возможности пошаговой отладки программы.

– Убедиться, что при нажатии F5 программа собирается и запускается во встроенном терминале.

**Задание 3. Типы данных и стандартный ввод.**

*Разработка программы для взаимодействия с пользователем и выполнения вычислений.*

– Объявить переменные различных типов: int, double, char, bool и std::string.

– Реализовать ввод данных с клавиатуры через поток std::cin.

– Написать программу, которая запрашивает у пользователя радиус круга и выводит его площадь, используя константу M_PI.

**Задание 4. Арифметические операции и форматирование.**

*Отработка навыков работы с операторами и манипуляторами вывода*.

– Реализовать вычисление среднего арифметического трех целых чисел с приведением результата к типу double.

– Использовать библиотеку iomanip для вывода чисел с фиксированной точностью (2 знака после запятой).

– Продемонстрировать работу оператора остатка от деления (%) на примере проверки числа на четность.

**Задание 5. Работа со строками std::string.**

*Изучение базовых методов работы с текстовыми объектами в*

*современном c++*.

– Реализовать ввод полного имени пользователя, состоящего из нескольких слов, через функцию std::getline().

– Вывести длину введенной строки, используя метод .length().

– Объединить (конкатенировать) несколько строк в одно приветственное сообщение.

**Задание 6. Фиксация работы в Git.**

*Синхронизация исходного кода и настроек среды с удаленным репозиторием*.

– Добавить в .gitignore исполняемые файлы (.exe) и папку .vscode, если она содержит локальные пути.

– Создать коммит с исходным кодом всех упражнений и кратким описанием проделанной работы.

– Выполнить push в ветку main на GitHub и проверить корректность отображения кода в облаке.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между компилятором, линковщиком и интерпретатором?

2. Зачем нужна директива #include <iostream> в начале программы?

3. Какую роль выполняет функция main() и что означает возвращаемое значение return 0?

4. В чем преимущество использования std::string перед массивами символов char[]?

5. Зачем нужно пространство имен (namespace) и что делает строка using namespace std?

6. Как с помощью g++ скомпилировать программу под конкретный стандарт языка (например, C++17)?

7. Что такое 'флаги компилятора' и для чего нужен флаг -Wall?

8. В чем отличие операторов префиксного и постфиксного инкремента (++i vs i++)?

9. Как в C++ вывести символ переноса строки, кроме использования ' '?

# ПРАКТИЧЕСКАЯ РАБОТА №11. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ И ЛОГИКА

## Цели и задачи

Цель практической работы: Изучение синтаксиса условных операторов и циклов, освоение логических операций и реализация алгоритмов с ветвлением и повторением действий.

Основные задачи:

– Изучить синтаксис тернарного оператора как сокращенной формы if-else;

– Освоить работу с логическими операторами && (И), || (ИЛИ), ! (НЕ);

– Научиться использовать операторы break и continue для управления ходом цикла.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork11.md

***Порядок выполнения работы:***

1. Код пишется в среде VS Code.
2. Для проверки условий используются операторы if-else и switch-case.
3. Циклы реализуются через конструкции for, while и do-while.
4. Особое внимание уделяется типам данных в условиях (bool) и предотвращению бесконечных циклов.
5. Итоговый код фиксируется в Git.

**Задание 1. Условный оператор и тернарная форма.**

*Создание программ с логическим ветвлением на основе входных данных*.

– Написать программу, которая определяет, является ли введенный год високосным.

– Реализовать проверку попадания точки с координатами (x, y) в заданную область на плоскости.

– Использовать тернарный оператор для поиска максимального из двух введенных чисел.

**Задание 2. Множественный выбор через switch.**

*Обработка дискретных значений с помощью оператора выбора*.

– Создать консольное меню для выбора арифметической операции (+, -, *, /) над двумя числами.

– Реализовать вывод названия дня недели по его порядковому номеру (1-7).

– Добавить обработку некорректного ввода через секцию default.

**Задание 3. Циклы со счетчиком (for).**

*Автоматизация итерационных вычислений и работа с числовыми рядами.*

– Вывести все простые числа в диапазоне от 2 до N, введенного пользователем.

– Вычислить сумму квадратов всех чисел от 1 до 100.

– Реализовать вывод таблицы умножения в виде аккуратно выровненного блока.

**Задание 4. Циклы с условием (while / do-while).**

*Организация циклов с неопределенным заранее количеством итераций.*

– Написать программу 'Сумматор', которая складывает вводимые числа, пока пользователь не введет 0.

– Реализовать проверку ввода: запрашивать число, пока оно не окажется в диапазоне [1, 10], используя do-while.

– Создать алгоритм разворота целого числа (например, из 123 получить 321) через остаток от деления.

**Задание 5. Вложенные циклы и псевдографика.**

*Отработка навыков построения сложных итерационных структур.*

– Вывести в консоль прямоугольный треугольник из символов '*', катеты которого равны N.

– Реализовать отрисовку шахматной доски заданного размера из

символов '#' и пробелов.

– Найти все трехзначные числа Армстронга (сумма кубов цифр которых равна самому числу).

**Задание 6. Синхронизация и отчетность.**

*Документирование результатов работы и обновление удаленного репозитория.*

– Оформить в README.md краткое описание алгоритма одной из задач (например, поиска простых чисел).

– Сделать коммит с заголовком 'Lab 2: Control structures implemented' и отправить его на GitHub.

– Убедиться, что в репозитории отсутствуют временные файлы компиляции (.o, .obj).

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между операторами 'И' (&&) и 'ИЛИ' (||) в сложных условиях?

2. Как работает 'короткое замыкание' (short-circuit) при вычислении логических выражений?

3. Почему в операторе switch-case после каждого блока case обычно ставится break?

4. В каких случаях цикл do-while предпочтительнее обычного while?

5. Что произойдет, если в условии цикла for пропустить все три выражения?

6. Как принудительно завершить выполнение текущей итерации цикла и перейти к следующей?

7. Какое значение возвращает логическое выражение в C++, если оно истинно?

8. Можно ли использовать вещественные числа (float/double) в качестве констант в switch?

9. Как избежать зацикливания при использовании переменной цикла внутри тела?

# ПРАКТИЧЕСКАЯ РАБОТА №12. ФУНКЦИИ И МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

## Цели и задачи

Цель практической работы: Изучение механизмов создания пользовательских функций, освоение способов передачи параметров (по значению, ссылке, указателю) и принципов разделения кода на модули.

Основные задачи:

– Изучить синтаксис объявления и определения функций в C++;

– Освоить передачу параметров по ссылке (&) и указателю (*) для изменения внешних переменных;

– Научиться использовать значения аргументов по умолчанию и перегрузку функций.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork12.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Функции разделяются на прототипы (объявления) и определения.
3. Демонстрируется разница между локальными и статическими переменными.
4. Особое внимание уделяется константным ссылкам для оптимизации передачи объектов.
5. Каждый логический блок оформляется в виде отдельной функции.
6. Итоговый код синхронизируется с GitHub.

**Задание 1. Базовые функции и перегрузка.**

*Создание набора перегруженных функций для обработки данных разных типов.*

– Написать функцию sum, принимающую два целых числа и возвращающую их сумму.

– Перегрузить функцию sum для работы с тремя целыми числами и с двумя числами типа double.

– Реализовать функцию printLine, которая выводит строку из символов '-' заданной длины (по умолчанию 20).

**Задание 2. Передача параметров по ссылке и указателю.**

*Отработка механизмов изменения данных вне области видимости*

*функции.*

– Написать функцию swap, которая меняет местами значения двух переменных, переданных по ссылке.

– Реализовать аналогичную функцию swap, использующую указатели для обмена значений.

– Создать функцию circleInfo, которая принимает радиус и через параметры-ссылки возвращает одновременно площадь и длину окружности.

**Задание 3. Рекурсивные алгоритмы.**

*Реализация функций, вызывающих самих себя для решения подзадач.*

– Написать рекурсивную функцию для вычисления n-го числа Фибоначчи.

– Реализовать функцию нахождения наибольшего общего делителя (НОД) двух чисел по алгоритму Евклида.

– Сравнить производительность рекурсивного и итерационного (циклического) способов вычисления факториала.

**Задание 4. Математическое моделирование в функциях.**

*Применение функций для решения прикладных вычислительных задач.*

– Написать функцию isPrime, которая возвращает true, если число простое, и false в противном случае.

– Реализовать функцию решения квадратного уравнения, принимающую коэффициенты и возвращающую количество корней (сами корни передавать через ссылки).

– Оформить ввод данных в main и вызов соответствующих расчетных функций.

**Задание 5. Статические переменные и области видимости.**

*Изучение жизненного цикла переменных внутри функциональных блоков.*

– Создать функцию callCounter, которая при каждом вызове выводит в консоль, сколько раз она была вызвана (использовать static).

– Продемонстрировать конфликт имен, создав глобальную и локальную переменные с одинаковым названием, и обратившись к глобальной через оператор '::'.

– Реализовать функцию-генератор простых чисел, сохраняющую состояние последнего найденного числа.

**Задание 6. Организация проекта и Git.**

*Подготовка чистого кода и фиксация прогресса в системе контроля версий.*

– Вынести прототипы всех созданных функций в начало файла перед функцией main.

– Добавить комментарии в стиле Doxygen (///) к каждой функции с описанием параметров и возвращаемого значения.

– Сделать коммит 'Lab 3: Functions and modularity' и отправить изменения в удаленный репозиторий.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между объявлением (прототипом) и определением функции?

2. Почему передача больших объектов по ссылке эффективнее, чем по значению?

3. Что такое 'константная ссылка' (const &) и зачем её использовать в параметрах?

4. Как работает перегрузка функций и по каким признакам компилятор их различает?

5. В каких случаях использование рекурсии может привести к переполнению стека (stack overflow)?

6. Какую роль играет ключевое слово inline перед определением функции?

7. Можно ли вернуть из функции массив в стиле C напрямую? Какие есть альтернативы?

8. Зачем нужны значения аргументов по умолчанию и в какой части функции они указываются?

9. Чем отличается автоматическая переменная от статической (static) внутри функции?

# ПРАКТИЧЕСКАЯ РАБОТА №13. МАССИВЫ И РАБОТА С ПАМЯТЬЮ

## Цели и задачи

Цель практической работы: Изучение способов хранения наборов данных, освоение адресной арифметики и механизмов выделения динамической памяти в куче (heap).

Основные задачи:

– Изучить синтаксис объявления статических массивов и способы их инициализации;

– Освоить работу с указателями (*), взятие адреса (&) и разыменование;

– Научиться выделять и освобождать память для одномерных и двумерных массивов в процессе выполнения программы.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork13.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.

2. Сравниваются статические массивы и динамические, созданные через оператор new.

3. Демонстрируется связь между указателями и массивами.

4. Особое внимание уделяется предотвращению утечек памяти через своевременный вызов delete.

5. Для отладки корректности работы с памятью рекомендуется использовать встроенные средства дебаггера.

**Задание 1. Статические массивы и алгоритмы обработки.**

*Реализация базовых операций над наборами данных фиксированного размера.*

– Создать статический массив из 10 целых чисел и заполнить его случайными значениями.

– Написать функцию, которая находит минимальный и максимальный элементы массива за один проход.

– Реализовать алгоритм реверсии (разворота) массива без использования дополнительных массивов.

**Задание 2. Указатели и адресная арифметика.**

*Изучение низкоуровневого доступа к данным через адреса в памяти.*

– Объявить указатель на переменную и изменить её значение через разыменование указателя.

– Продемонстрировать доступ к элементам массива, используя инкремент указателя вместо индексации (ptr++).

– Создать функцию, принимающую указатель на массив и его размер, для вычисления суммы всех элементов.

**Задание 3. Динамическое выделение памяти (new / delete).**

*Работа с объектами, размер которых становится известен только во время выполнения.*

– Запросить у пользователя размер массива N и выделить под него память в куче через new int[N].

– Заполнить динамический массив, выполнить над ним вычисления и корректно освободить память через delete[].

– Написать программу, которая динамически расширяет массив (создает новый большего размера и копирует данные), имитируя работу контейнера.

**Задание 4. Многомерные динамические массивы.**

*Создание матриц и структур данных с вложенной адресацией.*

– Реализовать динамическое выделение памяти под двумерную матрицу (массив указателей на массивы).

– Написать функцию для вывода матрицы в консоль в виде ровной сетки.

– Обеспечить построчное удаление памяти в обратном порядке во избежание утечек.

**Задание 5. Строки в стиле C и std::string.**

*Сравнение работы с массивами символов и современными строковыми*

*объектами.*

– Создать массив символов char[] и продемонстрировать его завершение нулевым символом.

– Использовать функции библиотеки cstring (strlen, strcpy) для манипуляции символьными массивами.

– Реализовать поиск подстроки в объекте std::string, введенном пользователем.

**Задание 6. Безопасность памяти и Git.**

*Проверка кода на ошибки и фиксация стабильной версии.*

– Проверить программу в отладчике на предмет обращения к несуществующим индексам массива.

– Убедиться, что для каждого вызова new в коде присутствует соответствующий delete.

– Сделать коммит 'Lab 4: Pointers and dynamic memory managed' и отправить изменения на GitHub.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между стеком (stack) и кучей (heap) при выделении памяти?

2. Почему размер статического массива должен быть константой времени компиляции?

3. Что такое 'нулевой указатель' (nullptr) и почему важно проверять указатели перед использованием?

4. Каким образом имя массива связано с указателем на его первый элемент?

5. Что произойдет, если забыть вызвать delete для динамически выделенной памяти?

6. В чем заключается опасность выхода за границы массива (buffer overflow)?

7. Как передать массив в функцию так, чтобы она не могла изменить его элементы?

8. Что такое 'утечка памяти' (memory leak) и как её обнаружить?

9. Как работает оператор разыменования (*) применительно к указателю?

# ПРАКТИЧЕСКАЯ РАБОТА №14. СТРУКТУРЫ ДАННЫХ И ОСНОВЫ ООП

## Цели и задачи

Цель практической работы: Изучение способов создания пользовательских типов данных, освоение принципов инкапсуляции и разграничения доступа к данным объекта.

Основные задачи:

– Изучить синтаксис объявления структур и классов в C++;

– Освоить создание конструкторов (включая конструктор по умолчанию и с параметрами);

– Научиться использовать методы доступа (геттеры и сеттеры) для защиты внутренних данных.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork14.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Сравнивается использование struct и class.
3. Демонстрируется применение модификаторов доступа (public, private).
4. Создаются конструкторы и деструкторы.
5. Логика классов разделяется на заголовочные файлы (.h) и файлы реализации (.cpp).
6. Итоговый проект фиксируется в Git.

**Задание 1. Простые структуры данных.**

*Организация связанных данных в единый пользовательский тип*.

– Создать структуру Point (точка на плоскости) с полями x и y.

– Написать функцию, вычисляющую расстояние между двумя объектами типа Point.

– Реализовать массив структур 'Студент' (имя, средний балл) и отсортировать его по убыванию баллов.

**Задание 2. Переход к классам и инкапсуляции.**

*Создание защищенных объектов с контролируемым доступом к полям*.

– Преобразовать структуру в класс Rectangle (прямоугольник) с

приватными полями ширины и высоты.

– Реализовать публичные методы setWidth и setHeight с проверкой на отрицательные значения.

– Добавить методы для вычисления площади и периметра фигуры.

**Задание 3. Жизненный цикл объекта (Конструкторы).**

*Управление инициализацией и очисткой ресурсов объекта*.

– Добавить в класс Rectangle конструктор с параметрами для мгновенной инициализации размеров.

– Реализовать деструктор, выводящий сообщение о завершении работы объекта в консоль.

– Продемонстрировать автоматический вызов деструктора при выходе объекта из области видимости.

**Задание 4. Методы и ключевое слово this.**

*Работа с контекстом объекта и внутренними указателями*.

– Добавить метод compareTo, который принимает другой объект Rectangle и возвращает true, если текущий прямоугольник больше.

– Использовать указатель this для разрешения конфликтов имен между параметрами метода и полями класса.

– Реализовать константные методы (const), которые гарантируют неизменность данных объекта.

**Задание 5. Композиция объектов.**

*Создание сложных типов данных на основе уже существующих классов*.

– Создать класс Circle (круг), использующий объект класса Point в качестве центра координат.

– Реализовать метод, проверяющий, находится ли заданная точка внутри данного круга.

– Оформить ввод параметров круга и точки с клавиатуры в функции main.

**Задание 6. Архитектура проекта и Git.**

*Синхронизация кода и документирование интерфейса классов.*

– Выделить описание класса в отдельный файл (например, Rectangle.h) и подключить его через #include.

– Добавить описание каждого метода в README.md в виде краткой технической спецификации.

– Сделать коммит 'Lab 5: Classes and encapsulation introduced' и отправить изменения на GitHub.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем заключается принципиальное отличие struct от class в языке C++?

2. Что такое 'инкапсуляция' и зачем скрывать поля класса в секции private?

3. Может ли у класса быть несколько конструкторов? Как компилятор понимает, какой вызвать?

4. Какую задачу выполняет деструктор и может ли он принимать аргументы?

5. Для чего нужны методы-геттеры и методы-сеттеры?

6. Зачем помечать методы как const и что произойдет, если попытаться изменить поле в таком методе?

7. Что такое 'список инициализации конструктора' и почему он предпочтительнее присваивания в теле?

8. Какую роль играет указатель this и какой у него тип данных?

9. Как создать массив объектов класса в динамической памяти?

# ПРАКТИЧЕСКАЯ РАБОТА №15. НАСЛЕДОВАНИЕ И ПОЛИМОРФИЗМ

## Цели и задачи

Цель практической работы: Изучение механизмов повторного использования кода через наследование и освоение динамического полиморфизма с использованием виртуальных функций.

Основные задачи:

– Изучить синтаксис наследования и модификаторы доступа (public, protected, private);

– Освоить объявление и переопределение виртуальных функций (virtual, override);

– Научиться создавать абстрактные классы с чисто виртуальными функциями.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork15.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Реализуется иерархия классов (базовый и производные).
3. Демонстрируется разница между ранним и поздним связыванием.
4. Используются указатели на базовый класс для управления объектами производных классов.
5. Особое внимание уделяется виртуальным деструкторам для предотвращения утечек памяти.

**Задание 1. Базовое наследование.**

*Создание иерархии классов с общими свойствами и поведением*.

– Создать базовый класс Shape (фигура) с защищенным полем 'название' и цветом.

– Реализовать производный класс Rectangle, наследующий Shape, и добавить ему специфические поля (ширина, высота).

– Продемонстрировать вызов конструктора базового класса из списка инициализации конструктора потомка.

**Задание 2. Виртуальные функции и полиморфизм.**

*Отработка механизма динамического определения вызываемого метода*.

– Добавить в класс Shape виртуальный метод draw(), выводящий информацию о фигуре.

– Переопределить метод draw() в классах Rectangle и Circle, используя ключевое слово override.

– В функции main создать массив указателей Shape* и вызвать метод draw() для объектов разных типов в цикле.

**Задание 3. Абстрактные классы и интерфейсы.**

*Проектирование строгих контрактов для дочерних классов.*

– Сделать класс Shape абстрактным, объявив чисто виртуальную функцию getArea() = 0.

– Реализовать расчет площади в каждом производном классе (Rectangle, Circle, Triangle).

– Убедиться, что создание экземпляра абстрактного класса Shape теперь невозможно (ошибка компиляции).

**Задание 4. Управление доступом protected.**

*Изучение видимости полей внутри иерархии наследования.*

– Изменить поля базового класса с private на protected и убедиться, что они доступны в методах потомков.

– Попытаться обратиться к protected-полям напрямую из функции main и зафиксировать ограничение доступа.

– Реализовать метод в потомке, который модифицирует состояние базового класса.

**Задание 5. Виртуальные деструкторы.**

*Обеспечение корректной очистки памяти при полиморфном удалении объектов.*

– Добавить вывод сообщения в деструкторы Shape и Rectangle.

– Создать объект Rectangle через указатель Shape* и удалить его через delete.

– Сделать деструктор в Shape виртуальным и пронаблюдать правильный порядок вызова деструкторов обоих классов.

**Задание 6. Документирование и Git.**

*Отработка фиксации объектно-ориентированной архитектуры*.

– Нарисовать в README.md простую схему иерархии классов (в виде списка или текстовой диаграммы).

– Сделать коммит 'Lab 6: Inheritance and polymorphism implemented' и отправить его на GitHub.

– Убедиться, что файлы реализации разделены по смыслу (Base.h, Derived.h).

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем разница между public и protected наследованием?

2. Что такое 'таблица виртуальных функций' (VTable) и как она работает?

3. Зачем нужно ключевое слово override и какие ошибки оно помогает предотвратить?

4. Может ли абстрактный класс иметь реализацию некоторых методов или конструктор?

5. Почему деструктор базового класса обязан быть виртуальным при наличии виртуальных методов?

6. Что произойдет, если производный класс не реализует чисто виртуальную функцию?

7. Как вызвать метод базового класса из переопределенного метода потомка?

8. В чем отличие статического связывания от динамического?

9. Можно ли создать массив объектов (не указателей) абстрактного класса?

# ПРАКТИЧЕСКАЯ РАБОТА №16. ШАБЛОНЫ И СТАНДАРТНАЯ БИБЛИОТЕКА КОНТЕЙНЕРОВ (STL)

## Цели и задачи

Цель практической работы: Изучение принципов обобщенного программирования, создание универсальных функций и классов, а также освоение базовых контейнеров и итераторов библиотеки STL.

Основные задачи:

– Изучить синтаксис объявления шаблонов (template <typename T>);

– Освоить работу с динамическим массивом std::vector и его методами (push_back, size, clear);

– Научиться использовать ассоциативные массивы std::map для хранения пар 'ключ-значение'.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork16.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.
2. Реализуются собственные шаблоны функций и классов.
3. Изучаются контейнеры std::vector, std::list и std::map.
4. Демонстрируется использование итераторов для обхода коллекций.
5. Код проверяется на эффективность и универсальность.
6. Итоговые решения синхронизируются с GitHub.

**Задание 1. Шаблоны функций.**

*Разработка универсальных алгоритмов, работающих с различными типами данных.*

– Написать шаблонную функцию findMax, возвращающую максимальное из двух значений любого типа.

– Реализовать шаблонную функцию swapValues для обмена значениями переменных (int, double, string).

– Создать шаблон функции для вывода любого массива в консоль через пробел.

**Задание 2. Шаблонные классы.**

*Создание обобщенных структур данных для хранения элементов произвольного типа.*

– Реализовать шаблонный класс Box (коробка), хранящий один элемент типа T.

– Добавить в класс методы для записи (set) и получения (get) содержимого.

– Продемонстрировать работу класса Box с типами int и std::string в функции main.

**Задание 3. Работа с вектором (std::vector).**

*Освоение наиболее часто используемого контейнера stl для динамических данных.*

– Создать вектор целых чисел, заполнить его значениями и удалить из него все четные числа.

– Использовать метод std::sort из библиотеки <algorithm> для сортировки вектора.

– Реализовать поиск элемента в векторе и вывести его индекс, если он найден.

**Задание 4. Ассоциативные контейнеры (std::map).**

*Организация быстрого поиска и хранения данных по ключу.*

– Создать телефонную книгу (std::map<string, string>), где ключ — имя, а значение — номер телефона.

– Реализовать добавление новых контактов и поиск номера по введенному имени.

– Вывести весь список контактов в алфавитном порядке имен.

**Задание 5. Итераторы и обход контейнеров.**

*Изучение универсального способа доступа к элементам любой коллекции.*

– Продемонстрировать обход вектора с использованием итераторов (vector<int>::iterator).

– Использовать константные итераторы (const_iterator) для безопасного чтения данных.

– Применить современный цикл 'range-based for' (for (const auto& item : container)) для вывода данных.

**Задание 6. Синхронизация и фиксация STL-проекта.**

*Документирование использования контейнеров и отправка кода в репозиторий.*

– Добавить в README.md краткую таблицу со сравнением производительности vector и list (на основе теории).

– Сделать коммит 'Lab 7: Templates and STL containers applied' и отправить изменения на GitHub.

– Убедиться, что в коде подключены необходимые заголовочные файлы (<vector>, <map>, <algorithm>).

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем заключается идея обобщенного программирования (Generics)?

2. Где должен находиться код реализации шаблона (в .h или .cpp файле) и почему?

3. В чем разница между std::vector и обычным динамическим массивом?

4. Что происходит с итераторами вектора при добавлении в него элементов (инвалидация)?

5. Какова сложность поиска элемента в std::map по сравнению со std::vector?

6. Для чего нужны итераторы и почему их называют 'обобщенными указателями'?

7. Как отсортировать контейнер в порядке убывания с помощью std::sort?

8. В чем разница между методом .at(i) и оператором [i] при доступе к вектору?

9. Что такое 'лямбда-выражение' и как оно может упростить работу с алгоритмами STL?

# ПРАКТИЧЕСКАЯ РАБОТА №17. ПОТОКИ ДАННЫХ И ОБРАБОТКА ИСКЛЮЧЕНИЙ

## Цели и задачи

Цель практической работы: Изучение механизмов файлового ввода-вывода, освоение способов обработки исключительных ситуаций и выполнение итогового консольного проекта.

Основные задачи:

– Изучить режимы открытия файлов (ios::in, ios::out, ios::app);

– Освоить синтаксис генерации исключений (throw) и их перехвата (catch);

– Научиться использовать стандартные исключения из библиотеки <stdexcept>.

*Формируемые компетенции:* ОПК-2

## Теоретическое обоснование

Перед выполнением практической работы необходимо ознакомиться с лекционным материалом.

## Оборудование и материалы

Для выполнения практической работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): google chrome / mozilla firefox, visual studio code, clion / microsoft visual studio 2022.

## Методика и порядок выполнения работы

Перед выполнением заданий рекомендуется ознакомиться с решением учебной задачи:

https://github.com/vberezina/ITiP/blob/main/LabWork17.md

***Порядок выполнения работы:***

1. Работа выполняется в VS Code.

2. Для работы с файлами используются классы ifstream и ofstream библиотеки <fstream>.

3. Ошибки обрабатываются через блоки try, catch и оператор throw.

4. Проект организуется в виде нескольких файлов (заголовочные и исходные).

5. Финальный программный продукт должен демонстрировать устойчивость к некорректным входным данным и сохранять состояние в текстовый файл.

**Задание 1. Файловый ввод-вывод.**

*Создание системы сохранения и загрузки данных из внешних текстовых файлов.*

– Написать программу, которая записывает массив объектов (например, 'Товар': название, цена) в файл data.txt.

– Реализовать чтение данных из этого файла и восстановление объектов в оперативной памяти (в векторе).

– Добавить проверку успешности открытия файла с помощью метода .is_open().

**Задание 2. Обработка исключительных ситуаций.**

*Повышение отказоустойчивости программы при возникновении логических ошибок.*

– Написать функцию деления двух чисел, которая генерирует исключение std::runtime_error при попытке деления на ноль.

– Реализовать блок try-catch в main для перехвата этого исключения и вывода информативного сообщения пользователю.

– Добавить обработку исключения std::out_of_range при обращении к несуществующему индексу вектора через метод .at().

**Задание 3. Парсинг текстовых файлов.**

*Обработка структурированной текстовой информации из внешних источников.*

– Подготовить файл config.txt с настройками в формате 'ключ=значение'.

– Написать скрипт, который считывает файл построчно, разделяет строки по знаку '=' и сохраняет настройки в std::map.

– Организовать пропуск пустых строк и строк-комментариев, начинающихся с символа '#'.

**Задание 4. Итоговый мини-проект: Консольная база данных.**

*Объединение всех навыков ooп, stl и работы с файлами в единую программу.*

– Разработать систему управления списком книг или студентов (добавление, удаление, просмотр).

– Реализовать автоматическое сохранение всей базы в файл при выходе из программы.

– Добавить автоматическую загрузку данных из файла при старте приложения.

**Задание 5. Валидация данных и логирование.**

*Контроль корректности вводимой информации и запись истории событий.*

– Создать функцию валидации ввода, которая заставляет пользователя вводить числовые значения там, где это необходимо.

– Реализовать 'логгер' — запись всех ошибок, возникших в процессе работы программы, в отдельный файл error.log.

– Продемонстрировать работу программы при попытке загрузки поврежденного или отсутствующего файла.

**Задание 6. Финальная публикация и Git.**

*Завершение курса и подготовка репозитория к итоговой аттестации*.

– Собрать все заголовочные файлы и файлы реализации в структурированный проект (папки src и include).

– Написать в README.md итоговую инструкцию по сборке и запуску вашего приложения.

– Сделать финальный коммит 'Course complete: Final project with file I/O and exceptions' и выполнить push.

## Содержание отчета и его форма

Отчет по практической работе должен содержать:

1. Номер и название практической работы; задачи практической работы.

2. Реализация каждого задания с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

Отчет о выполнении практической работы подписывается студентом и сдается преподавателю.

## Контрольные вопросы

1. В чем преимущество использования исключений перед возвратом кодов ошибок (например, -1)?

2. Что такое 'раскрутка стека' (stack unwinding) при возникновении исключения?

3. Какую роль играют манипуляторы потоков (например, std::endl, std::setw)?

4. Как открыть файл одновременно для чтения и для записи?

5. В чем разница между текстовым и бинарным режимами работы с файлами?

6. Как перехватить любое исключение, тип которого заранее неизвестен?

7. Почему не рекомендуется генерировать исключения в деструкторах классов?

8. Как проверить, что при чтении из файла был достигнут его конец (EOF)?

9. Что такое 'утечка ресурса' при исключении и как её избежать с помощью идиомы RAII?

# СПИСОК ЛИТЕРАТУРЫ

*Основная литература:*

1. Страуструп Б. Программирование: принципы и практика с использованием C++. — М.: Вильямс, 2016

2. Лафоре Р. Объектно-ориентированное программирование в C++. — СПб.: Питер, 2013

3. Лутц М. Изучаем Python, том 1. — М.: Диалектика, 2019

4. Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов. — М.: Эксмо, 2013

5. Чакон С., Штрауб Б. Pro Git. [электронный ресурс] — https://git-scm.com

6. Документация Typst: руководство по современной верстке. [электронный ресурс] — https://typst.app

7. Справочник по языку C++ и стандартной библиотеке STL. [электронный ресурс] — https://cppreference.com

8. MDN Web Docs: руководство по технологиям HTML и CSS. [электронный ресурс] — https://mozilla.org

*Дополнительная литература:*

9. Мейерс С. Эффективный современный C++. 42 рекомендации по использованию C++11 и C++14. — М.: Вильямс, 2016

10. Блохович Д. C++, шаблоны. Полное руководство. — М.: Вильямс, 2018

11. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017

12. Фримен Э., Робсон Э. Изучаем HTML5. — СПб.: Питер, 2012

13. C++ Core Guidelines. Bjarne Stroustrup and Herb Sutter. [электронный ресурс] — https://github.io

14. Python Design Patterns. Глава по паттернам проектирования. [электронный ресурс] — https://python-patterns.guide

15. Awesome Typst. Подборка шаблонов и инструментов для Typst. [электронный ресурс] — https://github.com

16. CSS-Tricks. Полное руководство по Flexbox и современным сеткам. [электронный ресурс] — https://css-tricks.com

*Интернет-ресурсы:*

17. Книга pro git (скотт чакон):

https://git-scm.com

Полное руководство по системе контроля версий git, охватывающее все аспекты от основ создания коммитов до работы с удаленными ветками и серверами

18. Документация typst:

https://typst.app

Официальный справочник по современной системе верстки документов, содержащий инструкции по синтаксису, вставке формул и созданию шаблонов

19. Портал mdn web docs:

https://mozilla.org

Всеобъемлющий справочник по технологиям html5 и css3 с подробным описанием тегов, свойств и интерактивными примерами верстки

20. Официальная документация python:

https://python.org

Полный перечень стандартных библиотек, типов данных и функций языка python с актуальными примерами использования для автоматизации задач

21. Справочник cppreference:

https://cppreference.com

Наиболее точный и актуальный онлайн-справочник по стандартам языка c++, контейнерам stl, алгоритмам и управлению памятью

22. Образовательный ресурс learncpp:

https://learncpp.com

Пошаговое руководство по изучению современного c++ с акцентом на лучшие практики программирования и архитектуру объектно-ориентированных систем

23. Руководство по flexbox (css-tricks):

https://css-tricks.com

Визуальное пособие по современной модели гибкой верстки, объясняющее принципы позиционирования элементов и создания адаптивных интерфейсов