Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Структурное подразделение «Сибирская школа геонаук (119)»

УТВЕРЖДЕНА:
на заседании ДОТ
Протокол №29 от 10 апреля 2025 г.

Рабочая программа дисциплины

«ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ / PROGRAMMING IN HIGH-LEVEL LANGUAGES»

Направление: 09.03.02 Информационные системы и технологии

Информационные технологии в науках о Земле и окружающей среде / Information Technologies in Earth and Environmental Sciences

Квалификация: Бакалавр

Форма обучения: очная

Документ подписан простой электронной подписью Составитель программы: Ланько Анна Викторовна Дата подписания: 14.12.2025	Документ подписан простой электронной подписью Утвердил: Ланько Анна Викторовна Дата подписания: 14.12.2025	Документ подписан простой электронной подписью Согласовал: Паршин Александр Вадимович Дата подписания: 13.01.2026

Год набора – 2025

Иркутск, 2025 г.

1 Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы

1.1 Дисциплина «Программирование на языке высокого уровня / Programming in High-Level Languages» обеспечивает формирование следующих компетенций с учётом индикаторов их достижения

Код, наименование компетенции	Код индикатора компетенции
ОПК ОС-6 Способность разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	ОПК ОС-6.1

1.2 В результате освоения дисциплины у обучающихся должны быть сформированы

Код индикатора	Содержание индикатора	Результат обучения
ОПК ОС-6.1	Способен составлять и реализовывать алгоритмы с использованием языка программирования высокого уровня	**Знать** алгоритмы и конструкцию языка программирования высокого уровня для разработки программ информационных систем (ИС). **Уметь** составлять и реализовывать алгоритмы на языке высокого уровня для практического применения в ИС и технологиях. **Владеть** навыками программирования высокого уровня для создания эффективных алгоритмических решений ИС.

2 Место дисциплины в структуре ООП

Изучение дисциплины «Программирование на языке высокого уровня / Programming in High-Level Languages» базируется на результатах освоения следующих дисциплин/практик: Нет

Дисциплина является предшествующей для дисциплин/практик: «Проектная деятельность / Project Development Practicum», «Объектно-ориентированные технологии и шаблоны проектирования /Object-Oriented Technologies and Design Patterns», «Основы робототехники / Fundamentals of Robotics»

3 Объем дисциплины

Объем дисциплины составляет – 7 ЗЕТ

Вид учебной работы	Трудоемкость в академических часах (Один академический час соответствует 45 минутам астрономического часа)		
	Всего	Семестр № 3	Семестр № 4
Общая трудоемкость дисциплины	252	108	144
Аудиторные занятия, в том	99	45	54

числе:			
лекции	33	15	18
лабораторные работы	66	30	36
практические/семинарские занятия	0	0	0
Самостоятельная работа (в т.ч. курсовое проектирование)	117	63	54
Трудоемкость промежуточной аттестации	36	0	36
Вид промежуточной аттестации (итогового контроля по дисциплине)	, Экзамен, Курсовой проект		Экзамен, Курсовой проект

4 Структура и содержание дисциплины

4.1 Сводные данные по содержанию дисциплины

Семестр № 3

№ п/п	Наименование раздела и темы дисциплины	Виды контактной работы						СРС		Форма текущего контроля
		Лекции		ЛР		ПЗ(СЕМ)				
		№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	
1	2	3	4	5	6	7	8	9	10	11
1	1. Введение в языки программирования высокого уровня	1	2							Устный опрос
2	2. Основы синтаксиса и типы данных	2	2	1	4			4	10	Устный опрос
3	3. Структуры управления выполнением программы	3	2	2	5					Устный опрос
4	4. Структуры данных: списки, кортежи, множества, словари	4	4	3, 4	12			1, 2, 4	26	Устный опрос
5	5. Функции и модульность кода	5	3	5	5			4	10	Устный опрос
6	6. Работа с файлами и обработка исключений	6	2	6	4			3, 4	17	Устный опрос
	Промежуточная аттестация									
	Всего		15		30				63	

Семестр **№ 4**

№ п/п	Наименование раздела и темы дисциплины	Виды контактной работы						СРС		Форма текущего контроля
		Лекции		ЛР		ПЗ(СЕМ)				
		№	Кол.	№	Кол.	№	Кол.	№	Кол.	

			Час.		Час.		Час.		Час.	
1	2	3	4	5	6	7	8	9	10	11
1	7. Объектно-ориентированное программирование в Python	1	2					1, 2	24	Устный опрос
2	8. Продвинутые структуры данных и алгоритмы	2	2	1	6			7	6	Устный опрос
3	9. Работа с базами данных и SQL в Python	3	4	2, 3	12			3, 5	14	Устный опрос
4	10. Веб-программирование и API	4	4	4	6			4	4	Устный опрос
5	11. Многопоточность и асинхронное программирование	5	4	5	6			6	6	Устный опрос
6	12. Тестирование и отладка программ	6	2	6	6					Устный опрос
	Промежуточная аттестация								36	Экзамен, Курсовой проект
	Всего		18		36				90	

4.2 **Краткое содержание разделов и тем занятий**

Семестр № 3

№	Тема	Краткое содержание
1	1. Введение в языки программирования высокого уровня	Классификация языков программирования (низкого и высокого уровня), характеристики языков высокого уровня (Python, Java, C#), преимущества интерпретируемости и компилируемости, установка и настройка среды разработки (интегрированная среда разработки, IDE), первая программа "Hello, World!", синтаксические особенности Python как языка высокого уровня.
2	2. Основы синтаксиса и типы данных	Переменные и их объявление, встроенные типы данных (целочисленные, числа с плавающей точкой, строки, логические значения), операции присваивания, арифметические операции (+, -, *, /, //, %, **), операции сравнения (==, !=, >, , >=, =), логические операции (and, or, not), преобразование типов (int(), float(), str(), bool()).
3	3. Структуры управления выполнением программы	Условные операторы (if, elif, else), вложенные условия, тернарный оператор, циклы while и for (с range(), enumerate()), операторы break, continue, pass, вложенные циклы, практические задачи на ветвление и циклы для обработки данных информационных систем.

4	4. Структуры данных: списки, кортежи, множества, словари	Создание и операции со списками (list: индексация, срезы, методы append(), extend(), pop(), sort()), кортежи (tuple: неизменяемость, распаковка), множества (set: уникальные элементы, операции объединения, пересечения, разности), словари (dict: ключи-значения, методы get(), keys(), values(), items()), comprehensions для списков и словарей.
5	5. Функции и модульность кода	Определение функций (def), параметры (позиционные, именованные, по умолчанию, *args, **kwargs), возвращаемые значения (return), область видимости (local, global, nonlocal), лямбда-функции, функциональное программирование (map(), filter(), reduce()), импорт модулей (import, from ... import), стандартная библиотека (math, random, datetime).
6	6. Работа с файлами и обработка исключений	Чтение и запись файлов (open(), read(), write(), readline(), writelines()), контекстные менеджеры (with), текстовые и бинарные файлы, форматы CSV и JSON, иерархия исключений (Exception, ValueError, FileNotFoundError, KeyError), конструкции try-except-else-finally, создание пользовательских исключений.

Семестр **№ 4**

№	Тема	Краткое содержание
1	7. Объектно-ориентированное программирование в Python	Классы и объекты, конструктор (init), методы экземпляра и классовые методы (@classmethod), статические методы (@staticmethod), наследование (однократное и множественное), полиморфизм, магические методы (str, repr, len, getitem), инкапсуляция (private атрибуты с _ и __).
2	8. Продвинутые структуры данных и алгоритмы	Списки смежности для графов, словари для хэш-таблиц, очереди (collections.deque), стеки, алгоритмы сортировки (сортировка пузырьком, выбором, вставками, merge sort), алгоритмы поиска (линейный, бинарный), рекурсия, глубина рекурсии и оптимизация хвостовой рекурсии.
3	9. Работа с базами данных и SQL в Python	Подключение к базам данных SQLite (sqlite3), параметризованные запросы, операции CRUD (Create, Read, Update, Delete), работа с PostgreSQL и MySQL через библиотеки, объектно-реляционное отображение (ORM, Object-Relational Mapping) с SQLAlchemy, транзакции и обработка ошибок БД.
4	10. Веб-программирование и API	Библиотека requests для HTTP-запросов (GET, POST, PUT, DELETE), обработка JSON-ответов, работа с REST API, фреймворк Flask (минимальный веб-сервер, маршруты, шаблоны Jinja2), создание простых веб-приложений для информационных систем.

5	11. Многопоточность и асинхронное программирование	Модуль threading (потоки, блокировка Lock, семафоры), модуль multiprocessing (процессы, Queue, Pool), асинхронное программирование (asyncio, async/await, coroutines), библиотека aiohttp для асинхронных HTTP-запросов, обработка параллельных задач в ИС.
6	12. Тестирование и отладка программ	Юнит-тестирование (unittest, pytest), тесты функций и классов, мок-объекты, покрытие кода, отладка (pdb, IDE debuggers), профилирование производительности (cProfile, timeit), логирование (logging модуль), документация кода (docstrings, Sphinx).

4.3 Перечень лабораторных работ

Семестр № 3

№	Наименование лабораторной работы	Кол-во академических часов
1	Лабораторная работа №1. Работа с типами данных и базовыми операциями	4
2	Лабораторная работа №2. Алгоритмы ветвления и циклов	5
3	Лабораторная работы №3. Работа со структурами данных Python	6
4	Лабораторная работа №4. Алгоритмы обработки коллекций данных	6
5	Лабораторная работа №5. Разработка модульных программ с функциями	5
6	Лабораторная работа №6. Обработка файлов и исключений в ИС	4

Семестр **№ 4**

№	Наименование лабораторной работы	Кол-во академических часов
1	Лабораторная работа №7. Реализация алгоритмов сортировки и поиска	6
2	Лабораторная работа №8. Разработка приложений с SQLite	6
3	Лабораторная работа №9. Интеграция с реляционными БД	6
4	Лабораторная работа №10. Создание REST API на Flask	6
5	Лабораторная работа №11. Параллельное выполнение задач	6
6	Лабораторная работа №12. Автоматизированное тестирование ПО	6

4.4 Перечень практических занятий

Практических занятий не предусмотрено

4.5 Самостоятельная работа

Семестр № 3

№	Вид СРС	Кол-во академических часов
1	Оформление отчетов по лабораторным и практическим работам	6
2	Подготовка к практическим занятиям (лабораторным работам)	10
3	Подготовка к сдаче и защите отчетов	7
4	Проработка разделов теоретического материала	40

Семестр **№ 4**

№	Вид СРС	Кол-во академических часов
1	Написание курсового проекта (работы)	20
2	Оформление отчетов по лабораторным и практическим работам	4
3	Подготовка к практическим занятиям (лабораторным работам)	4
4	Подготовка к сдаче и защите отчетов	4
5	Подготовка к экзамену	10
6	Подготовка презентаций	6
7	Проработка разделов теоретического материала	6

В ходе проведения занятий по дисциплине используются следующие интерактивные методы обучения: работа в малых группах

5 Перечень учебно-методического обеспечения дисциплины

5.1 Методические указания для обучающихся по освоению дисциплины

5.1.1 Методические указания для обучающихся по курсовому проектированию/работе:

Общие положения
Курсовое проектирование (КП) представляет собой практическую разработку полнофункционального программного приложения информационной системы (ИС) на языке Python, демонстрирующее компетенцию составления и реализации алгоритмов для практического применения в ИС и технологиях. Типовое задание: создание веб-приложения для управления данными (например, система учета студентов, складской учет, мониторинг метрик ИС) с использованием Flask/SQLite, алгоритмов обработки данных, многопоточностью и автоматизированным тестированием.
Процедура подготовки:
1) выбор темы и утверждение плана (1 неделя),
2) разработка и тестирование (6 недель),
3) написание отчета (1 неделя); защита включает 7-минутную презентацию с демонстрацией кода/приложения и ответы на вопросы комиссии.
Защита оценивается по критериям:
полнота реализации (40%),
качество кода (30%),

документация/тесты (20%),
презентация (10%).
Не допуск к защите при невыполнении 30% плана.
Рекомендации к определению целей и задач КП
Цели КП формулируются с учетом компетенции: "Разработать алгоритмически эффективное веб-приложение ИС на Python, интегрирующее структуры данных, БД, API и параллелизм для практического применения в информационных технологиях".
Задачи:
1) спроектировать модель данных и алгоритмы (сортировка, поиск, агрегация);
2) реализовать REST API на Flask с CRUD-операциями;
3) интегрировать SQLite/PostgreSQL с ORM;
4) обеспечить многопоточную/асинхронную обработку;
5) разработать юнит-тесты (покрытие 80%);
6) профилировать производительность и оптимизировать.
Задачи должны быть измеримыми (например, "обработка 1000 записей за 1с") и привязанными к лабораторным работам 7-12.
Рекомендуемое содержание КП
1. Введение (1-2 стр.): актуальность темы, цель/задачи, обзор аналогичных ИС.
2. Анализ требований (2-3 стр.): модель данных (ER-диаграмма), пользовательские сценарии (use cases), требования к производительности.
3. Проектирование (3-4 стр.): архитектура (MVC), алгоритмы (псевдокод сортировки/поиска), схемы БД, API endpoints.
4. Реализация (4-6 стр.): ключевые фрагменты кода (классы, функции, роуты), скриншоты интерфейса, описание модулей (data/, api/, tests/).
5. Тестирование и оптимизация (2-3 стр.): результаты pytest (таблица тестов), профилирование (cProfile графики), сравнение алгоритмов.
6. Анализ результатов (1-2 стр.): метрики (время отклика, покрытие), рекомендации по масштабированию.
7. Заключение (1 стр.): достижение компетенции, индикаторы (алгоритмы реализованы для ИС).
Приложения: полный исходный код (GitHub-репозиторий), UML-диаграммы, логи тестов.
Оформление результатов
Отчет объемом 20-30 стр. (шрифт Times New Roman 14, интервал 1.5, поля 2 см), титульный лист согласно СТО005-2020 ИРНИТУ (направление "Информационные системы и технологии", дисциплина, ФИО, группа).
Нумерация страниц снизу, оглавление, список литературы (не менее 10 источников: доки Python/Flask, книги по алгоритмам).
Код в моноширинном (monospace font) шрифте (Courier New 10), с номерами строк.
Графики/таблицы с подписями (например, "Таблица 1. Сравнение сортировок").
Репозиторий GitHub с README.md (установка, запуск: pip install -r requirements.txt; flask run), .gitignore, LICENSE.
Итоговый архив: отчет.pdf + src/ + tests/ + db.sqlite.
Правила оформления презентации
Презентация в PowerPoint/Google Slides (10-12 слайдов, шаблон университета):
1. Титульный (тема КП, ФИО).
2. Цели/задачи.
3. Архитектура (блок-схема).
4. Демо API (Postman скриншоты).
5. Ключевой код (2-3 фрагмента).
6. Тесты/производительность (графики).
7. Результаты/метрики.

8. Выводы.
Время: 5 мин доклад + 2 мин демо (живой запуск приложения).
Рекомендации к подготовке защиты
Репетируйте 3-5 раз с таймером, демонстрируйте живой запуск (localhost:5000, тестовые запросы).
Подготовьте ответы на типовые вопросы:
"Обоснуйте выбор алгоритма?",
"Как обеспечили безопасность API?",
"Сложность O(n)?",
"Покрытие тестов?".
Имейте резерв: статические скриншоты, видео демо (если нет сети). После защиты соберите отзывы для доработки.

5.1.2 Методические указания для обучающихся по лабораторным работам:

Лабораторная работа №1. Работа с типами данных и базовыми операциями
Цель работы: Освоить базовые типы данных языка программирования высокого уровня Python и операции над ними для разработки алгоритмов информационных систем.
Теоретическая справка:
Python — интерпретируемый язык высокого уровня с динамической типизацией, где переменные не требуют явного объявления типа. Встроенные типы: int (целочисленные, произвольной точности), float (числа с плавающей запятой, двойная точность IEEE 754), str (строки Unicode, неизменяемые, поддержка f-строк), bool (True/False). Арифметические операции: + сложение, - вычитание, * умножение, / деление (всегда float), // целочисленное деление, % остаток, ** возведение в степень. Операции сравнения возвращают bool: == равенство, != неравенство, , , =, =. Логические: and (конъюнкция), or (дизъюнкция), not (отрицание), короткое замыкание (short-circuit evaluation).
Преобразование типов: int("123"), float(3.14), str(42), bool(0) → False. В ИС типы используются для обработки данных: int/float для метрик, str для логов, bool для флагов. Операторы имеют приоритеты (арифметика сравнение логические), скобки меняют порядок.
Пример: result = (a + b) * c if condition else d.
Ход выполнения работы:
1. Создайте файл lab1_types.py, запустите Python 3.10+.
2. Объявите переменные разных типов: age = 25 (int), price = 19.99 (float), name = "User" (str), active = True (bool).
3. Выполните арифметические операции: вычислите площадь круга (πr^2), остаток от деления, возведите в степень.
4. Реализуйте сравнения: проверьте возраст 18, цену 20, преобразуйте в bool.
5. Создайте логическое выражение для проверки доступа: (age = 18) and active and (price = 50).
6. Используйте f-строки для вывода: f"Пользователь {name}, статус: {active}".
7. Протестируйте преобразования типов, обработайте ошибку ValueError при int("abc").
8. Сохраните вывод в файл results.txt.
Ожидаемый результат:
text
Площадь круга r=5: 78.54
Остаток 17%5=2
Доступ разрешен: True
Пользователь User, статус: True

Контрольные вопросы:
1. В чем преимущество динамической типизации Python?
2. Почему / всегда возвращает float?
3. Что такое короткое замыкание в логических операциях?

Лабораторная работа №2. Алгоритмы ветвления и циклов
Цель работы: Научиться реализовывать алгоритмы управления потоком выполнения для обработки данных в информационных системах.
Теоретическая справка:
Структуры управления: условные операторы if-elif-else для ветвления, циклы for (итерация по последовательностям, range(start, stop, step)), while (условие выполнения). Операторы break (выход из цикла), continue (пропуск итерации), pass (пустая инструкция). Вложенные циклы для матриц/таблиц данных ИС. for item in iterable эффективнее while для известного количества итераций. range(10) — 0..9, enumerate(lst) — индекс+элемент. В ИС: циклы для обработки записей БД, условия для валидации. Тернарный: value = "да" if x 0 else "нет". Пример поиска максимума:
python
max_val = max(lst) if lst else 0 # или цикл
for i, val in enumerate(data):
    if val threshold: process(val)
Ход выполнения работы:
1. Файл lab2_control.py.
2. Реализуйте меню калькулятора: while True, if choice == '1': сложение и т.д., '0' — break.
3. Напишите функцию поиска простого числа (for n in range(2, limit): if all(n%d!=0 for d in range(2,int(n**0.5)+1))).
4. Обработайте список температур: for temp in temps: if temp 30: print("Жара") elif temp 0: print("Мороз").
5. Создайте таблицу умножения 10×10 с вложенными for i in range(1,11): for j in range(1,11).
6. Используйте while для ввода чисел до 0, подсчитайте сумму четных.
7. Добавьте continue для пропуска отрицательных, break при сумме 100.
8. Выведите статистику в файл.
Ожидаемый результат:
text
Таблица умножения:
1×1=1 1×2=2 ... 10×10=100
Сумма четных: 120
Макс. простое 50: 47
Контрольные вопросы:
1. Когда использовать for, а когда while?
2. Разница break и continue?
3. Как enumerate упрощает циклы?

Лабораторная работа №3. Работа со структурами данных Python
Цель работы: Освоить коллекции Python для хранения и манипуляции данными информационных систем.
Теоретическая справка:
Коллекции: list (упорядоченный изменяемый массив, , срезы lst[1:3], методы append/pop/insert/remove/sort/reverse), tuple (неизменяемый, (1,2,3), распаковка a,b,c = tup), set (неупорядоченный уникальный, {1,2,3}, | - ^), dict (ключ-значение, {'key':value},

get('key', default), keys/values/items). List comprehensions: [x*2 for x in lst if x0]. В ИС: list для логов, dict для конфигураций, set для уникальных ID. collections.defaultdict, Counter. Mutable vs immutable: list изменяема, tuple нет.
Пример: users = {'id':1, 'name':'user'}; data = {k:v for k,v in users.items() if v}.
Ход выполнения работы:
1. lab3_collections.py.
2. Создайте list чисел, выполните срезы, sort(), sum().
3. Преобразуйте в set, добавьте дубликаты, операции union/intersection.
4. Создайте dict студентов {'Иванов':95, 'Петров':87}, выведите отсортированный по баллам.
5. Реализуйте comprehension: квадраты четных из [1..20].
6. Обработайте tuple дат, распакуйте min/max/avg.
7. defaultdict(list): группируйте по ключу.
8. Сохраните в JSON.

Ожидаемый результат:
text
Уникальные: {2,4,6,8}
Топ студент: Иванов - 95
Квадраты: [4,16,36,...]
Группы: {'A': [95,87], 'B': [75]}
Контрольные вопросы:
1. Когда использовать list vs tuple?
2. Преимущества dict над list для поиска?
3. Что делают set операции?

Лабораторная работа №4. Алгоритмы обработки коллекций данных
Цель работы: Разработать алгоритмы обработки больших коллекций данных для задач информационных систем.
Теоретическая справка:
Алгоритмы: flatten (списков списков), группировка, фильтрация, агрегация. itertools: chain(*lists), groupby, permutations. Функциональный стиль: map(lambda x:x*2, lst), filter(lambda x:x0, lst), functools.reduce. В ИС: обработка логов, агрегация метрик.
Сложность: O(n) линейная, O(n log n) сортировка.
Пример flatten: [item for sublist in matrix for item in sublist]. Pivot таблицы через dict of lists.
Ход выполнения работы:
1. lab4_algorithms.py.
2. Flatten матрицу 3×3 в list.
3. Группируйте числа по четности (dict even/odd).
4. map/filter/reduce: удвойте 10, суммируйте.
5. itertools.groupby(sorted_data, key=lambda x:x).
6. Создайте топ-N из списка оценок.
7. Pivot: строки — имена, столбцы — предметы.
8. Измерьте время $O(n^2)$ vs $O(n \log n)$.

Ожидаемый результат:
text
Flatten: [1,2,3,4,5,6,7,8,9]
Группы: even=[2,4], odd=[1,3]
Топ-3: [95,92,90]
Время: 0.001s vs 0.05s
Контрольные вопросы:
1. Разница map/filter/reduce?

2. Зачем itertools?
3. Как измерить сложность алгоритма?

Лабораторная работа №5. Разработка модульных программ с функциями (5 часов)

Цель работы: Освоить создание модульных программ с использованием функций для решения задач информационных систем.

Теоретическая справка:

Функции — переиспользуемые блоки кода с параметрами (позиционными, именованными func(name="default"), по умолчанию, *args для переменного числа, **kwargs для ключ-значение) и возвратом return value (или None). Область видимости: LEGB (Local, Enclosing, Global, Built-in), global x, nonlocal x. Лямбда: lambda x: x*2. Декораторы @decorator модифицируют функции. Функциональное программирование: map(func, iterable), filter(pred, iterable), functools.reduce(func, iterable). Генераторы yield для ленивых последовательностей. В ИС функции инкапсулируют логику: process_user_data(users). Рекурсия: факториал def fact(n): return 1 if n=1 else n*fact(n-1). Docstrings для документации. Пример:

python

```
def divide(a, b=2): return a/b
result = map(lambda x: x**2, range(5))
```

Ход выполнения работы:

1. Файл lab5_functions.py.
2. Напишите функции: sum_range(start, end), is_prime(n), factorial(n) (рекурсия).
3. Создайте функцию с *args: average(*numbers), с **kwargs: user_info(**data).
4. Реализуйте лямбда для сортировки списка словарей по ключу.
5. Используйте map/filter/reduce для обработки списка продаж.
6. Создайте генератор fib_gen(n) с yield.
7. Добавьте декоратор @timer для замера времени.
8. Протестируйте и выведите docstrings.

Ожидаемый результат:

text

```
Факториал 5: 120
Среднее: 7.5
Фибоначчи: 0,1,1,2,3,5
Время factorial(100): 0.0001s
```

Контрольные вопросы:

1. Разница *args и **kwargs?
2. Что такое область видимости LEGB?
3. Когда использовать генераторы?

Лабораторная работа №6. Обработка файлов и исключений в ИС (4 часа)

Цель работы: Научиться работать с файлами и обрабатывать исключения для надежных информационных систем.

Теоретическая справка:

Файлы: open('file.txt', 'r/w/a/b', encoding='utf-8'), методы read(), readline(), readlines(), write(), writelines(). Контекст with open() as f: автозакрытие. CSV: csv.reader/writer, JSON: json.load/dump. Исключения: try-except-else-finally, raise ValueError("msg"), иерархия BaseException → Exception → ValueError, IOError. Пользовательские: class ConfigError(Exception): pass. В ИС: логирование logging.error(), валидация входных файлов. pathlib.Path для ОС-независимых путей. Пример:

python

```
try:
   with open('data.json') as f:
```

```
        data = json.load(f)
except FileNotFoundError:
    print("Файл не найден")
finally:
    print("Очистка")
```

Ход выполнения работы:

1. lab6_files.py.
2. Создайте/прочитайте TXT, посчитайте слова.
3. Запишите CSV: имена, баллы; прочитайте с csv.reader.
4. Сохраните/загрузите dict в JSON, обработайте KeyError.
5. Реализуйте логгер ошибок в файл.
6. Создайте пользовательское исключение InvalidDataError.
7. Тестируйте try-except с файлами разными режимами.
8. Используйте pathlib для проверки существования.

Ожидаемый результат:

text

```
Слов: 150
CSV загружено: 10 записей
JSON: {'users': 5}
Ошибка: InvalidDataError logged
```

Контрольные вопросы:

1. Преимущества with для файлов?
2. Как создать пользовательское исключение?
3. Зачем else в try-except?

Лабораторная работа №7. Реализация алгоритмов сортировки и поиска

Цель работы: Разработать и сравнить алгоритмы сортировки/поиска для оптимизации информационных систем.

Теоретическая справка:

Сортировки: пузырьком O(n^2) for i in range(n-1): for j in range(n-i-1): if a[j]a[j+1]: swap, выбором (min индекс), вставками (shift), merge sort O(n log n) рекурсия, quicksort (pivot). Поиск: линейный O(n), бинарный O(log n) на отсортированном. Хэш-поиск O(1). sorted(lst, key=func, reverse=True), bisect. В ИС: сортировка логов, поиск пользователей. Стабильность: merge сохраняет порядок.

Пример quicksort:

python

```
def quicksort(arr):
    if len(arr) = 1: return arr
    pivot = arr[len(arr)//2]
    return quicksort([x for x in arr if x  pivot]) + [pivot] + quicksort([x for x in arr if x  pivot])
```

Ход выполнения работы:

1. lab7_sort_search.py.
2. Реализуйте bubble, selection, insertion sort.
3. Merge sort рекурсивно.
4. Линейный и бинарный поиск.
5. Сгенерируйте 10000 элементов, замерьте время всех.
6. Сравните с sorted().
7. Топ-K с heapq.nlargest.
8. График времени (matplotlib опционально).

Ожидаемый результат:

text

Bubble 1000: 0.5s, Merge: 0.01s
Бинарный поиск: 12 шагов
sorted() самый быстрый
Контрольные вопросы:
1. Сложность O-нотация?
2. Когда quicksort лучше merge?
3. Условия бинарного поиска?

Лабораторная работа №8. Разработка приложений с SQLite
Цель работы: Освоить работу с реляционными базами данных SQLite для хранения данных ИС.
Теоретическая справка:
SQLite — встраиваемая БД, sqlite3.connect('db.sqlite'). CRUD: CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT); INSERT INTO users VALUES(?,?); SELECT * FROM users WHERE id=?; UPDATE/DELETE. Параметризованные запросы :param или ? против SQL-инъекций. Транзакции conn.commit()/rollback(). Агрегаты COUNT, SUM, AVG, GROUP BY, HAVING, JOIN. Индексы CREATE INDEX. В ИС: локальные БД для конфигураций.
Пример:
python
import sqlite3
conn = sqlite3.connect('test.db')
c = conn.cursor()
c.execute('SELECT * FROM users')
print(c.fetchall())
Ход выполнения работы:
1. lab8_sqlite.py.
2. Создайте БД, таблицы users, orders.
3. INSERT 10 пользователей, 20 заказов.
4. SELECT с WHERE, JOIN, GROUP BY.
5. UPDATE баллы, DELETE неактивных.
6. Транзакция: добавьте, commit или rollback.
7. Экспорт в CSV.
8. Визуализация COUNT по группам.
Ожидаемый результат:
text
Users: 10, Orders: 20
AVG order: 1500
JOIN: 15 записей
Контрольные вопросы:
1. Зачем параметризованные запросы?
2. ACID свойства транзакций?
3. JOIN типы?

Лабораторная работа №9. Интеграция с реляционными БД
Цель работы: Подключить Python к PostgreSQL/MySQL и освоить ORM для сложных ИС.
Теоретическая справка:
ORM SQLAlchemy: engine = create_engine('postgresql://user:pass@localhost/db'), модели class User(Base): __tablename__ = 'users'; id = Column(Integer, primary_key=True). Session: session.add(user); session.commit(). Query: session.query(User).filter(User.age 18). Отношения one-to-many relationship('Order'). Миграции Alembic. В ИС: масштабируемые

БД. Raw SQL через text(). Пример: User.query.filter_by(name='Ivan').first() с Flask-SQLAlchemy.
Ход выполнения работы:
1. Установите psycopg2/SQLAlchemy, настройте локальную PostgreSQL.
2. lab9_orm.py: модели User, Product, Order.
3. Создайте таблицы Base.metadata.create_all().
4. CRUD через session: add, get, update, delete.
5. Запросы с join, filter, group_by.
6. Пагинация offset/limit.
7. Тестируйте отношения.
8. Миграция схемы.

Ожидаемый результат:
text
Users 18: 7
Orders by user: {'Ivan': 5}
Pagination page 1: 10 items

Контрольные вопросы:
1. Преимущества ORM над raw SQL?
2. Как работают отношения в SQLAlchemy?
3. Session lifecycle?

Лабораторная работа №10. Создание REST API на Flask
Цель работы: Разработать веб-API для интеграции компонентов информационных систем.
Теоретическая справка:
Flask: from flask import Flask, jsonify, request; app = Flask(__name__); @app.route('/users', methods=['GET','POST']). JSON: jsonify(data), request.json. База: Blueprint, SQLAlchemy. Аутентификация JWT. В ИС: микросервисы. flask run --debug.
Пример:
python
```
@app.route('/api/users/int:id')
def get_user(id):
    return jsonify({'id': id, 'name': 'User'})
```
Ход выполнения работы:
1. lab10_flask.py, pip install flask flask-sqlalchemy.
2. Создайте app, роуты /users GET/POST/PUT/DELETE.
3. Интегрируйте SQLite модель User.
4. Добавьте /stats с агрегацией.
5. Обработайте ошибки 404/400.
6. Тестируйте Postman/curl.
7. Blueprint для /api/v1.
8. Документация OpenAPI (flask-restx опционально).

Ожидаемый результат:
text
GET /users: [{"id":1,"name":"Ivan"}]
POST /users: 201 Created
Stats: {"count":10,"avg":85}

Контрольные вопросы:
1. REST методы и статусы?
2. Зачем Blueprint?
3. JSON безопасность?

Лабораторная работа №11. Параллельное выполнение задач
Цель работы: Реализовать многопоточность и асинхронность для высокопроизводительных ИС.
Теоретическая справка:
Threading: threading.Thread(target=func).start(), Lock with lock:, GIL ограничивает CPU. Multiprocessing: Process, Pool pool.map(func, tasks). Asyncio: async def, await asyncio.sleep(1), asyncio.gather(*tasks). aiohttp для async HTTP. В ИС: параллельная обработка запросов.
Пример:
python
```
import asyncio
async def fetch(url):
    async with aiohttp.ClientSession() as session:
        async with session.get(url) as resp:
            return await resp.text()
```
Ход выполнения работы:
1. lab11_parallel.py.
2. Threading: 10 потоков для вычислений.
3. Multiprocessing Pool.map факториал.
4. Asyncio: 10 корутин sleep+compute.
5. aiohttp: параллельно 5 URL.
6. Сравните время sequential vs parallel.
7. Lock для shared counter.
8. График производительности.

Ожидаемый результат:
text
```
Threads: 2.1s, Async: 1.2s, Seq: 10s
Counter: 1000 (no race)
```
Контрольные вопросы:
1. GIL проблема?
2. Process vs Thread?
3. async/await синтаксис?

Лабораторная работа №12. Автоматизированное тестирование ПО
Цель работы: Освоить юнит-тестирование и отладку для надежного ПО ИС.
Теоретическая справка:
Pytest/unittest: def test_func(): assert func(2)==4. Fixtures @pytest.fixture, parametrize @pytest.mark.parametrize. Mocks unittest.mock.patch. Покрытие pytest-cov. Отладка pdb.set_trace(), logging logger.info(). Профилирование cProfile.run(). В ИС: CI/CD тесты.
Пример:
python
```
import pytest
def test_add():
    assert add(1,2) == 3
```
Ход выполнения работы:
1. lab12_tests.py, pytest.
2. Напишите 5 функций (sort, search и т.д.).
3. Тесты: happy path, edge cases, exceptions.
4. Mocks для файлов/API.
5. Запустите pytest -v --cov.
6. pdb отладка бага.

7. Логирование уровней.
8. cProfile топ функций.

Ожидаемый результат:

text

Tests: 20 passed, 0 failed

Coverage: 95%

Profile: sort 40% time

Контрольные вопросы:

1. TDD принципы?
2. Mock назначение?
3. Покрытие кода метрики?

5.1.3 Методические указания для обучающихся по самостоятельной работе:

Рекомендации по самостоятельной работе:

1. Рекомендации по самостоятельной подготовке к лабораторным работам

- Изучите теоретический материал по теме лабораторной работы. Ознакомьтесь с учебниками, лекциями и дополнительными источниками, чтобы понимать цели и задачи работы, основные понятия и методы, используемые в лабораторном задании1.
- Внимательно ознакомьтесь с методическими указаниями и требованиями к лабораторной работе. Обратите внимание на последовательность выполнения этапов, используемое программное обеспечение, форматы исходных и выходных данных, требования к визуализации и анализу результатов.
- Подготовьте исходные данные. Проверьте наличие всех необходимых файлов, убедитесь в их корректности (форматы, структура, отсутствие ошибок и пропусков данных).
- Освойте необходимые функции и инструменты программного обеспечения. Повторите работу с теми модулями и инструментами, которые будут использоваться в лабораторной работе.
- Планируйте время. Разделите выполнение работы на этапы: подготовка данных, выполнение анализа, оформление визуализации, написание отчета.

2. Рекомендации по оформлению отчетов по лабораторным работам

- Структурируйте отчет по стандартной схеме:
- Титульный лист (название работы, ФИО, группа, дата)
- Цель работы
- Краткое описание исходных данных
- Описание используемых методов и программного обеспечения
- Последовательное изложение этапов работы с иллюстрациями (скриншотами, графиками, картами)
- Анализ полученных результатов (выявленные особенности, сравнение с теорией, интерпретация)
- Выводы и рекомендации
- Список использованных источников
- Используйте качественные иллюстрации. Все графические материалы должны быть четкими, снабжены подписями, масштабами, легендами и пояснениями.
- Формулируйте выводы по существу. Кратко и ясно отражайте основные результаты работы, выявленные закономерности, достоинства и ограничения применяемых методов.
- Оформляйте отчет в соответствии с требованиями ДОТ. Соблюдайте стандарты оформления текста, таблиц, рисунков и ссылок на источники.

3. Рекомендации по самостоятельной проработке отдельных разделов тем

- Изучайте рекомендованную литературу и дополнительные источники. Используйте учебники, статьи, электронные ресурсы, профессиональные базы данных и справочные материалы, указанные в рабочей программе дисциплины1.
- Выполняйте конспектирование ключевых понятий и алгоритмов. Составляйте краткие записи по основным определениям, алгоритмам, этапам работы с ПО, особенностям визуализации и анализа данных.
- Практикуйтесь в самостоятельном выполнении типовых заданий. Решайте задачи, связанные с обработкой и визуализацией геолого-геофизических данных, используя различные программные средства.
- Формулируйте вопросы и уточнения для обсуждения на занятиях. Записывайте непонятные моменты, чтобы получить разъяснения у преподавателя или в ходе дискуссии.
- Анализируйте примеры из практики. Изучайте реальные кейсы решения задач геофизики, сравнивайте разные подходы и делайте выводы о целесообразности их применения.

4. Общие рекомендации

- Развивайте навыки поиска и критического анализа информации. Пользуйтесь современными информационными ресурсами, анализируйте достоверность и актуальность найденных данных.
- Акцентируйте внимание на интеграции знаний и умений. Старайтесь связывать теоретические знания с практическими задачами, анализируйте, как выбранные методы и технологии влияют на качество и достоверность графического представления информации.
- Соблюдайте академическую честность. Все результаты, представленные в отчетах, должны быть получены самостоятельно, с обязательным указанием источников заимствованных данных и иллюстраций.

6 Фонд оценочных средств для контроля текущей успеваемости и проведения промежуточной аттестации по дисциплине

6.1 Оценочные средства для проведения текущего контроля

6.1.1 семестр 3 | Устный опрос

Описание процедуры.

Опрос может проводиться:
Фронтально — в форме беседы с группой, когда вопросы задаются всей группе, а ответы даются по очереди или по желанию.
Индивидуально — каждый студент отвечает на один или несколько вопросов, давая развернутый, связный ответ, часто с примерами и пояснениями.
Комбинированно — сочетаются оба подхода, а также используются дополнительные методы (например, письменные карточки, рецензирование ответов товарищей)

Критерии оценивания.

полнота и правильность ответа;
понимание и осознанность материала;
логичность и последовательность изложения;
корректность терминологии;
способность отвечать на уточняющие вопросы

6.1.2 семестр 4 | Устный опрос

Описание процедуры.

Опрос может проводиться:
Фронтально — в форме беседы с группой, когда вопросы задаются всей группе, а ответы даются по очереди или по желанию.
Индивидуально — каждый студент отвечает на один или несколько вопросов, давая развернутый, связный ответ, часто с примерами и пояснениями.
Комбинированно — сочетаются оба подхода, а также используются дополнительные методы (например, письменные карточки, рецензирование ответов товарищей)

Критерии оценивания.

полнота и правильность ответа;
понимание и осознанность материала;
логичность и последовательность изложения;
корректность терминологии;
способность отвечать на уточняющие вопросы

6.2 Оценочные средства для проведения промежуточной аттестации

6.2.1 Критерии и средства (методы) оценивания индикаторов достижения компетенции в рамках промежуточной аттестации

Индикатор достижения компетенции	Критерии оценивания	Средства (методы) оценивания промежуточной аттестации
ОПК ОС-6.1	полнота и правильность ответа; понимание и осознанность материала; логичность и последовательность изложения; корректность терминологии; способность отвечать на уточняющие вопросы	Устный опрос

6.2.2 Типовые оценочные средства промежуточной аттестации

6.2.2.1 Семестр 4, Типовые оценочные средства для проведения экзамена по дисциплине

6.2.2.1.1 Описание процедуры

Экзамен сдается в период экзаменационной сессии, предусмотренной учебным планом и календарным учебным графиком.
Студенты допускаются к сдаче экзамена по дисциплине при выполнении всех запланированных форм текущего контроля согласно рабочей программе дисциплины.
Примерные вопросы к экзамену:
1. Чем отличаются языки программирования высокого уровня от низкого уровня?

Приведите 3 преимущества Python для информационных систем.
2. Объясните динамическую типизацию. Что произойдет при x = 5; x ="строка"?
3. Какие арифметические операции в Python возвращают float? Покажите пример 17 // 5 и 17 / 5.
4. Напишите логическое выражение для проверки: возраст 18 ИЛИ (студент И активный статус).
5. Что выведет print(bool(""), bool(0), bool([]), bool({}))?
6. Напишите цикл for, выводящий таблицу умножения от 1 до 10 с использованием range(1,11).
7. Разница между break, continue и pass? Приведите пример с while.
8. Реализуйте тернарный оператор для присвоения: если x 0 то "положительное", иначе "неположительное".
9. Что выведет вложенный цикл: for i in range(3): for j in range(2): print(i,j)?
10. Напишите while для ввода чисел до 0 с подсчетом суммы четных.
11. Разница list, tuple, set, dict? Покажите неизменяемость tuple.
12. Напишите list comprehension: квадраты четных чисел от 1 до 20.
13. Операции множеств: union, intersection, difference. Пример для A={1,2,3}, B={2,3,4}.
14. Методы словаря: get(), keys(), items(). Что вернет d.get('key', 0)?
15. Что выведет lst = [1,2,3]; lst[1:]=[4,5]; print(lst)?
16. Параметры *args, **kwargs? Напишите функцию func(1,2, name="test", a=10).
17. Области видимости LEGB. Что изменит global x внутри функции?
18. Режимы открытия файлов 'r','w','a','b'? Зачем with open()?
19. Обработайте исключение: try: int("abc") except ValueError: print("Ошибка").
20. JSON vs CSV: как загрузить/сохранить {"users": [1,2]}?
21. Разница @staticmethod, @classmethod, обычный метод? Пример с self и cls.
22. Магические методы __init__, __str__, __len__. Создайте класс Point.
23. Наследование: базовый Animal, производный Dog(Animal). Вызов super().__init__().
24. Сложность сортировок: bubble O(n^2), merge O(n log n). Реализуйте insertion sort.
25. Бинарный поиск: условия, пример для отсортированного [1,3,5,7].
26. Рекурсия: факториал и глубина стека. Когда хвостовая рекурсия?
27. SQLite CRUD: INSERT, SELECT WHERE, JOIN. Параметризованные запросы ?.
28. SQLAlchemy ORM: Column, relationship, session.query().filter().
29. Flask роуты @app.route('/users/int:id', methods=['GET','POST']). Статусы 200/404.
30. Многопоточность: GIL, threading.Lock(), asyncio.gather(). Разница Process vs Thread.

Рекомендации по подготовке:
Решайте вопросы письменно с примерами кода
Запускайте код в Jupyter/IPython для проверки
Для каждой темы подготовьте 1 лабораторную работу из практики
Ожидаемый формат ответа на экзамене: код + объяснение (5-10 мин/вопрос)

6.2.2.1.2 Критерии оценивания

Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Ответ полный, логичный и структурированный, раскрывает все теоретические вопросы билета.	Ответ в целом полный, но есть незначительные неточности или упущены отдельные детали.	Ответ частичный, раскрывает основные положения, но есть существенные пробелы или	Ответ не раскрывает основные вопросы билета, содержит грубые ошибки или существенные пробелы.

Приведены корректные определения, пояснения, примеры и ссылки на нормативные документы (при необходимости). Практическое задание выполнено полностью, расчеты верны, использованы правильные методы и обоснования. Ответ демонстрирует глубокое понимание материала, самостоятельность мышления и умение применять знания на практике.	Теоретические вопросы раскрыты, приведены основные определения и примеры. Практическое задание выполнено правильно, но возможны несущественные ошибки или недостаточно подробные пояснения. Понимание материала хорошее, умение применять знания продемонстрировано.	ошибки в теории. Некоторые определения отсутствуют или даны неверно, примеры не приведены либо не соответствуют вопросу. Практическое задание выполнено частично, есть ошибки в расчетах или не все этапы решения отражены. Понимание материала поверхностное, самостоятельность ограничена.	Теоретические положения изложены неверно или отсутствуют. Практическое задание не выполнено либо выполнено неправильно, расчеты отсутствуют или неверны. Материал не усвоен, самостоятельность отсутствует.

6.2.2.2 Семестр 4, Типовые оценочные средства для курсовой работы/курсового проектирования по дисциплине

6.2.2.2.1 Описание процедуры

Подготовка к защите (1 неделя до даты):
Преподаватель утверждает допуск по отчету (проверка объема, полноты реализации, репозитория GitHub).
Студент подает: отчет (PDF), презентацию (PPTX, 10-12 слайдов), ссылку на GitHub, демо-видео (3 мин, если нет живого запуска).
Расписание защиты публикуется в LMS/на доске (группа по 3-5 чел./час).
Ход защиты (10 мин/студент):
Доклад (5 мин): презентация с демонстрацией живого приложения (localhost:5000, Postman для API).
Демонстрация (2 мин): запуск ключевых функций, тесты, профилирование в реальном времени.
Вопросы комиссии (3 мин): 3-5 вопросов по коду, алгоритмам, компетенции.
Оценка (протокол): баллы выставляются сразу, устное/письменное замечание.
Технические требования: ноутбук с Python 3.10+, Flask/PostgreSQL, Postman; резерв — скриншоты + видео.
Отказ от защиты = "неудовлетворительно".
Критерии оценки курсового проекта (максимум 100 баллов)

Отчет и материалы (40 баллов)
Критерий Описание Баллы
Полнота отчета Все разделы (введение-заключение), 20-30 стр., ГОСТ 10
Качество кода GitHub с README, requirements.txt, .gitignore, docstrings 10
Архитектура MVC, ER-диаграмма, блок-схемы алгоритмов 10
Документация UML, логи тестов, профилирование 10
Реализация и функциональность (30 баллов)
Критерий Описание Баллы
CRUD API Все операции работают, JSON-ответы 8
Алгоритмы Сортировка/поиск O(n log n), оптимизация 7
БД интеграция ORM/SQLite, транзакции, JOIN 7
Параллелизм Asyncio/Threading, 2x ускорение 8
Тестирование и производительность (20 баллов)
Критерий Описание Баллы
Юнит-тесты Pytest, покрытие 80%, mocks 8
Производительность cProfile, 1с на 1000 записей 6
Безопасность Параметризованные запросы, валидация 6
Защита (10 баллов)
Критерий Описание Баллы
Презентация 10-12 слайдов, четкость, тайминг 3
Демонстрация Живой запуск, без ошибок 3
Ответы на вопросы Связь с компетенцией ИС 4

6.2.2.2.2 Критерии оценивания

Отлично	**Хорошо**	**Удовлетворительно**	**Неудовлетворительно**
(90-100 баллов) Курсовой проект полностью соответствует требованиям компетенции и индикатора достижения. Реализована вся функциональность с превышением базовых требований (например, покрытие тестов >90%, асинхронная обработка с ускорением >3x, развертывание на Heroku/Docker). Код демонстрирует высокий уровень	(75-89 баллов) Проект полностью реализует заявленные цели и задачи, все ключевые функции работают корректно (CRUD API, алгоритмы O(n log n), тесты >80%). Возможны незначительные недочеты: упрощенная обработка ошибок, отсутствие одного вида параллелизма или покрытие 80-85%. Отчет соответствует ГОСТ, GitHub-репозиторий рабочий, защита	(60-74 баллов) Проект реализует базовую функциональность (CRUD + простые алгоритмы), но с существенными ограничениями: тесты 80%, отсутствие оптимизации производительности, ручные SQL-запросы без ORM, синхронная обработка. Отчет формально соответствует требованиям, но с ошибками оформления или неполными разделами. Защита проводится, ответы	(60 баллов, пересдача в течение 2 недель) Критические недостатки: отсутствие ключевых функций (нет API/БД), проект не запускается, тесты отсутствуют или 50%, GitHub пустой или без README. Отчет неполный (15 стр.) или с плагиатом. Защита не проведена или демонстрация невозможна. Компетенция не подтверждена. Пересдача: доработка по замечаниям комиссии + повторная защита (минус 10 баллов от максимума).

архитектуры (SOLID принципы), документация полная (Sphinx API docs), защита безупречная с глубоким анализом альтернативных решений. Демонстрируется способность самостоятельного развития проекта.	уверенная с правильными ответами на вопросы комиссии. Демонстрируется способность разработки практических алгоритмов для ИС.	на вопросы поверхностные. Минимально подтверждает индикатор "способен составлять и реализовывать алгоритмы".	

7 Основная учебная литература

1. Земсков, Ю. В. Основы программирования на языке Python : учебное пособие / Ю. В. Земсков, М. А. Кондрякова, И. В. Ребницкая. — Санкт-Петербург : СПБГУ ГА им. А.А. Новикова, 2024. — 102 с. — ISBN 978-5-907354-92-0. — Текст : электронный // Лань : электронно-библиотечная система.

2. Никитина, Т. П. Программирование. Основы Python для инженеров : учебное пособие для вузов / Т. П. Никитина, Л. В. Королев. — 3-е изд., стер. — Санкт-Петербург : Лань, 2026. — 156 с. — ISBN 978-5-507-51280-5. — Текст : электронный // Лань : электронно-библиотечная система.

3. Программирование на языке высокого уровня : методические указания по выполнению курсовой работы / Иркут. гос. техн. ун-т, 2007. - 27.

4. Аршинский В. Л. Программирование на языке высокого уровня. Основы алгоритмизации и программирования : учебное пособие / В. Л. Аршинский, И. А. Серышева, 2021. - 122.

8 Дополнительная учебная литература и справочная

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня : учеб. для вузов по направлению "Информатика и вычислит. техника" / Т. А. Павловская, 2006. - 460.

2. Крылов. Техника разработки программПрограммирование на языке высокого уровня, 2007. - 374.

3. Егорова Н. Н. Лабораторный практикум по курсу "Программирование на языке высокого уровня" : учебное пособие / Н. Н. Егорова, А. С. Дорофеев, 2003. - 71.

4. Программирование на языке высокого уровня : программа и методические указания по выполнению лабораторных работ (для студентов заочной формы обучения специальности ЭВМ) / Иркут. гос. техн. ун-т, 2011. - 63.

5. Федоров Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для вузов / Д. Ю. Федоров, 2024. - 227.

9 Ресурсы сети Интернет

1. http://library.istu.edu/
2. https://e.lanbook.com/

10 Профессиональные базы данных

1. http://new.fips.ru/
2. http://www1.fips.ru/

11 Перечень информационных технологий, лицензионных и свободно распространяемых специализированных программных средств, информационных справочных систем

1. Лицензионное программное обеспечение Системное программное обеспечение

2. Лицензионное программное обеспечение Пакет прикладных офисных программ

3. Лицензионное программное обеспечение Интернет-браузер

12 Материально-техническое обеспечение дисциплины

1. Учебная аудитория для проведения лекционных занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Оснащение: комплект учебной мебели, рабочее место преподавателя, доска. Мультимедийное оборудование (в том числе переносное): мультимедийный проектор, экран, акустическая система, компьютер с выходом в интернет.

2. Учебная аудитория для проведения лабораторных/практических (семинарских) занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации. Оснащение: комплект учебной мебели, рабочее место преподавателя, доска. Мультимедийное оборудование (в том числе переносное): мультимедийный проектор, экран, акустическая система, компьютер с выходом в интернет.