ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ
ВОЛГОГРАДСКИЙ ГАУ

СВЕДЕНИЯ О СЕРТИФИКАТЕ ЭП
Кому выдана: ФГБОУ ВО "Волгоградский ГАУ"
Сертификат: № 00ef6d90eba6d4a04a920647da5fea52c3
Владелец: Корчагина Ольга Александровна
Действителен: Действителен с 21.10.2024 по 14.01.2026

УТВЕРЖДАЮ

Декан эколого-мелиоративного факультета

_____ О. А. Корчагина

28 мая 2025 г.

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

по основной профессиональной образовательной программе

высшего образования 09.04.03 Прикладная информатика

направленность (профиль) «Управление сопровождением и проектированием создания информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы»

(год начала реализации образовательной программы – 2024)

1. История: учеб. пособие / Т. П. Назарова [и др.]; ФГБОУ ВПО Волгогр. ГАУ. - 2-е изд., испр. и доп. - Волгоград: Волгоградский ГАУ, 2013. - 216 с.

2. Черячукин, В.Г. История: учебное пособие. – 3 изд-е, исправ. и доп. / Черячукин В.Г., Т.П. Назарова, А.Н. Демидова, М.С. Маслов, С.В. Кривинченко, Г.В. Казаков. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2015. – 232 с.

3. Назарова, Т. П. Отечественная история в XIX веке: учеб. пособие / Т. П. Назарова, Н. В. Морозова; ФГБОУ ВО Волгогр. ГАУ. - Волгоград: Изд-во ВолГАУ, 2016. - 96 с.

4. Отечественная история с древнейших времен до конца XVIII века: учеб. Пособие/ Т. Н. Назарова [и др.]; ФГБОУ ВО Волгогр. ГАУ. - Волгоград: Изд-во ВолГАУ, 2016. - 88с.

5. Корабельников, И.С. Экономическая теория: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.03.03 Прикладная информатика / И.С. Корабельников. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

6. Екатериничева, Н.Г. Методические указания для написания контрольных работ по направлению подготовки 09.03.03 «Прикладная информатика», профиль (специализация) «Прикладная информатика в экономике» по дисциплине «Экономическая теория»/ Екатериничева Н.Г. - Волгоград, Волгоградский ГАУ, ИПК «Нива», 2017.- 20 с.

7. Машлыкина, Н.Д. Немецкий язык: базовые коммуникативные компетенции для студентов 1-2 курсов: учеб.-метод. пособие по нем. языку для студ. 1-2 курсов с.-х. вузов всех направлений и специальностей / Н. Д. Машлыкина; под общ. ред. А. В. Олянича; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2015. - 68 с.

8. Гузенко, Е.Ю. Безопасность жизнедеятельности: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.03.03 Прикладная информатика / Е.Ю. Гузенко. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

9. Михайлов, С.Н. Экономическая теория: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.03.03 Прикладная информатика / С.Н. Михайлов. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

10. Руднева, А.А. Право: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / А.А. Руднева. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

11. Маловичко, Р.И. Числовые и функциональные ряды: метод. указания к выполнению контрольной работы / Р. И. Маловичко; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2013. - 56 с.

12. Руководство к решению задач по дискретной математике: учеб.-метод. Пособие для бакалавров напр. подг.: 09.04.03 «Прикладная информатика», 38.03.05 «Бизнесинформатика», 35.03.06 «Агроинженерия», 13.03.02 «Электроэнергетика и электротехника» / сост. А.А. Шубович; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2015. - 88 с.

13. Шубович, А.А. Основы высшей и дискретной математики: справочник / А.А. Шубович, Ю.В. Клочков; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2015. - 68 с.

14. Ищанов, Т.Р. Теория вероятностей и математическая статистика: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Т.Р. Ищанов. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

15. Стрижакова, Е. А. Исследование операций и методы оптимизации для подготовки бакалавров направления «Прикладная информатика»: учеб.-метод. пособие. Ч. 1 / Е. А. Стрижакова, О. А. Заяц; ФГБОУ ВПО Волгогр. ГАУ. -Волгоград: Волгоградский ГАУ, 2013. - 100 с.

16. Стрижакова, Е. А. Исследование операций и методы оптимизации: учеб.-метод. пособие. Ч. 2 / Е. А. Стрижакова, О.А. Заяц; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2013. - 80 с.

17. Заяц, О.А. Исследование операций и методы оптимизации: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / О.А. Заяц. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

18. Ширяева, Е.В. Информационные системы и технологии: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева, Р.И. Пенькова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

19. Марченко, С.С. Е.В. Алгоритмизация и программирование: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / С.С. Марченко. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

20. Операционные системы, среды и оболочки: учеб.-метод. пособие для подготовки бакалавров напр. Прикладная информатика - 230700 / М. П. Васильев, М. В.

Филиппов; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2012. - 148 с.

21. Васильев, М.П. Операционные системы: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.03.03 Прикладная информатика / М.П. Васильев. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

22. Васильев, М.П. Вычислительные системы, сети и телекоммуникации: учеб. пособие для подг. бакалавров направ. «Прикладная информатика» / М. П. Васильев, А. Ю. Руденко; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2012. -192 с.

23. Васильев, М.П. Вычислительные системы, сети и телекоммуникации: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / М.П. Васильев. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

24. Кочеткова, О.В. Теория систем и системный анализ: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / О.В. Кочеткова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

25. Стрижакова, Е.А. Базы данных: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.А. Стрижакова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

26. Шагина, Ю.В. Русский язык и психология деловых коммуникаций: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Ю.В. Шагина – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

27. Матвеев, А. С. Методические рекомендации по выполнению курсового проекта по дисциплине «Проектирование информационных систем» / А. С. Матвеев; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2013. - 28 с.

28. Матвеев, А.С. Проектирование информационных систем: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / А.С. Матвеев. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

29. Серебрякова, М.Ф. Менеджмент: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Серебрякова М.Ф., Даева Т.В. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

30. Васильев, М.П. Информационная безопасность: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / М.П. Васильев. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

31. Марченко, С.С. Программная инженерия: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03

Прикладная информатика / С.С. Марченко. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

32. Матвеев, А. С. Методические рекомендации по выполнению курсового проекта по дисциплине «Проектный практикум» / А. С. Матвеев; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2015. - 28 с.

33. Казантинова, Г. М. Физическая культура студента = Physical education of a student: учебник / Г. М. Казантинова, Т.А. Чарова, Л. Б. Андрющенко; ФГБОУ ВО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2017. - 304 с.

34. Власова, Т. Н. Рабочая тетрадь по физической культуре / Т. Н. Власова, В. А. Кудинова, Е. А. Ряховская; ФГБОУ ВО Волгогр. ГАУ. - Изд. 4-е, перераб. -Волгоград: Волгоградский ГАУ, 2017. - 64 с.

35. Юшкин, В.Н. Основы киберфизических систем: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.03.03 Прикладная информатика / В.Н. Юшкин. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

36. Богданов, Е. П. Аналитическое и имитационное моделирование экономических процессов: учеб.-метод. пособие / Е. П. Богданов; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2013. - 116 с.

37. Ширяева, Е.В. Математическое и имитационное моделирование: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

38. Матвеев, А.С. Основы компьютерной электроники: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / А.С. Матвеев. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

39. Оспищев, Д. А. Методические рекомендации по выполнению курсовой работы по дисциплине "Интернет-программирование" / Д. А. Оспищев; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2014. - 20 с.

40. Марченко, С.С. Интернет-программирование: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / С.С. Марченко. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

41. Арьков, Д.П. Разработка и тестирование программных приложений: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Д.П. Арьков. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

42. Ширяева, Е.В. Системная архитектура информационных систем: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

43. Ширяева, Е.В. Интеллектуальные информационные системы: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева, Р.И. Пенькова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

44. Кочеткова, О.В. Управление информационными системами: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / О.В. Кочеткова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

45. Кочеткова, О.В. Реинжиниринг и управление бизнес-процессами: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / О.В. Кочеткова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

46. Ширяева, Е.В. Информационные системы бухгалтерского учета: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

47. Ширяева, Е.В. Методические указания по выполнению курсовой работы по дисциплине «Информационные системы бухгалтерского учета» для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / Е.В. Ширяева. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

48. Методические указания к лабораторным работам по дисциплине: «Информационные системы бухгалтерского учета» для подготовки бакалавров направления «Прикладная информатика» / А. Ю. Руденко, Л. В. Иванеева; ФГБОУ ВПО Волгогр. ГАУ. - Волгоград: Волгоградский ГАУ, 2012. - 32 с.

49. Кочеткова, О.В. Бизнес-инжиниринг: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / О.В. Кочеткова. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

50. Юшкин, В.Н. Управление требованию: методические рекомендации по изучению дисциплины для обучающихся по направлению подготовки 09.04.03 Прикладная информатика / В.Н. Юшкин. – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024.

ФГБОУ ВО Волгоградский ГАУ, 2025.

Министерство науки и высшего образования Российской Федерации
Департамент координации деятельности организаций
в сфере сельскохозяйственных наук
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Волгоградский государственный аграрный университет»

Кафедра «Информационные системы и технологии»

В.Н. Юшкин
С.С. Марченко
Р.И. Пенькова

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

Волгоград
Волгоградский ГАУ
2024

УДК 004.421(07)
ББК 32.973.26-018.2я7
Ю-313

Рецензент-
кандидат сельскохозяйственных наук, доцент кафедры «Информационные системы и технологии» ФГБОУ ВО Волгоградский ГАУ *О.А. Заяц*

Юшкин Владислав Николаевич
Марченко Сергей Сергеевич
Пенькова Раиса Ивановна

Ю-313 Методические рекомендации по изучению дисциплины «Алгоритмизация и программирование» для обучающихся по направлению подготовки *09.04.03 Прикладная информатика* / С.С. Марченко – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024. - 11 с.

Методические рекомендации содержат цель и задачи изучения дисциплины, раскрывают содержание дисциплины, включают описание оценочных средств текущего контроля, указания для обучающихся по освоению дисциплины, список рекомендуемой для изучения дисциплины литературы.

Предназначены для обучающихся по направлениям подготовки *09.04.03 Прикладная информатика*

УДК 004.421(07)
ББК 32.973.26-018.2я7

Рекомендовано учебно-методической комиссией экономического факультета ФГБОУ ВО Волгоградский ГАУ (протокол № 9 от 21.05.2024 г.)

© ФГБОУ ВО Волгоградский ГАУ, 2024
© Юшкин В.Н., Марченко С.С., Пенькова Р.И., 2024

Содержание

1. HTML — язык разметки веб-страниц
2. CSS — стилизация страниц
3. Bootstrap — фреймворк для адаптивного дизайна
4. PHP — серверный язык программирования
5. Взаимодействие HTML и PHP
6. Формы, GET/POST-запросы
7. Cookies и Sessions
8. Работа с файлами
9. Рекомендуемая литература

Целью изучения дисциплины является:

Целью освоения дисциплины «Интернет программирование» является формирование у студентов профессиональных компетенций, связанных со способностью к разработке приложений для сетей Интернет и выработка умений построения и исследования распределенных приложений, и интерактивных Интернет-страниц.

Изучение дисциплины направлено на решение следующих задач:

овладение навыками веб-программирования,
получение знаний о принципах программирования веб-приложений на языках высокого уровня, о современных системах программирования и тенденциях их развития, о программном обеспечении, а навыков решения задач предметной области

Соотношение планируемых результатов обучения по дисциплине с планируемыми результатами освоения образовательной программы:

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Планируемые результаты обучения по дисциплине
ПК-3. Способен осуществлять организационное и технологическое обеспечение интеграционного тестирования ИС (верификации)	ПК-3.1. Демонстрирует знания методов, подходов и инструментов для проведения интеграционного тестирования информационных систем ПК-3.2. Умеет применять современные средства тестирования при разработке информационных систем в условиях межкомпонентного взаимодействия ПК-3.3. Владеет навыками анализа результатов тестирования и подготовки рекомендаций по устранению выявленных ошибок в составных частях информационной системы	Знать основные принципы построения веб-приложений, обеспечивающих тестируемость: разделение слоёв, использование API, структурирование проекта, понятный интерфейс между компонентами. Уметь создавать интернет-приложения с учётом возможностей интеграционного тестирования: реализация RESTful API, работа с базой данных через ORM, использование сессий и cookies с возможностью эмуляции состояния. Владеть навыками проектирования и реализации интернет-приложений таким образом, чтобы ошибки в интеграции компонентов могли быть легко диагностированы и исправлены: корректная обработка исключений, логирование, документирование интерфейсов.

Основными этапами формирования компетенций в процессе изучении дисциплины является последовательное освоение содержательно связанных между собой тем (разделов) дисциплины.

🌐 *1. HTML — язык разметки веб-страниц*

Пример:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Пример страницы</title>
</head>
<body>
    <h1>Заголовок</h1>
    <p>Это абзац текста.</p>
    <ul>
        <li>Пункт списка 1</li>
        <li>Пункт списка 2</li>
    </ul>
</body>
</html>
```

Пояснение:

Этот пример демонстрирует минимальную структуру HTML-документа.

- *<!DOCTYPE html>* объявляет тип документа и версию HTML.
- *<html>* — корневой элемент, содержит всё содержимое страницы.
- В секции *<head>* находятся метаданные: заголовок, кодировка, подключаемые файлы.
- В секции *<body>* — видимое содержимое сайта: заголовки, параграфы, списки и т.д.

❓ Задания для самопроверки:

10. Напишите HTML-документ с двумя заголовками, одним параграфом и неупорядоченным списком из 3 элементов.
11. Добавьте в HTML-документ тег *<img>*, чтобы отобразить изображение из интернета.
12. Создайте ссылку, которая открывает новую вкладку при клике.
13. Объясните разницу между * * и **.
14. Какой тег используется для вставки видео?

🎨 *2. CSS — стилизация страниц*

Пример:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}
h1 {
```

```
    color: #333;
}
```

Пояснение:

Этот CSS-фрагмент задаёт стили для всей страницы.

- *font-family* определяет шрифт для всего текста.
- *background-color* задаёт цвет фона.
- Цвет заголовка первого уровня (*h1*) изменяется на темно-серый (*#333*).

Задания для самопроверки:

15. Примените стиль к параграфу: увеличьте размер шрифта до 18px и сделайте его курсивным.
16. Создайте класс *.highlight*, который окрашивает фон текста в жёлтый цвет.
17. Используя ID, измените цвет фона у одного конкретного заголовка.
18. Что такое каскадность в CSS? Приведите пример.
19. Как сделать так, чтобы все ссылки на странице были красными и без подчёркивания?

---

3. Bootstrap — фреймворк для адаптивного дизайна

Пример:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boot
strap.min.css" rel="stylesheet">

<div class="container">
  <div class="row">
    <div class="col-md-6">Колонка 1</div>
    <div class="col-md-6">Колонка 2</div>
  </div>
</div>
```

Пояснение:

Подключение Bootstrap происходит через CDN-ссылку.

- Контейнер *.container* центрирует содержимое и ограничивает ширину.
- *.row* создаёт строку сетки.
- *.col-md-6* делит строку на две равные колонки на средних экранах и выше.

Задания для самопроверки:

20. Создайте макет с 3 колонками на Bootstrap.
21. Добавьте кнопку с классом *btn btn-primary*.

22. Реализуйте карточку товара с изображением, заголовком и описанием.
23. Как сделать, чтобы макет был адаптивным под мобильные устройства?
24. Какие классы используются в Bootstrap для скрытия/показа элементов на разных устройствах?

□ *4. PHP — серверный язык программирования*

Пример:

```php
<?php
echo "Привет, мир!";
?>
```

Пояснение:

Это простейший PHP-скрипт, выводящий строку «Привет, мир!» на веб-страницу.

- Все PHP-команды заключаются между тегами *<?php ... ?>*.
- Функция *echo* выводит текст или переменную на экран.

□ Задания для самопроверки:

25. Напишите PHP-скрипт, который выводит текущее время в формате *чч:мм:сс*.
26. Создайте переменную *$name* и выведите приветствие: "Привет, $name!".
27. Напишите условие, которое проверяет, больше ли число 10, и выводит соответствующее сообщение.
28. Организуйте цикл, который выводит числа от 1 до 20.
29. Что такое область видимости переменных в PHP?

□ *5. Взаимодействие HTML и PHP*

Пример:

```php
<?php
$date = date("d.m.Y");
echo "<p>Сегодня: $date</p>";
?>
```

Пояснение:

В этом примере PHP генерирует HTML-код динамически.

- Функция *date()* получает текущую дату.
- Переменная *$date* подставляется в HTML-параграф, который отправляется браузеру.

❓ Задания для самопроверки:

30. Создайте PHP-скрипт, который выводит таблицу умножения от 1 до 10.
31. Сгенерируйте список пользователей из массива и выведите его в виде HTML-списка.
32. Напишите PHP-скрипт, который меняет цвет фона страницы в зависимости от времени суток.
33. Как передать данные из HTML-формы в PHP-скрипт?
34. Что такое *include* и *require* в PHP?

📝 *6. Формы, GET/POST-запросы*

Пример формы:

```
<form action="process.php" method="post">
    Имя: <input type="text" name="name"><br>
    <input type="submit" value="Отправить">
</form>
```

Обработка в PHP:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST["name"];
    echo "Привет, $name!";
}
?>
```

Пояснение:

- Форма отправляет данные на сервер через метод POST.
- В PHP проверяется тип запроса через *$_SERVER["REQUEST_METHOD"]*.
- Данные из формы доступны в массиве *$_POST*, где ключ — это значение атрибута *name* у поля формы.

❓ Задания для самопроверки:

35. Создайте форму регистрации с полями: имя, email, пароль.
36. Напишите обработчик, который проверяет, заполнены ли все поля.
37. Выведите ошибку, если email не прошёл проверку на формат.
38. Как защитить форму от XSS-атак?
39. Чем отличается метод GET от POST?

📝 *7. Cookies и Sessions*

Работа с cookies:

```
setcookie("user", "Иван", time() + 3600);
echo $_COOKIE["user"];
```

Работа с sessions:

```php
session_start();
$_SESSION["login"] = "user123";
echo $_SESSION["login"];
```

Пояснение:

- *`setcookie()`* создаёт cookie с именем *`user`* и значением *`Иван`* на 1 час.
- Через *`$_COOKIE`* можно получить значение cookie.
- *`session_start()`* запускает сессию, а *`$_SESSION`* используется для хранения данных на стороне сервера.

Задания для самопроверки:

40. Создайте cookie, который сохраняет имя пользователя на 1 день.
41. Реализуйте простую систему авторизации с использованием сессий.
42. Как удалить cookie?
43. Почему нельзя использовать сессии без *`session_start()`*?
44. Как защитить сессии от hijacking?

---

8. Работа с файлами

Чтение файла:

```php
$content = file_get_contents("example.txt");
echo $content;
```

Запись в файл:

```php
file_put_contents("example.txt", "Новое содержимое");
```

Загрузка файла:

```php
<form action="upload.php" method="post" enctype="multipart/form-data">
    Выберите файл: <input type="file" name="fileToUpload"><br>
    <input type="submit" value="Загрузить">
</form>

// upload.php
if ($_FILES["fileToUpload"]["error"] == 0) {
    $target = "uploads/" . basename($_FILES["fileToUpload"]["name"]);
    move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target);
    echo "Файл загружен.";
}
```

Пояснение:

- *`file_get_contents()`* считывает содержимое файла.
- *`file_put_contents()`* записывает новые данные в файл.
- Для загрузки файлов необходимо использовать атрибут *`enctype="multipart/form-data"`* в форме.

- Загруженный файл обрабатывается через массив *$_FILES*.

❓ Задания для самопроверки:

45. Создайте файл *data.txt* и запишите в него строку "Hello World".
46. Прочитайте этот файл и выведите его содержимое на странице.
47. Реализуйте форму загрузки изображения и выведите его на странице после загрузки.
48. Как защитить сайт от загрузки опасных файлов?
49. Как очистить содержимое файла средствами PHP?

📚 *Рекомендуемая литература*

50. **MDN Web Docs** — https://developer.mozilla.org/
51. **W3Schools** — https://www.w3schools.com/
52. **PHP Manual** — https://www.php.net/manual/
53. **Bootstrap Documentation** — https://getbootstrap.com/docs/
54. **HTML Academy** — https://htmlacademy.ru/

УЧЕБНОЕ ИЗДАНИЕ

Юшкин Владислав Николаевич
Марченко Сергей Сергеевич
Пенькова Раиса Ивановна

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

В авторской редакции

Компьютерная верстка

Подписано в печать 00.10.2022. Формат $60\times84^{1/16}$.
Усл. печ. л. 0,8. Тираж 50 экз. Заказ 00.
ИПК ФГБОУ ВО Волгоградский ГАУ «Нива».
400002, Волгоград, пр. Университетский, 26.

Министерство науки и высшего образования Российской Федерации
Департамент координации деятельности организаций
в сфере сельскохозяйственных наук
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Волгоградский государственный аграрный университет»

Кафедра «Информационные системы и технологии»

В.Н. Юшкин
С.С. Марченко
Р.И. Пенькова

ТЕХНОЛОГИИ АНАЛИЗА БОЛЬШИХ ДАННЫХ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

Волгоград
Волгоградский ГАУ
2025

УДК 004.421(07)
ББК 32.973.26-018.2я7
Ю-313

Рецензент-
кандидат сельскохозяйственных наук, доцент кафедры «Информационные системы и технологии» ФГБОУ ВО Волгоградский ГАУ *О.А. Заяц*

Юшкин, Владислав Николаевич
Марченко Сергей Сергеевич
Пенькова Раиса Ивановна

Ю-313 Методические рекомендации по изучению дисциплины «Технологии анализа больших данных» для обучающихся по направлению подготовки *09.04.03 Прикладная информатика* / В.Н. Юшкин, С.С. Марченко, Р.И. Пенькова – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2025. - 20 с.

Методические рекомендации содержат цель и задачи изучения дисциплины, раскрывают содержание дисциплины, включают описание оценочных средств текущего контроля, указания для обучающихся по освоению дисциплины, список рекомендуемой для изучения дисциплины литературы.

Предназначены для обучающихся по направлениям подготовки *09.04.03 Прикладная информатика*

УДК 004.421(07)
ББК 32.973.26-018.2я7

Рекомендовано учебно-методической комиссией экономического факультета ФГБОУ ВО Волгоградский ГАУ (протокол № 9 от 21.05.2024 г.)

© ФГБОУ ВО Волгоградский ГАУ, 2024
© Юшкин В.Н., Марченко С.С., Пенькова Р.И., 2024

Содержание

55. Основы анализа данных и инструментарий Python
56. Предобработка данных и исследовательский анализ (EDA)
57. Основы машинного обучения. Обучение без учителя
58. Модели регрессии и классификации
59. Ансамблевые методы и практика построения моделей
60. Рекомендуемая литература

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

Целью изучения дисциплины является:

формирование у студентов представления о задачах, методах и средствах анализа больших данных как деятельности, нацеленной на создание программных продуктов, отвечающих потребностям заказчиков, с соблюдением плановых сроков и бюджета разработки

Изучение дисциплины направлено на решение следующих задач:

- Обработка и хранение больших объемов данных;
- Анализ структурированных и неструктурированных данных;
- Методы машинного обучения и анализа данных;
- Визуализация данных;
- Биг-дата архитектура и инфраструктура.

Соотношение планируемых результатов обучения по дисциплине с планируемыми результатами освоения образовательной программы:

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Планируемые результаты обучения по дисциплине
ПК-2. Способен осуществлять разработку инструментов и методов проектирования бизнес-процессов заказчика в рамках управления работами по сопровождению и проектами создания (модификации) ИС	ПК-2.1. Демонстрирует знание подходов к моделированию и анализу бизнес-процессов	Знать современные методы сбора, хранения и анализа больших данных, применяемых при моделировании бизнес-процессов . Уметь использовать технологии анализа больших данных для выявления закономерностей в деятельности организации и поддержки принятия решений . Владеть навыками работы с инструментами визуализации и интерпретации данных для представления результатов анализа бизнес-процессов
	ПК-2.2. Умеет применять современные инструменты и методы проектирования бизнес-процессов	Знать основные инструменты анализа больших данных (Hadoop, Spark, SQL/NoSQL базы), используемые при автоматизации и оптимизации бизнес-процессов . Уметь применять методы машинного обучения и статистического анализа для прогнозирования поведения бизнес-процессов и оценки эффективности их модификаций . Владеть навыками разработки решений на основе анализа данных, направленных на улучшение процессов управления проектами и сопровождением информационных систем
	ПК-2.3. Владеет навыками формализации требований заказчика к бизнес-процессам и их цифровой трансформации	Знать принципы построения архитектур Big Data и их интеграции в информационные системы заказчика . Уметь переводить неформализованные требования в формальные модели данных и алгоритмы анализа. Владеть навыками документирования и визуализации потоков данных и бизнес-процессов с использованием диаграмм, графиков и отчетов

Основными этапами формирования компетенций в процессе изучении дисциплины является последовательное освоение содержательно связанных между собой тем (разделов) дисциплины.

1. Основы анализа данных и инструментарий Python

Вопрос 1. Введение в анализ данных: задачи, этапы работы, области применения

Анализ данных — это процесс извлечения полезной информации из набора данных.
Этапы:

- Сбор данных
- Подготовка и очистка
- Исследование и визуализация
- Построение модели
- Интерпретация результатов

Области применения: бизнес-аналитика, медицина, финансы, маркетинг, наука.

Вопрос 2. Язык Python как основной инструмент анализа данных

Python — один из самых популярных языков для анализа данных благодаря простоте, гибкости и наличию мощных библиотек: NumPy, pandas, matplotlib, seaborn, scikit-learn, XGBoost и др.

Вопрос 3. Работа с массивами данных с помощью библиотеки NumPy

NumPy — библиотека для работы с многомерными массивами и математических операций над ними.

Пример:
```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([[1, 2], [3, 4]])
print(a + b)
```

Вопрос 4. Обработка табличных данных с помощью pandas

pandas — библиотека для работы с табличными данными. Основные структуры: *`Series`* и *`DataFrame`*.

Пример:
```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head())
print(df.describe())
```

Вопрос 5. Базовые графики с matplotlib и seaborn

matplotlib — базовая библиотека для построения графиков.
seaborn — более высокоуровневый инструмент на основе matplotlib.

Пример:

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.histplot(df['age'], kde=True)
plt.title("Распределение возраста")
plt.show()
```

---

2. Предобработка данных и исследовательский анализ (EDA)

Вопрос 1. Подготовка данных: работа с пропусками и выбросами

Пропуски можно удалить или заполнить средним/медианой. Выбросы находят через boxplot или z-score и обрабатывают аналогично.

Пример:

```
df.fillna(df.mean(), inplace=True)
```

Вопрос 2. Преобразование категориальных признаков

Категориальные данные преобразуются в числовые с помощью One-Hot Encoding или Label Encoding.

Пример:

```
df_encoded = pd.get_dummies(df, columns=['gender'])
```

Вопрос 3. Нормировка и масштабирование данных

Масштабирование помогает алгоритмам работать эффективнее. Используют MinMaxScaler, StandardScaler.

Пример:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[['age', 'income']])
```

Вопрос 4. Описательные статистики и их интерпретация

Описательные статистики включают среднее, медиану, стандартное отклонение, мин/макс.

Пример:

```
print(df.describe())
```

Вопрос 5. Графический анализ распределений и взаимосвязей

Используются:

- Histogram — распределение
- Scatter plot — связь между двумя переменными
- Heatmap корреляций

Пример:
```
sns.heatmap(df.corr(), annot=True)
```

Вопрос 6. Снижение размерности: метод главных компонент (PCA)

PCA используется для уменьшения количества признаков, сохраняя максимум информации.

Пример:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
reduced = pca.fit_transform(scaled_data)
```

---

3. Основы машинного обучения

Вопрос 1. Введение в машинное обучение: типы задач и подходов

ML делится на:

- **Обучение с учителем** — есть целевая переменная
- **Без учителя** — нет меток, только поиск закономерностей
- **С подкреплением** — обучение через награду

Вопрос 2. Подготовка данных для моделирования

Включает:

- Разделение на признаки и целевую переменную
- Разбиение на обучающую и тестовую выборки
- Масштабирование

Пример:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
2)
```

Вопрос 3. Метод кластеризации K-средних

K-средних разделяет объекты на *k* групп на основе расстояний.

Пример:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)
```

Вопрос 4. Иерархическая кластеризация и DBSCAN

- **Иерархическая кластеризация** строит дерево кластеров.
- **DBSCAN** выделяет плотные области и игнорирует шум.

Пример:
```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X)
```

Вопрос 5. Оценка качества кластеризации

Для оценки используют силуэтный коэффициент, индекс Калински-Харабаса.

Пример:
```
from sklearn.metrics import silhouette_score

score = silhouette_score(X, labels)
print(f"Silhouette score: {score:.2f}")
```

---

4. Модели регрессии и классификации

Вопрос 1. Задача регрессии: линейная регрессия

Цель — предсказание числового значения на основе входных признаков.

Пример:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
preds = model.predict(X_test)
```

Вопрос 2. Проблемы мультиколлинеарности и переобучения

Мультиколлинеарность — высокая корреляция между признаками.
Переобучение — модель "запоминает" обучающие данные, но не обобщает.

Вопрос 3. Регуляризация: L1, L2 и ElasticNet

L1 — добавляет штраф за количество признаков.
L2 — штраф за большие коэффициенты.
ElasticNet — комбинация.

Пример:
```
from sklearn.linear_model import Ridge

ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
```

Вопрос 4. Задача классификации: логистическая регрессия

Логистическая регрессия предсказывает вероятность принадлежности к классу.

Пример:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

```
model.fit(X_train, y_train)
print(model.score(X_test, y_test))
```

Вопрос 5. Решающие деревья: принцип работы и построение

Решающее дерево делит данные по условиям, чтобы получить максимальную чистоту классов.

Пример:

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=3)
tree.fit(X_train, y_train)
```

Вопрос 6. Метрики оценки качества моделей

- Для регрессии: MSE, MAE, R^2
- Для классификации: accuracy, precision, recall, F1, ROC-AUC

Пример:

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

---

5. Ансамблевые методы и практика построения моделей

Вопрос 1. Идея ансамблевых методов: голосование и агрегирование

Ансамблевые методы объединяют несколько слабых моделей для повышения точности.

Вопрос 2. Метод случайного леса: устройство и преимущества

Случайный лес — ансамбль деревьев решений, работает лучше, чем одно дерево.

Пример:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
```

Вопрос 3. Градиентный бустинг: XGBoost, LightGBM, CatBoost

Градиентный бустинг последовательно исправляет ошибки предыдущих моделей.

Пример (LightGBM):

```
import lightgbm as lgb

train_data = lgb.Dataset(X_train, label=y_train)
params = {'objective': 'binary'}
model = lgb.train(params, train_data)
```

Вопрос 4. Перекрёстная проверка и подбор гиперпараметров

Cross-validation — оценка модели на разных частях данных. GridSearchCV — автоматический подбор параметров.

Пример:

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(rf, X, y, cv=5)
print(f"Средняя точность: {scores.mean():.2f}")
```

Вопрос 5. Полный цикл построения модели

Шаги:

61. Загрузка и подготовка данных
62. Разбиение на выборки
63. Обучение модели
64. Оценка
65. Тюнинг и финальная модель

Пример:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', RandomForestClassifier())
])

pipe.fit(X_train, y_train)
print(pipe.score(X_test, y_test))
```

---

6. Рекомендуемая литература

Основная литература:

1. Рашка, С. Python и машинное обучение / Себастьян Рашка. — М.: ДМК Пресс, 2020. — 424 с.
   ISBN 978-5-97060-731-1

2. Грась, М. Анализ данных на Python: эффективное получение, обработка и обогащение данных / Михаил Грась. — СПб.: Питер, 2021. — 352 с.
   ISBN 978-5-4461-1338-7

3. Мюллер, А. Введение в машинное обучение с использованием Python / Андреас Мюллер, Сара Гвидо. — М.: О’Рейли, 2017. — 392 с.
   ISBN 978-5-907114-03-7

4. Хастинг, Т. Научная работа с данными: анализ данных и машинное обучение на Python / Тейлор Арнольд, Лаура Хастингс, Марк Хансен. — М.: ДМК Пресс, 2021. — 300 с. ISBN 978-5-97060-882-0

5. Бродул, В. Машинное обучение для начинающих: от теории к практике / Владимир Бродул. — СПб.: БХВ-Петербург, 2022. — 288 с. ISBN 978-5-907556-09-2

6. Каллан, Р. Глубокое обучение. Полный курс / Ричард Каллан. — М.: Вильямс, 2020. — 736 с. ISBN 978-5-8459-2097-3

7. Маккинни, У. Python и анализ данных / Уэс Маккинни. — М.: Вильямс, 2019. — 512 с. ISBN 978-5-8459-2344-8

Дополнительная литература:

8. Чоллет, Ф. Глубокое обучение на Python / Фрэнсуа Чоллет. — СПб.: Питер, 2019. — 368 с. ISBN 978-5-4461-1044-2

9. Аллен, Э. Машинное обучение на практике / Эрик Аллен. — М.: ДМК Пресс, 2021. — 256 с. ISBN 978-5-97060-911-7

10. Вадасз, Д. Искусственный интеллект для веба / Даниэль Вадасз. — СПб.: БХВ-Петербург, 2022. — 320 с. ISBN 978-5-907556-32-0

УЧЕБНОЕ ИЗДАНИЕ

Юшкин Владислав Николаевич
Марченко Сергей Сергеевич
Пенькова Раиса Ивановна

АНАЛИЗ ДАННЫХ И МАШИННОЕ ОБУЧЕНИЕ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

В авторской редакции

Компьютерная верстка

Подписано в печать 00.10.2022. Формат $60\times84^{1/16}$.
Усл. печ. л. 0,75. Тираж 50 экз. Заказ 00.
ИПК ФГБОУ ВО Волгоградский ГАУ «Нива».
400002, Волгоград, пр. Университетский, 26.

Министерство науки и высшего образования Российской Федерации
Департамент координации деятельности организаций
в сфере сельскохозяйственных наук
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Волгоградский государственный аграрный университет»

Кафедра «Информационные системы и технологии»

В.Н. Юшкин
С.С. Марченко
Р.И. Пенькова

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

Волгоград
Волгоградский ГАУ
2024

УДК 004.421(07)
ББК 32.973.26-018.2я7
Ю-313

Рецензент-
кандидат сельскохозяйственных наук, доцент кафедры «Информационные системы и технологии» ФГБОУ ВО Волгоградский ГАУ *О.А. Заяц*

Марченко, Сергей Сергеевич

Ю-313 Методические рекомендации по изучению дисциплины «Алгоритмизация и программирование» для обучающихся по направлению подготовки *09.04.03 Прикладная информатика* / С.С. Марченко – Волгоград: ФГБОУ ВО Волгоградский ГАУ, 2024. - 20 с.

Методические рекомендации содержат цель и задачи изучения дисциплины, раскрывают содержание дисциплины, включают описание оценочных средств текущего контроля, указания для обучающихся по освоению дисциплины, список рекомендуемой для изучения дисциплины литературы.

Предназначены для обучающихся по направлениям подготовки *09.04.03 Прикладная информатика*

УДК 004.421(07)
ББК 32.973.26-018.2я7

Рекомендовано учебно-методической комиссией экономического факультета ФГБОУ ВО Волгоградский ГАУ (протокол № 9 от 21.05.2024 г.)

© ФГБОУ ВО Волгоградский ГАУ, 2024
© Юшкин В.Н., Марченко С.С., Пенькова Р.И., 2024

Содержание

66. Алгоритмы, алгоритмизация и языки программирования
67. Основы программирования на языке высокого уровня
68. Структурное программирование
69. Объектно-ориентированное программирование (ООП)
70. Рекомендуемая литература

Целью изучения дисциплины является:

сформировать компетенции для обеспечения обучающимся способности разрабатывать алгоритмы и программы, пригодные для практического применения

Изучение дисциплины направлено на решение следующих задач:

- Сформировать представление об основных понятиях теории алгоритмов и методах алгоритмизации для решения прикладных задач.
- Познакомить с основными формами алгоритмического представления и программной обработки данных в различных прикладных инфокоммуникационных системах и интегрированных средах промышленного назначения.
- Овладеть современными высокоуровневыми программными средами и инструментами разработки для программного решения прикладных задач различной степени сложности.
- Выработать навыки углубленного стилевого программирования, квалиметрии и планирования процесса коллективной разработки, применения промышленных отраслевых стандартов для поддержки инжинирингового процесса в сложных информационных системах.

Соотношение планируемых результатов обучения по дисциплине с планируемыми результатами освоения образовательной программы:

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Планируемые результаты обучения по дисциплине
ОПК-7. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	ОПК-7.1. Демонстрирует знания методов алгоритмизации задач и принципов написания программ, пригодных для практического применения	Знать основные методы алгоритмизации, типы алгоритмов, структуры данных и принципы программирования, необходимые для разработки программного обеспечения, пригодного к практическому применению.
	ОПК-7.2. Умеет применять на практике методы разработки алгоритмов и программирования для решения конкретных задач в рамках учебных и прикладных проектов	Уметь разрабатывать алгоритмы и писать программы на языках программирования высокого уровня для решения задач из предметной области, включая анализ входных данных, обработку информации и вывод результатов.
	ОПК-7.3. Владеет навыками реализации, тестирования и отладки алгоритмов и программ, ориентированных на практическое использование	Владеть навыками проектирования, реализации, тестирования и документирования программных решений, удовлетворяющих требованиям читаемости, эффективности и функциональности.

Основными этапами формирования компетенций в процессе изучении дисциплины является последовательное освоение содержательно связанных между собой тем (разделов) дисциплины.

Алгоритмы, алгоритмизация и языки программирования

Что такое алгоритм?

Алгоритм — это точное описание последовательности действий, необходимых для решения определённой задачи за конечное число шагов.

Примеры:

- Алгоритм приготовления кофе
- Инструкция по сборке мебели
- Последовательность действий для входа в аккаунт

 В программировании алгоритм — это основа любой программы.

Свойства алгоритма

Свойство	Описание
Дискретность	Алгоритм состоит из отдельных шагов (команд).
Определённость	Все команды должны быть однозначными.
Результативность	После выполнения всех шагов должен быть получен результат.
Конечность	Алгоритм должен завершаться за конечное число шагов.
Массовость	Должен работать при разных допустимых исходных данных.

Как описать алгоритм?

1. Словесное описание

Например:

> Чтобы найти максимальное число из двух: сравните числа, если первое больше второго — вернуть первое, иначе — второе.

2. Псевдокод

```
если a > b:
    макс = a
иначе:
    макс = b
```

3. Блок-схема

Используются стандартные фигуры:

Фигура	Назначение
Овал	Начало или конец
Прямоугольник	Действие
Ромб	Условие
Параллелограмм	Ввод/вывод данных

 Полезные инструменты: draw.io, Lucidchart, Canva.

4. Программа на языке программирования

```
a = int(input("Введите первое число: "))
b = int(input("Введите второе число: "))
print(max(a, b))
```

Типы алгоритмов

Тип	Характеристика	Пример
Линейный	Последовательное выполнение действий	Вычисление суммы двух чисел
Разветвлённый	Есть условия (`if`, `else`)	Проверка чётности числа
Циклический	Используются циклы (`for`, `while`)	Подсчёт факториала
Рекурсивный	Функция вызывает сама себя	Вычисление факториала через рекурсию

Языки программирования

Классификация:

- **Машинные коды** — нули и единицы
- **Ассемблер** — символическое представление машинных команд
- **Языки высокого уровня** — близки к человеческому языку (Python, Java, C++, JavaScript)

Интерпретация vs Компиляция

Тип	Особенность
Компиляция	Исходный код преобразуется в машинный заранее (например, C++)
Интерпретация	Код выполняется построчно во время работы программы (например, Python)

 Для обучения лучше использовать **Python**: простой синтаксис, богатые возможности, большое сообщество.

Этапы решения задач на ЭВМ

71. **Постановка задачи** — понять, что требуется.
72. **Анализ данных** — определить, какие данные нужны на входе и выходе.
73. **Алгоритмизация** — составить план решения.
74. **Программирование** — написать программу.
75. **Тестирование** — проверить работу программы на разных данных.
76. **Отладка** — исправить ошибки.
77. **Сопровождение** — дорабатывать и обновлять программу.

✍Не пренебрегайте этапом тестирования!

Что такое сложность алгоритма?

Сложность показывает, как быстро работает алгоритм и сколько ресурсов он использует.

Временная сложность:

- Измеряется в количестве операций
- Обозначается как **O(f(n))**

Примеры:

Сложность	Пример
O(1)	Получение элемента по индексу
O(n)	Линейный поиск
O(log n)	Бинарный поиск
O(n²)	Вложенные циклы (например, сортировка пузырьком)

 Чем меньше сложность — тем эффективнее алгоритм.

Практические задания

Задание 1. Блок-схемы

Постройте блок-схемы для следующих задач:

78. Найти площадь треугольника по трём сторонам (через формулу Герона).
79. Определить, является ли число чётным.
80. Найти большее из двух чисел.

Задание 2. Реализация на Python

Реализуйте следующие задачи:

81. Программа "Калькулятор" — принимает два числа и операцию (+, -, *, /), выводит результат.
82. Проверка деления на ноль.
83. Вычисление среднего арифметического трёх чисел.

Задание 3. Анализ алгоритма

Выберите один из своих алгоритмов и оцените его временную сложность.

Основы программирования на языке высокого уровня

Базовые конструкции языка программирования

Программа состоит из последовательности инструкций. Вот основные элементы:

- **Переменные** — хранят данные
- **Выражения** — вычисляют значения
- **Операторы** — управляют логикой программы
- **Функции** — повторно используемые блоки кода

- **Комментарии** — поясняют код

```
# Это комментарий
x = 5               # Переменная
result = x + 10     # Выражение
print(result)       # Оператор вывода
```

Переменные и константы

Переменные:

Хранят изменяемые данные. Именуются по определённым правилам.

```
name = "Иван"
age = 25
```

Константы:

Неизменяемые значения. В Python принято писать их заглавными буквами.

```
PI = 3.14159
MAX_ATTEMPTS = 3
```

 Python — язык с динамической типизацией: тип переменной определяется автоматически.

Типы данных

Тип	Пример
int	Целое число: *5*, *-3*, *1000*
float	Дробное число: *3.14*, *-0.5*
str	Строка: *"Hello"*, *'Мир'*
bool	Логическое значение: *True*, *False*
list	Список (массив): *[1, 2, 3]*
tuple	Неизменяемый список: *(1, 2, 3)*
dict	Словарь (ключ-значение): *{'name': 'Ivan', 'age': 25}*
set	Множество: *{1, 2, 3}*

 Чтобы узнать тип переменной: *type(x)*

Операторы

Арифметические:

```
+   -   *   /   //   %   **
```

Сравнения:

```
==   !=   >   <   >=   <=
```

Логические:

```
and   or   not
```

Присваивания:

```
=    +=    -=    *=    /=    //=
```

Условные конструкции

Условия позволяют выполнять разные действия в зависимости от входных данных.

```
if age >= 18:
    print("Доступ разрешён")
else:
    print("Доступ запрещён")
```

Можно добавить несколько условий:

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
```

Циклы

`for` — *перебор значений*

```
for i in range(5):
    print(i)
```

`while` — *выполняется, пока условие истинно*

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Чтобы выйти из цикла досрочно: *`break`*
Чтобы пропустить итерацию: *`continue`*

Функции и процедуры

Функция — это именованный блок кода, который можно вызвать из любого места программы.

```
def greet(name):
    print(f"Привет, {name}!")

greet("Анна")
```

Функции могут возвращать значение:

```
def add(a, b):
    return a + b
```

```
result = add(3, 5)
```

Работа с массивами и строками

Списки (массивы):

```
numbers = [1, 2, 3]
names = ["Анна", "Иван", "Света"]

# Добавление
numbers.append(4)

# Обращение по индексу
print(names[1])
```

Строки:

```
text = "Привет"
print(len(text))        # Длина строки
print(text.upper())     # Преобразование к верхнему регистру
print(text.split())     # Разделение строки
```

Обработка исключений

Используется для предотвращения аварийного завершения программы при ошибках.

```
try:
    x = int(input("Введите число: "))
    print(100 / x)
except ZeroDivisionError:
    print("Нельзя делить на ноль!")
except ValueError:
    print("Нужно ввести число!")
```

Ввод-вывод данных

Ввод:

```
name = input("Введите имя: ")
age = int(input("Введите возраст: "))
```

Вывод:

```
print("Здравствуйте,", name)
print(f"Ваш возраст: {age}")
```

Работа с файлами:

```
# Запись в файл
with open("data.txt", "w") as f:
    f.write("Привет, мир!")

# Чтение из файла
with open("data.txt", "r") as f:
```

```
content = f.read()
print(content)
```

Практические задания

Задание 1. Программы с условиями

84. Напишите программу, которая проверяет, является ли число положительным, отрицательным или нулём.

85. Создайте программу, определяющую, является ли год високосным.

Задание 2. Программы с циклами

86. Напишите программу, которая выводит все чётные числа от 1 до 100.

87. Реализуйте таблицу умножения (например, 10×10).

Задание 3. Функции

88. Напишите функцию, которая принимает радиус круга и возвращает его площадь.

89. Создайте функцию `factorial(n)`, вычисляющую факториал числа.

Задание 4. Работа со списками и строками

90. Напишите программу, которая считывает список чисел и выводит их сумму.

91. Создайте программу, которая принимает строку и выводит её в обратном порядке.

Задание 5. Обработка ошибок

92. Напишите программу, которая считывает два числа и делит первое на второе, корректно обрабатывая возможные ошибки.

Структурное программирование

Что такое структурное программирование?

Структурное программирование — это методология разработки программ, при которой программа строится из ограниченного набора управляющих конструкций:

- **следование** (последовательное выполнение),
- **ветвление** (`if`, `else`),
- **цикл** (`for`, `while`).

Кроме того, программа делится на отдельные части — **функции**, которые решают конкретные подзадачи.

 Этот подход позволяет писать **читаемый**, **тестируемый** и **повторно используемый** код.

Принципы структурного программирования

Принцип	Описание
Декомпозиция задачи	Разбиение большой задачи на более мелкие, понятные части.

Принцип	Описание
Модульность	Выделение отдельных частей программы в автономные модули или функции.
Иерархичность	Построение программы «сверху вниз»: сначала общая структура, затем детали.
Единственный вход и выход у функции	Каждая функция должна иметь один вход и один выход (в идеале).
Ограниченное использование переходов (goto)	Отказ от произвольных переходов в коде для повышения читаемости.

 Эти принципы помогают создавать программы, которые легче читать, тестировать и поддерживать.

Модульность и декомпозиция задач

Модульность

Программа состоит из отдельных **модулей** — файлов или функций, решающих определённую задачу.

Например:

```
# math_utils.py — модуль математических функций
def add(a, b):
    return a + b

def multiply(a, b):
    return a * b
```

В основном файле:

```
import math_utils

result = math_utils.add(5, 3)
print(result)
```

Декомпозиция

Это процесс разбиения задачи на подзадачи. Например, если нужно написать программу «Телефонный справочник», можно выделить такие функции:

- *add_contact()*
- *search_contact()*
- *delete_contact()*
- *list_contacts()*

Структура программы

Общая структура программы может выглядеть так:

```
main.py                 # Точка входа в программу
functions.py            # Функции приложения
data/contacts.txt       # Файл данных
README.md               # Описание проекта
requirements.txt        # Зависимости (при необходимости)
```

Внутри *main.py* обычно:

- Импортируются библиотеки
- Подключаются модули
- Выводится меню
- Обрабатывается выбор пользователя
- Вызываются нужные функции

Преимущества и ограничения структурного подхода

Преимущества	Ограничения
Улучшает читаемость кода	Может быть неудобно при работе с большими системами
Упрощает отладку и тестирование	Нет встроенной поддержки повторного использования данных
Позволяет использовать "сверху вниз" проектирование	Не всегда удобно моделировать реальные объекты
Повышает качество документации	Требует строгой организации кода
Облегчает совместную работу	

 Это хороший первый шаг в освоении программирования, но далее рекомендуется переходить к ООП.

Практические задания

Задание 1. Реализация простых функций

93. Напишите функцию *is_even(n)*, которая возвращает *True*, если число чётное.
94. Создайте функцию *max_of_three(a, b, c)*, возвращающую наибольшее из трёх чисел.

Задание 2. Проект "Телефонный справочник"

Создайте консольное приложение с возможностями:

- Добавление контакта
- Поиск контакта по имени
- Удаление контакта
- Вывод всех контактов

 Сохраняйте данные в текстовый файл.

Задание 3. Модульность

Разделите проект на несколько файлов:

- *main.py* — точка входа
- *contacts.py* — функции работы с контактами
- *utils.py* — вспомогательные функции

Задание 4. Обработка ошибок

Добавьте обработку ошибок:

- При попытке удалить несуществующий контакт
- При некорректном вводе данных

Объектно-ориентированное программирование (ООП)

Что такое ООП?

ООП (Object-Oriented Programming) — это стиль программирования, при котором программа строится вокруг **объектов**, представляющих собой экземпляры **классов**.

Пример:

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def drive(self):
        print(f"{self.brand} {self.model} поехал")

my_car = Car("Toyota", "Corolla")
my_car.drive()
```

 ООП позволяет моделировать реальные объекты: машины, пользователи, заказы, товары и т.д.

Основные принципы ООП

Принцип	Описание
Инкапсуляция	Скрытие деталей реализации объекта, предоставление только необходимого интерфейса
Наследование	Возможность создания новых классов на основе существующих
Полиморфизм	Объекты разных классов могут иметь методы с одинаковым названием, но разным поведением
Абстракция	Сосредоточение на важных характеристиках объекта, игнорируя лишние детали

 Эти четыре принципа являются основой ООП и часто называются **"четыре столпа ООП"**.

Классы и объекты

Класс

Это шаблон или чертёж, описывающий свойства и поведение объекта.

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print("Гав!")
```

Объект

Это конкретный экземпляр класса.

```
my_dog = Dog("Барон", 3)
print(my_dog.name)
my_dog.bark()
```

Наследование

Наследование позволяет одному классу (потомку) наследовать поля и методы другого (родителя).

```
class Animal:
    def eat(self):
        print("Животное ест")

class Cat(Animal):
    def meow(self):
        print("Мяу!")

my_cat = Cat()
my_cat.eat()   # Унаследовано от Animal
my_cat.meow()
```

Полиморфизм

Полиморфизм позволяет использовать один и тот же интерфейс для работы с разными типами данных.

```
class Bird:
    def fly(self):
        print("Птица летит")

class Airplane:
    def fly(self):
```

```
        print("Самолёт взлетает")

def let_fly(obj):
    obj.fly()

let_fly(Bird())       # Вывод: Птица летит
let_fly(Airplane())   # Вывод: Самолёт взлетает
```

Инкапсуляция и абстракция

Инкапсуляция

Сокрытие внутренней реализации объекта. В Python инкапсуляция не строгая, но можно указывать приватные атрибуты:

```
class BankAccount:
    def __init__(self):
        self.__balance = 0  # Приватный атрибут

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount

    def get_balance(self):
        return self.__balance
```

Абстракция

Создание упрощённой модели сложного объекта.

```
class CoffeeMachine:
    def make_coffee(self):
        """Пользователь не видит, как работает кофеварка внутри"""
        self.__heat_water()
        self.__grind_beans()
        print("Кофе готов")

    def __heat_water(self): ...
    def __grind_beans(self): ...
```

Композиция против наследования

Подход	Когда использовать
Наследование	Когда есть четкая иерархия типов ("является")
Композиция	Когда объект состоит из других объектов ("имеет")

Пример композиции:

```
class Engine:
    def start(self):
        print("Двигатель запущен")
```

```
class Car:
    def __init__(self):
        self.engine = Engine()

    def start(self):
        self.engine.start()
        print("Машина готова к поездке")
```

Паттерны проектирования (базовый уровень)

Паттерн (шаблон проектирования) — проверенное решение типовой задачи проектирования.

Примеры популярных паттернов:

Паттерн	Назначение
Singleton	Гарантирует, что у класса будет только один экземпляр
Factory	Создаёт объекты без указания конкретного класса
Observer	Реализует механизм подписки на события
Strategy	Позволяет изменять алгоритмы во время выполнения

 На данном этапе достаточно знать об их существовании и уметь применять простые примеры.

Работа с базами данных через ООП

Можно связать классы с таблицами в БД. Например, используя SQLite:

```
import sqlite3

class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def save(self):
        conn = sqlite3.connect('users.db')
        c = conn.cursor()
        c.execute("INSERT INTO users VALUES (?, ?)", (self.name, self.
email))
        conn.commit()
        conn.close()
```

 Для более удобной работы с БД в Python используют ORM, например, **SQLAlchemy** или **Peewee**.

Документирование и организация проекта в ООП-стиле

Документирование

Каждый класс и метод должен иметь docstring:

```python
class Rectangle:
    """Класс для представления прямоугольника."""

    def __init__(self, width, height):
        """Инициализирует прямоугольник с заданными сторонами."""
        self.width = width
        self.height = height
```

Организация проекта

Структура может выглядеть так:

```
project/
│
├── main.py                # Точка входа
├── models/                # Классы предметной области
│   └── product.py
├── services/              # Логика взаимодействия между объектами
│   └── order_service.py
├── utils/                 # Вспомогательные классы
│   └── logger.py
└── database.py            # Работа с БД
```

Практические задания

Задание 1. Создание классов

95. Создайте класс *Person* с атрибутами *name*, *age*.
96. Добавьте метод *get_info()*, выводящий информацию о человеке.
97. Создайте производный класс *Student*, добавив атрибут *student_id*.

Задание 2. Проект "Система учета товаров"

Реализуйте систему, которая:

- Может добавлять товары (название, цена, количество)
- Отображать список товаров
- Удалять товары
- Сохранять данные в SQLite

 Используйте классы и работу с БД.

Задание 3. Банковская система

Создайте классы:

- *BankAccount* — базовый класс с балансом и методами пополнения и снятия
- *SavingsAccount* — наследуется от *BankAccount*, добавляет начисление процентов
- *CheckingAccount* — имеет комиссию за снятие

Задание 4. CRUD с использованием классов

Создайте класс `User`, реализующий операции:

- Create (добавление в БД)
- Read (чтение из БД)
- Update (обновление данных)
- Delete (удаление пользователя)

Рекомендуемая литература

98. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. *"Алгоритмы: построение и анализ"*
99. Ахо А., Хопкрофт Дж., Ульман Дж. *"Структуры данных и алгоритмы"*
100. Лутц М. *"Изучаем Python"*
101. Бизли Д. *"Python. Подробный справочник"*
102. Свейгарт А. *"Автоматизация рутинных задач с помощью Python"*
103. Йенсен К., Вирт Н. *"Паскаль. Руководство для пользователя и описание языка"*
104. Мартин Роберт *"Чистый код"*
105. Эккель Брюс *"Философия Java"*
106. Фленаган Д., Матсумото Я. *"Ruby on Rails. Шаблоны проектирования"*

УЧЕБНОЕ ИЗДАНИЕ

Юшкин Владислав Николаевич
Марченко Сергей Сергеевич
Пенькова Раиса Ивановна

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Методические рекомендации
по изучению дисциплины
для обучающихся по направлению подготовки
09.04.03 Прикладная информатика

В авторской редакции

Компьютерная верстка

Подписано в печать 00.10.2022. Формат $60\times84^{1/16}$.
Усл. печ. л. 1,25. Тираж 50 экз. Заказ 00.
ИПК ФГБОУ ВО Волгоградский ГАУ «Нива».
400002, Волгоград, пр. Университетский, 26.