Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт математики, информационных технологий и физики
Кафедра вычислительных технологий
и интеллектуальных систем больших данных

М. М. Аббаси, А. П. Бельтюков

ОСНОВЫ КОМПЬЮТЕРНЫХ НАУК И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебно-методическое пособие



Ижевск
2026

УДК 004.2(075.8)
ББК 32.971.3я73
А137

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензенты: д-р техн. наук, профессор, зав. каф. «Автоматизированных систем управления» Уфимского университета науки и технологий В. В. Антонов,
канд. тех. наук, доцент, каф. вычислительных технологий и интеллектуальных систем больших данных Удмуртского государственного университета С. Г. Маслов.

Аббаси М. М., Бельтюков А. П.

А137 Основы компьютерных наук и вычислительной техники : учеб-метод. пособие / М. М. Аббаси, А. П. Бельтюков. – Ижевск: Удмуртский университет, 2026. – 83 с.

ISBN 978-5-4312-1376-2

Учебно-методическое пособие направлено на углублённое освоение дисциплины «Основы компьютерных наук и вычислительной техники». Пособие состоит из пять глав. Каждая глава посвящена базовым механизмам, моделям и компонентам, необходимым для проектирования компьютерной системы. Учебно-методические пособие предназначено для студентов, обучающихся по направлению «Прикладная информатика». Пособие также может быть использовано студентами других смежных направлений, изучающих принципы управления проектами и разработку нового компьютерного оборудования.

УДК 004.2(075.8)
ББК 32.971.3я73

ISBN 978-5-4312-1376-2

© Аббаси М. М., Бельтюков А. П., 2026
© ФГБОУ ВО «Удмуртский государственный университет», 2026

ВВЕДЕНИЕ

Компьютеры – это одна из ключевых частей нашей повседневной жизни, начиная от машин, которые мы используем для работы, заканчивая смартфонами и умными часами, на которые мы полагаемся.

Вычислительная техника – фундаментальная дисциплина, в основе которой лежит систематическое изучение алгоритмических процессов, представления данных, а также разработка аппаратного и программного обеспечения. По своей сути вычислительная техника – это электронное устройство, которое принимает исходные данные, обрабатывает их, превращая в полезную информацию, и сохраняет для дальнейшего использования. В этой области применяются ключевые принципы, в том числе алгоритмическое мышление, модульность и абстракция.

В основе функционирования компьютеров лежит архитектура фон Неймана, которая описывает последовательность операций между вводом, обработкой, хранением и выводом данных. Кроме того, в основе современных вычислительных систем лежат цифровые сигналы, которые обеспечивают надёжность за счёт представления данных в двоичном виде. Понимание этих основополагающих элементов – от аппаратной архитектуры до программных алгоритмов – позволяет эффективно решать задачи и внедрять инновации в сфере цифровых технологий.

Информатика – одна из дисциплин современной науки, в рамках которой мы изучаем различные аспекты компьютерных технологий, их развитие и применение в современном мире. Информатика включает в себя широкий спектр тем, таких как развитие компьютерных технологий (аппаратных и программных), применение компьютерных технологий в современной жизни, информационные технологии, компьютерные угрозы, компьютерная безопасность и т. д.

Для удобства восприятия мы разделили эту работу на несколько глав.

«Основы компьютерных наук» и «Вычислительная техника» являются основными и значимыми дисциплинами для студентов ба-

калавриата, обучающихся по направлению «Прикладная информатика». Это учебно-методическое пособие будет полезно студентам для понимания и выполнения сложных задач, изложенных в указанных дисциплинах. Эта работа разделена на несколько глав, в которых подробно рассматриваются компоненты и функции компьютеров и вычислительной техники. При подготовке этого пособия основное внимание уделялось тому, чтобы студенты изучили все современные концепции вычислительной техники и программной инженерии. Прочитав эту работу, студенты смогут проектировать как аппаратные, так и программные компоненты компьютерных систем. Студенты должны уметь измерять производительность своих компьютерных программ и проектировать компьютерную архитектуру для своих систем. В качестве примеров работ используются учебные работы обучающихся Удмуртского государственного университета.

ГЛАВА 1. КОМПЬЮТЕР И КОМПЬЮТЕРНЫЕ НАУКИ

1.1. Что такое компьютер?

Компьютером называется электронное устройство, работающее в соответствии с заранее заданным набором инструкций. В наше время с помощью компьютера мы можем решать самые разные проблемы. Обычно современный компьютер состоит только из одного обрабатывающего компонента, такого как центральный процессор, и оперативной памяти. Работа центрального процессора состоит из выполнения логических и арифметических операций арифметико-логическим устройством, а выбор операции осуществляется устройством управления. Периферийные устройства используются для обмена информацией с внешним миром.

Основное назначение компьютера – упрощать, ускорять и повышать точность выполнения задач. В современном обществе компьютеры используются практически во всех областях, что делает их неотъемлемой частью повседневной жизни.

1.1.1. Основные характеристики компьютерной системы

- Программируемость – позволяет выполнять различные задачи путем изменения программного обеспечения.
- Обработка данных – преобразует необработанные входные данные в значимые выходные данные.
- Хранение – имеет как временную, так и оперативную память (RAM).
- Автоматизация – выполняет сложные последовательности действий без вмешательства человека.
- Скорость и точность – выполняет миллиарды операций в секунду с высокой точностью.
- Универсальность – используется практически в любой области, например, наука, бизнес, образование, развлечения, здравоохранение и т. д.

1.1.2. Основные компоненты компьютеров

Типичный компьютер состоит из:

- центрального процессора (CPU) – компьютера. Он выполняет инструкции, выполняет вычисления и управляет потоком данных;
- память (RAM) – временное хранилище, используемое для хранения данных и программ, которые используются в данный момент. Они обеспечивают быстрый доступ к центральному процессору;
- устройства хранения – долговременное хранение данных и программного обеспечения (например, жесткие диски, SSD–накопители);
- устройства ввода – позволяют пользователям вводить данные (например, клавиатура, мышь, микрофон, сканер);
- устройства вывода – отображают или передают результаты (например, монитор, принтер, динамики);
- операционная система – программное обеспечение, управляющее аппаратными и программными ресурсами (например, Windows, macOS, Linux).

1.1.3. Визуализация компьютерного цикла

Компьютер выполняет четыре основные функции. Первая функция – это ввод данных в компьютер с помощью таких устройств, как клавиатура или мышь. Вторая функция – это обработка, при которой компьютер обрабатывает данные с помощью центрального процессора (CPU). Третья функция – вывод, при котором обработанные данные отображаются на таких устройствах, как мониторы или принтеры. Последняя функция – хранение, при котором данные сохраняются для использования в будущем.

1.2. Эволюция компьютера с течением времени

История компьютеров восходит к древним временам, когда для вычислений использовались такие простые инструменты, как счеты (abacus). В 19 веке Чарльз Бэббидж представил концепцию аналитической машины, которая считается основой современных компьютеров.

Разработка электронных вычислительных машин началась в 20 веке. Первые компьютеры использовали вакуумные трубки и были очень большими и дорогими. Со временем появились новые технологии, такие как транзисторы и интегральные схемы, которые сделали компьютеры меньше и мощнее.

1.2.1. Первое поколение компьютеров: (1940–1950-е годы)

- Большая машина, использующая вакуумные трубки для логических схем.
- ENIAC (1945, Дж. Преспер Экерт и Джон Мочли) – первый электронный компьютер общего назначения. весил 30 тонн, в нем использовалось 18 000 вакуумных ламп.
- EDVAC (1949) ввел концепцию хранимых программ (программы хранятся в памяти как данные).
- UNIVAC (1951) - первый коммерческий компьютер в США, который, как известно, предсказал президентские выборы 1952 года.

1.2.2. Второе поколение – Транзисторные (1950–1960-е годы)

- Транзисторы заменили вакуумные трубки.
- Более компактные, быстрые, надежные и энергоэффективные машины.
- TRADIC (1954) – первый компьютер на транзисторах.
- IBM 1401 (1959) – популярный компьютер для бизнеса.
- DEC PDP-1 (1960) – первый мини-компьютер для научных целей.
- Появились языки программирования: ФОРТРАН (1957), КОБОЛ (1959), АЛГОЛ.

1.2.3. Третье поколение – Интегральные схемы (1960–1970-е годы)

- Интегральные схемы (ИС), объединяющие несколько транзисторов на одном кристалле.
- IMB system/360 (1964): семейство совместимых компьютеров, состоящих из совместимых компьютеров разных размеров, устанавливающих отраслевые стандарты.
- Закон Мура (1965): Гордо Мур предсказал удвоение количества транзисторов каждые два года.
- Разработаны операционные системы (например, OS/360), позволяющие работать в многозадачном режиме. Устройства ввода-вывода включают, например, принтер, мышь, клавиатуру, монитор, дисковод. К компьютеру можно подключать разные устройства.

1.2.4. Четвертое поколение – микропроцессоры и персональные компьютеры (1970–1990-е годы)

- Микропроцессоры позволяют разместить весь центральный процессор на одном кристалле.
- Intel 4004 (1971): первые микропроцессоры.
- Xerox Alto (1973): первый компьютер с графическим интерфейсом пользователя (GUI) и мышью.
- Altair 8800 (1975):первый популярный комплект микрокомпьютеров.
- Apple II (1977): первый успешный массовый персональный компьютер.
- IBM PC (1981): установил стандарт для современных ПК (использовал процессор Intel и Microsoft DOS).
- Macintosh (1984): популяризировал графический интерфейс для домашних пользователей.

1.2.5. Современная эпоха и за ее пределами (1990-е – настоящее время)

- Интернет и всемирная паутина (1990-е годы).
- Мобильные вычисления.
- Облачные вычисления.
- Искусственный интеллект и машинное обучение.
- Квантовые вычисления.
- Встроенная система и Интернет вещей.

1.3. Типы компьютеров и их компонентов

Компьютеры можно разделить на различные типы в зависимости от их размера и назначения. Суперкомпьютеры являются наиболее мощными и используются для сложных научных исследований. Универсальные компьютеры используются крупными организациями для обработки данных.

Персональные компьютеры, такие как настольные компьютеры и ноутбуки, обычно используются отдельными лицами. Мобильные устройства, включая смартфоны и планшеты, также считаются компьютерами из-за их вычислительных возможностей.

Каждый тип компьютера предназначен для удовлетворения конкретных потребностей пользователя. Различные типы компьютеров можно увидеть на рисунке 1 ниже.



Рисунок 1. Типы компьютеров, используемые в нашей повседневной жизни

1.3.1. Суперкомпьютеры

Назначение: Предназначены для чрезвычайно сложных вычислений и высокоскоростной обработки.

Производительность: Измеряется в операциях с плавающей запятой в секунду (FLOPS), часто достигает петафлопс (1015 FLOPS) или даже эксафлопс (1018 FLOPS).

Приложения:

- Прогнозирование погоды и климатические исследования.
- Ядерное моделирование и научные исследования.
- Криптография и анализ данных.
- Обучение искусственному интеллекту.

Пример: Fugaku (Япония), summit (США), Aurora (США).

1.3.2. Универсальные компьютеры

Назначение: Обработка крупномасштабных массивов данных для критически важных приложений.

Ключевые функции:

- Высокая надежность и доступность (время безотказной работы составляет 99,999 %).
- Огромный объем хранилища и пропускная способность.
- Поддержка тысяч одновременных пользователей.

Приложение:

- Банковские и финансовые операции.
- Системы бронирования авиабилетов.
- Управление государственными органами и данными переписи населения.

Примеры: IBM серии Z.

1.3.3. Мини-компьютеры (компьютеры среднего класса)

Назначение: служить промежуточным звеном между мэйнфреймами и микрокомпьютерами.

Особенности:

- многопользовательская поддержка (пользователей меньше, чем мэйнфреймов);
- подходит для предприятий среднего размера;
- дешевле, чем мэйнфреймы.

Приложения:

- обработка бизнес-данных;
- научные исследования в небольших лабораториях. Системы промышленного контроля.

Историческая справка: Были популярны в 1960–1980-х годах. Сегодня они в значительной степени заменены мощными серверами.

1.3.4. Микрокомпьютеры (персональные компьютеры)

Однопользовательские компьютеры на базе микропроцессора.

Подтипы:

- **настольные компьютеры:** стационарные устройства с отдельным монитором, клавиатурой и системным блоком;
- **портативные компьютеры:** портативные компьютеры со встроенным экраном и клавиатурой;
- **нотбуки:** небольшие, легкие ноутбуки с более низкой производительностью (были популярны примерно в 2007–2011 годах);
- **ПК «все в одном»:** настольные компьютеры с интегрированными в монитор компонентами.

Использование: общие вычисления, офисная работа, образование, развлечения.

1.3.5. Серверы

Назначение: предоставлять услуги другим компьютерам (клиентам) по сети.

Типы по функциям:

- файловые серверы;
- веб-серверы;
- серверы баз данных;
- почтовые серверы.

Характеристики:

- оптимизирован для обеспечения надежности, скорости и безопасности;
- часто монтируется в стойку в центрах обработки данных;
- может содержать несколько процессоров и большой объем оперативной памяти.

1.4. Применение компьютеров и вычислительной техники в различных сферах

Компьютеры используются практически во всех сферах жизни и работы. Среди тех, кто быстро расширяет масштабы их использования, это.

1.4.1. Компьютер в образовательном процессе

Компьютеры произвели революцию в образовании, открыв доступ к онлайн-ресурсам, электронным библиотекам и учебным видеороликам. Платформы онлайн-обучения позволяют учащимся учиться из любой точки мира.

Учителя используют компьютеры для создания презентаций, подготовки уроков и проведения виртуальных занятий. Это сделало обучение более интерактивным и увлекательным.

1.4.2. Компьютер в бизнесе

В бизнесе компьютеры используются для различных целей, таких как бухгалтерский учет, анализ данных и коммуникация. Компании используют компьютеры для эффективного управления своими операциями.

Платформы электронной коммерции позволяют компаниям продавать товары онлайн, охватывая глобальную аудиторию. Компьютеры также помогают в принятии решений, анализируя данные и создавая отчеты.

1.4.3. Компьютер в здравоохранении

Компьютеры играют решающую роль в здравоохранении.

Они используются для ведения записей о пациентах, диагностики заболеваний и управления больничными операциями. Технологии медицинской визуализации основаны на компьютерах для получения точных результатов.

Врачи используют компьютеры для изучения состояния здоровья и улучшения лечения пациентов.

1.5. Преимущества использования компьютеров и вычислительной техники

Компьютеры обладают многими преимуществами. Они выполняют задачи быстро и точно. Они могут хранить большие объемы данных и легко извлекать их. Компьютеры улучшают связь и повышают производительность. Они также позволяют автоматизировать работу, снижая необходимость в ручной работе.

Компьютеры изменили практически все аспекты современной жизни. Вот структурированный обзор их ключевых преимуществ:

1.5.1. Скорость и эффективность

- **Быстрая обработка:** компьютеры выполняют сложные вычисления за секунды – задачи, на которые у человека ушли бы часы или дни.

- **Автоматизация:** рутинные операции (ввод данных, создание отчётов, ответы на электронные письма) могут быть автоматизированы, что экономит время и усилия.
- **Быстрый доступ к информации:** поисковые системы и базы данных обеспечивают мгновенный доступ к огромным объёмам данных.

1.5.2. Точность и безошибочность изложения

- **Минимальное количество ошибок:** компьютеры точно следуют запрограммированным инструкциям, что сокращает количество человеческих ошибок при расчётах и обработке данных.
- **Согласованность:** они дают одинаковые результаты при одинаковых условиях, что имеет решающее значение для научных исследований, финансового анализа и проектирования.
- **Надёжная обработка данных:** необходима в таких областях, как здравоохранение (диагностика), финансы (транзакции) и авиация (навигация).

1.5.3. Хранение данных и управление

- **Большая ёмкость хранилища:** храните огромные объёмы данных (документы, изображения, видео, базы данных) в компактных цифровых форматах.
- **Простота поиска:** быстрый поиск и доступ к сохранённой информации с использованием ключевых слов, тегов или метаданных.
- **Организованное управление:** базы данных и файловые системы помогают эффективно классифицировать данные и управлять ими.
- **Резервное копирование и восстановление:** данные можно регулярно резервировать и восстанавливать в случае потери или повреждения.

1.5.4. Коммуникация и возможности подключения

- **Глобальный охват:** электронная почта, приложения для обмена сообщениями, видео звонки и социальные сети мгновенно объединяют людей по всему миру.
- **Совместная работа в режиме реального времени:** такие инструменты, как Google Workspace, Microsoft Teams и Slack, позволяют командам работать удаленно.
- **Обмен информацией:** облачные хранилища и платформы для обмена файлами (Google Drive, Dropbox) упрощают обмен документами и мультимедиа.

1.5.5. Универсальность и многозадачность

- **Несколько приложений:** на одном компьютере можно запускать множество программ одновременно – для обработки текстов, просмотра веб-страниц, редактирования видео, игр и т. д.
- **Широкий спектр применения:** применяется в образовании, бизнесе, здравоохранении, развлечениях, исследованиях, дизайне и т. д.
- **Индивидуализация:** программное и аппаратное обеспечение может быть адаптировано к конкретным потребностям (например, для рабочих станций графического дизайна, научных вычислительных кластеров).

1.5.6. Доступ к информации и обучению

- **Онлайн-образование:** платформы электронного обучения (Coursera, Khan Academy) предлагают курсы от ведущих учебных заведений.
- **Электронные библиотеки и ресурсы:** доступ к книгам, исследовательским статьям, учебным пособиям и практическим руководствам.
- **Развитие навыков:** изучайте новые навыки (кодирование, язык программирования, дизайн) с помощью онлайн-обучающих программ и интерактивных инструментов.

1.5.7. Инновации и креативность

- **Инструменты для творчества:** программное обеспечение для графического дизайна (Adobe Photoshop), редактирования видео (Premiere Pro), создания музыки (FL Studio) и 3D-моделирования (Blender) открывают новые формы самовыражения.
- **Прототипирование и моделирование:** САПР (система автоматизированного проектирования) и программное обеспечение для моделирования помогают инженерам и архитекторам тестировать идеи перед началом строительства.
- **Искусственный интеллект и машинное обучение:** позволяют использовать передовые приложения, такие как распознавание голоса, анализ изображений и прогнозное моделирование.

1.5.8. Эффективность затрат и использования ресурсов

- **Сокращение использования бумаги**: цифровые документы сокращают затраты на печать и хранение.
- **Удаленная работа:** экономит время на дорогу и офисные помещения; позволяет гибко организовывать работу.
- **Энергоэффективные устройства:** современные компьютеры и ноутбуки потребляют меньше энергии, чем старые модели.
- **Масштабируемость:** облачные вычисления позволяют компаниям увеличивать или уменьшать объем ресурсов по мере необходимости, оптимизируя затраты.

1.5.9. Улучшенный процесс принятия решений

- **Анализ данных:** такие инструменты, как Excel, Tableau и Python, помогают анализировать тенденции и принимать обоснованные решения.
- **Бизнес-аналитика:** информационные панели и системы отчётности позволяют получать информацию о производительности в режиме реального времени.
- **Прогнозная аналитика:** прогнозируйте будущие результаты на основе исторических данных (продажи, погода, рыночные тенденции).

1.5.10. Доступность

- **Вспомогательные технологии:** программы чтения с экрана, голосовое управление, преобразование речи в текст и адаптивные интерфейсы помогают людям с ограниченными возможностями эффективно пользоваться компьютерами.
- **Языковой перевод:** средства перевода в режиме реального времени устраняют языковые барьеры.
- **Всеобщий доступ:** Интернет и мобильные устройства предоставляют информацию и услуги удаленным и недостаточно обслуживаемым сообществам.

1.6. Недостатки использования компьютеров и вычислительной техники

Несмотря на их преимущества, компьютеры также имеют недостатки. Чрезмерное использование может привести к проблемам со здоровьем, таким как перенапряжение глаз и снижение физической активности.

Угрозы безопасности, такие как хакерские атаки и вирусы, могут повредить данные. Чрезмерная зависимость от компьютеров может снизить квалификацию человека.

1.6.1. Проблемы со здоровьем

- **Синдром перенапряжения глаз и компьютерного зрения (CVS):** длительное время, проведённое за экраном, вызывает сухость глаз, ухудшение зрения и головные боли.
- **Проблемы с опорно-двигательным аппаратом:** неправильная осанка приводит к болям в шее, спине и плечах; повторяющиеся травмы от перенапряжения (например, синдром запястного канала) при наборе текста и использовании мыши.
- **Малоподвижный образ жизни:** длительное сидение увеличивает риск ожирения, сердечных заболеваний и диабета.

1.6.2. Угрозы кибербезопасности

- **Вирусы и вредоносные программы:** могут повредить файлы, украсть данные или получить контроль над системами.
- **Взлом и утечка данных:** несанкционированный доступ к личной, финансовой или корпоративной информации.
- **Фишинг и онлайн-мошенничество:** обманом заставить пользователей раскрыть пароли или финансовые данные.
- **Кража личных данных:** похищенные персональные данные используются для мошеннических действий.

1.6.3. Конфиденциальности

- **Сбор данных:** компании и приложения отслеживают поведение пользователей, их привычки в интернете и местоположение.
- **Слежка:** правительства и организации могут отслеживать онлайн-активность.
- **Утечки и несанкционированный обмен:** личная информация может стать достоянием общественности в результате утечки данных или неосторожного обмена.

1.6.4. Социальные и психологические последствия

- **Социальная изоляция:** чрезмерное время, проведённое за экраном, сокращает общение лицом к лицу и ослабляет отношения в реальном мире.
- **Зависимость:** навязчивое использование социальных сетей, игр или просмотра веб-страниц может нарушить повседневную жизнь и продуктивность.
- **Воздействие на психическое здоровье:** социальные сети могут вызывать беспокойство, депрессию и низкую самооценку из-за сравнения с другими.
- **Снижение концентрации внимания:** постоянные уведомления и многозадачность затрудняют сосредоточение на сложных задачах.

1.6.5. Влияние на экономику и занятость

- **Перемещение рабочих мест:** автоматизация и искусственный интеллект заменяют людей на производстве, в сфере обслуживания клиентов, ввода данных и на других должностях.
- **Нехватка навыков:** пожилым работникам или тем, у кого нет навыков работы с цифровыми технологиями, может быть трудно найти работу.
- **Экономическое неравенство:** высокотехнологичные рабочие места концентрируют богатство, в то время как низко квалифицированные рабочие места сокращаются.

1.6.6. Отвлекающие факторы и снижение производительности

- **Занятия, которые отнимают время:** социальные сети, видео, игры и просмотр новостей снижают эффективность работы или учёбы.
- **Миф о многозадачности:** одновременная работа с несколькими цифровыми задачами часто снижает общую производительность и увеличивает количество ошибок.
- **Перегрузка уведомлениями:** постоянные предупреждения мешают глубокой работе и концентрации.

1.6.7. Воздействие на окружающую среду

- **Электронные отходы:** выброшенные компьютеры и устройства содержат токсичные материалы (свинец, ртуть), которые наносят вред экосистемам, если их не перерабатывать должным образом.
- **Энергопотребление:** центры обработки данных и устройства используют большое количество электроэнергии, что приводит к выбросам углекислого газа.
- **Истощение ресурсов:** для производства требуются редкоземельные металлы и другие материалы конечного назначения.

1.6.8. Технические и финансовые затраты

- **Высокая первоначальная стоимость:** мощные компьютеры и периферийные устройства обходятся частным лицам и предприятиям дорого.
- **Обслуживание и модернизация:** регулярные обновления программного обеспечения, ремонт и замена оборудования увеличивают текущие расходы.
- **Зависимость от Интернета:** для многих современных приложений требуется стабильное подключение к Интернету, которое может быть доступно не везде.

1.6.9. Зависимость и снижение навыков

- **Чрезмерное использование инструментов:** средств проверки орфографии, калькуляторов, GPS и автозамены снижает необходимость в развитии базовых навыков.
- **Ухудшение памяти:** лёгкий доступ к информации в Интернете может ослабить способность запоминать и критическое мышление.
- **Беспомощность без технологий:** люди могут испытывать трудности, когда устройства выходят из строя или Интернет недоступен.

1.6.10. Цифровой разрыв

- **Неравенство в доступе:** не все могут позволить себе компьютеры или надёжный Интернет, что создаёт пробелы в образовании и возможностях.
- **Географическое неравенство:** сельские и отдалённые районы часто имеют плохую связь по сравнению с городами.
- **Разница в возрасте:** старшему поколению может не хватать цифровой грамотности, что ограничивает их участие в жизни современного общества.

Задание № 1

Требуется решить задачу из списка задач с чётким и подробным объяснением.

1. Что такое компьютер? Каковы основные аппаратные компоненты компьютера?

Чтобы решить задачу, изучите и разберитесь в тексте подзаголовка № 1.1 и подзаголовка № 1.1.2.

2. Какие поколения компьютерных систем можно выделить в процессе эволюции? Чем каждое поколение отличается от других?

Решение задачи приведено в подзаголовке № 1.2 данной главы. Необходимо прочитать и разобраться в разработке компонентов в разное время.

3. Каковы преимущества и недостатки использования компьютерных систем в повседневной жизни и на рабочем месте?

Решение этой проблемы представлено в подразделах № 1.5 и № 1.6. Для его реализации необходимо правильно понимать, как мы используем компьютерные системы в повседневной жизни.

4. Какие существуют типы компьютерных систем? Чем каждый тип отличается от других с точки зрения функциональности, размера и компонентов?

Решение этой задачи смотрите в подзаголовке № 1.3 данной главы.

ГЛАВА 2. АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Алгоритмы и структуры данных являются двумя наиболее фундаментальными и важными понятиями в области компьютерных наук. Они формируют основу для разработки эффективных программ, решения вычислительных задач и разработки надёжных программных систем. Каждое компьютерное приложение – от простых мобильных приложений до крупномасштабных корпоративных систем – для эффективного функционирования опирается на хорошо разработанные алгоритмы и соответствующие структуры данных.

Алгоритм можно определить, как конечную последовательность чётко определённых инструкций, которые выполняются для решения конкретной проблемы или выполнения конкретной задачи. Алгоритмы обеспечивают логический и систематический подход к решению проблемы, гарантируя, что задача может быть выполнена правильно и эффективно. В компьютерном программировании алгоритмы описывают шаги, необходимые для обработки входных данных и получения желаемого результата. Хороший алгоритм должен быть чётким, недвусмысленным, эффективным и способным завершаться после конечного числа шагов.

С другой стороны, структура данных относится к специализированному формату, используемому для организации, хранения и управления данными на компьютере таким образом, чтобы к ним можно было эффективно обращаться и изменять. Структуры данных определяют, как данные располагаются в памяти и как могут выполняться такие операции, как поиск, вставка, удаление и обновление. Выбор правильной структуры данных имеет решающее значение, поскольку это напрямую влияет на производительность и результативность алгоритма.

Алгоритмы и структуры данных тесно связаны и работают вместе для создания эффективных компьютерных программ. В то время как алгоритмы сосредоточены на логике и шагах, необходимых для решения проблемы, структуры данных сосредоточены на том, как хранятся и упорядочиваются данные, необходимые для выполне-

ния этих шагов. Например, эффективный алгоритм поиска может работать плохо, если базовая структура данных не подходит для выполняемой операции.

В современных вычислительных системах важность алгоритмов и структур данных значительно возросла из-за увеличения объёма данных и сложности программных систем. Эффективные алгоритмы помогают сократить время выполнения и потребление ресурсов, в то время как хорошо спроектированные структуры данных улучшают использование памяти и управление данными. Вместе они помогают разработчикам создавать масштабируемые и высокопроизводительные приложения.

Эти концепции широко применяются во многих областях компьютерных наук и информационных технологий. При разработке программного обеспечения они используются для разработки эффективных программ и приложений. В области искусственного интеллекта и машинного обучения алгоритмы обрабатывают большие массивы данных и позволяют принимать интеллектуальные решения. В системах баз данных структуры данных помогают быстро организовывать и извлекать информацию. Аналогичным образом, операционные системы используют алгоритмы для эффективного управления процессами, памятью и системными ресурсами.

Поэтому понимание алгоритмов и структур данных имеет важное значение для студентов, изучающих информатику, и разработчиков программного обеспечения. Овладение этими понятиями позволяет программистам писать оптимизированный код, повышать производительность программ и эффективно решать сложные вычислительные задачи. По этой причине алгоритмы и структуры данных считаются основным предметом в образовании по информатике и играют решающую роль в развитии современных технологий.

2.1. Что такое алгоритм?

Алгоритм – это конечная и чётко определённая последовательность инструкций, предназначенная для выполнения конкретной задачи или решения конкретной проблемы. В информатике алгоритмы представляют собой логическую процедуру, которой компьютерная программа может следовать для обработки входных данных и получения желаемого результата. Они служат основой всех вычислительных процессов и необходимы для разработки эффективных программных систем.

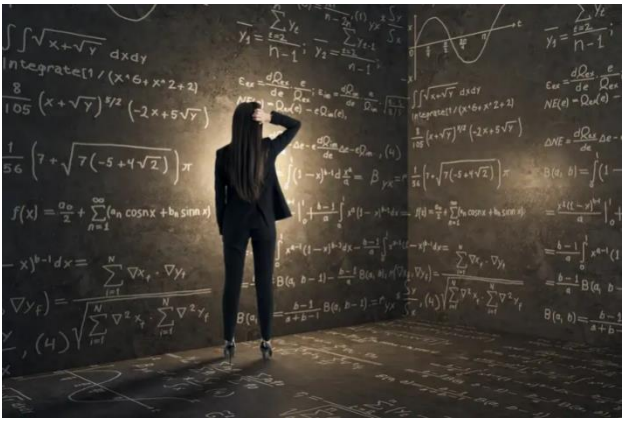


Рисунок 2. Демонстрирует мыслительный процесс, лежащий в основе использования алгоритма для решения сложной задачи

Алгоритм должен быть чётко сопределен, чтобы каждый шаг можно было выполнить без двусмысленности. Он также должен быть эффективным, что означает, что для выполнения задачи требуется разумное количество времени и ресурсов. Алгоритмы используются не только в компьютерных программах; они также используются в повседневной деятельности, например, при составлении рецепта, решении математической задачи или сборке устройства. Однако в вычислительной технике алгоритмы формулируются таким образом, что в конечном итоге могут быть реализованы с помощью языков программирования.

Ключевой характеристикой алгоритма является то, что он оперирует входными данными, обрабатывает их с помощью последовательности шагов и выдаёт выходные результаты. Например, если программе необходимо определить, существует ли определённый номер в списке номеров, алгоритм определит пошаговый процесс, необходимый для выполнения этого поиска.

2.1.1. Пример

Рассмотрим задачу поиска числа в списке. Предположим, что у нас есть список чисел, таких как:

10, 25, 30, 45, 50

Если мы хотим определить, существует ли в списке число 30, алгоритм может выполнить следующие действия:

1. Начните с первого элемента в списке.
2. Сравните текущий элемент с целевым номером.
3. Если номера совпадают, верните результат «Номер найден».
4. Если нет, перейдите к следующему элементу в списке.
5. Повторяйте процедуру до тех пор, пока не будет найден нужный номер или пока список не закончится.

Эта пошаговая процедура представляет собой простой алгоритм поиска.

2.2. Типы алгоритмов в зависимости от их функциональности

Алгоритмы можно разделить на несколько категорий в зависимости от типа задачи, которую они решают. К наиболее распространенным типам относятся:

2.2.1. Алгоритмы поиска

Алгоритмы поиска используются для определения местоположения определённого элемента в наборе данных. Они определяют, существует ли значение и, если да, то где оно находится. Примеры включают линейный поиск и бинарный поиск.

2.2.2. Алгоритмы сортировки

Алгоритмы сортировки упорядочивают данные в определённом порядке, например, по возрастанию или по убыванию. Сортировка важна для повышения эффективности поиска и обработки данных. В качестве примеров можно привести сортировку слиянием, быструю сортировку и пузырьковую сортировку.

2.2.3. Графовые алгоритмы

Графовые алгоритмы работают со структурами данных, известными как графы, которые состоят из узлов (вершин) и связей (рёбер). Эти алгоритмы широко используются в сетевом анализе, социальных сетях и оптимизации маршрутов. В качестве примеров можно привести алгоритм Дейкстры и поиск в глубину (DFS).

2.2.4. Алгоритмы динамического программирования

Динамическое программирование – это метод, используемый для решения сложных задач путём разбиения их на более мелкие подзадачи и сохранения результатов ранее решённых подзадач. Такой подход повышает эффективность за счёт исключения повторных вычислений. Динамическое программирование обычно используется в задачах оптимизации, таких как расчёт кратчайшего пути и распределение ресурсов.

2.3. Характеристики алгоритма

Чтобы алгоритм считался корректным, надёжным и практичным для реализации в компьютерных программах, он должен удовлетворять нескольким важным характеристикам. Эти характеристики гарантируют, что алгоритм может эффективно и последовательно решать задачу. В информатике хорошо разработанный алгоритм обычно оценивается на основе пяти фундаментальных свойств: входных данных, выходных данных, определённости, конечности и эффективности. Эти характеристики формируют теоретическую основу для разработки и анализа алгоритма.

2.3.1. Ввод

Алгоритм может принимать ноль или более входных данных, которые представляют собой данные, необходимые алгоритму для выполнения его задачи. Входные данные – это значения или параметры, предоставляемые алгоритму перед началом выполнения. Эти входные данные обрабатываются в несколько этапов для получения желаемых результатов.

Входные данные могут поступать из различных источников, таких как пользовательский ввод, файлы, базы данных, датчики или другие программы. Гибкость при приёме нулевых или множественных входных данных позволяет применять алгоритмы для решения широкого спектра задач.

Например, рассмотрим алгоритм, предназначенный для вычисления суммы двух чисел. В этом случае в качестве входных значений используются два числа, предоставленные пользователем. Алгоритм обрабатывает эти входные данные и выдаёт их сумму в качестве выходных данных.

Некоторые алгоритмы, например те, которые генерируют фиксированную последовательность чисел или выполняют задачи системного уровня, могут не требовать внешнего ввода. Однако большинство практических алгоритмов используют входные данные для получения значимых результатов.

2.3.2. Выход

Каждый алгоритм должен выдавать по крайней мере один результат. Выходные данные представляют собой конечный результат, полученный после обработки входных данных алгоритмом. Без выходных данных алгоритм не предоставил бы никакой значимой информации или решения.

Выходные данные могут принимать различные формы в зависимости от характера проблемы. Они могут включать числовые результаты, отсортированные данные, результаты поиска, графические представления или решения, основанные на логических условиях.

Например, в алгоритме поиска выходные данные могут указывать, существует ли определённый элемент в списке, и, если он найден, его положение в наборе данных.

В более сложных системах, таких как базы данных или приложения машинного обучения, выходные данные алгоритма могут служить входными данными для другого алгоритма, образуя часть более масштабного вычислительного процесса.

2.3.3. Определённость

Определённость относится к требованию, согласно которому каждый шаг алгоритма должен быть точно и недвусмысленно определён. Каждая инструкция должна быть понятной, чтобы её можно было правильно интерпретировать и выполнять без путаницы.

Если шаги алгоритма расплывчаты или двусмысленны, различные интерпретации могут привести к противоречивым результатам. Следовательно, алгоритмы должны определять точные операции и чётко определённые условия.

Например, инструкция «отсортировать числа» является слишком общей, если в алгоритме не указан точный метод, используемый для сортировки. Правильно определённый алгоритм будет описывать точную последовательность сравнений и перестановок, необходимых для упорядочивания чисел.

Пример конкретных инструкций:

1. Сравните первый элемент со вторым элементом.
2. Если первый элемент больше второго, поменяйте их местами.
3. Перейдите к следующей паре элементов.
4. Повторяйте процесс, пока весь список не будет отсортирован.

Поскольку каждый шаг чётко определён, алгоритм может быть последовательно реализован на любом языке программирования.

2.3.4. Конечность

Конечность означает, что алгоритм должен завершиться после конечного числа шагов. Другими словами, алгоритм не может работать бесконечно; в конечном итоге он должен достичь точки остановки и выдать результат.

Эта характеристика важна, поскольку алгоритм, который выполняется вечно и не даёт результата, бесполезен в практических вычислительных приложениях.

Например, рассмотрим алгоритм, предназначенный для вычисления факториала числа. Алгоритм многократно перемножает числа, пока не достигнет значения 1. Поскольку число уменьшается на каждом шаге, процесс в конечном итоге останавливается, что гарантирует завершение работы алгоритма.

Пример вычисления факториала 5:

$5 \times 4 \times 3 \times 2 \times 1 = 120$

Поскольку алгоритм завершается после определённого количества шагов, он удовлетворяет свойству конечности.

Однако алгоритмы, включающие циклы, должны быть тщательно разработаны, чтобы избежать бесконечных циклов, при которых условие остановки никогда не достигается.

2.3.5. Эффективность

Эффективность означает, что каждый шаг алгоритма должен быть простым, выполнимым за ограниченный промежуток времени с использованием доступных вычислительных ресурсов. Необходимые операции должны быть достаточно простыми, чтобы их мог выполнить компьютер или даже человек с помощью бумаги и карандаша.

Эффективный алгоритм позволяет избежать чрезмерно сложных или непрактичных операций, которые практически невозможно выполнить. Каждый шаг должен представлять собой фундаментальную операцию, такую как сравнение, сложение, присвоение или перемещение данных.

Например, в алгоритме поиска сравнение двух чисел считается эффективным шагом, поскольку оно простое и может быть быстро выполнено компьютерным процессором.

Пример эффективных шагов в алгоритме:

1. Считайте значение переменной.
2. Сравните два числа.
3. Присвойте значение переменной.
4. Перейдите к следующему элементу в списке.

Эти операции просты и могут быть выполнены эффективно, что делает алгоритм практичным для реальных приложений.

2.4. Что такое структура данных?

Структура данных – это систематизированный способ организации, хранения и управления данными в компьютере, обеспечивающий эффективный доступ к ним, их обработку и изменение. В информатике структуры данных играют решающую роль в разработке эффективных алгоритмов и программных систем. Они определяют, как данные располагаются в памяти и как выполняются такие операции, как вставка, удаление, поиск и обновление.

Эффективное управление данными имеет важное значение в современных вычислительных системах, поскольку программам часто приходится обрабатывать большие объёмы информации. Хорошо продуманная структура данных позволяет программам быстро выполнять операции и эффективно использовать ресурсы памяти. Поэтому выбор подходящей структуры данных является важным шагом в решении вычислительных задач и разработке высокопроизводительных приложений.

Структуры данных тесно связаны с алгоритмами. В то время как алгоритмы определяют пошаговые процедуры решения задач, структуры данных определяют, как организуются и хранятся данные, требуемые этими алгоритмами. Эффективность алгоритма часто зависит от типа используемой структуры данных. Например, поиск элемента в отсортированной структуре может быть намного быстрее, чем поиск в несортированном списке.

В практических приложениях структуры данных используются в операционных системах, системах управления базами данных, искусственном интеллекте, компьютерных сетях и многих других областях вычислительной техники. Они позволяют программам управлять сложными взаимосвязями данных и выполнять такие задачи, как хранение пользовательской информации, обработка транзакций и анализ больших наборов данных.

2.4.1. Типы структур данных, используемых для хранения данных

В информатике широко используется несколько типов структур данных. Каждая структура предназначена для обработки данных определенным образом в зависимости от типа требуемых операций.

2.4.2. Array (Массив)

Массив – одна из самых простых и часто используемых структур данных. Он состоит из набора элементов, хранящихся в смежных ячейках памяти, где к каждому элементу можно получить доступ с помощью индекса или позиции.

Например, массив, содержащий пять чисел, может выглядеть следующим образом:

Индекс: 0 1 2 3 4

Данные: 10 20 30 40 50

Массивы обеспечивают быстрый доступ к элементам, поскольку индекс напрямую определяет местоположение в памяти. Однако массивы обычно имеют фиксированный размер, что означает, что их длина должна быть определена заранее.

2.4.3. Linked Lists (Связанный список)

Это динамическая структура данных, в которой элементы, называемые узлами, соединены с помощью указателей. Каждый узел содержит две части: сами данные и ссылку (указатель) на следующий узел в последовательности.

В отличие от массивов, связанные списки не требуют смежных ячеек памяти. Это позволяет им легко увеличивать или уменьшать громкость во время выполнения программы.

Пример:

10 → 25 → 30 → 45 → NULL

Эти структуры данных полезны, когда требуется часто вставлять и удалять элементы.

2.4.4. Stack (Стек)

Это линейная структура данных, которая следует принципу «Последний вход – первый выход» (LIFO). Это означает, что последний добавленный элемент удаляется первым.

Наиболее распространённые операции со стеком включают:

- Push – добавление элемента в начало стека;
- Pop – удаление верхнего элемента;
- Peek – просмотр верхнего элемента без его удаления.

Стеки обычно используются в таких приложениях, как вычисление выражений, операции отмены в программных приложениях и управление вызовами функций в языках программирования.

2.4.5. Queues (Очередь)

Это ещё одна линейная структура данных, которая следует принципу «Первый пришёл – первый вышел» (FIFO). В очереди элемент, который вставляется первым, удаляется первым.

Основные операции с очередью включают:

- Enqueue (постановка в очередь) – добавление элемента в конец очереди;
- Dequeue (удаление из очереди) – удаление элемента из списка.

Очереди широко используются в системах планирования, управления задачами принтера и передачи данных по сети.

2.4.6. Tree (Дерево)

Это иерархическая структура данных, состоящая из узлов, соединенных ребрами. У него есть корневой узел, и каждый узел может иметь несколько дочерних узлов, образующих отношения «родитель–потомок».

Это структура данных обычно используются для представления иерархических данных, таких как файловые системы, организационные структуры и индексы баз данных.

Простым примером является бинарное дерево, где каждый узел может иметь не более двух дочерних элементов.

2.4.7. Graphs (Граф)

Граф – это структура данных, состоящая из набора вершин (узлов), соединённых рёбрами. Графы используются для представления взаимосвязей между объектами.

Примеры применения графов в реальном мире включают:

- социальные сети (связи между людьми);
- транспортные сети (дороги между городами);
- коммуникационные сети (соединения между компьютерами).

Графы широко используются в алгоритмах, связанных с кратчайшими путями, сетевым анализом и системами маршрутизации.

2.5. Важность алгоритмов и структур данных

2.5.1. Повышение эффективности программы

Эффективные алгоритмы позволяют программам быстро и точно обрабатывать большие объемы данных. Продуманные структуры данных сокращают время, необходимое для выполнения таких операций, как поиск, вставка и удаление данных.

2.5.2. Сокращение использования памяти

Оптимизированные структуры данных обеспечивают эффективное использование ресурсов памяти. Это особенно важно в системах, которые обрабатывают огромные массивы данных или работают с ограниченными аппаратными ресурсами.

2.5.3. Решение сложных задач

Многие реальные проблемы связаны со сложными взаимосвязями между элементами данных. Алгоритмы обеспечивают структурированный подход к решению этих проблем с помощью систематических шагов.

2.5.4. Оптимизация производительности системы

Высокопроизводительным системам требуются оптимизированные алгоритмы для эффективной обработки больших рабочих нагрузок. Более быстрые алгоритмы сокращают время обработки и повышают скорость отклика приложений.

Крупные технологические компании, такие как Google, Microsoft и Amazon, в значительной степени полагаются на передовые алгоритмы и сложные структуры данных для обработки огромных объёмов данных и эффективного предоставления услуг.

2.5.5. Применение в различных областях информационных технологий

Алгоритмы и структуры данных применяются во многих современных технологических системах.

2.5.6. Поисковые системы

Поисковые системы используют сложные алгоритмы ранжирования для определения релевантности веб-страниц. Например, Google использует передовые алгоритмы для анализа миллиардов веб-страниц и отображения пользователям наиболее релевантных результатов.

2.5.7. Навигационные системы

Навигационные приложения используют графические алгоритмы для расчёта кратчайшего пути между двумя точками. Эти алгоритмы анализируют дорожную сеть и данные о дорожном движении, чтобы обеспечить оптимальные маршруты.

2.5.8. Платформы социальных сетей

Платформы социальных сетей, такие как Facebook, анализируют отношения между пользователями с помощью графовых алгоритмов. Эти алгоритмы помогают предлагать друзей, выявлять сообщества и рекомендовать контент.

2.5.9. Искусственный интеллект

Системы искусственного интеллекта используют алгоритмы для изучения шаблонов из больших наборов данных. Модели машинного обучения требуют эффективных структур данных для хранения и обработки обучающих данных.

Задание № 2

Решение задач, приведённых в списке, требует понимания алгоритмов и различных типов структур данных, используемых для хранения информации в памяти компьютерных систем. Также необходимо понимать разницу между способами хранения данных в памяти компьютера и то, какая структура данных подходит для хранения того или иного типа данных в компьютерных системах.

1. Что такое алгоритм и структура данных?

Чтобы решить задачу, изучите и разберитесь в тексте подзаголовка № 2.1 и подзаголовка № 2.4.

2. В чем разница между линейными и нелинейными структурами данных?

Анализ данных приведён в разделе 2.4.1. Анализ данных требует понимания всех типов структур данных, используемых для хранения данных в памяти компьютера во время и после их обработки в центральном процессоре (CPU) компьютера.

3. Почему алгоритмы так важны? Какими характеристиками должен обладать хороший алгоритм?

Решение проблемы представлено в подразделе № 2.3 этой главы. Необходимо ознакомиться со всеми характеристиками, которые делают алгоритм хорошим и отличают его от плохого.

4. Какие бывают типы алгоритмов? Объясните на примере.

Решение этой проблемы описано в подразделе № 2.2 данной главы. Для наглядности приведём пример, упомянутый в подразделе № 2.1.1.

ГЛАВА 3. КОМПЬЮТЕРНЫЕ СЕТИ И ИНТЕРНЕТ

3.1. Компьютерная сеть

Это группа взаимосвязанных устройств, таких как компьютеры, смартфоны, принтеры и серверы, которые взаимодействуют друг с другом и совместно используют ресурсы. Сети необходимы для коммуникации и обмена данными. Они позволяют пользователям получать доступ к файлам, использовать устройства общего пользования и подключаться к Интернету.

Основные характеристики сетей:

- могут быть небольшими или большими;
- поддерживают связь;
- обеспечивают совместное использование ресурсов;
- могут быть проводными или беспроводными.

3.2. Интернет

Это глобальная сеть взаимосвязанных сетей. Он объединяет миллионы частных, общественных, академических, деловых и правительственных сетей по всему миру.

Основные характеристики:

- **Область применения:** по всему миру.
- **Размер:** миллиарды подключенных устройств.
- **Структура:** «сеть сетей» – соединяет ANs, WAN и другие сети.
- **Протоколы:** использует стандартизированные протоколы связи, такие как TCP/IP, для обеспечения обмена данными между устройствами.
- **Доступ:** общедоступный и доступный для всех, у кого есть подключение.
- **Сервисы:** включает электронную почту, просмотр веб-страниц, видеозвонки, онлайн-игры, облачные сервисы и т. д.
- **Управление:** децентрализованное – ни одна организация не владеет и не контролирует весь Интернет целиком.

Пример:

Когда вы посещаете веб-сайт, ваше устройство подключается через локальную сеть к более широкому Интернету, получая доступ к серверам, расположенным в любой точке мира.

3.3. Типы сетей

Сети классифицируются на основе географического охвата, принадлежности, технологии передачи и других критериев.

3.3.1. Personal Area Network (PAN)

- **Зона действия:** 1–10 метров (личное пространство).
- **Назначение:** подключение персональных устройств, принадлежащих одному пользователю.
- **Технологии:** Bluetooth, NFC (Near Field Communication), инфракрасное излучение, USB.

Примеры:

- смартфон подключен к беспроводным наушникам;
- умные часы синхронизируются с телефоном;
- ноутбук подключен к беспроводной мыши.

3.3.2. Локальная вычислительная сеть (LAN)

- **Зона покрытия:** ограниченная территория – дом, офис, школа или здание.
- **Назначение:** совместное использование ресурсов (принтеры, файлы, Интернет) в пределах локального местоположения.
- **Технологии:** Ethernet (проводной), Wi-Fi (беспроводной – тогда называемый WLAN).
- **Особенности:** высокая скорость, хорошая безопасность.

Примеры:

- домашняя сеть Wi-Fi;
- офисная сеть, соединяющая компьютеры и принтеры.

3.3.3. Территориальная сеть кампуса (CAN)

- **Зона покрытия:** группа близлежащих зданий или кампус.
- **Назначение:** объединение нескольких локальных сетей в рамках одной организации.
- **Технологии:** Ethernet, волоконная оптика.

Примеры:

- университетский городок, соединяющий учебные корпуса, библиотеку и общежития;
- корпоративная сеть кампусов.

3.3.4. Городская вычислительная сеть (MAN)

- **Зона покрытия:** город или столичный регион (десятки километров).
- **Назначение:** обеспечение высокоскоростной связи по всему городу.
- **Технологии:** волоконная оптика, микроволновая связь, Metro Ethernet.

Примеры:

- общегородская сеть интернет-провайдеров;
- сеть, соединяющая больницы, школы и полицейские участки для обмена данными.

3.3.5. Глобальная сеть (WAN)

- **Покрытие:** большие географические области – в городах, странах и на континентах.
- **Назначение:** объединение нескольких локальных сетей в единую сеть.
- **Технологии:** выделенные линии, спутниковые каналы связи, сети общего пользования (телефонные системы).
- **Особенности:** более сложные и потенциально менее безопасные, чем сети меньшего размера.

Примеры:

- интернет (крупнейшая глобальная сеть);
- корпоративная сеть, соединяющая офисы в разных странах.

3.3.6. Виртуальная частная сеть (VPN)

- **Определение:** защищённое соединение по общедоступной сети (обычно через Интернет).
- **Назначение:** удалённый доступ к частным сетям с использованием шифрования.

Варианты использования:

- сотрудники получают доступ к ресурсам компании из дома;
- защита данных при использовании общедоступного Wi-Fi.

3.3.7. Интранет

- **Определение:** частная сеть внутри организации.
- **Доступ:** только для авторизованных пользователей (сотрудников).
- **Протоколы:** TCP/IP, HTTP, FTP.
- **Содержимое:** внутренние документы, порталы для сотрудников, системы управления персоналом.

3.3.8. Экстранет

- **Определение:** расширение внутренней сети для отдельных внешних пользователей.
- **Доступ:** ограниченный доступ для партнёров, поставщиков или клиентов.
- **Цель:** безопасное сотрудничество с доверенными сторонними организациями.

Пример:

- портал поставщиков для управления заказами.

3.3.9. Internetwork

- **Определение:** совокупность двух или более независимых сетей, соединённых вместе.
- **Функция:** обеспечивает связь между устройствами в разных сетях.
- **Самый крупный пример:** сам Интернет.

3.4. Использование Интернета

Сети и Интернет изменили почти все аспекты современной жизни. Вот подробный обзор их основных применений.

3.4.1. Общение

Сети обеспечивают быстрые, надёжные и разнообразные способы коммуникации:

- **Электронная почта:** отправка сообщений и файлов по всему миру (например, Gmail, Outlook).
- **Мгновенный обмен сообщениями:** обмен текстовыми сообщениями в режиме реального времени (например, WhatsApp, Telegram, Slack).
- **Голосовые и видеозвонки:** VoIP-сервисы, такие как Skype, Zoom, Google Meet, Teams.
- **Социальные сети:** платформы для обмена обновлениями, фотографиями, видео и общения с другими пользователями (например, Facebook, X).
- **Дискуссионные форумы и сообщества:** обмен идеями и решение проблем (например, Reddit, Stack Overflow).

3.4.2. Совместное использование ресурсов

Сети позволяют нескольким пользователям получать доступ к оборудованию, программному обеспечению и данным и совместно использовать их:

- **оборудование:** принтеры, сканеры, устройства хранения данных (NAS), совместно используемые в локальной сети;

- **файлы и данные:** облачные хранилища (Google Drive, Dropbox, OneDrive) и сетевые хранилища;
- **программное обеспечение:** централизованная установка программного обеспечения, доступная нескольким пользователям (например, Office 365, корпоративные ERP-системы);
- **подключение к Интернету:** совместное использование одного подключения к Интернету несколькими устройствами (через маршрутизатор в домашней или офисной сети).

3.4.3. Доступ к информации

Интернет является крупнейшим в мире хранилищем информации:

- **веб-сайты и блоги:** новости, учебные пособия, исследования, развлечения;
- **онлайн-энциклопедии:** Википедия и другие базы знаний;
- **электронные библиотеки:** Google Books, Project Gutenberg, академические базы данных (JSTOR, IEEE Xplore);
- **поисковые системы:** Google, Bing, Duck Duck Go для быстрого поиска любой информации.

3.4.4. Электронная коммерция и онлайн-банкинг

Цифровые транзакции произвели революцию в электронной коммерции и финансах:

- **онлайн-покупки:** Amazon, eBay, Ali Express, местные интернет-магазины;
- **цифровые платежи:** PayPal, Apple Pay, Google Pay, Stripe, SberPay, Tinkoff Pay;
- **онлайн-банкинг:** управление счетами, перевод средств, оплата счетов (через банковские веб-сайты или приложения);
- **криптовалюты:** Биткоин, Эфириум и другие транзакции, основанные на блокчейне.

3.4.5. Образование и электронное обучение

Сети поддерживают обучение на протяжении всей жизни и дистанционное образование:

- **MOOC:** Coursera, edX, Udemy, Stepik.
- **Виртуальные классы:** Zoom, Google Classroom, Moodle.
- **Обучающие видеоролики:** YouTube tutorials, Khan Academy.
- **Цифровые учебники и ресурсы:** онлайн-справочники и научные журналы.

3.4.6. Развлечения и медиа

Потоковое вещание и цифровой контент доминируют в современном мире:

- **потоковое видео:** YouTube, Netflix, Disney+, Кинопоиск, IVI;
- **потоковое воспроизведение музыки:** Spotify, Apple Music, Yandex Music, VK Music;
- **онлайн-игры:** многопользовательские игры, облачные игры (GeForce Now, Xbox Cloud Gaming);
- **электронные и аудиокниги:** Kindle, Book mate, Story tel.

3.4.7. На работе и в бизнесе

Сети необходимы для современных рабочих мест и организаций:

- **удалённая работа:** доступ к ресурсам компании из любой точки мира (VPN, облачные приложения);
- **инструменты для совместной работы**: Microsoft Teams, Slack, Asana, Trello, Notion;
- **облачные вычисления:** SaaS (Google Workspace), IaaS (AWS, Yandex Cloud), PaaS-решения;
- **корпоративные системы:** ERP (SAP), CRM (Salesforce), инструменты управления проектами;
- **видеоконференцсвязь:** регулярные совещания и глобальные конференции (Zoom, Webex).

3.4.8. Интернет вещей (IoT)

Подключенные устройства автоматизируют и оптимизируют окружающую среду:

- **умные дома:** термостаты, освещение, камеры видеонаблюдения, голосовые помощники (например, Alexa, Google Home);
- **промышленный интернет вещей:** датчики и автоматизация в производстве, логистике, сельском хозяйстве;
- **носимые устройства:** фитнес-трекеры, умные часы, отслеживающие данные о состоянии здоровья.

3.4.9. Здравоохранение

Цифровые сети улучшают уход за пациентами и научные исследования:

- **телемедицина:** удалённые консультации и диагностика;
- **электронные медицинские карты (EHR):** централизованные данные о пациентах, доступные авторизованным поставщикам;
- **медицинские исследования:** глобальный обмен данными и сотрудничество;
- **мониторинг здоровья:** удалённое наблюдение за пациентами с помощью подключённых устройств.

3.4.10. Государственные услуги и управление

Правительства используют сети для более эффективного обслуживания граждан:

- **порталы электронного правительства:** онлайн-регистрация налоговых деклараций, заявки на получение лицензий, социальные услуги;
- **оповещения общественности:** экстренные уведомления через мобильные сети;
- **платформы открытых данных:** предоставление государственных данных в открытом доступе для обеспечения прозрачности и инноваций.

3.4.11. Безопасность и видеонаблюдение

Сети повышают безопасность и мониторинг:

- **системы видеонаблюдения:** подключенные камеры с удаленным доступом;
- **обнаружение вторжений:** сетевой мониторинг киберугроз;
- **системы контроля доступа:** интеллектуальные замки и аутентификация с помощью сетевых устройств.

3.5. Будущие тенденции развития сетей

3.5.1. 5G и 6G

5G уже трансформирует отрасли благодаря:

- сверхвысоким скоростям передачи данных (до 10 Гбит/с);
- чрезвычайно низкой задержке (1–10 мс).;
- широкому спектру подключения устройств (до 1 миллиона устройств на км^2).

Ключевые области применения 5G:

- интернет вещей (IoT) и «умные города»;
- автономные транспортные средства и связь «от машины ко всему» (V2X);
- удалённая хирургия и телемедицина;
- промышленная автоматизация и Индустрия 4.0;
- иммерсивные технологии (AR/VR).

6G (ожидается примерно к 2030 году) позволит продвинуться в этом направлении еще дальше:

- терагерцовые частоты (от 100 ГГц до 3 ТГц);
- субмиллисекундная задержка;
- интегрированное зондирование и связь;
- голографическая связь;
- расширенная интеграция искусственного интеллекта на сетевом уровне;
- повсеместное подключение (включая сети космического базирования).

Проблемы 6G:

- управление спектром и помехи;
- потребление энергии;
- безопасность и конфиденциальность;
- затраты на развёртывание инфраструктуры.

3.5.2. Web 3.0 (Децентрализованный интернет)

Web 3.0 представляет собой переход к децентрализованному Интернету, в котором пользователи имеют больший контроль над своими данными и цифровыми активами.

Основные технологии:

- блокчейн и технология распределенного реестра;
- смарт-контракты;
- децентрализованные приложения (D Apps);
- взаимозаменяемые токены (NFT);
- децентрализованные автономные организации (DAO).

Ключевые характеристики:

- децентрализация: данные распределяются по узлам, а не хранятся на центральных серверах;
- владение пользователями: отдельные лица контролируют свою цифровую идентификацию и данные;
- надежные системы: транзакции проверяются с помощью механизмов консенсуса;
- интероперабельность: бесшовное взаимодействие между различными платформами и сервисами;
- семантическая паутина: машины лучше понимают контент и контекст;
- интеграция искусственного интеллекта: интеллектуальные агенты помогают пользователям ориентироваться в Интернете.

Потенциальные приложения:

- децентрализованные финансы (DeFi);
- креативная экономика с прямой монетизацией;

- безопасное управление цифровыми идентификационными данными;
- прозрачные системы управления.

Проблемы:

- проблемы масштабируемости;
- сложность взаимодействия с пользователем;
- неопределённость в регулировании;
- энергопотребление блокчейн-сетей.

3.5.3. Интеграция искусственного интеллекта и машинного обучения

Искусственный интеллект будет играть все более важную роль в управлении сетью.

Текущие и будущие приложения:

- автоматизированная настройка и оптимизация сети;
- профилактическое обслуживание и обнаружение аномалий;
- прогнозирование трафика и балансировка нагрузки;
- динамическое распределение ресурсов;
- обнаружение угроз кибербезопасности и реагирование на них;
- управление качеством обслуживания (QoS);
- самовосстанавливающиеся сети, которые автоматически восстанавливаются после сбоев.

Специальные технологии:

- машинное обучение (ML) для сетевой аналитики;
- глубокое обучение (DL) для распознавания сложных образов;
- обучение с подкреплением для адаптивной маршрутизации;
- обработка естественного языка (NLP) для интерфейсов сетевого управления.

3.5.4. Расширение интернета вещей (IoT)

IoT продолжит расти в геометрической прогрессии, подключая миллиарды новых устройств:

Тенденции:

- массовый интернет вещей (глобальные сети с низким энергопотреблением);
- критически важный интернет вещей (сверхнадежные приложения с низкой задержкой);
- промышленный интернет вещей (IIoT) для интеллектуального производства;
- интернет вещей в здравоохранении для удаленного мониторинга пациентов;
- инфраструктура «умных городов».

Расширяющие возможности технологии:

- разделение сети (создание виртуальных сетей для конкретных приложений);
- передовые вычисления (обработка данных ближе к источнику);
- туманные вычисления (парадигма распределенных вычислений);
- глобальные сети с низким энергопотреблением (LPWAN).

3.5.5. Квантовые технологии

Квантовые технологии произведут революцию в сетях и безопасности.

Квантовые вычисления:

- решение сложных задач оптимизации сетевой маршрутизации;
- ускорение криптографических операций;
- усовершенствование алгоритмов искусственного интеллекта для управления сетью.

Квантовая коммуникация:

- квантовое распределение ключей (QKD) для надёжного **шифрования;**
- квантовые сети для безопасной коммуникации;
- системы связи на основе запутанности.

Квантовый интернет:

- подключение квантовых компьютеров по всему миру;
- распределенные квантовые вычисления;
- сверхзащищенные каналы связи.

3.5.6. Спутниковый интернет

Глобальный охват с помощью спутниковых группировок:

- Star link от SpaceX;
- Project Kuiper от Amazon;
- One Web;
- низкоорбитальные спутники (LEO), обеспечивающие высокоскоростной доступ в Интернет по всему миру;
- преодоление цифрового неравенства в отдалённых районах.

3.5.7. Повышенная безопасность

Новые подходы к кибербезопасности:

- архитектура с нулевым уровнем доверия;
- биометрическая аутентификация;
- гомоморфное шифрование (вычисления на основе зашифрованных данных);
- управление идентификационными данными на основе блокчейна;
- обнаружение угроз с использованием искусственного интеллекта;
- технологии, обеспечивающие конфиденциальность (дифференцированная конфиденциальность, безопасные многопользовательские вычисления).

3.5.8. Основные проблемы и соображения

Технические проблемы:

- управление экспоненциальным ростом объема данных;
- обеспечение надежности и отказоустойчивости сети;
- стандартизация новых технологий;
- энергоэффективность и экологичность.

Социальные и экономические проблемы:

- цифровой разрыв и равный доступ;
- сокращение рабочих мест из-за автоматизации;
- этичное использование искусственного интеллекта и данных;
- регулирование децентрализованных систем;
- интеллектуальная собственность в децентрализованных средах.

Проблемы безопасности и конфиденциальности:

- защита от угроз квантовых вычислений;
- защита миллиардов устройств Интернета вещей;
- предотвращение подделок и создания синтетических медиабаз;
- сохранение конфиденциальности в средах с гиперсвязью.

3.5.9. Искусственный интеллект и автоматизация

Интеллектуальное управление сетью. ИИ будет играть все большую роль в:

- прогнозировании сетевых сбоев;
- оптимизации маршрутизации трафика;
- обнаружении и устранении угроз безопасности;
- самовосстанавливающихся сетях, которые автоматически восстанавливаются после сбоев.

3.5.10. Самооптимизирующиеся системы

- Алгоритмы машинного обучения позволят сетям адаптироваться в режиме реального времени, динамически распределяя ресурсы и настраивая параметры для достижения максимальной эффективности.
- Персонализированный пользовательский опыт.
- Искусственный интеллект будет анализировать поведение пользователей, предлагая контент и услуги с учетом индивидуальных потребностей.
- Корректируйте настройки сети для достижения оптимальной производительности.
- Предвидеть требования пользователя до того, как они будут четко сформулированы.

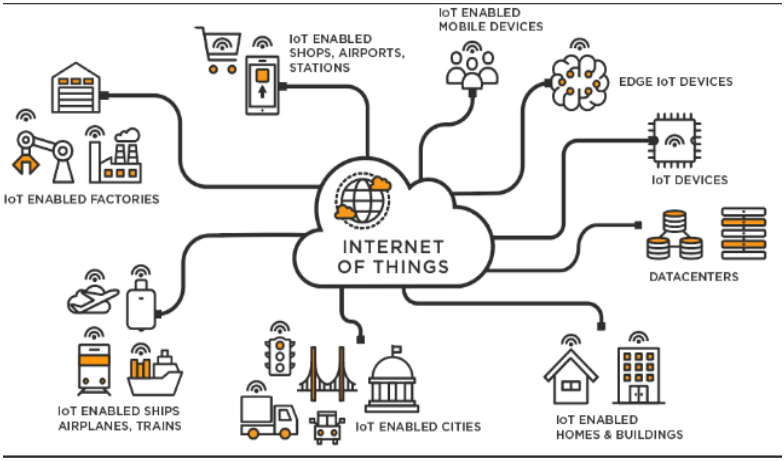


Рисунок 3. Пример интернета вещей (IoT)

Задание № 3

1. Что такое компьютерная сеть? Что такое Интернет?

Чтобы решить задачу, изучите и разберитесь в тексте подзаголовка № 3.1 и подзаголовка № 3.2.

2. Каковы основные типы компьютерных сетей по географическому охвату?

Ответ на эти вопросы содержится в подзаголовке № 3.3 и его подразделах. Чтобы дать точный ответ, необходимо понять, как географическое расположение компьютерных систем и их аппаратных ресурсов влияет на выбор конкретной сети.

3. В чем разница между Интернетом и Всемирной паутиной (WWW)?

Решение этой задачи приведено в подзаголовке № 3.2 и № 3.5. Это можно понять, если разобраться во взаимосвязи между Интернетом и Всемирной паутиной и в том, как они дополняют друг друга при выполнении своих задач.

4. Подробно объясните следующее с помощью примеров.

- Что такое VPN?
- Как обычно используется Интернет?
- Что такое Web 3.0?
- Что такое IoT?

Для решения поставленной задачи необходимо понимание тексты, указанные в заголовке № 3 и его подзаголовках.

ГЛАВА 4. ОПЕРАЦИОННАЯ СИСТЕМА

Операционная система (OS) – это фундаментальное системное программное обеспечение, которое управляет аппаратными и программными ресурсами компьютера, обеспечивая интерфейс для пользователей и приложений. Он выступает в качестве посредника между аппаратными компонентами, такими как центральный процессор (CPU), оперативная память, устройства хранения данных и периферийные устройства ввода/вывода, и программными приложениями, которые обеспечивают правильную работу этих компонентов. Без операционной системы пользователям пришлось бы напрямую взаимодействовать с оборудованием, используя низкоуровневые инструкции, что было бы сложным, подверженным ошибкам и непрактичным.

Важность операционных систем значительно возросла с развитием вычислительной техники. Первые компьютеры были большими, дорогими и управлялись специалистами, но сегодня операционные системы встроены практически в каждое цифровое устройство, от настольных компьютеров и ноутбуков до смартфонов, встраиваемых систем, облачных серверов и устройств Интернета вещей (IoT).

Современные операционные системы не только управляют ресурсами, но и предоставляют такие услуги, как безопасность, создание сетей, аутентификация пользователей, управление файлами и координация процессов. Хорошо спроектированная операционная система обеспечивает эффективное использование ресурсов, стабильность системы, безопасность и удобство для пользователя, что делает её центральным элементом современных вычислений.

4.1. История эволюции операционных систем

Развитие операционных систем можно понять, изучив основные поколения компьютерных систем. Каждое поколение предъявляло новые требования и приводило к появлению новых конструкций и концепций операционных систем.

4.1.1. Первое поколение (1940–1950-е годы): без операционной системы

На раннем этапе развития вычислительной техники операционных систем не существовало. Первые компьютеры, такие как ENIAC и UNIVAC, программировались вручную с помощью переключателей, соединительных кабелей и перфокарт. Каждая программа должна была загружаться в память с помощью физического носителя и выполняться индивидуально. Не существовало концепции автоматического планирования заданий или интерактивного использования. Программистом и оператором часто был один и тот же человек, и для написания и запуска программ им требовалось глубокое знание аппаратного обеспечения.

Этот процесс отнимал много времени и был подвержен ошибкам. Простые ошибки при подключении или перфорации платы могли привести к сбоям или неверным результатам. Поскольку не было уровня абстракции, позволяющего скрыть детали оборудования, каждая машина, как правило, имела свой собственный уникальный набор команд и рабочий процесс, что ограничивало переносимость и повторное использование программного обеспечения. Однако этот период заложил основу для последующей автоматизации и осознания того, что для управления сложностями необходим промежуточный системный уровень.

4.1.2. Второе поколение (1950–1960-е годы): Batch system (система пакетной обработки)

Для повышения эффективности и сокращения времени простоя процессора были внедрены пакетные операционные системы. При пакетной обработке задания (программы и данные) собирались в пакеты и обрабатывались последовательно без участия пользователя во время выполнения. Пользователи записывали свои программы на перфокартах или магнитной ленте и передавали их операторам компьютеров, которые затем группировали их и вводили в машину.

Операционная система, часто в виде простой программы-монитора, поочерёдно контролировала загрузку и выполнение каждого задания. Это уменьшило количество требуемых ручных настроек и повысило общую производительность системы.

Однако пакетным системам не хватало гибкости и интерактивности. После отправки пакета пользователю приходилось ждать завершения обработки и получения выходных данных, что затрудняло отладку и работу, требующую больших затрат времени.

4.1.3. Третье поколение (1960–1970-е): мультипрограммирование и Time sharing (разделение времени)

Третье поколение компьютеров ввело концепцию мультипрограммирования и разделения времени. При мультипрограммировании в памяти одновременно находится несколько программ, и операционная система переключает центральный процессор между ними, чтобы он был занят, пока другие ожидают выполнения операций ввода/вывода. Это улучшило общую загрузку процессора и позволило более эффективно использовать дорогостоящие аппаратные ресурсы.

Разделение времени расширило эту модель, позволив поддерживать одновременную работу нескольких интерактивных пользователей. Каждый пользователь подключался через терминал, и ему предоставлялся небольшой отрезок времени для работы процессора в циклическом режиме. Несмотря на то, что процессор быстро переключался между пользователями, каждый пользователь чувствовал, что у него есть своя выделенная машина. Эта эпоха ознаменовала начало интерактивных вычислений, когда пользователи могли выполнять команды, редактировать файлы и получать немедленную обратную связь от системы.

В этот период появились такие системы, как MULTICS и ранний UNIX, которые привнесли такие ключевые идеи, как иерархические файловые системы, абстрагирование процессов, механизмы защиты и дизайн портативных операционных систем.

4.1.4. Четвертое поколение (1980–2000-х годов): персональный компьютер (ПК, персональная электронно-вычислительная машина, ПЭВМ)

Развитие микропроцессоров привело к появлению персональных компьютеров. Вместо больших компьютеров (mainframes), совместно используемых многими пользователями, отдельные люди могли приобретать относительно недорогие компьютеры для личного и делового использования. Операционные системы эволюционировали, чтобы стать более удобными в использовании и доступными для пользователей, не имеющих технических навыков.

Графические интерфейсы пользователя (GUI) заменили или дополнили интерфейсы командной строки, позволяя пользователям взаимодействовать с системой с помощью окон, значков, меню и указательных устройств, таких как мышь. Такие системы, как MS DOS, Windows, классическая Mac OS и ранние настольные версии UNIX, определили возможности персональных компьютеров. Операционные системы в эту эпоху начали поддерживать преимущественную многозадачность, мультимедиа, сетевое взаимодействие и управление устройствами plug and play.

Операционные системы с открытым исходным кодом, особенно UNIX-подобные системы, такие как Linux, завоевали популярность среди серверов и технических пользователей. Сетевые возможности стали стандартом, что способствовало развитию локальных вычислительных сетей (LAN), Интернета и клиент-серверных архитектур.

4.1.5. Современная эпоха / эра (2000-е – настоящее время)

Современные операционные системы уделяют большое внимание производительности, безопасности, масштабируемости, возможностям подключения и удобству использования. Они должны поддерживать широкий спектр аппаратных платформ и сценариев использования, включая настольные компьютеры, ноутбуки, смартфоны,

планшеты, встроенные системы, игровые консоли и крупномасштабные облачные инфраструктуры.

Ключевые тенденции включают поддержку виртуализации, облачных вычислений, многоядерных процессоров и распределенных систем. Мобильные операционные системы, такие как Android и iOS, оптимизированы для сенсорных интерфейсов, управления питанием и экосистем приложений. Серверные и облачные операционные системы должны обеспечивать виртуализацию, контейнеризацию, высокую доступность и согласование ресурсов в больших масштабах. Безопасность стала главной проблемой, поскольку операционные системы включают в себя такие функции, как контроль доступа, шифрование, изолированная среда и механизмы безопасной загрузки. Кроме того, интеграция искусственного интеллекта и машинного обучения все чаще используется для таких задач, как оптимизация производительности, обнаружение аномалий и адаптивное планирование ресурсов.

4.2. Основные функции операционных систем

Хотя операционные системы различаются по дизайну и реализации, они разделяют несколько основных функций, которые позволяют системе функционировать правильно и эффективно.

4.2.1. Управление процессами (Process Management)

Управление процессами включает в себя управление выполнением программ. Процесс можно рассматривать как выполняемую программу, а также её текущее состояние, включая счётчик программ, регистры, переменные и выделенные ресурсы. Операционная система отвечает за:

- создание и завершение процессов;
- планирование процессов для выполнения в центральном процессоре;
- управление состояниями процессов (готовность, запуск, ожидание, завершение);

- управление взаимодействием между процессами (IPC);
- предоставление механизмов синхронизации (таких как семафоры и мьютексы).

Операционная система использует алгоритмы планирования, чтобы решить, какой процесс должен выполняться в любой момент времени. Эти алгоритмы направлены на оптимизацию загрузки процессора, пропускной способности, времени отклика и справедливости. В качестве примеров можно привести First Come First Serve (FCFS), Shortest Job First (SJF), приоритетное планирование и циклический поиск. В многоядерных и многопроцессорных системах операционная система также должна управлять балансировкой нагрузки между процессорами.

4.2.2. Управление памятью (Memory Management)

Управление памятью отвечает за управление основной памятью компьютера (RAM). Операционная система должна выделять память процессам, отслеживать, какие части памяти используются, и освобождать память, когда она больше не нужна. Целью является максимальное использование памяти, предотвращение взаимодействия процессов друг с другом и обеспечение удобной абстракции программирования.

Ключевые методы включают в себя:

- непрерывное распределение, при котором каждому процессу выделяется отдельный непрерывный блок памяти;
- разбиение на страницы (pagging), при котором память разделяется на блоки фиксированного размера, называемые страницами и фреймами, что обеспечивает непрерывное распределение и упрощает управление памятью;
- сегментацию (segmentation), при которой память делится на сегменты переменного размера на основе логических единиц, таких как код, данные и стек;

- виртуальную память (virtual memory), которая увеличивает видимый объем памяти за счет использования дискового пространства для хранения частей процессов, которые в данный момент не нужны в физической оперативной памяти.

Виртуальная память позволяет системам запускать программы, объем которых превышает объем физически доступной памяти, и поддерживает мультипрограммирование, предоставляя каждому процессу собственное виртуальное адресное пространство. Однако чрезмерное использование виртуальной памяти может привести к сбоям в работе, когда система тратит большую часть своего времени на замену страниц на диске.

4.2.3. Управление файловой системой (File System Management)

Файловая система – это компонент операционной системы, который управляет хранением и извлечением данных на устройствах хранения, таких как жесткие диски, SSD-накопители и съемные носители. Она организует данные в файлы и каталоги, обеспечивая иерархическую структуру, по которой пользователи и приложения могут перемещаться.

Операционная система отвечает за:

- создание, удаление, чтение и запись файлов;
- управление каталогами и путями доступа;
- хранение метаданных (таких как размер файла, временные метки и разрешения);
- обеспечение контроля доступа и разрешений;
- обеспечение целостности данных и возможности восстановления.

4.2.4. Управление устройствами (Device Management)

Управление устройствами включает в себя управление и координацию аппаратных компонентов, таких как принтеры, диски, клавиатуры, дисплеи и сетевые интерфейсы. Операционная система использует драйверы устройств – специализированные программные модули – для взаимодействия с этими устройствами. Драйверы преобразуют общие команды операционной системы в операции, специфичные для устройства.

Операционная система должна обрабатывать:

- инициализацию и настройку устройства;
- операции ввода-вывода (I/O);
- обработку прерываний, когда устройствам требуется внимание центрального процессора;
- буферизацию, кэширование и буферизацию очереди для оптимизации производительности.

Предоставляя единый интерфейс для широкого спектра устройств, операционная система позволяет приложениям выполнять операции ввода-вывода без необходимости знать детали реализации каждого устройства.

4.2.5. Безопасность и защита (Security and Protection)

Безопасность и защитные механизмы обеспечивают доступ к ресурсам только разрешённым пользователям и процессам. Операционная система обеспечивает соблюдение политик безопасности посредством:

- проверки личности пользователей (например, паролей, биометрических данных, токенов).
- определения того, какие действия разрешено выполнять пользователю (например, разрешения на доступ к файлам, списки контроля доступа).
- шифрования, защиты конфиденциальности данных как при хранении, так и при передаче.

- разделения и ограничения процессов, их доступа к системным ресурсам и предотвращение их взаимодействия друг с другом.
- постоянной записи событий, связанных с безопасностью, для анализа и подготовки оптимального ответа на них.

Современные системы также интегрируются с брандмауэрами и механизмами для защиты целостности самой операционной системы.

4.3. Управление процессами и потоками

Управление процессами и потоками играет ключевую роль в обеспечении параллелизма и быстродействия операционной системы. По мере усложнения приложений и внедрения многоядерного аппаратного обеспечения эффективное управление процессами и потоками становится все более важным.

Процесс – это выполняемая программа, включая ее код, данные, стек и связанные с ними ресурсы операционной системы.

Поток – это меньшая единица выполнения в рамках процесса, использующая то же адресное пространство и ресурсы, что и другие потоки в этом процессе.

4.3.1. Состояния процесса и переходы

Процессы обычно переходят из одного состояния в другое в течение своего жизненного цикла. Хотя точные названия могут различаться в разных операционных системах, общие состояния включают:

- **Новое:** процесс создаётся.
- **Готово:** процесс подготовлен к запуску и ожидает выделения ресурсов центрального процессора.
- **Запущено:** процесс в данный момент выполняется на центральном процессоре.
- **Ожидание (или блокировка):** процесс не может быть запущен до тех пор, пока не произойдёт внешнее событие, например, завершение ввода-вывода.
- **Завершено:** процесс завершил выполнение и очищается.

Операционная система поддерживает структуры данных, часто называемые блоками управления процессами (PCBS), для отслеживания состояния, регистров процессора, сопоставлений памяти и использования ресурсов каждым процессом. Когда операционная система переключает процессор с одного процесса на другой – переключение контекста – это сохраняет состояние текущего процесса и загружает состояние следующего.

4.4. Современные тенденции в области операционных систем

Операционные системы продолжают развиваться в соответствии с новыми аппаратными возможностями, требованиями приложений и средами развёртывания. Последние тенденции включают интеграцию в облако и управление с помощью искусственного интеллекта.

Одной из основных тенденций является глубокая интеграция операционных систем с платформами облачных вычислений. Настольные и мобильные операционные системы все чаще внедряют облачные хранилища и сервисы непосредственно в пользовательскую среду, обеспечивая бесперебойную синхронизацию файлов, размещение приложений и удалённый доступ на разных устройствах. Корпоративные операционные системы предназначены для эффективной работы в виртуализированных и контейнерных облачных инфраструктурах, обеспечивая гибкость, многопользовательскую работу и автоматизированную оркестровку.

Другой важной тенденцией является развитие IoT (интернета вещей) и операционных систем для передовых вычислений. Для устройств Интернета вещей требуются лёгкие, энергоэффективные и часто поддерживающие работу в режиме реального времени операционные системы, такие как FreeRTOS, Zephyr и специализированные встроенные дистрибутивы Linux. Эти системы оптимизированы для работы с ограниченными объёмами памяти и вычислительной мощности, но при этом поддерживают сетевое взаимодействие, безопасность и иногда оперативное обновление (OTA). Передовые

операционные системы все чаще включают в себя функции локальной обработки данных и передовой искусственный интеллект, позволяя устройствам выполнять логические выводы с помощью машинного обучения, не всегда полагаясь на удалённые облачные серверы.

Искусственный интеллект и машинное обучение также оказывают более широкое влияние на разработку операционных систем. Встраиваемый Linux и другие платформы теперь поддерживают такие платформы, как TensorFlow Lite, для запуска рабочих нагрузок с использованием искусственного интеллекта непосредственно на периферийных устройствах, что позволяет принимать решения в режиме реального времени в таких областях, как промышленная автоматизация, мониторинг здравоохранения и автономные системы. В то же время производители операционных систем изучают возможности внутреннего машинного обучения для решения таких задач, как обнаружение аномалий, профилактическое обслуживание и адаптивное планирование ресурсов.

Наконец, операционные системы становятся все более ориентированными на безопасность и обновление, особенно в IoT (Интернет вещей) и встраиваемых системах. Такие функции, как безопасная загрузка, зашифрованное хранилище и долгосрочная поддержка (LTS) ядер, помогают поддерживать доверие на протяжении длительного жизненного цикла, характерного для встраиваемых и промышленных систем. В совокупности эти тенденции указывают на то, что операционные системы становятся более распределенными, интеллектуальными и тесно связанными с сетевыми сервисами, чем когда-либо прежде.

Задание № 4

1. Что такая операционная система? Назовите основные функции операционной системы.

Чтобы решить задачу, изучите и разберитесь в тексте подзаголовка № 4.1 и подзаголовка № 4.2.

2. Что такое процесс и поток в операционной системе? Как они связаны?

Решение задачи приведено в подзаголовке № 4.3 и подзаголовка № 4.2 данной главы. Необходимо прочитать и понять все характеристики процесса и его состояний во время выполнения.

3. Какова основная роль операционной системы?

Чтобы ответить на этот вопрос, необходимо внимательно изучить подзаголовки 4.2. Поскольку операционная система играет ключевую роль во всех аспектах работы компьютера, важно понимать все её функции.

ГЛАВА 5. ПРОГРАММНАЯ ИНЖЕНЕРИЯ В СОВРЕМЕННОЙ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1. Что такое программная инженерия?

Программная инженерия – это системный и упорядоченный подход к проектированию, разработке, тестированию, развертыванию и сопровождению программных систем. Она применяет инженерные принципы к созданию программного обеспечения, чтобы разрабатывать надежные, эффективные, масштабируемые и удобные в обслуживании программные решения.

В отличие от простого программирования, программная инженерия охватывает весь жизненный цикл программных систем, включая планирование, анализ требований, проектирование архитектуры, тестирование, сопровождение и документирование.

5.2. Цели и задачи программной инженерии

Основная цель программной инженерии – применение систематических, упорядоченных и поддающихся количественной оценке подходов к разработке, эксплуатации и сопровождению программного обеспечения. Поскольку современные организации в значительной степени полагаются на цифровые системы, программное обеспечение должно быть не только функциональным, но и надёжным, безопасны и масштабируемым.

Программная инженерия направлена на создание программных решений, отвечающих требованиям пользователей и соответствующих высоким стандартам качества, эффективности и устойчивости. Ниже перечислены основные цели и задачи программной инженерии.

В этой архитектуре команда во время её выполнения можно напрямую обращаться к памяти. Основным недостатком этого подхода является то, что для него требуется больший размер временной памяти, а поскольку доступ к данным осуществляется из памяти во время выполнения команды, требуется больше времени для доступа к этой памяти.

5.2.1. Создание высококачественных программных систем

Одной из фундаментальных задач программной инженерии является разработка высококачественных программных систем, которые отвечают, как функциональным, так и нефункциональным требованиям. Высококачественное программное обеспечение выполняет поставленные перед ним задачи правильно, эффективно и последовательно, обеспечивая при этом положительный пользовательский опыт.

Качество программного обеспечения обычно оценивается с использованием нескольких параметров, включая:

- **корректность:** программное обеспечение должно выполнять функции, указанные в требованиях, без ошибок;
- **удобство использования**: система должна быть простой в освоении и эксплуатации пользователями;
- **эффективность**: программное обеспечение должно оптимально использовать системные ресурсы, такие как центральный процессор, оперативная память и хранилище;
- **надёжность**: система должна функционировать без сбоев в течение определённого периода времени;
- **безопасность:** программное обеспечение должно защищать данные и предотвращать несанкционированный доступ.

Инженеры-программисты обеспечивают качество с помощью тестирования программного обеспечения, анализа кода, процессов обеспечения качества и соблюдения стандартов разработки программного обеспечения.

Пример:

Банковская система должна корректно обрабатывать финансовые операции, такие как пополнение счета, снятие средств и переводы. Если программное обеспечение выдаёт неверные расчёты или даёт сбой во время транзакции, это может привести к финансовым потерям и недовольству клиентов. Поэтому для обеспечения высокого качества программного обеспечения необходимы тщательное тестирование и контроль качества.

5.2.2. Обеспечение надёжности и безопасности программного обеспечения

Ещё одна важная задача программной инженерии – обеспечение надёжности и безопасности программных систем. Под надёжностью понимается способность программного обеспечения корректно и стабильно работать в заданных условиях в течение определённого периода времени.

Надёжное программное обеспечение должно:

- работать без частых сбоев;
- быстро восстанавливаться после непредвиденных ошибок;
- сохранять стабильную производительность с течением времени.

Безопасность не менее важна для современных программных систем, поскольку приложения часто работают с конфиденциальными данными, такими как личная информация, финансовые отчёты и конфиденциальная деловая информация. Инженеры-программисты должны внедрять механизмы защиты, чтобы оградить системы от таких угроз, как несанкционированный доступ, утечка данных, вредоносное ПО и кибератаки.

К распространенным методам обеспечения безопасности относятся:

- шифрование данных;
- системы аутентификации и авторизации;
- методы безопасного кодирования;
- регулярное тестирование безопасности и оценка уязвимостей.

Пример:

Платформа для онлайн-покупок должна обеспечивать защиту данных кредитных карт и личной информации клиентов. Такие меры безопасности, как шифрование транзакций и защищенные системы входа, помогают предотвратить кражу конфиденциальных данных хакерами.

5.2.3. Управление крупными и сложными проектами

Современные программные системы часто являются большими и сложными, и разрабатываются командами из множества инженеров. Управление такими проектами требует структурированного планирования, координации и мониторинга.

Разработка программного обеспечения предоставляет различные методологии управления проектами, которые помогают командам эффективно справляться со сложностями. К ним относятся такие модели разработки, как:

- Водопадная модель;
- Гибкая разработка;
- Скрам;
- Методик DevOps.

Благодаря правильному управлению проектами, Разработка программного обеспечения, который помогает обеспечить:

- цели проекта являются четко определенными;
- задачи развития организованы и назначены;
- мониторинг прогресс и контролируемой;
- определены риски и управлять.

Такой структурированный подход предотвращает путаницу, уменьшает количество ошибок при разработке и гарантирует успешное завершение проектов.

Пример:

Для разработки крупной системы планирования ресурсов предприятия (ERP) для международной компании могут потребоваться сотни разработчиков, работающих над различными модулями, такими как финансы, управление персоналом, складской учёт и работа с клиентами. Методологии программной инженерии помогают координировать работу этих команд и интегрировать все модули в единую функциональную систему.

5.2.4. Сокращение сроков и стоимости разработки

Эффективная разработка программного обеспечения направлена на минимизацию времени и затрат при сохранении высокого качества. Разработка программного обеспечения достигается за счет применения систематических процессов, платформ разработки и средств автоматизации.

Несколько методов помогают сократить время и затраты на разработку:

- повторное использование существующих программных компонентов;
- использование платформ разработки и библиотек;
- автоматизированные инструменты тестирования и развертывания;
- эффективное планирование проектов.

Следуя структурированным процессам разработки, организации могут избежать таких распространенных проблем, как задержки проекта, перерасход бюджета и ненужные доработки.

Пример:

Команда разработчиков, создающая мобильное приложение, может использовать существующие фреймворки и готовые библиотеки вместо того, чтобы разрабатывать каждую функцию с нуля. Это значительно сокращает время разработки и общую стоимость проекта.

5.2.5. Повышение обслуживаемости и масштабируемости

Программные системы часто приходится обновлять и расширять после их первого выпуска. Поэтому ещё одна важная задача программной инженерии – обеспечить удобство сопровождения и масштабируемость.

Под удобством сопровождения понимается то, насколько легко можно модифицировать программное обеспечение для исправления ошибок, повышения производительности или добавления новых функций.

В хорошо спроектированном программном обеспечении используются:

- модульная архитектура;
- четкая документация;
- хорошо структурированный и читаемый код.

Масштабируемость – это способность системы справляться с растущими рабочими нагрузками, количеством пользователей или объемами данных без снижения производительности.

Инженеры-программисты проектируют системы с учетом масштабируемости, используя такие методы, как:

- распределенные вычисления;
- облачная инфраструктура;
- архитектура микросервисов.

Пример:

Платформа социальных сетей, изначально рассчитанная на несколько тысяч пользователей, со временем может потребоваться для поддержки миллионов пользователей. Масштабируемая системная архитектура гарантирует, что платформа сможет справиться с таким ростом без сбоев и замедления работы.

5.2.6. Поддержка командной работы и сотрудничества между разработчиками

Разработка программного обеспечения редко является индивидуальной задачей. Как правило, над ней работают команды разработчиков, дизайнеров, тестировщиков и менеджеров проектов. Поэтому в программной инженерии особое внимание уделяется эффективному взаимодействию и коммуникации между членами команды.

Взаимодействие обеспечивается за счёт использования различных инструментов и методов, в том числе:

- системы контроля версий (например, Git);
- инструментов управления проектами (например, Jira, Trello);
- системы непрерывной интеграции и развёртывания;
- методов ревью кода.

Эти инструменты позволяют нескольким разработчикам одновременно работать над разными частями программного обеспечения, сохраняя согласованность и избегая конфликтов.

Эффективное взаимодействие также повышает:

- обмен знаниями;
- качество кода;
- эффективность разработки;
- прозрачность проекта.

Пример:

При разработке крупного веб-приложения фронтенд-разработчики, бэкенд-разработчики и инженеры по базам данных должны координировать свою работу. Использование систем контроля версий и платформ для совместной работы позволяет всем членам команды эффективно вносить свой вклад, не мешая друг другу.

5.3. История и эволюция программной инженерии

За последние несколько десятилетий сфера программной инженерии претерпела значительные изменения по мере того, как компьютерные системы становились все более мощными, а программные приложения – все более сложными. На заре компьютерной эры разработка программного обеспечения была относительно простой и заключалась в написании небольших программ отдельными программистами. Однако по мере того, как организации все больше зависели от компьютерных систем, необходимость в систематических и структурированных подходах к разработке программного обеспечения становилась все более очевидной.

Эволюция программной инженерии во многом была обусловлена трудностями, с которыми приходилось сталкиваться при разработке крупных и сложных программных систем. Эти трудности в конечном итоге привели к появлению современных методологий, инструментов и лучших практик программной инженерии.

5.3.1. Ранние вычисления и разработка программного обеспечения

В первые годы развития вычислительной техники, особенно в 1940-х и 1950-х годах, разработка программного обеспечения сильно отличалась от современной структурированной инженерной дисциплины. В то время компьютеры использовались в основном для научных вычислений, военных операций и решения базовых задач по обработке данных.

В тот период программное обеспечение обычно писалось непосредственно на машинном языке, состоящем из двоичных инструкций (состоящих из нулей и единиц), которые компьютерное оборудование могло выполнять напрямую. Написание программ на машинном языке было чрезвычайно сложным и трудоёмким процессом, поскольку программистам приходилось досконально разбираться во всех внутренних механизмах работы компьютера.

Поскольку компьютерные системы были относительно простыми и имели ограниченные возможности, программы, разрабатываемые в тот период, были небольшими и понятными. Большую часть программного обеспечения писали отдельные программисты, а не большие команды разработчиков.

5.3.2. Особенности ранней разработки программного обеспечения

Ранний этап разработки программного обеспечения имел несколько отличительных особенностей:

- **Прямое программирование на машинном уровне:** программисты писали инструкции непосредственно на машинном языке или на очень низкоуровневом языке ассемблера. Это требовало глубокого знания аппаратной архитектуры.
- **Небольшой размер программ:** программы, как правило, были короткими и предназначались для выполнения конкретных задач, таких как математические вычисления или обработка данных.

- **Отсутствие структурированных методов разработки:** не существовало формальных методологий разработки, стандартов программной инженерии или процессов систематического планирования.
- **Минимальная документация:** большинство программ практически не сопровождались документацией, из-за чего другим разработчикам было сложно понять или изменить код.
- **Разработка с привязкой к аппаратному обеспечению:** Программное обеспечение было тесно привязано к конкретным аппаратным системам, то есть программы, написанные для одного компьютера, часто не работали на другом.

5.3.3. Кризис программного обеспечения

К 1960-м годам компьютеры стали более мощными и все чаще использовались в государственных, коммерческих и промышленных целях. В результате программные системы стали значительно увеличиваться в размерах и усложняться.

Однако методы разработки программного обеспечения развивались не так быстро, как аппаратные технологии. Многие организации пытались разрабатывать крупные программные системы, используя те же неформальные подходы к программированию, что и при создании небольших программ.

Это привело к ряду серьёзных проблем, в том числе:

- **перерасход бюджета** – на многие программные проекты уходило гораздо больше средств, чем планировалось изначально;
- **срыв сроков** – на реализацию проектов часто уходило гораздо больше времени, чем ожидалось;
- **низкое качество программного обеспечения** – многие системы содержали множество багов и ошибок;
- **сбои в работе** – некоторые программные системы полностью выходили из строя или работали не так, как предполагалось.

Эти широко распространённые проблемы стали известны как «кризис программного обеспечения».

5.3.4. Причины кризиса программного обеспечения

Кризису программного обеспечения способствовало несколько факторов:

- **рост сложности программного обеспечения:** программные системы становились все более масштабными и сложными, что затрудняло их проектирование, внедрение и поддержку;
- **отсутствие методологий разработки:** не существовало стандартизированных процессов разработки и инженерных принципов, которыми можно было бы руководствоваться при создании программного обеспечения;
- **некачественный анализ требований:** многие проекты начинались без чётко сформулированных требований, что приводило к путанице и частым изменениям в процессе разработки;
- **недостаточное тестирование:** процессы тестирования зачастую были недостаточно эффективными, что приводило к появлению программного обеспечения с многочисленными ошибками;
- **ограниченные возможности для совместной работы:** коммуникация и координация между командами разработчиков были затруднены из-за отсутствия передовых инструментов управления проектами и совместной работы.

5.3.5. Влияние кризиса в сфере разработки программного обеспечения

Кризис в сфере разработки программного обеспечения выявил острую необходимость в более дисциплинированном и системном подходе к разработке. В результате исследователи и специалисты отрасли начали рассматривать разработку программного обеспечения как инженерную дисциплину, аналогичную традиционным областям, таким как гражданское или машиностроительное проектирование.

Этот сдвиг в конечном итоге привёл к официальному становлению такой области, как программная инженерия, в рамках которой появились структурированные методологии разработки, методы обеспечения качества, подходы к управлению проектами и стандарты документирования.

5.4. Зарождение программной инженерии

Термин «программная инженерия» был официально введён в 1968 году на конференции по программной инженерии NATO, которая была организована для решения растущих проблем в сфере разработки программного обеспечения. В то время многие крупномасштабные программные проекты терпели крах из-за плохого планирования, нечетко сформулированных требований и отсутствия структурированных методов разработки. Эти широко распространённые проблемы в совокупности получили название «кризис в сфере разработки программного обеспечения».

На конференции NATO собрались ученые-компьютерщики, инженеры и отраслевые эксперты, чтобы обсудить возможные пути решения этих проблем. В ходе конференции участники предложили рассматривать разработку программного обеспечения как формальную инженерную дисциплину, аналогичную таким областям, как гражданское или электротехническое проектирование.

Основная цель заключалась в том, чтобы внедрить систематические методы, стандарты и лучшие практики разработки программных систем. Эта новая дисциплина была призвана повысить качество программного обеспечения, снизить риски при разработке и сделать крупные программные проекты более управляемыми.

5.4.1. Ключевые цели, представленные на конференции

Внедрение программной инженерии подчеркнуло важность нескольких важных принципов:

- **систематизированные процессы разработки:** разработка программного обеспечения должна осуществляться в соответствии со структурированными методологиями и чётко определёнными этапами, а не с использованием неформальных подходов к программированию;
- **инженерные принципы:** программное обеспечение должно разрабатываться с использованием таких инженерных принципов, как планирование, моделирование, тестирование и контроль качества;

- **методы управления проектами:** для контроля сроков, бюджета и координации работы команды необходимо применять надлежащие методы управления проектами;
- **стандартизация и документирование:** программные системы должны быть тщательно задокументированы, чтобы облегчить их понимание, обслуживание и взаимодействие между разработчиками.

Признание программной инженерии как отдельной дисциплины положило начало современным практикам разработки программного обеспечения, что привело к созданию моделей разработки, стандартов программирования и методологий программной инженерии, которые широко используются сегодня.

5.5. Модели разработки программного обеспечения

Модели разработки программного обеспечения описывают этапы или процессы, которые необходимо пройти для создания программного обеспечения. Для разработки программного обеспечения используются различные модели, такие как:

5.5.1. Модель «Водопад»

Модель «Водопад» – это линейный последовательный подход к разработке, при котором необходимо завершить один этап, прежде чем переходить к следующему.

Преимущества

- простой и понятный;
- хорошо структурированная документация.

Недостатки

- сложно адаптировать изменения требований.

5.5.2. Модель гибкой разработки (Agile Model)

Agile делает упор на итеративную разработку, совместную работу и гибкость.

Ключевые принципы

- взаимодействие с заказчиком;
- быстрые циклы разработки;
- постоянное совершенствование.

Agile позволяет разработчикам быстро адаптироваться к меняющимся требованиям.

5.5.3. Спиральная модель

Спиральная модель объединяет проектирование, разработку и анализ рисков в циклический процесс.

Каждая фаза спирали включает в себя:

1. Планирование.
2. Анализ рисков.
3. Проектирование.
4. Оценку.

Эта модель особенно полезна при работе над крупными проектами с высоким уровнем риска.

5.5.4. Итеративная модель

В итеративной модели программное обеспечение разрабатывается в несколько этапов, на каждом из которых новая версия улучшает предыдущую. Такой подход позволяет разработчикам постепенно совершенствовать систему и учитывать обратную связь.

5.5.5. V-образная модель

V-модель предполагает верификацию и валидацию на протяжении всего процесса разработки. Каждому этапу разработки соответствует этап тестирования, что обеспечивает высокое качество программного обеспечения.

Задание № 5

1. Что такое программная инженерия? Какие задачи решает программное обеспечение?

Чтобы решить задачу, изучите и разберитесь в тексте подзаголовка № 5.1 и подзаголовка № 5.2.

2. Что такое программный кризис? Как он помогает разрабатывать качественное программное обеспечение?

Решение этой задачи приведено в подзаголовке 5.3.2. Для решения проблемы необходимо понять, что именно привело к кризису в сфере программного обеспечения и как выход из этого кризиса повлиял на развитие программной инженерии.

3. Какие модели используются при разработке программного обеспечения? Чем каждая модель отличается от других? Каковы преимущества и недостатки каждой из этих моделей?

Решение этой задачи смотрите в подзаголовке № 5.5 данной главы. Необходимо понять, какая из этих моделей подходит для разработки конкретного типа программного обеспечения. Размер и функциональные требования к программному обеспечению играют важную роль при выборе конкретной модели для разработки.

4. Каковы характеристики хорошего программного обеспечения?

Для решения задачи необходимо полностью изучить эту главу, особенно первые две подзаголовке 5.1 и 5.2.

ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

Было замечено, что изучение основ информатики даёт необходимые навыки решения проблем, которые помогают ориентироваться в современном цифровом мире. Оно учит разбивать большие проблемы на маленькие, легко решаемые задачи. Кроме того, оно помогает понять, как работают программное обеспечение, интернет и данные. В современном мире это играет важную роль для карьерного роста. Базовые знания в области технологий расширяют возможности трудоустройства практически в любой современной отрасли. Благодаря технологиям люди лучше осведомлены о своих правах и о том, как защитить свою конфиденциальность. Это позволяет нам принимать взвешенные решения в вопросах цифровой конфиденциальности, безопасности и использования новых технологий, таких как искусственный интеллект, в повседневной жизни. Основы информатики нужны не только инженерам-программистам. Это важнейший жизненный навык, который помогает каждому адаптироваться, творить и решать проблемы в технологичном обществе.

Компьютеры являются частью нашей повседневной жизни. Понимание их устройства и принципов работы помогает нам использовать их более эффективно. Для студентов, изучающих информатику и информационные технологии, вычислительная техника является, пожалуй, самым фундаментальным предметом в области компьютерных наук. Без вычислительной техники не было бы и компьютерных наук.

Многие повседневные действия студентов, будь то просмотр веб-страниц, отправка электронных писем или написание документов, связаны с основами вычислительной техники.

В этом учебно-методическом пособии студенты знакомятся с современным компьютерным оборудованием, в том числе с принципами работы микропроцессоров и методами программирования. Студенты также узнают об истории компьютерных систем, развитии вычислительной техники, идеях, лежащих в основе её развития, и факторах, которые со временем способствовали совершенствованию компьютерных систем.

В работе также рассматриваются фундаментальные технологические принципы, лежащие в основе компьютерной системы и обеспечивающие её работоспособность.

Это учебно-методическое пособие представляет собой комплексное и обобщённое исследование компьютеров и вычислительной техники, которого нет в других научных работах и методологиях в области вычислительной техники. Данная методология основана на обширных исследованиях и отражает современные концепции вычислительной техники. В ней рассматриваются современные проблемы и угрозы для производительности компьютерных систем, а также предлагаются способы их решения путём реорганизации компонентов компьютерной архитектуры и планирования времени между их выполнением, например, между работой памяти, регистров, стека, арифметико-логического устройства и т. д.

Эта работа также дает студентам возможность провести исследование и предложить новое решение для устранения проблем, возникающих при разработке программного обеспечения. В исследовании рассматриваются некоторые распространенные методы, которые часто используются для решения этих проблем.

Это учебно-методическое пособие содержит достаточно информации о будущем вычислительной техники и компьютерных систем.

СПИСОК ЛИТЕРАТУРЫ

1. Sipser M. Introduction to the theory of computation. 3rd ed. Boston: Cengage Learning, 2012.

2. Patterson D. A., Hennessy J. L. Computer organization and design RISC-V edition: The hardware/software interface. 2nd ed. Morgan Kaufmann, 2021.

3. Russell S., Norvig P. Artificial intelligence: A modern approach. 4th ed. Pearson, 2021.

4. Poole D., Mackworth A. Artificial intelligence: Foundations o computational agents. 2nd ed. Cambridge: Cambridge University Press, 2017.

5. Abbasi M. M. Computer Architecture: A Detailed Study: Software and Hardware aspects of Computer Architecture. LAP LAMBERT Academic Publishing, 2015. ISBN 978-3659790546.

6. Silberschatz A., Galvin P. B., Gagne G. Operating System Concepts. 2018. ISBN 978-1118063330.

7. Tanenbaum A. S., Bos H. Modern Operating Systems. 4th ed. 2014. ISBN 978-0133591620.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 3
ГЛАВА 1. Компьютер и компьютерные науки 5
1.1. Что такое компьютер? 5
1.2. Эволюция компьютера с течением времени 7
1.3. Типы компьютеров и их компонентов 9
1.4. Применение компьютеров и вычислительной техники в различных сферах 12
1.5. Преимущества использования компьютеров и вычислительной техники 13
1.6. Недостатки использования компьютеров и вычислительной техники 17
Задание № 1 21
ГЛАВА 2. Алгоритмы и структуры данных 22
2.1. Что такое алгоритм? 24
2.2. Типы алгоритмов в зависимости от их функциональности 25
2.3. Характеристики алгоритма 26
2.4. Что такое структура данных? 30
2.5. Важность алгоритмов и структур данных 33
Задание № 2 36
ГЛАВА 3. Компьютерные сети и Интернет 37
3.1. Компьютерная сеть 37
3.2. Интернет 37
3.3. Типы сетей 38
3.4. Использование Интернета 41
3.5. Будущие тенденции развития сетей 45
Задание № 3 52
ГЛАВА 4. Операционная система 53
4.1. История эволюции операционных систем 53
4.2. Основные функции операционных систем 57
4.3. Управление процессами и потоками 61
4.4. Современные тенденции в области операционных систем 62
Задание № 4 64

ГЛАВА 5. Программная инженерия в современной разработке программного обеспечения 65
5.1. Что такое программная инженерия? 65
5.2. Цели и задачи программной инженерии 65
5.3. История и эволюция программной инженерии 71
5.4. Зарождение программной инженерии 75
5.5. Модели разработки программного обеспечения 76
Задание № 5 78
ЗАКЛЮЧЕНИЕ И ВЫВОДЫ 79
СПИСОК ЛИТЕРАТУРЫ 81

Учебное издание

Аббаси Мохсин Маншад
Бельтюков Анатолий Петрович

**ОСНОВЫ КОМПЬЮТЕРНЫХ НАУК
И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

Учебно-методическое пособие

Авторская редакция

Подписано в печать 12.10.2026. Формат 60x84 1/16.
Усл. печ. л. 4,88. Уч. изд. л. 4,16.
Тираж 100 экз. Заказ № 1466

Издательский центр «Удмуртский университет»
426004, Ижевск, Ломоносова, 4Б, каб. 021
Тел./факс: + 7 (3412) 263-751, E-mail: editorial@udsu.ru

Типография Издательского центра «Удмуртский университет»
426034, Ижевск, ул. Университетская, 1, корп. 2.
Тел. 68-57-18