MIPT aspirantura

Anna K. Zvereva

August 2024

1 1. Пределы последовательности. Критерий Коши. Существование предела у монотонно возрастающей ограниченной сверху последовательности. Теорема Больцано-Вейерштрасса о существовании сходящейся подпоследовательности у ограниченной последовательности.

1.1 Предел последовательности

Пусть $\{a_n\}$ — числовая последовательность. Число A называется пределом последовательности $\{a_n\}$, если для любого $\varepsilon > 0$ существует такое число N, что для всех $n > N$ выполняется неравенство:

$$|a_n - A| < \varepsilon.$$

Обозначение:

$$\lim_{n \to \infty} a_n = A.$$

1.2 Критерий Коши

Последовательность $\{a_n\}$ сходится тогда и только тогда, когда для любого $\varepsilon > 0$ существует число N такое, что для всех $m, n > N$ выполняется:

$$|a_n - a_m| < \varepsilon.$$

Это означает, что последовательность $\{a_n\}$ является фундаментальной, если она сходится.

1.3 Существование предела у монотонно возрастающей ограниченной сверху последовательности

Если последовательность $\{a_n\}$ монотонно возрастает и ограничена сверху, то она сходится. Это утверждение следует из принципа полноты вещественных чисел. Формально:

$$\text{Если } a_1 \leq a_2 \leq a_3 \leq \ldots \text{ и } \sup\{a_n\} < \infty, \text{ то } \lim_{n\to\infty} a_n \text{ существует.}$$

1.4 Теорема Больцано-Вейерштрасса

Теорема Больцано-Вейерштрасса утверждает, что из любой ограниченной последовательности $\{a_n\}$ можно выделить сходящуюся подпоследовательность. Формально:

Если последовательность $\{a_n\}$ ограничена, то существует подпоследовательность $\{a_{n_k}\}$ такая, что $\lim_{k\to\infty}$

2 2. Числовые ряды. Критерий Коши. Признаки сходимости (признаки сравнения, признак Даламбера, признак Лейбница, признак Дирихле).

2.1 Числовые ряды

Пусть $\{a_n\}$ — последовательность чисел. Числовым рядом называется выражение:

$$\sum_{n=1}^{\infty} a_n = a_1 + a_2 + a_3 + \ldots$$

Рассматриваем частичные суммы ряда:

$$S_N = \sum_{n=1}^{N} a_n$$

Если существует конечный предел последовательности частичных сумм S_N при $N \to \infty$, то говорят, что ряд сходится. В противном случае ряд расходится.

2.2 Критерий Коши для рядов

Ряд $\sum_{n=1}^{\infty} a_n$ сходится тогда и только тогда, когда для любого $\varepsilon > 0$ существует число N такое, что для всех $m > n > N$ выполняется:

$$|a_{n+1} + a_{n+2} + \cdots + a_m| < \varepsilon.$$

2.3 Признак сравнения

Пусть даны два ряда $\sum a_n$ и $\sum b_n$, где $a_n \geq 0$ и $b_n \geq 0$ для всех n. Если существует постоянная $C > 0$, такая что для всех n выполнено неравенство $a_n \leq Cb_n$, то:

- Если ряд $\sum b_n$ сходится, то сходится и ряд $\sum a_n$.
- Если ряд $\sum a_n$ расходится, то расходится и ряд $\sum b_n$.

2.4 Признак Даламбера

Для ряда $\sum a_n$ рассмотрим предел:

$$\lim_{n\to\infty} \frac{a_{n+1}}{a_n} = L.$$

- Если $L < 1$, то ряд сходится.
- Если $L > 1$, то ряд расходится.
- Если $L = 1$, то признак Даламбера не даёт информации.

2.5 Признак Лейбница

Пусть a_n — убывающая последовательность положительных чисел такая, что $\lim_{n\to\infty} a_n = 0$. Тогда знакочередующийся ряд:

$$\sum (-1)^n a_n$$

сходится.

2.6 Признак Дирихле

Пусть a_n — последовательность таких чисел, что частичные суммы $A_N = \sum_{n=1}^{N} a_n$ ограничены, а $\{b_n\}$ — монотонная последовательность чисел, стремящаяся к нулю. Тогда ряд:

$$\sum a_n b_n$$

сходится.

3 3. Предел функции. Непрерывные функции. Свойства непрерывных функций на отрезке.

3.1 Предел функции

Пусть функция $f(x)$ определена в некоторой окрестности точки x_0, за исключением, возможно, самой точки x_0. Число A называется пределом функции $f(x)$ при $x \to x_0$, если для любого $\varepsilon > 0$ существует такое $\delta > 0$, что

для всех x, удовлетворяющих условию $0 < |x - x_0| < \delta$, выполняется неравенство:

$$|f(x) - A| < \varepsilon.$$

Обозначение:

$$\lim_{x \to x_0} f(x) = A.$$

3.2 Непрерывные функции

Функция $f(x)$ называется непрерывной в точке x_0, если:

$$\lim_{x \to x_0} f(x) = f(x_0).$$

Это означает, что предел функции в точке совпадает со значением функции в этой точке.

3.3 Свойства непрерывных функций на отрезке

Пусть функция $f(x)$ непрерывна на отрезке $[a, b]$.

3.3.1 Теорема Вейерштрасса об ограниченности

Любая функция, непрерывная на замкнутом отрезке $[a, b]$, ограничена, то есть существует такое число $M > 0$, что для всех $x \in [a, b]$ выполнено неравенство:

$$|f(x)| \leq M.$$

3.3.2 Теорема Вейерштрасса о достижимости верхней и нижней грани

Любая функция, непрерывная на замкнутом отрезке $[a, b]$, достигает на этом отрезке своих максимального и минимального значений. То есть существуют такие точки $c, d \in [a, b]$, что:

$$f(c) = \max_{x \in [a,b]} f(x), \quad f(d) = \min_{x \in [a,b]} f(x).$$

3.3.3 Теорема Коши о промежуточных значениях

Пусть функция $f(x)$ непрерывна на отрезке $[a, b]$ и $f(a) \cdot f(b) < 0$. Тогда существует точка $c \in (a, b)$, такая что $f(c) = 0$. Это означает, что непрерывная функция на отрезке принимает все промежуточные значения между $f(a)$ и $f(b)$.

3.4 Обобщение на многомерный случай

Для функций нескольких переменных аналогом понятия непрерывности является непрерывность по каждой переменной в отдельности. Пусть $f(x_1, x_2, \ldots, x_n)$ определена в окрестности точки $(x_1^0, x_2^0, \ldots, x_n^0)$. Функция f непрерывна в точке $(x_1^0, x_2^0, \ldots, x_n^0)$, если для любого $\varepsilon > 0$ существует $\delta > 0$, такое что для всех $(x_1, x_2, \ldots, x_n)$, удовлетворяющих условию:

$$\sqrt{(x_1 - x_1^0)^2 + (x_2 - x_2^0)^2 + \cdots + (x_n - x_n^0)^2} < \delta,$$

выполняется:

$$|f(x_1, x_2, \ldots, x_n) - f(x_1^0, x_2^0, \ldots, x_n^0)| < \varepsilon.$$

3.5 Односторонние пределы монотонных функций

Пусть функция $f(x)$ монотонна на интервале (a, b). Тогда в каждой точке этого интервала существуют конечные односторонние пределы:

$$\lim_{x \to c-} f(x) \quad \text{и} \quad \lim_{x \to c+} f(x).$$

3.6 Теорема о непрерывности обратной функции

Пусть $f(x)$ монотонна и непрерывна на отрезке $[a, b]$. Тогда обратная функция $f^{-1}(y)$ также непрерывна на отрезке $[f(a), f(b)]$.

3.7 Равномерная непрерывность

Функция $f(x)$ называется равномерно непрерывной на множестве D, если для любого $\varepsilon > 0$ существует $\delta > 0$, такое что для всех $x_1, x_2 \in D$, удовлетворяющих условию $|x_1 - x_2| < \delta$, выполняется неравенство:

$$|f(x_1) - f(x_2)| < \varepsilon.$$

В отличие от обычной непрерывности, значение δ зависит только от ε, но не от точки x.

4 4. Дифференцируемые функции одной и нескольких переменных. Производные и дифференциал. Формула Тейлора для функций (одной и нескольких переменных). Ряды Тейлора. Элементарные функции. Теорема о неявных функциях.

4.1 Дифференцируемые функции одной переменной

Пусть $f(x)$ определена в некоторой окрестности точки x_0. Производной функции $f(x)$ в точке x_0 называется предел отношения приращения функции к приращению аргумента при стремлении приращения аргумента к нулю:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

Функция называется дифференцируемой в точке x_0, если существует её производная в этой точке.

4.2 Дифференцируемые функции нескольких переменных

Пусть $f(x_1, x_2, \ldots, x_n)$ — функция нескольких переменных, определенная в некоторой окрестности точки $(x_1^0, x_2^0, \ldots, x_n^0)$. Частной производной функции f по переменной x_i называется предел:

$$\frac{\partial f}{\partial x_i}(x_1^0, x_2^0, \ldots, x_n^0) = \lim_{\Delta x_i \to 0} \frac{f(x_1^0, \ldots, x_i^0 + \Delta x_i, \ldots, x_n^0) - f(x_1^0, \ldots, x_n^0)}{\Delta x_i}.$$

Функция называется дифференцируемой в точке, если существуют все её частные производные и линейная комбинация этих производных задаёт линейное приближение функции в данной точке.

4.3 Дифференциал функции

Дифференциалом функции одной переменной $f(x)$ называется выражение:

$$df = f'(x)\,dx.$$

Для функции нескольких переменных $f(x_1, x_2, \ldots, x_n)$ дифференциал первого порядка выражается как:

$$df = \frac{\partial f}{\partial x_1}\,dx_1 + \frac{\partial f}{\partial x_2}\,dx_2 + \cdots + \frac{\partial f}{\partial x_n}\,dx_n.$$

4.4 Формула Тейлора для функций одной переменной

Формула Тейлора для функции $f(x)$, дифференцируемой n раз в точке x_0, имеет вид:

$$f(x) = f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+o((x-x_0)^n).$$

4.5 Формула Тейлора для функций нескольких переменных

Формула Тейлора для функции $f(x_1, x_2, \ldots, x_n)$ в окрестности точки $(x_1^0, x_2^0, \ldots, x_n^0)$ имеет вид:

$$f(x_1, x_2, \ldots, x_n) = f(x_1^0, x_2^0, \ldots, x_n^0)+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}(x_i-x_i^0)+\frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 f}{\partial x_i \partial x_j}(x_i-x_i^0)(x_j-x_j^0)+\ldots$$

4.6 Ряды Тейлора

Ряд Тейлора для функции $f(x)$ — это бесконечный ряд, представляющий функцию в виде:

$$f(x) = \sum_{n=0}^{\infty}\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n,$$

где $f^{(n)}(x_0)$ — n-я производная функции f в точке x_0.

4.7 Элементарные функции

Элементарные функции — это функции, которые могут быть выражены с помощью конечного числа операций сложения, умножения, деления, возведения в степень и извлечения корня, а также применения тригонометрических, показательных и логарифмических функций. Примеры элементарных функций:

$$e^x,\ \log x,\ \sin x,\ \cos x,\ \tan x,\ x^n\ (\text{при } n \in R).$$

4.8 Теорема о неявных функциях (без доказательства)

Пусть $F(x, y)$ — функция, дифференцируемая в окрестности точки (x_0, y_0), причём $F(x_0, y_0) = 0$ и $\frac{\partial F}{\partial y}(x_0, y_0) \neq 0$. Тогда существует такая функция $y = f(x)$, определённая в некоторой окрестности точки x_0, что $F(x, f(x)) = 0$ и $f(x_0) = y_0$. Более того, функция $f(x)$ дифференцируема, и её производная выражается как:

$$f'(x) = -\frac{\frac{\partial F}{\partial x}(x, f(x))}{\frac{\partial F}{\partial y}(x, f(x))}.$$

5 5. Вычисление пределов с помощью формулы Тейлора. Достаточные условия монотонности дифференцируемой функции. Выпуклые функции. Достаточные условия выпуклости функции два раза дифференцируемой на интервале. Асимптоты.

5.1 Вычисление пределов с помощью формулы Тейлора

Формула Тейлора позволяет вычислять пределы путём разложения функции в ряд Тейлора около точки, в которой вычисляется предел. Пусть функция $f(x)$ имеет разложение в ряд Тейлора около точки x_0:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots$$

Если при $x \to x_0$ старшие члены ряда дают основную вкладку в изменение функции, то можно отбросить малые члены и использовать приближённую формулу для нахождения предела.

Пример:

$$\lim_{x \to 0} \frac{\sin x}{x} = \lim_{x \to 0} \frac{x - \frac{x^3}{3!} + O(x^5)}{x} = 1.$$

5.2 Достаточные условия монотонности дифференцируемой функции

Пусть $f(x)$ — дифференцируемая функция на интервале (a, b). Тогда:

- Если $f'(x) > 0$ на всём интервале (a, b), то функция $f(x)$ строго возрастает на этом интервале.
- Если $f'(x) < 0$ на всём интервале (a, b), то функция $f(x)$ строго убывает на этом интервале.

5.3 Выпуклые функции

Функция $f(x)$ называется выпуклой на интервале (a, b), если для любых $x_1, x_2 \in (a, b)$ и любого $\lambda \in [0, 1]$ выполняется неравенство:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2).$$

Если выполняется обратное неравенство, то функция называется вогнутой.

5.4 Достаточные условия выпуклости функции два раза дифференцируемой на интервале

Пусть $f(x)$ дважды дифференцируема на интервале (a, b). Тогда:

- Если $f''(x) \geq 0$ для всех $x \in (a, b)$, то функция $f(x)$ выпукла на интервале (a, b).
- Если $f''(x) > 0$ для всех $x \in (a, b)$, то функция строго выпукла на интервале (a, b).

5.5 Асимптоты

Асимптоты функции — это прямые, к которым график функции стремится при $x \to \infty$ или $x \to -\infty$.

- **Горизонтальная асимптота**: Прямая $y = A$ называется горизонтальной асимптотой функции $f(x)$, если:

$$\lim_{x \to \infty} f(x) = A \quad \text{или} \quad \lim_{x \to -\infty} f(x) = A.$$

- **Вертикальная асимптота**: Прямая $x = a$ называется вертикальной асимптотой функции $f(x)$, если:

$$\lim_{x \to a-} f(x) = \pm\infty \quad \text{или} \quad \lim_{x \to a+} f(x) = \pm\infty.$$

- **Наклонная асимптота**: Прямая $y = kx + b$ называется наклонной асимптотой функции $f(x)$, если:

$$\lim_{x \to \infty} (f(x) - (kx + b)) = 0.$$

6 6. Экстремумы функций одной и нескольких переменных. Необходимые условия экстремума. Достаточные условия экстремума для дифференцируемых функций.

6.1 Экстремумы функций одной переменной

Точка x_0 называется точкой экстремума функции $f(x)$, если в этой точке функция принимает либо локальный максимум, либо локальный минимум. Точка x_0 является:

- точкой локального максимума, если существует такая окрестность $U(x_0)$, что для всех $x \in U(x_0)$ выполняется $f(x) \leq f(x_0)$;
- точкой локального минимума, если существует такая окрестность $U(x_0)$, что для всех $x \in U(x_0)$ выполняется $f(x) \geq f(x_0)$.

6.2 Необходимое условие экстремума для функции одной переменной

Пусть функция $f(x)$ дифференцируема в точке x_0. Если x_0 — точка локального экстремума, то производная функции в этой точке равна нулю:

$$f'(x_0) = 0.$$

Это условие является необходимым, но не достаточным для существования экстремума.

6.3 Достаточное условие экстремума для функции одной переменной

Пусть $f(x)$ дважды дифференцируема в точке x_0, и $f'(x_0) = 0$. Тогда:

- Если $f''(x_0) > 0$, то x_0 — точка локального минимума.
- Если $f''(x_0) < 0$, то x_0 — точка локального максимума.
- Если $f''(x_0) = 0$, то требуется дальнейшее исследование (например, через знак производных более высокого порядка).

6.4 Экстремумы функций нескольких переменных

Пусть $f(x_1, x_2, \ldots, x_n)$ — функция нескольких переменных. Точка $(x_1^0, x_2^0, \ldots, x_n^0)$ называется точкой локального экстремума, если:

- это точка локального максимума, если существует такая окрестность $U(x_1^0, x_2^0, \ldots, x_n^0)$, что для всех точек этой окрестности выполняется $f(x_1, x_2, \ldots, x_n) \leq f(x_1^0, x_2^0, \ldots, x_n^0)$;
- это точка локального минимума, если существует такая окрестность $U(x_1^0, x_2^0, \ldots, x_n^0)$, что для всех точек этой окрестности выполняется $f(x_1, x_2, \ldots, x_n) \geq f(x_1^0, x_2^0, \ldots, x_n^0)$.

6.5 Необходимое условие экстремума для функций нескольких переменных

Если функция $f(x_1, x_2, \ldots, x_n)$ дифференцируема в точке $(x_1^0, x_2^0, \ldots, x_n^0)$ и эта точка является точкой экстремума, то все частные производные в этой точке равны нулю:

$$\frac{\partial f}{\partial x_1}(x_1^0, x_2^0, \ldots, x_n^0) = 0, \quad \frac{\partial f}{\partial x_2}(x_1^0, x_2^0, \ldots, x_n^0) = 0, \quad \ldots, \quad \frac{\partial f}{\partial x_n}(x_1^0, x_2^0, \ldots, x_n^0) = 0.$$

Это условие является необходимым, но не достаточным для существования экстремума.

6.6 Достаточные условия экстремума для функций нескольких переменных

Пусть функция $f(x_1, x_2, \ldots, x_n)$ дважды дифференцируема, и точка $(x_1^0, x_2^0, \ldots, x_n^0)$ удовлетворяет необходимому условию экстремума (все частные производные равны нулю). Тогда для определения характера этой точки исследуют знаки вторых производных с помощью матрицы Гессе H, элементы которой — это вторые частные производные функции:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

- Если матрица Гессе положительно определена в точке $(x_1^0, x_2^0, \ldots, x_n^0)$, то эта точка является точкой локального минимума.
- Если матрица Гессе отрицательно определена, то это точка локального максимума.
- Если матрица Гессе не определена, то характер точки остаётся неопределённым.

7 7. Интеграл Римана. Необходимые и достаточные условия интегрируемости функции по Риману. Теорема о среднем. Первообразная, формула Лейбница-Ньютона. Формула интегрирования по частям. Несобственные интегралы. Признаки сходимости несобственных интегралов. Признак Дирихле.

7.1 Интеграл Римана

Интегралом Римана функции $f(x)$ на отрезке $[a, b]$ называется предел интегральных сумм при стремлении разбиения Δx к нулю:

$$\int_a^b f(x)\,dx = \lim_{\Delta x \to 0} \sum_{i=1}^{n} f(\xi_i)\Delta x_i,$$

где разбиение $[a, b]$ на n частей даёт отрезки длиной Δx_i, и ξ_i — это произвольная точка внутри i-го отрезка.

7.2 Необходимые и достаточные условия интегрируемости функции по Риману

Функция $f(x)$ интегрируема по Риману на отрезке $[a, b]$, если она ограничена на этом отрезке и множество точек её разрыва имеет меру ноль (то есть если $f(x)$ непрерывна почти всюду). Это необходимое и достаточное условие интегрируемости.

7.3 Теорема о среднем

Если функция $f(x)$ непрерывна на отрезке $[a, b]$, то существует такая точка $c \in [a, b]$, что:

$$\int_a^b f(x)\,dx = f(c)(b - a).$$

Это означает, что значение интеграла равно значению функции в некоторой точке, умноженному на длину отрезка.

7.4 Первообразная и формула Лейбница-Ньютона

Функция $F(x)$ называется первообразной функции $f(x)$ на отрезке $[a, b]$, если для всех $x \in [a, b]$ выполняется $F'(x) = f(x)$.

Формула Лейбница-Ньютона выражает значение определённого интеграла через первообразную:

$$\int_a^b f(x)\,dx = F(b) - F(a),$$

где $F(x)$ — первообразная функции $f(x)$.

7.5 Формула интегрирования по частям

Если функции $u(x)$ и $v(x)$ дифференцируемы на отрезке $[a, b]$, то справедлива формула интегрирования по частям:

$$\int_a^b u(x)v'(x)\,dx = [u(x)v(x)]_a^b - \int_a^b u'(x)v(x)\,dx.$$

7.6 Несобственные интегралы

Несобственные интегралы возникают, когда пределы интегрирования бесконечны или подынтегральная функция имеет разрыв в пределах интегрирования.

7.6.1 Несобственные интегралы с бесконечными пределами

Несобственный интеграл на бесконечном промежутке $[a, \infty)$ определяется как предел:

$$\int_a^\infty f(x)\,dx = \lim_{b\to\infty} \int_a^b f(x)\,dx.$$

Интеграл сходится, если этот предел существует и конечен.

7.6.2 Несобственные интегралы с разрывами

Если функция $f(x)$ имеет разрыв в точке $c \in [a, b]$, то несобственный интеграл определяется как предел:

$$\int_a^b f(x)\,dx = \lim_{\epsilon\to 0} \left(\int_a^{c-\epsilon} f(x)\,dx + \int_{c+\epsilon}^b f(x)\,dx \right).$$

Интеграл сходится, если этот предел существует и конечен.

7.7 Признаки сходимости несобственных интегралов

7.7.1 Признак сравнения

Пусть $0 \le f(x) \le g(x)$ для всех $x \ge a$. Если интеграл $\int_a^\infty g(x)\,dx$ сходится, то сходится и интеграл $\int_a^\infty f(x)\,dx$.

7.7.2 Признак Дирихле

Пусть $f(x)$ и $g(x)$ — функции, определённые на промежутке $[a, \infty)$, такие что:

- $f(x)$ имеет ограниченные частичные суммы $\int_a^t f(x)\,dx$ для всех $t \ge a$,
- $g(x)$ — монотонная функция и $g(x) \to 0$ при $x \to \infty$.

Тогда несобственный интеграл $\int_a^\infty f(x)g(x)\,dx$ сходится.

8 8. Понятие кратного интеграла по Риману. Сведение кратного интеграла к повторному. Замена переменных в кратных интегралах.

8.1 Понятие кратного интеграла по Риману

Кратный интеграл обобщает понятие определённого интеграла на функции нескольких переменных. Например, двойной интеграл по области D для функции $f(x, y)$ записывается как:

$$\iint_D f(x, y)\,dx\,dy.$$

Он вычисляется как предел интегральных сумм, аналогичных определению интеграла Римана для одной переменной:

$$\iint_D f(x,y)\,dx\,dy = \lim_{\Delta x,\Delta y\to 0}\sum_{i=1}^{n}\sum_{j=1}^{m} f(x_i,y_j)\Delta x_i\Delta y_j.$$

8.2 Сведение кратного интеграла к повторному

Кратный интеграл можно свести к повторному, если область интегрирования представляет собой прямоугольник. Для двойного интеграла по прямоугольной области $D = [a,b]\times[c,d]$, справедливо:

$$\iint_D f(x,y)\,dx\,dy = \int_a^b\left(\int_c^d f(x,y)\,dy\right)dx.$$

Этот процесс можно обобщить на интегралы с большим числом переменных (например, тройные интегралы):

$$\iiint_D f(x,y,z)\,dx\,dy\,dz = \int_a^b\int_c^d\int_e^f f(x,y,z)\,dz\,dy\,dx.$$

8.3 Замена переменных в кратных интегралах

При вычислении кратных интегралов иногда удобно перейти к новым переменным. Пусть $x = x(u,v)$ и $y = y(u,v)$ — замена переменных. Тогда двойной интеграл по области D преобразуется:

$$\iint_D f(x,y)\,dx\,dy = \iint_{D'} f(x(u,v),y(u,v))\left|\frac{\partial(x,y)}{\partial(u,v)}\right|du\,dv,$$

где $\frac{\partial(x,y)}{\partial(u,v)}$ — якобиан преобразования.

Якобиан для преобразования (x,y) в (u,v) определяется как:

$$\frac{\partial(x,y)}{\partial(u,v)} = \begin{vmatrix}\frac{\partial x}{\partial u} & \frac{\partial x}{\partial v}\\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v}\end{vmatrix}.$$

Аналогичные правила применяются для тройных и более кратных интегралов при замене переменных.

9 9. Понятие гладкой кривой, гладкой поверхности, их параметрическое задание. Определение длины кривой, площади куска поверхности. Криволинейные интегралы первого и второго рода. Поверхностные интегралы первого и второго рода.

9.1 Гладкая кривая

Кривая L в пространстве называется гладкой, если её параметрическое задание $r(t) = (x(t), y(t), z(t))$ имеет непрерывные производные первого порядка по параметру t на отрезке $[a, b]$. То есть компоненты $x(t)$, $y(t)$, $z(t)$ дифференцируемы и их производные непрерывны на этом интервале.

9.2 Гладкая поверхность

Поверхность S называется гладкой, если она может быть задана параметрически через две переменные u и v: $r(u, v) = (x(u, v), y(u, v), z(u, v))$, где функции $x(u, v)$, $y(u, v)$ и $z(u, v)$ имеют непрерывные частные производные первого порядка по u и v.

9.3 Определение длины кривой

Пусть кривая L задана параметрически как $r(t) = (x(t), y(t), z(t))$, $t \in [a, b]$. Тогда длина этой кривой вычисляется по формуле:

$$L = \int_a^b \sqrt{\left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2 + \left(\frac{dz(t)}{dt}\right)^2}\, dt.$$

9.4 Определение площади куска поверхности

Пусть поверхность S задана параметрически как $r(u, v) = (x(u, v), y(u, v), z(u, v))$, $(u, v) \in D$. Тогда площадь поверхности вычисляется по формуле:

$$A = \iint_D \left| \frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} \right| du\, dv,$$

где $\times$ обозначает векторное произведение.

9.5 Криволинейные интегралы первого рода

Криволинейный интеграл первого рода для функции $f(x, y, z)$ вдоль кривой L определяется как:

$$\int_L f(x, y, z)\, ds = \int_a^b f(x(t), y(t), z(t)) \sqrt{\left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2 + \left(\frac{dz(t)}{dt}\right)^2}\, dt.$$

Этот интеграл вычисляется как сумма значений функции вдоль кривой с учётом длины дуги.

9.6 Криволинейные интегралы второго рода

Криволинейный интеграл второго рода для векторного поля $\vec{F}(x,y,z) = P(x,y,z)\vec{i} + Q(x,y,z)\vec{j} + R(x,y,z)\vec{k}$ вдоль кривой L определяется как:

$$\int_L P\,dx + Q\,dy + R\,dz = \int_a^b \left(P(x(t),y(t),z(t))\frac{dx(t)}{dt} + Q(x(t),y(t),z(t))\frac{dy(t)}{dt} + R(x(t),y(t),z(t))\frac{dz(t)}{dt} \right) dt.$$

Этот интеграл измеряет работу векторного поля вдоль кривой.

9.7 Поверхностные интегралы первого рода

Поверхностный интеграл первого рода для функции $f(x,y,z)$ по поверхности S определяется как:

$$\iint_S f(x,y,z)\,dS = \iint_D f(x(u,v),y(u,v),z(u,v)) \left| \frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} \right| du\,dv.$$

Этот интеграл вычисляется как сумма значений функции на поверхности с учётом площади элементов поверхности.

9.8 Поверхностные интегралы второго рода

Поверхностный интеграл второго рода для векторного поля $\vec{F}(x,y,z)$ по поверхности S определяется как:

$$\iint_S \vec{F} \cdot d\vec{S} = \iint_D \vec{F}(r(u,v)) \cdot \left(\frac{\partial r}{\partial u} \times \frac{\partial r}{\partial v} \right) du\,dv.$$

Этот интеграл измеряет поток векторного поля через поверхность.

10 10. Формула Грина на плоскости. Формула Гаусса-Остроградского. Формула Стокса. Дифференциальные операции. Градиент, дивергенция, ротор (вихрь). Криволинейные интегралы, не зависящие от пути интегрирования. Потенциальные векторные поля. Полный дифференциал, необходимые условия, достаточные условия.

10.1 Введение

Формулы Грина, Гаусса-Остроградского и Стокса — это важные теоремы, связывающие интегралы по области, поверхности или кривой с интегралами по её границе. Они играют ключевую роль в векторном анализе и используются в физике для описания потоков и циркуляции. Дифференциальные операции, такие как градиент, дивергенция и ротор, описывают изменение векторных и скалярных полей, а потенциальные векторные поля помогают описать силы, которые можно представить в виде градиента потенциальной энергии.

10.2 Формула Грина на плоскости

Формула Грина связывает двойной интеграл по области с криволинейным интегралом по её границе. Пусть D — область на плоскости с гладкой границей C, и пусть $P(x, y)$ и $Q(x, y)$ — функции с непрерывными частными производными на D. Тогда формула Грина имеет вид:

$$\oint_C P\,dx + Q\,dy = \iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dx\,dy.$$

Эта формула используется для вычисления криволинейных интегралов через двойные интегралы по области.

10.3 Формула Гаусса-Остроградского

Формула Гаусса-Остроградского (также называемая дивергентной теоремой) связывает поток векторного поля через поверхность с дивергенцией этого поля внутри объёма, ограниченного этой поверхностью. Пусть V — объём с гладкой границей S, и пусть $\vec{F}(x, y, z)$ — векторное поле с непрерывными частными производными на V. Тогда:

$$\iint_S \vec{F} \cdot d\vec{S} = \iiint_V (\nabla \cdot \vec{F})\,dV,$$

где $\nabla \cdot \vec{F}$ — дивергенция поля $\vec{F}$. Эта формула позволяет выражать поток через интеграл от дивергенции по объёму.

10.4 Формула Стокса

Формула Стокса связывает криволинейный интеграл вдоль границы поверхности с поверхностным интегралом от ротора векторного поля через эту поверхность. Пусть S — гладкая поверхность с гладкой границей C, и пусть $\vec{F}(x, y, z)$ — векторное поле. Тогда:

$$\oint_C \vec{F} \cdot d\vec{r} = \iint_S (\nabla \times \vec{F}) \cdot d\vec{S},$$

где $\nabla \times \vec{F}$ — ротор поля $\vec{F}$. Эта формула используется для вычисления циркуляции векторного поля через ротор.

10.5 Дифференциальные операции

- **Градиент.** Градиент скалярной функции $f(x, y, z)$ — это вектор, показывающий направление наибольшего возрастания функции. Определяется как:
$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right).$$

- **Дивергенция.** Дивергенция векторного поля $\vec{F}(x, y, z) = (F_1, F_2, F_3)$ измеряет величину источника или стока в данной точке пространства:
$$\nabla \cdot \vec{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z}.$$

- **Ротор.** Ротор векторного поля $\vec{F}(x, y, z)$ измеряет локальную циркуляцию поля:
$$\nabla \times \vec{F} = \left(\frac{\partial F_3}{\partial y} - \frac{\partial F_2}{\partial z}, \frac{\partial F_1}{\partial z} - \frac{\partial F_3}{\partial x}, \frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right).$$

10.6 Криволинейные интегралы, не зависящие от пути интегрирования

Если криволинейный интеграл $\int_C \vec{F} \cdot d\vec{r}$ не зависит от пути интегрирования, то поле $\vec{F}$ является потенциальным, и его можно записать как градиент некоторой скалярной функции $f(x, y, z)$:

$$\vec{F} = \nabla f.$$

В этом случае интеграл зависит только от начальной и конечной точек пути, а не от его формы.

10.7 Потенциальные векторные поля

Векторное поле $\vec{F}$ называется потенциальным, если существует скалярная функция f, такая что $\vec{F} = \nabla f$. Потенциальные поля не имеют вихрей, то есть их ротор равен нулю:

$$\nabla \times \vec{F} = 0.$$

10.8 Полный дифференциал

Полный дифференциал функции $f(x, y, z)$ выражает малое изменение функции в зависимости от изменения всех её переменных:

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz.$$

10.9 Необходимые и достаточные условия существования полного дифференциала

Для того чтобы функция $f(x, y, z)$ имела полный дифференциал, её частные производные первого порядка должны быть непрерывны. Это условие является как необходимым, так и достаточным для существования полного дифференциала.

11 11. Функциональные последовательности и ряды. Равномерная сходимость. Критерий Коши равномерной сходимости. Признак Вейерштрасса равномерной сходимости непрерывных функций. Теоремы о почленном интегрировании и дифференцировании функциональных рядов.

11.1 Введение

Функциональные последовательности и ряды представляют собой последовательности или ряды функций, которые могут сходиться к другой функции на каком-либо множестве. Важно различать два типа сходимости: поточечную и равномерную. Равномерная сходимость играет важную роль, так как позволяет проводить операции интегрирования и дифференцирования почленно, сохраняя свойства предельной функции.

11.2 Функциональные последовательности и ряды

Пусть $\{f_n(x)\}$ — последовательность функций, определённых на множестве D. Мы говорим, что функциональная последовательность $\{f_n(x)\}$ сходится

к функции $f(x)$ на D, если для любого $x \in D$ выполнено:

$$\lim_{n\to\infty} f_n(x) = f(x).$$

Функциональным рядом называется ряд вида:

$$\sum_{n=1}^{\infty} f_n(x).$$

Сумма такого ряда, если он сходится, также является функцией, определённой на D.

11.3 Равномерная сходимость

Последовательность функций $\{f_n(x)\}$ называется равномерно сходящейся к функции $f(x)$ на множестве D, если для любого $\varepsilon > 0$ существует такое N, что для всех $n > N$ и для всех $x \in D$ выполняется:

$$|f_n(x) - f(x)| < \varepsilon.$$

Это более сильное требование по сравнению с поточечной сходимостью, так как оно требует сходимости функции одинаково быстро на всём множестве D.

11.4 Критерий Коши равномерной сходимости

Последовательность функций $\{f_n(x)\}$ равномерно сходится на множестве D, если для любого $\varepsilon > 0$ существует такое N, что для всех $n, m > N$ и для всех $x \in D$ выполняется:

$$|f_n(x) - f_m(x)| < \varepsilon.$$

Этот критерий является аналогом критерия Коши для числовых последовательностей.

11.5 Признак Вейерштрасса равномерной сходимости непрерывных функций

Ряд $\sum_{n=1}^{\infty} f_n(x)$ равномерно сходится на множестве D, если существует последовательность положительных чисел $\{M_n\}$ такая, что:

$$|f_n(x)| \le M_n \quad \text{для всех} \quad x \in D$$

и ряд $\sum_{n=1}^{\infty} M_n$ сходится. Этот признак позволяет установить равномерную сходимость через сравнение с рядом из чисел.

11.6 Теоремы о почленном интегрировании и дифференцировании функциональных рядов

- **Теорема о почленном интегрировании.** Если последовательность функций $\{f_n(x)\}$ равномерно сходится к функции $f(x)$ на множестве D и все функции $f_n(x)$ интегрируемы на D, то справедливо:

$$\int_D \left(\lim_{n\to\infty} f_n(x) \right) dx = \lim_{n\to\infty} \int_D f_n(x)\, dx.$$

  Это означает, что при равномерной сходимости можно почленно интегрировать ряд.

- **Теорема о почленном дифференцировании.** Если последовательность функций $\{f_n(x)\}$ и их производные $\{f_n'(x)\}$ равномерно сходятся на множестве D, то справедливо:

$$\frac{d}{dx}\left(\sum_{n=1}^{\infty} f_n(x) \right) = \sum_{n=1}^{\infty} f_n'(x).$$

  То есть производную от ряда можно находить почленно при условии равномерной сходимости функций и их производных.

12 12. Интегралы, зависящие от параметра. Равномерная сходимость несобственного интеграла, зависящего от параметра. Теоремы о непрерывности и дифференцируемости интегралов, зависящих от параметра.

12.1 Введение

Интегралы, зависящие от параметра, представляют собой выражения вида $\int_a^b f(x,\alpha)\, dx$, где α — параметр. Такие интегралы часто встречаются в физике и математике при исследовании функций, зависящих от нескольких переменных. Важно исследовать их свойства, такие как сходимость, непрерывность и возможность дифференцирования по параметру.

12.2 Интегралы, зависящие от параметра

Интегралом, зависящим от параметра α, называется выражение:

$$I(\alpha) = \int_a^b f(x,\alpha)\, dx,$$

где $f(x,\alpha)$ — функция, зависящая как от переменной x, так и от параметра α. Интересуют вопросы о том, как изменяется интеграл при изменении параметра α.

12.3 Равномерная сходимость несобственного интеграла, зависящего от параметра

Пусть $I(\alpha) = \int_a^\infty f(x, \alpha)\, dx$ — несобственный интеграл, зависящий от параметра α. Этот интеграл называется равномерно сходящимся на множестве значений параметра α, если для любого $\varepsilon > 0$ существует число $A > a$, такое что для всех α и для всех $b > A$ выполнено:

$$\left| \int_b^\infty f(x, \alpha)\, dx \right| < \varepsilon.$$

Равномерная сходимость несобственного интеграла позволяет утверждать, что интеграл хорошо ведёт себя для всех значений параметра α.

12.4 Теорема о непрерывности интегралов, зависящих от параметра

Пусть $f(x, \alpha)$ — непрерывная функция на $[a, b] \times \Lambda$, где Λ — множество значений параметра α. Если для всех $x \in [a, b]$ интеграл $\int_a^b f(x, \alpha)\, dx$ сходится и $f(x, \alpha)$ равномерно непрерывна по α, то интеграл $I(\alpha)$ непрерывен по α:

$$\lim_{\alpha \to \alpha_0} I(\alpha) = I(\alpha_0),$$

где $I(\alpha_0) = \int_a^b f(x, \alpha_0)\, dx$.

12.5 Теорема о дифференцируемости интегралов, зависящих от параметра

Пусть $f(x, \alpha)$ имеет непрерывную частную производную $\frac{\partial f(x,\alpha)}{\partial \alpha}$ на $[a, b] \times \Lambda$. Тогда справедливо:

$$\frac{d}{d\alpha} \left(\int_a^b f(x, \alpha)\, dx \right) = \int_a^b \frac{\partial f(x, \alpha)}{\partial \alpha}\, dx,$$

если выполняются следующие условия:

- $\frac{\partial f(x,\alpha)}{\partial \alpha}$ непрерывна по α,
- существует такая функция $g(x)$, что $|\frac{\partial f(x,\alpha)}{\partial \alpha}| \leq g(x)$, и интеграл $\int_a^b g(x)\, dx$ сходится.

Эта теорема позволяет дифференцировать интегралы по параметру, если соблюдены определённые условия регулярности функции $f(x, \alpha)$.

13 13. Ряды Фурье по тригонометрической системе. Сходимость рядов Фурье для кусочно-гладких функций. Порядок убывания коэффициентов Фурье для l-раз непрерывно-дифференцируемой функции. Равномерная сходимость ряда Фурье для непрерывно-дифференцируемой функции. Равномерное приближение непрерывных функций на отрезке тригонометрическими полиномами и многочленами.

13.1 Введение

Ряды Фурье позволяют разложить периодическую функцию в сумму тригонометрических функций (синусов и косинусов). Они играют важную роль в анализе и обработке сигналов, а также в решении задач математической физики. Важно исследовать условия сходимости ряда Фурье, поведение его коэффициентов, а также возможность равномерного приближения функций тригонометрическими полиномами.

13.2 Ряды Фурье по тригонометрической системе

Для периодической функции $f(x)$ с периодом 2π её ряд Фурье записывается в виде:

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx) \right),$$

где коэффициенты Фурье a_n и b_n вычисляются по формулам:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx, \quad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx.$$

Этот ряд выражает функцию $f(x)$ в виде бесконечной суммы тригонометрических функций.

13.3 Сходимость рядов Фурье для кусочно-гладких функций

Если функция $f(x)$ является кусочно-гладкой на интервале $[-\pi, \pi]$ (то есть она имеет конечное число разрывов первого рода и конечное число точек, где её производная не существует, но конечна), то ряд Фурье сходится к функции $f(x)$ в каждой точке, где $f(x)$ непрерывна, и к среднему значению

в точках разрыва:

$$\lim_{N\to\infty}\sum_{n=1}^{N}(a_n\cos(nx)+b_n\sin(nx))=\frac{f(x+0)+f(x-0)}{2}.$$

13.4 Порядок убывания коэффициентов Фурье для l-раз непрерывно-дифференцируемой функции

Если функция $f(x)$ является l раз непрерывно-дифференцируемой, то её коэффициенты Фурье убывают с порядком $O(n^{-(l+1)})$. Это означает, что чем более гладкая функция, тем быстрее убывают её коэффициенты Фурье:

$$a_n, b_n = O\left(\frac{1}{n^{l+1}}\right).$$

Чем выше порядок гладкости функции, тем быстрее коэффициенты стремятся к нулю.

13.5 Равномерная сходимость ряда Фурье для непрерывно-дифференцируемой функции

Если функция $f(x)$ является непрерывно-дифференцируемой на интервале $[-\pi,\pi]$, то её ряд Фурье сходится к ней равномерно. Это значит, что для любого $\varepsilon>0$ существует такое число N, что для всех $x\in[-\pi,\pi]$ и всех $n>N$:

$$\left|f(x)-\sum_{k=0}^{n}(a_k\cos(kx)+b_k\sin(kx))\right|<\varepsilon.$$

13.6 Равномерное приближение непрерывных функций на отрезке тригонометрическими полиномами и многочленами

Согласно теореме Вейерштрасса, любая непрерывная функция $f(x)$ на отрезке $[a,b]$ может быть сколь угодно точно приближена тригонометрическими полиномами. Это означает, что для любой непрерывной функции $f(x)$ и любого $\varepsilon>0$ существует тригонометрический полином $P_n(x)$ такой, что:

$$\sup_{x\in[a,b]}|f(x)-P_n(x)|<\varepsilon.$$

Этот результат можно распространить на многочлены в случае не периодических функций.

14 14. Принцип наименьшего действия. Уравнения Лагранжа второго рода для голономных систем. Функция Лагранжа. Интегралы движения.

14.1 Введение

Принцип наименьшего действия и уравнения Лагранжа второго рода — это фундаментальные принципы аналитической механики. Они описывают движение физических систем через минимизацию специальной функции, называемой функцией Лагранжа. Эти принципы применимы для широкого круга задач в физике, включая механические и электромагнитные системы. Интегралы движения позволяют находить законы сохранения для систем.

14.2 Принцип наименьшего действия

Принцип наименьшего действия утверждает, что движение механической системы происходит так, что действие S принимает минимальное значение. Действие определяется как интеграл от функции Лагранжа $L(q, \dot{q}, t)$ по времени:

$$S = \int_{t_1}^{t_2} L(q, \dot{q}, t)\, dt,$$

где q — обобщённые координаты системы, $\dot{q}$ — их производные по времени (обобщённые скорости), а $L(q, \dot{q}, t)$ — функция Лагранжа. Принцип наименьшего действия гласит, что истинная траектория системы — это та, при которой действие S минимально.

14.3 Уравнения Лагранжа второго рода для голономных систем

Голономная система — это система, в которой все связи могут быть выражены в виде уравнений, зависящих только от обобщённых координат и времени. Для таких систем уравнения Лагранжа второго рода выводятся из принципа наименьшего действия и имеют вид:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0,$$

где L — функция Лагранжа, q_i — обобщённые координаты, $\dot{q}_i$ — обобщённые скорости, а i — номер обобщённой координаты.

Эти уравнения позволяют описывать динамику системы, исходя из её кинетической и потенциальной энергии.

14.4 Функция Лагранжа

Функция Лагранжа L для механической системы определяется как разность кинетической энергии T и потенциальной энергии U:

$$L(q, \dot{q}, t) = T(q, \dot{q}, t) - U(q, t).$$

Кинетическая энергия T зависит от обобщённых координат и скоростей, а потенциальная энергия U зависит только от координат и времени. Эта функция является основой для записи уравнений Лагранжа второго рода.

14.5 Интегралы движения

Интегралы движения — это величины, сохраняющиеся во времени в процессе движения системы. Они связаны с симметриями системы. Примером интеграла движения является закон сохранения энергии, который следует из того, что функция Лагранжа не зависит явно от времени:

$$E = T + U = \text{const}.$$

Другими важными интегралами движения являются закон сохранения импульса и закон сохранения момента импульса, которые следуют из симметрий системы относительно сдвигов в пространстве и вращений соответственно.

Если функция Лагранжа не зависит от какой-либо обобщённой координаты q_i, то соответствующий обобщённый импульс $p_i = \frac{\partial L}{\partial \dot{q}_i}$ является интегралом движения:

$$p_i = \text{const}.$$

15 15. Уравнения Гамильтона для голономных систем. Функция Гамильтона. Уравнения Гамильтона через обобщённые координаты и импульсы.

15.1 Введение

Уравнения Гамильтона — это альтернативный способ описания динамики механических систем, который является более общим и мощным по сравнению с уравнениями Лагранжа. В этой формулировке движение системы описывается через обобщённые координаты и импульсы, что делает её особенно полезной в квантовой механике и статистической физике. Функция Гамильтона связана с энергией системы и используется для вывода уравнений движения.

15.2 Функция Гамильтона

Функция Гамильтона H определяется как преобразование Лежандра от функции Лагранжа. Для механической системы с обобщёнными координатами q_i, обобщёнными скоростями $\dot{q}_i$ и обобщёнными импульсами p_i, функция Гамильтона определяется как:

$$H(q, p, t) = \sum_i p_i \dot{q}_i - L(q, \dot{q}, t),$$

где L — это функция Лагранжа, а $p_i = \frac{\partial L}{\partial \dot{q}_i}$ — обобщённые импульсы.

Функция Гамильтона в механике часто совпадает с полной энергией системы, то есть суммой кинетической и потенциальной энергии:

$$H = T + U.$$

15.3 Уравнения Гамильтона для голономных систем

Уравнения Гамильтона являются системой дифференциальных уравнений первого порядка, которые описывают эволюцию системы. Они имеют вид:

$$\dot{q}_i = \frac{\partial H}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H}{\partial q_i},$$

где q_i — обобщённые координаты, p_i — обобщённые импульсы, а H — функция Гамильтона. Эти уравнения описывают, как меняются обобщённые координаты и импульсы системы во времени.

15.4 Уравнения Гамильтона через обобщённые координаты и импульсы

Основное преимущество уравнений Гамильтона по сравнению с уравнениями Лагранжа заключается в том, что они являются уравнениями первого порядка относительно времени. Это позволяет легче анализировать поведение систем, особенно в многомерных пространствах.

Для каждой обобщённой координаты q_i и соответствующего ей обобщённого импульса p_i система уравнений Гамильтона имеет вид:

$$\dot{q}_i = \frac{\partial H}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H}{\partial q_i}.$$

Эти уравнения описывают динамику системы во времени, где $\dot{q}_i$ — это скорость изменения обобщённой координаты, а $\dot{p}_i$ — скорость изменения обобщённого импульса.

Таким образом, уравнения Гамильтона дают полное описание эволюции механической системы через изменения координат и импульсов.

15.5 Преимущества гамильтоновой механики

Гамильтонова формализация более удобна для перехода к квантовой механике, где обобщённые координаты и импульсы становятся операторами. Кроме того, этот формализм позволяет легко вводить канонические преобразования, которые сохраняют структуру уравнений движения и используются для упрощения задач механики.

16 ЛИНЕЙНАЯ АЛГЕБРА

Линейная алгебра

1. Линейное пространство

Линейное пространство — это множество векторов, для которых определены операции сложения и умножения на число. Линейная зависимость — это ситуация, когда один вектор можно выразить через другие. Если это невозможно, векторы называются линейно независимыми. Размерность пространства — это количество линейно независимых векторов, образующих его базис. Базис — это минимальный набор векторов, с помощью которых можно выразить любой вектор пространства.

2. Матрицы и действия над ними

Матрица — это прямоугольная таблица чисел. Детерминант квадратной матрицы — это скаляр, характеризующий её свойства, например, возможность обращения. Ранг матрицы — это максимальное число линейно независимых строк или столбцов. Эквивалентность двух матриц — это ситуация, когда они описывают одно и то же линейное преобразование, но в разных базисах.

3. Системы линейных уравнений

Система линейных алгебраических уравнений — это набор уравнений с переменными, которые можно решить с помощью линейной алгебры. Однородная система имеет только тривиальное решение (все переменные равны нулю), если определитель матрицы системы не равен нулю. Критерий Кронекера-Капелли помогает понять, когда система совместна, т.е. имеет решения.

4. Линейные преобразования

Линейное преобразование — это функция, которая отображает векторы одного пространства в векторы другого, сохраняя операции сложения и умножения на число. Матрица линейного преобразования меняется при смене

базиса. Область значений — это множество всех векторов, которые могут быть результатом линейного преобразования.

5. Собственные векторы и собственные значения

Собственный вектор — это вектор, который после применения линейного преобразования меняет только свою длину, но не направление. Собственное значение — это множитель, на который изменяется собственный вектор. Жорданова форма матрицы позволяет упростить вычисления с линейными преобразованиями.

6. Скалярное произведение

Скалярное произведение — это операция, которая сопоставляет двум векторам число. В Евклидовом пространстве оно можно вычислить как произведение длин векторов на косинус угла между ними. Ортонормированный базис — это базис, где все векторы ортогональны друг другу и имеют единичную длину. Процесс ортогонализации Грама-Шмидта позволяет построить такой базис.

7. Самосопряжённые преобразования

Самосопряжённое преобразование — это линейное преобразование, которое совпадает с транспонированным и комплексно-сопряжённым преобразованием. Такие преобразования имеют реальные собственные значения и ортогональные собственные векторы.

8. Ортогональные преобразования

Ортогональное преобразование сохраняет длины векторов и углы между ними. Ортогональные матрицы имеют особое свойство: их обратная матрица равна транспонированной. Переход от одного ортонормированного базиса к другому можно осуществить с помощью ортогональной матрицы.

9. Квадратичные формы

Квадратичная форма — это выражение, содержащее сумму произведений координат вектора на некоторые коэффициенты. В процессе приведения к каноническому виду квадратичные формы упрощаются. Закон инерции Сильвестра утверждает, что числа положительных, отрицательных и нулевых коэффициентов при квадратичных формах не зависят от выбора базиса.

Обыкновенные дифференциальные уравнения

1. Элементарные методы интегрирования уравнений первого порядка

Уравнения первого порядка решаются методами разделения переменных, методом интегрирующего множителя и другими. Пример уравнения с разделяющимися переменными:

$$\frac{dy}{dx} = g(x)h(y)$$

Решение может быть получено путем разделения переменных и интегрирования.

2. Теоремы существования и единственности решения задачи Коши

Задача Коши для уравнения первого порядка состоит в нахождении решения при заданных начальных условиях. Формулировка теоремы существования и единственности для уравнения вида:

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

Если функция $f(x, y)$ и её частная производная по y непрерывны, то существует единственное решение задачи Коши.

3. Линейные уравнения n-го порядка с постоянными коэффициентами

Линейное однородное уравнение с постоянными коэффициентами:

$$a_n \frac{d^n y}{dx^n} + a_{n-1} \frac{d^{n-1} y}{dx^{n-1}} + \cdots + a_1 \frac{dy}{dx} + a_0 y = 0$$

Для решения используется характеристическое уравнение.

4. Решение неоднородного уравнения

Неоднородное уравнение решается методом вариации произвольных постоянных или методом неопределённых коэффициентов. Пример неоднородного уравнения:

$$y'' + p(x)y' + q(x)y = f(x)$$

Общее решение состоит из суммы общего решения однородного уравнения и частного решения неоднородного.

5. Линейные уравнения n-го порядка с переменными коэффициентами

Для таких уравнений вводится понятие фундаментальной системы решений. Пример уравнения:

$$y'' + p(x)y' + q(x)y = 0$$

Определитель Вронского:

$$W(y_1, y_2, \ldots, y_n) = \det \begin{vmatrix} y_1 & y_1' & \ldots & y_1^{(n-1)} \\ y_2 & y_2' & \ldots & y_2^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & y_n' & \ldots & y_n^{(n-1)} \end{vmatrix}$$

Если $W(x) \neq 0$, то решения линейно независимы.

6. Системы линейных уравнений первого порядка

Системы уравнений могут быть записаны в матричной форме:

$$\mathbf{Y}' = A(x)\mathbf{Y}$$

Для решения используется метод вариации произвольных постоянных и другие методы.

7. Понятие об уравнениях с разрывными коэффициентами

Уравнения с разрывными коэффициентами требуют специального подхода, например, введения слабого решения. Одним из важных случаев является задача нахождения обобщённых решений, когда стандартные методы не применимы.

8. Автономные системы и фазовые траектории

Автономная система имеет вид:

$$\frac{d\mathbf{X}}{dt} = \mathbf{F}(\mathbf{X})$$

Решение таких систем можно исследовать с помощью фазовой плоскости и фазовых траекторий.

9. Первые интегралы автономных систем

Первый интеграл автономной системы — это функция, которая остаётся постоянной вдоль траектории системы. Для однородных уравнений первого порядка можно использовать метод разделения переменных:

$$\frac{dy}{dx} = \frac{f(x)}{g(y)}$$

10. Элементы вариационного исчисления

Задачи вариационного исчисления включают нахождение экстремумов функционалов. Простейшая задача:

$$J(y) = \int_{x_1}^{x_2} F(x, y, y')dx$$

Наиболее известное уравнение — это уравнение Эйлера-Лагранжа:

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\frac{\partial F}{\partial y'} = 0$$

Теория функций комплексного переменного

1. Функция одной комплексной переменной

Функция комплексной переменной $f(z)$, где $z = x + iy$, дифференцируема, если она удовлетворяет условиям Коши-Римана:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

где $f(z) = u(x, y) + iv(x, y)$, а $u(x, y)$ и $v(x, y)$ — действительные и мнимые части функции соответственно. Эти условия связывают частные производные по действительной и мнимой частям. Геометрический смысл модуля и аргумента функции комплексного переменного заключается в интерпретации модуля как расстояния от начала координат до точки на комплексной плоскости, а аргумента как угла между положительным направлением оси x и вектором, направленным к этой точке. Производная функции $f(z)$ выражается как:

$$f'(z) = \lim_{\Delta z \to 0} \frac{f(z + \Delta z) - f(z)}{\Delta z}$$

2. Равенство нулю интеграла от дифференцируемой функции по замкнутому контуру

Интегральная форма теоремы Коши утверждает, что если функция $f(z)$ аналитична в области, ограниченной замкнутым контуром γ, то интеграл

этой функции по контуру равен нулю:

$$\oint_\gamma f(z)\,dz = 0$$

Это важное свойство аналитических функций. Если внутри контура находится особая точка, то значение интеграла можно вычислить с помощью вычетов. Интегральная теорема Коши также позволяет вычислять интегралы от дифференцируемых функций по замкнутым контурам, что приводит к обобщению — формуле интегрального представления Коши:

$$f(z_0) = \frac{1}{2\pi i} \oint_\gamma \frac{f(z)}{z - z_0}\,dz$$

где z_0 — точка внутри контура γ.

3. Понятие функции регулярной в точке и в области

Функция называется регулярной (или аналитической) в точке z_0, если она имеет конечные производные всех порядков в этой точке. Разложение функции в ряд Тейлора в окрестности точки z_0:

$$f(z) = \sum_{n=0}^{\infty} a_n (z - z_0)^n$$

где $a_n = \frac{f^{(n)}(z_0)}{n!}$. Первая теорема Абеля описывает круг сходимости степенного ряда. Если радиус сходимости равен R, то ряд сходится внутри круга радиуса R. Почленное дифференцирование и интегрирование степенных рядов возможны внутри круга сходимости. Дифференцируемость функции в области приводит к её регулярности, а регулярность функции в точке связана с её аналитичностью.

4. Разложение в ряд Тейлора функции

Если функция $f(z)$ дифференцируема в некоторой окрестности точки z_0, то её можно разложить в ряд Тейлора:

$$f(z) = \sum_{n=0}^{\infty} \frac{f^{(n)}(z_0)}{n!} (z - z_0)^n$$

Это разложение действует для любой точки, если функция аналитична в этой точке. Для особых точек используется ряд Лорана, который включает отрицательные степени:

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z - z_0)^n$$

Ряд Лорана полезен для анализа особенностей функций, таких как полюсы и существенно особые точки.

5. Изолированные особые точки

Изолированная особая точка — это точка, в которой функция не является аналитичной, но она аналитична в некоторой окрестности этой точки. Особые точки классифицируются на три типа:

- Устранимая особая точка: функция может быть продолжена до аналитической функции в этой точке.
- Полюс: функция стремится к бесконечности при приближении к этой точке.
- Существенно особая точка: поведение функции вблизи этой точки очень сложное, и функция не имеет предела.

Разложение функции в ряд Лорана вокруг особой точки позволяет описать её поведение:

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z - z_0)^n$$

Коэффициенты a_n дают информацию о типе особой точки.

6. Понятие вычета в изолированной особой точке

Вычет функции в изолированной особой точке z_0 — это коэффициент a_{-1} в разложении Лорана:

$$\operatorname{Res}(f, z_0) = a_{-1}$$

Теорема о вычетах позволяет вычислять интегралы от аналитических функций по контурам, содержащим особые точки:

$$\oint_\gamma f(z)\, dz = 2\pi i \sum \operatorname{Res}(f, z_k)$$

где сумма берётся по всем особым точкам внутри контура γ.

7. Разложение мероморфных функций на элементарные дроби

Мероморфные функции — это функции, которые аналитичны во всех точках, кроме полюсов. Их можно разложить на элементарные дроби. Пример бесконечного разложения целой функции:

$$f(z) = \sum_{n=1}^{\infty} \frac{a_n}{(z - z_n)^n}$$

Такие разложения позволяют упростить анализ сложных функций и вычисление интегралов в случае полюсов.

8. Теорема единственности регулярной функции

Теорема единственности утверждает, что если регулярная функция принимает заданные значения на последовательности точек, предельной точкой которых является точка внутри области, то такая функция единственна. Аналитическое продолжение используется для расширения области определения функции за пределы круга сходимости. Пример основного многофункционального элементарного выражения:

$$f(z) = z^{1/n}, \quad \ln z$$

Эти функции имеют несколько ветвей и требуют рассмотрения на римановой поверхности.

9. Конформные отображения

Конформные отображения — это отображения, которые сохраняют углы между пересекающимися линиями. Дробно-линейные отображения имеют вид:

$$f(z) = \frac{az + b}{cz + d}, \quad ad - bc \neq 0$$

Это отображение сохраняет углы, но может менять величину. Общая теорема Римана утверждает, что любое односвязное область на комплексной плоскости можно конформно отобразить на единичный круг, сохраняя угловые соотношения.

Уравнения математической физики

1. Линейные дифференциальные уравнения в частных производных 2-го порядка

Эти уравнения включают производные по нескольким переменным. Их приводят к каноническому виду с помощью замены переменных. Уравнения классифицируются на эллиптические, гиперболические и параболические. Примером является уравнение Лапласа (эллиптическое):

$$\Delta u = 0$$

2. Линейные дифференциальные уравнения 2-го порядка на плоскости

Эти уравнения описывают процессы в двумерном пространстве. Понятие характеристик позволяет найти области влияния начальных условий. Приведение к каноническому виду помогает упростить решение. Задача Коши — это нахождение решения при заданных начальных условиях. Пример уравнения:

$$u_{tt} - c^2 u_{xx} = 0$$

Теорема Коши-Ковалевской утверждает существование и единственность решения.

3. Понятие корректной краевой задачи для уравнений в частных производных

Краевая задача включает условия на границе области. Примером является задача Коши для уравнения Лапласа, где условия заданы на границе. Классические задачи математической физики — это задачи теплопроводности, колебания струн и волновые уравнения. Задачи Дирихле и Неймана — это задачи с заданием граничных условий для уравнений Лапласа.

4. Интегральные уравнения Фредгольма

Интегральные уравнения описывают зависимости, когда искомая функция входит под знак интеграла. Уравнения с вырожденным ядром допускают более простое решение. Теорема Фредгольма утверждает существование решений для уравнений второго рода. Пример:

$$\phi(x) - \lambda \int_a^b K(x,t)\phi(t)\,dt = f(x)$$

5. Интегральные уравнения Фредгольма второго рода

В этих уравнениях ядро является симметричным, что упрощает анализ. Теорема Гильберта-Шмидта утверждает, что симметричное ядро можно разложить по собственным функциям. Решения выражаются через ряды по этим собственным функциям.

6. Задача Штурма-Лиувилля

Эта задача — это уравнение для собственных значений и собственных функций. Задачи краевого типа можно свести к интегральным уравнениям. Функции Грина используются для решения краевых задач:

$$\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) + \lambda w(x) y = 0$$

где λ — собственное значение.

7. Задача Коши для волнового уравнения

Волновое уравнение описывает колебания струны:

$$u_{tt} = c^2 u_{xx}$$

Формула Даламбера даёт решение при заданных начальных данных. Теорема Коши утверждает существование и единственность решения.

8. Смешанные задачи для гиперболических уравнений

Метод Фурье используется для решения таких задач, разбивая решение на сумму гармонических колебаний. Метод разделения переменных помогает свести задачу к обычным дифференциальным уравнениям.

9. Задача Коши для уравнения теплопроводности

Это уравнение описывает распределение температуры во времени:

$$u_t = ku_{xx}$$

Теорема существования и единственности решения гарантирует корректность задачи. Фундаментальное решение можно найти через преобразование Фурье.

10. Смешанная задача для уравнения теплопроводности

Эта задача решается методом разделения переменных, который сводит уравнение к системе обычных дифференциальных уравнений. Принцип максимума утверждает, что максимумы и минимумы температуры достигаются на границах области.

11. Уравнение Лапласа и Пуассона

Гармонические функции удовлетворяют уравнению Лапласа:

$$\Delta u = 0$$

Принцип максимума утверждает, что функция достигает своих экстремумов на границе области. Формула Грина используется для нахождения решения уравнения в области.

12. Задача Дирихле для уравнения Лапласа

Задача Дирихле заключается в нахождении решения уравнения Лапласа при заданных значениях функции на границе. Решение в круге можно найти методом Фурье. Теорема существования и единственности утверждает, что решение существует и единственно для заданных граничных условий.

13. Задача Неймана для уравнения Лапласа

Задача Неймана отличается от задачи Дирихле тем, что на границе заданы значения производной функции. Степень неопределённости решения определяется добавлением константы.

Вычислительная математика

1. Решение систем нелинейных уравнений

Для решения систем нелинейных уравнений часто применяется метод Ньютона, который основан на итерациях и имеет квадратичную скорость сходимости. Теорема о сходимости утверждает, что метод сходится при определённых условиях. Простой итерационный метод анализируется по параметру, что позволяет улучшить сходимость. Метод продолжения по параметру используется для нахождения решений в сложных системах.

2. Численное дифференцирование

Численное дифференцирование — это аппроксимация производной с помощью разностных схем. Основные схемы включают центральную разность для первой и второй производных. Ошибка аппроксимации зависит от шага сетки:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Ошибка округления и шаг численного дифференцирования влияют на точность вычислений.

3. Численное интегрирование задачи Коши для системы ОДУ

Методы решения задач Коши включают метод сеток, простейшие разностные схемы (например, явная и неявная схемы Эйлера). Схемы различаются по устойчивости и точности. Ошибка аппроксимации и выбор критерия шага сетки важны для точности. Пример явной схемы Эйлера:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

4. Методы типа Рунге-Кутты

Методы Рунге-Кутты обеспечивают более точную аппроксимацию решений ОДУ по сравнению с простыми методами. Четвёртый порядок метода является наиболее популярным. Пример метода Рунге-Кутты четвёртого порядка:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

где k_1, k_2, k_3, k_4 — промежуточные вычисления.

5. Жёсткие системы ОДУ

Жёсткие системы ОДУ возникают, когда в системе присутствуют процессы с разными временными масштабами. Для их решения применяются численные методы, устойчивые к жёсткости, такие как метод A-устойчивости или

метод продолжения. Жёсткость может приводить к потере точности при использовании стандартных методов.

6. Нелинейные краевые задачи для ОДУ

Методы "стрельбы"и функциональные методы (например, метод Ньютона) используются для решения нелинейных краевых задач. Эти методы сводят задачу к решению системы нелинейных уравнений в функциональном пространстве.

7. Линейные краевые задачи для систем ОДУ

Краевые задачи для систем ОДУ решаются с использованием метода прогонки для систем с трёхдиагональной матрицей. Этот метод устойчив и эффективен для задач с гладкими решениями. Также используется метод конечных разностей, который позволяет свести задачу к системе линейных уравнений.

8. Линейные краевые задачи с большим параметром

При больших параметрах в краевых задачах возникают нестабильности, что требует применения методов прогонки или других устойчивых методов, таких как методы с переменными коэффициентами. Сведение задачи к устойчивым краевым условиям позволяет улучшить сходимость решений.

9. Прогонка в разностной задаче Штурма-Лиувилля

Алгоритм прогонки используется для решения краевых задач Штурма-Лиувилля. Этот метод эффективен для задач с регулярными коэффициентами. Реализация метода заключается в сведении задачи к трёхдиагональной системе.

10. Метод сеток для уравнения теплопроводности

Простейшие разностные схемы, такие как явная и неявная схемы, используются для численного решения уравнения теплопроводности. Реализация явной схемы простая, но менее устойчивая. Неявная схема более устойчива и применяется для больших временных шагов.

11. Нелинейные уравнения, их разностная аппроксимация

Реализация численного метода для нелинейных уравнений требует выбора устойчивых схем. Методы Ньютона и прогонки могут быть адаптированы для нелинейных уравнений.

12. Устойчивость разностных схем

Свойства устойчивости численных методов обеспечиваются за счёт анализа зависимости решений от начальных условий. Спектральный анализ помогает определить устойчивость схем, используя понятие собственных чисел и функций.

13.Двумерное уравнение теплопроводности

Для решения двумерного уравнения используются явные и неявные схемы. Метод переменных направлений разделяет задачу на несколько одномерных задач для повышения эффективности. Спектральный анализ позволяет оценить устойчивость схемы.

14. Решение уравнений Пуассона методом сеток

Метод сеток для решения уравнения Пуассона включает разностные аппроксимации, простые итерационные методы, такие как метод Якоби или метод Зейделя, и спектральный анализ для улучшения сходимости.

15. Метод переменных направлений для решения уравнения Пуассона

Этот метод позволяет решать уравнения в нескольких направлениях, разделяя задачу на более простые одномерные. Выбор оптимального параметра и анализ устойчивости являются важными этапами.

16. Численные методы решения задач механики сплошной среды

Консервативные методы используются для сохранения физических величин в задачах механики сплошной среды. Схемы разностных методов строятся так, чтобы удовлетворять законам сохранения.

17. Метод поиска экстремумов функций

Методы поиска экстремумов включают градиентные методы и методы Ньютона. Эти методы используются для нахождения оптимальных решений задач.

18. Постановка некорректных задач

Некорректные задачи возникают, когда решение не зависит непрерывно от начальных данных. Для решения таких задач применяются методы регуляризации, которые вводят априорную информацию для стабилизации решения.

19. Основная идея регуляризации

Регуляризация основана на введении дополнительной информации для корректировки решения. Тихоновская регуляризация является одним из популярных методов, она минимизирует сумму невязки и гладкости решения:

$$\min \left(||Ax - b||^2 + \alpha ||x||^2 \right)$$

где α — параметр регуляризации.

Алгоритмы

1. Понятие алгоритма

Алгоритм — это конечная последовательность шагов, предназначенная для решения задачи или выполнения вычислений. Алгоритмы описывают процесс в точной форме, обычно в виде математических операций.

2. Понятие пространственной сложности алгоритма

Пространственная сложность алгоритма — это количество памяти, которое необходимо алгоритму для выполнения задачи в зависимости от размера входных данных. Обычно обозначается как $S(n)$, где n — размер входных данных.

3. Понятие временной сложности алгоритма

Временная сложность — это количество операций, которые выполняет алгоритм, в зависимости от размера входных данных. Обозначается как $T(n)$. Примером может быть линейная сложность $O(n)$, где количество шагов пропорционально размеру входных данных.

4. Машина Тьюринга

Машина Тьюринга — это абстрактная математическая модель вычислений, которая использует бесконечную ленту и набор правил для манипуляции символами. Она способна моделировать любой алгоритм.

5. Машина Поста

Машина Поста — это абстракция вычислительной машины, похожая на машину Тьюринга, но с более простыми правилами. Она также универсальна и может выполнять любые вычисления, описываемые алгоритмами.

6. Нормальные алгоритмы Маркова

Алгоритмы Маркова — это форма рекурсивных вычислений, основанная на правилах переписывания строк. Они используют набор продукций (правил) для преобразования символов и моделируют вычисления.

7. Проблема останова

Проблема останова заключается в том, что невозможно определить, завершится ли произвольная программа на произвольном входе или будет работать бесконечно. Это было доказано Алоном Тьюрингом.

8. Алгоритмы сортировки

Алгоритмы сортировки используются для упорядочивания элементов. Примеры включают сортировку пузырьком, быструю сортировку и сортировку слиянием. Их сложность варьируется от $O(n \log n)$ до $O(n^2)$.

9. Жадный алгоритм

Жадный алгоритм — это алгоритм, который на каждом шаге выбирает локально оптимальное решение, надеясь, что оно приведёт к глобально оптимальному результату. Примером является задача о сдаче разменной суммы.

10. Бинарный поиск

Бинарный поиск — это алгоритм для поиска элемента в отсортированном массиве, который на каждом шаге делит массив пополам и проверяет центральный элемент. Время выполнения — $O(\log n)$.

11. Динамическое программирование (2, 3-мерное)

Динамическое программирование — это метод оптимизации, который решает сложные задачи, разбивая их на подзадачи и запоминая результаты для избежания повторных вычислений. Пример: нахождение наибольшей общей подпоследовательности.

12. Динамическое программирование на подотрезках

Этот метод применяется для задач, которые могут быть разбиты на независимые подзадачи, например, задачи о парентезировании матриц. Результаты предыдущих вычислений используются для ускорения последующих.

13. Динамическое программирование по профилю

Этот метод использует состояние задачи, представляя её в виде профиля. Пример — задачи, связанные с графами или последовательностями.

14. Дискретная и непрерывная задача о рюкзаке

Задача о рюкзаке — это задача оптимального выбора предметов с учётом ограничений по весу и объёму. Существует два варианта: дискретная задача, когда предметы нельзя делить, и непрерывная, когда можно брать доли предметов.

15. Задача о наименьшем общем предке (Least Common Ancestor, LCA)

LCA — это задача нахождения наименьшего общего предка двух узлов в дереве. Это важная задача в теории графов с множеством применений в компьютерной науке.

16. Классы сложности алгоритмов (P, NP)

Класс P включает задачи, которые могут быть решены за полиномиальное время. Класс NP — это задачи, решения которых можно проверить за полиномиальное время. Неизвестно, равны ли классы P и NP.

17. Префикс-функция

Префикс-функция для строки s — это массив, в котором $[i]$ обозначает длину наибольшего собственного префикса строки, который одновременно является её суффиксом. Используется в алгоритмах поиска подстрок.

18. Z-функция. Алгоритм Кнута-Морриса-Пратта

Z-функция строки s — это массив, в котором $Z[i]$ показывает длину наибольшего префикса строки s, который совпадает с её суффиксом, начиная с позиции i. Алгоритм Кнута-Морриса-Пратта использует префикс-функцию для эффективного поиска подстрок за $O(n + m)$ времени.

19. Алгоритм Ахо-Корасик

Алгоритм Ахо-Корасик предназначен для поиска множества шаблонов в тексте. Он строит автомат и осуществляет одновременный поиск всех шаблонов за линейное время от длины текста.

20. Расширенный алгоритм Евклида

Расширенный алгоритм Евклида находит не только наибольший общий делитель (НОД) двух чисел, но и коэффициенты, такие что $ax + by = \text{НОД}(a, b)$. Используется в криптографии и теории чисел.

21. Решето Эратосфена

Алгоритм для нахождения всех простых чисел до заданного числа n. Работает за время $O(n \log \log n)$, последовательно исключая кратные простых чисел.

22. Алгоритмы сжатия информации без потерь

Алгоритмы сжатия информации без потерь, такие как Huffman и LZW, уменьшают объём данных без потерь информации. Они используют статистику символов и шаблоны для кодирования данных.

23. Алгоритм Евклида

Алгоритм Евклида находит наибольший общий делитель (НОД) двух чисел с помощью последовательного деления с остатком:

$$\text{НОД}(a, b) = \text{НОД}(b, a\%b)$$

24. Решето Эратосфена

Алгоритм, который позволяет находить все простые числа до n, исключая кратные числа начиная с наименьших простых чисел. Алгоритм работает за $O(n \log \log n)$.

25. Вычислительная сложность алгоритмов сложения, умножения, возведения в степень

Сложность сложения и умножения чисел зависит от их длины. Наивные алгоритмы работают за время $O(n)$ для сложения и $O(n^2)$ для умножения. Быстрые алгоритмы, такие как алгоритм Карацубы, сокращают сложность умножения до $O(n^{\log_2 3})$.

26. Асимптотический закон распределения простых чисел. Алгоритмы проверки чисел на простоту

Асимптотический закон распределения простых чисел утверждает, что количество простых чисел до n приближается к $n/\log n$. Для проверки чисел на простоту используются алгоритмы, такие как тест Ферма, тест Миллера-Рабина и другие.

Процедурное программирование

1. Базовые алгоритмические конструкции

Основные конструкции в программировании включают условные операторы (if), операторы множественного выбора (case/switch/select) и циклы.

Циклы могут быть с предусловием (for, while) или постусловием (do/while). Оператор перехода (goto) используется для изменения порядка выполнения программы.

2. Разбиение программ на процедуры и модули

Программы могут быть разбиты на отдельные модули и процедуры, что улучшает их структуру и читаемость. Стек используется для управления вызовами функций. Глобальные переменные видны во всей программе, тогда как локальные переменные видны только внутри определённого блока.

3. Базовые типы и структуры данных

Основные типы данных включают числа (целые, вещественные), строки, множества, массивы и записи (struct). Программы могут рассматриваться как комбинации алгоритмов и структур данных, что является фундаментом структурированного программирования.

4. Понятие структурированного программирования

Структурированное программирование основывается на использовании чётко организованных алгоритмов и данных. Принцип: "программа = алгоритмы + структуры данных". Алгоритм Дейкстры демонстрирует важность структурирования для поиска кратчайших путей.

Структуры данных

1. Стек

Стек — это структура данных, в которой доступ к элементам осуществляется по принципу LIFO (последний пришёл — первый вышел). Операции включают добавление (push) и удаление (pop).

2. Очередь

Очередь работает по принципу FIFO (первый пришёл — первый вышел). Элементы добавляются в конец очереди и удаляются из начала.

3. Очередь с приоритетами

Элементы в такой очереди имеют приоритеты. Элемент с наивысшим приоритетом удаляется первым, вне зависимости от времени его добавления.

4. Деки

Дек (двусторонняя очередь) позволяет добавлять и удалять элементы с обоих концов.

5. Односвязные списки

Это структура данных, в которой каждый элемент содержит указатель на следующий элемент списка. Доступ к элементам осуществляется последовательно.

6. Двусвязные списки

Каждый элемент в двусвязном списке содержит указатели как на следующий, так и на предыдущий элементы, что позволяет осуществлять доступ в обоих направлениях.

7. Куча

Куча — это структура данных, представляющая собой бинарное дерево, в котором каждый узел удовлетворяет свойству кучи: для каждой пары дочерних узлов родительский узел больше (для max-кучи) или меньше (для min-кучи).

8. Бинарное дерево

Бинарное дерево — это дерево, в котором каждый узел имеет не более двух потомков (левый и правый). Часто используется для поиска и сортировки данных.

9. Декартово дерево

Это сбалансированное бинарное дерево, в котором узлы сортируются по двум ключам: одному для организации бинарного дерева поиска и другому для обеспечения структуры кучи.

10. Хэш-таблицы

Хэш-таблица использует хэш-функцию для вычисления индекса, по которому хранится элемент. Обеспечивает быстрый доступ к данным при вставке и поиске.

11. Двоичный контейнер (Range Minimum Query, RMQ)

RMQ — это структура данных, предназначенная для быстрого ответа на запросы о минимальных значениях в диапазоне массива. Используются алгоритмы, такие как разбиение на блоки и дерево отрезков.

12. Дерево отрезков

Дерево отрезков — это структура данных, которая позволяет эффективно выполнять запросы и обновления на отрезках массива, например, нахождение суммы или минимума на подотрезке.

13. Дерево Фенвика

Дерево Фенвика используется для эффективного вычисления префиксных сумм и обновления данных в массиве. Оно позволяет обрабатывать запросы за $O(\log n)$.

14. Красно-чёрное дерево

Это сбалансированное бинарное дерево поиска, которое гарантирует, что его глубина будет логарифмической по количеству узлов. Используется для реализации таких структур, как множества и словари.

15. Понятие отображения (map)

Map — это структура данных, которая сопоставляет ключи и значения. Реализуется через бинарные деревья или хэш-таблицы, обеспечивая быстрый доступ по ключу.

Язык программирования C++

1. Препроцессор

Препроцессор C++ обрабатывает директивы, такие как #include и #define, до компиляции программы.

2. Ветвления

Ветвления реализуются с помощью условных операторов (if, else, switch) для выполнения разных блоков кода в зависимости от условий.

3. Циклы

Циклы (for, while, do-while) позволяют выполнять набор инструкций многократно до выполнения условия.

4. Функции

Функции в C++ используются для повторного использования кода. Они могут возвращать значения и принимать параметры.

5. Массивы

Массивы — это структуры данных для хранения элементов одного типа. Индексация начинается с нуля.

6. Арифметика указателей

Указатели содержат адреса в памяти, и с ними можно выполнять арифметические операции для перемещения по элементам массива.

7. Рекурсия

Рекурсия — это процесс, когда функция вызывает саму себя для решения задачи, пока не будет достигнуто базовое условие.

8. Структуры

Структуры — это пользовательские типы данных, которые могут содержать разные типы данных под одним именем.

9. Объединение (union)

Union позволяет хранить разные типы данных в одной и той же области памяти, но только один тип в каждый момент времени.

10. Стандартная библиотека C

C++ поддерживает все функции стандартной библиотеки C, такие как работа с файлами и строки.

11. Библиотека STL

STL (Standard Template Library) предоставляет контейнеры (vector, list, map) и алгоритмы для работы с данными.

12. Библиотеки Boost

Boost — это набор высококачественных библиотек C++, расширяющих возможности стандартной библиотеки.

13. Стандарты C++11, C++14

C++11 и C++14 — это обновлённые версии стандарта языка C++, которые включают новые возможности, такие как лямбда-выражения, улучшения в многопоточности и авто-типизация.

Язык программирования Java

1. Виртуальная машина Java

Виртуальная машина Java (JVM) — это среда выполнения, которая позволяет Java-программам быть независимыми от платформы.

2. Управление памятью

Java использует автоматическую сборку мусора для управления памятью, освобождая объекты, которые больше не используются.

3. Передача примитивных типов в функции

Примитивные типы в Java передаются по значению, что означает копирование их значений при передаче в функции.

4. Передача ссылочных типов в функции

Ссылочные типы (объекты) передаются по ссылке, что позволяет изменять состояние объекта внутри функции.

5. Проблема изменения ссылки внутри подпрограммы

При передаче ссылочных типов изменяется только содержимое объекта, но не сама ссылка.

6. Статические инициализаторы

Статические инициализаторы в Java используются для инициализации статических переменных класса при его загрузке.

7. Удаление неиспользуемых объектов и метод finalize

Метод ‘finalize()‘ может быть переопределён для выполнения кода перед удалением объекта сборщиком мусора.

8. Проблема деструкторов для сложно устроенных объектов

Java не поддерживает деструкторы, как в C++, что затрудняет управление ресурсами, не связанными с памятью, такими как файлы и сетевые соединения.

9. Сборка мусора

Сборка мусора — это автоматический процесс освобождения памяти в Java, который устраняет необходимость вручную управлять памятью.

Архитектура ЭВМ

1. Архитектура ЭВМ (Гарвардская, фон Неймановская)

Фон Неймановская архитектура использует единую память для хранения данных и инструкций, тогда как Гарвардская архитектура разделяет их на разные области памяти.

2. Набор команд процессора (CISC, RISC, VLIW)

CISC (Complex Instruction Set Computing) — процессоры с большим набором сложных инструкций.RISC (Reduced Instruction Set Computing) использует упрощённый набор команд. VLIW (Very Long Instruction Word) позволяет процессорам выполнять несколько инструкций одновременно.

3. Кэш и ускорение работы с его использованием

Кэш-память — это высокоскоростная память, расположенная между процессором и основной памятью, что ускоряет доступ к часто используемым данным.

4. Соглашения о вызове

Соглашения о вызове определяют способ передачи параметров в функции и возврата значений. Примеры: cdecl, fastcall, stdcall.

5. Представление целых чисел

Целые числа представляются в двоичном формате с использованием дополнительного кода для представления отрицательных чисел.

6. Представление чисел с плавающей точкой

Числа с плавающей точкой представлены в виде мантиссы и порядка. Стандарт IEEE 754 используется для представления таких чисел.

7. Язык ассемблера

Ассемблер — это низкоуровневый язык программирования, который напрямую управляет работой процессора через команды.

8. Выполнение команд на конвейере

Конвейеризация позволяет процессору выполнять несколько инструкций одновременно, разделяя их на этапы (выборка, декодирование, выполнение).

9. Средства отладки и инструментирования

Для отладки и анализа программ используются такие инструменты, как ‘valgrind‘, ‘gdb‘, и профилировщики, которые помогают отслеживать ошибки и производительность.

10. Статические и динамические библиотеки

Статические библиотеки компилируются в исполняемый файл, а динамические загружаются во время выполнения программы, что уменьшает размер исполняемого файла.

11. Профилирование программ

Профилирование — это процесс анализа производительности программы для выявления "узких мест которые замедляют выполнение.

12. Статистический метод изучения программ

Этот метод использует данные о частоте выполнения различных частей программы для оптимизации её работы.

13. Динамический метод изучения программ

Включает мониторинг выполнения программы в реальном времени для изучения её поведения.

14. Средства защиты программного обеспечения

Методы защиты включают использование шифрования, аутентификацию и встроенные механизмы защиты данных от несанкционированного доступа.

15. Аппаратные средства защиты от несанкционированного доступа

Включают технологии шифрования данных и идентификации оборудования, такие как электронные ключи и TPM (Trusted Platform Module).

Принципы построения современных операционных систем

1. Классификация операционных систем

Операционные системы классифицируются по типам: многозадачные, однопользовательские, многопользовательские, распределённые, реального времени и встраиваемые.

2. Операционные системы реального времени

ОС реального времени обеспечивают предсказуемое время выполнения задач и используются там, где задержки недопустимы (например, в системах управления).

3. Понятие процесса, виды процессов

Процесс — это программа в состоянии выполнения. Виды процессов включают системные и пользовательские процессы, фоновые задачи и задачи с приоритетом.

4. Файлы. Структура файловой системы

Файл — это именованная область на диске, в которой хранятся данные. Файловые системы управляют хранением данных, примеры: FAT, NTFS, ext4.

5. Управление памятью

Операционные системы используют разные методы управления памятью: одиночное распределение, страничное и сегментное распределение. Свопинг перемещает данные между оперативной памятью и диском.

6. Взаимодействие процессов

Процессы взаимодействуют с помощью механизмов IPC (межпроцессное взаимодействие): пайпы, очереди сообщений, сокеты, семафоры и разделяемая память. Это позволяет обмениваться данными и сигналами.

7. Виды виртуализации

Виртуализация позволяет запускать несколько виртуальных машин на одном физическом сервере. Виды: аппаратная виртуализация, паравиртуализация, контейнеризация.

8. Способы разделения ресурсов

ОС использует разные механизмы для предотвращения тупиков (deadlock) и управления доступом к разделяемым ресурсам, например, семафоры и мьютексы.

9. Системы безопасности

Системы безопасности управляют учётными записями пользователей, обеспечивают контроль доступа и ведение журналов событий (logs).

10. Понятие идентификации и аутентификации

Идентификация — это процесс определения личности пользователя, а аутентификация — это проверка подлинности личности с помощью паролей, биометрии или токенов.

11. Классификация угроз безопасности

Основные угрозы безопасности включают вирусы, трояны, несанкционированный доступ и утечку данных. ОС использует системы контроля доступа и антивирусные программы для защиты.

12. Модель управления доступом (УД)

Модель УД описывает управление доступом к ресурсам системы. Схемы УД включают списки контроля доступа (ACL), мандатную модель и метки безопасности. Примеры: SELinux, AppArmor.

Объектно-ориентированное программирование

1. Объект = данные + методы работы с ними

Объект — это сущность в ООП, которая содержит данные (свойства) и методы для работы с этими данными. Абстракция позволяет скрывать сложные детали реализации.

2. Инкапсуляция

Инкапсуляция заключается в сокрытии данных и предоставлении доступа к ним только через методы класса, что упрощает модификацию и оптимизацию программ.

3. Наследование

Наследование позволяет создавать новые классы на основе существующих, повторно используя код. Полиморфизм обеспечивает возможность работы с объектами разных классов через один интерфейс.

4. Свойства и события

Свойства (properties) — это данные объекта, доступные для чтения и/или записи. События (events) позволяют объектам реагировать на внешние действия.

5. Диаграммы классов

Диаграммы классов представляют объектно-ориентированную структуру программы, включая классы, их атрибуты, методы и связи между ними.

6. Обработка внешних ситуаций. Исключения

Исключения — это механизмы для обработки ошибок во время выполнения программы. Система исключений позволяет предотвращать аварийные завершения программы и корректно обрабатывать ошибки.

7.Полиморфизм

Полиморфизм позволяет объектам разных классов обрабатывать вызовы однотипных методов по-разному. Это достигается через перегрузку и переопределение методов.

Сетевые технологии

1. Концепция клиент-сервер

Клиент-сервер — это модель взаимодействия, где клиент запрашивает услуги или данные у сервера. Примеры: веб-браузеры (клиенты) и веб-серверы.

2. Распределённые вычислительные системы

Эти системы объединяют несколько компьютеров для совместного решения задач. Модель OSI описывает 7 уровней взаимодействия в открытых системах.

3. Архитектура компьютерных сетей

Структура сети включает топологию (например, звезда, кольцо, шина), архитектуру (например, Интернет, локальные сети) и методы управления потоками данных.

4. Протокол IPv4

IPv4 — это протокол, использующий 32-битные IP-адреса. Маски подсети помогают разделять сети на подгруппы. IPv6 использует 128-битные адреса.

5. Сетевые системные вызовы

Системные вызовы (socket, bind, listen и т.д.) в ОС обеспечивают создание и управление сетевыми соединениями.

6. Протоколы TCP и UDP

TCP обеспечивает надёжную передачу данных, включая контроль целостности и порядок сообщений. UDP — более быстрый, но менее надёжный протокол.

7. Сериализация/десериализация

Процесс преобразования объектов в поток байтов (сериализация) и обратное преобразование (десериализация) для передачи по сети.

8. Основы языка HTML

HTML — язык разметки для создания веб-страниц. Основные теги включают заголовки, параграфы, ссылки и изображения.

9. Система доменных имён (DNS)

DNS переводит доменные имена (например, www.example.com) в IP-адреса, необходимые для маршрутизации запросов.

10. Латентность сети, RTT

RTT (Round-Trip Time) — это время, за которое данные проходят от источника до получателя и обратно. Латентность влияет на скорость соединения.

11. Удалённый вызов процедур (RPC)

RPC позволяет программе вызвать функцию на удалённом сервере, как если бы она выполнялась локально.

12. Электронная почта и WWW

Принципы электронной почты (SMTP, POP3) и WWW (HTTP/HTTPS) описывают механизмы доставки сообщений и предоставления веб-контента.

13. Протоколы прикладного уровня

Протоколы прикладного уровня обеспечивают взаимодействие между клиентом и сервером на уровне приложений, например, HTTP, FTP, SMTP.

14. Архитектура "тонкого клиента"

Тонкий клиент — это устройство с минимальными вычислительными ресурсами, которое полагается на сервер для выполнения большинства задач.

15. Межсетевые экраны (firewalls)

Межсетевые экраны блокируют или разрешают сетевые подключения в зависимости от настроенных правил. Несанкционированный доступ может быть ограничен с помощью блокирующих экранов.

16. Атаки и сбои сети

DDoS-атаки (Distributed Denial of Service) перегружают систему запросами, что делает её недоступной. Отказоустойчивые системы помогают поддерживать работу сети при сбоях.

17. Технологии COM/DCOM и CORBA

COM (Component Object Model) и CORBA (Common Object Request Broker Architecture) обеспечивают взаимодействие между приложениями через стандартные интерфейсы.

Компьютерная графика

1. Представление цвета в ЭВМ

Цвета в компьютерах представлены с использованием моделей RGB (красный, зелёный, синий) или CMYK (голубой, пурпурный, жёлтый, чёрный).

2. Графические форматы

Форматы изображений могут быть векторными (SVG) или растровыми (JPEG, PNG), различаясь методом хранения данных.

3. Векторные и растровые форматы

Векторные форматы описывают изображения математически, а растровые — как набор пикселей.

4. Проекции

Проекция — это метод отображения трёхмерных объектов на двумерную плоскость, например, ортогональная и перспективная проекции.

5. Метод маршрирования кубиков

Метод используется для рендеринга поверхностей на основе трёхмерных данных, таких как томографические изображения.

6. Быстрое преобразование Фурье

Алгоритм для вычисления преобразования Фурье, который применяется для обработки изображений и сигналов, ускоряя вычисления.

7. Сжатие данных с потерей качества

JPEG — это пример сжатия с потерями, где изображения компрессируются с уменьшением качества, чтобы снизить размер файлов.

8. Графический интерфейс пользователя (GUI)

GUI позволяет пользователям взаимодействовать с программами через графические элементы (кнопки, окна, меню).

Искусственный интеллект

1. Машинное обучение

Машинное обучение — это область ИИ, которая позволяет системам обучаться на данных для улучшения своей производительности без явного программирования.

2. Нейросети

Нейросети — это модели, вдохновлённые работой мозга, состоящие из нейронов, которые обрабатывают информацию и учатся на данных.

3. 3 закона робототехники

Законы робототехники, предложенные Айзеком Азимовым, регулируют поведение роботов, включая защиту людей и повиновение приказам, если это не противоречит первому закону.

1. Первый закон: робот не может причинить вред человеку или через своё бездействие допустить, чтобы человеку был причинён вред.

2. Второй закон: робот должен подчиняться указаниям человека, кроме случаев, когда такие указания противоречат первому закону роботехники.

3. Третий закон: робот должен защищать собственную безопасность, если только такие действия не противоречат первому или второму закону роботехники.

4. Графические ускорители в машинном обучении

Графические процессоры (GPU) используются для ускорения вычислений в задачах машинного обучения, особенно при обработке больших объёмов данных.

Методы анализа и распознавания данных

1. Задача распознавания

Задача распознавания заключается в классификации данных (например, изображений или звуков) на основе определённых характеристик.

2. Задача классификации

Классификация — это задача отнесения объекта к одному из нескольких заранее определённых классов на основе признаков.

3. Ошибки первого и второго рода

Ошибка первого рода (ложное срабатывание) — это случай, когда отвергается истинная гипотеза. Ошибка второго рода (ложноотрицательный результат) — это случай, когда принимается ложная гипотеза.

Теория графов

1. Вершины и рёбра

Граф состоит из множества вершин и рёбер, которые соединяют пары вершин. Вершины представляют собой объекты, а рёбра — связи между ними.

2. Ориентированные и неориентированные графы

В ориентированных графах рёбра имеют направление, указывая на порядок соединения вершин. В неориентированных графах рёбра не имеют направления, и связь между вершинами симметрична.

3. Матрица смежности

Матрица смежности представляет граф в виде двумерного массива, где строка и столбец соответствуют вершинам, а элементы указывают на наличие или отсутствие рёбер между ними.

4. Матрица инцидентности

Матрица инцидентности описывает связь между вершинами и рёбрами. В ориентированном графе строки соответствуют вершинам, а столбцы — рёбрам. Элементы показывают, какая вершина инцидентна (связана) с каким ребром.

5. Дерево

Дерево — это связный ациклический граф. В дереве любая пара вершин соединена единственным путём. Деревья широко используются для поиска и сортировки данных.

6. Формула Кэли

Формула Кэли определяет количество остовных деревьев в полном графе: n^{n-2}, где n — количество вершин.

7. Обход в глубину (DFS)

Обход в глубину — это алгоритм поиска, который исследует граф, двигаясь по рёбрам как можно глубже, прежде чем перейти к соседям вершины. Применяется для поиска путей, циклов и компонент связности.

8. Обход в ширину (BFS)

Обход в ширину исследует граф слоями, начиная с вершины и посещая всех её соседей, а затем переходя к соседям соседей. Этот метод используется для поиска кратчайшего пути в неориентированных графах.

9. Поток в графе

Задачи на поток в графах описывают передачу ресурса (например, данных или воды) через сеть рёбер с ограничениями на их пропускную способность. Алгоритм Форда-Фалкерсона используется для нахождения максимального потока.

10. Маршруты, цепи, циклы

Маршрут — это последовательность рёбер, соединяющих вершины. Цепь — это маршрут без повторяющихся рёбер, а цикл — это цепь, которая возвращается в исходную вершину.

11. Эйлеров путь

Эйлеров путь проходит через все рёбра графа ровно один раз. Если граф имеет более двух вершин нечётной степени, эйлеров путь невозможен.

12. Гамильтонов путь

Гамильтонов путь проходит через каждую вершину графа ровно один раз. Поиск такого пути является NP-полной задачей.

13. Алгоритм Флойда

Алгоритм Флойда решает задачу поиска кратчайших путей между всеми парами вершин в графе. Это динамический алгоритм, работающий за время $O(n^3)$.

14. Алгоритм Дейкстры

Алгоритм Дейкстры находит кратчайший путь от одной вершины до всех остальных в графе с неотрицательными весами рёбер.

15. Алгоритм Крускала

Алгоритм Крускала находит минимальное остовное дерево, добавляя рёбра по возрастанию веса, пока не будут соединены все вершины графа.

16. Алгоритм Диница

Алгоритм Диница — это улучшенный метод для нахождения максимального потока в сети. Он использует обход в ширину для нахождения "слоёв"в графе.

17. Двудольные графы. Паросочетания

Двудольный граф — это граф, вершины которого можно разделить на два множества, и рёбра соединяют вершины только из разных множеств. Паросочетание — это набор рёбер, не имеющих общих вершин.

18. Планарность графа

Граф называется планарным, если его можно нарисовать на плоскости без пересечения рёбер. Примером планарного графа является граф Куратовского.

Базы данных

1. СУБД

Системы управления базами данных (СУБД) обеспечивают организацию, хранение и доступ к данным. Логическая структура описывает, как данные видны пользователю, а физическая структура — как они хранятся на диске. Средства обеспечения целостности данных включают транзакции и блокировки.

2. Реляционная модель данных

Реляционные базы данных организуют данные в таблицы. Нормализация используется для уменьшения дублирования данных и зависимости между таблицами. ER-диаграммы описывают сущности и связи между ними. SQL — это язык для работы с реляционными базами данных.

3. Клиент-серверные и трёхуровневые архитектуры

Клиент-серверная архитектура разделяет базу данных и приложение на два уровня: клиент (интерфейс пользователя) и сервер (обработка данных). Трёхуровневая архитектура добавляет промежуточный слой для обработки логики приложения.

4. Хранилища данных

Хранилища данных отличаются от операционных баз данных тем, что используются для анализа больших объёмов данных. Они организуются в многомерные модели (например, OLAP). Витрины данных используются для создания отчётов и анализа.

5. Безопасность баз данных

Угрозы безопасности БД включают несанкционированный доступ, утечку данных и внутренние ошибки. Меры защиты включают шифрование, аутентификацию и контроль доступа. Важно соблюдать конфиденциальность и целостность данных.

6. Управление доступом к БД

Управление доступом включает назначение прав пользователям и группам, а также использование ролей для контроля за действиями. Привилегии могут быть системными и объектными (например, доступ к таблице).

7. Транзакции и блокировки

Транзакции — это последовательности операций, которые выполняются атомарно. Блокировки применяются для обеспечения изоляции транзакций и предотвращения конфликтов при параллельном доступе.

8. Ссылочная целостность

Ссылочная целостность обеспечивает согласованность данных между связанными таблицами. Внешние ключи используются для связывания строк в разных таблицах.

9. SQL-инъекции

SQL-инъекции — это атаки, которые используют уязвимости в запросах SQL для несанкционированного доступа к базе данных. Защита от таких атак включает параметризированные запросы и проверку ввода данных.

Технологический цикл разработки ПО

1. Итеративная (спиральная) модель разработки ПО

Итеративная модель разработки программного обеспечения предполагает создание продукта через повторяющиеся циклы. На каждом этапе происходит уточнение требований, разработка, тестирование и выпуск промежуточной версии. Конечный релиз создаётся путём накопления улучшений и исправлений, что минимизирует риски на каждом этапе.

2. Анализ и проектирование ПО

Анализ включает сбор и анализ требований к системе, что необходимо для понимания того, что нужно разработать. Проектирование предусматривает создание архитектуры и деталей реализации. CASE-средства помогают автоматизировать проектирование сложных систем, создавая схемы, модели и диаграммы.

3. Управление и планирование

Управление проектом включает распределение ресурсов, оценку временных рамок и управление рисками. Контроль проекта осуществляется через вехи

(milestones), которые являются контрольными точками для оценки прогресса. Управление рисками помогает минимизировать возможные проблемы до того, как они повлияют на проект.

4. Системы ведения версий

Системы ведения версий используются для отслеживания изменений в исходном коде и других компонентах проекта. Они позволяют сохранять историю изменений, возвращаться к предыдущим версиям и управлять ветками разработки. Это помогает командам разработчиков координировать работу над проектом.

5. Прогресс промышленного тестирования

Тестирование — это важная часть контроля качества программного обеспечения. Промышленное тестирование включает различные виды проверок: функциональные, нагрузочные, интеграционные и регрессионные тесты. Регистрация ошибок и их исправление помогают улучшить стабильность и надёжность программы.

6. Автоматизация сборки программ

Автоматизация сборки позволяет сократить количество ошибок при ручной компиляции и сборке программ. Утилиты, такие как Make, Maven, или Gradle, используются для автоматизации процесса компиляции, тестирования и сборки проектов. Это ускоряет разработку и снижает вероятность человеческих ошибок.

7. Понятие технического задания (ТЗ)

Техническое задание — это документ, описывающий требования к программному продукту. Он включает функциональные и нефункциональные требования, сроки разработки и требования к качеству. ТЗ является основой для проектирования и разработки программного обеспечения.

8. Принципы тестирования. Разработка, ориентированная на тесты (TDD)

Тестирование проводится для выявления и исправления ошибок в программном обеспечении. Классификация дефектов включает функциональные, производственные и интерфейсные ошибки. TDD (Test-driven development) предполагает написание тестов до реализации функциональности, что улучшает качество кода.

9. Системы контроля версий

Системы контроля версий, такие как CVS, SVN, Mercurial и Git, обеспечивают управление изменениями в исходном коде. Они позволяют разработчикам работать над проектом параллельно, отслеживая все изменения и разрешая конфликты.

10. Системы управления качеством (QA)

Системы управления качеством помогают обеспечивать, что программное обеспечение соответствует установленным требованиям. QA включает процессы планирования качества, тестирования, анализа дефектов и управления улучшениями.

11. Методы структурного проектирования

Методы структурного проектирования позволяют разбить сложные системы на модули для упрощения разработки и тестирования. Различают методы проектирования сверху вниз и снизу вверх. Диаграммы потоков данных (DFD) и структурные диаграммы (STD) помогают визуализировать структуру системы.

Информационная безопасность

1. Понятие неотказуемости

Неотказуемость — это свойство системы, гарантирующее, что участник коммуникации не сможет отрицать факт своего участия в транзакции. Это важно для обеспечения доверия в цифровых взаимодействиях, таких как электронная подпись. Модели систем неотказуемости включают методы шифрования и использования временных меток.

2. Модель системы обеспечения неотказуемости

Модель системы обеспечения неотказуемости включает в себя методы, которые предотвращают возможность отказа от своих действий, таких как подписка или отправка сообщения. Это достигается с помощью криптографических механизмов и протоколов безопасности. Информация, необходимая для обеспечения неотказуемости, включает цифровые подписи и сертификаты.

3. Архитектура информационной безопасности

Архитектура информационной безопасности включает в себя меры защиты, такие как шифрование, управление доступом и межсетевые экраны. Последствия реализации угроз безопасности могут включать утечку данных,

кражу конфиденциальной информации и разрушение систем. Источники угроз включают внешние атаки, внутренние ошибки и вредоносное ПО.

4. Источники угроз информационной безопасности

Основные источники угроз включают кибератаки, вирусы, несанкционированный доступ и утечку данных. Внутренние угрозы могут возникать из-за ошибок сотрудников или намеренных действий. Защита информации включает внедрение политик безопасности, регулярное обновление программного обеспечения и использование антивирусных систем.

3. Цели, задачи, способы и средства обеспечения информационной безопасности

Информационная безопасность охватывает такие аспекты, как доступность, конфиденциальность и целостность данных. Доступность подразумевает обеспечение возможности доступа к данным в нужное время, конфиденциальность защищает данные от несанкционированного доступа, а целостность обеспечивает их защиту от изменений. Идентифицируемость позволяет отслеживать действия пользователей для повышения безопасности.

4. Понятие целостности

Целостность — это свойство, гарантирующее, что данные остаются неизменными и точными. Модели обеспечения целостности включают контроль доступа, контроль версий и проверку целостности. Основные способы защиты целостности данных включают контрольные суммы и цифровые подписи.

5. Формальные модели безопасности

Формальные модели безопасности описывают правила и политики, которые должны соблюдать информационные системы для защиты данных. Эти модели включают атрибутивные модели, которые описывают, кто и как может взаимодействовать с объектами системы. Примеры моделей включают модель Белл-ЛаПадулы (конфиденциальность) и модель Биба (целостность).

6. Критерии и классы защищённости средств вычислительной техники и информационных систем

Средства защиты классифицируются по уровням защищённости. Например, для компьютеров используются механизмы аутентификации, системы защиты на уровне операционных систем, межсетевые экраны и антивирусы. Эти классы защищённости позволяют определять уровень безопасности, обеспечиваемый системой.

7. Идентификация и аутентификация в программах

Идентификация пользователя — это процесс определения его личности, а аутентификация подтверждает подлинность. Методы включают пароли, смарт-карты и биометрию. Системы аутентификации должны хранить пароли в защищённом виде, например, с помощью хеширования.

8. Компьютерные вирусы

Компьютерные вирусы — это вредоносные программы, которые могут копировать себя и заражать другие программы или файлы. Жизненный цикл вируса включает этапы заражения, активации и распространения. Существуют разные виды вирусов: файловые, загрузочные, макровирусы.

9. Программные закладки

Программные закладки — это скрытые функции, встроенные в программное обеспечение, которые могут использоваться для несанкционированного доступа. Методы защиты от закладок включают межсетевые экраны, системы контроля целостности и антивирусы.

10. Атаки типа "маскарад"и их виды

Атака типа "маскарад"заключается в том, что злоумышленник выдаёт себя за законного пользователя системы для получения доступа к данным. Такие атаки включают фишинг, кражу сессий и подделку адресов. Методы защиты включают двухфакторную аутентификацию и шифрование сеансов.

11. Модели аудита безопасности и системы оповещения об опасности

Аудит безопасности позволяет отслеживать действия пользователей и выявлять аномалии в системе. Системы обнаружения вторжений (IDS) анализируют сетевой трафик и события для выявления подозрительной активности и предупреждения о возможных угрозах.

Криптография

1. Криптография и криптоанализ

Криптография — это наука о защите информации с помощью шифрования. Криптоанализ занимается взломом криптографических систем. Криптографические протоколы разрабатываются с учётом конфиденциальности, целостности и аутентификации, чтобы обеспечить надёжную защиту данных.

2. Подстановочные шифры

Подстановочные шифры заменяют буквы или символы текста другими символами по определённому правилу. Пример — моноалфавитная замена, где каждый символ заменяется на один и тот же другой символ. Перестановочные шифры меняют порядок символов без их замены.

3. Стеганография

Стеганография — это метод скрытия информации внутри других данных, например, в изображениях или аудиофайлах. Стеганография используется для скрытной передачи сообщений, чтобы само существование сообщения осталось незамеченным.

4. Защита информации гаммированием

Гаммирование — это метод шифрования, при котором исходный текст комбинируется с псевдослучайной последовательностью, называемой гаммой. Этот метод часто используется в симметричных шифрах для повышения безопасности.

5. Генераторы псевдослучайных последовательностей

Генераторы псевдослучайных последовательностей (ПСП) используются для создания последовательностей, которые кажутся случайными, но на самом деле основаны на начальном значении (сиде). Линейные и нелинейные регистры сдвига широко применяются в криптографии для генерации таких последовательностей.

6. Проблемы распространения ключей

Распространение ключей — это одна из главных проблем в криптографии, особенно в симметричных системах, где один и тот же ключ используется для шифрования и дешифрования. Для решения этой проблемы используются асимметричные системы, где есть публичный и приватный ключи.

7. Криптосистема RSA

RSA — это асимметричная криптосистема, основанная на сложности разложения больших чисел на простые множители. Она используется для шифрования и цифровых подписей, обеспечивая как безопасность, так и аутентичность.

8. Криптосистема Диффи-Хеллмана

Алгоритм Диффи-Хеллмана используется для безопасного обмена ключами по незащищённому каналу. Основная идея заключается в использовании

степеней модулей больших простых чисел для вычисления общего ключа, который трудно перехватить.

9. Каноническое разложение натуральных чисел

Каноническое разложение числа — это представление числа в виде произведения простых чисел. Это основа для многих криптографических алгоритмов, включая RSA.

10. Протоколы по модулю малых чисел

Квадратичные вычеты и квадратичные законы взаимности, такие как закон Гаусса, используются для работы с числовыми системами по модулю, что важно для криптографических вычислений. Протоколы по модулю используются в шифровании для повышения безопасности.

11. Протоколы разделения секретов

Протоколы разделения секретов позволяют нескольким участникам совместно владеть секретом таким образом, что только все вместе они могут восстановить исходные данные. Этот метод полезен в ситуациях, где нужно обеспечить высокую степень доверия и безопасности.

12. Протоколы с нулевым разглашением

Протоколы с нулевым разглашением позволяют доказать, что сторона владеет определённой информацией, не раскрывая саму информацию. Эти протоколы используются в блокчейне и других распределённых системах для аутентификации и проверки.

13. Модели обеспечения криптополитики

Модели криптополитики определяют правила использования криптографии в организациях. Они устанавливают стандарты для шифрования, распределения ключей и защиты информации, а также описывают ответственность за выполнение этих стандартов.

14. Понятие конфиденциальности

Конфиденциальность в криптографии означает защиту информации от несанкционированного доступа. Для её обеспечения используются методы шифрования, распределение ключей и аутентификация. Цель состоит в том, чтобы только авторизованные лица могли получить доступ к данным.

15. Модели защиты информации

Модели защиты информации описывают, как данные должны защищаться в информационных системах. Это включает защиту от внешних угроз, контроль доступа и обеспечение целостности данных. Информационные системы должны соответствовать определённым стандартам безопасности, чтобы эффективно предотвращать утечки данных.

Алгебра логики

1. Логические переменные

Логические переменные принимают значения "истина"(True) или "ложь"(False). Они используются для представления высказываний и событий в логических выражениях и операциях.

2. Основные операции алгебры логики

К основным операциям алгебры логики относятся: отрицание (¬), конъюнкция (, логическое "и"), дизъюнкция (, логическое "или") и исключающее "или"(). Эти операции используются для составления логических выражений и построения булевых функций.

3. Таблица истинности

Таблица истинности представляет все возможные значения логических переменных и результат логической операции для каждого из этих значений. Она используется для анализа логических функций и проверки их правильности.

4. Полнота системы функций

Система функций называется полной, если с её помощью можно выразить любую логическую функцию. Например, конъюнкция, дизъюнкция и отрицание образуют полную систему логических функций.

Теория формальных языков

1. Понятие языка

Формальный язык — это множество строк, составленных из конечного алфавита символов. Языки используются для описания синтаксиса программирования, коммуникации и автоматизации задач.

2. Формальная грамматика

Формальная грамматика описывает правила построения строк в языке. Она включает терминальные и нетерминальные символы, а также правила вывода строк.

3. Контекстно-свободная грамматика

Контекстно-свободная грамматика определяет, что каждая производная строка может быть выведена независимо от её окружения. Она широко используется для описания синтаксиса языков программирования.

4. Контекстно-зависимая грамматика

Контекстно-зависимые грамматики учитывают окружающий контекст символов при выводе правил. Они более мощные, чем контекстно-свободные, но сложнее в обработке.

5. Конечные автоматы

Конечный автомат — это модель вычислений, которая принимает входные символы и находится в одном из конечного числа состояний. Он используется для распознавания простых языков и регулярных выражений.

6. Магазинные автоматы

Магазинный автомат — это расширение конечного автомата, которое имеет память в виде стека. Он используется для распознавания контекстно-свободных языков.

7. Суффиксный массив

Суффиксный массив — это структура данных, которая хранит все суффиксы строки в лексикографическом порядке. Он полезен для быстрого поиска подстрок.

8. Суффиксный автомат

Суффиксный автомат — это конечный автомат, который строится для всех суффиксов строки. Он позволяет эффективно находить повторяющиеся подстроки.

Классификация языков программирования

1. Процедурные языки

Процедурные языки описывают программы в виде последовательности шагов или процедур, которые последовательно выполняются. Примеры: C, Pascal. Основное внимание уделяется функциям и процедурам, разделяющим код на логические блоки.

2. Логические языки

Логические языки основаны на логическом выводе и правилах. Примером является Prolog, где разработчик описывает факты и правила, а язык выводит заключения на их основе.

3. Функциональные языки

Функциональные языки основаны на математической концепции функций. Программы в этих языках представляются как набор вычислений, а не инструкций. Примеры: Haskell, Lisp.

4. Языки разметки (XML, TeX)

Языки разметки используются для описания структуры и форматирования текста. Примеры: XML для структурированных данных и TeX для форматирования научных текстов.

Вычислительная геометрия

1. Понятие точки и вектора

Точка определяется своими координатами в пространстве, а вектор — направлением и величиной.Векторы и точки образуют основные структуры для построения геометрических объектов и выполнения вычислений.

2. Скалярное произведение векторов

Скалярное произведение векторов — это величина, равная произведению их длин и косинуса угла между ними. Оно используется для вычисления углов и проверки ортогональности векторов.

3. Векторное произведение

Векторное произведение двух векторов в пространстве даёт вектор, перпендикулярный к плоскости, образованной исходными векторами. Этот метод полезен для определения направлений в трёхмерном пространстве.

4. Ориентированная площадь треугольника

Ориентированная площадь треугольника определяется через векторное произведение его сторон и даёт информацию о направлении обхода вершин (по часовой стрелке или против).

5. Предикат "по часовой стрелке"

Предикат используется для тестирования, находятся ли три точки в порядке по часовой стрелке или против неё. Это важно для задач, связанных с определением видимости и построением выпуклых оболочек.

6. Расстояния от точки до прямой

Расстояние от точки до прямой вычисляется через проекцию вектора, соединяющего точку с любой точкой на прямой. Это базовый инструмент в геометрических вычислениях.

7. Нахождение точки пересечения двух прямых

Прямые пересекаются, если у них совпадают координаты хотя бы одной точки. Это можно вычислить с помощью решения системы уравнений, описывающих прямые.

8. Пересечение окружности и прямой

Прямая может пересекать окружность в двух, одной или ни одной точке. Эти точки пересечения можно найти, решив квадратное уравнение, которое получается при подстановке уравнения прямой в уравнение окружности.

9. Выпуклая оболочка (O(N log N))

Выпуклая оболочка множества точек — это наименьший выпуклый многоугольник, содержащий все точки. Алгоритм Грэхема строит выпуклую оболочку за время $O(N \log N)$, сортируя точки и применяя метод сканирования.

10. Метод сканирующей прямой

Метод сканирующей прямой — это техника для решения задач пересечения геометрических объектов. Прямая проходит через множество точек, и проверяются пересечения объектов по мере продвижения этой прямой.

Параллельное программирование

1. Разновидности параллельных архитектур. SISD - MIMD

Параллельные архитектуры классифицируются по типу обработки данных и инструкций. SISD (Single Instruction, Single Data) обрабатывает одну инструкцию с одним потоком данных, как традиционный процессор. MIMD (Multiple Instruction, Multiple Data) позволяет нескольким процессорам одновременно выполнять различные инструкции с различными потоками данных, что используется в высокопроизводительных вычислениях.

2. Разновидности параллельных архитектур. Общая и разделяемая память

В архитектурах с общей памятью все процессоры имеют доступ к одной области памяти, что упрощает взаимодействие между ними, но требует сложных механизмов синхронизации. В архитектурах с разделяемой памятью каждая задача имеет свой собственный адресное пространство, что снижает конфликты, но усложняет взаимодействие между процессорами.

3. MPI — определение и основные принципы

MPI (Message Passing Interface) — это стандарт для организации обмена сообщениями в параллельных вычислениях. Он включает функции для отправки и получения сообщений, а также для синхронизации процессов. MPI-коммуникаторы (например, MPI_COMM_WORLD) представляют группы процессов, с которыми можно обмениваться данными.

4. MPI_Init() и MPI_Finalize()

MPI_Init() и MPI_Finalize() — это функции инициализации и завершения работы с MPI. Все MPI-программы должны начинаться с вызова MPI_Init(), а завершаться вызовом MPI_Finalize(), чтобы корректно начать и завершить параллельные вычисления.

5. MPI_Send() и MPI_Recv()

MPI_Send() используется для отправки сообщений между процессами, а MPI_Recv() — для их получения. Эти функции обеспечивают базовый механизм передачи данных между узлами в параллельной программе.

6. Блокирующие и неблокирующие посылки

Блокирующие посылки (MPI_Send) приостанавливают выполнение программы до тех пор, пока сообщение не будет отправлено. Неблокирующие посылки (MPI_Isend) позволяют программе продолжать выполнение, не дожидаясь завершения отправки данных.

7. Блокирующий и неблокирующий приём

Аналогично посылкам, блокирующий приём (MPI_Recv) останавливает выполнение программы, пока не будет получено сообщение. Неблокирующий приём (MPI_Irecv) позволяет программе продолжать выполнение, ожидая получения данных в фоновом режиме.

8. MPI_Bcast()

MPI_Bcast() используется для рассылки данных от одного процесса всем остальным в коммуникаторе. Это полезно для передачи общих данных, которые требуются всем процессам.

9. MPI_Reduce()

MPI_Reduce() собирает данные от всех процессов и применяет указанную операцию (например, сумму или максимум) к этим данным, а затем отправляет результат в указанный процесс.

10. MPI_Scatter() и MPI_Gather(), MPI_Barrier()

MPI_Scatter() используется для разделения данных между процессами, отправляя каждому процессу свою часть данных. MPI_Gather() собирает данные от всех процессов в один процесс. MPI_Barrier() синхронизирует выполнение процессов, гарантируя, что все процессы достигли определённой точки перед продолжением выполнения.

11. Сортировки и их распараллеливание

Распараллеливание сортировок (например, all2all, all2one) позволяет распределить задачу сортировки между несколькими процессами. Это снижает время выполнения задачи за счёт параллельной обработки данных.

12. Схемы взаимодействия процессов при сортировках

Процессы взаимодействуют друг с другом через передачу данных во время сортировки. Существуют схемы, такие как "блочная"и "циклическая которые определяют, как распределяются данные между процессами для оптимизации сортировки.

13. Схема взаимодействия типа "гиперкуб"

Схема взаимодействия типа "гиперкуб"представляет собой структуру, где каждый процесс соединён с другими процессами через логические "рёбра"в многомерном пространстве.Такая топология обеспечивает эффективный обмен сообщениями и синхронизацию.

14. Распараллеливание агрегации массивов

Агрегация массивов — это процесс объединения данных из нескольких источников. Распараллеливание агрегации позволяет выполнять эту операцию одновременно на нескольких процессах, что ускоряет выполнение задачи.

15. Закон Амдала

Закон Амдала описывает теоретическое ограничение на ускорение параллельных программ в зависимости от доли задачи, которая не может быть распараллелена. Чем больше серийной работы, тем меньше прирост скорости при добавлении дополнительных процессоров.

16. Закон Густавсона-Барсиса

Закон Густавсона-Барсиса предлагает другую модель, согласно которой производительность системы может улучшаться с увеличением количества процессоров, если задача масштабируется по данным. Это противоположный подход Закону Амдала.

17. Сортировка Бетчера

Сортировка Бетчера — это параллельный алгоритм сортировки, основанный на слиянии упорядоченных подмассивов. Он эффективно распределяет ресурсы между процессами, обеспечивая быструю сортировку данных.

18. Декомпозиция по данным

Декомпозиция по данным — это метод разделения задачи на части, которые обрабатываются разными процессами. Этот подход используется для распределённых вычислений, где каждый процесс работает с определённой частью данных.

19. Топологии, MPI_Cart_create() и MPI_Cart_coords()

MPI поддерживает топологии процессов в виде решёток (карты), где каждый процесс имеет координаты в пространстве. MPI_Cart_create() создаёт такую решётку, а MPI_Cart_coords() возвращает координаты процесса в этой решётке.

20. Топологии, MPI_Cart_sub() и MPI_Cart_rank()

MPI_Cart_sub() создаёт подрешётки из уже существующей решётки процессов, позволяя группировать процессы по подзадачам. MPI_Cart_rank() возвращает ранг процесса в топологии.

21. MPI_Cart_shift() и векторные операции

MPI_Cart_shift() используется для сдвига данных между соседними процессами в топологии, что полезно для организации потоков данных в векторных операциях и других задачах.

22. Декомпозиция неравномерных сеток

Неравномерные сетки требуют более сложной декомпозиции, поскольку процессы могут иметь разные объёмы работы. Распределение нагрузки между процессами в этом случае требует динамической балансировки.

23. Функции ожидания: MPI_Wait() и MPI_Test()

MPI_Wait() используется для ожидания завершения операции, такой как отправка или получение сообщения. MPI_Test() проверяет, завершена ли операция, не останавливая выполнение программы, что полезно для неблокирующих операций.

24. Графические ускорители

Графические ускорители (GPU) используются для параллельных вычислений, предоставляя множество ядер для одновременной обработки больших объёмов данных. Они широко применяются в задачах машинного обучения и вычислительной графике.

25. Технология CUDA

CUDA — это платформа для разработки программ на графических процессорах от NVIDIA. Она позволяет выполнять параллельные вычисления, используя архитектуру GPU, что значительно ускоряет вычислительные задачи.

26. Язык OpenCL

OpenCL — это стандарт, обеспечивающий параллельные вычисления на различных платформах, включая CPU, GPU и другие процессоры. Это универсальный язык для разработки кросс-платформенных параллельных программ.

27. Программируемые логические интегральные схемы (FPGA)

FPGA (Field-Programmable Gate Arrays) — это интегральные схемы, которые могут быть перепрограммированы для выполнения специфических задач параллельной обработки. Они используются в задачах, требующих

высокой производительности и низкой задержки, например, в обработке потоков данных в реальном времени.