

## Предел числовой [zorich]последовательности

Определение. Число  $A \in \mathbb{R}$  называется пределом числовой последовательности  $\{x_n\}$ , если для любой окрестности  $V(A)$  точки  $A$  существует такой номер  $N$  (выбираемый в зависимости от  $V(A)$ ), что все члены последовательности, номера которых больше  $N$ , содержатся в указанной окрестности точки  $A$ .

$$\left(\lim_{n \rightarrow \infty} x_n = A\right) := \forall V(A) \exists N \in \mathbb{N} \forall n > N (x_n \in V(A)) \quad (1)$$

и соответственно

$$\left(\lim_{n \rightarrow \infty} x_n = A\right) := \forall \varepsilon > 0 \exists N \in \mathbb{N} \forall n > N (|x_n - A| < \varepsilon). \quad (2)$$

## Предел функции

Определение. Итак, число  $A$  называется пределом функции  $f : E \rightarrow \mathbb{R}$  при  $x$ , стремящемся по множеству  $E$  к точке  $a$  (предельной для  $E$ ), если для любой окрестности точки  $A$  найдется проколота окрестность точки  $a$  в множестве  $E$ , образ которой при отображении  $f : E \rightarrow \mathbb{R}$  содержится в заданной окрестности точки  $A$ .

$$(\lim_{E \ni x \rightarrow a} f(x) = A) := \forall V_{\mathbb{R}}(A) \exists \dot{U}_E(a) (f(\dot{U}_E(a)) \subset V_{\mathbb{R}}(A)) \quad (3)$$

## Замечательные пределы

Первый замечательный предел:

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1 \quad (4)$$

Второй замечательный предел:

$$\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^x = e. \quad (5)$$

## Разложение функции в ряд Тейлора

$$P_n(x_0; x) = P_n(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \quad (6)$$

Определение. Алгебраический полином, заданный соотношением (6), называется полиномом Тейлора<sup>1</sup> порядка  $n$  функции  $f(x)$  в точке  $x_0$ .

Нас будет интересовать величина

$$f(x) - P_n(x_0; x) = r_n(x_0; x) \quad (7)$$

уклонение полинома  $P_n(x)$  от функции  $f(x)$ , называется часто остатком, точнее,  $n$ -м остатком или  $n$ -м остаточным членом формулы Тейлора:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + r_n(x_0; x) \quad (8)$$

Также давайте разложим наиболее часто используемые функции по формуле (8):

$$e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{n!}x^n + O(x^{n+1}) \quad (9)$$

$$\cos x = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots + \frac{(-1)^k}{2k!}x^{2k} + O(x^{2k+2}) \quad (10)$$

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \dots + \frac{(-1)^k}{(2k+1)!}x^{2k+1} + O(x^{2k+3}) \quad (11)$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \dots + \frac{(-1)^{n-1}}{n}x^n + O(x^{n+1}) \quad (12)$$

---

<sup>1</sup>Б. Тейлор (1685 - 1731) – английский математик

# Интеграл Римана

Определение. Функция  $f$  называется интегрируемой по Риману на отрезке  $[a, b]$ , если для нее существует указанный в пункте () предел интегральных сумм при  $\lambda(P) \rightarrow 0$  (т.е. если для нее определен интеграл Римана).

Множество всех функций, интегрируемых по Риману на отрезке  $[a, b]$ , будет обозначаться через  $\mathfrak{R}[a, b]$ .

$$\int_a^b f(x)dx := \lim_{\lambda(P) \rightarrow 0} \sum_{i=1}^n f(\xi_i) \Delta x_i$$

## Формула Тейлора

Теорема. Если функция  $f : U(x) \rightarrow \mathbb{R}$  определена и принадлежит классу  $C^{(n)}(U(x); \mathbb{R})$  в окрестности  $U(x) \subset \mathbb{R}^m$ , а отрезок  $[x, x+h]$  полностью содержится в  $U(x)$ , то имеет место равенство

$$\begin{aligned} & f(x^1 + h^1, \dots, x^m + h^m) - f(x^1, \dots, x^m) = \\ & = \sum_{k=1}^{n-1} \frac{1}{k!} (h^1 \delta_1 + \dots + h^m \delta_m)^k f(x) + r_{n-1}(x; h), \end{aligned}$$

где

$$r_{n-1}(x; h) = \int_0^1 \frac{(1-t)^{n-1}}{(n-1)!} (h^1 \delta_1 + \dots + h^m \delta_m)^n f(x + th) dt \quad (13)$$

## Интеграл по гладкой поверхности

Определение. (интеграла от  $k$ -формы  $\omega$  по заданной картой  $\varphi : I \rightarrow S$  гладкой  $k$ -мерной поверхности).

$$\int_S \omega := \lim_{\lambda(P) \rightarrow 0} \sum_i \omega(x_i)(\varepsilon_1, \dots, \varepsilon_k) = \lim_{\lambda \rightarrow 0} \sum_i (\varphi * \omega)(\tau_i)(\tau_1, \dots, \tau_k).$$

Если применить это определение к  $k$ -форме  $f(t)dt^1 \wedge \dots \wedge dt^k$  на  $I$  (когда  $\varphi$  – тождественное отображение), то очевидно, получим, что:

$$\int_I f(t)dt^1 \wedge \dots \wedge dt^k = \int_I f(t)dt^1 \dots dt^k.$$

Таким образом, из ( ) следует, что

$$\int_{S=\varphi(I)} \omega = \int_I \varphi * \omega, \quad (14)$$

а последний интеграл, как видно из равенства ( ), сводится к обычному кратному интегралу от соответствующей форме  $\varphi * \omega$  функции  $f$  на промежутке  $I$ .

## Формула Стокса в $\mathbb{R}^3$

Утверждение. Пусть  $S$  – ориентированная кусочно гладкая компактная двумерная поверхность с краем  $\delta S$ , лежащая в области  $G \subset \mathbb{R}^3$ , в которой задана гладкая 1-форма  $\omega = P dx + Q dy + R dz$ . Тогда имеет место соотношение

$$\begin{aligned} \int_{\delta S} P dx + Q dy + R dz = \iint_S \left( \frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z} \right) dy \wedge dz + \\ + \left( \frac{\partial P}{\partial z} - \frac{\partial R}{\partial x} \right) dz \wedge dx + \left( \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx \wedge dy, \end{aligned}$$

где ориентация края  $\delta S$  берется согласованной с ориентацией поверхности  $S$ .



## Алгебра форм

Пусть  $X$  - линейное пространство, а  $F^k : X^k \rightarrow \mathbb{R}$  - вещественнозначная  $k$ -форма на  $X$ . Если  $e_1, \dots, e_n$  - базис в  $X$ , а  $x_1 = x^{i_1} e_{i_1}, \dots, x_k = x^{i_k} e_{i_k}$  - разложение векторов  $x_1, \dots, x_k \in X$  по этому базису, то в силу линейности  $F_k$  по каждому аргументу

$$F^k(x_1, \dots, x_k) = F^k(x_1^{i_1} e_{i_1}, \dots, x_k^{i_k} e_{i_k}) = F^k(e_{i_1}, \dots, e_{i_k}) x^{i_1} \cdot \dots \cdot x^{i_k} = a_{i_1 \dots i_k} x^{i_1} \cdot x^{i_k}. \quad (15)$$

График функции  $f(x) = \sin \frac{1}{x}$



# Аксиоматика и некоторые общие свойства множества действительных чисел

Определение. Множество  $\mathbb{R}$  называется множеством действительных (вещественных) чисел, а его элементы – действительными (вещественными) числами, если выполнен следующий комплекс условий, называемый аксиоматикой вещественных чисел:

(I) Аксиомы сложения. Определено отображение (операции сложения)

$$+ : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

сопоставляющее каждой упорядоченной паре  $(x, y)$  элементов  $x, y$  из  $\mathbb{R}$  некоторый элемент  $x + y \in \mathbb{R}$ , называемый суммой  $x$  и  $y$ . При этом выполнены следующие условия:

1<sub>+</sub>. Существует нейтральный элемент 0 (называемый в случае сложения нулем):

$$\exists 0 \forall x : x + 0 = x.$$

2<sub>+</sub>. Для любого элемента  $x \in \mathbb{R}$  имеется элемент  $-x \in \mathbb{R}$ , называемый противоположным к  $x$ , такой, что

$$x + (-x) = (-x) + x = 0.$$

3<sub>+</sub>. Операция  $+$  ассоциативна, т.е. для любых элементов  $x, y, z \in \mathbb{R}$  выполнено

$$x + (y + z) = (x + y) + z.$$

4<sub>+</sub>. Операция  $+$  коммутативна, т.е. для любых элементов  $x, y \in \mathbb{R}$  выполнено

$$x + y = y + x$$

(II) Аксиомы умножения. Определено отображение (операция умножения)

$$\bullet : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R},$$

сопоставляющее каждой упорядоченной паре  $(x, y)$  элементов  $x, y \in \mathbb{R}$  некоторый элемент  $x \cdot y \in \mathbb{R}$ , называемый произведением  $x$  и  $y$ , причем так, что выполнены следующие условия:

1<sub>•</sub>. Существует нейтральный элемент  $1 \in \mathbb{R} \setminus 0$  (называемый в случае умножения единицей) такой, что

$$\exists 1 \forall x : x \cdot 1 = x.$$

2<sub>•</sub>. Для любого элемента  $x \in \mathbb{R} \setminus 0$  имеется элемент  $x^{-1} \in \mathbb{R}$ , называемый обратным, такой, что

$$x \cdot x^{-1} = x^{-1} \cdot x = 1.$$

3<sub>•</sub>. Операция  $\bullet$  ассоциативна, т.е. для любых  $x, y, z \in \mathbb{R}$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z.$$

4<sub>•</sub>. Операция  $\bullet$  коммутативна, т.е. для любых  $x, y \in \mathbb{R}$

$$x \cdot y = y \cdot x.$$

Заметим, что по отношению к операции умножения множество  $\mathbb{R} \setminus 0$ , как можно проверить, является (мультипликативной) группой.

## Умножение матриц

Теорема 1. Произведение  $\varphi A \varphi B$  двух линейных отображений с матрицами  $A$  и  $B$  является линейным отображением с матрицей  $C = AB$ . Другими словами

$$\varphi A \varphi B = \varphi AB. \quad (16)$$

Мы можем забыть о линейных отображениях и находить произведение  $AB$  двух произвольных матриц  $A, B$ , имея в виду, однако, что символ  $AB$  имеет смысл только в том случае, когда число столбцов в матрице  $A$  совпадает с числом строк в матрице  $B$ . Именно при этом условии выполняется правило умножения  $i$ -й строки  $A_{(i)}$  на  $j$ -й столбец  $B^{(j)}$ , согласно которому

$$A_{(i)} B^{(j)} = (a_{i1}, \dots, a_{is}) [b_{1j}, \dots, b_{sj}] \quad (17)$$

Следствие. Умножение матриц ассоциативно:

$$A(BC) = (AB)C \quad (18)$$

Действительно, произведение матриц соответствует произведению линейных отображений (теорема 1 и соотношение 16). К тому же результату можно прийти вычислительным путём, используя непосредственно соотношение (17).  $\square$

Обратим ещё внимание на так называемые законы дистрибутивности:

$$(A + B)C = AC + BC, \quad D(A + B) = DA + DB \quad (19)$$

где  $A, B, C, D$  – произвольные матрицы размеров соответственно  $m \times s, m \times s, s \times n, n \times m$ .

Действительно, полагая  $A = (a_{ij}), B = (b_{ij}), C = (c_{ij})$ , мы получим для любых  $i, j$  равенство (используя дистрибутивность в  $\mathbb{R}$ )

$$\sum_{k=1}^n (a_{ik} + b_{ik}) c_{kj} = \sum_{k=1}^n a_{ik} c_{kj} + \sum_{k=1}^n b_{ik} c_{kj}, \quad (20)$$

левая часть которого дает элемент  $g_{ij}$  матрицы  $(A + B)C$ , а правая – элементы  $h_{ij}$  и  $h'_{ij}$  матриц  $AC$  и соответственно  $BC$ . Второй закон дистрибутивности (19) проверяется совершенно аналогично.

## Транспонирование матриц

Будем говорить, что матрицы

$$A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{vmatrix}, \quad A^T = \begin{vmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{vmatrix} \quad (21)$$

размеров  $m \times n$  и  $n \times m$  соответственно получаются друг из друга транспонированием – заменой строк на столбцы, а столбцов на строки.

## Инварианты линейных групп

Линейной группой степени  $n$  мы, как обычно, называем любую подгруппу в  $GL(n, P)$ , где  $P$  – некоторое поле. В дальнейшем можно считать  $P = \mathbb{R}$  или  $P = \mathbb{C}$ . Если  $G$  – абстрактная группа и  $\Phi : G \rightarrow GL(n, \mathbb{C})$  – её линейное представление, то пару  $(G, \Phi)$  мы тоже будем называть линейной группой. Линейные преобразования  $\Phi_g$  действуют на столбцы переменных  $x_1, \dots, x_n$ :

$$\begin{pmatrix} \Phi_g(x_1) \\ \vdots \\ \Phi_g(x_n) \end{pmatrix} = \Phi_g \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad (22)$$

Они переводят любую форму (однородный многочлен)  $f$  степени  $m$  снова в форму степени  $m$ :

$$(\widetilde{\Phi_g} f)(x_1, \dots, x_n) = f(\Phi_{g^{-1}}(x_1), \dots, \Phi_{g^{-1}}(x_n)). \quad (23)$$

Определение. Форма  $f \in P_m$ , остающаяся неподвижной при действии  $\widetilde{\Phi_g}$  (т.е.  $\widetilde{\Phi_g} f = f \ \forall g \in G$ ), называется (целым) инвариантом степени  $m$  линейной группы  $(G, \Phi)$ .

# Начала тензорного исчисления

**1. Понятие то тензорах.** Разумной общности можно достичь, ограничившись лишь полилинейными отображениями некоторого специального вида.

Определение. Пусть  $\mathfrak{R}$  – поле,  $V$  – векторное пространство над  $\mathfrak{R}$ ,  $V^*$  – сопряженное к  $V$  пространство,  $p$  и  $q$  – целые числа  $\geq 0$ ,

$$V^p \times (V^*)^q = \underbrace{V \times \dots \times V}_p \times \underbrace{V^* \times \dots \times V^*}_q \quad (24)$$

– декартово произведение  $p$  экземпляров пространства  $V$  и  $q$  экземпляров пространства  $V^*$ . Всякое  $(p+q)$  – линейное отображение

$$f : V^p \times (V^*)^q \rightarrow \mathfrak{R} \quad (25)$$

называется тензором на  $V$  типа  $(p, q)$  и валентности (или ранга)  $p+q$ .

**2. Произведение тензоров.** Вначале пусть

$$f : V_1 \times \dots \times V_r \rightarrow \mathfrak{R} \quad g : W_1 \times \dots \times W_s \rightarrow \mathfrak{R} \quad (26)$$

– произвольные полилинейные формы. Это значит, что  $V_i, W_j$  – никак не связанные друг с другом векторные пространства.

Определение. Под тензорным произведением  $f$  и  $g$  понимают отображение

$$f \otimes g : V_1 \times \dots \times V_r \times W_1 \times \dots \times W_s \rightarrow \mathfrak{R}, \quad (27)$$

Определенное формулой

$$(f \otimes g)(v_1, \dots, v_r; w_1, \dots, w_s) = f(v_1, \dots, v_r)g(w_1, \dots, w_s). \quad (28)$$

Существенно подчеркнуть, что переменные  $V_i$  независимы от переменных  $W_j$ .

Резюмируем сказанное:

- 1 $\otimes$  операция умножения  $\otimes$  определена для тензоров произвольных типов;
- 2 $\otimes$  валентность произведения равна сумме валентностей сомножителей;
- 3 $\otimes$  тензорное произведение ассоциативно и дистрибутивно, но не коммутативно.

## Дифференциальное исчисление. Основные теоремы.

Теорема. Пусть  $f : U(x_0) \rightarrow \mathbb{R}$  – функция класса  $C^{(2)}(U(x_0); \mathbb{R})$ , определенная в окрестности  $U(x_0) \subset \mathbb{R}^m$  точки  $x_0 = (x_0^1, \dots, x_0^m) \in \mathbb{R}$ , и пусть  $x_0$  – критическая точка этой функции  $f$ .

Если в тейлоровском разложении

$$f(x_0^1 + h^1, \dots, x_0^m + h^m) = f(x_0^1, \dots, x_0^m) + \frac{1}{2!} \sum_{i,j=1}^m \frac{\delta^2 f}{\delta x^i \delta x^j}(x_0) h^i h^j + o(\|h\|^2) \quad (29)$$

функции в точке  $x_0$  квадратичная форма

$$\sum_{i,j=1}^m \frac{\delta^2 f}{\delta x^i \delta x^j}(x_0) h^i h^j \equiv \delta_{ij} f(x_0) h^i h^j \quad (30)$$

- а) знакоопределена, то в точке  $x_0$  функция имеет локальный экстремум, который является строгим локальным минимумом, если квадратичная форма (30) положительно определена, и строгим локальным максимумом, если она отрицательно определена;
- б) может принимать значения разных знаков, то в точке  $x_0$  функция экстремума не имеет.

◀ Пусть  $h \neq 0$  и  $x_0 + h \in U(x_0)$ . Представим соотношение (29) в виде

$$f(x_0 + h) - f(x_0) = \frac{1}{2!} \|h\|^2 \left[ \sum_{i,j=1}^m \frac{\delta^2 f}{\delta x^i \delta x^j}(x_0) \frac{h^i}{\|h\|} \frac{h^j}{\|h\|} + o(1) \right] \quad (31)$$

где  $o(1)$  есть величина, бесконечно малая при  $h \rightarrow 0$

Из (31) видно, что знак разности  $f(x_0 + h) - f(x_0)$  полностью определяется знаком величины, стоящей в квадратных скобках. Этой величиной мы теперь и займемся.



## Теорема о определителях квадратной матрицы

Теорема. Определители любой квадратной матрицы  $A$  и транспонированной к ней матрицы  $A^T$  совпадают:

$$\det A^T = \det A \quad (32)$$

Доказательство. Положив  $A = (a_{ij})$ ,  $A^T = (a'_{ij})$ , где  $a'_{ij} = a_{ji}$ , и заметив, что  $k = \pi(\pi^{-1}k)$  для любой перестановки  $\pi \in S_n$  и для любого номера  $k \in \{1, 2, \dots, n\}$ , мы видим, что упорядочение множителей произведения  $a'_{1,\pi_1} \dots a'_{n,\pi_n}$  в соответствии с перестановкой  $\pi^{-1}$  дает

$$a'_{1,\pi_1} \dots a'_{n,\pi_n} = a'_{\pi^{-1}1, \pi(\pi^{-1}1)} \dots a'_{\pi^{-1}n, \pi(\pi^{-1}n)} = a'_{\pi^{-1}1, 1} \dots a'_{\pi^{-1}n, n} = a_{1, \pi^{-1}1} \dots a_{n, \pi^{-1}n}. \quad (33)$$

Если учесть ещё, что  $\varepsilon_\pi = \varepsilon_{\pi^{-1}}(\varepsilon_\pi \varepsilon_{\pi^{-1}} = \varepsilon_{\pi\pi^{-1}} = \varepsilon_e = 1)$ , а  $\{\pi^{-1} \mid \pi \in S_n\} = \{\pi \mid \pi \in S_n\}$  (поскольку  $\pi \mapsto \pi^{-1}$ ) – биективное отображение из  $S_n$  в  $S_n$ ), то по формуле нахождения определителя матрицы имеем

$$\det A^T = \sum_{\pi \in S_n} \varepsilon_\pi a'_{1, \pi^{-1}1} \dots a'_{n, \pi^{-1}n} = \sum_{\sigma \in S_n} \varepsilon_\sigma a_{1, \sigma_1} \dots a_{n, \sigma_n} = \det A \quad (34)$$

# Организация стандартной библиотеки C++

Средства стандартной библиотеки определены в пространстве имен *std* и расположены в некотором наборе заголовочных файлов, реализующих большую часть этих средств. Перечисление этих заголовочных файлов дает представление о стандартной библиотеке и поясняет направление ее рассмотрение.

Ниже в данном разделе мы приводим список заголовочных файлов стандартной библиотеки, сгруппированный по функциональности, и сопровождаемый краткими пояснениями.

Стандартный заголовочный файл, начинающийся на букву *s*, эквивалентен соответствующему заголовочному файлу стандартной библиотеки языка C. Для каждого файла *<X.h>*, определяющего часть стандартной библиотеки языка C в глобальном пространстве имен и в пространстве имен *std*, имеется заголовочный файл *<cX>*, определяющий те же имена исключительно в пространстве имен *std*.

Контейнеры	
<i>&lt;array&gt;</i>	одномерный массив элементов <i>T</i> , в количестве <i>N</i>
<i>&lt;vector&gt;</i>	одномерный динамический массив элементов <i>T</i>
<i>&lt;list&gt;</i>	двусвязный список элементов <i>T</i>
<i>&lt;deque&gt;</i>	двусторонняя очередь элементов <i>T</i>
<i>&lt;stack&gt;</i>	стек элементов <i>T</i>
<i>&lt;map&gt;</i>	упорядоченный ассоциативный контейнер элементов <i>T</i>
<i>&lt;set&gt;</i>	множество элементов <i>T</i>
<i>&lt;bitset&gt;</i>	множество булевских переменных

Ассоциативные контейнеры *multimap* и *multiset* находятся в файлах *<map>* и *<set>*, соответственно. Контейнер *priority map* объявляется в *<queue>*.

# Комплексные числа

Подобно тому, как в области  $\mathbb{Q}$  рациональных чисел алгебраическое уравнение  $x^2 = 2$  не имело решений, уравнение  $x^2 = -1$  не имеет решений в области действительных чисел  $\mathbb{R}$ , и подобно тому, как вводя внешний по отношению к  $\mathbb{Q}$  символ  $\sqrt{2}$  в качестве решения уравнения  $x^2 = 2$ , мы увязываем его с операциями в  $\mathbb{Q}$  и получаем новые числа вида  $r_1 + \sqrt{2}r_2$ , где  $r_1, r_2 \in \mathbb{Q}$ , можно ввести символ  $i$  в качестве решения уравнения  $x^2 = -1$  и связать это внешнее по отношению к  $\mathbb{R}$  число  $i$  с действительными числами и арифметическими операциями в  $\mathbb{R}$ .

Реализуем теперь намеченную программу.

**а. Алгебраическое расширение поля  $\mathbb{R}$ .** Итак, вводим (следуя обозначению Эйлера) новое число  $i$  – мнимую единицу, такое что  $i^2 = -1$ .

Взаимодействие  $i$  с действительными числами должно состоять в том, что можно умножать  $i$  на числа  $y \in \mathbb{R}$ , т.е. необходимо появляются числа вида  $iy$ , и складывать такие числа с вещественными, т.е. появляются числа вида  $x + iy$ , где  $x, y \in \mathbb{R}$ .

Если мы хотим, чтобы на множестве объектов вида  $x + iy$ , которые мы вслед за Гауссом назовем *комплексными числами*, были определены привычные операции коммутативного сложения и коммутативного умножения, дистрибутивного относительно сложения, то необходимо положить по определению, что

$$(x_1 + iy_1) + (x_2 + iy_2) := (x_1 + x_2) + i(y_1 + y_2) \quad (35)$$

и

$$(x_1 + iy_1) \cdot (x_2 + iy_2) := (x_1x_2 - y_1y_2) + i(x_1y_2 + x_2y_1). \quad (36)$$

Два комплексных числа  $x_1 + iy_1, x_2 + iy_2$  считаются равными в том и только в том случае, когда  $x_1 = x_2$  и  $y_1 = y_2$ .

Отождествим числа  $x \in \mathbb{R}$  с числами вида  $x + i \cdot 0$ , а  $i$  – с числом  $0 + i \cdot 1$ . Роль нуля в множестве комплексных чисел, как видно из (35), играет число  $0 + i \cdot 0 = 0 \in \mathbb{R}$ , роль единицы, как видно из (36), – числа  $1 + i \cdot 0 = 1 \in \mathbb{R}$ .

Из свойства вещественных чисел и определений (35), (36) следует, что множество комплексных чисел является полем, содержащим  $\mathbb{R}$  в качестве подполя.

**б. Геометрическая интерпретация поля  $\mathbb{C}$ .** Комплексное число  $z = x + iy$  мы можем отождествить с упорядоченной парой  $(x, y)$  действительных чисел, называемых соответственно действительной частью и мнимой частью комплексного числа  $z$  (обозначения:  $x = \operatorname{Re} z, y = \operatorname{Im} z$ <sup>2</sup>)

Но тогда, считая пару  $(x, y)$  декартовыми координатами точки плоскости  $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ , можно отождествить комплексные числа с точками этой плоскости или с двумерными векторами с координатами  $(x, y)$ .

В такой векторной интерпретации покоординатное сложение (35) комплексных чисел соответствует правилу сложения векторов. Кроме того, такая интерпретация естественно приводит также к понятию модуля  $|z|$  комплексного числа  $z$  как модуля или длины соответствующего ему вектора  $(x, y)$ , т.е.

$$|z_1 - z_2| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (37)$$

---

<sup>2</sup>От лат. *realis* (вещественный) и *imaginarius* (мнимый).

## Таблица производных основных функций

Функция $f(x)$	Производная $f'(x)$	Ограничения на область изменения аргумента $x \in \mathbb{R}$
1. $C$ (const)	0	
2. $x^a$	$ax^{a-1}$	$x > 0$ при $a \in \mathbb{R}$ $x \in \mathbb{R}$ при $a \in \mathbb{N}$
3. $a^x$	$a^x \ln a$	$x \in \mathbb{R} (a > 0, a \neq 1)$
4. $\log_a  x $	$\frac{1}{x \ln a}$	$x \in \mathbb{R} \setminus 0 (a > 0, a \neq 1)$
5. $\sin x$	$\cos x$	
6. $\cos x$	$-\sin x$	
7. $\tan x$	$\frac{1}{\cos^2 x}$	$x \neq \frac{\pi}{2} + \pi k, \quad k \in \mathbb{Z}$
8. $\operatorname{ctg} x$	$-\frac{1}{\sin^2 x}$	$x \neq \pi k, \quad k \in \mathbb{Z}$

## Хранение одного из нескольких выбранных типов в контейнере или переменной

Объединения (union) C++03 могут содержать только очень простые типы под названием **простая структура данных (POD)**. Например в C++03 нельзя хранить `std::string` или `std::vector` в объединении.

Вы знаете о концепции **Неограниченных объединений (unrestricted unions)** в C++11? Позвольте мне кратко рассказать вам о них. C++11 ослабляет требования для объединений, но вы должны сами управлять созданием и уничтожением не-POD-типов. Вы должны вызывать конструирование или уничтожение по месту (in-place construction/destruction) и запомнить, какой тип хранится в объединении. Огромный объем работы, не так ли?

Можно ли в C++03 получить переменную, которая ведет себя как неограниченное объединение C++ и которая управляет временем жизни объекта, запоминая его тип?

### Подготовка...

Мы будем работать с библиотекой header-only, которая проста в использовании. Все, что требуется для этого рецепта, - базовые знания C++.

### Как это делается...

Позвольте представить вам библиотеку `Boost.Variant`.

1. Библиотека `Boost.Variant` может хранить любые типы, указанные во время компиляции. Она также управляет созданием или уничтожением по месту, и ей даже не требуется стандарт C++11:

```
#include <boost/variant.hpp>
#include <iostream>
#include <vector>
#include <string>

int main() {
    typedef boost::variant<int, const char* std::string> my_var_t;
    std::vector<my_var_t> some_values;
    some_values.push_back(10);
    some_values.push_back("Hello, _there!");
    some_values.push_back(std::string("Wow!"));

    std::string& s = boost::get<std::string>(some_values.back());
    s += "_That_is_great\n";

    std::cout << s << '\n';
    return 0;
}
```

Здорово, правда?

2. `Boost.Variant` не имеет пустого состояния, но у нее есть функция `empty()`, которая бесполезна и всегда возвращает значение `false`. Если вам нужно представить пустое состояние, просто добавьте простой тип первым шаблонным параметром

`boost::variant`. Если `Boost.Variant` содержит этот тип, интерпретируйте его как пустое состояние. Вот пример, в котором мы будем использовать тип `boost::blank` для представления пустого состояния:

```
void example1() {
    // The default constructor creates an instance of boost::blank.
    boost::variant<boost::blank, int, const char*, std::string> var;

    // The which() method returns the index of the type currently
    // contained in the variant.
    assert(var.which() == 0); // boost::blank

    var = "Hello, _dear_reader";
    assert(var.which() != 0);
}
```

3. Можно получить значение из `boost::variant`, используя два подхода:

```
void example2() {
    boost::variant<int, std::string> variable(0);

    //When using the method below, an exception may be thrown
    // boost::bad_get, if the actual value in variable is not int
    int s1 = boost::get<int>(variable);

    // If the actual value in the variable is not an int,
    // NULL will be returned
    int* s2 = boost::get<int>(&variable);
}
```

## Как это работает...

Класс `boost::variant` содержит массив байтов и хранит значения в этом массиве. Размер массива определяется во время компиляции путем применения функции `sizeof` и функции для определения выравнивания (alignment) каждого из типов шаблонов. При присваивании или создании класса `boost::variable` предыдущее значение уничтожается по месту, а новое значение создается поверх массива байтов с использованием оператора `placement new`.

## Дополнительно...

`Boost.Variant` обычно не выделяет память динамически и не требует RTTI. Это чрезвычайно быстрая библиотека, и она широко используется другими библиотеками Boost. Для достижения максимальной производительности убедитесь, что в шаблонном списке типов в первой позиции указан простой тип (POD), `boost::variant` использует rvalue-ссылки C++11, если они доступны в вашем компиляторе.

Библиотека `Boost.Variant` является частью стандартна c++17, `std::variant` имеет некоторые отличия от `Boost.Variant`

## Вставка картинок



Рис. 1: Картинка из статьи про SSH

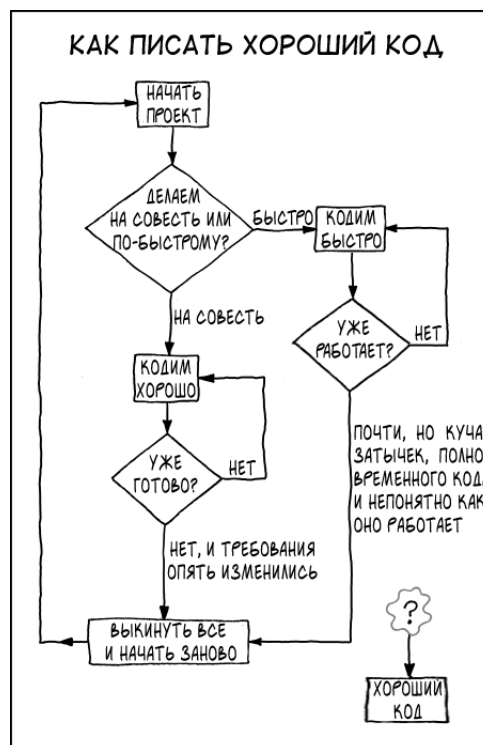


Рис. 2: Картинка о том, как писать хороший код :D

# Прямоугольная матрица

Пусть есть два конечных множества:

- Номера строк:  $M = \{1, 2, \dots, m\}$ ;
- номера столбцов  $N = \{1, 2, \dots, n\}$ , где  $m, n \in \mathbb{N}$ .

Назовём матрицей  $A$  размера  $m \times n$  (читается  $m$  на  $n$ ) ( $m$  - **строк**,  $n$  - **столбцов**) с элементами из некоторого кольца или поля  $\mathcal{K}$  отображение вида  $A : M \times N \rightarrow \mathcal{K}$ . Матрица записывается как

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & a_{ij} & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad (38)$$

где элемент матрицы  $a_{ij} = a(i, j)$  находится на пересечении  $i$ -й строки и  $j$ -го столбца.

- $i$ -я строка матрицы  $A(i, ) = (a_{i1} \ a_{i2} \ \cdots \ a_{in})$ ;

- $j$ -й столбец матрицы  $A(, j) = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix}$ .

При этом количество элементов матрицы равно  $m \cdot n$ .

- каждую строку матрицы можно интерпретировать как вектор в  $n$ -мерном координатном пространстве  $\mathcal{K}^n$ ;
- каждый столбец матрицы – как вектор в  $m$ -мерном координатном пространстве  $\mathcal{K}^m$ .

Сама матрица естественным образом интерпретируется как вектор в пространстве  $\mathcal{K}^{mn}$ , имеющем размерность  $mn$ . Это позволяет ввести покомпонентное сложение матриц и умножение матрицы на число (см. ниже); что касается матричного умножения, то оно существенным образом опирается на прямоугольную структуру матрицы.

## Обозначения

Если необходимо дать развёрнутое представление матрицы в виде таблицы, то используют запись вида

$$\begin{pmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{pmatrix}, \quad \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}, \quad \left\| \begin{array}{ccccc} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{array} \right\|$$

Можно встретить как обозначения с круглыми скобками «(...)», так и обозначения с квадратными скобками «[...]». Реже можно встретить обозначения с двойными прямыми линиями «||...||».



# Рекуррентное перечисление файлов в каталоге

Существуют функции и классы стандартной библиотеки для чтения и записи данных в файлы. Но до появления C++17 в ней не было функций вывода списка файлов в каталоге, получения типа файла или получения прав доступа к файлу.

Давайте посмотрим, как можно исправить эту несправедливость с помощью Boost. Мы будем создавать программу, которая перечисляет имена файлов, находящихся в выбранной директории, а также в её подкаталогах.

## Подготовка

Знание основ C++ более чем достаточно для использования этого рецепта. Этот рецепт требует линковки с библиотеками `boost_system` и `boost_filesystem`.

## Как это делается?

Этот рецепт посвящен переносимым оберткам для работы с файловой системой.

```
#include <iostream>
#include <boost/filesystem.hpp>

int main() {
    boost::filesystem::recursive_directory_iterator begin("./");
    boost::filesystem::recursive_directory_iterator end;

    for (; begin != end; begin++) {
        std::cout << *begin << std::endl;
    }
    return 0;
}
```

Готово. Теперь, если мы запустим программу, она выведет что-то вроде этого:

```
"/a.out"
"/ex1.cpp"
```

## Как это работает...

Функции и классы `Boost.Filesystem` просто оборачивают системные вызовы для работы с файлами.

Обратите внимание на использование знака `"/."`. Системы POSIX используют косую черту для указания путей; Windows по умолчанию использует обратную косую черту. Тем не менее Windows также понимает косую черту, а даже если бы не понимала, то библиотека Boost позаботилась бы о неявном преобразовании формата пути.

## Дополнительно...

`Boost.Filesystem` является частью C++17. Все содержимое в C++17 находится в одном заголовочном файле `<filesystem>` в пространстве имен `std::filesystem::`. Версия `<filesystem>` стандартной библиотеки несколько отличается от Boost-версии, в основном за счет использования перечислений с областью видимости (`enum class` там), где Boost.Filesystem использовала просто перечисления без области видимости.

## Список литературы

- [1] Владимир Антонович Зорич. *Математический анализ. Часть I.* 2020, с. 564. ISBN: 978-5-4439-4030-4.
- [2] Владимир Антонович Зорич. *Математический анализ. Часть II.* 2019, с. 675. ISBN: 978-5-4439-1303-2.
- [3] Алексей Иванович Кострыкин. *Введение в алгебру. Основы Алгебры. Часть I.* 2020, с. 272. ISBN: 978-5-4439-4116-5.
- [4] Антон А. Полухин. *Разработка приложений на C++ с использованием Boost. Рецепты, упрощающие разработку вашего приложения.* 2020, с. 346. ISBN: 987-5-97060-868-5.
- [5] Бьерн Страуструп. *Язык программирования C++.* 2020, с. 1136. ISBN: 978-5-7989-0425-9.