# Применение алгоритмов пространственного разбиения в задачах вычислительной геометрии

*Д.А. Гладких, Э.М. Вихтенко*

*Тихоокеанский государственный университет, Хабаровск*

**Аннотация:** Рассмотрены и исследованы, применительно к задаче моделирования обтекания воздухом тела сложной формы, алгоритмы и структуры пространственного разбиения: kd-дерево, BVH. Использование данных алгоритмов позволяет существенно сократить время вычислений при поиске столкновений воздушных частиц между собой и с обтекаемым телом.
**Ключевые слова:** обнаружение столкновений, вычислительная геометрия, алгоритмы и структуры данных, kd-дерево, BVH-дерево.

## Введение

Вычислительная геометрия – это раздел геометрии, изучающий использование численных методов для решения планиметрических и стереометрических задач. Возможности, предоставляемые вычислительной геометрией, используются преимущественно при организации вычислений для задач, в которых нельзя или очень трудоемко получить аналитическое решение, в том числе, в системах физического моделирования, в компьютерных играх, в графических приложениях и других [1].

В данной работе рассматриваются алгоритмы пространственного разбиения для решения задачи моделирования обтекания тела сложной формы воздушным потоком. Моделирование процессов, связанных с расчетами обтекания тел в жидкой или воздушной среде, является актуальной проблемой при решении прикладных инженерных задач [2-4].

## Постановка задачи

В разрабатываемой авторами системе моделирования физических процессов требуется рассчитать коэффициент сопротивления воздуха при движении тела сложной формы. Исходными данными задачи является форма и скорость движения тела, а также физические параметры окружающей

среды – температура воздуха, влажность и давление. В дальнейших расчетам зададим следующие условия: давление 101325 Па, температура 273 К, влажность воздуха 0%.

Для численного моделирования используется метод дискретных элементов [5], заключающийся в данной задаче в разбиении сплошной среды (воздуха) на систему частиц $\omega_i$ ($i = 1,2,\dots,n; n$ – число частиц), каждая из которых является сферой с заданными массой и радиусом. В таком случае математическая модель взаимодействия тела и воздуха значительно упрощается: вместо системы уравнений Навье-Стокса можно рассмотреть задачу Коши для системы из $n$ дифференциальных уравнений, являющихся записью второго закона Ньютона для частицы $\omega_i$,

$$\vec{F}_i = m_i \frac{d\vec{v}_i}{dt} = m_i \frac{d^2\vec{r}_i}{dt^2}, \quad i = 1,2,\dots,n, \tag{1}$$

где $\vec{F}_i$ – вектор равнодействующих сил, действующих на частицу $\omega_i$; $m_i$ – масса частицы $\omega_i$; $\vec{v}_i$ – вектор скорости $\omega_i$; $\vec{r}_i$ – вектор перемещения $\omega_i$; $t$ – время. Пусть все воздушные частицы одинакового размера, $m_i = m$ для всех $i = 1,2,\dots,n$.

Система (1) решается методом Рунге-Кутты. На $k$-ом шаге метода требуется задать величины сил $\vec{F}_i(t_k)$ для каждой частицы $\omega_i$, которые вычисляются, как результат взаимодействия (абсолютно упругого удара) частицы $\omega_i$ с другими частицами воздуха, а также с обтекаемым телом.

Одной из основных сложностей при реализации вычислений является необходимость проверки частиц на столкновение друг с другом и с обтекаемым телом. Поскольку при использовании простого перебора

возможных пар $(\omega_i, \omega_j)$ из $n$ объектов на каждом шаге метода Рунге-Кутты будет выполнено $n(n-1)/2$ проверок на столкновение, а количество частиц в симуляции исчисляется сотнями тысяч, то это ведёт к быстрой деградации производительности при увеличении количества объектов.

Зависимость среднего времени выполнения шага метода Рунге-Кутты от количества частиц приведена на рис. 1. Зелёным цветом отмечены точки, нанесённые на график в результате аппроксимации исходной зависимости, так как реальное время выполнения расчетов для большого числа частиц измерить не удается. Из графика видно, что уже при 10000 частиц один шаг выполняется за 4 секунды.

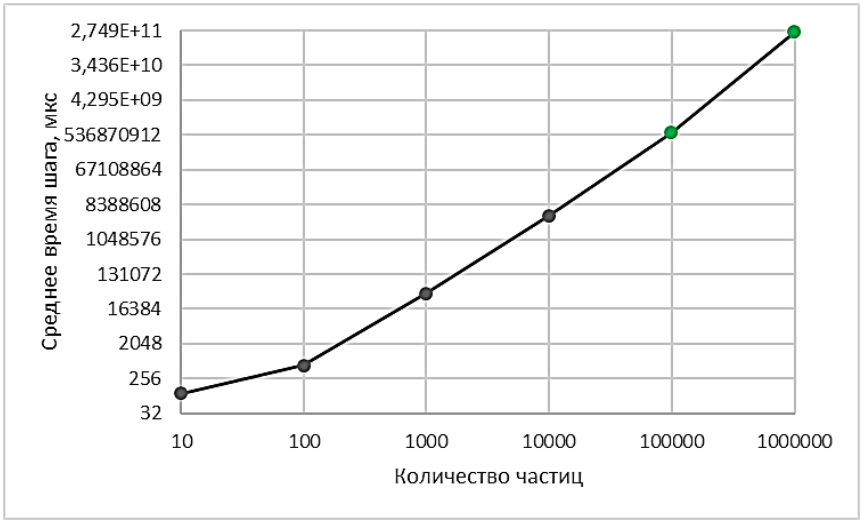


Рис. 1. – Зависимость среднего времени шага метода Рунге-Кутты от количества частиц

Повышение скорости выполнения расчётов требует снижения числа проверок на столкновение частиц, для чего могут быть использованы алгоритмы пространственного разбиения [6, 7].

## Теоретические основы

Идея пространственного разбиения исходит из теории множеств [8]. Если во множестве присутствует *n* объектов, и эти объекты удается разбить на два непересекающихся подмножества, то между элементами этих

подмножеств нет необходимости в выполнении перекрёстных проверок на пересечение. Если рассматривать идеальный случай, когда в каждом подмножестве находится равное количество элементов, тогда количество проверок $N$ будет равным:

$$N = 2\left[\frac{\frac{n}{2}\left(\frac{n}{2}-1\right)}{2}\right] = \frac{n^2}{4} - \frac{n}{2}, \quad (2)$$

что в

$$\frac{2n^2-2n}{n^2-2n} = 2\left[\frac{n}{n^2-2n}+1\right] = 2\left[\frac{1}{n-2}+1\right], \quad (3)$$

раз меньше, чем без использования разбиения. Из соотношения (3) видно, что предложенный способ даёт выигрыш как минимум в 2 раза, когда $n \geq 2$.

Продолжив деление объектов на непересекающиеся подмножества, можно добиться такого состояния, когда в каждой группе будет содержаться лишь малое количество элементов, которые можно проверить на столкновение простым перебором. Очевидно, что данный способ позволяет сохранить точность, уменьшив количество вычислений.

**Грубая проверка на столкновение. Ограничивающий объём**

Увеличить производительность системы можно, сначала грубо оценив потенциальные пары столкновения. Для реализации таких проверок необходимо упростить геометрическую форму объекта. Для этого вокруг исходного тела описывается тело примитивной геометрической формы: прямоугольный параллелепипед, сфера, эллипсоид и так далее [9]. Когда все объекты представлены одним примитивом, вычислительная сложность грубой проверки сильно падает.

Пусть в качестве примитива выбран прямоугольный параллелепипед, выровненный по осям координат. Для того, чтобы задать такой параллелепипед, используются четыре параметра: радиус-вектор некоторой

его базовой точки (центра) и половины длин его сторон. Преимуществом использования данного примитива является простая проверка. Два таких параллелепипеда пересекаются, если выполнены условия:

$$\begin{cases} |A_{c_x} - B_{c_x}| \le A_{h_x} + B_{h_x} \\ |A_{c_y} - B_{c_y}| \le A_{h_y} + B_{h_y} \\ |A_{c_z} - B_{c_z}| \le A_{h_z} + B_{h_z} \end{cases} \quad (4)$$

где $A_{c_\bullet}$, $B_{c_\bullet}$ – координаты точек центра первого и второго прямоугольных параллелепипедов соответственно; $A_{h_\bullet}$, $B_{h_\bullet}$ – половина длин сторон первого и второго прямоугольных параллелепипедов соответственно.

Очевидно, что при выборе частицы в форме сферы все параллелепипеды можно задать как кубы одинакового размера.

## Kd-дерево

Одна из алгоритмических структур пространственного разбиения – это kd-дерево [10, 11]. Название «kd» расшифровывается как «k-мерный», что отражает суть структуры: это дерево поиска, в котором поиск осуществляется по нескольким измерениям.

Исходное пространство разбивается плоскостью на несколько подпространств, являющихся узлами дерева, в которые и попадают частицы $\omega_i$. Но возникает проблема неоднозначности отнесения объектов, пересекаемых плоскостью, к одному узлу. Такая проблема решается следующим образом: либо при разбиении объект добавляется во все узлы (техника называется «object split» – разбиение по объекту), либо ограничивающий объём одного из узлов расширяется, чтобы включить этот объект целиком («spatial split» – разбиение по пространству), в него и будет добавлен проблемный объект. Каждый из методов имеет свои достоинства и недостатки: в первом случае увеличивается скорость проверки на

столкновения, но растёт и расход памяти. Второй случай решает проблему с памятью, поскольку не происходит дублирования объектов, однако стоимость использования дерева для проверки на столкновение также выше. В работе используется способ с добавлением проблемных объектов в каждый узел.

Использование ускоряющих алгоритмов с техникой разбиения по объекту может привести к тому, что в оба узла попадёт пара пересекающихся геометрических тел, результатом чего непременно станет то, что на этапе прохода по дереву в поисках столкнувшихся элементов данная пара будет добавлена в список проверяемых объектов несколько раз, что, в случае положительного результата, в свою очередь, приводит к неправильным результатам моделирования. Данная проблема решается использованием при реализации алгоритма структуры «множество» (набор неповторяющихся элементов).

Классический алгоритм построения kd-дерева выглядит следующим образом (в работе используются двоичные деревья из-за удобства их использования):

1. Добавить в корневой узел все объекты, вычислить ограничивающий объём, добавить корневой узел в стек.
2. Пока стек не пуст:
   2.1. Достать из стека вершину.
   2.2. Если выполнен критерий остановки деления (об этом будет изложено далее), то перейти к шагу 2.
   2.3. Используя какую-нибудь эвристику, выбрать разбивающую плоскость (этот вопрос рассматривается далее), рассчитать ограничивающий объём по обе стороны от плоскости.
   2.4. Добавить объекты, пересекающиеся с левым ограничивающим телом, в левый узел текущей вершины.

2.5. Если в левом узле нет объектов, очистить память, занимаемую узлом. Иначе добавить левый узел в стек.

2.6. Добавить объекты, пересекающиеся с правым ограничивающим телом, в правый узел текущей вершины.

2.7. Если в правом узле нет объектов, очистить память, занимаемую узлом. Иначе добавить правый узел в стек.

2.8. Для экономии памяти очистить память, занимаемую текущим узлом.

Критерием остановки деления здесь может выступать количество объектов в узле, глубина дерева, измерения ограничивающего объёма. В работе применяются первый и третий критерии.

Существует множество эвристик, с помощью которых выбирается плоскость разбиения. Среди них стоит выделить:

1. разбиение по центру плоскостью, нормалью к которой является ось, вдоль которой размер ограничивающего объёма максимален;
2. разбиение с чередованием разбивающих плоскостей: смена плоскости на каждом шаге либо на каждом уровне дерева;
3. разбиение по медиане, т.е. разбиение с использованием сортировки и дальнейшего выбора объекта посередине выборки;
4. использование математического аппарата: минимизация функции стоимости поиска в дереве (т.н. «эвристика площади поверхности», англ. «Surface area heuristic» – SAH).

Реализация первых трёх пунктов очевидна. Четвёртый же требует дополнительных пояснений.

Критерием хорошо построенного дерева в отличие от его сбалансированности здесь является функция стоимости поиска, определённая, как:

$$s = s_0 + \frac{S_l n_l + S_r n_r}{s_p}, \quad (5)$$

где $s$ – SAH; $s_0$ – стоимость поиска вершины как листа; $S_l$, $S_r$ – площадь поверхности ограничивающего объёма левого и правого поддеревьев соответственно; $n_l$, $n_r$ – количество объектов в левом и в правом поддеревьях соответственно; $s_p$ – вычисленное значение эвристики площади поверхности для родительского узла.

Анализируя данную функцию, можно прийти к выводу, что если удастся найти её минимум, то в таком случае большое количество не занятого объектами пространства будет убрано, что повысит скорость поиска в таком дереве.

Для нахождения минимума функции *s* можно рассматривать наиболее удаленные объекты, но в этом случае придётся использовать сортировку объектов, что существенно замедлит процесс вычислений. Можно разбить пространство на несколько интервалов, что немного уменьшит качество построенного дерева, но значительно сократит скорость его построения по сравнению с первым рассмотренным вариантом.

## BVH-дерево

Другой разновидностью структур пространственного разбиения является иерархия ограничивающих объёмов (*англ.* Boundary volume hierarchy – BVH [12, 13]). Принцип построения такого дерева отличается от используемого при построении kd-дерева тем, что в отличие от последнего сначала с помощью какой-нибудь эвристики выбирается то, какие объекты попадут в каждый узел исходной вершины, а потом вокруг них описывается ограничивающий объём. Очевидным преимуществом такого способа является экономия вычислительных ресурсов, затрачиваемых на проверку попадания объекта в заранее построенный ограничивающий объём. Этот ограничивающий объём всё равно придётся вычислить, но для этого

необходимо в нашем случае вычислить границы прямоугольного параллелепипеда с гранями, перпендикулярными осям координат, а не проверить дважды за итерацию цикла пересечение двух таких параллелепипедов.

Техники построения рассмотренных деревьев практически идентичны, отличия заключаются в том, что либо список объектов просто делится пополам, либо сначала используется сортировка, а потом деление, либо критерием также выступает функция SAH.

## Вычислительные эксперименты

На рис. 2 представлены результаты выполнения расчетов при использовании различных алгоритмов пространственного разбиения. Для тестирования генерировались тесты со случайными данными, время указано, как среднее время по десяти тестам.

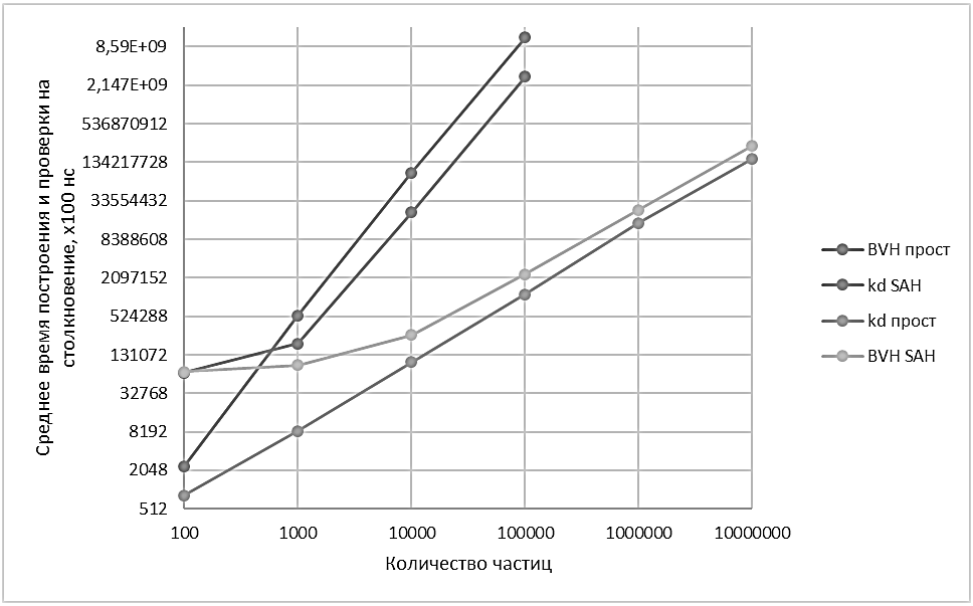


Рис. 2. – Сравнение времени построения деревьев пространственного разбиения

На рисунке отсутствуют серии, связанные с kd- и BVH-деревьями с сортировкой, поскольку из-за использования функций сортировки, имеющей сложность порядка $O(n \log n)$, для большого количества частиц, время,

затраченное на построение такого дерева, не оправдывается скоростью поиска в нём. По результатам тестирования самым лучшим в плане производительности оказался вариант kd-дерева с простым делением пространства пополам.

Сравнение времени выполнения шага метода Рунге-Кутты решения исходной системы с использованием и без использования оптимизации приведено на рис. 3. Здесь синим цветом изображено среднее время выполнения расчетов без использования алгоритмов пространственного разбиения, оранжевым – время после выполненной оптимизации. Из графика видно, что прирост скорости имеет логарифмическую зависимость. Интересным является то, что в случае 10 и 100 частиц время во втором случае больше, чем в случае простого перебора. Это связано с использованием алгоритма пространственного разбиения, который каждый раз пытается разделить количество частиц на группы. Таким образом, в ходе данного теста была получена граница применения алгоритма: делить группу размером до 100 частиц становится невыгодно.

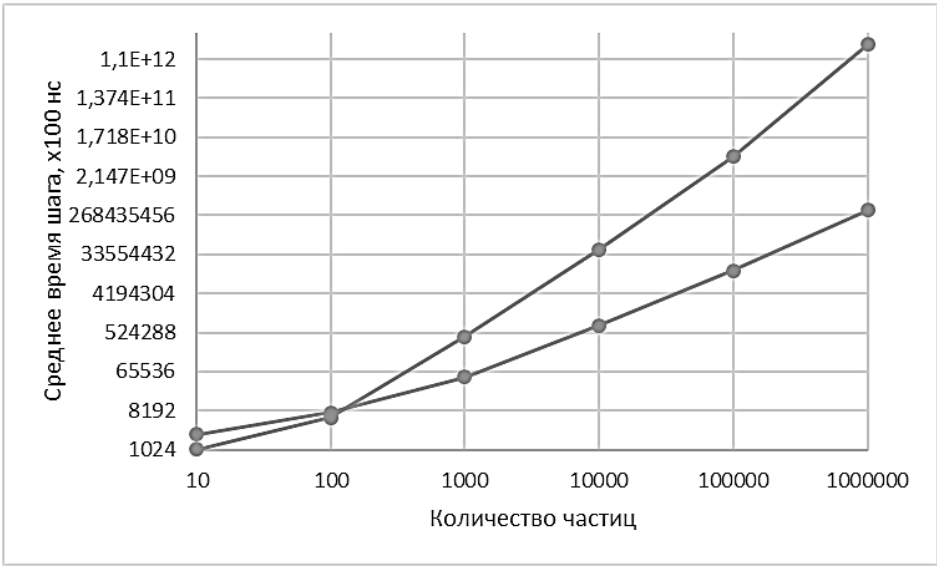


Рис. 3. – Сравнение среднего времени выполнения шага метода Рунге-Кутты с использованием и без использования оптимизации

## Заключение

Полученные в ходе проведенных вычислительных экспериментов результаты позволяют выбрать оптимальные, на взгляд авторов, методы для реализации численного моделирования в решении задачи обтекания воздушным потоком тела сложной структуры. По результатам тестирования выбор сделан в пользу алгоритма, использующего kd-дерево с простым делением пространства пополам.

## Благодарность за финансовую поддержку работы

*Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации, дополнительное соглашение № 075-02-2023-932 от 16 февраля 2023 г.*

## Литература

1. Беляков Д. В. Геометрический анализ областей неоднозначности угла атаки в задаче о движении аэродинамического маятника в потоке квазистатической среды // Инженерный вестник Дона, 2023, № 3. URL: ivdon.ru/ru/magazine/archive/n3y2023/8257.

2. Карсян А. Ж., Цуриков А. Н. Разработка алгоритма и программного приложения для реализации математической модели воздействия потока жидкости на тело // Инженерный вестник Дона, 2017, № 1. URL: ivdon.ru/ru/magazine/archive/n1y2017/3986.

3. Денисихина Д. М., Иванова Ю. В., Мокров В. В. Численное моделирование истечения из современных воздухораспределительных устройств // Инженерный вестник Дона, 2018, № 2. URL: ivdon.ru/ru/magazine/archive/N2y2018/4972.

4. Волков К. Н., Гимадиев В. А., Добров Ю. В., Карпенко А. Г. Численное моделирование гиперзвукового обтекания полусферы с учетом неравновесных физико-химических процессов в высокотемпературном

воздухе // Вычислительные методы и программирование, 2022, Т. 23, № 3. С. 248-274. URL: num-meth.ru/index.php/journal/article/view/1211/1199.

5. Shwartz S. R., Richardson D. C., Michel P. An implementation of the soft-sphere discrete element method in a high-performance parallel gravity tree-code // Granular Matter, 2012, № 14. Pp. 363–380.

6. Барышникова М. Ю., Брянская Е. В., Иванов В. А. Исследование методов разбиения пространства с целью ускорения работы алгоритмов обнаружения столкновений движущихся объектов // Школа Науки, 2021, № 6. С. 13-18.

7. Лысых А. И., Кинев И. Е., Жданов Д. Д. Оптимизация многоуровневой ускоряющей структуры пространственного разбиения хеш-таблиц и kd-дерева // Труды Международной конференции по компьютерной графике и зрению "Графикон", 2023, № 33. С. 125-138. URL: graphicon.ru/html/2023/papers/paper_012.pdf

8. Майника Э. Алгоритмы оптимизации на сетях и графах. М.: Мир, 1981, 324 с.

9. Wald I. Realtime Ray Tracing and Interactive Global Illumination / PhD thesis. Hamburg: Saarland University, 2004. 311 p.

10. Bentley J. L., Friedman J. H. Data structures for range searching // ACM Computing Surveys (CSUR), 1979, Vol. 11, № 4. P. 397-409.

11. Chen Q. P., Xue B., Siepmann J. I. Using the k-d tree data structure to accelerate Monte Carlo simulations // Journal of Chemical Theory and Computation, 2017, Vol. 13, № 4. P. 1556-1565.

12. Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989. 272 с.

13. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. М.: Диалектика, 2020. 1328 с.

## References

1. Beljakov D. V. Inzhenernyj vestnik Dona, 2023, № 3. URL: ivdon.ru/ru/magazine/archive/n3y2023/8257.

2. Karsjan A. Zh., Curikov A. N. Inzhenernyj vestnik Dona, 2017, № 1. URL: ivdon.ru/ru/magazine/archive/n1y2017/3986.

3. Denisihina D. M., Ivanova Ju. V., Mokrov V. V. Inzhenernyj vestnik Dona, 2018, № 2. URL: ivdon.ru/ru/magazine/archive/N2y2018/4972.

4. Volkov K. N., Gimadiev V. A., Dobrov Ju. V., Karpenko A. G. Vychislitel'nye metody i programmirovanie, 2022, T. 23, № 3. pp. 248-274. URL: num-meth.ru/index.php/journal/article/view/1211/1199.

5. Shwartz S. R., Richardson D. C., Michel P. Granular Matter, 2012, № 14. Pp. 363–380.

6. Baryshnikova M. Ju., Brjanskaja E. V., Ivanov V. A. Shkola Nauki, 2021, № 6. pp. 13-18.

7. Lysyh A. I., Kinev I. E., Zhdanov D. D. Trudy Mezhdunarodnoj konferencii po komp'juternoj grafike i zreniju "Grafikon", 2023, № 33. pp. 125-138. URL: graphicon.ru/html/2023/papers/paper_012.pdf

8. Majnika Je. Algoritmy optimizacii na setjah i grafah [Optimization algorithms on networks and graphs]. M.: Mir, 1981, 324 s.

9. Wald I. Realtime Ray Tracing and Interactive Global Illumination / PhD thesis. Hamburg: Saarland University, 2004. 311 p.

10. Bentley J. L., Friedman J. H. ACM Computing Surveys (CSUR), 1979, Vol. 11, № 4. pp. 397-409.

11. Chen Q. P., Xue B., Siepmann J. I. Journal of Chemical Theory and Computation, 2017, Vol. 13, № 4. pp. 1556-1565.

12. Virt N. Algoritmy i struktury dannyh [Algorithms and data structures]. M.: Mir, 1989. 272 p.

13. Kormen T., Lejzerson Ch., Rivest R., Shtajn K. Algoritmy: postroenie i analiz [Algorithms: construction and analysis]. M.: Dialektika, 2020. 1328 p.

**Дата поступления: 16.11.2023**

**Дата публикации: 3.01.2024**