**Список вопросов к экзамену по дисциплине ОАиП**

1. Понятие сложности алгоритма, оценки времени исполнения.

Понятие сложности алгоритма является ключевым в области компьютерных наук, так как оно помогает оценить, насколько быстро или медленно выполняется алгоритм, а также понять, как изменяется время выполнения алгоритма при изменении размера входных данных.

Важно отметить, что оценка сложности алгоритма по времени может быть приблизительной и зависит от многих факторов, таких как используемый язык программирования, аппаратное обеспечение, оптимизация кода и другие.

Дело в том, что **время выполнения написанной программы зависит от переданных входных данных.**

Вот две сложности, связанные с программированием.

* **Временная**: время, необходимое для обработки входных данных.
* **Пространственная**: количество места, требуемое для обработки входных данных.

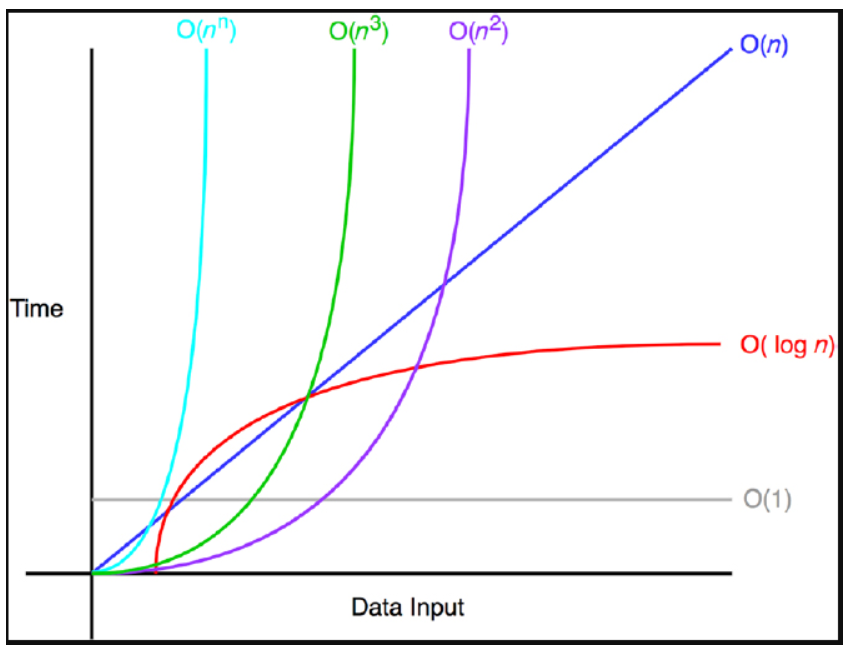
Когда говорят о Time Complexity или просто Time, то речь идёт именно о количестве операций.

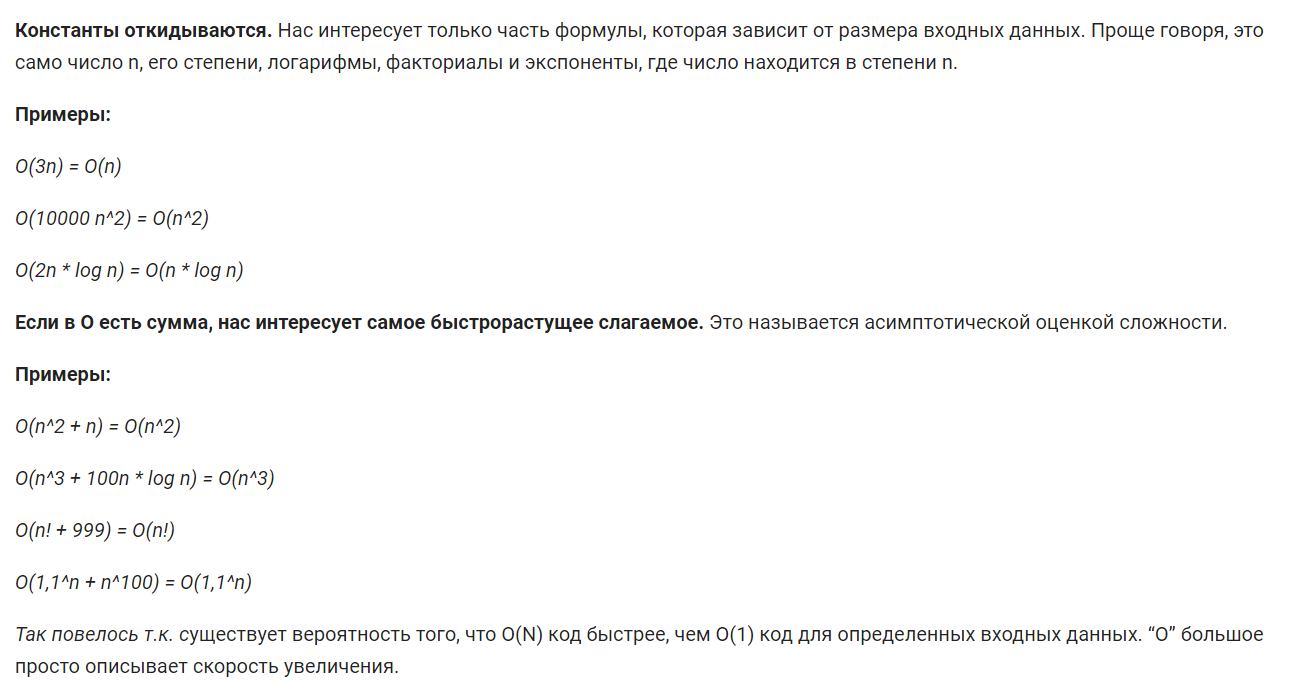
Сложность алгоритма - это мера количества ресурсов (времени или памяти), требуемых для выполнения алгоритма в зависимости от размера входных данных.

Оценка времени исполнения алгоритма обычно выражается в виде асимптотической нотации, такой как "O-нотация".

Этот термин используется в программировании для описания вычислительной сложности алгоритма. В частности, он позволяет оценить, сколько времени требуется для запуска программы.  
Проще говоря, термином “О” большое определяется, *как время выполнения растет по мере увеличения входных данных.*

* **Увеличение времени выполнения.**Поскольку время, необходимое для выполнения программы, зависит от процессора компьютера, используется “О” большое, чтобы показать, как меняется время выполнения.
* **Ввод данных.**Поскольку проверяется не только время, которое требуется для выполнения программы, но и ввод, в нотации “О” большое есть “n”, которое определяет количество элементов обрабатываемых входных данных. Поскольку время выполнения растет с увеличением размера входных данных, эту величину можно представить в виде O(n).
* **По мере увеличения входных данных.**По мере увеличения входных данных программам иногда требуется больше времени для выполнения, что приводит к проблемам с производительностью. Поэтому разрабатываемую программу нужно проверять по мере увеличения входных данных.





1. Общая классификация вычислительных алгоритмов.

Вычислительные алгоритмы могут быть классифицированы по различным критериям. Одна из основных классификаций основывается на типе задач, которые алгоритм решает. Вот общая классификация вычислительных алгоритмов:

* Сортировка и поиск: Алгоритмы, предназначенные для упорядочивания данных (сортировки) или поиска элемента в наборе данных (поиск). Примеры таких алгоритмов включают сортировку пузырьком, сортировку слиянием, быструю сортировку, двоичный поиск и т.д.
* Графы и сети: Алгоритмы, связанные с работой с графами и сетями. Это могут быть алгоритмы обхода графа (например, поиск в глубину или поиск в ширину), алгоритмы кратчайшего пути (например, алгоритм Дейкстры или алгоритм Флойда-Уоршелла), алгоритмы поиска потокового максимума и т.д.
* Геометрические алгоритмы: Алгоритмы, используемые для решения геометрических задач, таких как нахождение пересечений, построение выпуклой оболочки, вычисление площади и т.д. Примеры включают алгоритм Грэхема для построения выпуклой оболочки, алгоритм Бентли-Оттмана для пересечений отрезков и т.д.
* Рекурсивные алгоритмы: Алгоритмы, основанные на принципе рекурсии, когда задача разбивается на подзадачи того же типа. Примеры таких алгоритмов включают быструю сортировку, бинарный поиск, факториал и т.д.
* Алгоритмы машинного обучения: Алгоритмы, используемые для построения моделей и извлечения информации из данных. Это могут быть алгоритмы классификации (например, метод опорных векторов или случайный лес), алгоритмы кластеризации (например, алгоритм k-средних или DBSCAN), алгоритмы регрессии и т.д.

Конкретные алгоритмы в каждой из этих категорий могут быть реализованы на языке C++. С++ предоставляет богатый набор инструментов и библиотек для разработки и реализации различных вычислительных алгоритмов.

Разделяй и властвуй - это метод решения задач, основанный на разбиении большой задачи на более мелкие подзадачи, которые решаются отдельно, а затем объединяются в единое решение. Этот подход часто используется для ускорения вычислений и оптимизации производительности алгоритмов.

Жадные алгоритмы - это метод решения задач, основанный на жадной стратегии выбора наилучшего решения на каждом шаге. Этот подход часто используется для оптимизации решения задач, когда необходимо выбрать из множества вариантов наилучшее решение.

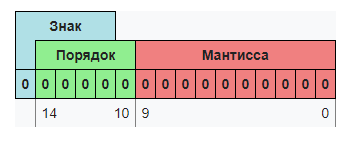
1. Точность представления чисел.





можно считать аналогом экспоненциальной записи чисел, но только в памяти компьютера.

Число с плавающей запятой состоит из набора отдельных двоичных разрядов, условно разделенных на так называемые **знак** (англ. *sign*), **порядок** (англ. *exponent*) и **мантиссу** (англ. *mantis*). В наиболее распространённом формате (стандарт IEEE 754) число с плавающей запятой представляется в виде набора битов, часть из которых кодирует собой мантиссу числа, другая часть — показатель степени, и ещё один бит используется для указания знака числа (0 — если число положительное, 1 — если число отрицательное).При этом порядок записывается как целое число в [коде со сдвигом](http://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB:_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B9_%D0%BA%D0%BE%D0%B4,_%D0%BA%D0%BE%D0%B4_%D1%81%D0%BE_%D1%81%D0%B4%D0%B2%D0%B8%D0%B3%D0%BE%D0%BC,_%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), а мантисса — в [нормализованном виде](http://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB&mobileaction=toggle_view_desktop#.D0.9D.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D1.8C.D0.BD.D0.B0.D1.8F_.D0.B8_.D0.BD.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D0.B8.D0.B7.D0.BE.D0.B2.D0.B0.D0.BD.D0.BD.D0.B0.D1.8F_.D1.84.D0.BE.D1.80.D0.BC.D0.B0), своей дробной частью в двоичной системе счисления. Вот пример такого числа из 16 двоичных разрядов:

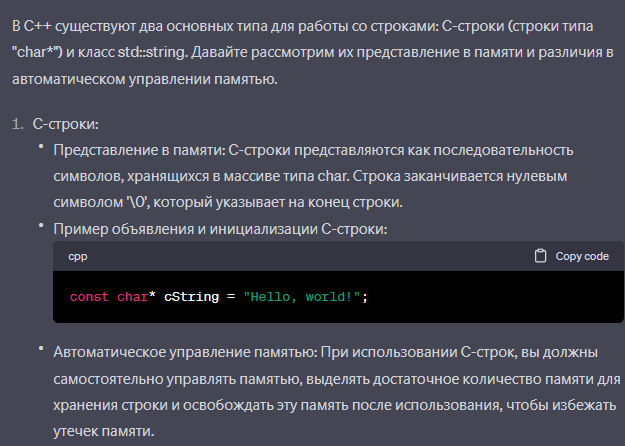


Знак — один бит, указывающий знак всего числа с плавающей точкой. Порядок и мантисса — целые числа, которые вместе со знаком дают представление числа с плавающей запятой в следующем виде: 

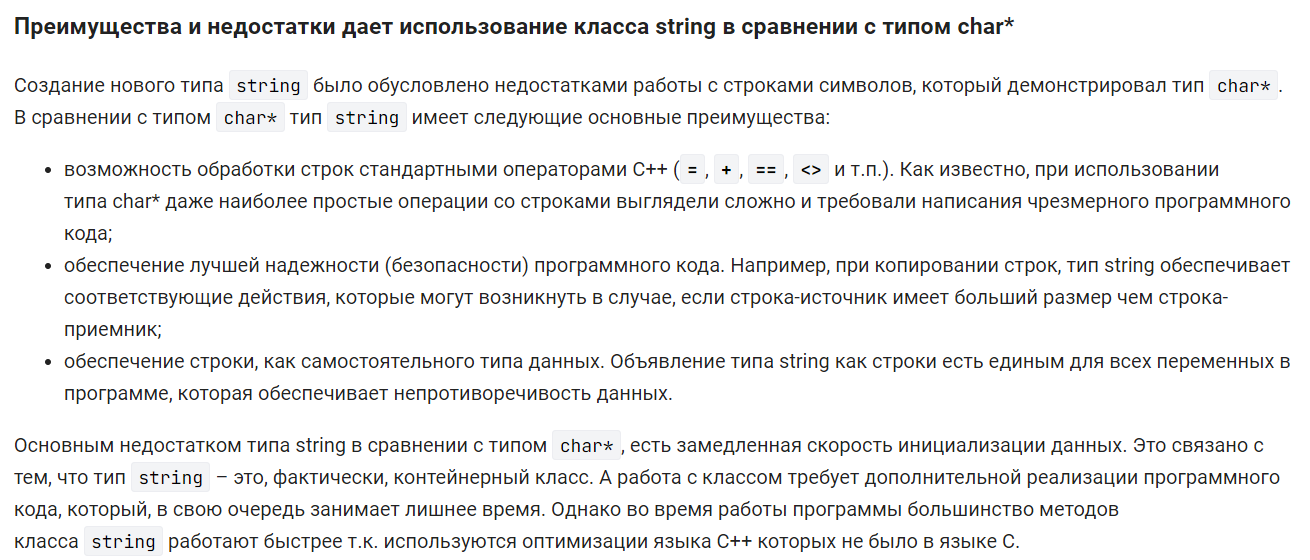
Порядок также иногда называют **экспонентой** или просто **показателем степени**.

При этом лишь некоторые из вещественных чисел могут быть представлены в памяти компьютера точным значением, в то время как остальные числа представляются приближенными значениями.

1. Строковый тип в C++. Представление в памяти. Автоматическое управление памятью для Си строк и std::string.



управление памятью для C-строк должно выполняться вручную с помощью функций malloc, calloc, realloc и free.





1. Файлы. Двоичное и текстовое представление файлов. Стандартные файлы.

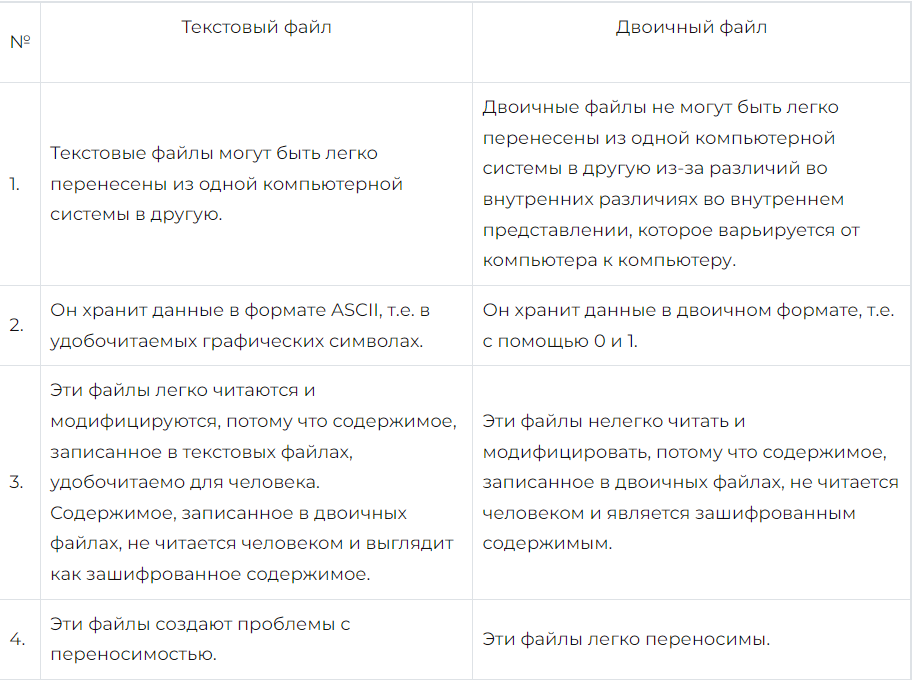
Файл – именованный объект, хранящий данные на каком-либо носителе (дискета, винчестер, СD). В программировании файлы обычно используются для чтения и записи данных.  
Файл не имеет фиксированной длины, т.е. может увеличиваться и уменьшаться.  
Перед работой с файлом его необходимо открыть, а после работы – закрыть.  
Принято различать:

• бинарные файлы;

• текстовые файлы.

Текстовый файл — это файл, в котором данные хранятся в виде символов ASCII, и обычно используется для хранения потока символов. Текстовые файлы организованы вокруг строк, каждая из которых заканчивается символом новой строки (‘\n’).  
Двоичный файл отличается от текстового тем, что данные в нем представлены во внутренней форме. А поскольку при внутреннем представлении используется двоичная система счисления, то файлы называются двоичными. По существу, двоичный файл является аналогом внутренней (оперативной, физической) памяти – неограниченным массивом байтов с возможностью непосредственного обращения (произвольного доступа) к любой его части.

Текстовый – чтение(ifstream), запись(ofstream)   
Бинарный – чтение(read), запись(write)

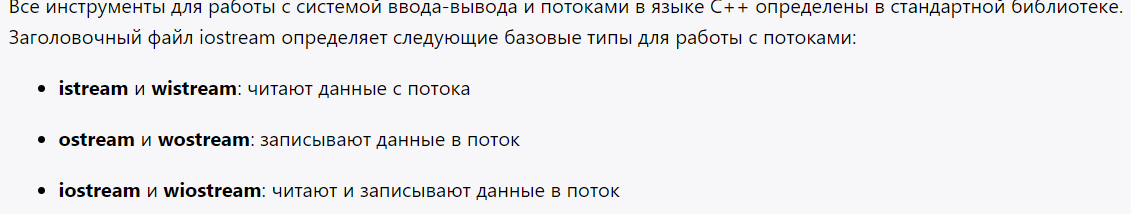
\

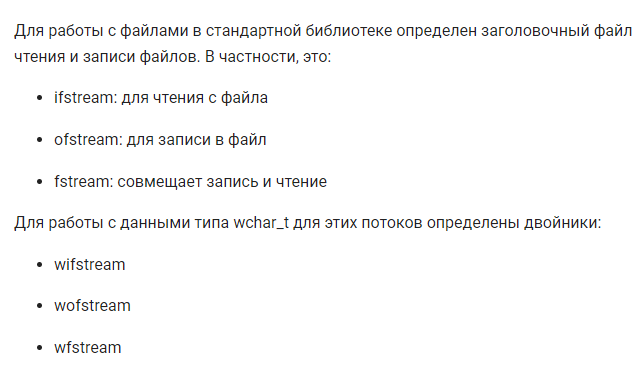


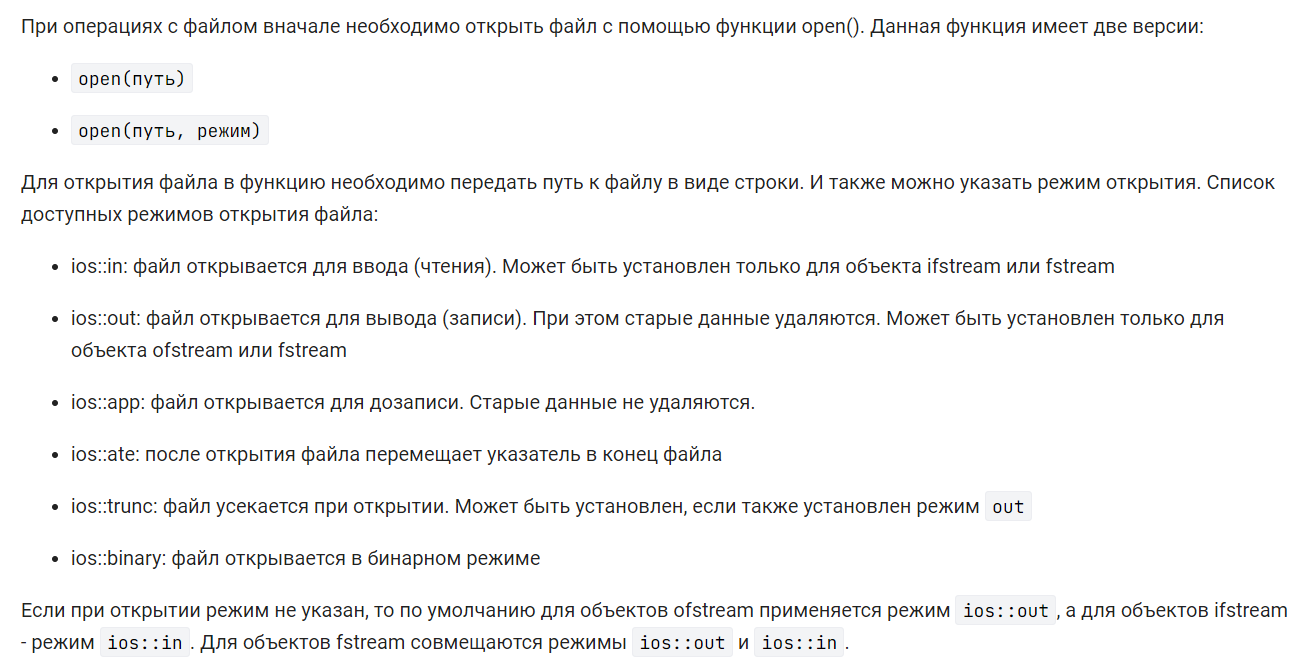


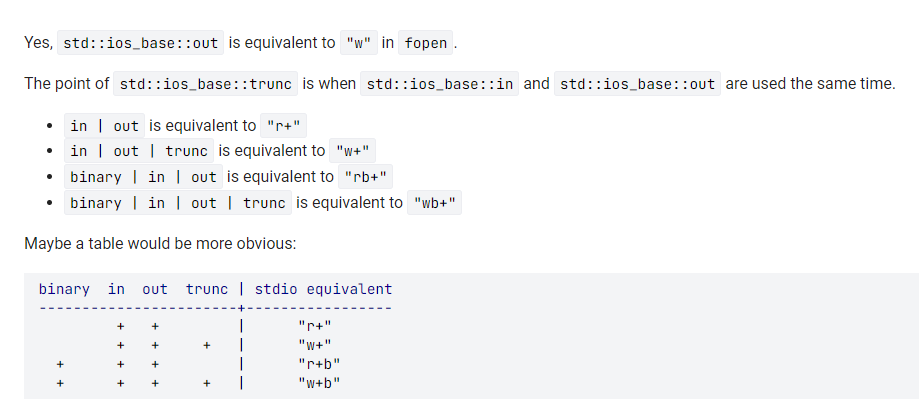
1. Понятие потока. Открытие и закрытие файлов. Операции ввода-вывода. Указатель чтения-записи в файле.

Поток (stream) - это абстракция ввода-вывода данных в C++. Он представляет собой последовательный поток данных, который может быть прочитан или записан. В C++ существуют потоки для различных типов ввода-вывода, включая файловые потоки.

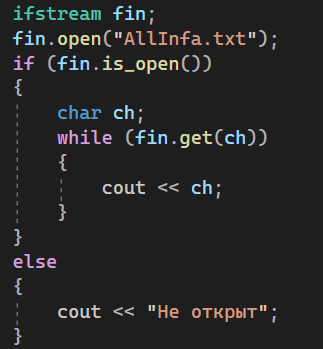




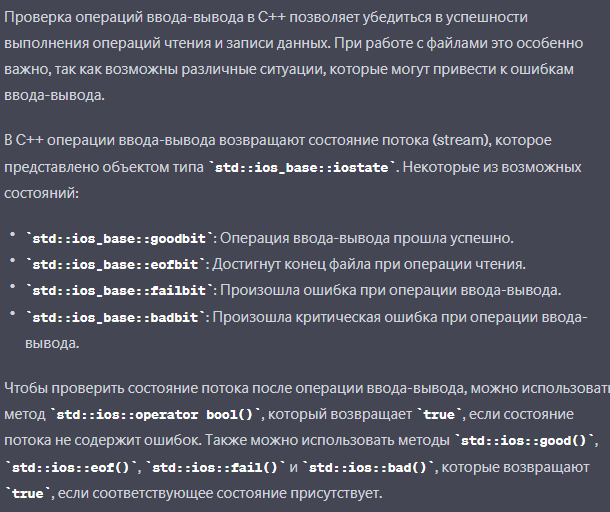
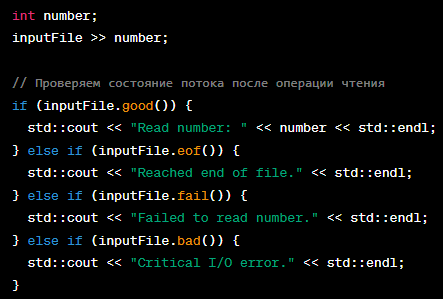


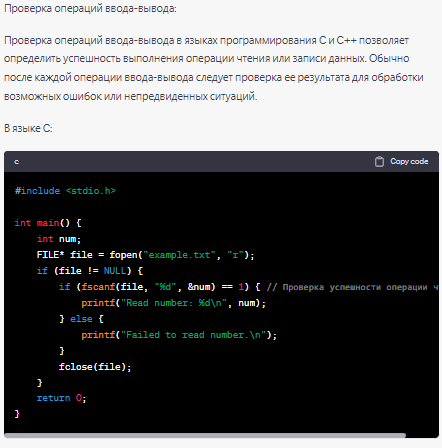


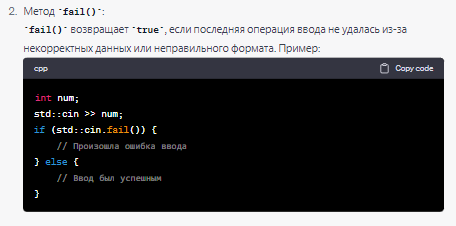
out.close(); // закрываем файл  
После открытия файла вы можете выполнять операции ввода и вывода данных. Для чтения данных из файла используется оператор >>, а для записи данных в файл - оператор <<. Вывод:



Указатель чтения-записи в файле представляет текущую позицию в файле, на которую указывает операция чтения или записи. В C++ указатель чтения-записи управляется автоматически.  
При открытии файла указатель чтения-записи устанавливается в начало файла. После операции чтения или записи указатель автоматически продвигается вперед, указывая на следующие данные. Если файл открыт в режиме добавления (std::ofstream::app), то указатель устанавливается в конец файла перед каждой операцией записи.  
Если вы хотите управлять указателем чтения-записи в файле вручную, можно использовать функции seekg() для установки указателя чтения и seekp() для установки указателя записи.  
  


1. Проверка операций ввода-вывода. Пример.  
     
     
   

****

****

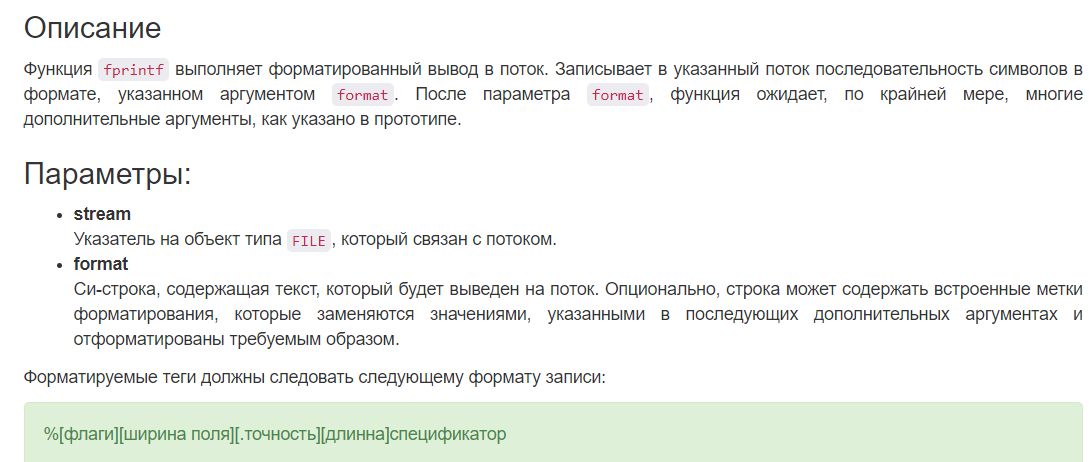
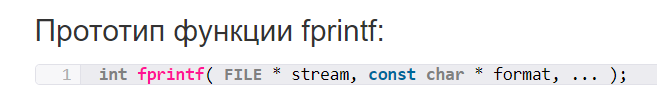
1. Функции для модификации содержимого файла: futc, getc, feof, fputs, fgets, fprintf, fscanf, rewind.

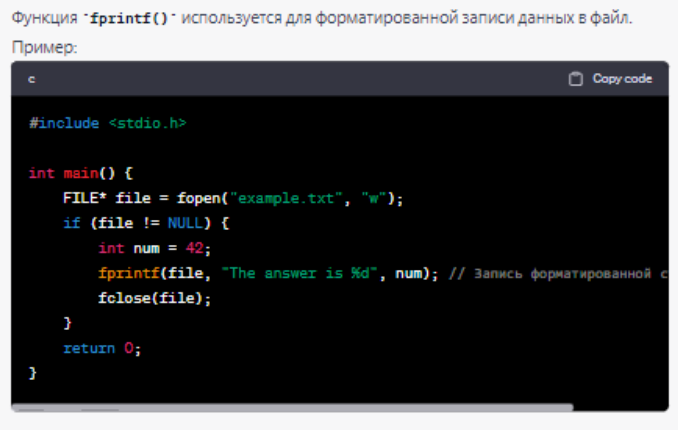
Функция getc() является частью стандартной библиотеки в C и C++. Она используется для считывания одного символа из файла.  
Функция getc() принимает один параметр - указатель на файл, из которого будет осуществляться считывание символа.

feof() - проверяет, достигнут ли конец файла.

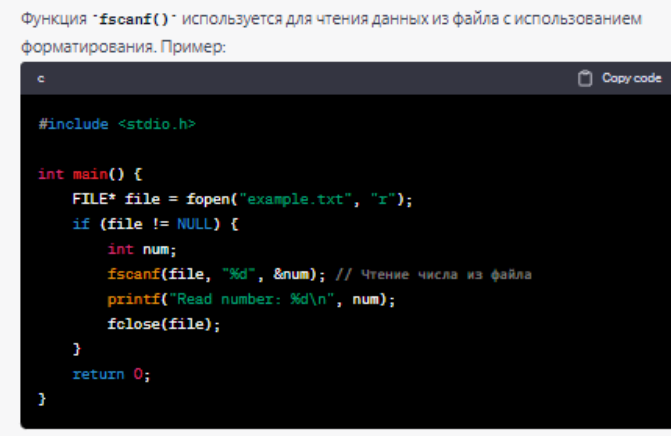
Функция fputs() является частью стандартной библиотеки в C и C++. Она используется для записи строки в файл.  
Функция fputs() принимает два параметра: строку, которую нужно записать, и указатель на файл, в который производится запись. Она возвращает неотрицательное значение в случае успеха и значение EOF (End-of-File) в случае ошибки.

Функция fgets() является частью стандартной библиотеки в C и C++. Она используется для считывания строки из файла.  
Функция fgets() принимает три параметра: указатель на массив символов, в который будет сохранена считанная строка, максимальное количество символов для чтения (включая символ завершения строки) и указатель на файл, из которого будет осуществляться чтение. Она возвращает указатель на массив символов, содержащий считанную строку, или значение NULL в случае достижения конца файла или возникновения ошибки.



Функция fscanf() является частью стандартной библиотеки в C и C++. Она используется для форматированного считывания данных из файла.  
Функция fscanf() принимает три параметра: указатель на файл, из которого будет производиться считывание, форматная строка, определяющая ожидаемую структуру данных, и указатели на переменные, в которые будут сохранены считанные значения. Она возвращает количество успешно считанных значений или значение EOF (End-of-File) в случае достижения конца файла или возникновения ошибки.

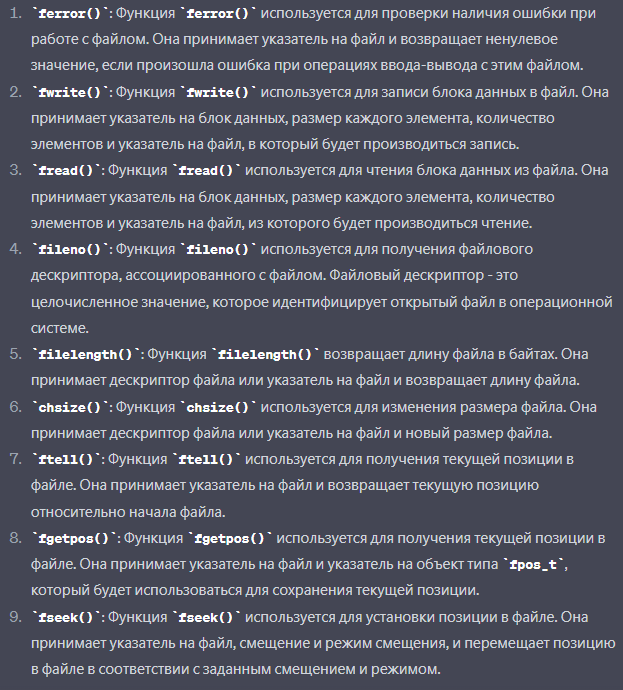


rewind() - устанавливает указатель чтения/записи в начало файла.

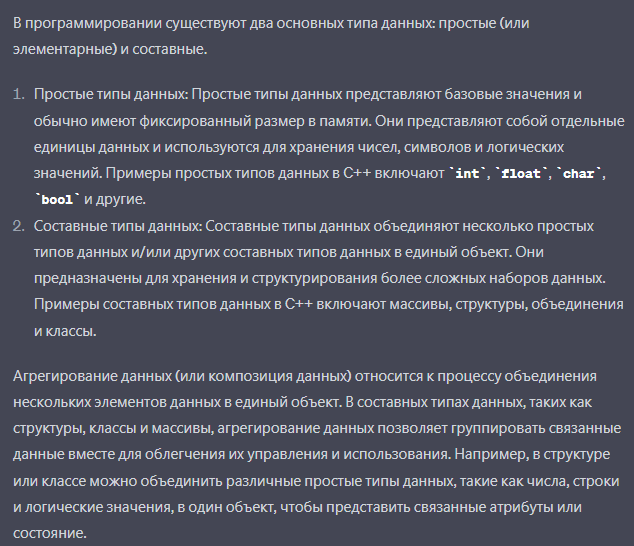
1. Функции для модификации содержимого файла: ferror, fwrite, fread, fileno, filelength, chsize, ftell, fgetpos, fseek.

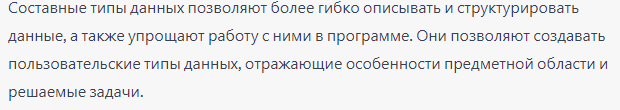


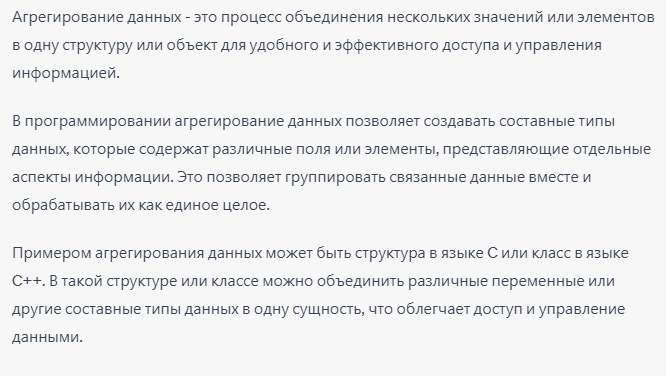
Так с FILE\* писать и в 8 и в 9 вопросах



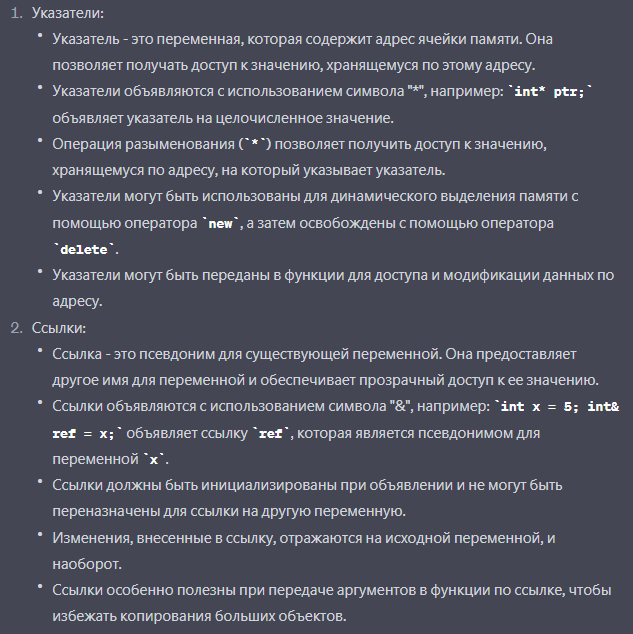
1. Типы данных – простые и составные. Агрегирование данных.

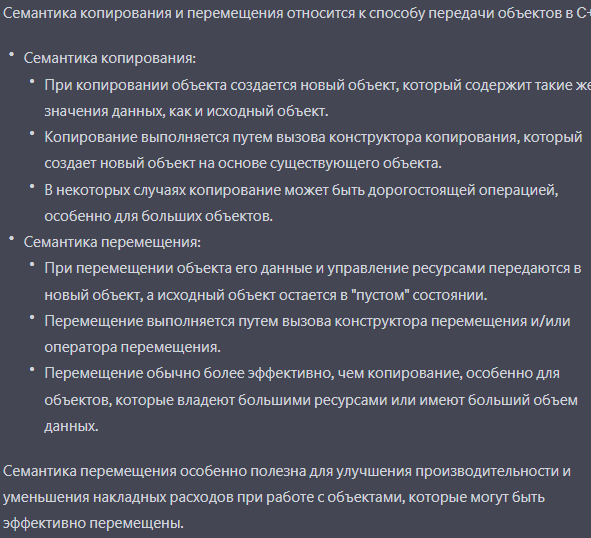


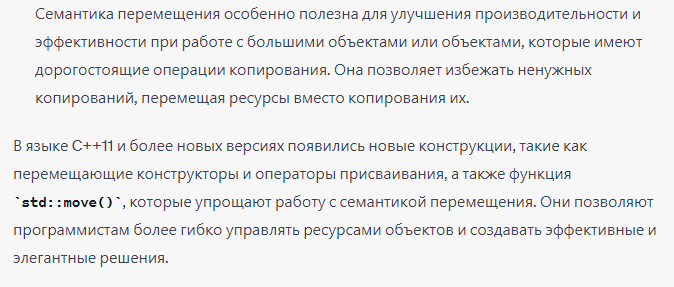
****

****

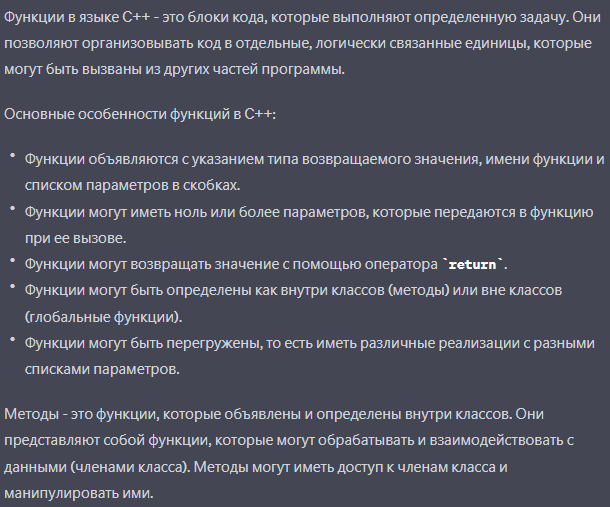
1. Указатели и ссылки в языке С++. Семантика копирования и перемещения.





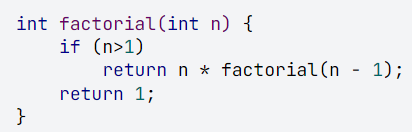
****

1. Функции в языке С++. Понятие метода. Рекурсия.

  
**constexpr** — указывает, что возвращаемое значение функции является константой, значение которой может быть определено во время компиляции.

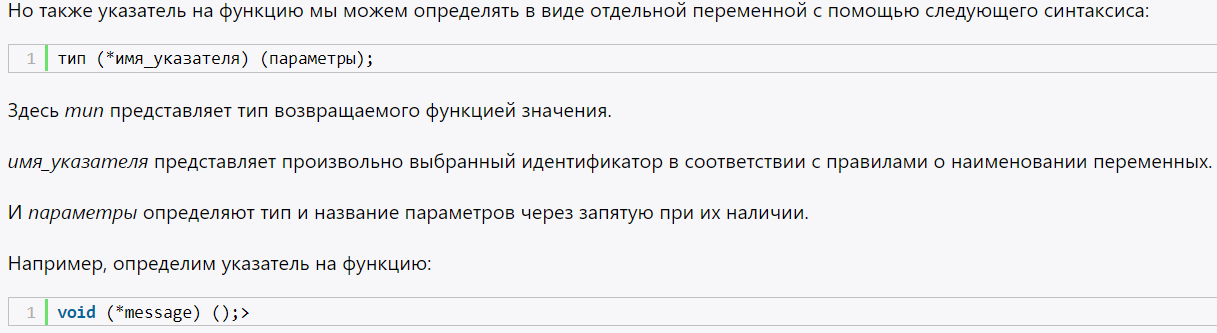
**inline** — отдает компилятору команду заменять каждый вызов функции ее кодом. Подстановка может улучшить эффективность кода в сценариях, где функция выполняется быстро и многократно вызывается во фрагментах, являющихся критическими для производительности программы.

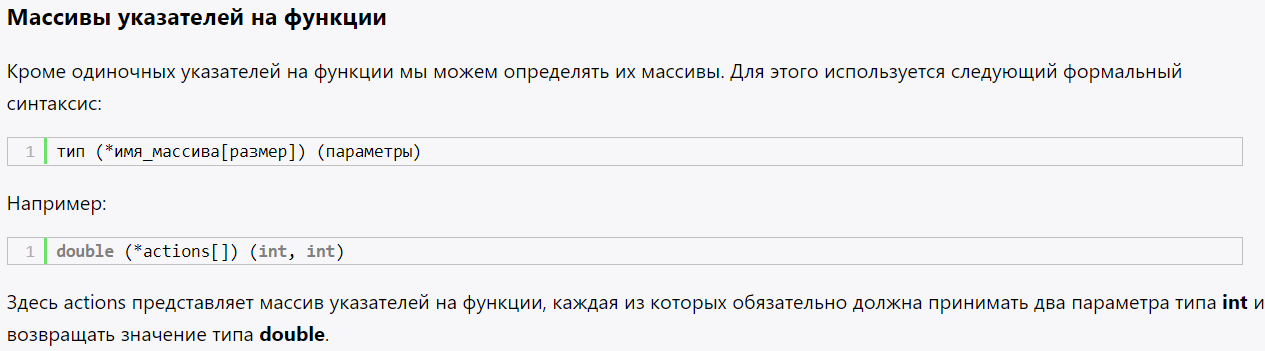
Выражение **noexcept** , указывающее, может ли функция вызывать исключение. В следующем примере функция не создает исключение, если is\_pod выражение принимает значение **true**.

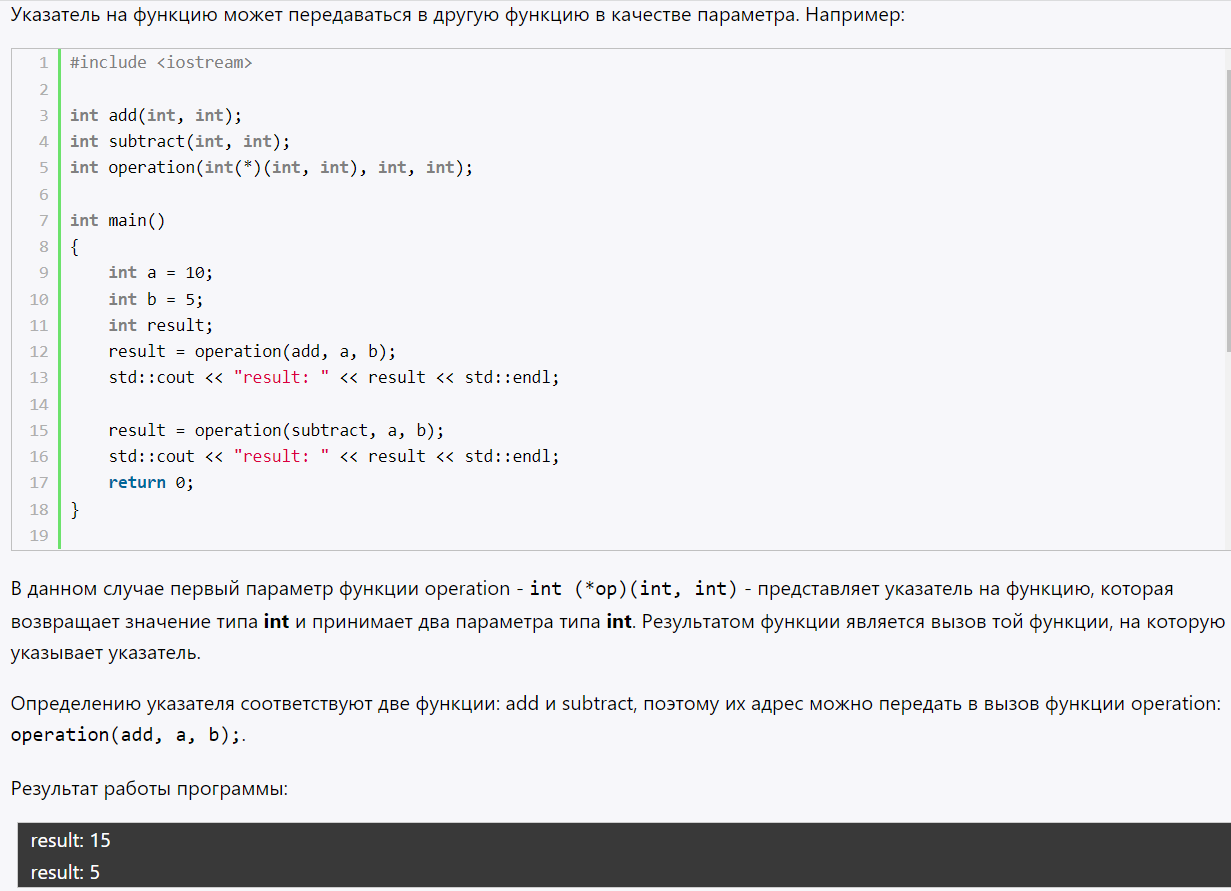
Рекурсия - это процесс, при котором функция вызывает саму себя в своем теле. При каждом рекурсивном вызове запоминаются предыдущие значения внутренних локальных переменных и полученных параметров функции. Чтобы следующий шаг рекурсивного вызова отличался от предыдущего, значение как-минимум одного из параметров функции должно быть изменено.  


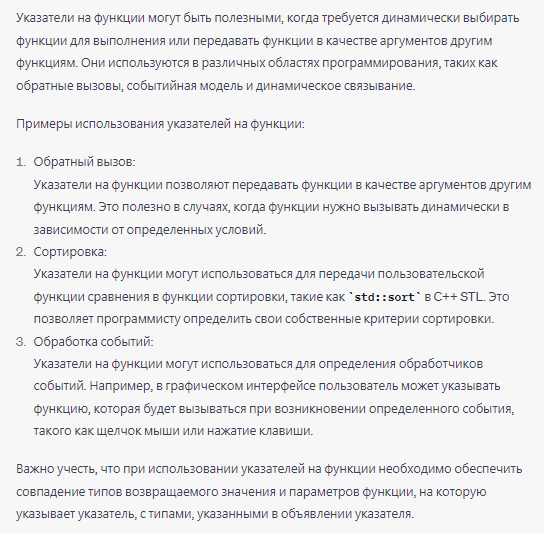
1. Указатели на функции в языке С++.

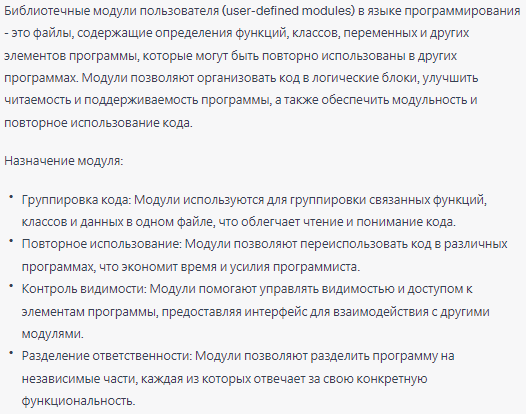
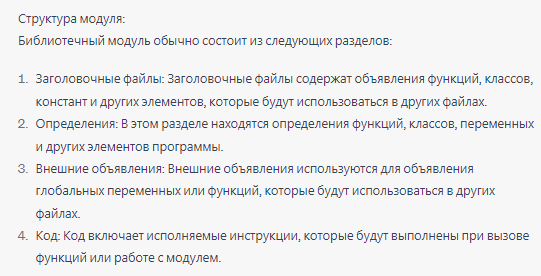
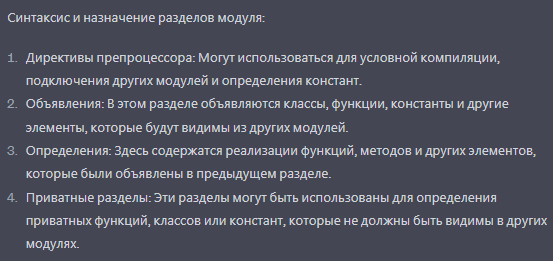
Указатель на функцию (function pointer) хранит адрес функции. По сути указатель на функцию содержит адрес первого байта в памяти, по которому располагается выполняемый код функции.

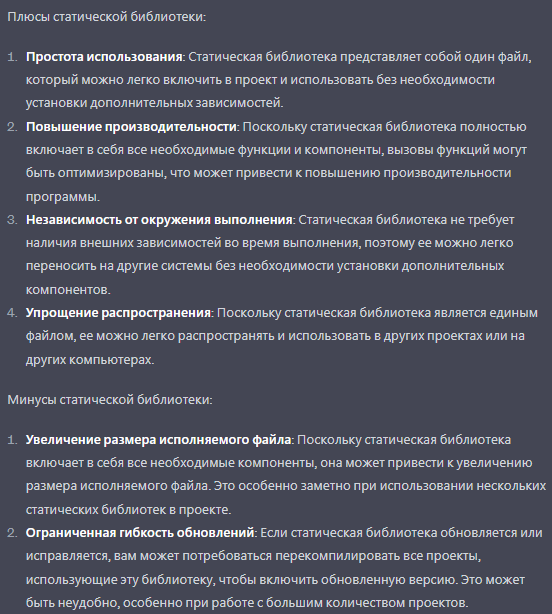
****

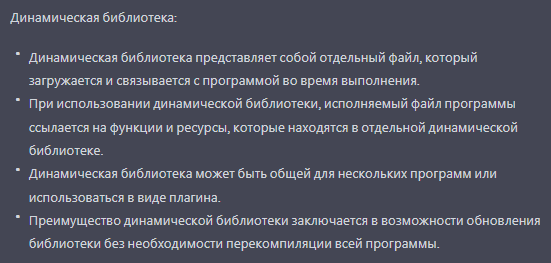
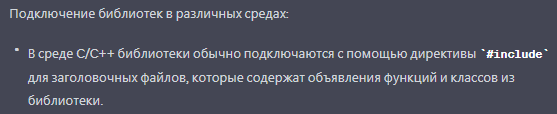
1. Библиотечные модули пользователя. Назначение модуля. Структура модуля. Синтаксис и назначение разделов модуля. Пример.  
   ****  
   ****  
   
2. Программирование статистических и динамических библиотек. Подключение в различных средах.

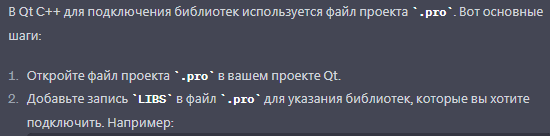
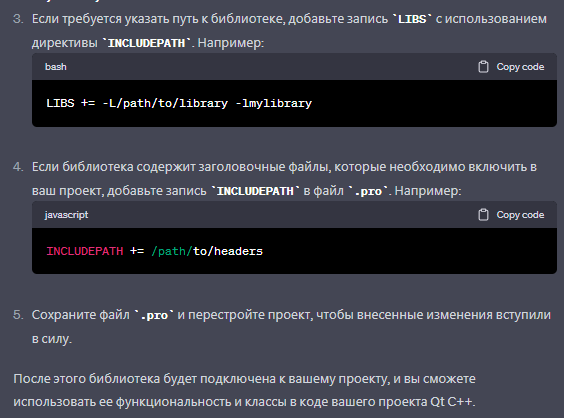
Программирование статических и динамических библиотек является важной частью разработки программного обеспечения.   
Плюсы разбивать программу на несколько частей:

* обычно сложная программа разбивается на несколько отдельных частей, которые отлаживаются отдельно и зачастую разными способами.
* при исправлении в одном модуле не надо снова транслировать все остальные.
* при компоновке во многих системах можно подключать модули, написанные на других ЯП.

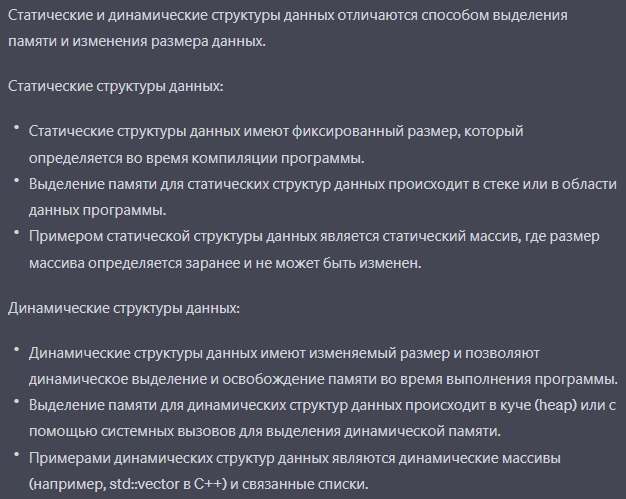
Библиотека объектных файлов – несколько объектных файлов, которые используются для хранения функций и ресурсов отдельно от исполняемого файла. Позволяет ускорить процесс компоновки программы, так как поиск функций и переменных в объектных файлах библиотеки происходит намного быстрее, чем в исполняемом файле.

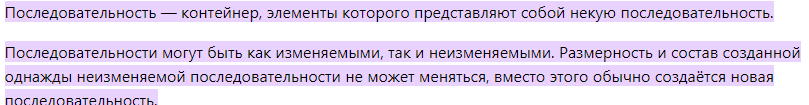


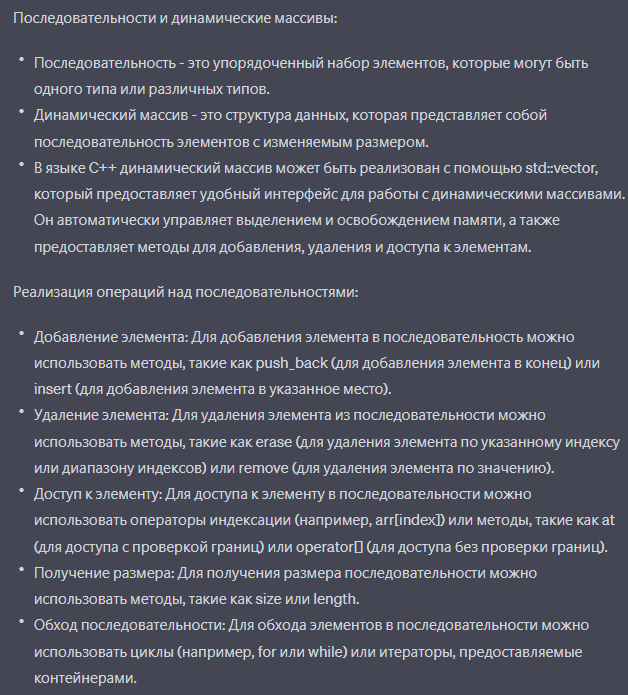
  
НО УВЕЛИЧИВАЕТ ПОТРЕБЛЯЕМУЮ ПАМЯТЬ И УВЕЛИЧИВАЕТ ВРЕМЯ ВЫПОЛНЕНИЯ.  
  


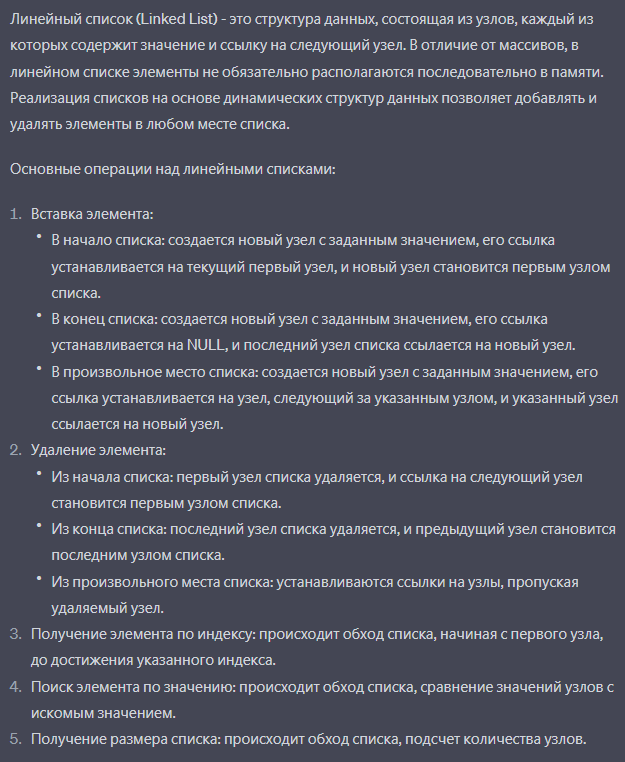
1. Статические и динамические структуры данных. Последовательности и динамические массивы. Реализация операций над последовательностями.

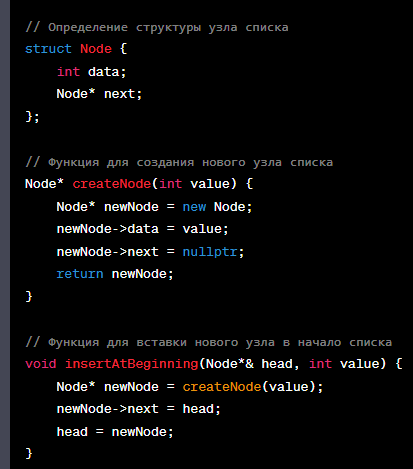




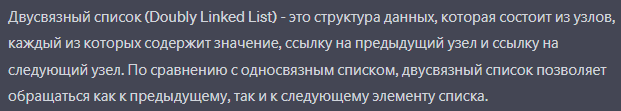


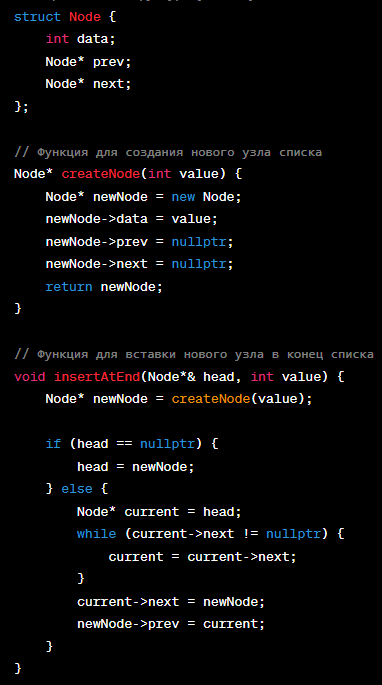
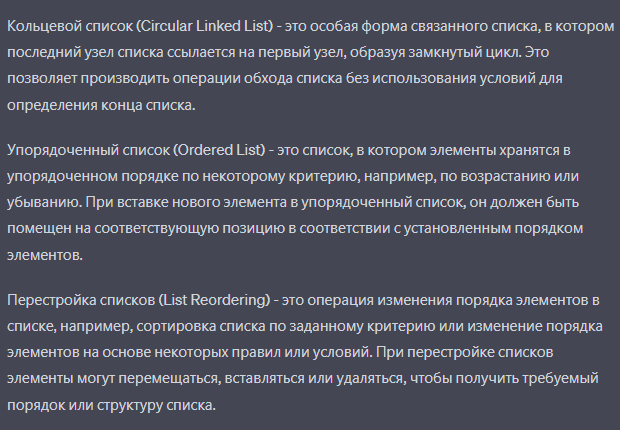
1. Линейные списки – основные операции. Реализация списков на основе динамических структур.



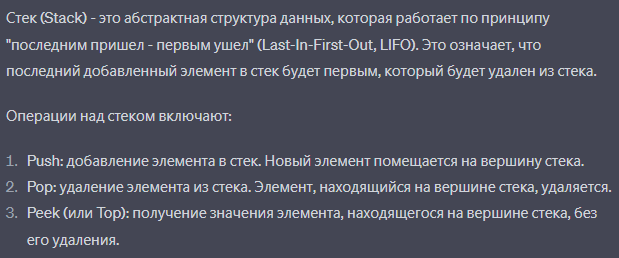


1. Двусвязный список и его программная реализация. Кольцевые списки. Упорядоченные списки и перестройка списков.

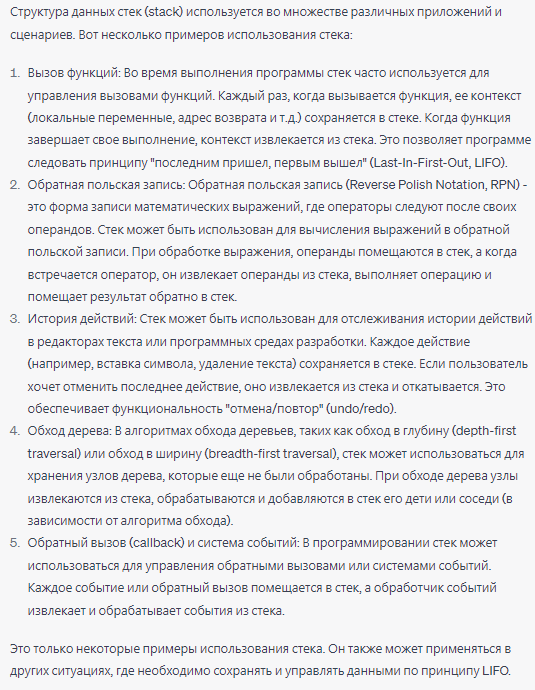


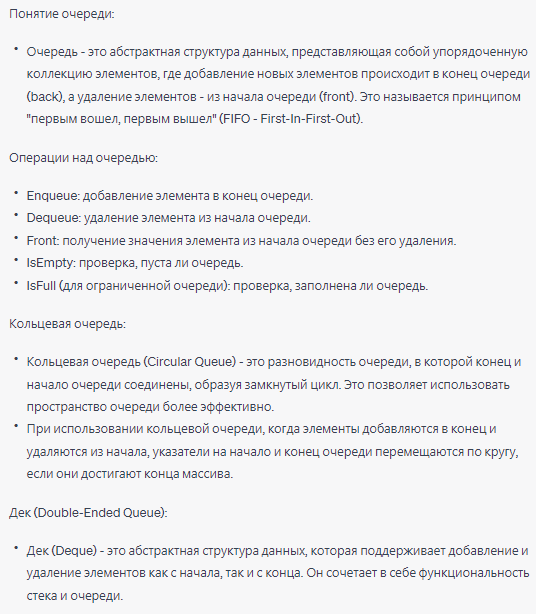
1. Понятие стека. Операции над стеком. Программная реализация стека на основе статического массива.



Пример реализации в папке

****

1. Понятие очереди. Операции над очередями. Кольцевая очередь. Деки. Программная реализация очереди на основе статического массива.

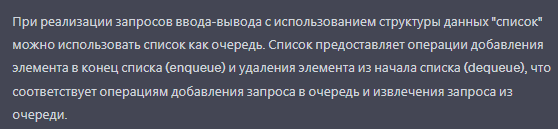
****Дек (Deque) - это структура данных, которая позволяет добавлять и удалять элементы как в начало, так и в конец контейнера. Является двусторонней очередью, которая позволяет добавлять и удалять элементы с обоих концов очереди. Поддерживает операции удаления, добавления и доступа к элементам с обоих сторон.

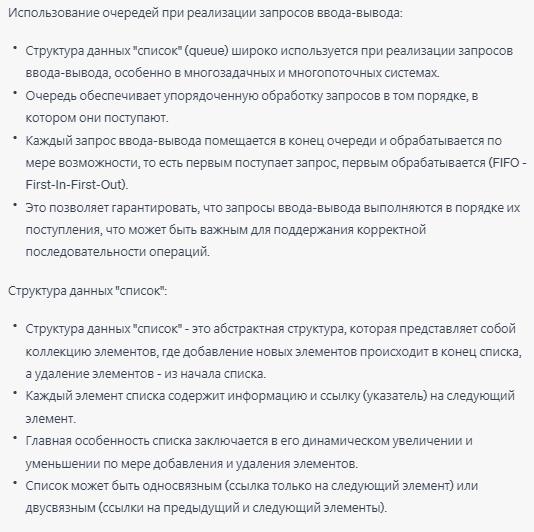
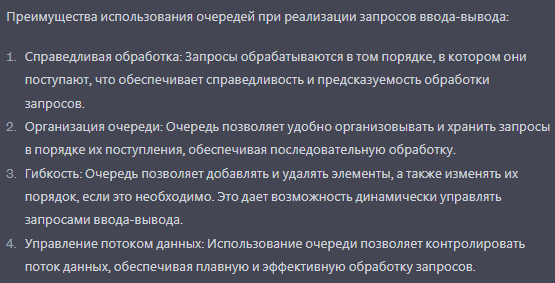
Реализация очереди есть в папке

1. Использование очередей при реализации запросов ввода-вывода Структура данных «список».

Реализация очереди есть в папке

Реализация списка есть в папке

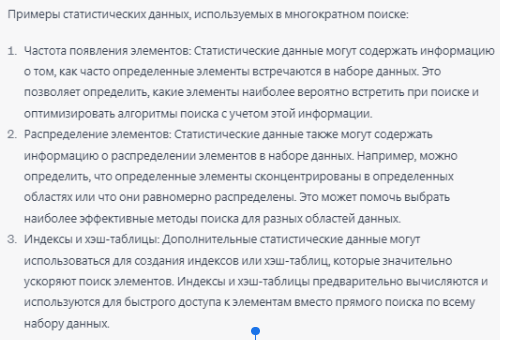


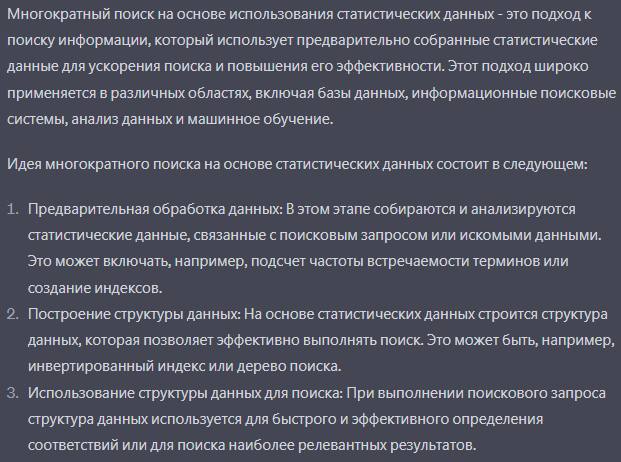
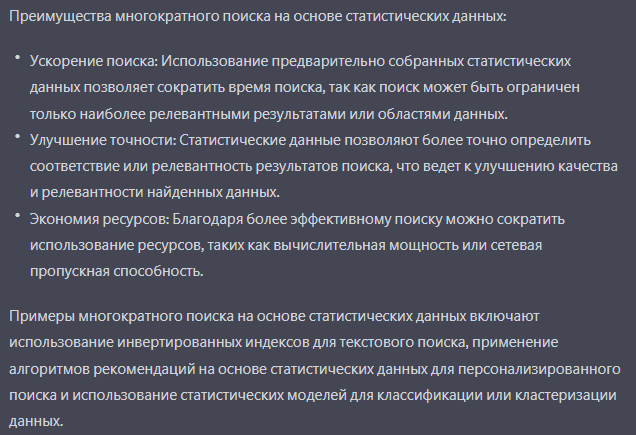
  


1. Программная реализация очереди на основе статического массива.  
     
   РЕАЛИЗАЦИЯ В ПАПКЕ

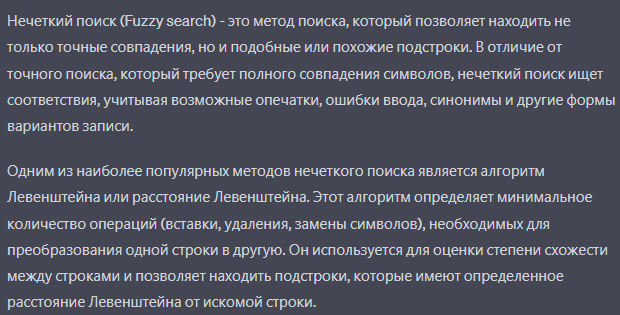
1. Многократный поиск на основе использования статистических данных.

Статистические данные - это числовые или категориальные данные, полученные путем сбора информации и их последующего анализа. Они используются для описания и изучения различных явлений, процессов или популяций.



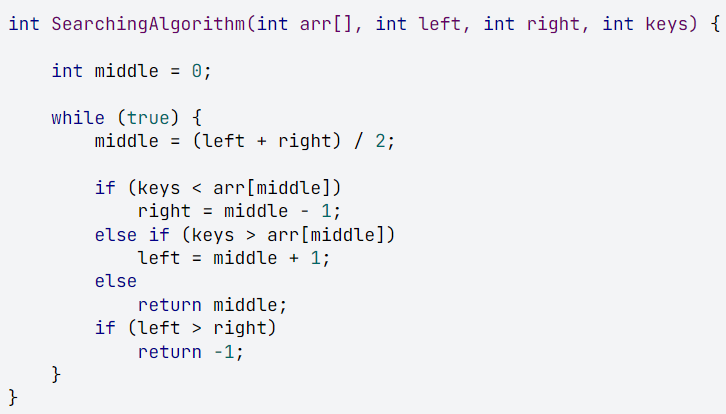
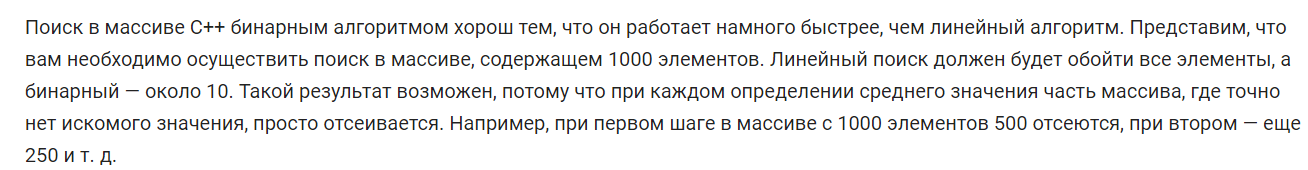
  


1. Нечеткий поиск – поиск «подобной» подстроки. Бинарный поиск.

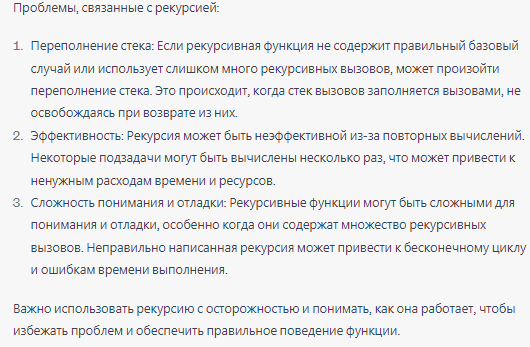


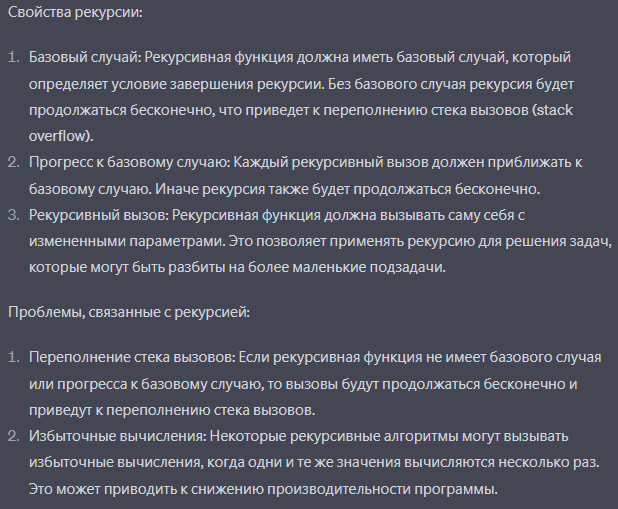
Алгоритмы нечеткого поиска (также известного как поиск по сходству или fuzzy string search) являются основой систем проверки орфографии и полноценных поисковых систем вроде Google или Yandex. Например, такие алгоритмы используются для функций наподобие «Возможно вы имели в виду …» в тех же поисковых системах.

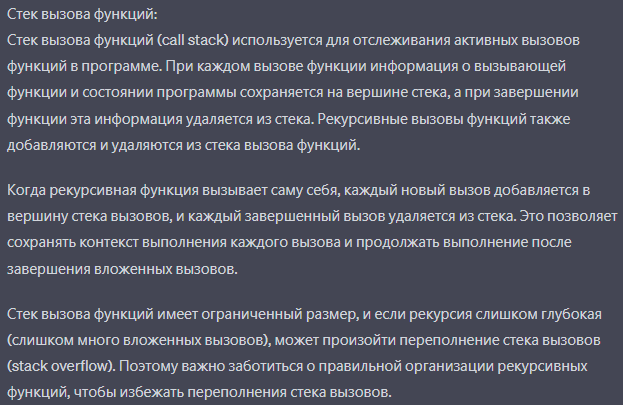
Бинарный поиск в массиве на С++ подходит для **отсортированных массивов**.

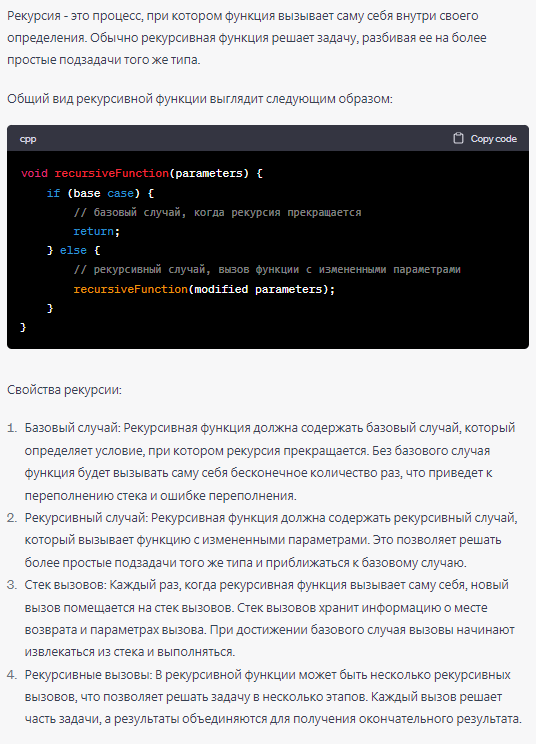
  


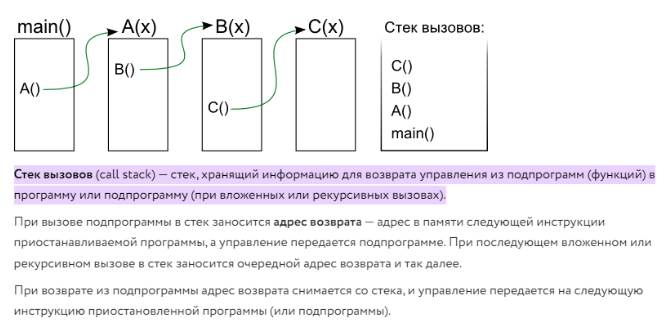
1. Рекурсия: общий вид, свойства, проблемы. Стек вызова функций.

Рекурсия - это процесс, при котором функция вызывает саму себя в своем теле. Функция, которая вызывает саму себя, называется рекурсивной функцией, а процесс повторных вызовов называется рекурсией.  
****

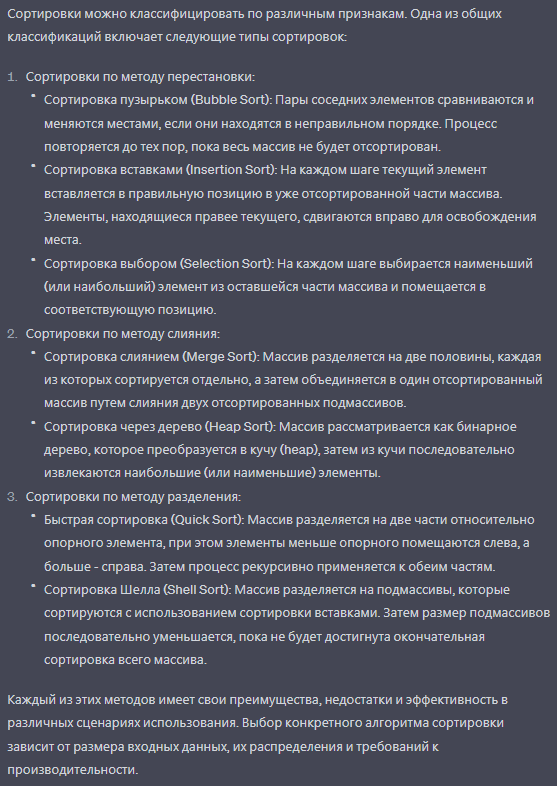


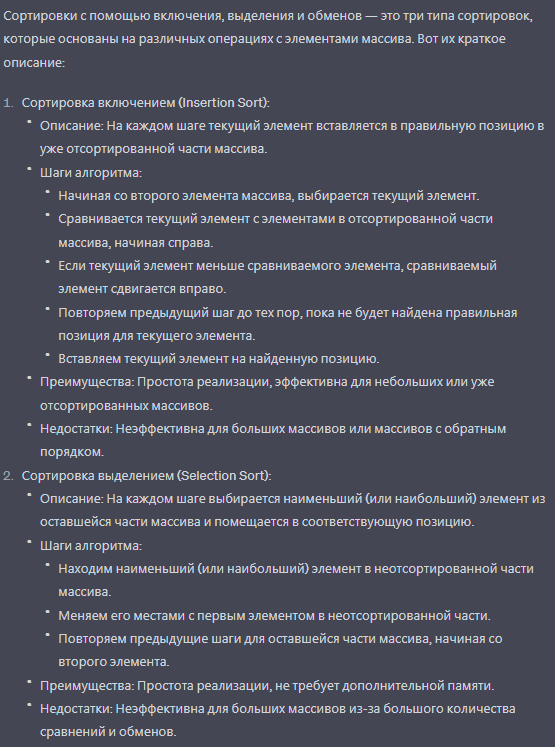


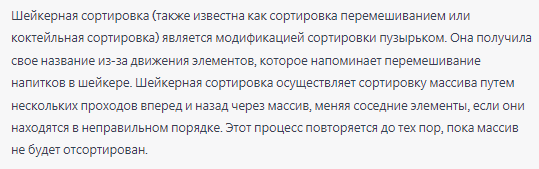
****

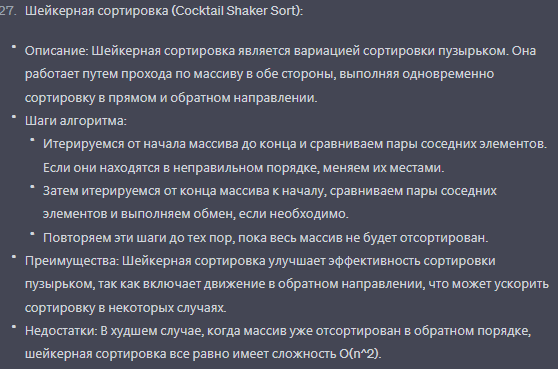


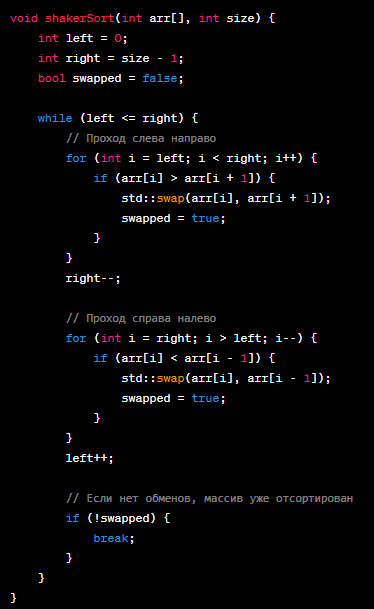
1. Сортировки – общая классификация. Сортировки с помощью включения, выделения, обменов.

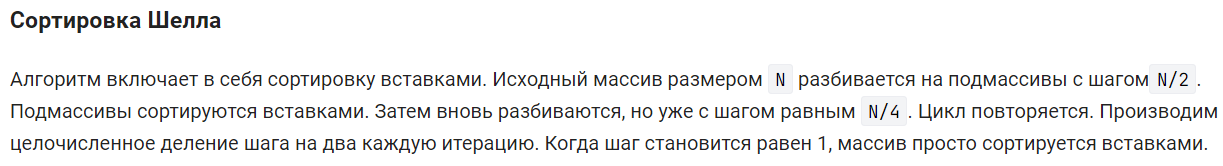


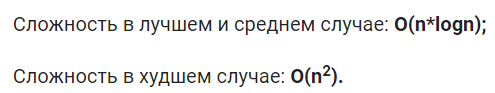
  

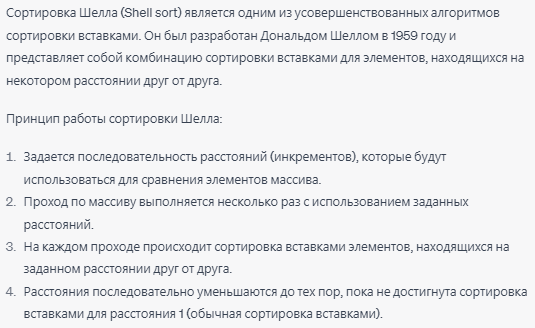

1. Шейкерная сортировка. Сортировка Шелла.  
   ****

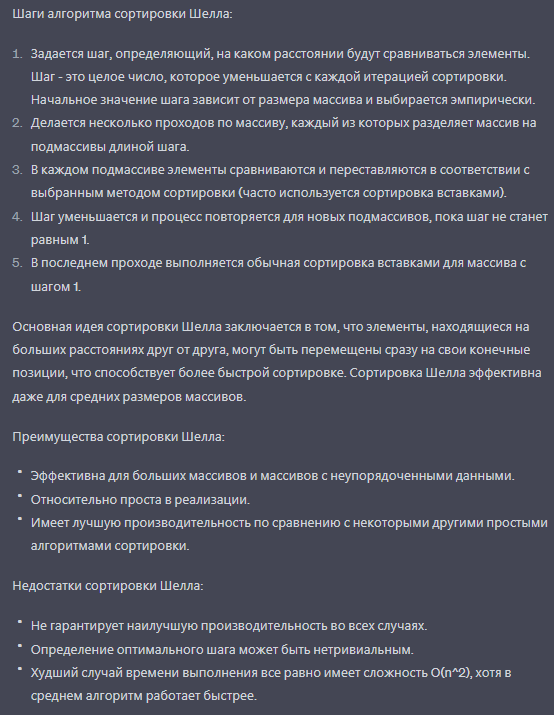


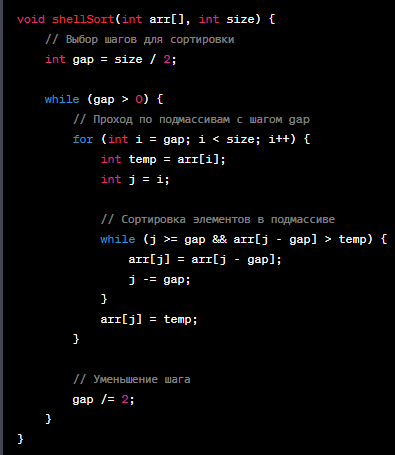




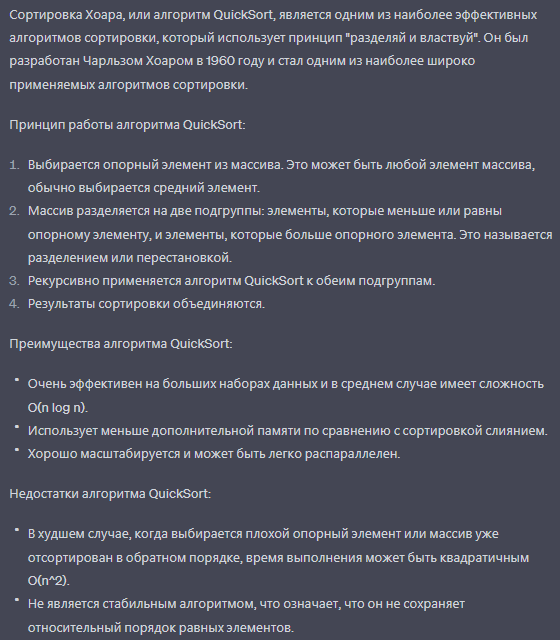


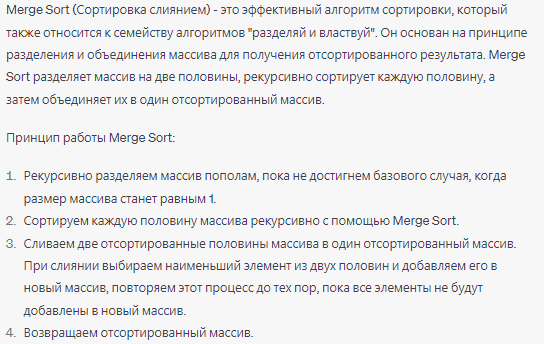
****

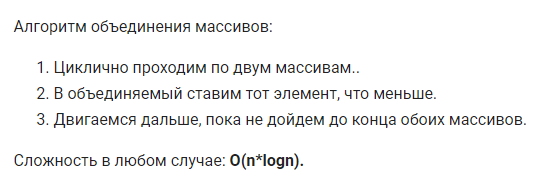
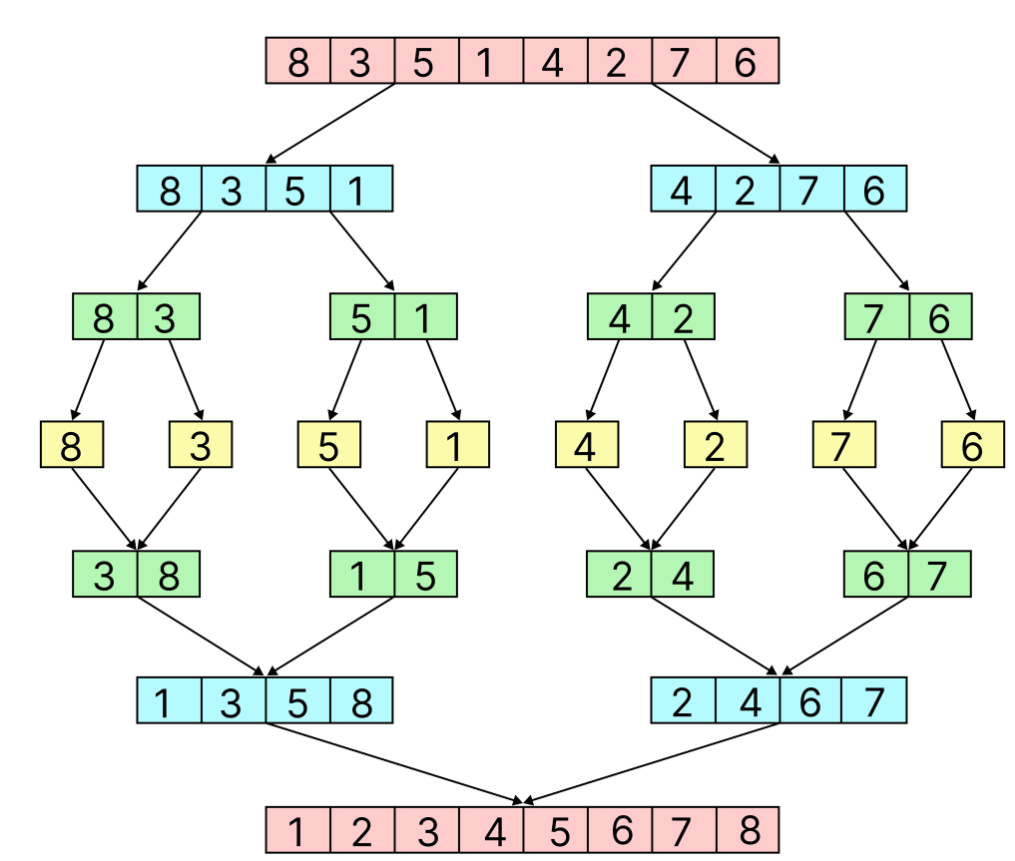




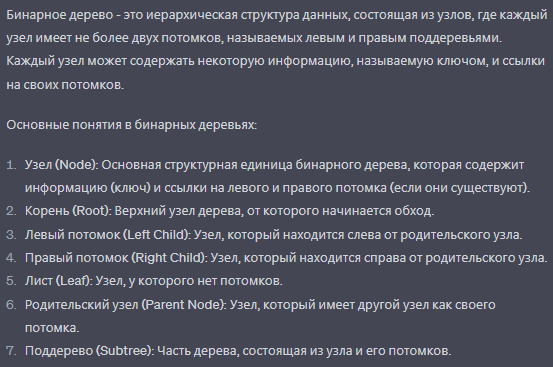
1. Сортировка Хоара – алгоритм QuickSort. Сортировка слиянием.



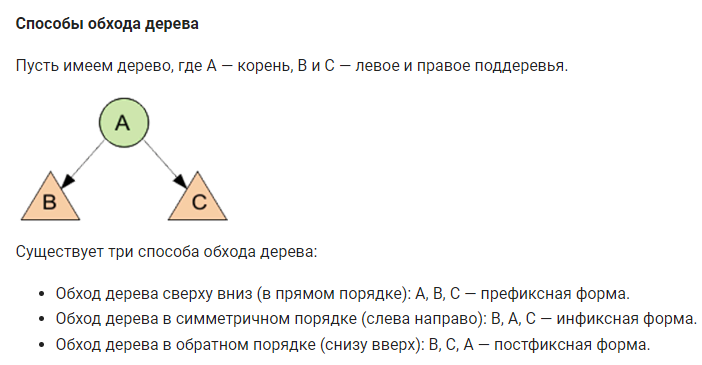
****

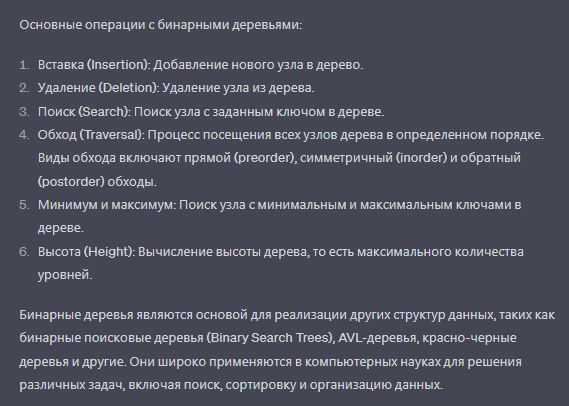
   


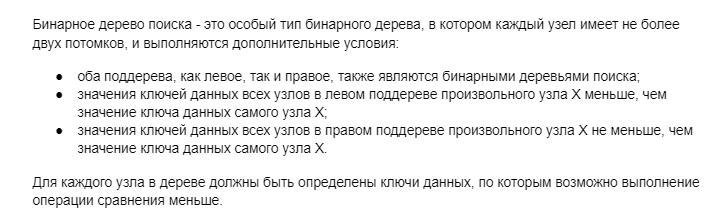
1. Бинарные деревья – основные понятия. Основные операции с бинарными деревьями.



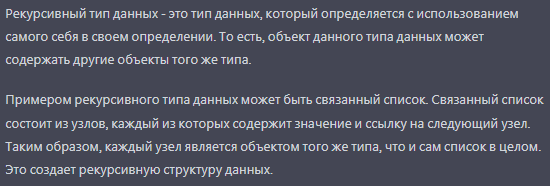


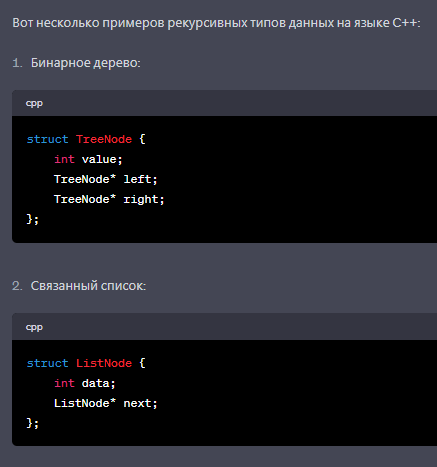




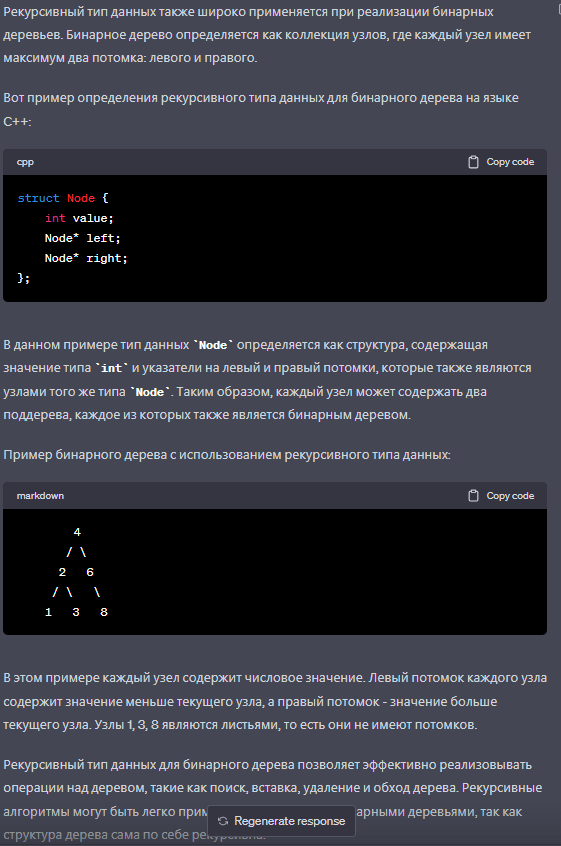


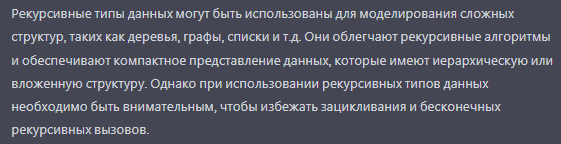
1. Понятие рекурсивного типа данных.



  
Понятие рекурсивного типа данных относится к структурам данных, которые определяются через самих себя. То есть, тип данных используется в определении самого себя. Это позволяет создавать сложные структуры, состоящие из более простых элементов того же типа.

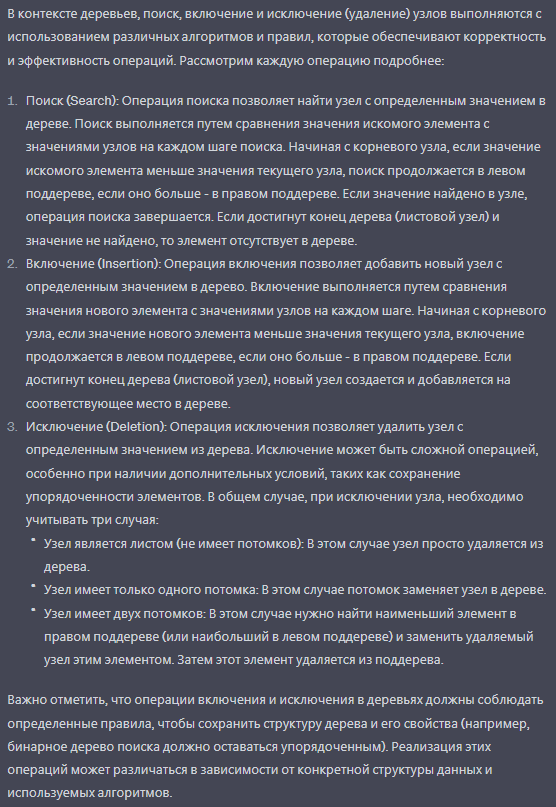
Рекурсивные типы данных могут быть определены в различных областях программирования, таких как функциональное программирование, объектно-ориентированное программирование и другие. Они играют важную роль в построении сложных структур, которые могут быть неограниченно вложены друг в друга.



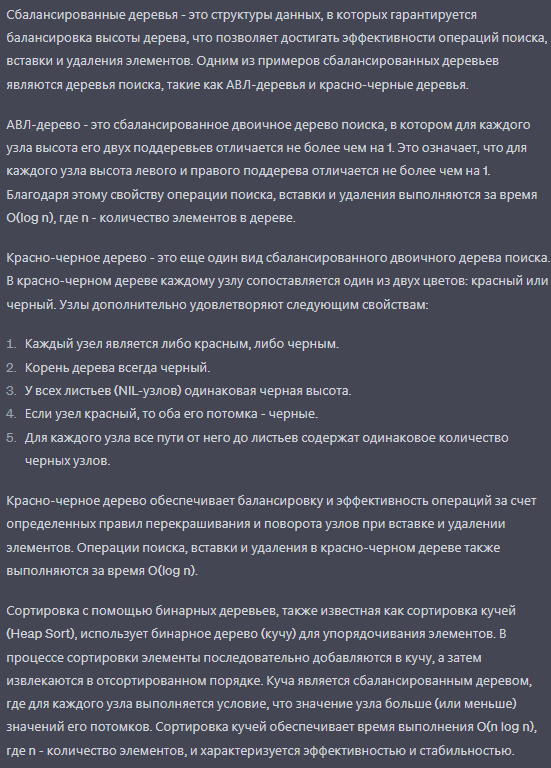


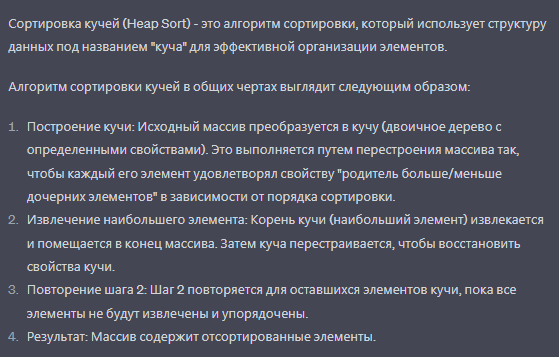
Рекурсивные типы данных предоставляют гибкий и мощный способ моделирования сложных структур. Они широко используются в программировании для решения задач, связанных с обработкой и манипулированием данными, таких как поиск, сортировка, обход и т.д.

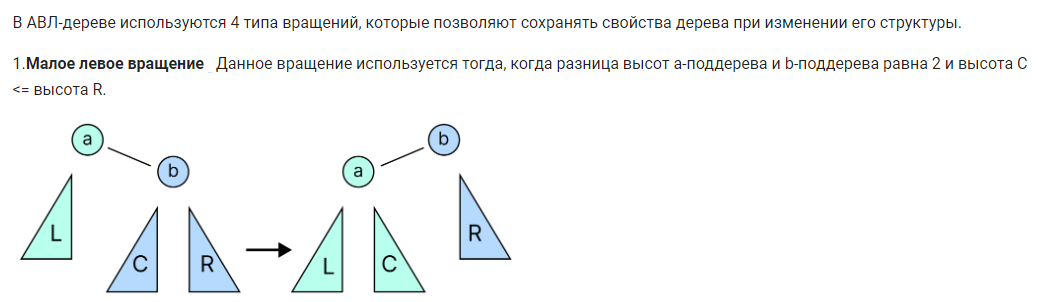
1. Поиск и включение для деревьев. Исключение для деревьев.



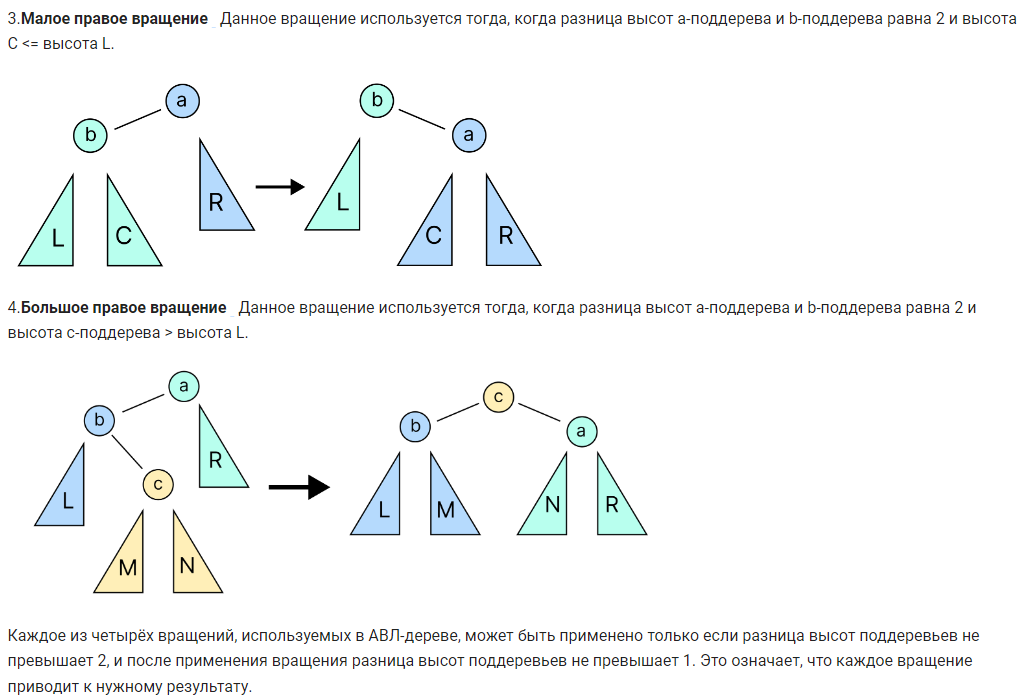
1. Сбалансированные деревья. Сортировка с помощью бинарных деревьев (кучи).



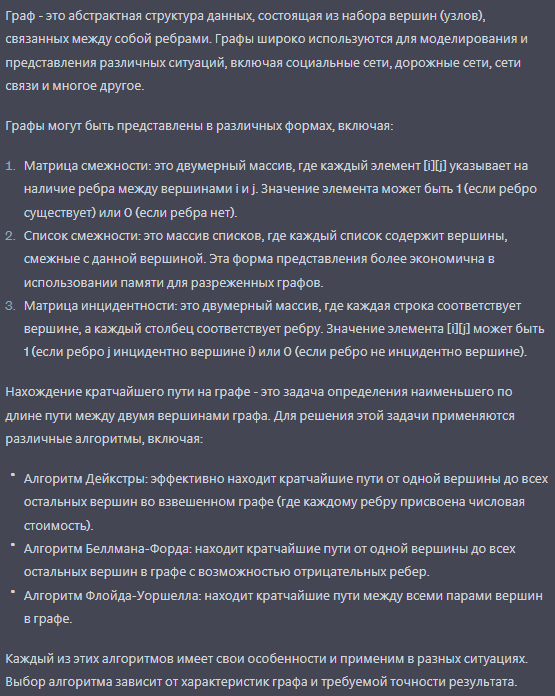
  
Данная сортировка является оптимальной при получении данных путём непосредственного чтения из [потока](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (например, файла, сокета или консоли).

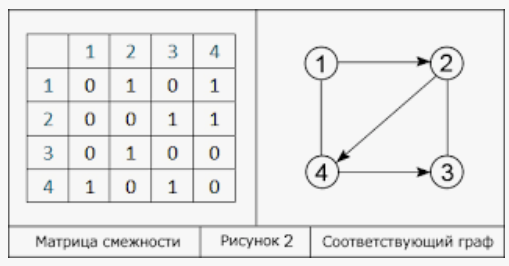


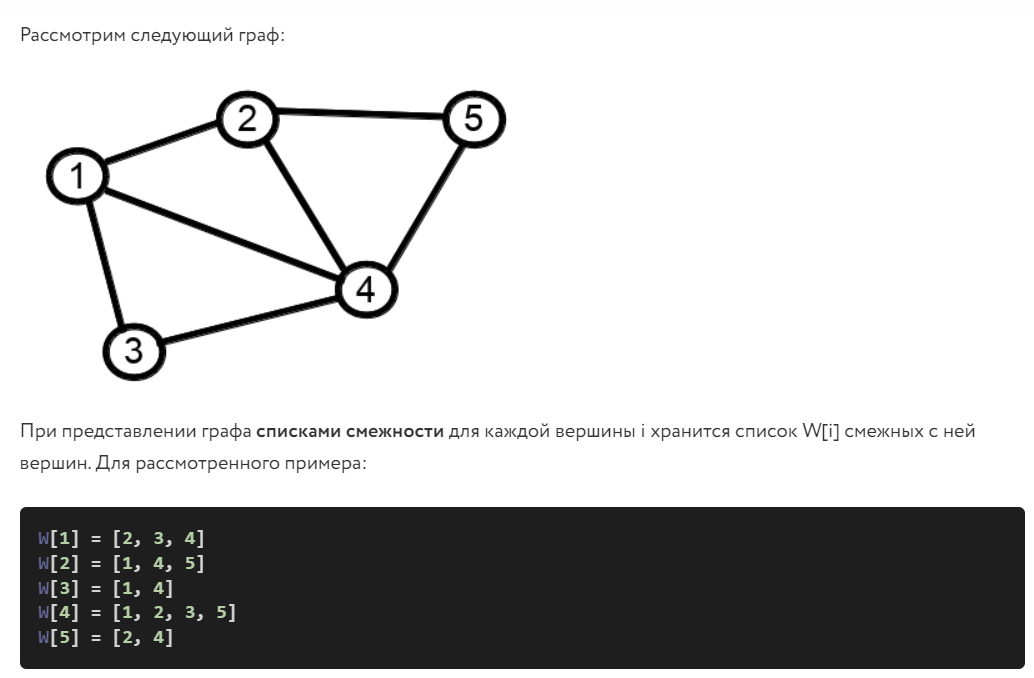


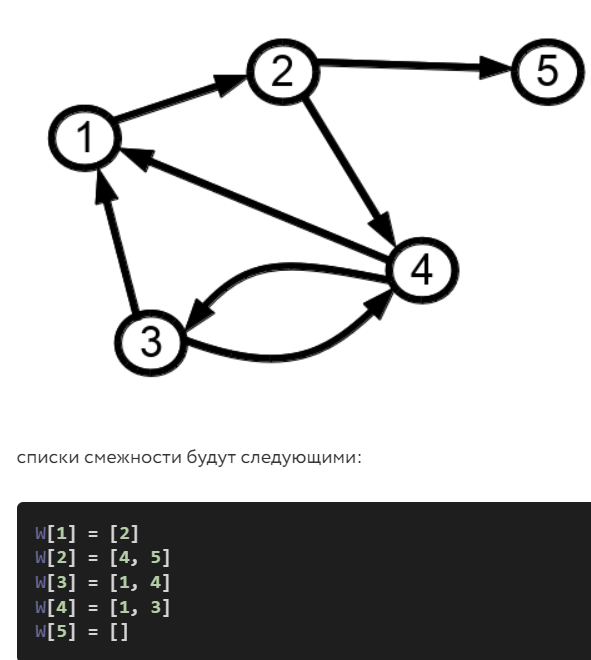


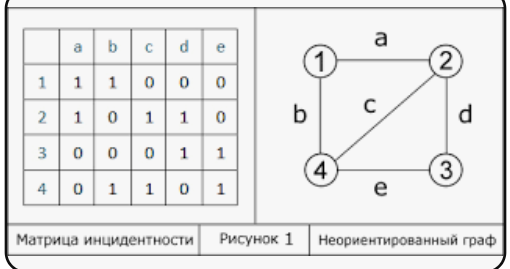
1. Графы и возможные формы их описания. Нахождение кратчайшего пути на графе.



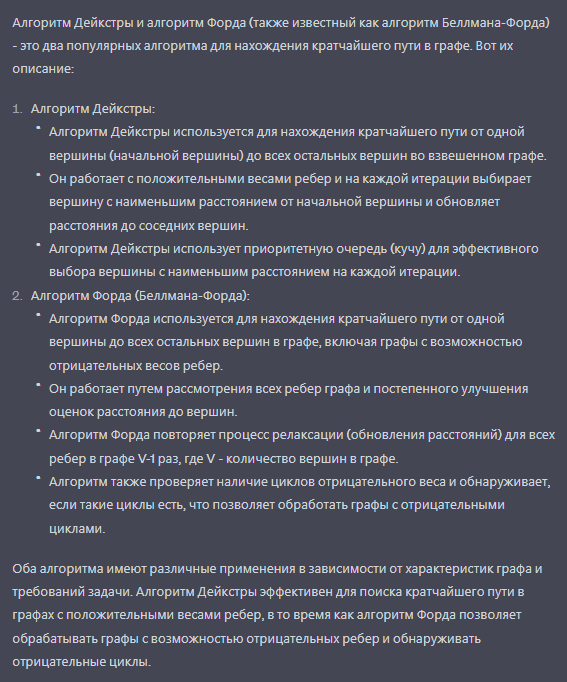








1. Алгоритм Дейкстры, алгоритм Форда.



<https://www.youtube.com/watch?v=54LoGg7dfsc&ab_channel=rozalinaag> – алгоримт дейкстры

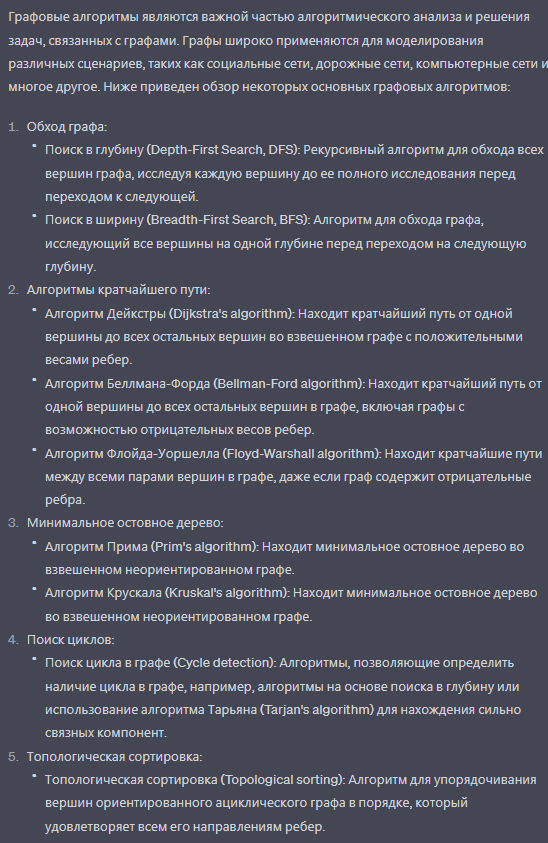
**Алгоритм Беллмана — Форда** — алгоритм поиска кратчайшего [пути](https://ru.wikipedia.org/wiki/%D0%9F%D1%83%D1%82%D1%8C_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) во [взвешенном графе](https://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%B2%D0%B5%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84). За время

***O ( |V| \* |E| )*** (где |V| - количество вершин графа, а |E| - количество ребер)

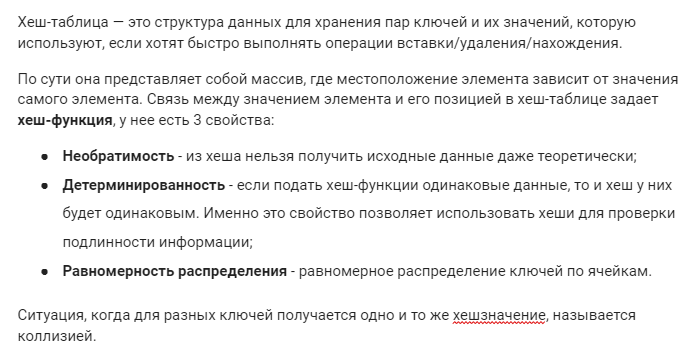
1. Графовые алгоритмы – обзор.

Применяются раскраска графа (При раскраске графов элементам графа присваиваются цвета с учётом определённых условий. Раскраска вершин — наиболее часто используемый метод окраски графов. При этом вершины графа окрашиваются с использованием k цветов, а любым двум соседним вершинам должны соответствовать разные цвета. Другие методы окраски — раскраска рёбер и раскраска граней.

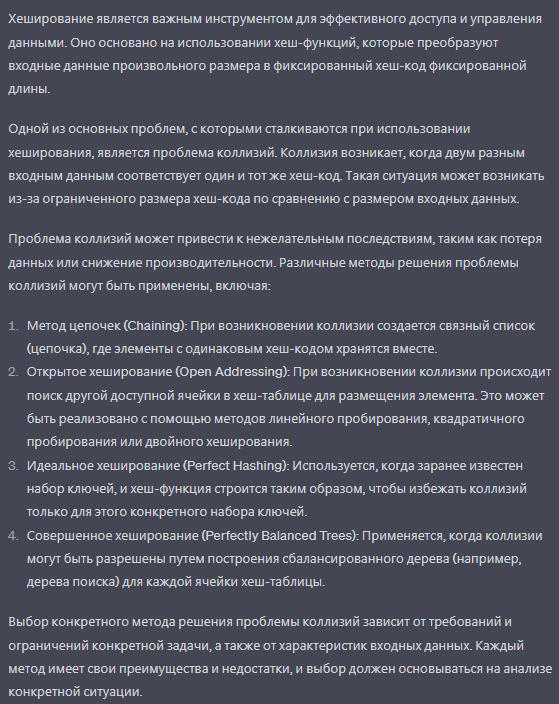
Применяется поиск максимального потока (Можно смоделировать граф в виде сети потоков с весами рёбер в качестве пропускной способности этих потоков. В задаче максимального потока требуется найти такой путь потока, который может обеспечить максимально интенсивность потока.)

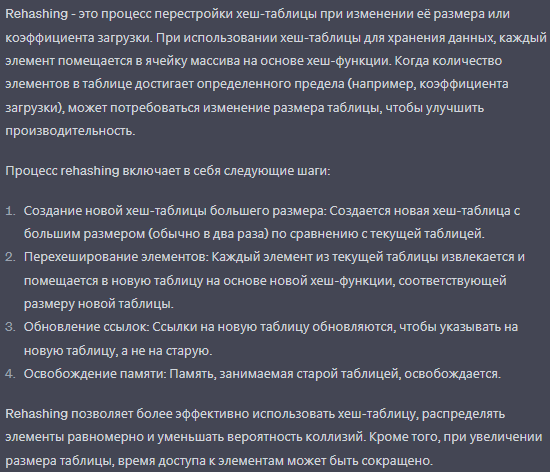


1. Хеширование, проблемы коллизий.

Хеширование — это преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины. Преобразование данных происходит с использованием математических формул (хеш-функций).  
Для эффективного хеширования и минимизации коллизий важно выбрать подходящую хеш-функцию. Хорошая хеш-функция должна равномерно распределять ключи по всему диапазону хеш-значений и обеспечивать минимальное количество коллизий.

Если коллизии становятся слишком частыми, можно применять различные стратегии для улучшения хеширования, такие как изменение размера хеш-таблицы, использование более сложных хеш-функций или применение более сложных методов разрешения коллизий.





1. Методы обработки коллизий в хешировании.

**ПУНКТ 36**

1. Хеш-таблица на основе перемешанной таблицы

Перемешанная таблица используется для хранения элементов, где ключи элементов преобразуются в адреса хранения в таблице. В общем случае, не всегда возможно обеспечить взаимно-однозначное преобразование ключа в адрес хранения элемента. В результате могут возникать коллизии, когда двум разным ключам соответствует один и тот же адрес в таблице. Чтобы снизить количество коллизий, используют функцию расстановки, которая стремится обеспечить равномерное отображение ключей в адреса хранения.

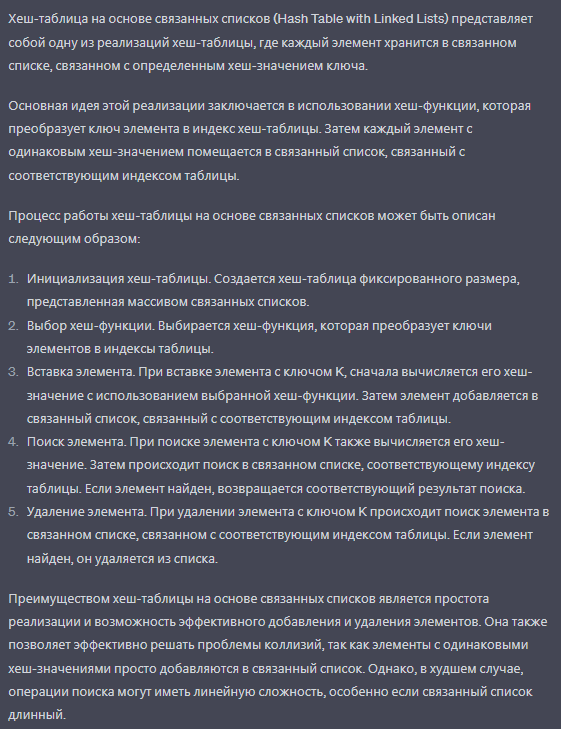
Перемешанная таблица состоит из основной области и области переполнения. Основная область отображается вектором и используется для хранения элементов, у которых нет коллизий. Область переполнения представляет собой семейство списков, где элементы помещаются при возникновении коллизий.

В таблицу можно включать новые элементы и выполнять операции поиска элементов. При включении элементов, пока не возникнет коллизия, они помещаются в основную область таблицы. При возникновении коллизии элементы помещаются в список, связанный с соответствующим элементом основной таблицы.

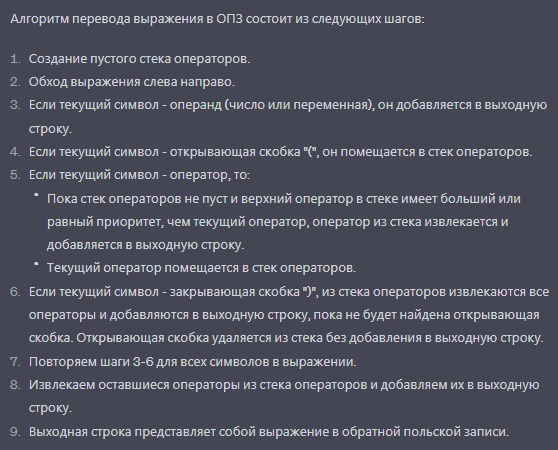
Один из распространенных методов перемешивания называется перемешивание сцеплением. При использовании этого метода таблица явно разделяется на основную область и область переполнения. Основная область отображается вектором, а область переполнения представляет собой семейство списков.

При операциях поиска элемента в таблице, необходимо просмотреть только элементы с одинаковыми значениями производного ключа. Средняя длина поиска в таблице может быть оценена с использованием формулы, которая зависит от размера основной области таблицы и общего количества элементов.

1. Хеш-таблица на основе связанных списков



1. Построение обратной польской записи выражения.

ОПЗ – форма записи математических выражений, в которой операторы расположены после своих операндов. В ОПЗ нет скобок и приоритет операций задаётся порядком следования операторов. Также ОПЗ называют постфиксной записью  
xy+ = x + y  
ОПЗ широко применяется в компьютерных выражениях, так как позволяет упростить вычисления и сократить использование памяти. Используется в калькуляторах, компьютерах, интерпретаторах и в некоторых базах данных.  


Пример обычного математического выражения и его эквивалентной записи в обратной польской нотации:

Обычная запись: 2 + 3 \* 4

Обратная польская запись: 2 3 4 \* +

Для вычисления значения выражения в обратной польской записи, мы используем стек данных. Мы проходим по выражению слева направо и выполняем следующие шаги:

Если текущий элемент является числом, помещаем его в стек.

Если текущий элемент является оператором, достаем два последних числа из стека, выполняем операцию и помещаем результат обратно в стек.

Повторяем шаги 1 и 2 для каждого элемента выражения.

По окончании выражения в стеке остается только одно число - результат вычисления выражения.

Пример вычисления значения выражения в обратной польской записи:

Выражение: 2 3 4 \* +

Шаги:

Добавляем 2 в стек: [2]

Добавляем 3 в стек: [2, 3]

Добавляем 4 в стек: [2, 3, 4]

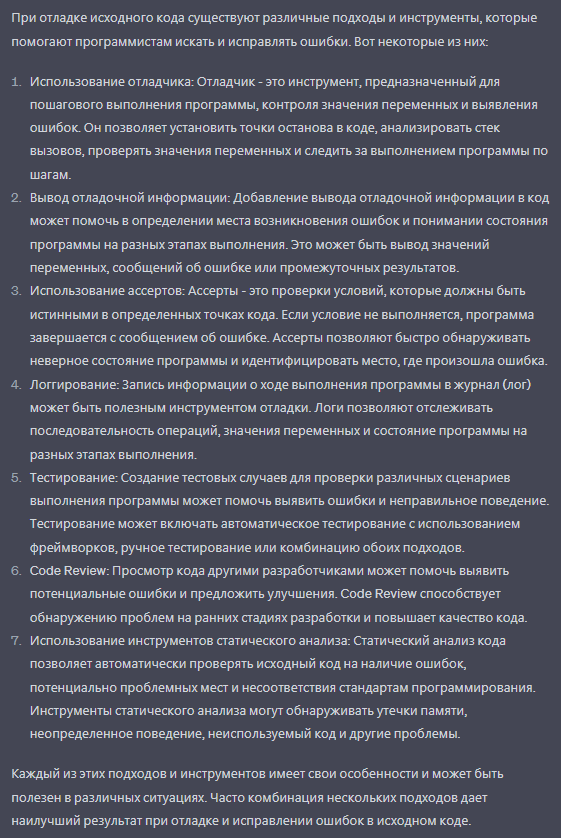
Выполняем операцию умножения 3 \* 4 = 12, получаем: [2, 12]

Выполняем операцию сложения 2 + 12 = 14, получаем: [14]

Результатом вычисления выражения в обратной польской записи "2 3 4 \* +" является число 14.

Обратная польская запись имеет несколько преимуществ, таких как отсутствие неоднозначности и необходимости использования скобок для определения порядка операций. Она также удобна для вычисления выражений с помощью стека данных и может быть использована в калькуляторах и интерпретаторах языков программирования.

1. Подходы и инструменты к отладке исходного кода



1. Директивы подпрограмм. Неявная рекурсия. Пример.

Директивы подпрограмм являются инструкциями, используемыми в программировании для определения и использования подпрограмм (также известных как функции, процедуры или методы). Директивы подпрограмм позволяют разработчикам создавать модульный и структурированный код, разбивая его на более мелкие и повторно используемые части.

