1. **Из каких частей состоит абстрактная машина, с которой мы работаем? На основе какой машины она построена?**

Абстрактная машина состоит из нескольких ключевых компонентов, включая процессор, оперативную память (RAM), устройства ввода-вывода и хранилище данных. Она базируется на физической машине, то есть реальном компьютере или аппаратуре.

1. **Дайте определение машинному слову. Попытайтесь предположить, как будет действовать машина с машинным словом, равным 4 (5, 6, 7, 8) байтам, в ответ на запрос передачи 3 байт информации между компонентами. Какова причина возникновения машинного слова?**

Машинное слово - это основная единица данных, с которой оперирует процессор. Если машинное слово состоит из 4 байтов, то при запросе передачи 3 байт информации между компонентами процессору потребуется выполнить дополнительные операции для выравнивания данных или использовать дополнительные биты для передачи необходимой информации. Причина возникновения машинного слова связана с эффективностью обработки информации. Обработка информации в блоках по 4 байта позволяет центральному процессору выполнять операции над этими блоками информации более эффективно, чем если бы он обрабатывал каждый байт отдельно. Это также упрощает архитектуру компьютера и позволяет более эффективно использовать его ресурсы. Если данные не выровнены по размеру машинного слова, могут возникнуть проблемы с производительностью из-за дополнительных операций с выравниванием данных в памяти. Процессоры могут иметь механизмы для обработки таких случаев, но это может привести к дополнительным издержкам. Дайте определения трансляции и транслятору.

1. **Дайте определения трансляции и транслятору.**

Трансляция - это процесс преобразования программного кода из одного языка в другой. Транслятор - это программное обеспечение, которое выполняет этот процесс.

1. **Дайте определение времени трансляции и времени выполнения.**

Время трансляции - это время, затраченное на преобразование исходного кода программы в машинный код, который может быть выполнен процессором. Это процесс, который обычно выполняется компилятором или интерпретатором. Время трансляции включает в себя анализ исходного кода, генерацию промежуточного представления программы, оптимизацию и генерацию конечного машинного кода.

Время выполнения - это время, затраченное на фактическое выполнение программы на компьютере. Это включает в себя время, затраченное на запуск программы, выполнение ее инструкций и завершение работы.

1. **Дайте определение интерпретатору и компилятору.**

Интерпретатор - это программа, которая выполняет исходный код программы построчно, преобразуя его в машинные команды на лету. Компилятор - это программа, которая преобразует весь исходный код программы в машинный код заранее, перед выполнением программы.

1. **Дайте определение слову языка программирования.**

Слово языка программирования - это наименьшая значимая единица информации в языке программирования. Слово в языке программирования представляет собой основные строительные блоки программы, включая ключевые слова, идентификаторы, литералы и операторы, которые определяют структуру и поведение программы.

1. **Дайте определение зарезервированному слову языка программирования.**

Зарезервированное слово языка программирования - это слово, которое имеет специальное значение в языке программирования и не может быть использовано в качестве идентификатора или переменной.

1. **Можно ли в программе на языке Си дать переменной имя «if» («IF», «cos», «If», «return», «whILE»)?**

В языке Си нельзя давать переменным имена, совпадающие с зарезервированными словами (if, while, return и т.д.). Имена с другим регистром (IF, whILE и т.д.) можно использовать

1. **Дайте определение точке входа и точке выхода.**

Точка входа - это место в программе, с которого начинается её выполнение. Точка выхода - это место в программе, где выполнение программы завершается.

1. **Может ли у приложения отсутствовать точка входа? Может ли она быть не единственна?**

У приложения обязательно должна быть точка входа, но она может не быть единственной.

1. **Может ли у приложения отсутствовать точка выхода? Может ли она быть не единственна?**

Точка выхода не обязана быть единственной и может отсутствовать, если программа выполняется бесконечно.

1. **Как можно найти точку входа в проекте на языке Си?**

Точку входа в проекте на языке Си можно найти по имени функции main(), которая является стартовой точкой выполнения программы.

1. **Препроцессор языка Си работает до этапа трансляции или после?**

Препроцессор языка C работает до этапа трансляции и выполняет различные преобразования над исходным кодом программы.

1. **Какие команды препроцессора Вы знаете?**

Некоторые команды препроцессора C: #define, #include, #ifdef, #endif, #ifndef, #else и т. д

1. **Что определяет язык программирования? Может ли транслятор не поддерживать стандарт?**

Язык программирования определяется набором правил синтаксиса и семантики, которые определяют его поведение. Транслятор может не поддерживать все стандартные функции языка программирования.

1. **Дайте определение стандартной библиотеке.**

Стандартная библиотека - это набор предопределенных функций и типов данных, которые предоставляются в рамках языка программирования.

1. **Какие модули стандартной библиотеки Вы уже знаете?**

Некоторые модули стандартной библиотеки C: stdio.h, math.h, limits.h, stdint.h и т. д.

1. **Можно ли давать свои макроопределения математических констант, которые уже есть в стандартной библиотеке?**

Это можно делать. Однако, использование своих макроопределений для констант, которые уже есть в стандартной библиотеке, может привести к проблемам. Во-первых, это может привести к путанице и сложностям при поддержке и обновлении кода. Во-вторых, это может привести к проблемам с совместимостью, если другие части кода или библиотеки ожидают использования стандартных констант.

1. **Можно ли не давать макроопределение константы, которая используется в программе более двух раз?**

Да, можно не давать макроопределение константы, которая используется в программе более двух раз. Однако, это может привести к снижению читаемости и поддержки кода. Макроопределения констант позволяют заменить повторяющиеся значения на более понятные имена. Это особенно полезно, когда константа используется несколько раз в программе. Однако, если константа используется только два раза или меньше, то использование макроопределения может быть излишним

1. **Дайте определение типу данных.**

Тип данных - это абстракция, которая определяет набор значений и операций, которые могут быть применены к этим значениям

1. **Дайте определение простому типу данных.**

Простой тип данных - это тип данных, который представляет собой единичное значение, например, целое число или символ.

1. **Обязана ли машина обеспечивать аппаратную поддержку какого-либо типа данных?**

Машина не обязана обеспечивать аппаратную поддержку любого типа данных, но должна поддерживать базовые операции с данными.

1. **Перечислите целочисленные типы в Си.**

Целочисленные типы в C: int, short, long, long long, и их беззнаковые версии.

1. **Каков размер каждого из целочисленных типов в Си? Дайте оценку максимальному числу в каждом из них.**

Размеры целочисленных типов в C зависят от архитектуры компьютера. Обычно int - 4 байта, short - 2 байта, long - 4 или 8 байт, long long - 8 байт

1. **Как ведут себя переменные каждого из целых типов при переполнении?**

Переменные целых типов ведут себя по-разному при переполнении. Например, при переполнении переменной типа int происходит переход к минимальному или максимальному значению, в зависимости от направления переполнения.

1. **Опишите концепцию опциональной особенности, обязательной при имплементации, на примере фиксированных целых в Си.**

Типы данных int16\_t, int32\_t, int64\_t является опциональной особенностью языка C, предоставляющей возможность использования фиксированного целого числа. В то же время, эта особенность является обязательной при имплементации согласно стандарту C99, что означает, что компиляторы, поддерживающие этот стандарт, должны предоставлять данный тип данных. Использование фиксированных целых типов данных может быть важно для портируемости программ между разными платформами, где размеры стандартных целых типов могут различаться. Поэтому использование опциональных, но обязательных при имплементации, особенностей языка программирования, таких как фиксированные целые типы, может повысить переносимость и надежность программного обеспечения.

Имплементация - программная или аппаратная реализация какого-либо протокола, алгоритма, технологии

1. **Перечислите типы чисел с плавающей точкой (ЧПТ) в Си. Связаны ли они с типами с теми же названиями в других языках? Опишите их связь со стандартом IEEE 754.**

Типы чисел с плавающей точкой в Си: float, double, long double. Стандарт IEEE 754 устанавливает форматы представления ЧПТ, спецификации для арифметики с плавающей точкой, обработку специальных значений (например, бесконечность, NaN) и операции над ними, форматы для одинарной и двойной точности. Этот стандарт принят многими языками программирования, и поэтому типы чисел с плавающей точкой, как правило, имеют схожие характеристики и поведение в различных языках.

1. **Можно ли сравнивать два ЧПТ через «двойное равно»? Как правильно сравнивать два ЧПТ? Как правильно сравнивать два ЧПТ, исходя из предположения, что все используемые ЧПТ близки к единице по модулю?**

ЧПТ нельзя сравнивать через “двойное равно”, так как ЧПТ не могут быть представлены точно. Они хранятся в виде: 1 бит - знак, 8 бит - порядок, 23 - мантисса. ЧПТ хранит 12-15 знаков после точки - то есть фиксированное число значений. Правильно сравнивать ЧПТ, беря их разницу по модулю и сравнивая ее с выбранной точность (обычно обозначается, как эпсилон):

fabs(a-b) < eps

1. **Дайте определение значащему разряду.**

Значащий разряд числа - это цифра, которая определяет его старший порядок разрядов, то есть влияет на значение числа в десятичной системе счисления.

1. **Найдите значение выражения: 1e-12 + 1e+12.**

1e+12

1. **Найдите значение выражения: 1e-12 - 1e+12.**

-1e+12

1. **Найдите значение выражения: 1e-16 + 1e-17.**

1.1e-16

1. **Найдите значение выражения: 1e+64 + 1e+64.**

2e+64

1. **Могут ли машина и человек предполагать разную точность у полученного в ходе вычислений ЧПТ?**

Могут, так как на хранение ЧПТ выделяется определенный участок памяти в компьютере и число переводится в двоичную СС, из-за чего также теряется точность. Из-за этого число в компьютере может отличаться от числа вычисленного человеком

1. **Найдите значение выражения: 3.1234567891234 + 2.1f.**

Будет примерно 5.223456. Дальнейшие цифры могут варьироваться в зависимости от машины. Это связано с неточным представлением ЧПТ в компьютере

1. **Опишите используемый Вами алгоритм выбора типов для решения задачи.**

Если ожидается введение пользователем целого числа в пределах 32 бит, а также выполнения операций всех кроме деления, то выбираю тип int. Если ожидается введение значений гораздо больших для решения задачи - long long int. Если ожидается, что будут использоваться вещественные числа или операция деления (не целочисленного), то выбираю тип double. Беззнаковый тип выбираю только если указано в условии задачи.

1. **Как работают явное и неявное приведения типов в Си?**

Явное приведение типов в Си: (имя типа) приводимое выражение. При попытке приведения к типу данных с наименьшим диапазоном значением потеряем часть информации

Неявное приведение типов в си: преобразование производит сам си, без участия человека. Числа приводятся к большему рангу при арифметических операциях, операциях сравнения и операциях присваивания. Может приводиться не к большему рангу, например: int32, in64 - если оба числа помещаются в int32, то int64 неявно приводится к int32

Если в арифметических операциях участвуют значения разных типов, то компилятор неявно пытается привести их к одному типу

1. **Правда ли, что в выражении (4 <= 3.9) транслятор сначала приведёт тип значения левого операнда к double?**

Да, так как перед выполнением сравнения транслятор сначала приведет операнды к высшему рангу. В данном случае высший ранг double (int -> double), поэтому 4 -> 4.0. Затем уже будет выполнена операция сравнения

1. **Можно ли поменять тип у переменной в языке Си?**

Нет, нельзя. Можно изменить тип значения переменной, но не тип самой переменной, так как в Си работает статическая типизация.

1. **Дайте определение переменной. Попытайтесь отталкиваться от знания составных частей абстрактной машины и свойств переменной, которых у неё не может не быть.**

Переменная – именованный участок памяти, обладающий некоторым типом. В С переменная сначала должна быть определена. Такое определение связывает переменную с ее типом и резервирует место в памяти. В контексте абстрактной машины, переменная может быть представлена как ячейка памяти, которая имеет уникальное имя и может хранить значения различных типов данных. Эта ячейка памяти может быть инициализирована значением при объявлении переменной, и затем это значение может быть изменено в процессе выполнения программы.

1. **Опишите концепцию статической типизации**

Приём, широко используемый в языках программирования, при котором переменная, параметр подпрограммы, возвращаемое значение функции связывается с типом в момент объявления и тип не может быть изменён позже (переменная или параметр будут принимать, а функция — возвращать значения только этого типа)

1. **Опишите тип данных int**

Размер int от short до long (т. к. точный размер зависит от машины). Используется для хранения целый чисел. Операции над типом int включают арифметические операции (сложение, вычитание, умножение, деление), операции побитового сдвига, операции сравнения и присваивания.

1. **Дайте определение замкнутому относительно операции 𝐴 типу данных.**

Тип данных замкнутый относительно операции - это тип данных, который при выполнении операции А возвращает сам себя. При /, % int на int вернется int - int замкнут относительно операции /, %

1. **Замкнут ли тип int относительно целочисленного деления? Относительно сложения?**

Относительно целочисленного деления и сложения int замкнут (при сложении, целочисленном делении двух int будет int)

1. **Замкнут ли тип double относительно умножения? Деления?**

Тип double не замкнут относительно умножения и деления. Может вернуть тип float

1. **Составьте «таблицы неявного приведения» для четырёх арифметических операций над числами типов int, long int, long long int, float, double, int32\_t, int64\_t.**

|  | int | long int | long long int | float | double | int32\_t | int64\_t |
| --- | --- | --- | --- | --- | --- | --- | --- |
| int | **int** | **long int** | **long long int** | **float** | **double** | int32\_t/int | **int64\_t** |
| long int | **long int** | **long int** | **long long int** | **float** | **double** | **long int** | **int64\_t** |
| long long int | **long long int** | **long long int** | **long long int** | **float** | **double** | **long long int** | long long int/int64\_t |
| float | **float** | **float** | **float** | **float** | **double** | **float** | **float** |
| double | **double** | **double** | **double** | **double** | **double** | **double** | **double** |
| int32\_t | int32\_t/int | **long int** | **long long int** | **float** | **double** | **int32\_t** | **int64\_t** |
| int64\_t | **int64\_t** | **int64\_t** | long long int/int64\_t | **float** | **double** | **int64\_t** | **int64\_t** |

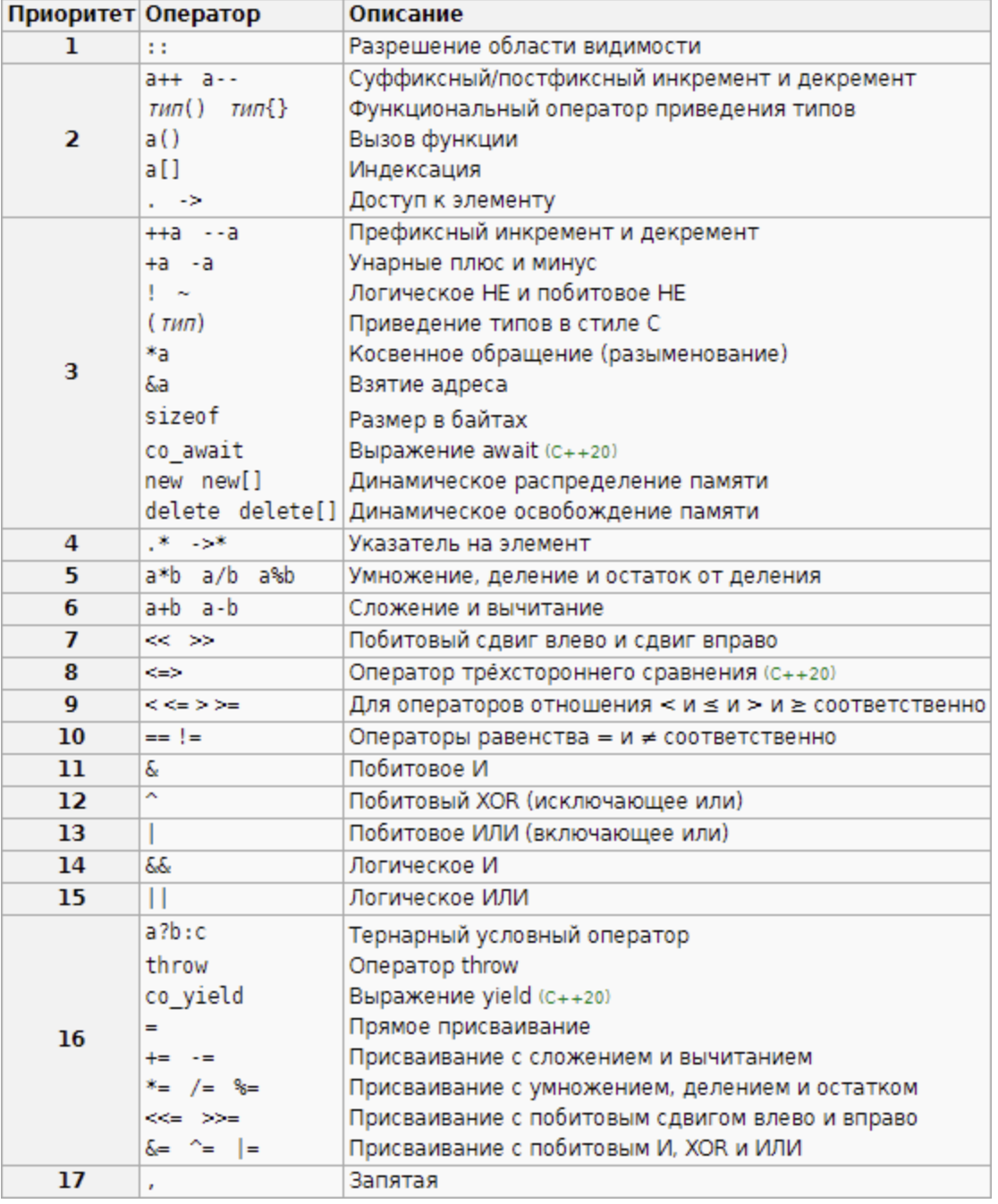
1. **Может ли целое число служить логическим выражением?**

Да. 0 = False, остальное = True

1. **Какое целое число всегда ложно? Только ли оно?**

Только 0 (с. 253, с. 43)

1. **Определён ли порядок арифметических вычислений?**

Да, определен 

Но не определен порядок вычисления подвыражений, т.е. в выражении (a + b) \* (c + d) не определено, какая скобка выполнится первой

1. **Укажите порядок выполнения операций: a+b+c+a+c+d+b**

((((((a+b)+c)+a)+c)+d)+b) - сложение левоассоциативно

1. **Укажите порядок выполнения операций: f(a) + f(b) + f(f(a))**

[[f(a) + f(b)] + f(f(a))] - так как операция сложения левоассоциативна, но вызов функций может происходить в любом порядке и строго не определен

1. **Можем ли мы так писать и почему: rc = scanf("%d", &a) + scanf("%d", &b);**

Можем, так как scanf возвращает количество введенных аргументов в типе данных int

1. **Можем ли мы так писать и почему: rc = scanf("%d%d", &a, a);**

Не можем, так как в scanf передается указатель на переменную, а переменная а имеет тип int

1. **Каковы особенности вычислений с ЧПТ, в частности, будет ли оптимизировано вычисление: x = y - y; z = y / y;**

Если y = 0, то при вычислении z возникнет ошибка. В остальных случаях x = 0, z = 1

1. **Найдите значение суммы 1039 слагаемых: 1e-40 + 1e-40 + ... + 1e-40 + 1.0**

1.1

1. **Определён ли порядок вычисления «аргументов» (левого и правого операндов) логических И/ИЛИ?**

Для операторов И/ИЛИ определен порядок вычисления аргументов. Сначала вычисляется левый операнд. Для И: если левый ложный, то правый не вычисляется и все выражение ложное. Для ИЛИ: если левый истинный, то правый не вычисляется и все выражение истинно

1. **Сколько существует булевых функций двух (трёх, 𝑛) аргументов?**

Существует 256 различных булевых функций от трех переменных. Для каждой из трех переменных есть 2 возможных значений (истина или ложь), следовательно, всего существует 2^2^3 = 256 различных комбинаций значений переменных для булевых функций от трех переменных. Следовательно для n аргументов существует 2^2^n булевых функций

1. **Опишите концепцию ленивого вычисления логических выражений в языке Си.**

Концепция ленивого вычисления логических выражений заключается в том, что вычисления прекращаются, когда результат становится определен (ex1 && ex2 - ex2 не вычисляется, если ex1 ложь) (*c.89*)

1. **Присутствует ли ленивая логическая схема в других языках? Можно ли на неё полагаться при написании программы на Си?**

Да, ленивая логическая схема есть в других языках: Haskell, Scala. Да, на них можно полагаться, так как она поддерживается всегда. Это может увеличивать производительность, пропуская определенные вычисления (*с. 89*)

1. **Можем ли мы так писать и почему: if(scanf("%d", &x) == 1 && x > 0)**

Да, можем, так как логические выражения выполняются слева направо, а значит после проверки введения x, он проверится еще на положительность

1. **Можем ли мы так писать и почему: if(scanf("%d", &x) && x > 0)**

Нет, не можем, так как левое условие не проверяет точно ли введен x. Если был введен EOF, то scanf вернет -1, но в х будет мусор, поэтому проверка x>0 будет некорректной

1. **Можем ли мы так писать и почему: rc = scanf("%d", &a); rc = rc || scanf("%d", &b);**

Мы можем так писать, так как если было подано хоть одно значение, то rc будет равняться 1, иначе 0

1. **Найдите значение выражения: 27 && 13 && 1 && 12.**

1, так как все числа ненулевые, а значит при выполнении операции && будет возвращаться 1

1. **Дайте определение операторным скобкам.**

Операторные скобки используются для задания порядка выполнения операций в выражениях

1. **Дайте определение оператору.**

Оператор в программировании - это команда, обозначающая определенное математическое или логическое действие, выполняемое с данными (операндами).

1. **Опишите синтаксис условного оператора.**

Синтаксис условного оператора:

if (выражение\_1)

оператор\_1

else if (выражение\_2)

оператор\_2

...

else if (выражение\_n)

оператор\_n

else

оператор

1. **Какие виды циклов Вы знаете? Какие ключевые слова использует каждый из них?**

* с постусловием - do-while
* с предусловием - while, for
* счетный - чаще всего имеется в виду for

1. **Чем отличаются выражение и инструкция? Верна ли с точки зрения синтаксиса каждая из следующих строк:**

* for (a = 2; for (a = 2; a < 54; a++); a++)
* for (for (a = 2; a < 54; a++); a < 54; a++)
* for (a = 2; a < 54; for (a = 2; a < 54; a++))
* Синтаксис неверен, вторым аргументом должно быть условие для завершения тела цикла
* Синтаксис неверен, первым аргументом должен быть шаг инициализации
* Синтаксис неверен, третьим аргументом должно быть задано выражение, выполняющееся в конце каждой итерации (изменение а)

1. **Можно ли через один цикл переписать любой другой? Если да, то через какой?**

Да, каждый цикл можно переписать через while

1. **Дайте определение числовой последовательности.**

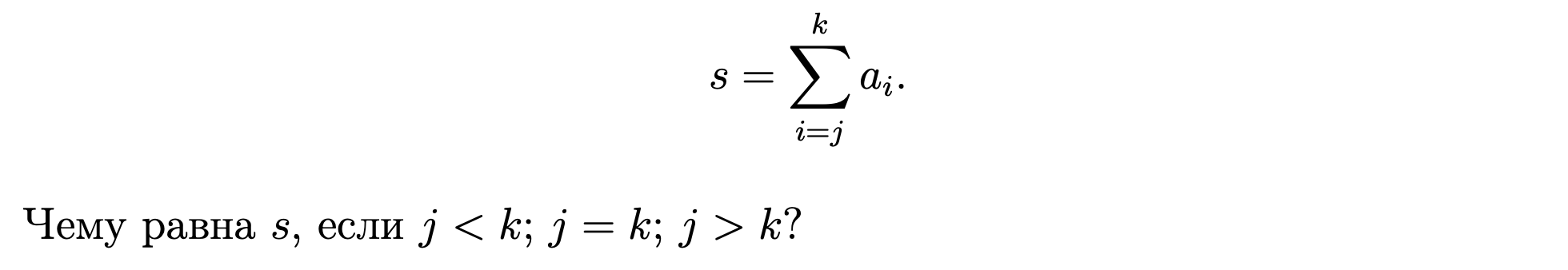
Числовая последовательность - это последовательность чисел, в которой каждый последующий элемент получается из предыдущего по определенному правилу.

1. **Дайте определение рекуррентной формуле.**

Рекуррентная формула - это формула, которая определяет последующие элементы последовательности через предыдущие

1. **Дайте определение простой рекуррентной формуле.**

Простая рекуррентная формула - это формула, в которой каждый элемент последовательности выражается через предыдущий однозначно

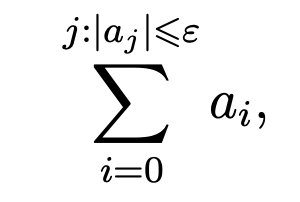
1. **Опишите правила чтения квантора суммы на примере**:

Если j < k: s равно сумме элементов с индексами от j до k

Если j=k: s равно значению aj

Если j > k: s равно 0

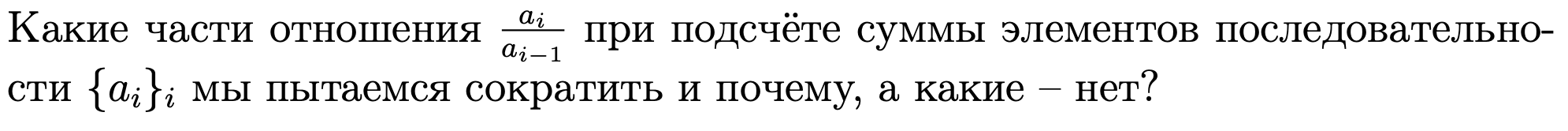
1. **Можно ли подсчитать, если последовательность {ai}i не является сходящейся?**



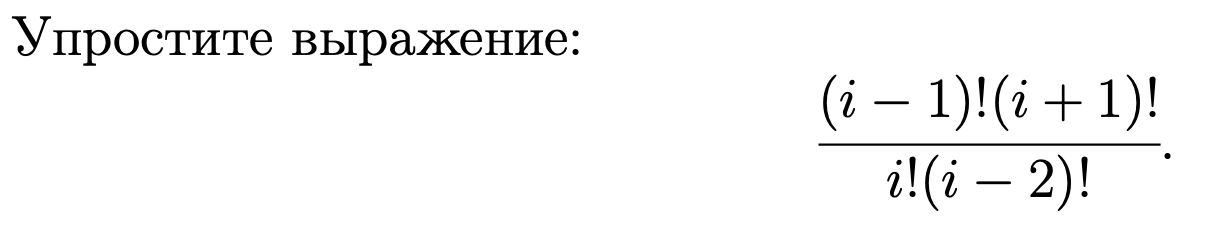
В случае расходящегося ряда это не будет иметь смысла, так как сумма расходящегося ряда бесконечна. Поэтому при любом взятом епсилон, мы получим сильные отличия от реальной суммы

1. **Приведите пример такой последовательности, сумма из предыдущего пункта которой, посчитанная до 𝜀 = 10-6, отличается от суммы ряда больше, чем на 10-6. Можно ли оценивать остаточный член суммы с помощью 𝜀 из задачи?**

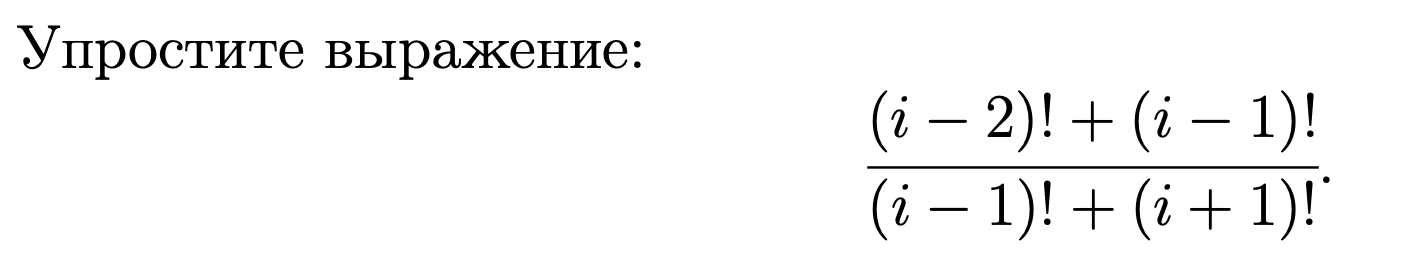
Пример - гармонический ряд. Даже подсчитав до 10-6 мы получим сильное отличие от реальной суммы, так как реальная сумма бесконечна

1. 

При подсчете суммы элементов последовательности мы пытаемся сократить те части отношения ai, которые сходятся к нулю при увеличении i, так как они не вносят существенного вклада в общую сумму. Однако те части, которые остаются значимыми при больших i, сохраняются в общей сумме.

1. 

(i-1)(i+1)

1. 

i/(i^3-1)

1. **Опишите концепцию ввода данных по концевому признаку.**

Концепция ввода данных по концевому признаку заключается в том, что программа продолжает читать данные из потока ввода до тех пор, пока не достигнет символа, обозначающего конец ввода (например, отрицательное число, EOF)

1. **Применим ли ввод по концевому признаку к задаче нахождения максимума (минимума) среди чисел, среднего арифметического (геометрического) чисел?**

Да