Урок 4. Еще немного математики. Код и псевдокод

Проценты

Дробь ¹⁄₁₀₀ получила название процент.

Дробь ¹⁄₁₀₀ означает, что нечто разделено на сто частей и от этих ста частей взята одна часть. Значит процентом является одна сотая часть.

Например, ¹⁄₁₀₀ от одного метра составляет 1 см. Один метр разделили на сто частей, и взяли одну часть (вспоминаем, что 1 метр это 100 см). А одна часть из этих ста частей составляет 1 см. Значит один процент от одного метра составляет 1 см.

²⁄₁₀₀ от одного метра уже составляет 2 сантиметра. В этот раз один метр разделили на сто частей и взяли оттуда не одну, а две части. А две части из ста составляют два сантиметра. Значит два процента от одного метра составляет 2 сантиметра.

Проценты встречались настолько часто, что люди заменили дробь ¹⁄₁₀₀ на специальный значок 1%. Также эта запись заменяет собой десятичную дробь 0,01 потому что если перевести обычную дробь ¹⁄₁₀₀ в десятичную дробь, то мы получим 0,01.

1% = ¹⁄₁₀₀ = 0,01

Нахождение процентов:
Чтобы найти процент от числа, нужно число умножить на процент, выраженный в виде десятичной дроби.

Пример:

Найти 50% от 2200 компьютеров.

Запись 50% заменяет собой запись пятьдесят сотых , а если перевести эти пятьдесят сотых в десятичную дробь, то мы получим 0,5
Теперь для нахождения 50% от 2200, достаточно будет умножить число 2200 на десятичную дробь 0,5

2200 × 0,5 = 1100

Задание для закрепления

1. Найти 40% от 300 евро.
2. Швейная фабрика выпустила 1200 костюмов. Из них 32% составляют костюмы нового фасона. Сколько костюмов нового фасона выпустила фабрика?

Нахождение числа по его проценту:
Зная процент от числа, можно узнать всё число.

Чтобы найти число по его проценту, нужно известное число разделить на данный процент, и полученный результат умножить на 100.

Пример:

Предприятие выплатило нам 6000 евро за работу, и это составляет 2% от общей прибыли, полученной предприятием. Зная свою долю, и сколько процентов она составляет, мы можем узнать общую прибыль.

Сначала нужно узнать сколько евро составляет один процент.

Если два процента от общей прибыли составляют 6000 евро, то нетрудно догадаться, что один процент составляет 3000 евро. Чтобы получить эти 3000 евро, нужно 6000 разделить на 2

6000 : 2 = 3000

Мы нашли один процент от общей прибыли, т.е. 1/100 . Если одна часть это 3000, то для определения ста частей, нужно 3000 умножить на 100

3000 × 100 = 300000

Мы нашли общую прибыль. Она составляет триста тысяч евро.

Задание для закрепления

Число 35 это 7% от какого-то неизвестного числа. Найти это неизвестное число.

Нахождение процентного отношения двух чисел:
Чтобы найти, сколько процентов одно число составляет от другого, нужно ту часть, о которой спрашивается, разделить на общее количество и умножить на 100%.

 **Пример:**

В секретном чате 25 человек. 10 из них — девочки. Сколько процентов девочек в чате?

Делим 10 на 25, полученную дробь переведем в проценты.

10/25 * 100% = 2/5 * 100% = 2 * 100/5 = 40%

Ответ: в чатике 40% девочек.

Задание для закрепления

1. В прошлом месяце билет стоил 110 евро. А в этом месяце на 12% больше. Сколько стоит билет в этом месяце?
2. Александру срочно понадобились деньги и он взял на один год в долг 7000 евро под 8% ежемесячно. Сколько денег он вернет через год?

Теория вероятности

Теория вероятности - это раздел математики, который изучает закономерности случайных явлений: случайные события, случайные величины, их свойства и операции над ними.

Теория вероятности простым языком (сокр. «тервер») — это особый раздел математики, который ищет закономерности в случайных событиях.

Математика — это точная наука цифр, поэтому она придает более точную возможность рассчитать исход случайного события.

Знать хотя бы основы теории вероятности нужно каждому человеку. Мы живем в непостоянном мире, где все построено на вероятностях и случайностях. Поэтому для развития правильного мировоззрения нужно хотя бы понимать, что и как может произойти.

Люди по натуре своей не сильно любят случайности. Большинству людей предпочтительнее постоянство, справедливость, определенность и объяснение всего происходящего. К примеру, в более ранние времена, люди не настолько были образованы и технологически подкованы, поэтому много чего происходящего они не способны были объяснить — так рождались суеверия и предрассудки. А подкрепляла их закономерность совпадений, так получилось и с «черной кошкой». Люди просто стали замечать частотные совпадения между неприятностью и черным котом, перешедшим дорогу.

Теория вероятности — это то, что актуально как в быту, так и в точных науках: математике, химии, генетике, программировании и т. д.

Если простым языком, то теория вероятности изучает весь наш быт и окружающий мир:

- случайность событий;
- случайность величин;
- случайность процессов;
- свойство и возможность контролировать эти случайности.

Главным словом в теории вероятности является само слово «вероятность». Люди очень часто в обычной жизни употребляют это слово, даже не обращая на него внимания:

- «Вечером, вероятно, будет дождь»;
- «На выходных, вероятно, я буду работать»;
- «Невероятно, как это получилось?»;
- «Есть вероятность, что я разбогатею»;

То есть, употребляя подобные фразы, люди интуитивно уже используют теорию вероятности, пытаясь предположить о том, что произойдет или не произойдет какое-то событие. Теория вероятности как математический раздел дает такую же оценку случайностям, но только используя цифры, формулы и закономерности.

 **Событие - это базовое понятие теории вероятности. События бывают достоверными, невозможными и случайными.**

- Достоверным является событие, которое в результате испытания обязательно произойдет. Например, камень упадет вниз.
- Невозможным является событие, которое заведомо не произойдет в результате испытания. Например, камень при падении улетит вверх.
- Случайным называется событие, которое в результате испытания может произойти, а может не произойти. Например, из колоды карт вытащили туза.

 **Исход - это любой результат испытания, который представляет собой появление определенного события.**

В частности, при подбрасывании монеты возможно 2 исхода (случайных события): выпадет орёл, выпадет решка. Естественно, подразумевается, что данное испытание проводится в таких условиях, что монета не может встать на ребро или, скажем, зависнуть в невесомости.

Остановимся подробнее на классической вероятности:

Вероятностью события A называют отношение числа m благоприятствующих этому событию исходов к общему числу n всех равновозможных несовместных элементарных исходов, образующих полную группу:

$$P(A)=m/n$$

Свойство 1. Вероятность достоверного события равна единице

Свойство 2. Вероятность невозможного события равна нулю.

Свойство 3. Вероятность случайного события есть положительное число, заключенное между нулем и единицей.

Итак, вероятность любого события удовлетворяет двойному неравенству $0 \le P(A) \le 1$

 **Пример:**

1. На экзамен вынесено 60 вопросов, Андрей не выучил 3 из них. Найдите вероятность того, что ему попадется выученный вопрос.

   Решение.

Андрей выучил 60 − 3 = 57 вопросов. Поэтому вероятность того, что на экзамене ему попадется выученный вопрос равна

$$\frac{57}{60} = \frac{19}{20} = 0,95.$$

2. В среднем из 1400 садовых насосов, поступивших в продажу, 7 подтекают. Найдите вероятность того, что один случайно выбранный для контроля насос не подтекает.

   Решение.

   В среднем из 1400 садовых насосов, поступивших в продажу, 1400 − 7 = 1393 не подтекают. Значит, вероятность того, что один случайно выбранный для контроля насос не подтекает, равна

$$\frac{1393}{1400} = 0,995.$$

3. В случайном эксперименте симметричную монету бросают дважды. Найдите вероятность того, что орел выпадет ровно один раз.

   Решение.

   Равновозможны 4 исхода эксперимента: орел-орел, орел-решка, решка-орел, решка-решка. Орел выпадает ровно один раз в двух случаях: орел-решка и решка-орел. Поэтому вероятность того, что орел выпадет ровно 1 раз, равна

$$\frac{2}{4} = \frac{1}{2} = 0,5.$$

Задание для закрепления

1. В фирме такси в данный момент свободно 20 машин: 10 черных, 2 желтых и 8 зеленых. По вызову выехала одна из машин, случайно оказавшаяся ближе всего к заказчице. Найдите вероятность того, что к ней приедет зеленое такси.

   Вероятность того, что к заказчице приедет зеленое такси равна

   $$\frac{8}{20} = \frac{4}{10} = 0,4.$$

2. В случайном эксперименте симметричную монету бросают трижды. Найдите вероятность того, что выпадет хотя бы две решки.

   Решение.

   Всего возможных исходов — 8: орел-орел-орел, орел-орел-решка, орел-решка-решка, орел-решка-орел, решка-решка-решка, решка-решка-орел, решка-орел-орел, решка-орел-решка. Благоприятными являются четыре: решка-решка-решка, решка-решка-орел, решка-орел-решка, орел-решка-решка. Следовательно, искомая вероятность равна 4 : 8 = 0,5.

3. В случайном эксперименте бросают две игральные кости. Найдите вероятность того, что в сумме выпадет 8 очков. Результат округлите до сотых.

   Количество исходов, при которых в результате броска игральных костей выпадет 8 очков, равно 5: 2+6, 3+5, 4+4, 5+3, 6+2. Каждый из кубиков может выпасть шестью вариантами, поэтому общее число исходов равно 6·6 = 36. Следовательно, вероятность того, что в сумме выпадет 8 очков, равна

   $$\frac{5}{36} = 0,138...$$

Алгебра событий

1. Сложение событий

   Суммой двух событий A и B называется событие A+B, которое состоит в том, что наступит или событие A, или событие B, или оба события одновременно. В том случае, если события несовместны, последний вариант отпадает, то есть может наступить или событие A, или событие B.

   Несовместными называются события, в которых появление одного из событий исключает появление другого (при условии одного и того же испытания). Простейшим примером несовместных событий является пара противоположных событий.

2. Умножение событий

   Произведением двух событий A И B называют событие AB, которое состоит в совместном появлении этих событий. Иными словами, умножение AB означает, что при некоторых обстоятельствах наступит и событие A, и событие B.

Есть несколько подходов для исчисления теории вероятности. Самые простые из них мы с вами разберем, чтобы сформировать понимание о том, что такое тервер.

1. Вероятность и зависимые события

Этот метод используется, когда нужно определить вероятность в событиях, которые взаимосвязаны и зависят от исходов друг друга. Приведем простой пример.

Вы решили подарить другу на день рождения торт. Заказали курьерскую доставку торта, указали улицу, дом, подъезд, этаж, но не знаете номер квартиры, отсюда возникает вероятность того в какой квартире наш друг. Если мы НЕ указали квартиру, то друг всегда в одной квартире и нет вероятности его перемещения. Поэтому перед доставщиков торта будет выбор среди 3-х дверей. Теперь можно рассчитать, какова вероятность, что курьер попадет к другу с первого же раза.

Со стороны доставщика имеем 3 вероятных события:

- Доставщик постучит в 1-ю дверь;
- Доставщик постучит во 2-ю дверь;
- Доставщик постучит в 3-ю дверь.

Но в нашу статистику включается еще и друг. Он тоже добавляет 3 вероятных события:

- Друг может оказаться за 1-й дверью;
- Друг может оказаться за 2-й дверью;
- Друг может оказаться за 3-й дверью.

Вот и получается, что у нас может быть 9 вариантов развития событий: 3*3=9. Из них положительных вариантов, когда курьер позвонит в дверь к другу, — 3. Поэтому если отследить вероятность, что с первого раза курьер попадет в нужную дверь, то получается: 3/9 или 1/3.

2. Вероятность и независимые события

В данной ситуации искомая вероятность не зависит от благоприятного исхода событий и, соответственно, события не имеют влияния между собой.

Данный вид вероятности получается, когда решения принимаются с помощью монеты. То есть, загадывая на «орла», шанс, что выпадет именно он, равен 50% или ½.

Если бросков несколько подряд, то вероятность, что очередной раз выпадет «орел», уменьшается. Это происходит, потому что вступает в бой вероятность последовательности. То есть когда вы бросаете один раз, то вариантов два: «орел» или «решка», или ½, как мы уже говорили. Но если бросаете 5 раз подряд, то вариантов куда больше и шанс, что выпадет: «орел», «орел», «орел», «орел», «орел» — невелик. Рассчитывается так: ½ * ½ * ½ * ½ * ½ = 1/32.

3. Условные вероятности

Условные вероятности возникают в том случае, когда шанс, что наступит какое-то событие, зависит от какого-то условия. Это очень хорошо видно, когда стоит вопрос о погоде:

Есть ли вероятность, что идет дождь, когда вы слышите громовые раскаты?

Есть ли вероятность, что идет дождь, когда вы видите, что на улице солнце?

Тут хорошо прослеживается, что если слышны/видны определенные условия, то вероятность становится больше/меньше.

Например, в урне находятся 3 белых шара и 2 черных. Из урны вынимается один шар, а затем второй. Событие *B* – появление белого шара при первом вынимании. Событие *A* – появление белого шара при втором вынимании.

Очевидно, что вероятность события *A*, если событие *B* произошло, будет

$$P(A \mid B) = \frac{2}{4} = \frac{1}{2}.$$

Вероятность события *A* при условии, что событие *B* не произошло, будет

$$P(A \mid \overline{B}) = \frac{3}{4}.$$

Алгоритмы

Алгоритмы - это последовательность шагов или инструкций, предназначенных для выполнения определенной задачи или решения определенной проблемы.

Алгоритмы используются в различных областях, включая математику, информатику, программирование, а также повседневную жизнь.

Алгоритмы можно сравнить с рецептом приготовления блюда. Подобно тому, как рецепт дает нам последовательность шагов для приготовления пищи, алгоритм предоставляет последовательность действий для достижения желаемого результата. Важно, чтобы алгоритм был последовательным и понятным, чтобы другие люди могли следовать этим инструкциям и достичь того же результата.

Представь, что у тебя есть задача сделать бутерброд. Давайте составим алгоритм для этой задачи:

1. Возьми два куска хлеба.
2. Возьми нож и масло.
3. Положи масло на нож и намажь один кусок хлеба.
4. Возьми нарезанные продукты (например, ветчину, сыр, овощи).
5. Положи нарезанные продукты на намазанный кусок хлеба.
6. Положи второй кусок хлеба сверху.
7. Готово! Ты сделал бутерброд.

Пример алгоритма можно представить в виде задачи по нахождению наибольшего числа в списке:

Пример:

Найти наибольшее число в списке целых чисел.

Алгоритм:

1. Инициализировать переменную max первым элементом списка.
2. Для каждого числа x в списке:
   Если x больше значения переменной max, присвоить макс значение x.

3. После прохода по всем числам, значение переменной max будет наибольшим числом в списке.

В результате выполнения этого алгоритма мы найдем наибольшее число в заданном списке.

Алгоритмы помогают систематизировать и структурировать решение задач, делая процесс решения более понятным и эффективным.

Введение вычислительной сложности и использование метрик производительности алгоритмов - важные концепции при анализе и выборе алгоритмов для решения задач. Давайте подробнее разберем эти понятия:

- Вычислительная сложность: Это понятие отражает, насколько алгоритм эффективен с точки зрения времени выполнения в зависимости от размера входных данных. Она позволяет понять, как алгоритм будет вести себя в худшем случае. Вычислительную сложность можно оценить как функцию от размера входных данных (часто обозначаемого как "n"). Быстрорастущая сложность может привести к значительному увеличению времени выполнения на больших данных. Сложность бывает временная (количество операций для получения результата) и емкостная (количество памяти, нужное для получения результата).

- Память: Память, которую потребует алгоритм, также важна. Она может быть выражена через количество дополнительной памяти, необходимой для хранения временных данных и структур, используемых в алгоритме. Память может оказать влияние на производительность, особенно если алгоритм требует большого объема дополнительной памяти.

- Метрики производительности: Оценка производительности алгоритмов связана с измерением количества операций, которые алгоритм выполняет в зависимости от размера входных данных (часто опять же обозначаемого как "n"). Например, в сортировке это может быть количество сравнений и обменов элементов. Такие метрики позволяют сравнивать различные алгоритмы и выбирать наиболее подходящий для конкретной задачи.

Что такое код

Программирование – это процесс превращения алгоритмов в специальный набор инструкций, который может выполнить компьютер.

Программный код — это совокупность инструкций, написанных на определенном языке программирования, которые компьютеры могут выполнять для выполнения различных задач.

Код — это основа любой программы или приложения, позволяющая разработчикам описывать, как должна работать программа, какие действия она должна выполнять и как взаимодействовать с пользователем.

Программный код необходим для создания программного обеспечения, начиная от простых приложений до сложных систем. Он используется для автоматизации задач, управления оборудованием, разработки веб-сайтов, мобильных приложений, игр, научных исследований и многого другого. Код позволяет превратить идеи и алгоритмы в работающие решения, которые могут выполняться на компьютерах, серверах и других устройствах.

Каким бывает код:

- **Машинный код** состоит из инструкций, которые могут непосредственно исполняться процессором. Эти инструкции представлены в виде двоичного кода и являются низкоуровневыми, то есть сложными для понимания человеком.
- **Ассемблерный код** — это низкоуровневый язык программирования, который использует запоминание для представления машинных команд. Он ближе к машинному коду, но более понятен для человека.
- **Высокоуровневые языки программирования**, такие как Python, JavaScript, Java и C++, позволяют писать код, который более понятен человеку. Эти языки абстрагируют сложные детали аппаратного обеспечения, делая программирование более удобным и эффективным.

В чем пишут код:

Код пишется в текстовых редакторах или интегрированных средах разработки (IDE).

Текстовые редакторы, такие как Notepad++, Sublime Text и Visual Studio Code, позволяют писать и редактировать код с подсветкой синтаксиса и другими полезными функциями.

IDE, такие как PyCharm для Python или Visual Studio для C#, предлагают более широкий набор инструментов, включая отладчики, средства тестирования и автоматизации сборки, которые облегчают разработку сложных проектов.

Из чего состоит код:

Набор правил, по которым пишется код, называется синтаксисом. Синтаксис поясняет, какие команды можно использовать, какой должна быть структура кода, как правильно расставлять связи, передавать аргументы и использовать разные операторы. Его можно сравнить с правилами любого естественного языка.

Синтаксис языка программирования ничего не говорит о смысле программы. Он отвечает только за правильность написания.

Код состоит из команд, связей между ними и других элементов синтаксиса. Вот какими они бывают.

Сначала договоримся об общих понятиях.

 **Командами мы будем называть непосредственные указания для компьютера, что сделать. Например, напечатать слово: print(“слово”).**

 **Связями будем называть разные элементы, связывающие команды друг с другом. Чаще всего это знаки пунктуации и различные операторы.**

Программный код состоит из следующих основных элементов:

 **Переменные. Когда пользователь оперирует какими-то значениями по нескольку раз, ему бывает нужно куда-то их записать. Для этого в языках программирования существуют переменные. У переменной есть имя, тип и значение.**

- Имя показывает, как обращаться к переменной. Например, если мы объявили a = 5, то переменная называется a.
- Значение – это данные, которые лежат в переменной. Для названной выше переменной a это число 5.
- Тип данных показывает, какой вид информации находится в переменной: число, буква, строка или что-то более сложное. Есть простые и составные типы данных. В первых хранятся примитивные значения вроде чисел и строк, во вторых – сложные конструкции из нескольких примитивов или даже функций.

📎**Константы. Так называют переменные, значение которых нельзя изменить. Оно задается раз и навсегда.**

📎**Ключевые слова. Ключевые слова — это особые зарезервированные слова, которые используются для технических целей.**

Например, значения True и False, «истинно» или «ложно». Зачастую эти слова — не команды: они рассказывают компьютеру о каком-то значении или формате.

📎**Идентификаторы. Это имена, которые программисты дают сущностям в коде.**

📎**Значения и литералы. Литералы еще называют безымянными константами. Это значения какого-то типа, которые используются в коде, но не привязаны к переменной.**

Например, когда мы пишем print(“слово”), строка «слово» — это литерал.

📎**Знаки пунктуации и символы. Символы чаще всего бывают связями. Иногда — операторами. Это «знаки препинания» для языка программирования: точка, двоеточие, запятая, точка с запятой и так далее.**

Они помогают структурировать программу. Например, скобки () после функции обрамляют данные, которые нужно передать ей при запуске. А сами данные перечисляются через запятую, чтобы отделить одно от другого.

📎**Операции, операторы и операнды. Операции — это определенные действия с данными: сложение, вычитание, сравнение и так далее.**

Причем речь не всегда идет о действиях в математическом смысле — это просто хороший наглядный пример.

Операции состоят из операндов и операторов.

- Операнд – это переменная или литерал, что-то, с чем мы будем работать.
- Оператор – это символ или слово для обозначения действия.

Например, в операции a + 2 переменная a и литерал 2 будут операндами, а знак + оператором.

Функции. Иногда набор команд бывает нужно объединить в один блок, чтобы потом вызывать его как одну большую команду. Это возможно. Такие блоки в программировании называются функциями.

У функции чаще всего есть имя (исключения встречаются, но редко) и список аргументов — данных, которые передаются ей при вызове. Когда программист вызывает функцию, она выполняет заложенные в ней действия.

Комментарии. В большинстве языков есть возможность писать комментарии — текстовые блоки, которые ничего не делают и нужны для удобства разработчика.

Они выделяются специальными символами. Компилятор или интерпретатор игнорирует комментарии и ничего с ними не делает.

Как выглядит программный код на языке Python:

```
JavaScript
def greet(name):
    return f"Hello, {name}!"

print(greet("World"))
```

В этом примере на Python определена функция greet, которая принимает имя и возвращает приветственное сообщение. Затем эта функция вызывается с аргументом "World".

Псевдокод

Псевдокод — это неформальный способ описания алгоритма, который использует обычный язык и понятные для человека выражения.

Псевдокод помогает разработчикам и другим заинтересованным сторонам понять, как работает алгоритм, без необходимости писать реальный программный код. Это отличный инструмент для планирования и обсуждения программных решений.

Псевдокод используется для планирования и документирования алгоритмов до их реализации в реальном программном коде. Он служит мостом между идеями и реальным кодом, помогая:

- Объяснять сложные алгоритмы простым языком.
- Понимать и обсуждать логику программы до ее написания.
- Упрощать процесс программирования, разбивая задачу на более мелкие шаги.

Псевдокод помогает в решении следующих задач:

- **Планирование**: Он позволяет разработчикам детально продумывать алгоритмы перед их реализацией.
- **Документация**: Псевдокод служит отличным средством для документирования алгоритмов и их логики.
- **Обучение**: Это полезный инструмент для обучения программированию, так как помогает понять основы алгоритмизации без изучения синтаксиса конкретного языка программирования.
- **Коммуникация**: Псевдокод облегчает обсуждение алгоритмов между разработчиками, дизайнерами и менеджерами проектов.

Использование псевдокода имеет несколько преимуществ:

- **Простота**: Псевдокод прост для понимания и написания, что делает его доступным для широкого круга людей.
- **Гибкость**: Его можно адаптировать под любые требования и цели, так как он не привязан к конкретному языку программирования.
- **Ясность**: Он помогает четко представить структуру и логику алгоритма, что упрощает его реализацию в коде.
- **Универсальность**: Псевдокод может быть использован на любом этапе разработки, от первоначального планирования до финальной документации.

Вот несколько шагов для написания псевдокода:

1. **Определите задачу**: Четко сформулируйте, что нужно решить.
2. **Разбейте задачу на подзадачи**: Разделите основную задачу на более мелкие шаги.
3. **Определите входные и выходные данные**: Укажите, какие данные необходимы для выполнения алгоритма и какие результаты ожидаются.
4. **Опишите основные шаги алгоритма**: Используя обычный язык, опишите каждый шаг алгоритма.
5. **Используйте стандартные конструкции**: Применяйте условные операторы (если...то), циклы (для, пока) и другие стандартные конструкции для описания логики.
6. **Проверьте логичность и последовательность**: Убедитесь, что ваш псевдокод логичен и последователен.

После написания псевдокода важно проверить его на наличие ошибок и логических несоответствий. Задайте себе несколько вопросов:

1. Поймет ли этот псевдокод пользователь, который не знает процесс?
2. Как просто будет «перестроить» псевдокод в код используемого языка программирования?
3. Полностью ли описаны действия, все ли моменты учтены?
4. Поймёт ли пользователь все наименования объектов?

При написании псевдокода часто используются следующие стандартные конструкции:

Условные операторы:

```
Unset
Если (условие) Тогда
    действие1
Иначе
    действие2
Конец Если
```

Циклы:

```
Unset
Для i от 1 до n
    действие
Конец Для
```

```
Unset
Пока (условие)
    действие
Конец Пока
```

Функции и процедуры:

```
Unset
Функция имя_функции(параметры)
    действие
    вернуть значение
Конец Функции
```

```
Unset

Процедура имя_процедуры(параметры)

       действие

Конец Процедуры
```

Пример:

Допустим, вы хотите создать программу для подсчета среднего балла студентов по их оценкам. Вместо того, чтобы сразу погружаться в сложный код на языке программирования, вы начинаете с псевдокода, чтобы упростить задачу.

```
Unset

Начало
```

```
  Создать список оценок студентов
  Сумма = 0


Начало цикла 1
возьмем первого студента
Для каждой оценки в списке оценок:
Начало цикла 2
Переберем все оценки для первого студента
Добавить оценку к Сумме
   Средний балл = Сумма / Количество оценок в списке
  Вывести Средний балл
конец цикла 2
конец цикла 1
Конец
```

Пояснение:

Создать список оценок студентов – здесь мы имеем в виду, что у нас уже есть набор данных с оценками, который мы хотим обработать.

Сумма = 0 – начинаем с нуля, чтобы постепенно добавлять к этому числу все оценки.

Для каждой оценки в списке оценок – это цикл, который проходит по всем элементам списка (оценкам), чтобы выполнить определенные действия для каждой оценки.

Добавить оценку к Сумме – здесь каждую оценку из списка мы прибавляем к общей сумме, чтобы в итоге получить общую сумму всех оценок.

Средний балл = Сумма / Количество оценок в списке – после того как мы нашли сумму всех оценок, мы делим эту сумму на количество оценок, чтобы найти средний балл.

Вывести Средний балл – последний шаг, где мы показываем результат нашей программы, то есть средний балл студентов.

Задание для закрепления

Задача 1: Написать алгоритм для нахождения максимального из трех чисел.

Псевдокод:

```
Unset
Вход: три числа a, b, c
взять первое число a
Если (a > b) Тогда
    Если (a > c) Тогда
        вернуть a
    Иначе
        вернуть c
Иначе
    Если (b > c) Тогда
        вернуть b
    Иначе
        вернуть c
Конец Если
```

Объяснение: Этот псевдокод сравнивает три числа и возвращает наибольшее из них. Сначала проверяется, больше ли a, чем b. Если да, то далее проверяется, больше ли a, чем c. Если нет, то проверяется, больше ли b, чем c.

Задача 2: Написать алгоритм для проверки, является ли данное число четным.

Псевдокод:

```
Unset
Вход: число n
Если (n % 2 == 0) Тогда
    вернуть "Четное"
Иначе
    вернуть "Нечетное"
Конец Если
```

Объяснение: Этот псевдокод проверяет остаток от деления числа n на 2. Если остаток равен нулю, то число четное, иначе — нечетное.

Задача 3: Написать алгоритм для нахождения суммы чисел от 1 до n.

Псевдокод:

```
Unset
Вход: число n
сумма = 0
Для i от 1 до n
    сумма = сумма + i
Конец Для
вернуть сумма
```

Объяснение: Этот псевдокод использует цикл для добавления каждого числа от 1 до n к переменной сумма, после чего возвращает итоговую сумму.

Задача 4: Написать алгоритм для определения, является ли данное число положительным, отрицательным или нулем.

Псевдокод:

```
Unset
Вход: число n
Если (n > 0) Тогда
    вернуть "Положительное"
Иначе Если (n < 0) Тогда
    вернуть "Отрицательное"
Иначе
    вернуть "Ноль"
Конец Если
```

Объяснение: Этот псевдокод проверяет, больше ли число нуля, меньше ли оно нуля или равно нулю, и возвращает соответствующее сообщение.