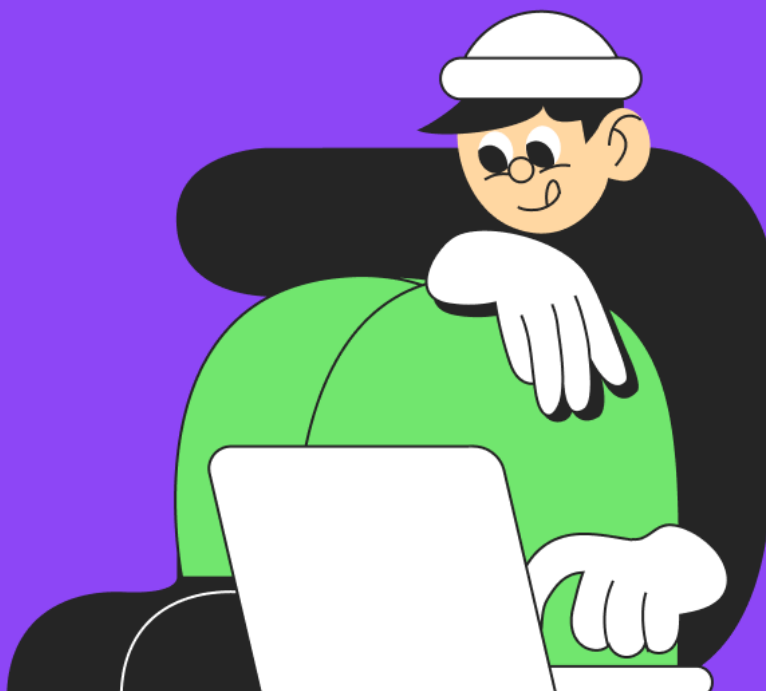
GeekBrains

Введение в информатику



Оглавление

Приветствие

На этом уроке

Что такое информатика

- Разделы информатики
- С чем работает информатика

Количество информации

- Как оценить, достаточно ли мы получили информации
- Свойства информации
- Численная мера информации
- Задача 1. Как оценить количество полученной информации?
  - Решение

Информация и компьютер

- Каким образом эти нолики и единички хранятся в компьютере?
- Надо ли программисту знать физику

Типы информации

- Тип информации: текст
  - Задача 2. Самостоятельная работа
    - Решение
  - Сжатие информации
  - Распространённые таблицы кодировок
- Тип информации: числа
  - Задача 3. Сложение чисел
    - Решение
  - Задача 4. Умножение чисел
    - Решение
  - Задача 5. Самостоятельная работа

Решение

Тип информации: числа беззнаковые целые

Зачем нужны системы счисления и как между ними путешествовать

Задача 6. Перевод числа в двоичное представление

Решение

Зачем нужны разные системы счисления?

Вывод

Тип информации: графическая информация

Заключение

[00.00.43]

Приветствие

Приветствую вас на вводной лекции по информатике. Сегодня мы поговорим о школьной информатике. Большая часть информации, скорее всего, вам уже известна, и вы просто повторите материал. А те, кто только знакомится с предметом, откроет для себя что-то новое.

Когда я готовил эту презентацию, в шаблоне был слайд «Здесь должна быть информация, за которую меня должны зауважать студенты». Не знаю, насколько вы меня зауважаете, но о себе несколько слов скажу.

В сферу IT я вошёл в далёком, по-моему, 2011 или 2012 году. Это был стартап по разработке игр. Мы с ребятами создали интересный, в какой-то степени, проект, в рамках которого пытались обучать логическому мышлению через игры путём нахождения стратегии. Чтобы в них выигрывать, требовалось делать определённые шаги. Такие игры мы и создавали.

Следующим этапом, когда переехал в Москву, решил посотрудничать с Microsoft. Я попал в студенческую программу партнёров Microsoft, где был лидером России два года.

В 2016 году я стал членом жюри конкурса Microsoft Imagine Cup, о котором вы наверняка слышали. Это достаточно интересный конкурс, особенно для тех, кто хочет войти в IT.

В 2018 году в сотрудничестве со Skillbox я занялся одним интересным проектом. Стал автором программы курса «Разработчик C#», поэтому кто-то меня уже знает.

В 2020 году я покинул большинство своих проектов и занялся YouTube. Так что меня можно найти и там. Поэтому обязательно посмотрите мой канал и, возможно, вы откроете для себя что-то новое.

[00.00.43]

На этом уроке

1. Узнаем, что такое информатика.
2. Поговорим о количестве информации.
3. Разберём типы информации.

[00.02.37]

Что такое информатика

В большинстве случаев информатика занимается сбором, хранением, обработкой, передачей и анализом информации. В некоторых случаях добавляется такой пункт, как оценка. Но главная задача информатики — принимать какие-то решения.

Например, если у нас есть просто информация, то непонятно, что с ней делать. А если мы получили информацию для какой-то цели, в этом случае всё прекрасно — именно такая информатика нам и нужна.

[00.03.10]

Разделы информатики

В большинстве случаев под информатикой люди понимают некий условный скучный курс, где надо открыть «Вордик» или сделать что-то подобное. На самом деле информатика — это довольно интересная наука, которая делится на три больших раздела.

Каждый раздел занимается чем-то своим. В рамках этой лекций текущего курса мы познакомимся с базами данных. Но это будет немного позже. Мы, конечно же, затронем тему алгоритмов и структур данных. Те, кто захочет пойти в компьютерную безопасность или в кибербезопасность, обязательно столкнулся с компьютерными сетями. А те, кто уйдёт в программирование, затронут тему анализа, чтобы делать приложения более

производительными и т. д. Поэтому отмечу сразу, что здесь мы проходим всё-таки школьный курс. Не будем вникать в какие-то суперподробности, поговорим о них позже.

[00.04.10]

С чем работает информатика

Информатика работает с информацией. Технически мы получаем её посредством различных органов чувств, например, что-то увидев, услышав или потрогав. В большинстве случаев после получения информации мы понимаем, достаточно ли нам её для принятия решения.

[00.04.31]

Количество информации

Как оценить, достаточно ли мы получили информации

Допустим, мы увидели чёрный квадрат. Да, чёрный квадрат, размещённый на тёмном фоне, даёт меньше информации. Тем не менее подумаем, сколько информации мы получили. Кто-то скажет, что это просто чёрный квадрат. Теперь добавим немного больше информации, например, рамку. У кого-то сразу же возникнет мысль, что это чёрный квадрат К. С. Малевича. На основе этого приходит ещё больше информации:

- смысл, который заложил автор;
- для чего этот чёрный квадрат;
- что было до него: ходят слухи о какой-то замалёванной картине.

То есть в некоторых случаях для принятия решения такой картины недостаточно. Требуется что-то ещё. С другой стороны, если бы я показал картину Шишкина, наверное, было бы всё гораздо понятней.

[00.05.35]

Свойства информации

Теперь подумаем, какими свойствами обладает информация в принципе. Я не буду зачитывать всё, что есть на слайде. Если потребуется, поставьте видео на паузу и прочитайте написанное самостоятельно. В любом случае в дальнейших слайдах я ещё затрону то, о чём здесь рассказывается.

Теперь оценим ситуацию, где есть стакан с какой-то жидкостью. Порассуждаем, как будут воспринимать полученную информацию люди с разным типом характера. Например, если этот стакан возьмёт оптимист, он скажет: «Вау, у нас теперь есть стакан с водой!». Пессимист ответит: «Это всего лишь полстакана воды. Как вообще с этим можно жить?». Реалист скажет: «Ну, полстакана воды — это, конечно, хорошо, но могло быть и больше». А скептик возразит: «Слушайте, а вода ли это вообще?». Таким образом, в зависимости от того, через какую призму вы смотрите на информацию, такие выводы и будете делать.

[00.06.40]

Численная мера информации

Теперь подумаем, как оценить ту или иную ситуацию, с точки зрения информатики, так как мы работаем с информацией на компьютере.

Допустим, у нас есть лампочка. Лампочка может быть включена или выключена.

Сколько информации мы получили и как её оценить?

В этом случае информатики придумали одну интересную величину. Они договорились, что выключенная лампочка будет обозначаться нулём, а включённая — единицей. В итоге **величину, которая принимает только два значения, информатики назвали битом**.

Хватит ли нам одного бита, чтобы хранить абсолютно всю и любую информацию?

Добавим второй бит и договоримся, что выключенная лампочка хранится, как 00, а включённая — как 11. Таким образом, можно добавить ещё какие-нибудь цвета и сказать, что 00 — выключено, 01 — красный, 10 — синий, а 11 — жёлтый. В таком случае мы уже передаём или получаем эту информацию в количестве 2 бит.

Сколько бит нужно, чтобы хранить разный объём информации?

Это тоже легко посчитать. Если мы представляем 1 бит, как 0 и 1, то чтобы представить 2 бита, надо взять 1 бит, продублировать эту информацию и добавить 0 и 1 для каждой пары. Так, в двух битах информации мы можем хранить 4 разных состояния. Аналогичным образом будут производиться действия и для 3 бит. Мы возьмём, по сути, то, что предназначалось для двух, увеличим это в 2 раза и снова добавим нули и единицы. То же самое делаем для 4 бит и более.

Таким образом, 1 биту соответствуют 2 разных состояния, 2 — 4 состояния; 3 — 8, и так далее. Самые внимательные уже уловили связь со степенями двойки. Так и есть. Если взять какое-то число бит и добавить степень двойки, то есть 2 во 2, 2 в 3, 2 в 4, получится то количество информации или способов хранения каких-то данных, которое можно использовать.

Теперь попытаемся оценить, **сколько будет занимать бит средний трек длительностью 3,5 минуты.** Например, стандартный, который вы слушаете время от времени.

Если посчитать свойства этого трека, окажется, что там будет примерно 72142028 бит. Разберёмся, много это или мало. Для этого посмотрим слайд, на котором показано около 1 300 символов нулей и единиц. Это 16-й шрифт. Уменьшим шрифт до 5, представим ноликами и единичками весь слайд и получим около 13 000 символов. Таким образом, условный трек длительностью 3,5 минуты будет занимать примерно 6 000 слайдов. Теперь представьте, сколько будет весить условный фильм.

Всегда ли используются только биты для определения объёма информации?

Для определения объёма информации не всегда используются только биты. В большинстве случаев информация разбивается на условные блоки, равные, как правило, 8 битам. Если возьмём 8 бит, то получим один байт. Таким же образом мы можем сгруппировать байты в количестве 1024 штуки и получить один килобайт. А, сгруппировав 1024 килобайт, получим один мегабайт. Далее идёт гигабайт, терабайт и другие величины, которые в основном используют те, кто, администрирует сервера. Например, в петабайтах объём информации измеряется легко.

[00.10.57]

Задача 1. Как оценить количество полученной информации?

Теперь решим задачу 8 класса. Допустим, вам захотелось стать разработчиком. У товарища, уже проходящего какой-то онлайн-курс, вы спрашиваете, где он его покупал. Друг даёт номер

телефона, по которому вы начинаете звонить и спрашивать, можно ли приобрести этот курс. После того как вы изъявили желание, вас спрашивают: «Хотите ли купить курс „*Sharp*-разработчик“?». И вы отвечаете: «Нет». Дальше вас спрашивают: «Хотите ли вы купить курс „Python-разработчик“?». Вы снова говорите: «Нет». Таким образом, выяснилось, что у компании, где вы планировали приобрести курс, есть 27 различных курсов. 26 раз вам сказали неверно, предложив неподходящие курсы, а 27 оказался именно тем, который хотели приобрести.

Сколько бит информации вы получили?

Такая задача для людей непросвещённых может показаться сложной. Но в большинстве случаев в информатике, как и в математике, тоже встречается много лишних слов. Главное — отбросить лишнее.

[00.12.52]

Решение

Начнём отбрасывать. Предыстория про знакомого, который что-то кому-то посоветовал, неинтересна. Нас интересует то, что вы 26 раз сказали слово «нет», а в 27 раз — «да». Это и есть важная информация. Значит, по факту вам надо вспомнить слайд со степенями двойки.

Открываете его и смотрите: 4 бита позволяют хранить 16 различных состояний, а 5 бит — 32 различных состояния. Число 27, а именно столько профессий предлагалось, лежит между этими числами. Как следствие, надо решить, сколько же бит вы получили — 4 или 5. Для этого поставьте видео на паузу и ответьте на этот вопрос. Очевидно, что число 27 не влезает в 4 бита, поэтому правильный ответ — 5 бит.

Будьте внимательны, когда вам задают такие задачи. Надо быть уверенным в том, что вы правильно поняли вопрос. Сейчас я сформулировал вопрос так: «Сколько бит требуется для указания номера профессии, которую вы хотели приобрести?». Ответ прост — 5 бит. А если бы вопрос стоял так: «Сколько бит информации вы получили бы?», — потребовались уточнения.

[00.13.44]

Информация и компьютер

Каким образом эти нолики и единички хранятся в компьютере?

Большинство из вас, скорее всего, знает, что у компьютера есть память. А кто-то даже слышал о временной и постоянной памяти. Но нас это не сильно интересует, потому что принцип хранения данных там примерно одинаковый.

Допустим, вы захотели изучить этот вопрос. Сначала заходите на какой-то ресурс и узнаёте, что вся информация хранится в ноликах и единичках. Следующим этапом у вас возникает желание узнать, как этот процесс происходит, и вы начинаете вникать в суть.

[00.14.32]

Надо ли программисту знать физику

И сейчас я задам такой интересный вопрос, на который, скорее всего, нет правильного ответа: «Надо ли программистам знать математику, а точнее, физику?». Для этого рассмотрим одну картинку.

Если мы увеличим картинку с чипом, позволяющим хранить информацию, то в глубине увидим простой физический элемент, называемый транзистором. Не пугайтесь, я не буду рассказывать, как он работает, с точки зрения физики. Однако идея заключается в том, что на самом деле весь ваш компьютер — набор каких-то транзисторов. Если захочется детальнее узнать об этой теме, воспользуйтесь QR-кодом со ссылкой на Хабр.

Таким образом, если, отвечая на предыдущий вопрос, вы сказали: «Нет», — я, наверное, отвечу, что это верно. Потому что на вопрос: «Знаешь ли ты, как действительно происходят процессы на таком глубинном уровне?», — точно отвечу: «Нет». Но, с другой стороны, если к вам подойдёт пятиклассник и скажет: «Как эти нолики и единички хранятся?», — а вы не сможете ответить, будет грустно. В таком случае можно ответить «Да». От себя же отмечу, что мне, как программисту, это не потребовалось.

[00.15.59]

Типы информации

Тип информации: текст

Посмотрим, какие вообще бывают типы информации. Самый, наверное, простой, с которым вы точно работали — это текст.

Каким образом текст хранится на нашем компьютере?

Чтобы ответить на этот вопрос, я расскажу о введении в таблицы кодировок и о том, как они получаются.

Возьмём обычную табличку, например, русский алфавит, и просто пронумеруем буквы. Я начинаю нумерацию с нуля, потому что в большинстве случаев всё на компьютере нумеруется с нуля. А для вас, как для будущих программистов, это важно. То есть ставим в соответствии с буквой А ноль, затем — один, и так далее.

После этого надо дать название получившейся таблице. Пусть она будет называться GeekTable. Позже мы рассмотрим и другие названия. Теперь у нас есть таблица кодировок.

[00.16.58]

Задача 2. Самостоятельная работа

А сейчас подумаем, каким образом компьютер будет хранить условное предложение: «Привет, как дела?». Для этого каждой букве надо поставить соответствующую цифру. Сделайте это **самостоятельно**, потратив на задание 3–5 минут.

[00.22.17]

Решение

Технически достаточно переписать нужные символы и коды, которые соответствуют этим символам. Но есть интересный момент. В сообщении вы увидели, например, запятую, пробел и вопросительный знак. Но их нет в нашей табличке. Как следствие, надо было добавить эти символы. Я добавил только запятую и вопросительный знак, но пробел тоже полноценный символ, который должен быть здесь. После этого надо переписать числовые коды каждой буквы. В итоге получается последовательность, как на слайде. Самые внимательные заметят, что последний код не 34, а 35. Это тот самый недостающий пробел. Просто в таблицу он не вместился, и его пришлось добавить рядом.

Как мы помним, компьютер хранит исключительно нолики и единички. Соответственно, он не может содержать число 35 в привычном для нас формате. Компьютер будет хранить его в нулях и единицах, поэтому нам потребуется перевести число из десятичной системы представления в двоичную, что и представлено на текущем слайде.

О том, как всё это перевести в подходящий формат, поговорим позже. А пока посмотрите: мы поставили соответствие нолику тот же нолик, единичке — единичку, а двойке уже соответствует 10. Будьте внимательны, когда вы говорите не о десятичной системе счисления, то числа начинают называться по-другому. То есть не 10, а 1 и 0. Не 101, а 1, 0, 1 и так далее. И обязательно добавляйте, в какой системе счисления работаете. В нашем случае это двоичная система.

[00.24.18]

Сжатие информации

Компьютер, как бы странным это ни казалось, считается тупой машиной. И когда вы уже знаете, что десятичные цифры представляются в двоичном виде, и даже переписали всё в должный вид, снова появляется проблема. Эта проблема связана с тем, что компьютер не может хранить один символ, например, пятью битами, второй символ — одним битом, а третий — пятью битами, и так далее. Поэтому он будет выбирать всегда определённое количество.

Как я уже говорил на раннем слайде, порция будет заключаться в 8 битах. Поэтому нам потребуется добавить лишние нули, чтобы всё было примерно поровну. Стоит отметить, что **на слайде я представляю информацию только порциями по 6 бит, но в общем случае, говоря о реальных машинах, там будет 8 бит.**

Это отсылка к сжатию информации. Так работают архиваторы — просто отбрасывают ненужное и использую только то, что надо.

[00.25.30]

Распространённые таблицы кодировок

В большинстве случаев каждый раз придумывать такие таблицы кодировок не требуется, потому что всё уже давным-давно придумано за нас. Самая известная классическая таблица кодировок — **ACSII**. Это те самые 120, в первом представлении — 128, символов, латиница и ещё какие-то системные символы. На слайде они есть.

Первые символы, представленные в таблице на слайде — NUL и другие непонятные обозначения. Это всего-навсего некоторые системные символы. Например, NUL — это так

называемый пустой символ. Есть также символ BACKSPACE (BS), отвечающий за удаление и стоящий в таблице кодировок восьмым. Следующими идут звуковые символы, которые хранятся в этой таблице кодировок со времён терминалов.

Таблицу кодировок ACSII расширяли, дополняя региональными символами. В таком случае, если мы говорим о русском алфавите, таблицу дополняли кириллицей. В итоге получилось не 128 символов, а 256, так как некоторые из них могли пригодиться.

Самая простая таблица кодировок — ACSII, но есть и другие. С некоторыми вы, скорее всего, сталкивались — это **UTF-8**, **UTF-16** и ещё несколько десятков. Чтобы посмотреть остальные, откройте программу Sublime Text. Знать все таблицы необязательно, но иметь представление, что такое таблица кодировок, надо. Старые программисты в начале своей карьеры могли столкнуться с тем, что в браузере почему-то открываются какие-то непонятные символы. Это как раз таки было связано с тем, что пользователь применял не ту таблицу кодировок, которой пользовался программист.

Итак, мы познакомились с тем, как хранится текстовая информация на компьютере.

[00.27.27]

Тип информации: числа

Теперь попробуем разобраться, как хранится числовая информация. Забегая вперёд, скажу, что это немного сложнее, чем может показаться. На слайде я привёл так называемый семисегментный индикатор. В зависимости от того, какой из сегментов загорится, мы увидим ту или иную цифру. Это может быть нолик, единичка и так далее.

Если возьмём, например, один такой элемент, у нас, естественно, в него могут попасть цифры 01234 и так далее, до 9. То есть 10 различных комбинаций. А если возьмём 2 таких сегмента, получится 100 различных комбинаций. И, соответственно, если добавить ещё один сегмент, выйдет уже 1 000, и так далее.

Теперь обратимся к классической математике. Возьмём, например, число 997, увеличим его на 1 и получим число 998. Следующий элемент будет равен 999, 1 000 и т. д. Ничего необычного. Но у компьютера очень жёсткий регламент по памяти. То есть он не может просто взять и записать куда-то новую порцию данных. Ему об этом надо сказать. Это небольшая отсылка к такому понятию, как **размер типов данных**, о чём мы ещё поговорим.

На следующем этапе появляется 4 бита, и добавилась 1 000. Это хорошо. Далее может получиться 9 997, а потом 9 998. И когда нам потребуется 10 000, а компьютер не знает, откуда брать дополнительную память, получится число 0 (ноль). Таким образом, мы знакомимся с

понятием **«переполнение типа»**. То есть нам нужен какой-то дополнительный сегмент для отображения данных, а у нас его нет. По факту должно было быть 10 000, но сегмента нет, единичка пропала, и получился 0. С такой ошибкой часто встречаются начинающие программисты, и это вызывает большие проблемы. Так что просто вспоминайте об этом, когда компьютер выдаёт какой-нибудь StackOverflowException или что-то подобное, например, OverflowException.

[00.29.33]

Задача 3. Сложение чисел

Теперь посмотрим, как подобные комбинации проворачивает компьютер в двоичной системе. Для этого обратимся к привычной десятичной системе. Если мы захотим сложить число 9 с 28 и с 90, то воспользуемся **ассоциативным законом**, или **сочетательным законом**, который наверняка помнят любители математики.

[00.29.58]

Решение

Итак, чтобы узнать сумму 9 + 28, складываем всё в столбик. 9 + 8 — это 17. Пишем 7, а 1 держим в уме. Наверняка вы делали это в школе. Далее складываем 2 и 1 и получаем число 3. Записываем ответ, равный 37.

К текущему результату (к 37) добавим 90. Сложив 0 и 7, получим 7. А к 9 прибавив 3, выйдет 12. Впереди записываем 2, к 0 прибавляем 1, тот самый недостающий разряд, и получаем число 127. В классической математике никаких проблем нет. Однако у компьютера всё не так просто.

В частности, если речь идёт о двоичной математике, то работают немного другие законы. А именно, когда вы складываете 9 и 1, получаете обнуление одного разряда и увеличение следующего. В двоичной системе существует только два числа — 0 и 1. Как следствие, сумма 0 + 1 даст 1. Сложив 1 и 0, тоже получится 1. Сумма 0 + 0 выдаст 0. А 1 + 1 обнулит один разряд и увеличит следующий. Поэтому, сложив 1 + 1, получится 10 в двоичной математике или в двоичной системе счисления.

Следующим этапом сложим, например, 1101 и 111. Таким образом, 1 + 1 по нашим законам — это 10. Пишем 0, а 1 держим в уме. На следующем этапе складываем 0 и 1 и получаем 1. 1 в уме плюс 1 — опять 10. 0 пишем, а 1 уходит в следующий разряд. Далее 1 + 1 — это сумма 10 + 1, дающая 11. 1 пишем в текущий разряд, а оставшийся 1 держим в уме. На следующем этапе снова складываем 1 + 1 и получаем 10. Записываем 0, убрав 1 в дополнительный разряд.

В идеальном случае числа будут складываться. Можете открыть, например, классический калькулятор Windows, mac или какой-либо ещё, поддерживающий сложение чисел в двоичной системе счисления, и получить тот же результат.

Проверим это. Число 1101 в десятичном представлении будет выглядеть как 13. Соответственно, 7 — это 111. Если сложить эти числа, получится 20. Таким образом, независимо от того, в какой системе счисления вы работаете, результат всё равно окажется одинаковым.

Любопытно, что изначально тот тип данных, который вы использовали, может включать только 4 бита. В этом случае, как я уже говорил ранее, получается **переполнение**. То есть, когда вам надо было записать 1 в новый разряд (добавить новый разряд), получилось бы 010. По всем законам математики нули, идущие в начале, то есть в так называемых позиционных системах счисления, не играют особой роли. Поэтому от 0 можно просто избавиться. Так вы получите 100, что и будет ошибкой переполнения. То есть сумма 13 + 7 даст 4. И это не парадокс, а законы математики в компьютере.

[00.33.43]

Задача 4. Умножение чисел

Далее посмотрим, как работает умножение в двоичной математике. В общем, это выглядит так же, как и в 5 классе, когда мы учились перемножать обычные десятичные числа столбиком.

[00.33.57]

Решение

Итак, чтобы умножить 101 на число 11, снова потребуются законы двоичной математики, где 0 x 0 = 0. Если 0 умножить на 1, будет 0. А если 1 умножить на 0, тоже получится 0. И 1 x 1 — это 1, что неудивительно.

Далее возьмём число 101, умножим на 1 и получим как раз таки 101. На следующем этапе 101 умножим на вторую единицу и запишем таким же образом, как это делалось для десятичных чисел.

На следующем этапе сложим два двоичных числа и получим результат. Собственно, это ответ. 101 умножить на 11 есть не что иное, как 1111. Можете опять же запустить калькулятор и проверить.

[00.34.48]

Задача 5. Самостоятельная работа

Число 25 надо умножить на 2. Возможно, вы увидите какие-то интересные закономерности. Умножьте на 2 следующее число, а потом ещё раз, чтобы проследить некоторую закономерность.

[00.39.09]

Решение

Когда число 25 мы перевели в двоичное представление, то увидели 11001. Далее умножили на 10 и получили число 50 в двоичном представлении. Те, кто стал умножать 50 на 2, получили 1100100. А тот, кто умножил 100 на 2, получил 11001000.

Теперь обратите внимание на то, как 25 трансформировалось в число 200. Примерно то же самое вы можете наблюдать, когда умножаете на основание системы счисления в десятичных числах. То есть умножение 20 на 10 просто добавляет 0 к текущему числу, которое мы умножали. На следующем этапе, если надо число, например, 250 умножить на 10, просто дописываем ноль. Таким образом, в двоичной математике умножение на 2 выглядит также: берём число и умножаем его на 2, просто дописав 0. В компьютерных науках это называется **смещением бит**. Есть смещение влево, то есть умножение числа, и, соответственно, смещение вправо — обратная операция умножения, деление.

[00.39.28]

Тип информации: числа беззнаковые целые

Итак, мы узнали, как хранятся целые положительные или, скорее, неотрицательные числа. Теперь посмотрим, как компьютер представляет отрицательное число.

Здесь всё не так просто, как может показаться. Компьютер не может просто хранить минус, для этого требуется отдельная инструкция.

Ранее мы хранили 1 000 значений в 4 битах, пока поработаем в десятичном представлении, но, чтобы добавить знак (отрицательные числа), самый крайний, крайний левый, или старший разряд будет хранить как раз таки знак числа. Если там стоит 0, договариваемся, что это положительное число. А если стоит 1, значит, отрицательное.

Таким образом, если у нас есть число 0000, и самый старший разряд будет хранить знак, как я уже сказал: 0 — положительный знак, а 1 — отрицательный, то в положительной части числа будут храниться так:

- 0000 — 0;
- 0001 — 1;
- 0010 — 2;
- 0011 — 3.

А если мы добавим к ним отрицательные числа, то получится следующее:

- 1010 — -3;
- 1001 — -2;
- 1000 — -1.

Здесь, кстати, мы также нарываемся на проблему переполнения типа. Только теперь уже не для 4, а для 3 бит. Подумайте почему.

Ещё одно важное замечание касается так называемых вещественных чисел. Например, чтобы хранить 2,5 или 2,124, используется более сложная форма. Возможно, мы поговорим об этом на семинарских занятиях, но сейчас опустим это момент.

[00.41.18]

Зачем нужны системы счисления и как между ними путешествовать

Ранее я говорил о таблице кодировок, о таких числах, как 01234, 30, 35 и так далее. Далее из этих десятичных представлений мы каким-то образом получили двоичное. Разберёмся, как это получилось.

[00.41.39]

Задача 6. Перевод числа в двоичное представление

Допустим, у нас есть число 13 в десятичном представлении. Подумаем, как представить его в двоичном представлении. Для этого вспомним знания, полученные в 6 классе на уроках математики, где мы делили уголком.

[00.41.51]

Решение

Итак, мы будем делить 13 на основание системы счисления, то есть на 2. Чтобы 13 разделить на 2, возьмём по 6. На следующем этапе 6 умножим на 2 и получим 12. Соответственно, 13 - 12 = 1.

Большая часть из вас сейчас скажет, что здесь надо добавить запятую, а потом нарисовать 0. **Но мы делим только в целых числах**. Если вдруг подзабыли этот момент, то вспомните. Нас интересует так называемое неполное частное, остаток. Погуглите, если вам это интересно.

Следующим этапом просто запоминаем единицу, которую мы получили в остатке. Она войдёт в двоичное представление нашего числа. Далее берём 6 и снова делим на основание системы счисления, то есть на 2. Деля 6 на 2, берём по 3. 3 умножаем на 2 и получаем 6. А разница 6 - 6 даёт 0. 0 — это остаток. Остаток должен быть всегда меньше той системы счисления, в которую переводится число.

Снова запоминаем этот нолик. На следующем этапе 3 делим на 2 и берём по 1. 3 - 2 даёт 1. Единицу запоминаем. Таким образом, мы будем продолжать, независимо от того, насколько бы большое число ни использовалось.

В нашем случае операция деления заканчивается на 4 шаге. Мы поняли это по нулю. Технически вы можете продолжить эту операцию: 0, поделённый на 2, даст ноль, и так далее. Но это будут так называемые незначащие нули. Дело в том, что, когда вы захотите превратить десятичное представление числа в двоичное, понадобится переписать числа в обратном порядке. Единицу, которую мы получили первой, запишем как последнее число двоичного представления. Ноль встанет предпоследним. Следующая единица будет третьей и четвёртой в нашем случае. Нолики, как я сказал ранее, ни о чём не говорят, поэтому просто опустим их.

Теперь понятно, как наша фраза «Привет, как дела?» превратилась в набор ноликов и единичек. Нолики, стоящие в начале, говорят нам, что компьютер хранит информацию порциями, а одна такая порция равняется 8 битам или одному байту.

[00.44.31]

Зачем нужны разные системы счисления?

Технически вы уже поняли, зачем нужна двоичная система счисления. Но, помимо этого, есть ещё и троичные, четверичные, шестнадцатеричные и другие системы, можно придумать любую. О том, зачем они нужны, поговорим на семинарах.

[00.44.58]

Вывод

Теперь мы знаем:

- как хранятся числа, причём как числа положительные, то есть беззнаковые, так и отрицательные, знаковые;
- как хранится текст.

Внимание! *Так как в рамках таблицы кодировок мы можем представить цифру, это не значит, что всё будет работать так же, как в примере двоичной математики.*

[00.45.15]

Тип информации: графическая информация

Теперь узнаем, как хранится графическая информация на компьютере. Возьмём прямоугольную картинку, пронумеруем столбики и строчки, а также договоримся: чёрный цвет закрашиваем единичкой, а белый — ноликом.

Представим картинку, где единичка будет соответствовать закрашенному элементу, а нолик — не закрашенному. Мы получим так называемое монохромное изображение, состоящее только из двух цветов. Но вы скажете, что картинки состоят из 16 миллионов и нескольких тысяч символов. Действительно, если договориться хранить информацию о цвете ни в одном бите, а в большем количестве бит, например, в формате, представленном на слайде, то 0 будет отвечать за белый цвет, 01 — за синий цвет, а 10 — за красный. В этом случае мы получим трёхцветное изображение. Несложно догадаться, что здесь остаётся резерв под цвет 11.

Теперь договоримся, что 8 бит или 1 байт будет хранить информацию о количестве красного цвета, 1 бит отведём под зелёный цвет и 1 бит — под синий. Наверняка вы уже услышали такие слова, как **Red-Green-Blue,** или просто **RGB**. Идея заключается в том, что, смешивая разное количество цветов, каждый раз получается какой-то новый оттенок. Это и есть современное представление цвета, когда 1 пиксель кодируется 3 байтами.

В качестве примера возьмём цифровую палитру классического Paint, который есть в операционной системе Windows. Здесь, кроме красного, зелёного и синего цветов (114, 63, 243), есть и шестнадцатеричное представление. Так удобнее хранить информацию. То есть технически, если бы мы использовали классическое десятичное представление, то в самом худшем представлении получилось 9 цифр. В шестнадцатеричном представлении таких цифр будет только 6.

Перевод из десятичной системы в шестнадцатеричную не отличается от перевода в двоичную систему. Только вместо деления на 2 придётся делить на 16.

Таким образом, фраза «Привет, как дела?» в двоичной, десятичной и шестнадцатеричной системах будет выглядеть по-разному. Например, в шестнадцатеричном формате вначале добавляется 0x, потому что есть такие стандарты, а также ещё какое-то количество нулей. С такой информацией вы могли встречаться, когда вылетал синий экран. Олды, скорее всего, видели такой формат на Windows XP. На Windows 10 также встречается шестнадцатеричный формат. В других операционных системах на нижних строчках каких-то сайтов иногда демонстрируется тот самый код ошибки, изучив который можно получить дополнительную информацию.

[00.48.40]

Заключение

На этом всё. Это самое простое введение в школьный курс информатики. На семинарах нас ждут задачи на вычисление количества информации, переводы из различных систем счисления и ещё много интересного. Так что обязательно приходите.