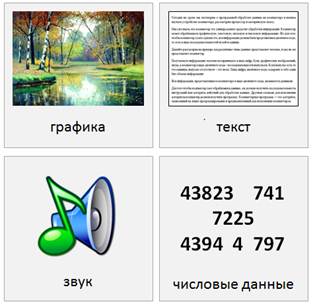
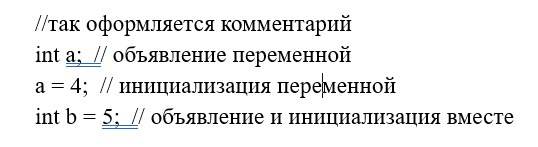
**«Что такое переменная?»**

Любая компьютерная программа производит какие-то математические или логические операции и отображает результат своей работы на экране. Многие программы используют графические и звуковые эффекты. Современные игры почти все работают с объемными 3D-эффектами. Все это, вычисления, эффекты, графика - результат вычислений, который появляется благодаря размещению большого объема чисел в ячейках памяти компьютера и работе с ними. Задача программиста — написать программу, а это значит заполнить память компьютера нужными данными, произвести с ними необходимые действия, получить результат и вывести его пользователю в нужном виде. Соответственно, для того чтобы написать полезную программу, нужно знать, как организовать ячейки памяти для хранения данных, как работать с этими данными и, в конечном итоге, вывести эти данные на экран.



Для того, чтобы записать что-то в память компьютера нужно создать так называемую переменную. Переменная — это «ячейка» оперативной памяти компьютера, в которой может храниться какая-либо информация. И, конечно, логично, что если мы хотим поработать с какой-то ячейкой памяти, то нужно дать ей имя.

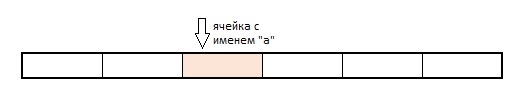
Самой простой и независимой частью программы на любом языке программирования является инструкция или стейтмент. Это такой аналог предложения в русском языке. Фраза, которая несет в себе смысловую нагрузку. И как в русском языке предложение завершается точкой, в программировании стейтмент обязательно завершается точкой с запятой.



Простейшая необходимая инструкция в программе — это объявление и инициализация переменной.

Помимо стейтментов в языке программирования существуют комментарии, которые не несут в себе никакой смысловой нагрузки для компьютера, но очень помогают программисту ориентироваться в коде программы. Комментарии пишутся после двух знаков слеша «/».

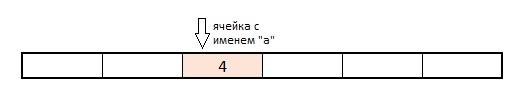
Первая инструкция — это объявление переменной. Объявить переменную – это значит сказать компьютеру, чтобы он выделил ячейку памяти и дал этой ячейке тип и имя, которое мы указали, в данном случае наша переменная имеет тип int, а имя - «a»



Что такое тип переменной. Мы привыкли, что в математике есть целые числа и дробные. В русском языке мы пользуемся буквами и составляем слова из них. Мы можем рассуждать логическими понятиями – «правда» или «ложь», «да» или «нет». Поэтому компьютеру мы тоже хотим дать такую возможность. Для этого и созданы различные типы, про которые мы подробно поговорим на следующих занятиях, а пока попробуем воспользоваться одним из таких типов данных – int, который означает целые числа.

Так вот с помощью строки объявления переменной мы сказали компьютеру, что хотим завести в его памяти ячейку, у которой будет имя «a». Какую конкретно ячейку выделит для нас компьютер и где она будет расположена в его памяти, нам не важно. Важно, что, компьютер будет знать, что в его памяти есть ячейка с именем «a» и когда мы захотим с ней поработать, он будет работать именно с этой конкретной ячейкой.

Второй стейтмент на слайде — это инициализация переменной. Инициализировать переменную – это значит положить в ячейку с ее именем какое-то значение.



Таким образом мы уже используем нашу ячейку, которую завели в предыдущем стейтменте. Компьютер знает, что у него в памяти есть ячейка с именем «a», поэтому он обращается именно к ней и помещает в нее значение «4».

Нужно запомнить, что переменная связывает между собой три понятия – это тип, имя и значение. Без связки этих трех свойств невозможно производить какие-то действия с переменными и памятью компьютера.

То есть первое, что вы должны сделать – это сказать компьютеру, что вам нужна ячейка, в которой вы будете хранить данные определенного типа, в нашем случае целое число, и дадите этой ячейке имя. А затем эту ячейку нужно заполнить какими-то данными.

В языке программирования Java есть правила, которые определяют какие имена переменным можно использовать, а какие нельзя.

1 Идентификатор не может быть ключевым словом. В языке С++ есть список слов, которые зарезервированы, то есть мы не можем использовать их для своих целей, поскольку каждое из этих ключевых слов уже определяет какую-то инструкцию.

Список ключевых слов представлен на следующем слайде:

2 Идентификатор может состоять только из букв (нижнего или верхнего регистра), цифр или символов подчёркивания. Это означает, что все другие символы и пробелы — запрещены.

3 Идентификатор должен начинаться с буквы (нижнего или верхнего регистра). Он не может начинаться с цифры или с другого какого-то символа.

6 И самое важное правило из всех - используйте в качестве идентификаторов только те имена, которые реально описывают то, чем является объект. Особенно это важно для больших программ. Но привыкать нужно сразу.

Если, например, вам нужно в программе посчитать количество яблок, то назовите переменную appleCount и вы всегда сможете по имени переменной понять, что хранится в ячейке с этим именем.

**Данные в памяти**

Не только обычному пользователю, но и современному программисту редко приходится обращаться к внутреннему представлению информации в компьютере. Конечно, для того и разработаны языки программирования высокого уровня, такие как Java или С++, чтобы не задумываться о внутреннем устройстве компьютерной памяти и вычислений. Но хороший программист должен обязательно понимать, как работает компьютер, как он хранит данные, работает с ними, чтобы избежать ошибок, которые могут возникнуть в процессе вычислений в его программе. Например, мы привыкли, что в математике сравнение чисел происходит с помощью знаков «>» и «<», причем неважно какие это числа - целые или дробные. А знаете ли вы, что в программировании нельзя сравнивать два числа вещественного типа с помощью обычных арифметических знаков сравнения. Почему? Попробуем разобраться…

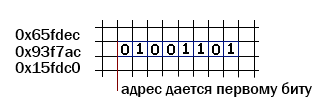


Слайд №1. Понятие бита и хранение данных в памяти компьютера

Компьютерная память состоит из огромного числа маленьких элементов. Каждый такой маленький элемент похож на выключатель, который может быть либо включен, либо выключен. Когда выключатель включен и поддерживает связь в цепочке, ему приписывают значение равное 1, а когда выключатель выключен и связь в цепочке разрывается, ему приписывается значение 0. Каждый такой элемент называют битом. Бит, как вы уже поняли, может принимать значение — 0 или 1. Именно поэтому компьютер использует двоичную систему счисления для хранения данных. Как это ни странно, но в современных компьютерах, биты не используются по отдельности, а объединяются по группам — байтам.

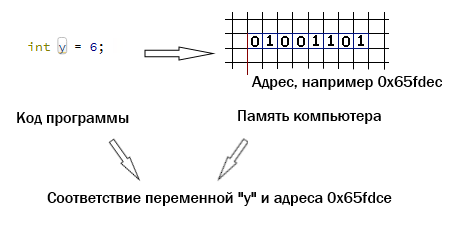
В русском языке отдельные буквы не имеют никакого смысла, но мы из букв составляем слова, а вот слово уже несет в себе какую-то смысловую нагрузку. Также и в компьютере, отдельные биты не используются сами по себе. Для удобства использования, биты собраны в кучки по 8 штук. Эта кучка из 8 битов называется байт. Вот и получается, что бит — это буква, а байт — это уже слово.

А теперь представим себе книгу… Каждое слово в книге мы можем пронумеровать и тогда будем точно знать под каким номером какое слово находится. Также и в компьютере — каждый байт имеет свою позицию, позиция байта в памяти компьютера называется адрес. Адреса позволяют найти и получить доступ к байту, который находится в определённом месте в памяти.



Слайд №2. Адреса в памяти компьютера

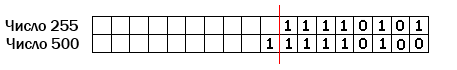
Мы не будем вдаваться в подробный разбор формирования адреса, остановимся на основных аспектах. На слайде представлены различные адреса. Важно то, что компьютер сам занимается распределением адресов для переменных, программист на это повлиять никак не может. Причем каждый раз запуская заново свою программу, вы увидите, что у одной и той же переменной может быть разный адрес. В этом состоит отличие от приведенной нами аналогии размещения слов в книге. Если в книге порядок слов зафиксирован раз и навсегда, то размещение значений переменных непостоянно и зависит только от компьютера.



Слайд №3. Связь переменной и адреса

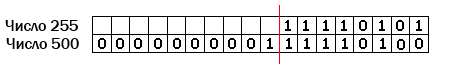
Но как вы уже поняли из предыдущих занятий, программист не работает напрямую с адресами, а использует переменные. Когда программист определяет переменную, например по имени «y», в коде программы, компьютер выделяет память для хранения значения этой переменной и создает связь «имя переменной» – «адрес в памяти». Всякий раз, когда компьютер встречает переменную «y» в коде программы, он понимает, что для того, чтобы получить значение этой переменной «y», ему нужно заглянуть в ячейку памяти c адресом «0x65fdce». Программисту не нужно беспокоиться о том, какой конкретно адрес памяти был выделен для его переменной «y». Он просто использует идентификатор переменной «y», а компьютер конвертирует это имя в соответствующий адрес памяти. Таким образом создается связь: «имя переменной» – «адрес переменной» – «значение переменной». Получается, что компьютеру не нужны переменные, они нужны программистам, потому что работать с адресами достаточно сложно, а с именами переменных намного понятнее и удобнее.

Мы уже знакомились с двоичной системой счисления и способом перевода числа из десятичной системы в двоичную и обратно. Каждый раз, когда программист задает значение переменной, компьютер переводит это значение в двоичную систему счисления.



Слайд №5. Представление чисел

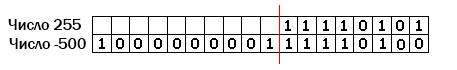
Согласно этому переводу число «255» в двоичной системе счисления имеет вид «11110101». Оно вполне вмещается в 8 бит одного байта. А число «500» в двоичной системе счисления имеет вид «1 11110100». И мы видим, что число «500» уже не помещается в один байт.



Слайд №6. Представление чисел

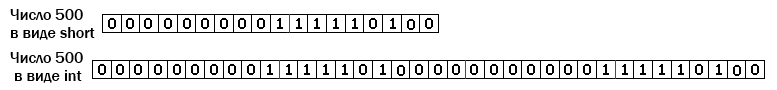
Поэтому для числа «500» компьютер выделит уже не один байт, а два и запишет его в таком виде «00000001 11110100».

А если нам нужно отрицательное число? Принято под знак числа отводить первый бит из задействованных байтов. Если число положительное, то есть знак «+», то бит будет равен 0. А если число отрицательное, то есть имеет знак «-», то бит будет равен 1. Значит число «-500» будет таким «10000001 11110100».



Слайд №7. Представление чисел

А если нам нужно записать очень большое число? Компьютер увеличит количество использованных байтов, например до 4 или 8. А как компьютер узнает, сколько байтов ему нужно использовать для записи числа? Эту информацию ему должен предоставить программист с помощью указания типа для переменной. Мы с вами знакомы уже с несколькими типами данных — это целочисленный тип данных, который называется int и вещественный тип данных, который называется double. Помимо того, что они отличаются тем, что хранят разные типы данных, они занимают разное количество памяти в компьютере. Тип int занимает 4 байта, а тип double 8 байт.

Слайд №8. Различие типов

Чем больше байт выделяется под используемый тип данных, тем большее число мы сможем сохранить в этих байтах. Если же мы попытаемся записать большее значение в переменную, чем позволяет ее тип, то получим неправильное значение.

Таким образом задача программиста правильно определять нужный тип данных для переменных и использовать именно его.

А зачем нужно столько типов данных? Посмотрим на наш слайд №8. Как вы думаете с какой из переменных компьютеру будет легче работать? Конечно, с переменной типа short. Для этого и существует несколько различных типов данных, чтобы там, где нужно облегчать работу компьютеру, а значит ускорять работу программы и использовать меньше компьютерной памяти.

**Типы Данных**

Одной из основных особенностей C# является то, что данный язык является строго типизированным. А это значит, что каждая переменная имеет определенный тип и данный тип строго определен специальными зарезервированными ключевыми словами C#. Тип данных определяет диапазон значений, которые может хранить переменная или константа.

Переменные не что иное, как зарезервированные места памяти для хранения значений. Поэтому при создании переменной Вы резервируете некоторое пространство в памяти, а ОС, основываясь на типе данных, который присвоен переменной, выделяет требуемое количество памяти и решает, что может быть сохранено в зарезервированную память.

Существует два типа данных в C#:

\* простые или примитивные типы данных;

\* ссылочные типы данных (ссылка/объект).

Примитивные типы данных

Есть восемь типов данных, поддерживаемых C#. Все размеры примитивных типов строго фиксированы и не зависят от машинной архитектуры. Основные типы данных предопределены языком и названы по ключевому слову.

Как записать буквы и символы? Самое простое – пронумеровать их.

Сначала зайдем на сайт https://unicode-table.com, где можно посмотреть значения всех символов UNICODE

Иногда возникают ситуации, когда у вас есть величина какого-то определенного типа, а вам нужно ее присвоить переменной другого типа. Для некоторых типов это можно проделать и без приведения типа, в таких случаях говорят об автоматическом (неявном) преобразовании типов. В других случаях программист должен сам провести преобразование типа к нужному — это называется явным преобразованием типов.

Автоматические (неявные) преобразования

При изучении типов данных уже говорилось о том, что целочисленные типы данных byte и short при вычислении Java всегда приводит к типу int, а тип float к типу double. Фактически это и есть неявное преобразование типов.

![](./Pictures/1.jpg)

Стрелками на рисунке показано, какие преобразования типов могут выполняться автоматически без потери точности. Пунктирными стрелками показаны автоматические преобразования с потерей точности.

Преобразования от меньшего типа к большему называются расширяющими и именно эти преобразования проходят без потери данных и без потери точности.

```java

//Пример кода

byte a = 40;

byte b = 50;

byte с = 100;

int d = a\* b / с;

```

Можно сказать, что существуют следующие правила повышения типов

Если в выражении используются переменные типов byte, short и int, то во избежание переполнения тип всего выражения автоматически повышается до int. Если же в выражении тип хотя бы одной переменной - long, то и тип всего выражения тоже повышается до long.

Если выражение содержит операнды типа float, то и тип всего выражения автоматически повышается до float. Если же хотя бы один из операндов имеет тип double, то тип всего выражения повышается до double. По умолчанию Java рассматривает все литералы с плавающей точкой, как имеющие тип double.

Преобразования от большего типа к меньшему называются сужающими и в этих случаях вся ответственность по переносу данных ложится на программиста.

```java

//Пример кода

int a = 2147483000;

int b = a\*2;

System.out.println(b);//-1296

```

### Явные преобразования

Все остальные преобразования типов, не рассмотренные ранее, осуществляются явным образом, а именно указанием нужного типа, заключенного в круглые скобки. Обычно это сужающие преобразования от типа с большей разрядностью к типу с меньшей разрядностью. При явном преобразовании программист должен внимательно оценить все риски потери данных и выбрать тип, который сведет эти риски к нулю.

```java

//Пример кода

byte b = 50;

b = b\* 2;

b = (byte) (b\* 2);

Ошибка: Error:(5, 14) java: incompatible types: possible lossy conversion from int to byte

```При преобразовании значений с плавающей точкой к целочисленным значениям, происходит усечение дробной части.

``java

//Пример кода

double a = 56.9898;

int b = (int)a;

System.out.println(b); // 56```