2\_5 адресация памяти и переменные

**1. Устройство памяти**

У каждого компьютера есть оперативная память. Что же это такое, какими свойствами обладает и, самое главное, какая нам от этого польза?

Каждая программа (в том числе и программы, написанные на Java) перед выполнением загружается в оперативную память. В оперативной памяти находится код программы (который исполняется процессором) и данные программы (которые в память помещает сама программа).

Что же такое оперативная память и на что она похожа?

Представьте себе Excel 😎 Страница в Exсel'е состоит из ячеек, и у каждой ячейки есть её уникальный номер (A1, A2, ... B1, B2). Зная номер ячейки, всегда можно записать в неё какое-то значение или же получить значение, которое там хранится. Память компьютера устроена очень похоже.

Программа и её данные во время работы хранятся в памяти. Вся память компьютера представлена в виде маленьких ячеек – байт. У каждой ячейки есть её уникальный номер – 0, 1, 2, 3, ...; (нумерация начинается с нуля). Зная номер ячейки, мы можем сохранить в эту ячейку какие-то данные. Или взять их из неё. В одних ячейках хранится код программы – набор команд для процессора, в других – данные этой программы. Номер ячейки также называют адресом ячейки.

Процессор умеет исполнять команды из загруженной в память программы. Почти все команды процессора — это что-то типа: взять данные из некоторых ячеек → сделать с ними что-то → результат поместить в другие ячейки

Объединяя сотни простых команд, мы получаем сложные и полезные команды.

Когда в коде программы объявляется переменная, ей выделяется кусочек ещё не использованной памяти. Обычно это несколько байт. При объявлении переменной обязательно нужно указать тип информации, которую программа будет хранить в ней: числа, текст, или другие данные. Ведь не зная тип информации, не ясно, какого размера блок памяти нужно выделить под переменную.

На заре компьютерной отрасли программы работали просто с номерами ячеек памяти, но потом для удобства программистов ячейкам стали давать имена. Уникальное имя переменной — это в первую очередь для удобства программистов: программа во время работы отлично справилась бы и с номерами.

**2. Переменные в памяти**

Всего в Java есть 4 типа данных для хранения целых чисел. Это byte, short, int и long.

|  |  |  |
| --- | --- | --- |
| **Тип** | **Размер, байт** | **Происхождение имени** |
| byte | 1 | Byte, т.к. занимает один байт памяти |
| short | 2 | Сокращение от Short Integer |
| int | 4 | Сокращение от Integer |
| long | 8 | Сокращение от Long Integer |

Также в Java есть 2 вещественных типа — float и double:

|  |  |  |
| --- | --- | --- |
| **Тип** | **Размер, байт** | **Происхождение имени** |
| float | 4 | Сокращение от Floating Point Number |
| double | 8 | Сокращение от Double Float |

Каждый раз, когда выполнение программы доходит до команды создания переменной, ей выделяется небольшая область памяти (размер зависит от типа переменной).

Адресом переменной считается адрес первой ячейки выделенного под нее блока памяти.

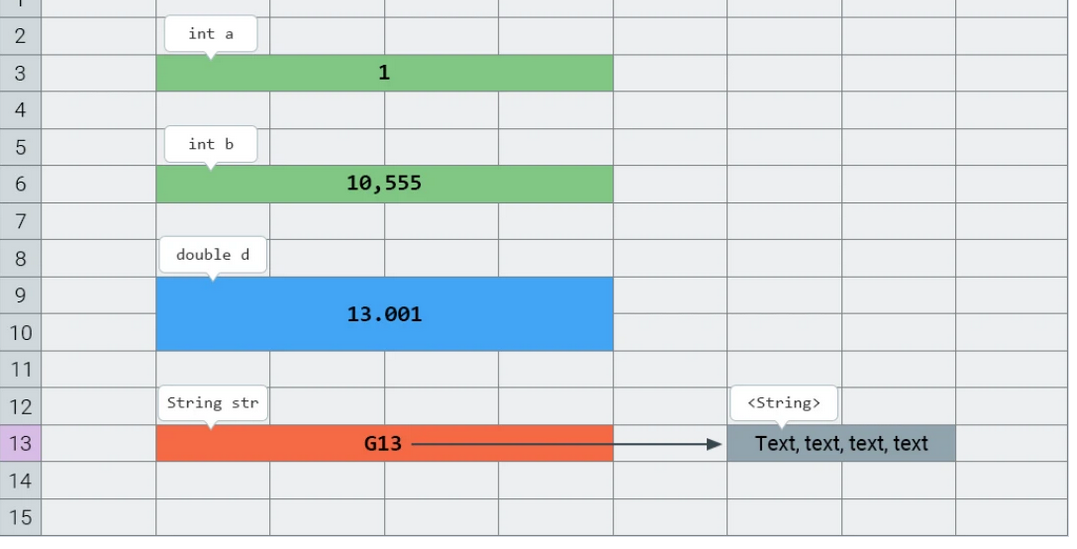
Java-программам запрещено напрямую обращаться к памяти. Вся работа с памятью происходит только через Java-машину.

НЕТ Указателей в Java

**3. Тип String в памяти**

Тип String может хранить большие объемы данных, поэтому это не просто тип данных, а полноценный класс.

Сами данные типа String (текст) помещаются в специальный объект, под который выделяется память, а уже адрес этого объекта помещается в переменную, под которую тоже выделяется память.



Переменная a типа int занимает 4 байта и хранит значение 1.

Переменная b типа int занимает 4 байта и хранит значение 10,555. Запятая - это не дробная часть числа, а разделение разрядов. Дробная часть отделяется точкой

Переменная d типа double занимает 8 байт и хранит значение 13.001.

Переменная str типа String занимает 4 байта и хранит значение G13 — адрес первой ячейки объекта, содержащего текст.

Объект типа String (содержащий текст) хранится отдельным блоком памяти. Адрес его первой ячейки хранится в переменной str.

**4. Почему в программировании всё нумеруют с нуля**

Люди очень часто удивляются, почему в программировании почти везде принято считать с нуля. Дело в том, что есть очень много ситуаций, когда считать с нуля удобнее (хотя есть и ситуации, когда удобнее считать с 1).

Самая простая из таких ситуаций — это адресация памяти. Если вашей переменной выделили 4 байта памяти и у вас есть X – адрес первого байта, то какие будут адреса у всех байтов? X+0, X+1, X+2, X+3. Вот мы уже и получили группу байтов с индексами 0, 1, 2, 3.

Когда мы думаем об относительном адресе внутри какого-либо блока данных, всегда получаем нумерацию с нуля. Это и есть первая и самая распространенная причина счета с нуля.