§2. ТИПЫ ДАННЫХ. КЛАСС

Цель занятия: рассмотреть многообразие примитивных типов языка Java, изучить хранение целых чисел в компьютере и базово познакомиться с понятием класса.

Тип данных

Переменные позволяют выделять именованные области данных в памяти для дальнейшего к ним обращения по названию. К сожалению, выделенная область памяти конечна и необходимо определить порядок хранения в ней значений. Переменная абстрагирует нас от работы с адресами и объединяет несколько ячеек памяти.

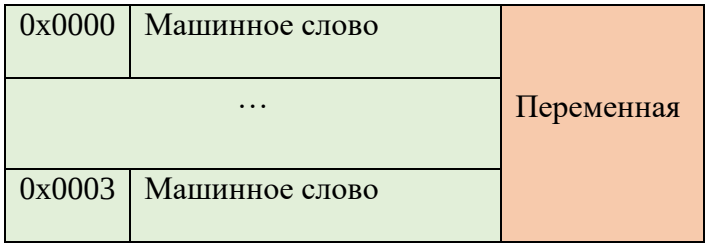


Рис. 1 Принципиальная схема хранения переменной

Ячейка памяти считается неделимой, т. е. она не может принадлежать нескольким переменным одновременно, поэтому минимальный размер переменной равен размеру **машинного слова** — единичной ячейки памяти.

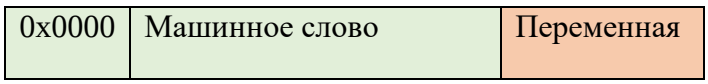


Рис. 2 Переменная минимального размера

Если абстрагироваться от ячеек памяти, то можно обобщить, что в рамках архитектуры фон Неймана **переменная** — это просто область двоичных данных.

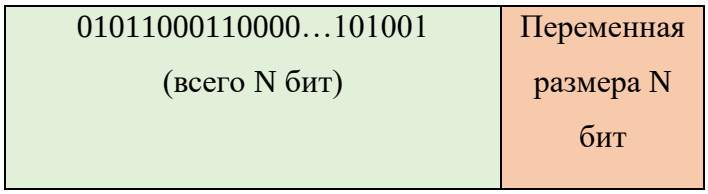


Рис. 3 Переменная как набор двоичных данных

Переменная обладает конечным **размером** и, следовательно, **множеством допустимых значений**.

Тип данных — это множество значений и операций над ними.

Чтобы задать тип данных, необходимо определить его область допустимых значений и операции над этим типом.

Перечисляемый тип (enum)

Простейший способ задать множество значений типа — **перечислить** их.

Давайте для примера рассмотрим тип “**вокзал Санкт-Петербурга**”, множество значений которого состоит из действующих пассажирских вокзалов Санкт-Петербурга: Финляндский, Московский, Витебский, Балтийский и Ладожский. Сопоставим каждому двоичному значению переменной соответствующий вокзал.

Двоичное значение	Числовое представление	Вокзал
000	0	Финляндский
001	1	Московский
010	2	Витебский
011	3	Балтийский
100	4	Ладожский
101	5	Не определен
110	6	
…		

Рис. 4 Переменная как набор двоичных данных

На языке Java перечисляемый тип называется **enum** (от англ. enumeration) и задается тем же способом: последовательным перечислением всех возможных значений.

```
enum RailwayTerminal {
    Finlyandskiy,
    Moskovskiy,
    Vitebskiy,
    Baltiyskiy,
    Ladoszhskiy
}
```

Код 1 (Demo.Enumeration). Перечисляемый тип enum на Java

Мы можем создать переменные перечисляемого типа **enum** подобно тому, как мы создавали числовые переменные **int** и **double**.

```
RailwayTerminal AllegroTerminal = RailwayTerminal.Finlyandskiy;
RailwayTerminal SapsanTerminal = RailwayTerminal.Moskovskiy;
System.out.println(AllegroTerminal);
System.out.println(SapsanTerminal);
```

```
Finlyandskiy

Moskovskiy
```

Код 2 (Demo.Enumeration). Объявление переменных enum на Java

Числовой тип

Назревает хоть весьма зрелый, но всё же школьный вопрос — можно ли **перечислить** все целые числа ℤ? Да, можно. Например, зафиксируем число 0 и будем последовательно прибавлять 1 в одном направлении и убавлять 1 в другом.

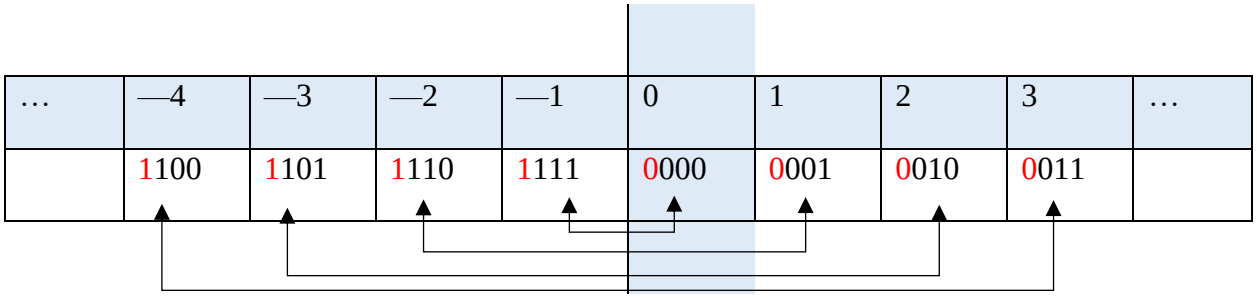


Рис. 5 Построение числового ряда

Но множество целых чисел бесконечно. Это значит, что нужно сузить рассматриваемый диапазон чисел, чтобы определить на нем **целочисленный тип**.

Поскольку вычислительная техника использует двоичный формат хранения, то сначала выделяют количество **бит** для числа, а затем определяют десятичные границы.

Нулевой разряд (выделен красным на рис. 5) имеет смысл знака числа, т. е. равен единице, если число отрицательное.

Например, рассмотрим тип **byte** (байт), размер которого равен восьми битам.

—128	—127	…	—2	—1	0	1	…	126	127
1000 0000	1000 0001		1111 1110	1111 1111	0000 0000	0000 0001		0111 1110	0111 1111
Отрицательные числа					Неотрицательные числа				

Рис. 6 Область значений для типа byte

Обратите внимание, что числовая ось разделена пополам, а первое число во второй половине — ноль. Это характерно для всего в мире программирования — начинаться с нуля: числа, индексы, значения, адреса и пр.

Рис. 7 Программисты любят шутить на этот счет и очень обижаются, когда их пытаются переубедить в том, что нумерация в повседневности начинается с единицы (pikabu)

Для нашего удобства в Java предопределено пять целочисленных типов, которые мы можем использовать для хранения значений подходящих диапазонов.

Тип Java	Обертка	Размер в битах	Левая граница	Правая граница	Десятичный порядок $< 10^n$
byte	Byte	8	—128	127	3
short	Short	16	—32768	32767	5
char	Char	16	0	65 535	5
int	Integer	32	—2 147 483 648	2 147 483 647	8
long	Long	64	—9 223 372 036 854 775 808	9 223 372 036 854 775 807	17

Рис. 8 Целочисленные типы языка Java

Большие ли это значения и достаточно ли их на практике? Далее представлена таблица некоторых характерных величин для сравнения.

Наименование величины	Десятичное значение	Нужно бит для хранения
Людей на Земле	$7.9 \cdot 10^9$	33
Денег на планете в долларах	$1.2 \cdot 10^{15}$	51
Байт передается через Интернет каждый год	$2.0 \cdot 10^{21}$	71
Число элементарных частиц во Вселенной	$3.28 \cdot 10^{80}$	268
Произвольная величина x	x	$\lceil \log_2 x \rceil$

Рис. 9 Некоторые занимательные величины

Мы можем использовать подчеркивания для того, чтобы разделять разряды чисел и улучшать читаемость кода. Для числовых литералов типа **long** необходимо приписывать букву L к концу их записи.

```
byte b = 5; // Вокзалов в СПб
short s = 685; // Школ в СПб
int i = 146_171_015; // Население РФ
long l = 7_902_614_691L; // Население мира
```

*Код 3. **Объявление** и **инициализация** целочисленных переменных с помощью **числовых литералов** — фиксированных числовых значений*

По умолчанию, целочисленные числовые литералы имеют тип **int**.

Переполнение числового типа

Очевидно, сумма двух равных по модулю и противоположных по знаку чисел равна нулю. Это не зависит от системы счисления — десятичной, двоичной или любой другой, т. е. все правила арифметики сохраняются.

$$x_{10} + (-x_{10}) = 0$$

$$x_2 + (-x_2) = 0$$

$$x_n + (-x_n) = 0$$

Однако компьютер выполняет арифметические операции **по модулю**. Поскольку на целочисленную переменную выделяется конечное число двоичных разрядов (бит), то при арифметических операциях может возникнуть ситуация **переполнения типа** (overflow).

Например, для числа 7 в байтовой записи:

$7_{10} + (-7_{10}) = 0$

$$7_2 + (-7_2) = 0000\ 0111_2 + (-0000\ 0111_2) = 0000\ 0111_2 + 1111\ 1001_2$$
$$= 1\ 0000\ 0000_2 = [\text{происходит переполнение разряда}] = 0000\ 0000_2$$
$$= 0$$

Выделенный желтым разряд выходит за границы определенных на типе разрядов, поэтому мы должны отбросить его.

В рассмотренном случае переполнение типа помогает нам производить арифметические операции, иначе выражение $7_2 + (-7_2)$ было бы равно 256, а не 0. Но на практике переполнение чаще влечет негативные последствия.

Классической иллюстрацией к этой проблеме выступает **хранение времени**. Исторически сложилось, что время в компьютерах хранится как неотрицательное количество секунд, прошедших с полуночи 1 января 1970 года по Гринвичскому меридиану (UNIX timestamp), в стандартном незнаковом 32-битном **int**.

С самого начала понимали, что переполнение типа произойдет через 2 147 483 647 секунд, но много ли это? Это соответствует третьему часу ночи 19 января 2038 года, что уже совсем скоро. Программисты всего мира озадачены разрешением этой проблемы, поскольку иначе человечеству угрожает настоящий цифровой апокалипсис, и мы вернемся в 1970-ый год.

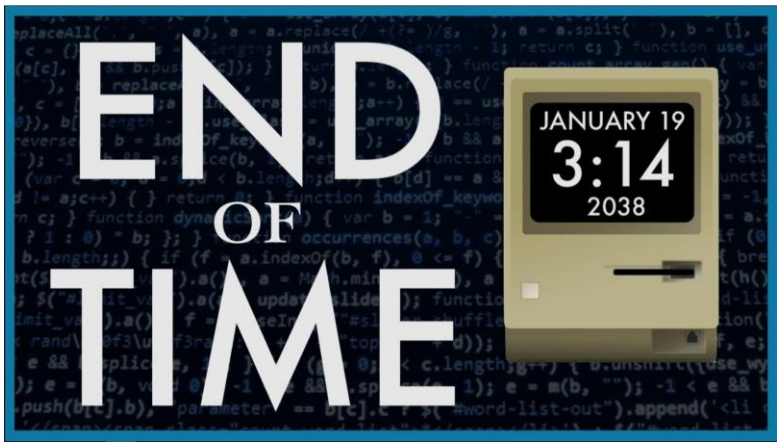


Рис. 10 Видео на английском языке о проблеме времени (YouTube)

Примечательно, что попытка поставить дату, предшествующую 1 января 1970 года, на iPhone приводит к его необратимому “окирпичиванию”.

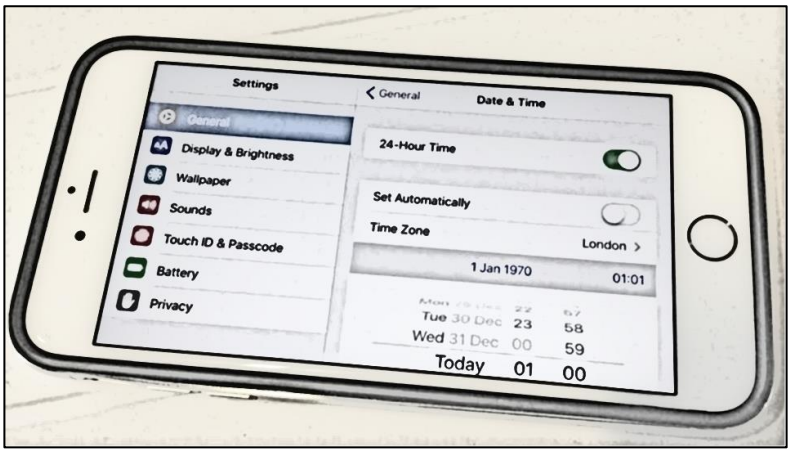


Рис. 11 Меню настроек iOS в iPhone (MacWorld)

Вещественные числа (с плавающей запятой)

Давайте вспомним, какие выделяют множества чисел.

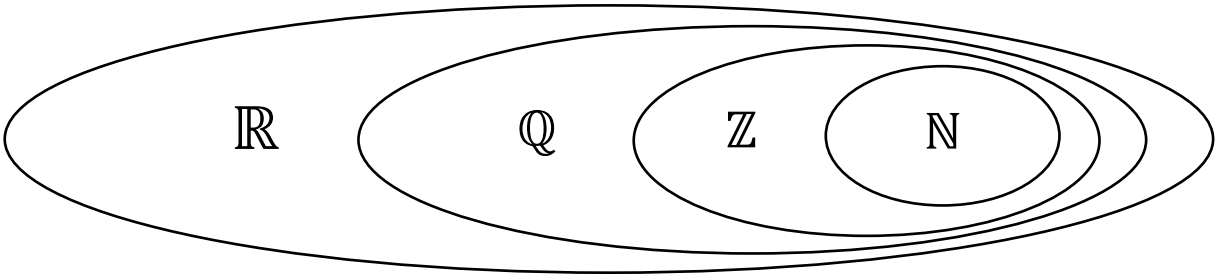


Рис. 12 Множества чисел

$\mathbb{N}$ — *натуральные числа (1, 2, 3, …)*

$\mathbb{Z}$ — *целые числа (…, -3, -2, -1, 0, 1, 2, 3, …)*

$\mathbb{Q}$ — *рациональные числа (представимы в виде дроби* $\frac{n \in \mathbb{Z}}{m \in \mathbb{N}}$*; например, -7.4, 29.3)*

$\mathbb{R}$ — *вещественные (+ иррациональные: например,* $\sqrt{5}, \pi, \sqrt{5-\sqrt{2}}$*)*

Вещественные числа $\mathbb{R}$ отличаются от целых $\mathbb{Z}$ тем, что их множество неперечислимо, т. е. на любом отрезке числовой оси помещается бесконечно много вещественных чисел.

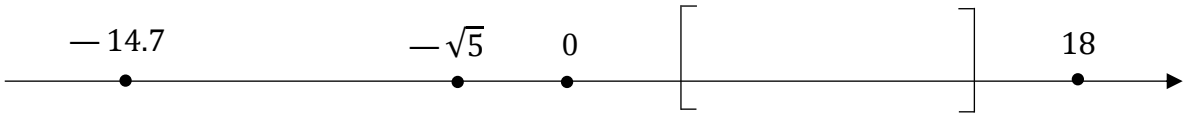


Рис. 13 На выделенном отрезке, как и на любом другом ненулевом, содержится бесконечное подмножество $\mathbb{R}$

Как в таком случае организовать хранение вещественных чисел? — Давайте представим их как **диапазон с выделенной точностью**. Абстрагируясь от теории хранения чисел с плавающей запятой, рассмотрим вещественные типы в языке Java.

Тип Java	Обертка	Размер в битах	Помещается десятичных знаков	Название точности
float	Float	32	7	Одинарная
double	Double	64	15	Двойная

Рис. 14 Вещественные типы на Java

Считается правилом хорошего тона использовать тип **double** двойной точности при вычислениях и хранении их результатов, а **float** — только для программных констант небольшой точности.

По умолчанию, вещественные числовые литералы имеют тип **double**. Чтобы явно задать значение в типе float, необходимо приписать к числовому литералу букву f.

```
double pi = 3.1415926535897; // число Пи
float q = 5.5f; // %, ключевая ставка Банка России
```

*Код 4. **Объявление** и **инициализация** переменных вещественных типов*

Для вещественных чисел верны все операции, определенные для целых чисел, но добавляется и множество новых:

Функция оператора	Формат использования	Комментарий
import java.lang.*;		
Нижняя целая граница $\lfloor a \rfloor$	Math.floor(a)	Ближайшее целое число к числу a, не большее его
	Math.floor(40) == 40 Math.floor(40.7) == 40	Math.floor(-40) == -40 Math.floor(-40.7) == -41
Верхняя целая граница $\lceil b \rceil$	Math.ceil(b)	Ближайшее целое число к числу a, не меньшее его
	Math.ceil(40) == 40 Math.ceil(40.7) == 41	Math.ceil(-40) == -40 Math.floor(-40.7) == -40
Натуральный логарифм $\ln a$	Math.log(a)	Чтобы найти логарифм с произвольным основанием, используют свойство $\log_a b = \frac{\ln b}{\ln a}$
Возведение числа a в степень b	Math.pow(a, b)	

Рис. 15 Операции над вещественными числами.

Логический тип

Наверное, самым простым является **логический тип**, булев тип или булиан (**boolean** или bool), названный в честь английского математика Джорджа Буля.

Переменные логического типа могут принимать одно из двух значений: да/истина (**true**) или нет/ложь (**false**).

Рис. 16 Тумблер тоже можно воспринять как значение логического типа.

Как можно заметить, значение логического типа несет информацию величиной в один **бит**. Логический тип применяется повсеместно: с помощью него мы можем записывать условия, а также хранить параметры.

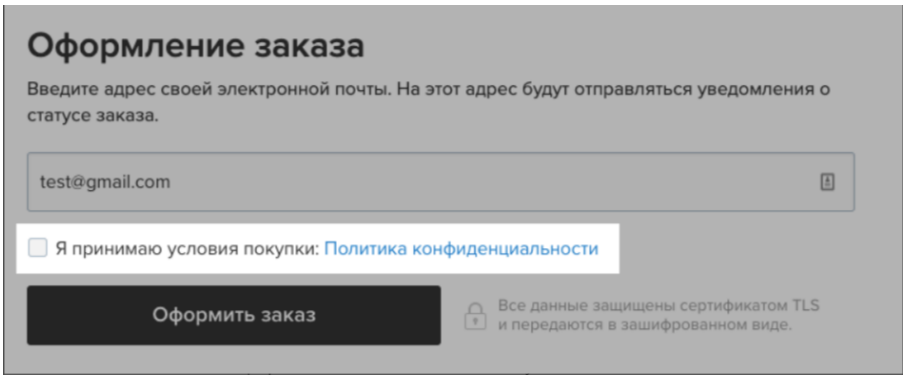


Рис. 17 Галочки в формах на сайтах — нагляднейшая иллюстрация для булевого значения

Значения булевых переменных записывают с помощью слов **true** и **false**.

```
// направление движения поезда метро
boolean toNorth = true;
// принять политику конфиденциальности
boolean acceptPolicy = false;
// сохранять куки на сайте
boolean saveCookies = true;
```

Код 5. Булевые (логические) переменные на Java

Для сравнения и проверки на равенство чисел используют следующие операторы, возвращающие значение типа boolean: a < b, a <= b, a > b, a >= b, a == b.

Символьный тип

До этого в разделе о целых числах мы упомянули тип **char**.

Тип Java	Обертка	Размер в битах	Левая граница	Правая граница	Десятичный порядок $< 10^n$
char	Char	16	0	65 535	5

Рис. 18 Целочисленный тип char

Этот тип действительно имеет числовое представление, но его главное предназначение — кодирование **символов** в UTF-16. Начало кодовой таблицы совпадает с таблицей ASCII, но продолжается символами национальных алфавитов и смайлами.

```
char D = 'D';
char Ы = 'Ы';
System.out.println(D);
System.out.println(Ы);
System.out.println((int) D);
System.out.println((int) Ы);
```

```
D
Ы
68
1067
```

Код 6. Объявление символьных типов

Обратите внимание, что каждый символ имеет свой порядковый номер, который мы вывели с помощью явного приведения к числу.

Строковый тип

Строка — это набор символов. Мы можем записать их с помощью строкового литерала.

```
String str = "Это строка. Здесь можно записывать мысли и не только...";
```

Код 7. Объявление строковой переменной

Класс

Мы рассмотрели целочисленные, вещественные, логический, символьный, строковый и перечисляемый типы. Класс позволяет объединить несколько переменных различных типов в одну составную переменную.

```
class Person {
    String name;
    short age;
    boolean isMale;
}
```

Код 8. Класс как агрегат атрибутов

Класс объявляют с помощью служебного слова **class** и перечисляют в нем внутренние переменные.

```
Person p = new Person();
p.name = "Ivan";
p.age = 16;
p.isMale = true;

System.out.println(p.name);
System.out.println(p.age);
System.out.println(p.isMale);
```

```
Ivan
16
true
```

Код 9. Создание переменной классового типа

Чтобы создать значение переменной класса, используют служебное слово **new**. С помощью оператора “**.**” мы можем обратиться к переменным класса и задать их значения.

Задачи

A. Сравнение вещественных чисел (прием)

В стандартном входе даны два вещественных числа.

Выведите true, если они равны, и false, если они не равны.

Входные данные: два вещественных числа (с плавающей точкой).

Выходные данные: “true” или “false”.

System.in	System.out
3.146 3.146	true
-0.433 1.242	false

B. Приведение long к типу integer

В стандартном входе дано одно целое число.

Определите, можно ли сохранить его в типе **integer**.

Входные данные: целое число в диапазоне типа **long**.

Выходные данные: “true” или “false”.

System.in	System.out
-3211	true
372036854775	false

C. Бит в числе

В стандартном входе дано одно целое число.

Определите минимальное количество бит, необходимое для записи этого числа в двоичной системе счисления.

Входные данные: целое число.

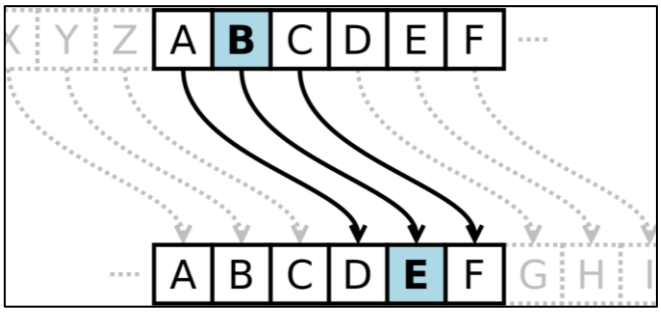
Выходные данные: число бит.

System.in	System.out	Комментарий к тесту
3	2	$3_{10} = 11_2$
45	6	$45_{10} = 101101_2$

D. Шифр Цезаря

Простейшим методом шифровки сообщений является код Цезаря. Суть его в следующем: все символы в тексте сдвигаются на фиксированное число в алфавите.

Например, на следующей иллюстрации показан сдвиг на 3 символа.



Напишите программу, осуществляющую такой сдвиг.

Входные данные: один символ и целый сдвиг.

Выходные данные: символ в UTF-16, зашифрованный кодом Цезаря.

System.in	System.out
A 5	F
8 -4	4

E. Эпокалипсис

Классической иллюстрацией переполнения типа является проблема 2038 года (см. Переполнение числового типа), возникающая из-за использования беззнакового 32-битного integer.

Представьте, что мы находимся в штаб-квартире IBM и выбираем, сколько бит нужно использовать для хранения времени (в той же схеме от 1970 года). Начальник настаивает на том, чтобы очередное переполнение произошло не ранее полуночи 1 января года N.

Определите минимальное возможное количество бит для нового временно́го типа, удовлетворяющего требованию начальника.

Входные данные: целый год $N \geq 1970$.

Выходные данные: количество бит для нового целочисленного знакового типа (аналог **integer** с другим количеством бит).

Указание. *Считайте, что в году содержится 365.25 дней.*

System.in	System.out
2038	32
1980	30

F. Вклад

Алина накопила S тысяч рублей и 3 сентября 2021 года принесла их в банк, чтобы оформить вклад. Условия вклада просты: каждый год в полночь на 3 сентября банк увеличивает текущую сумму на Q процентов.
Прошло T лет, и Алина приняла решение немедленно забрать свой вклад.

Помогите Алине проверить, не обхитрил ли ее банк, и определите прибыль, которую она должна получить.

Входные данные: сумма вклада S тыс. рублей, процентная ставка Q в %, время T в годах. $S, Q, T \in \mathbb{Q}$ и > 0.

Выходные данные: прибыль (разность конечной и начальной сумм) в тыс. рублей.

System.in	System.out	Комментарий к тесту
10 10 1	1	
100 5 1.8	5	Алина не дождалась 3 сентября второго года вклада и получила проценты лишь за первый год.

I. Творческое задание. Перечисления

Спроектируйте два перечисляемых типа из интересной вам области. В каждом из двух типов должно быть не менее трех допустимых значений.
Например, ветки метрополитена Петербурга, школьные оценки и пр. (повторять эти примеры не следует)

II. Творческое задание. Классы

Спроектируйте два класса. В каждом классе должно быть не менее четырех атрибутов. Создайте переменные типа ваших классов и заполните их атрибуты из стандартного потока ввода.

Например, биографическая анкета человека, описание телефона или другой техники в интернет-магазине и пр. (эти примеры можно использовать)

Вопросы

Ответы на следующие вопросы вы можете получить, в том числе написав программы.

1. Какой (-ие) тип (-ы) данных вы бы выбрали для хранения:
    a. Номера химического элемента из периодической системы Д. И. Менделеева
    b. Среднего значения оценок за четверть в школе по предмету
    c. Названия станции метрополитена
    d. Курса иностранных валют

e. Цвет в формате RGB

2. Представим, что время в компьютерах (см. Переполнение числового типа) бы хранилось не в переменной типа **int** (32 бита), а в переменной типа **long** (64 бита). Определите, в каком году произойдет переполнение этого типа. Хватит ли этого значения на весь оставшийся период существования человечества?