**ЛАБОРАТОРНАЯ РАБОТА №1**

**Краткая справка**

Строка Java - это набор символов, собранных вместе, например, слово «Hello» или фраза «практика совершенствует». Создайте строку в коде, написав ее символы между двойными кавычками.

String str = "Hello";

На этом рисунке показан строковый объект в памяти, состоящий из отдельных символов.  Цифры индекса означают 0, 1, 2 .. позже.

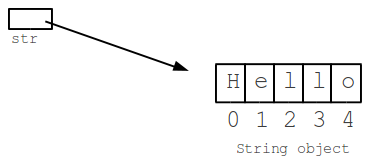


Рис.1

Строки не просто состоят из букв az. Символами могут быть знаки препинания и другие разносторонние символы. Например, в строке «hi» 3-й символ - это пробел. Это все работает и со строками, хранящимися в переменных, например:

String fruit = "яблоко";

String stars = "\*\*\*";

Строка а = фрукты + звезды; // а "яблоко \*\*\*"

Длина» строки - это просто число символов в ней. Таким образом, «hi» - это длина 2, а «Hello» - это длина 5. Метод length () в строке возвращает ее длину, например:

String a = "Hello";

int len ​​= a.length (); // длина 5

Для проверки на совпадение 2 строк, используется метод equals (). Строка «HELLO» считается отличной от строки «hello».  Используйтся == для сравнения примитивных значений, таких как int и char. Существует вариант equals (), называемый equalsIgnoreCase (), который сравнивает две строки, игнорируя различия в верхнем и нижнем регистре.

Метод indexOf (String target) выполняет поиск слева направо внутри заданной строки для «целевой» строки. Метод indexOf () возвращает номер индекса, в котором целевая строка была найдена впервые, или -1, если цель не найдена. Как и equals (), метод indexOf () «чувствителен» к регистру, поэтому символы в верхнем и нижнем регистре считаются разными.

int indexOf (String target, int fromIndex) - выполняет поиск **цели** слева направо, как обычно, но начинает поиск с заданного fromIndex. На самом деле fromIndex не обязательно должен быть действительным. Если оно отрицательное, поиск происходит с начала строки. Если fromIndex больше длины строки, то будет возвращено -1.

int lastIndexof (String target, int fromIndex) - выполняет поиск справа налево, начиная с заданного fromIndex.

**Операторы сравнения Java: <, <=,>,> =**

**Существует четыре оператора сравнения:**

<меньше чем

> больше чем

<= меньше или равно (то есть ≤)

> = больше или равно (то есть ≥)

Если тело цикла while или оператора if состоит из множества операторов, то использутся фигурные скобки {} для группировки операторов в теле.

При написании сравнений часто есть разные способы сказать одно и то же. Например, если проверить, является ли оценка int 100 или более, любое из следующего сделает то же самое:

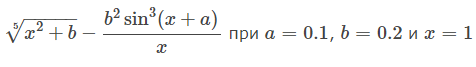
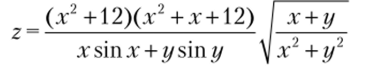
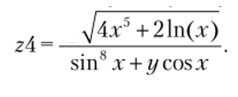
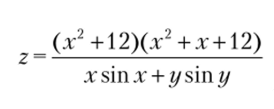
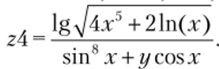
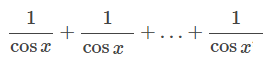
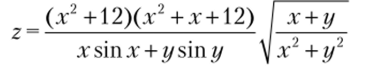
if (оценка> = 100) {... // версия 1

if (100 <= оценка) {... // версия 2

if (оценка> 99) {... // версия 3

**Практическая часть:**

Вычислите значение выражений на языке java:

1. (a+4b)(a−3b)+a2 , при при a=2 и b=3
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. Пользователь вводит три числа. Найдите среднее арифметическое этих чисел, а также разность удвоенной суммы первого и третьего чисел и утроенного второго числа.
11. Пользователь вводит сторону квадрата. Найдите периметр и площадь квадрата.
12. 
13. 
14. Даны катеты прямоугольного треугольника. Найдите площадь, периметр и гипотенузу треугольника.
15. Пользователь вводит количество дней, указывает процент скидки и вводит сумму. Рассчитать прибыль, если за каждый день сумма увеличивается на 3 $  и затем применяется скидка.
16. 
17. Из трехзначного числа x вычли его последнюю цифру. Когда результат разделили на 10, а к частному слева приписали последнюю цифру числа x, то получилось число 237. Найти число x.
18. 
19. 
20. 
21. 
22. 
23. 
24. 
25. 
26. Выведите на экран квадрат из нулей и единиц, причем нули находятся только на диагонали квадрата. Всего в квадрате сто цифр.
27. Вывести на экран 20 строк. В строках с четными номерами вывести по 10 чисел, равных номеру строки. В строках с нечетными номерами вывести десять единиц.
28. 
29. 
30. 