**Лабораторная работа №9.**

# **Одномерные массивы**

Часто во время выполнения программы необходимо хранить большое количество значений. Пусть, например, вам нужно считать 100 чисел, вычислить их среднее значение и узнать, сколько чисел больше этого значения. Следовательно, программа сначала считывает числа и вычисляет их среднее значение, затем сравнивает каждое число со средним значением, чтобы определить, превосходит ли оно его. Для выполнения этой задачи все числа должны храниться в переменных, т.е. необходимо объявить 100 переменных и 100 раз написать практически идентичный код. Писать программу таким образом было бы нецелесообразно. Итак, каким образом мы будем решать эту проблему?

Необходим эффективный и организованный подход. Java и большинство других языков программирования высокого уровня обеспечивают структуру данных, которая называется массив, для хранения последовательной коллекции фиксированного размера из элементов одного типа данных. В вышеописанной ситуации можно сохранить все 100 чисел в одном массиве и получить к ним доступ через одну переменную.

На следующем рисунке показан массив из 100 элементов типа double, к которым можно обратиться с помощью одной переменной x.

[](https://cs.sberuniversity.online/image/full/full/resize/398569d0-662c-11eb-9aa9-0242ac180005)

*При создании массива фиксируется его размер. Для доступа к элементу массива используется индексированная переменная.*

Массив используется для хранения коллекции данных, но чаще мы будем рассматривать его как набор переменных одного типа. Вместо объявления отдельных переменных, например, **number0, number1, …, number99**, объявляется одна *переменная массива*, например, **numbers**, и используются *индексированные переменные*, например, **numbers[0], numbers[1]**, ..., **number[99]** для доступа к отдельным элементам массива. В этом подразделе описывается, как объявлять, создавать и обрабатывать массивы с помощью индексов.

**Объявление массивов**

Чтобы использовать в программе массив, необходимо объявить переменную для хранения ссылки на массив и указать тип элемента массива.

*Синтаксис объявления переменной массива:*

ТипЭлемента[] ссылкаНаМассив;

или

ТипЭлемента ссылкаНаМассив[]; *// Допускается, но менее предпочтителен*

**ТипЭлемента** может быть любым типом данных, а все элементы массива будут иметь один и тот же тип данных. Например, в следующем коде объявляется переменная **myList**, которая ссылается на массив элементов типа **double**.

double[] myList;

или

double myList[]; *// Допускается, но менее предпочтителен*

**Создание массивов**

В отличие от объявления переменных примитивного типа данных, при объявлении переменной массива выделяется память не для всего массива, а только для ссылки на массив. Если переменная не содержит ссылку на массив, то значение этой переменной равно **null**. До тех пор пока массив не создан, нельзя присвоить значения его элементам. После объявления переменной массива можно создать массив с помощью оператора **new** и присвоить переменной ссылку на него с помощью следующего синтаксиса:

ссылкаНаМассив = new ТипЭлемента[размерМассива];

Это предложение, во-первых, создает массив, используя **new ТипЭлемента [размерМассива]**, и, во-вторых, присваивает переменной **ссылкаНаМассив** ссылку на только что созданный массив.

Объявление переменной массива, создание массива и присваивание переменной ссылки на массив можно объединить в одном предложении следующим образом:

ТипЭлемента[] ссылкаНаМассив = new ТипЭлемента[размерМассива];

или

ТипЭлемента ссылкаНаМассив[] = new ТипЭлемента[размерМассива];

Пример такого предложения:

double[] myList = new double[10];

В этом предложении объявляется переменная массива **myList**, создается массив из **10** элементов типа **double** и присваивается переменной **myList** ссылка на этот массив. Для присваивания значений элементам массива используется следующий синтаксис:

ссылкаНаМассив[индекс] = значение;

Например, в следующем коде инициализируется массив **myList**:

myList[0] = 5.6;

myList[1] = 4.5;

myList[2] = 3.3;

myList[3] = 13.2;

myList[4] = 4.0;

myList[5] = 34.33;

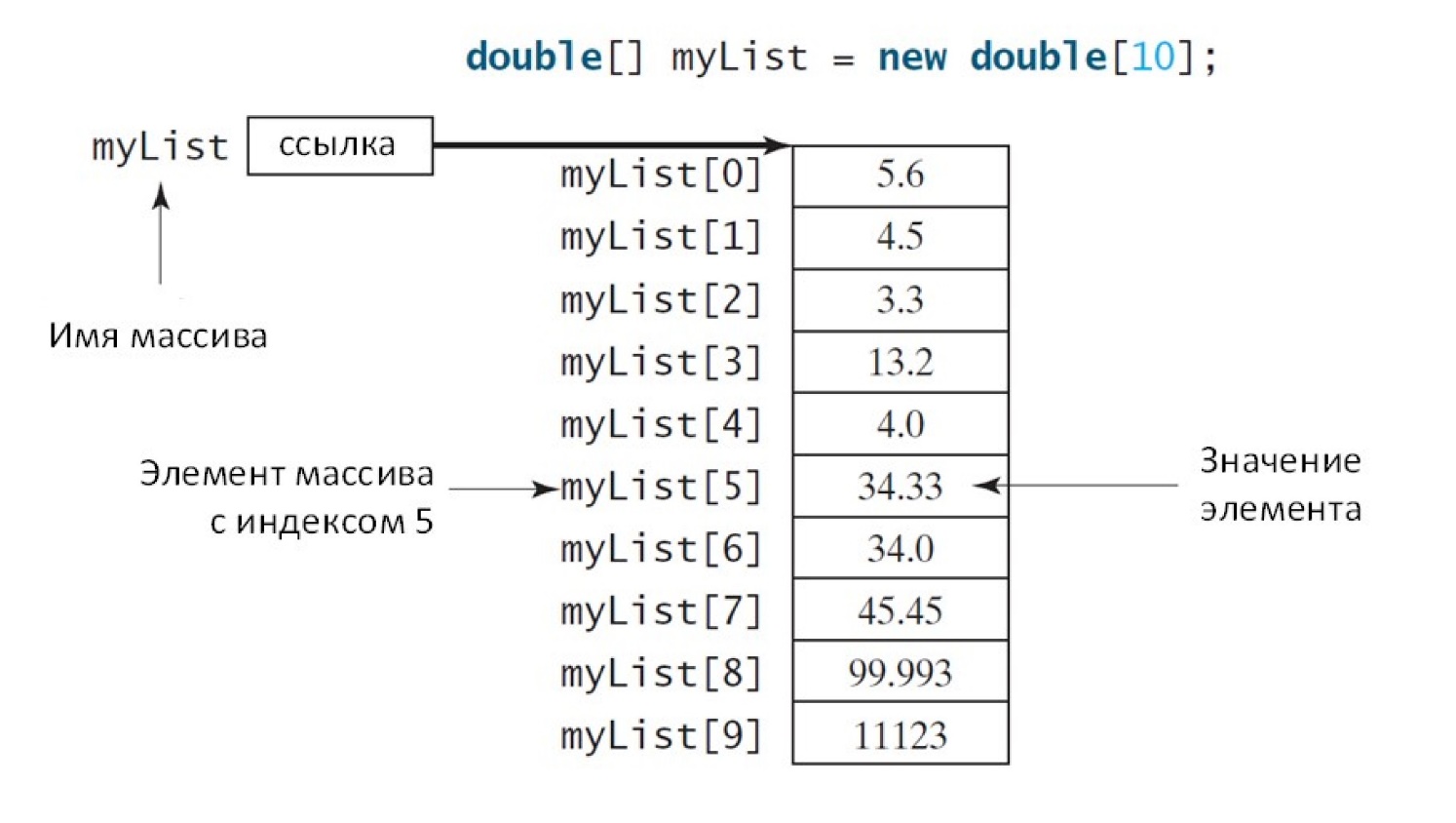
myList[6] = 34.0;

myList[7] = 45.45;

myList[8] = 99.993;

myList[9] = 11123;

Этот массив показан на следующем рисунке.

[](https://cs.sberuniversity.online/image/full/full/resize/d8e2e460-646c-11eb-9e55-0242ac180005)

* Заметка

Переменная массива, которая, казалось бы, содержит сам массив, фактически содержит ссылку на этот массив. Строго говоря, переменная массива и массив — это разные понятия, но большую часть времени это различие можно игнорировать. Таким образом, для простоты дальнейшего изложения будем говорить, что **myList** является массивом, и не будем акцентировать внимание на том, что на самом деле **myList** является переменной, которая содержит ссылку на массив элементов типа **double.**

**Размер массива и значения по умолчанию**

При выделении памяти для массива необходимо задать *размер* массива, указав количество элементов, которые могут быть в нем сохранены. После создания массива его размер не может быть изменен. Размер можно получить с помощью **ссылкаНаМассив.length**. Например, **myList.length** равно **10**.

При создании массива его элементам по умолчанию присваивается значение **0** для примитивных числовых типов данных, **\u0000** — для символьного типа и **false** — для типа **boolean**.

**Доступ к элементам массива**

Доступ к элементу массива осуществляется через индекс. Индексы массива начинаются с **0**, то есть они находятся в диапазоне от **0** до **ссылкаНаМассив.length - 1**. На предыдущем рисунке массив **myList** содержит **10** значений типа **double**, а индексы массива находятся в диапазоне от **0** до **9** включительно.

Каждый элемент массива представляется с помощью следующего синтаксиса, называемого индексированной переменной:

ссылкаНаМассив[индекс];

Например, **myList[9]** представляет последний элемент массива **myList**.

Индексированная переменная может использоваться так же, как обычная переменная. Например, в следующем коде в **myList[2]** суммируются значения **myList[0]** и **myList[1]**:

myList[2] = myList[0] + myList[1];

В следующем цикле переменной **myList[0]** присваивается **0**, **myList[1]** присваивается **1**, ...,**myList[9]** присваивается **9**:

for (int i = 0; i < myList.length; i++) {

myList[i] = i;

}

**Инициализаторы массивов**

Язык Java имеет краткую нотацию, называемую *инициализатором*массива, которая сочетает объявление, создание и инициализацию массива в одном предложении с помощью следующего синтаксиса:

ТипЭлемента[] ссылкаНаМассив = {значение0, значение1, ..., значениеk};

Например, предложение

double[] myList = {1.9, 2.9, 3.4, 3.5};

объявляет, создает и инициализирует массив **myList** с четырьмя элементами, что эквивалентно следующим предложениям:

double[] myList = new double[4];

myList[0] = 1.9;

myList[1] = 2.9;

myList[2] = 3.4;

myList[3] = 3.5;

* Внимание

В синтаксисе инициализатора массива не используется оператор **new**. С помощью инициализатора массива можно объявить, создать и инициализировать массив только в одном-единственном предложении. Разделение этих операций может привести к синтаксической ошибке. Таким образом, следующее предложение является недопустимым:

double[] myList;

myList = {1.9, 2.9, 3.4, 3.5}; *// Не допустимо*

**Обработка массивов**

При обработке элементов массива часто используется цикл **for** по следующим причинам:

1. Все элементы массива одного типа данных, поэтому они могут быть последовательно многократно обработаны единым способом с помощью цикла.
2. Так как известен размер массива, логично использовать счетный цикл, например, цикл **for**.

Пусть массив создан следующим образом:

double[] myList = new double[10];

Далее приведены примеры обработки массива:

1. Инициализация массива с входными значениями

В следующем цикле инициализируется массив **myList** с входными данными от пользователя:

1. java.util.Scanner input = new java.util.Scanner(System.in);
2. System.out.print("Введите " + myList.length + " значений: ");
3. for (int i = 0; i < myList.length; i++)
4. myList[i] = input.nextDouble();
5. Инициализация массива со случайными значениями

В следующем цикле инициализируется массив **myList** со случайными значениями от **0.0** до **100.0**:

1. for (int i = 0; i < myList.length; i++) {
2. myList[i] = Math.random() \* 100;
3. }

3. Отображение массива

Для отображения массива необходимо вывести каждый элемент массива с помощью цикла, например:

for (int i = 0; i < myList.length; i++) {

System.out.print(myList[i] + " ");

}

Массив типа **char[]** можно отобразить с помощью одного предложения **println()**. Например, в следующем коде отображается **Moscow**:

char[] city = {'M', 'o', 's', 'c', 'o', 'w'};

System.out.println(city);

4. Суммирование всех элементов массива

Используйте переменную с именем **total** для хранения суммы. Инициализируйте **total** со значением **0**. С помощью цикла прибавьте к сумме каждый элемент массива, например:

double total = 0;

for (int i = 0; i < myList.length; i++) {

total += myList[i];

}

1. Поиск наибольшего элемента массива
2. Используйте переменную с именем **max** для хранения наибольшего элемента. Инициализируйте **max** со значением **myList[0]**. Чтобы найти наибольший элемент в массиве **myList**, сравните каждый элемент с **max** и обновите **max**, если текущий элемент больше **max**.
3. double max = myList[0];
4. for (int i = 1; i < myList.length; i++) {
5. if (myList[i] > max)
6. max = myList[i];
7. }
8. Поиск наименьшего индекса наибольшего элемента массива
9. Часто требуется найти наибольший элемент в массиве. Если массив имеет несколько элементов с одинаковым наибольшим значением, то найдите наименьший индекс среди этих элементов. Предположим, массив **myList** равен **{1, 5, 3, 4, 5, 5}**. Наибольший элемент равен**5**, а наименьший индекс для **5** равен **1**. Используйте переменную с именем **max** для хранения наибольшего элемента, а переменную с именем **indexOfMax** для хранения индекса наибольшего элемента. Инициализируйте max с **myList[0]**, а **indexOfMax** с **0**. Сравните каждый элемент в **myList** с **max** и обновите **max** и**indexOfMax**, если элемент больше **max**.
10. double max = myList[0];
11. int indexOfMax = 0;
12. for (int i = 1; i < myList.length; i++) {
13. if (myList[i] > max) {
14. max = myList[i];
15. indexOfMax = i;
16. }
17. }

7. Перетасовка элементов массива

Во многих приложениях необходимо случайным образом переупорядочивать элементы массива. Чтобы выполнить перетасовку, для каждого элемента **myList[i]** генерируется случайный индекс **j**, и **myList[i]** с **myList[j]** переставляются следующим образом:

for (int i = 0; i < myList.length; i++) {

*// Сгенерировать случайный индекс j*

int j = (int)(Math.random() \* myList.length);

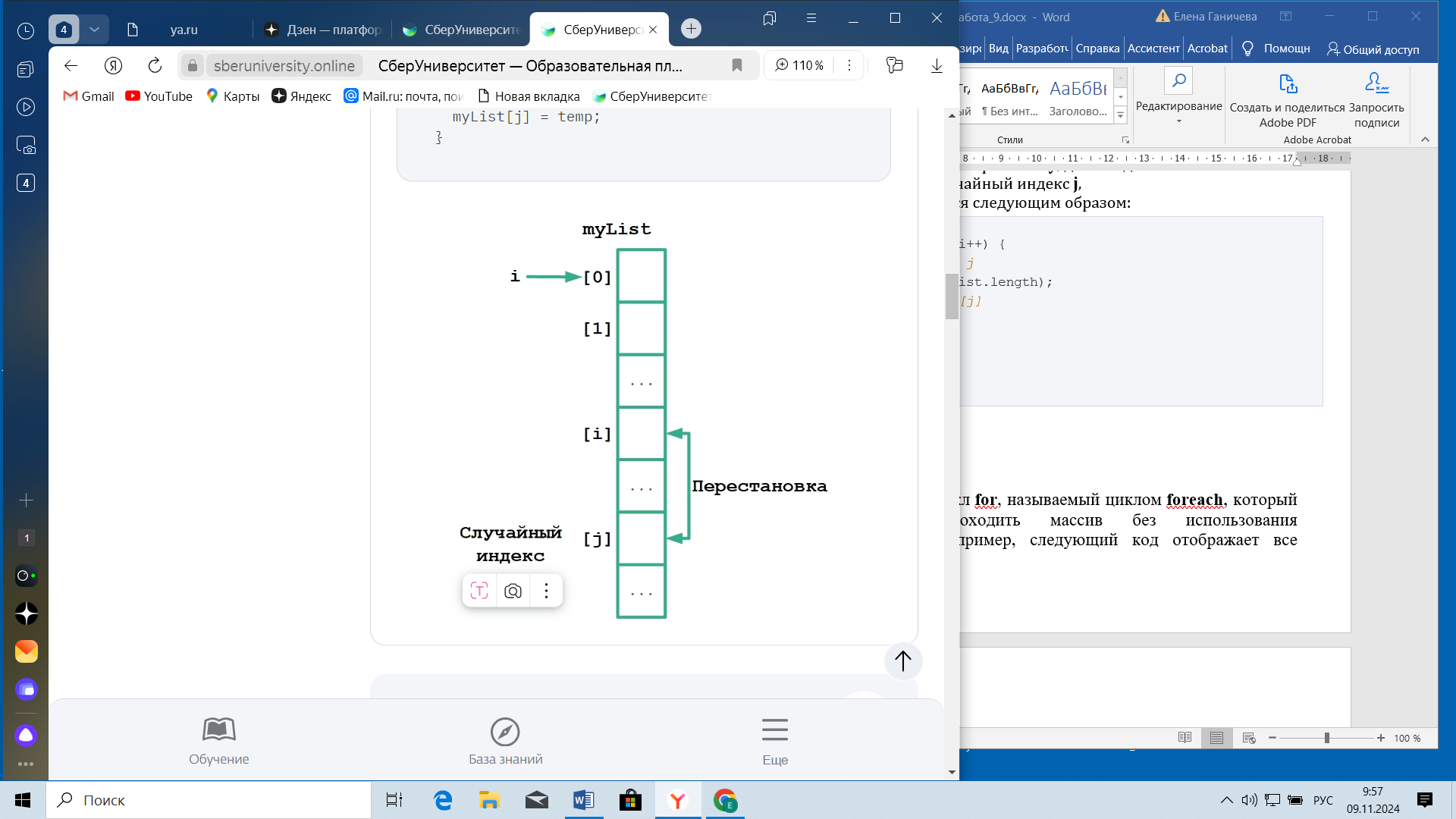
*// Переставить myList[i] и myList[j]*

double temp = myList[i];

myList[i] = myList[j];

myList[j] = temp;

}



8. Сдвиг элементов массива

Иногда необходимо сдвигать элементы массива влево или вправо. Пример смещения элементов на одну позицию влево и заполнения последнего элемента значением первого элемента:

double temp = myList[0]; *// Сохраняет первый элемент*

*// Сдвинуть элементы влево*

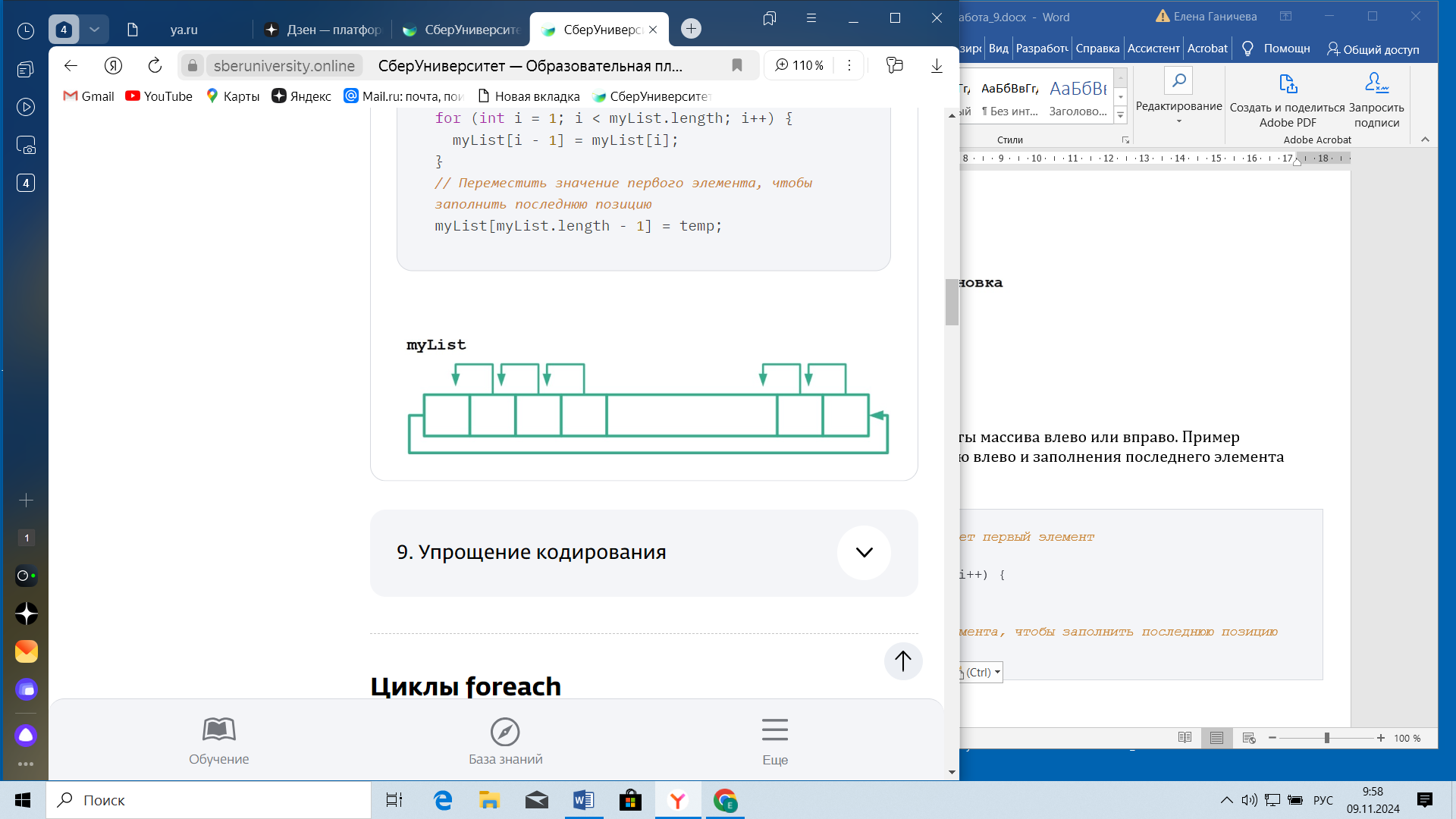
for (int i = 1; i < myList.length; i++) {

myList[i - 1] = myList[i];

}

*// Переместить значение первого элемента, чтобы заполнить последнюю позицию*

myList[myList.length - 1] = temp;



9. Упрощение кодирования

Массивы можно использовать для значительного упрощения кодирования некоторых задач. Допустим, вы хотите получить имя заданного месяца по его номеру. Если имена месяцев хранятся в массиве, то доступ к имени месяца для заданного месяца осуществляется непосредственно через индекс. Следующий код предлагает пользователю ввести номер месяца и отображает имя этого месяца:

String[] months = {"январь", "февраль", ..., "декабрь"};

System.out.print("Введите номер месяца (от 1 до 12): ");

int monthNumber = input.nextInt();

System.out.println("Месяц - " + months[monthNumber - 1]);

Если массив месяцев не используется, то необходимо определять имя месяца с помощью длинного многовариантного предложения **if-else** следующим образом:

if (monthNumber == 1)

System.out.println("Месяц - январь");

else if (monthNumber == 2)

System.out.println("Месяц - февраль");

...

else

System.out.println("Месяц - декабрь");

**Циклы foreach**

В Java поддерживается удобный цикл **for**, называемый циклом **foreach**, который позволяет последовательно проходить массив без использования индексированной переменной. Например, следующий код отображает все элементы массива **myList**:

for (double e: myList) {

System.out.println(e);

}

Этот код можно прочитать как «для каждого элемента **e** в **myList** выполните следующие действия». Обратите внимание, что переменная **e** должна быть объявлена с тем же типом данных, что элементы **myList**.

Как правило, синтаксис цикла **foreach** следующий:

for (ТипЭлемента элемент: ссылкаНаМассив) {

*// Обработать элемент*

}

Если требуется пройти по массиву в другой последовательности или изменить элементы массива, то необходимо использовать индексированную переменную.

* Внимание

Попытка доступа к массиву вне его границ — это распространенная ошибка программирования, которая выбрасывает исключение во время выполнения **ArrayIndexOutOfBoundsException.** Чтобы избежать этой ошибки, не обращайтесь к индексам после **arrayRefVar.length - 1** или просто используйте цикл **foreach**, если это возможно.

Программисты часто ошибочно ссылаются на первый элемент в массиве с помощью индекса **1**, а не **0**. Это называется ошибкой завышения на единицу. Другой распространенной ошибкой в цикле является использование **<=**, где должно использоваться **<**. Например, следующий цикл является некорректным:

for (int i = 0; i <= list.length; i++)

System.out.print(list[i] + " ");

Здесь оператор **<=** следует заменить на **<**. В этом случае можно избежать ошибки завышения на единицу с помощью цикла **foreach**.

**Задание для самопроверки:**

Что будет отображено в консоли следующим кодом?

int x = 30;

int[] numbers = new int[x];

x = 60;

System.out.println("x равно " + x);

System.out.println("Размер массива numbers равен " + numbers.length);

#### **Ответ:**

x равно 60 Размер массива numbers равен 30

**Задание для самопроверки:**

С какими из следующих утверждений вы согласны?

1. Каждый элемент массива одного и того же типа данных.

2. Размер массива фиксируется после объявления ссылки на массив.

3. Размер массива фиксируется после создания массива.

4. Элементы массива должны быть примитивного типа данных.

#### **Ответ:**

1. Согласен.  
2. Не согласен.  
3. Согласен.  
4. Не согласен.

**Задание 1.**

Какие из следующих предложений являются допустимыми?

1. int i = new int(30);

2. double d[] = new double[30];

3. char[] r = new char(1..30);

4. int i[] = (3, 4, 3, 2);

5. float f[] = {2.3, 4.5, 6.6};

6. char[] c = new char();

**Ваш вариант ответа:**

#### **Ответ**

1. Не допустимо.

2. Допустимо.

3. Не допустимо.

4. Не допустимо.

5. Не допустимо.

6. Не допустимо.

**Задание 2.**

Найдите и исправьте ошибки в следующем коде:

public class TestArray1 {

public static void main(String[] args) {

double[100] r;

for (int i = 0; i < r.length(); i++);

r(i) = Math.random \* 100;

}

}

**Ваш вариант ответа:**

**Задание 3.**

Что будет отображено в консоли следующим кодом?

public class Test {

public static void main(String[] args) {

int list[] = {1, 2, 3, 4, 5, 6};

for (int i = 1; i < list.length; i++)

list[i] = list[i - 1];

for (int i = 0; i < list.length; i++)

System.out.print(list[i] + " ");

}

}

**Ваш вариант ответа:**

**Решите задачу:**

**Задача 1. Минимальная процентная ставка**

Напишите программу, которая находит минимальную процентную ставку по кредиту в разных банках. Для взаимодействия с пользователем создайте графический интерфейс.

Исходные данные: количество процентных ставок, процентные ставки.

Результат: минимальная процентная ставка по банковскому кредиту.

Например,

Введем количество процентных ставок равным **5**.

Введем процентные ставки **17, 8.5, 11.9, 13, 10.5**.

Минимальная процентная ставка по банковскому кредиту равна **8.5**.

**Ваш ответ на задачу 1:**