

Лекция 7.1

Използване на масиви в Java

Основни теми

- Въведение в масиви
- Приложение на масиви за съхранение на данни и извличане на данни от таблици и списъци със стойности.
- Деклариране на масив, инициализацията му и рефериране на отделни стойности от масива.
- Примери
- Задачи

- 7.1 Въведение
 - 7.2 Масиви
 - 7.3 Деклариране и създаване на масиви
 - 7.4 Примери за приложения на масиви
- Задачи

Литература:

Java How to Program, Sixth Edition, глава 7

7.1 Въведение

- **Масиви**

- Структури от данни т.е. съвкупност от логически свързани елементи от определен тип данни
- **Масивите се състоят от данни от един и същ тип**
- **Масивите са референтен тип данни**
- ***Броят на елементите* определя дължината на масива**
 - **Веднъж създаден, дължината на масива остава неизменна**

Широк кръг приложения поради възможността да се обработват елементите на масива в цикъл (специализирана **for** команда)

7.2 Масиви

- **Определение**

- Масивът е подредена група от променливи (наричани елементи или компоненти) съдържащи **стойности от един и същ тип данни**
- Елементите могат да са от **примитивен** или **референтен** тип
- За достъп до определен елемент на масив и съответно до неговата стойност се позиция, наричана ***index*** или ***subscript***
- Масивите са **референтен тип данни**- представят се от променлива, която реферира обект от тип масив в ***Heap*** паметта

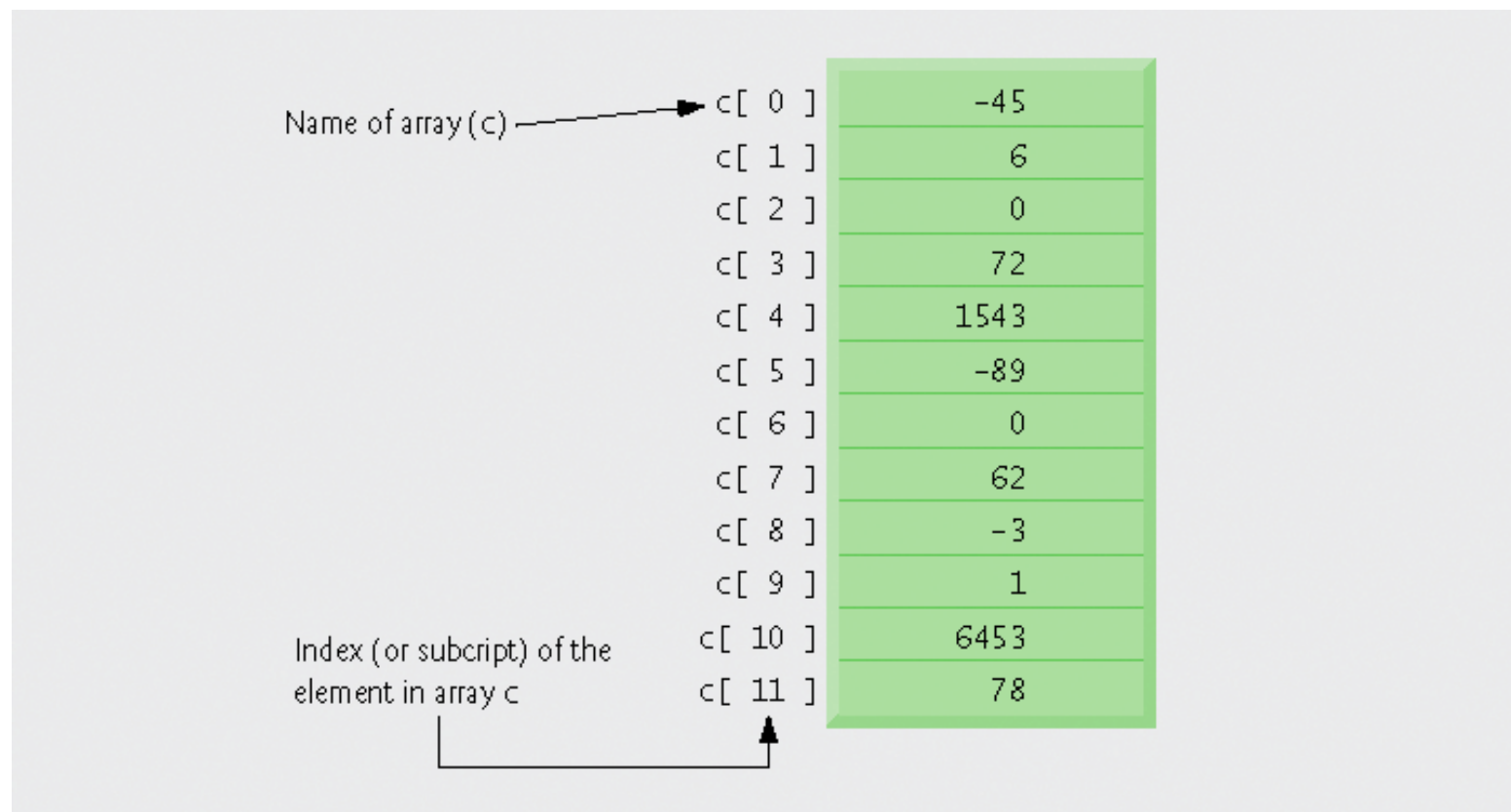


Fig. 7.1 | Един масив с 12 елемента (дължината на масива е 12).

7.2 Масиви

- Фигурата изобразява логическото представяне за масив от цели числа (целочислен масив), **рефериран** (именуван) с променливата **с**.
- Този масив има **12** елемента
- Програмно, всеки от тези елементи се реферира като се използва името на масива (или същото, променливата реферираща масива), следвана позицията (индекса) на този елемент в масива, заградена в квадратни скоби (**[]**).
- **Първият** елемент на **ВСЕКИ** масив има **индекс нула** и се нарича още нулевия елемент.
- Така, елементите на масива с се означават като **с[0], с[1], с[2]** и т.н..
Последният елемент в този масив има индекс **11**, което е с **1 по-малко от дължината** (12) на масива **с**.
- Променливите, рефериращи масиви (**имената на масиви**) спазват **правилата за писане на останалите променливи** на Java

7.2 Масиви ...

- **Индекси**

- Наричат се още **subscript**
- Задават **позицията на даден елемент** в квадратни скоби
- Винаги е **положително число** или **резултат от пресмятане на целочислен израз**
- Изменя се между **0** и **дължината на масива минус единица**

```
a = 5;  
b = 6;  
c[ a + b ] += 2;
```

- Добавяне на **2** към **c[11]**

Обичайна грешка при програмиране 7.1

Използване на променлива от тип *long* за индекс води до грешка при компилация.

Всеки индекс трябва да е от тип *int* или тип , който може да се сведе неявно до *int*, а именно *byte, short or char*, но не и *long*.

Обичайна грешка при програмиране 7.1

Използване на отрицателни стойности за индекс или стойности по-големи или равни на дължината на масива води до грешка при изпълнение на програмата и нейното прекратяване.

7.2 Массиви ...

- По- подробно за массива **c**
 - С променливата **c** е означено *името* на массива
 - **c.length** връща дължината на массива **c**
 - **c** има 12 елемента (**c[0]**, **c[1]**, ... **c[11]**)
 - Стойността на **c[0]** е равна на **-45**
 - Стойността на **c[1]** е равна на **6**
 - За пресмятане на *сумата от първите три елемента* на массива **c** и *присвоим резултата в променливата* **sum**, трябва да напишем на *Java* следната команда

```
sum = c[ 0 ] + c[ 1 ] + c[ 2 ] ;
```

7.3 Деклариране и създаване на масиви

- Масивите са обекти от референтен тип
 - Съхраняват се в Heap паметта, където заемат непрекъснатата област
 - Създават се **динамично** с ключовата дума **new**
 - Пример за масив с елементи от примитивен тип

```
int c[] = new int[ 12 ];
```

- Еквивалентно на

```
int c[]; // декларира масив променлива  
c = new int[ 12 ]; // създава масив
```

- Пример за масив с елементи от референтен тип

```
String b[] = new String[ 100 ];
```

Обичайна грешка при програмиране 7.2

Задаването в квадратни скоби на броя на елементите на масив в неговата декларация (т.е., `int c[12] ;`) е синтактична грешка.

Правила за добро програмиране 7.1

За по- добра читаемост на програмите, декларирайте по една променлива на декларация.

Отделяйте всяка декларация на отделен ред и добавяйте коментар след края на декларацията.

7.3 Деклариране и създаване на масиви

- При деклариране на масив може да се групира типа и данните с квадратните скоби с цел да се укаже, че всички променливи в тази декларация са променливи, рефериращи масиви от същия тип данни. Например декларацията

```
double[] array1, array2;
```

Указва, че **array1** и **array2** са “масив от **double**” променливи.

- Горната декларация е еквивалентна на:

```
double array1[];
```

```
double array2[];
```

а също на

```
double[] array1;
```

```
double[] array2;
```

Обичайна грешка при програмиране 7.3

Деклариране на повече от един масив в една декларация може да доведе до трудни за откриване грешки.

Разгледайте декларацията

```
int[] a, b, c;
```

Ако **a**, **b** и **c** би следвало да са масиви, то декларацията е правилна-поставяне на скоби след типа данни указва, че **ВСИЧКИ** от променливите в тази декларация са променливи рефериращи масиви. В случай, че само променливата **a** се иска да реферира масив, а променливите **b** и **c** се иска да са **int** променливи, то тази декларация е неправилна и **би следвало да се използва декларация от вида**

```
int a[], b, c;
```

7.4 Примери за използване на масиви

Основни стъпки в използването на масиви

- Деклариране на масиви
- Създаване на масиви
- Инициализиране елементите на масиви
- Обработка на елементи на масив

7.4 Примери за използване на масиви

- Създаване и инициализиране на масив- **пример**
 - Деклариране на масив
 - Създаване на масив
 - Инициализиране елементите на масиви

Outline

```
1 // Fig. 7.2: InitArray.java
2 // Creating an array.
```

```
3
4 public class InitArray
5 {
6     public static void main( String args[] )
7     {
```

```
8     int array[]; // declare array named array
```

```
9
10    array = new int[ 10 ]; // create the sp
```

```
11
12    System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings
```

```
13
14    // output each array element's value
```

```
15    for ( int counter = 0; counter < array.length; counter++ )
```

```
16        System.out.printf( "%5d%8d\n", counter, array[ counter ] );
```

```
17    } // end main
```

```
18 } // end class InitArray
```

Деклариране **array** като
масив от **int** елементи

Създаване на **10 int** елемента
за **array**; всеки **int** се
инициализира на **0** по подразбиране

array.length връща
дължината на **array**

Всеки **int** елемент на
масива е инициализиран
на **0** по подразбиране

array[counter] връща **int** стойността
на елемента на позиция **index** в масива
array

InitArray.java

Ред 8
Декларира array като
масив от **int**
елементи

Ред 10
Създава 10 **int**
елементи за array;
всеки **int** елемент е
инициализиран на **0** по
подразбиране

Ред 15
array.length връща
дължината на **array**

Ред 16
array[counter] връща
int стойността на
index място в масива
array

Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0



7.4 Примери за използване на масиви

- Използване на инициализиращ списък
 - Списък от стойности, разделени със запетая и ограден с фигурни скоби (**{ }**)

```
int n[] = { 10, 20, 30, 40, 50 };
```
 - Създава и инициализира масив от пет елемента
 - Индексът на масива може да взима стойности
0, 1, 2, 3, 4
 - Няма нужда от ключовата дума **new**

```

1 // Fig. 7.3: InitArray.java
2 // Initializing the elements of an array with an array initializer.
3
4 public class InitArray
5 {
6     public static void main( String args[] )
7     {
8         // initializer list specifies the value for each element
9         int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11         System.out.printf( "%s%s\n", "Index", "value" ); // column headings
12
13         // output each array element's value
14         for ( int counter = 0; counter < array.length; counter++ )
15             System.out.printf( "%5d%8d\n", counter, array[ counter ] );
16     } // end main
17 } // end class InitArray

```

Декларира **array** като
масив от **int** елементи

Outline

InitArray.java

Ред 9
Декларира **array**
като масив от
int елементи

Ред 9
Компилаторът
използва
инициализиращ
списък за да
създаде **array**

Компилаторът **използва**
инициализиращ списък за
да създаде **array**

Program output

Index	value
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37



7.4 Примери за използване на масиви

- **Пресмятане на израз и съхраняване на резултата в елементите на масив**
 - **Пример: Инициализираме елементите на масив с дължина с четни числа и табулиране на стойностите на елементите на масива**

Outline

```
1 // Fig. 7.4: InitArray.java
2 // Calculating values to be placed into elements of an array.
```

```
3
4 public class InitArray
5 {
6     public static void main( String args[] )
7     {
8         final int ARRAY_LENGTH = 10; // declare c
9         int array[] = new int[ ARRAY_LENGTH ]; //
10
11         // calculate value for each array element
12         for ( int counter = 0; counter < array.length; counter++ )
13             array[ counter ] = 2 + 2 * counter;
14
15         System.out.printf( "%s%8s\n", "Index", "value" ); // column headings
16
17         // output each array element's value
18         for ( int counter = 0; counter < array.length; counter++ )
19             System.out.printf( "%5d%8d\n", counter, array[ counter ] );
20     } // end main
21 } // end class InitArray
```

Деклариране на константа `ARRAY_LENGTH`
с използване на `final` атрибут

Деклариране и създаване на
array с 10 int елемента

InitArray.java

Ред 8
Деклариране на
константа

Ред 9
Деклариране и
създаване на
array с 10 int
елемента

Ред 13
Използване на
индекс в array

Използване на
индекс в array за
присвояване на
стойност на елемент
от array

Program output

Index	value
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	16
8	18
9	20



Правила за добро програмиране 7.2

Константите се наричат още **read-only променливи**. Такива променливи правят програмите по-читаеми отколкото използването на конкретни стойности (например, 10). Една **read-only променлива** като **ARRAY_LENGTH** ясно определя целите за използването на ѝ, докато една константна стойност може да има различно значение в зависимост от контекста на използването ѝ.

Обичайна грешка при програмиране 7.4

Повторно присвояване на стойност на `read-only` променлива води до грешка при компилация.

Обичайна грешка при програмиране 7.5

Опит за използване на `read-only` променлива преди инициализирането ѝ води до грешка при компилация.

7.4 Примери за използване на масиви

- **Сумиране на елементите на масив**
 - **Елементите на масив може да представляват един ред от стойности получени по определен начин (пресмятане, наблюдения, статистически)**
 - **Често се налага сумирането на тези стойности (средно аритметично, математическо очакване)**

```

1 // Fig. 7.5: SumArray.java
2 // Computing the sum of the elements of an
3
4 public class SumArray
5 {
6     public static void main( String args[] )
7     {
8         int array[] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
9         int total = 0;
10
11         // add each element's value to total
12         for ( int counter = 0; counter < array.length; counter++ )
13             total += array[ counter ];
14
15         System.out.printf( "Total of array elements: %d\n", total );
16     } // end main
17 } // end class SumArray

```

Деклариране на array с
инициализиращ списък

Outline

SumArray.java

Ред 8
Деклариране на
array с
инициализиращ
списък

Редове 12-13
Сумиране на
елементите на array

Сумиране на елементите на
array

Program output

Total of array elements: 849



7.4 Примери за използване на масиви

- Използване на стълбови диаграми за визуализиране на масив от данни

- Височината на стълбовете определя големината на стойностите в зададен мащаб (пропорционално)
- Пример: Разпределение на брой оценки

Всеки елемент на масива **array** задава брой оценки в интервал от **10**

[0, 9], [10, 19], , , , [80, 89], [90– 99], [100]

Outline

BarChart.java

(1 от 2)

Ред 8

Деклариране на
array с
инициализиращ
списък

Ред 19

Използване на флаг 0 за
извеждане на водещи нули
0 за извеждане на
водещи нули пред
едноцифрени
стойности

Редове 23-24

За всеки елемент на
array, print на
съответен брой '*'



```

1 // Fig. 7.6: BarChart.java
2 // Bar chart printing program.
3
4 public class BarChart
5 {
6     public static void main( String args[] )
7     {
8         int array[] = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };
9
10        System.out.println( "Grade distribution:" );
11
12        // for each array element, output a bar of the chart
13        for ( int counter = 0; counter < array.length; counter++ )
14        {
15            // output bar label ( "00-09: ", ..., "90-99: ", "100: " )
16            if ( counter == 10 )
17                System.out.printf( "%5d: ", 100 );
18            else
19                System.out.printf( "%02d-%02d: ",
20                                counter * 10, counter * 10 + 9 );
21
22            // print bar of asterisks
23            for ( int stars = 0; stars < array[ counter ]; stars++ )
24                System.out.print( "*" );
25
26            System.out.println(); // start a new line of output
27        } // end outer for
28    } // end main
29 } // end class BarChart

```

Деклариране на array с
инициализиращ списък

Използване на флаг 0 за
извеждане на водещи нули
пред едноцифрени стойности

За всеки елемент на array,
print на съответен брой '*'

Outline

BarChart.java

(2 от 2)

Program output

Grade distribution:

```
00-09:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
100: *
```



7.4 Примери за използване на масиви

- Използване на елементите на масив като броячи (акумулиращи стойност променливи)
 - Използване на поредица от променливи за обобщаване на данни (гласуване, проверка за поява на определени събития или признаци)
 - Пример: Честотата, която се падат числата **1– 6** при хвърляне на зарче
 - Елиминира нуждата от използване на **switch** команда!

Outline

RollDie.java

Ред 10
Деклариране на **frequency** като масив от 7 **int** елемента

Редове 13-14
Генериране на **6000** случайни цели числа в интервала 1-6

Ред 14
Нарастване с 1 на елемента от масива **frequency** с индекс, съответстващ на падналото се число

Program output



Деклариране на **frequency** като масив от 7 **int** елемента

Генериране на **6000** случайни цели числа в интервала 1-6

Нарастване с 1 на елемента от масива **frequency** с индекс, съответстващ на падналото се число

```

1 // Fig. 7.7: RollDie.java
2 // Roll a six-sided die 6000 times.
3 import java.util.Random;
4
5 public class RollDie
6 {
7     public static void main( String args[] )
8     {
9         Random randomNumbers = new Random(); // random number generator
10        int frequency[] = new int[ 7 ]; // array of frequency counters
11
12        // roll die 6000 times; use die value as frequency index
13        for ( int roll = 1; roll <= 6000; roll++ )
14            ++frequency[ 1 + randomNumbers.nextInt( 6 ) ];
15
16        System.out.printf( "%s%10s\n", "Face", "Frequency" );
17
18        // output each array element's value
19        for ( int face = 1; face < frequency.length; face++ )
20            System.out.printf( "%4d%10d\n", face, frequency[ face ] );
21    } // end main
22 } // end class RollDie

```

Face	Frequency
1	988
2	963
3	1018
4	1041
5	978
6	1012

7.4 Примери за използване на масиви

- **Използване на масиви за изследване на статистически извадки**
 - **40 студента оценяват качеството на храната**
 - **Въведена е скала 1-10 : 1 означава много лоша, 10 означава отлична**
 - **Получени са 40 анкети и резултатите са въведени в масив от цели елементи**
 - **Обобщаване на резултатите**

Outline

```

1 // Fig. 7.8: StudentPoll.java
2 // Poll analysis program.
3
4 public class StudentPoll
5 {
6     public static void main( String args[] )
7     {
8         // array of survey responses
9         int responses[] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
10             10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5,
11             4, 8, 6, 8, 10 };
12         int frequency[] = new int[ 11 ]; // array of frequency counters
13
14         // for each answer, select responses element and use that value
15         // as frequency index to determine element to increment
16         for ( int answer = 0; answer < responses.length; answer++ )
17             ++frequency[ responses[ answer ] ];
18
19         system.out.printf( "%s%10s", "Rating", "Frequency" );
20
21         // output each array element's value
22         for ( int rating = 1; rating < frequency.length; rating++ )
23             system.out.printf( "%d%10d", rating, frequency[ rating ] );
24     } // end main
25 } // end class StudentPoll

```

Declare responses as array to store 40 responses (1 of 2)

Declare frequency as array of 11 int and ignore the first element

For each response, increment frequency values at index associated with that response

Line 12
Declare frequency as array of 11 int

17 response, increment frequency values at index associated with that response



Outline

StudentPoll.java

(2 of 2)

Program output

Rating	Frequency
--------	-----------

1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3



Съвет за избягване на грешки 7.1

Исключения (exception) получени при изпълнение сигнализират за настъпване на грешка.

Такава грешка от тип изключение (ArrayIndexOutOfBoundsException) настъпва при опит за достъп до елемент на масива извън неговите граници.

Съвет за избягване на грешки 7.2

При използване на цикъл за обработка на елементи на масив, проверявайте дали индексът и винаги по-голям или равен на 0, а също и по-малък от дължината на масива.

Задачи

1. Използвайте едномерен масив за програмиране на приложение на Java, което решава следната задача.
 - Програмата въвежда пет числа, всяко от които е между 10 и 100, включително.
 - При прочитане на числото то се извежда на печат (на текстовия прозорец) само ако това число не е било въведено преди.
 - Отчетете “най- лошия “ вариант при който всички пет числа са различни.
 - Използвайте масив с най- малка възможна дължина за решаване на задачата. Извеждайте на печат пълното множество от всички различни числа, при въвеждане на ново различно от тях число .

Задачи

2. Напишете *class Sample* на Java и методи в този клас, които да реализират следното :
- Извеждат на печат стойността на 7- мия елемент на целочислен масив, подаден като аргумент на метода
 - Инициализират на 8 първите 5 елемента на целочислен масив, подаден като аргумент на метода.
 - Сумират първите 100 елемента на масив с плаваща запетая с двойна точност , подаден като аргумент на метода.
 - Копират първите 11- елемента на целочислен масив *a* в началото на целочислен масив *b* , където масивите *a* и *b* са подадени като аргументи на метода.
 - Определете най- малкия елемент на целочислен масив, подаден като аргумент на метода
 - Пресметнете и върнете след приключване на изпълнението на метода скаларното произведение на два масива с плаваща запетая с двойна точност, подадени като аргументи на метода.
 - Напишете приложение на Java , което тества тези методи на *class Sample* .

Задачи

3. Напишете приложение, което симулира хвърляне на две зарчета. Приложението да използва обект от *class Random* за хвърляне на първото, после и за хвърляне на второто зарче. След хвърляне и двесте зарчета да се сметне сумата на точките, показване от двете зарчета. Всяко от зарчетата показва цяло число от 1 до 6, така че сумата от точките на двете зарчета ще е между 2 до 12, при което 7 се очаква да е най- често получаваната сума, а 2 и 12 най- рядко получаваните суми.
- Фигурата на следващия slide показва 36 възможни комбинации от сумата на точките на двете зарчетата.
 - Вашето приложение да симулира 36,000 хвърляния на двете зарчета. Използвайте едномерен масив за преброяване колко пъти се е паднала всяка от възможните суми на точките от двете зарчета. .

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Задачи

- Изведете резултатите в две колонки , където първата колонка е сумата от точките, а до нея се изписва колко пъти се е паднала тази сума.
- Добавете и каква част (в проценти) от всички хвърляния се пада на тази сума. За справка още изведете очакваното процентно съотношение за да се падне съответната сума (например, от таблицата по- долу се вижда, че има шест начина да се падне сумата 7 , така че очакваното процентно съотношение от всички хвърляния да се падне 7 е $1 / 6$) .