Java

极夜酱

目录

Chapter 1 数组

1.1 一维数组

1.1.1 数组（Array）

一个变量只能存储一个内容，如果需要存储更多数据，就需要使用数组解决问题。一个数组变量可以存放多个数据，数组是一个值的集合，它们共享同一个名字，数组中的每个变量都能被其下标所访问。

```
1 int[] number = new int[10];
2 float[] grade = new float[50];
```

a[0]	a[1]	a[2]	a[3]	a[4]

- 元素：数组中的每个变量
- 大小：数组的容量
- 下标 / 索引（index）：元素的位置，下标从 0 开始，必须为非负整数

1.1.2 数组初始化

一维数组可以在声明时进行初始化：

```
1 int[] arr = {3, 6, 8, 2, 4, 0, 9, 7, 1, 5};
2 int[] arr = new int[] {3, 6, 8, 2, 4, 0, 9, 7, 1, 5};
```

很多时候在使用数组之前需要将数组的内容全部清空，这可以利用循环来实现。

一维数组初始化

```
1 public class InitArr {
```

```
    public static void main(String[] args) {
        int[] arr = new int[100];
        for(int i = 0; i < arr.length; i++) {
            arr[i] = 0;
        }
    }
}
```

数组最大值和最小值

```
public class MaxMin {
    public static void main(String[] args) {
        int[] num = {7, 6, 2, 9, 3, 1, 4, 0, 5, 8};
        int max = num[0];
        int min = num[0];

        for(int i = 1; i < num.length; i++) {
            if(num[i] > max) {
                max = num[i];
            } else if(num[i] < min) {
                min = num[i];
            }
        }

        System.out.println("max = " + max);
        System.out.println("min = " + min);
    }
}
```

运行结果

```
max = 9
min = 0
```

1.1.3 for-each

for-each 环是 for 循环的特殊简化版。

```
for(dataType var : set) {
    // code
}
```

遍历数组

```
public class ForEach {
    public static void main(String[] args) {
        int[] arr = {7, 6, 2, 9, 3, 1, 4, 0, 5, 8};
        for(int elem : arr) {
            System.out.print(elem + " ");
        }
    }
}
```

运行结果

```
7 6 2 9 3 1 4 0 5 8
```

1.2 二维数组

1.2.1 二维数组（2D Array）

二维数组包括行和列两个维度，可以看成是由多个一维数组组成。

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

二维数组可以在声明时进行初始化：

```
int[][] arr = new int[2][3];
int[][] arr = {{1, 2, 3}, {4, 5, 6}};
```

初始化二维数组

```
public class Init2dArr {
    public static void main(String[] args) {
        int[][] arr = new int[3][4];
        for(int i = 0; i < arr.length; i++) {
            for(int j = 0; j < arr[i].length; j++) {
                arr[i][j] = 0;
            }
        }
    }
}
```

矩阵运算

矩阵的加法/减法是指两个矩阵把其相对应元素进行加减的运算。

矩阵加法：两个 $m \times n$ 矩阵 A 和 B 的和，标记为 $A + B$，结果为一个 $m \times n$ 的矩阵，其内的各元素为其相对应元素相加后的值。

矩阵减法：两个 $m \times n$ 矩阵 A 和 B 的差，标记为 $A - B$，结果为一个 $m \times n$ 的矩阵，其内的各元素为其相对应元素相减后的值。

$$\begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1-0 & 3-0 \\ 1-7 & 0-5 \\ 1-2 & 2-1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -6 & -5 \\ -1 & 1 \end{bmatrix}$$

```java
public class Matrix {
    public static void main(String[] args) {
        int[][] A = {
            {1, 3},
            {1, 0},
            {1, 2}
        };
        int[][] B = {
            {0, 0},
            {7, 5},
            {2, 1}
        };
        int[][] C = new int[3][2];

        System.out.println("矩阵加法");
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 2; j++) {
                C[i][j] = A[i][j] + B[i][j];
                System.out.print(String.format("%3d", C[i][j]));
            }
            System.out.println();
        }

```

```
24          System.out.println("矩阵减法");
25          for(int i = 0; i < 3; i++) {
26              for(int j = 0; j < 2; j++) {
27                  C[i][j] = A[i][j] - B[i][j];
28                  System.out.print(String.format("%3d", C[i][j]));
29              }
30              System.out.println();
31          }
32      }
33  }
```

运行结果

```
矩阵加法
1  3
8  5
3  3
矩阵减法
1  3
-6 -5
-1  1
```

1.3 字符

1.3.1 字符（Character）

单个的字符是一种特殊的类型，是用单引号表示字符字面量。每一个字符都有其对应的码值。

ASCII 全称 American Standard Code for Information Interchange（美国信息交换标准代码），一共定义了 128 个字符。

ASCII	字符	ASCII	字符	ASCII	字符	ASCII	字符
0	NUT	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	’	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u

22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	/	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

表 1.1: ASCII 码表

ASCII 码

```
1 public class ASCII {
2     public static void main(String[] args) {
3         for(int i = 0; i < 128; i++) {
4             System.out.println(String.format("%c = %d", i, i));
5         }
6     }
7 }
```

1.4 字符串

1.4.1 字符串（String）

字符串是用双引号所表示的 0 个或多个字符的组合。字符串变量使用 String 表示，String 是一个类，String 的变量是对象的管理者而非所有者。

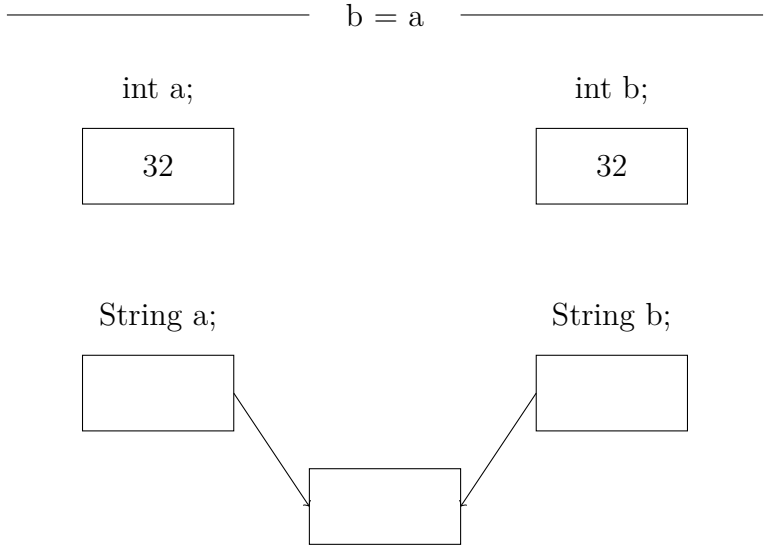

图 1.1: 字符串引用

通过调用 Scanner 类中的 nextLine() 可以获取用户输入的字符串。

创建字符串对象

```
1  import java.util.Scanner;
2
3  public class StringObj {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          System.out.print("输入字符串： ");
7          String str = scanner.nextLine();
8          System.out.println(str);
9          scanner.close();
10     }
11 }
```

运行结果

```
输入字符串：Hello World!
Hello World!
```

1.4.2 字符串比较

字符串的比较分为两种：

1. 【==】运算符用于比较是否是同一个对象。
2. equals() 用于比较字符串的内容是否相同。

字符串比较

```java
public class StringEqual {
    public static void main(String[] args) {
        String s1 = new String("hello");
        String s2 = new String("hello");

        if(s1 == s2) {
            System.out.println("s1和s2是同一个对象");
        }

        if(s1.equals(s2)) {
            System.out.println("s1与s2内容相同");
        }

        if(s1.equalsIgnoreCase(s2)) {
            System.out.println("s1与s2忽略大小写内容相同");
        }
    }
}
```

运行结果

```
s1与s2内容相同
s1与s2忽略大小写内容相同
```

1.4.3 字符串操作

字符串是对象，它包含了一系列的常用操作，对它的所有操作都是通过【.】运算符进行的。

length()：计算字符串长度

计算字符串长度

```
1 public class StringLength {
2     public static void main(String[] args) {
3         String str = "Hello World!";
4         System.out.println(str.length());
5     }
6 }
```

运行结果

```
12
```

charAt()：访问字符串中的字符

字符串中的每一个下标位置都是一个单个的字符，下标的范围从 0 到 length() - 1。

访问字符串中的字符

```
1 public class CharAt {
2     public static void main(String[] args) {
```

```
        String str = "Hello World!";
        System.out.println(str.charAt(4));
    }
}
```

运行结果

```
o
```

计算字符串中某个字符出现的次数

```
import java.util.Scanner;

public class CountOccurence {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int cnt = 0;        // 出现次数

        System.out.print("输入字符串：");
        String str = scanner.nextLine();
        System.out.print("输入待统计字符：");
        char c = scanner.nextLine().charAt(0);

        int n = str.length();
        for(int i = 0; i < n; i++) {
            if(str.charAt(i) == c) {
                cnt++;
            }
        }

        System.out.println(c + "在" + str + "中出现了" + cnt + "次");
        scanner.close();
    }
}
```

运行结果

```
输入字符串：Hello World
输入待统计字符：l
l在Hello World中出现了3次
```

substring()：获取子串

- substring(n)：获取第 n 个位置到末尾的全部内容。
- substring(begin, end)：获取从 begin 到 end 位置之前的内容。

获取子串

```
1 public class Substring {
2     public static void main(String[] args) {
3         String str = "Hello World!";
4         System.out.println(str.substring(6));
5         System.out.println(str.substring(3, 10));
6     }
7 }
```

运行结果

```
World!
lo Worl
```

indexOf()：查找

- indexOf(c)：获取字符 c 所在的位置，返回-1 表示不存在。
- indexOf(c, n)：从第 n 个位置开始查找字符 c。
- indexOf(t)：获取字符串 t 所在的位置。

查找

```java
public class IndexOf {
    public static void main(String[] args) {
        String str = "Hello World!";
        System.out.println(str.indexOf('o'));
        System.out.println(str.indexOf('l', 4));
        System.out.println(str.indexOf("llo"));
    }
}
```

运行结果

```
4
9
2
```

大小写转换

- toLowerCase()：将字符串转换为小写。
- toUpperCase()：将字符串转换为大写。

大小写转换

```java
public class UpperLowerCase {
    public static void main(String[] args) {
        String str = "Hello World!";
        System.out.println(str.toLowerCase());
        System.out.println(str.toUpperCase());
    }
}
```

运行结果

```
hello world
HELLO WORLD
```

替换

- replace(c1, c2)：将所有字符 c1 替换为字符 c2，返回新字符串。
- replace(s1, s2)：将所有子串 s1 替换为子串 s2，返回新字符串。

字符串替换

```
1 public class Replace {
2     public static void main(String[] args) {
3         String str = "Hello World!";
4         System.out.println(str.replace('l', '*'));
5         System.out.println(str.replace("ll", "##"));
6     }
7 }
```

运行结果

```
He**o Wor*d!
He##o World!
```

split(regex)：字符串分割

根据匹配给定的正则表达式拆分字符串，返回拆分后的字符串数组。

字符串替换

```
1 public class Split {
2     public static void main(String[] args) {
3         String str = "This is a string.";
4         String[] s = str.split(" ");
5         for(String item : s) {
6             System.out.println(item);
7         }
8     }
9 }
```

运行结果

```
This
is
a
string.
```

统计单词个数

```java
import java.util.Scanner;

public class CountWord {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("输入英语句子： ");
        String str = scanner.nextLine();
        // "\\s+"表示一个或多个空格、回车、制表符等空白符
        String[] words = str.split("\\s+");
        System.out.println("单词个数： " + words.length);
        for(String word : words) {
            System.out.println("\t" + word);
        }
        scanner.close();
    }
}
```

运行结果

```
输入英语句子: This is a string.
单词个数: 4
This
is
a
string.
```