Лабораторная работа №2. Основы и простейшие команды языка R

Контрольные вопросы и задания

Задание 1 Перечислить по памяти зарезервированные слова языка R. Сколько их всего? Объяснить смысл NULL, Inf, NA, NaN. Получить эти значения несколькими способами в результате выполнения каких-либо операций.

Ответ Зарезервированные слова: `if`, `else`, `repeat`, `while`, `function`, `for`, `in`, `next`, `break`, `TRUE`, `FALSE`, `NULL`, `Inf`, `NaN`, `NA`, `NA_integer_`, `NA_real_`, `NA_complex_`, `NA_character_`

`NULL` - Пустое значение

```
> empty <- list()
> empty[1]
[[1]]
NULL
```

`Inf` - Бесконечность

```
> 1 / 0
[1] Inf
> -1 / 0
[1] -Inf
```

`NA` - Отсутствующее значение

```
> a <- 1:10
> a[11]
[1] NA
```

`NaN` - Нечисловое значние

```
> 0 * Inf
[1] NaN
> Inf / Inf
[1] NaN
> sqrt(-10)
[1] NaN
```

Задание 2 Создать десять переменных разных типов, проверить их тип с помощью функции typeof(). Проверить с помощью этой функции типы следующих констант: 29, 23i, -34L, 2/3, 4/2, 0xA, 0XbL – 120L, 0XbL – 120, 0XbL * 17. Объяснить полученные результаты.

Ответ

```
> variables <- list(1:3, letters, 1.2, complex(),
+                   numeric(5), integer(5), NaN,
+                   NA, NULL, FALSE, list(1))
> for (i in variables) {
+ var <- i
+ print(typeof(var))
+ }
[1] "integer"
[1] "character"
[1] "double"
[1] "complex"
[1] "double"
[1] "integer"
[1] "double"
[1] "logical"
[1] "NULL"
[1] "logical"
[1] "list"

> consts <- list(29, 23i, -34L, 2/3, 4/2, 0xA,
+                0XbL - 120L, 0XbL - 120, 0XbL * 17)
> for (i in consts) {
+     cat(i, ': ', typeof(i), '\n')
+ }
29 : double
0+23i : complex
-34 : integer
0.6666667 : double
2 : double
10 : double
-109 : integer
-109 : double
187 : double
```

Для выполнения некоторых операций производятся неявные преобразования. Например, `0XbL - 120` integer 0XbL приводится к double и ответ получаем в double.

Задание 3 Проверить работу арифметических операторов и приоритет выполнения операций в сложных выражениях. Выполнить серию из 10 вычислений по различным самостоятельно придуманным формулам. Использовать все операторы из таблицы 1.

Выяснить правило, по которому выполняется расчет, если в формуле используется целочисленное деление и остаток от деления. Убедиться, что значения выражений дают предсказуемый вами результат.

Ответ

```
library(stringi)

a <- 7
b <- 3
c <- 5
d <- 10

print(stri_c("(a + b) * c) = ", (a + b) * c))
print(stri_c("(a + b) * (c - d)) = ", (a + b) * (c - d)))
print(stri_c("a / b * c = ", d / c * 7))
print(stri_c("a / b * c ^ d = ", a / b * c ^ d))
print(stri_c("d * a %% b = ", d * a %% b))
print(stri_c("(d * a) %% b = ", (d * a) %% b))
print(stri_c("d * a %/% b = ", d * a %/% b))
print(stri_c("(d * a) %/% b = ", (d * a) %/% b))
print(stri_c("(d * a) ^ 2 %/% b = ", (d * a) ^ 2 %/% b))
print(stri_c("((d * a) ^ 2) %/% b = ", ((d * a) ^ 2) %/% b))
```

```
[1] "(a + b) * c) = 50"
[1] "(a + b) * (c - d)) = -50"
[1] "a / b * c = 14"
[1] "a / b * c ^ d = 22786458.3333333"
[1] "d * a %% b = 10"
[1] "(d * a) %% b = 1"
[1] "d * a %/% b = 20"
[1] "(d * a) %/% b = 23"
[1] "(d * a) ^ 2 %/% b = 1633"
[1] "((d * a) ^ 2) %/% b = 1633"
```

Действие	Оператор	Приоритет
Возведение в степень	$\wedge$	4
Остаток от деления	%%	3
Целочисленное деление	%/%	3
Деление	/	2
Умножение	$\times$	2
Вычитание	-	1
Сложение	+	1

Задание 4 Исследовать правила неявного преобразования типов, выполнив примеры, аналогичные примерам из п.8. Определить правила преобразования для переменных следующих типов:

integer и double
integer и logical
logical и character
double и logical

double и character

По результатам исследования сформулировать общее правило преобразования типов в R.

Ответ Приведение типов в R происходит слева направо по цепочке:

logical – integer – double – character

Следовательно имеем следующие преобразования:

```
integer и double     ->  double
integer и logical    ->  integer
logical и character  ->  character
double и logical     ->  double
double и character   ->  character
```

Задание 5 Исследовать различие в работе логических операторов «Поэлементное И» и «Сокращенное И», для чего подготовить примеры, аналогичные примерам из п.9.

Ответ «Поэлементное И» (&) объединяет каждый элемент первого вектора с соответствующим элементом второго вектора (или подставляет их) и выдает TRUE, если оба элемента имеют значение TRUE:

```
> q1 <- c(1, 2, 5, 4, 7, 8, 4)
> q2 <- c(1:6, 4)
> q3 <- (q1 == 4) & (q2 == 4)
> print(q3)
[1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
```

«Сокращенное И» (&&) принимает первый элемент обоих векторов и дает TRUE, только если оба (или условия) TRUE:

```
> q1 <- c(1, 2, 3, 4, 7)
> q2 <- c(1, 4, 3, 2)
> q3 <- (q1 == 1) && (q2 == 1)
> print(q3)
[1] TRUE
```

Задание 6 Последовательно выполнить операции:

3/7

3/7 – 0.4285714

Объяснить полученный результат.

Ответ

```
> 3/7
[1] 0.4285714
> 3/7 - 0.4285714
[1] 2.857143e-08
```

Результат деления 3 на 7 примерно на 2.857143×10^{-8} больше числа 0.4285714. При этом при выводе на экран R по умолчания оставляет только 7 знаков после запятой.

Посмотреть число с большей точность можно так:

```
> format(round(3/7, 20), nsmall=20)
[1] "0.42857142857142854764"
```

Задание 7 Последовательно выполнить операции:

sqrt(2)*sqrt(2)

(sqrt(2)*sqrt(2))-2

Объяснить полученный результат.

Ответ

```
> sqrt(2)*sqrt(2)
[1] 2
> (sqrt(2)*sqrt(2))-2
[1] 4.440892e-16
```

Floating-point arithmetic. Так считает компьютер. Чтобы избежать такого можно использовать числа с фиксированной точкой.