Задания

Задача 1:

В переменной `income` сохранено значение дохода компании за последний месяц в рублях.

1. Создайте переменную `income` и присвойте ей любое (желательно разумное) значение.
2. Создайте переменную `log_income`, в которой содержится натуральный логарифм значения дохода.
3. Создайте переменную `income_pre` и присвойте ей значение 500000. Напишите строку кода, которая позволит узнать, в каком месяце, текущем или предыдущем, доход компании был больше.

Работа программы:

```
> task_1()
Доход этого месяца: 500001
В этом месяце доход был больше на  1
>
```

```
> task_1()
Доход этого месяца: 500000
[1] "Доход одинаков"
>
```

```
Доход этого месяца: 3000
В предыдущем месяце доход был больше на  497000
>
```

Листинг:

```
task_1 <- function()
{
 income <-  as.numeric(readline("Доход этого месяца: "))
 #log_income
 income_pre <- 500000
 if(income_pre>income)
 {
  cat("В предыдущем месяце доход был больше на ", (income_pre-income))
 }else if (income_pre<income)
 {
  cat("В этом месяце доход был больше на ", (income-income_pre))
 }else
 {
  print("Доход одинаков")
 }
}
```

Задача 2:

В двух переменных сохранены некоторые значения:

```
x <- 2
y <- 4
```

Напишите код, который позволит поменять значения в переменных `x` и `y` местами, то есть получить следующее:

```
x
## [1] 4
y
## [1] 2
```

- *Внимание:* Ваш код должен работать для любых значений `x`, `y`. Создавать дополнительные переменные можно!

Работа программы:

```
> task_2()
x: 4
y: 2
[1] 2
[1] 4
>
```

```
> task_2()
x: 6
y: 7
[1] 7
[1] 6
>
```

Листинг:

```
task_2 <- function()
{
 x <- as.numeric(readline("x: "))
 y <- as.numeric(readline("y: "))
 x_exch <- y
 y_exch <- x
 y <- y_exch
 x <- x_exch
 print(x)
 print(y)
}
```

Задача 3:

Даны следующие данные

```
x <- 3.5
y <- "2,6"
z <- 1.78
h <- TRUE
```

Определите типы переменных.

Работа программы:

```
> task_3()
[1] "logical"
>
```

Листинг:

```
task_3 <- function()
{
 x <- 3.5    # numeric
 y <- "2,6"  # character
 z <- 1.78   # numeric
 h <- TRUE   # logical
 class(x);
 class(y);
```

```
 class(z);
 class(h);
}
```

Задача 4:

Создайте вектор `q`, состоящий из следующих значений: 4, 7, -1, 21, 2, 0, 14.

- Создайте вектор `q_sq`, состоящий из квадратов значений вектора `q`.
- Создайте вектор `q_log`, состоящий из натуральных логарифмов значений вектора `q`. Напишите (просто комментарием), почему в векторе `q_log` есть пропущенные значения. Возможно, Вам потребуется почитать про логарифмы (Википедии хватит).
- Выведите на экран неотрицательные значения вектора `q`.
- Выведите на экран индексы элементов вектора `q`, которые кратны 7.

Выведите на экран элементы вектора `q_log`, которые кратны 2 и больше 5.

Работа программы:

```
> task_4()
[1]  16  49   1 441   4   0 196
[1] 1.3862944 1.9459101       NaN 3.0445224 0.6931472      -Inf 2.6390573
[1]  4  7 21  2  0 14
[1] NA NA
Предупреждение:
В task_4() : созданы NaN
>
```

некоторые значения пустые

так как никакое число в степени не дает отрицательно и 0

Листинг:

```
task_4 <- function()
{
 q <- c(4, 7, -1, 21, 2, 0, 14)
 q_sq <- q^2
 print(q_sq)
 q_log <- log(q,exp(1))
 print(q_log)
 print(q[q >= 0])
 which(q%%7 == 0)
 print(q_log[q_log %%2 == 0] & q_log[q_log > 5])
}
```

Задача 5:

Политолог Мебейн (Walter R. Mebane) считает, что большая доля избирательных участков со значениями явки, заканчивающихся на 0 или 5, свидетельствует о фальсификациях результатов выборов. Аргументирует он это чисто психологическими причинами: если значения явки сочиняют люди, то они более склонны записывать круглые числа и числа, кратные 5.

Перед Вами вектор значений явки на избирательных участках в районе F страны Флатландии:

```
turnout <- c(100, 124, 121, 130, 150, 155, 144, 132, 189, 145, 125, 110, 118, 129, 127)
```

1. Выведите на экран индексы избирательных участков, где явка, согласно Мебейну, выглядит подозрительной (значения явки, кратные 10 или 5).
2. Определите долю таких подозрительных участков, выразите ее в процентах и округлите ответ до второго знака после запятой.

Работа программы:

```
> task_5()
[1] 7
[1] 15
>
```

Листинг:

```
task_5 <- function()
{
 turnout <- c(100, 124, 121, 130, 150, 155, 144, 132, 189, 145, 125, 110, 118, 129, 127)
 which((turnout %%5 == 0) | (turnout %%10 == 0))
 index <- length(turnout[turnout %%5 == 0])
 print(index)
 dole <- length(turnout)
 print(dole)
 part <- round((index/dole)*100, 2)
}
```

Задача 6:

Дан вектор `z`:

```
z <- c(8, NA, 7, 10, NA, 15, NA, 0, NA, NA, 87)
```

Выведите на экран индексы элементов вектора `z`, которые являются пропущенными значениями.

Работа программы:

```
> task_6()
[1]  2  5  7  9 10
>
```

Листинг:

```
task_6 <- function()
{
 z <- c(8, NA, 7, 10, NA, 15, NA, 0, NA, NA, 87)
 which(is.na(z))
}
```

Задача 7:

Дан вектор `s`:

```
s <- c("4,5", "6,8", "9,2", "1,75")
```

Получите, основываясь на векторе `s`, числовой вектор `n` (тип `numeric`).

Работа программы:

```
> task_7()
4.5 6.8 9.2 1.75
>
```

Листинг:

```
task_7 <- function()
{
 s <- c("4,5", "6,8", "9,2", "1,75")
 n <- as.numeric(gsub(",", ".", s))
 cat(n)
}
```

Задача 8:

Решите систему уравнений

$$\begin{pmatrix} 1 & 1 \\ 50 & 75 \end{pmatrix} x = \begin{pmatrix} 100 \\ 6625 \end{pmatrix} \quad (1.1)$$

Примените к системе (1.1) первую трансформацию Гаусса и решите полученную систему уравнений.

Задание: для матрицы

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 7 & 6 & 9 \\ 3 & 6 & 3 & 8 \\ 4 & 9 & 8 & 2 \end{pmatrix}$$

вычислите

- обратную матрицу A^{-1} (`solve`) и транспонированную матрицу $A^{\top}$ (`t`);
- след матрицы (сумму диагональных элементов) $\mathrm{tr}\,(A)$ (`diag`);
- определитель матрицы $\det(A)$;
- алгебраическое дополнение к элементу $A_{2,3}$: взятый с обратным знаком определитель матрицы A, у которой удалены вторая строка и третий столбец (`A[-2,-3]` или `A[c(1,3,4),c(1,2,4)]`).

Работа программы:

```
> task_8()
Решение системы уравнений
     [,1]
[1,]   35
[2,]   65
Обратная матрица
           [,1]        [,2]        [,3]        [,4]
[1,]  0.6809524 -0.70476190  0.42380952  0.11428571
[2,] -0.7047619  0.32380952 -0.01904762  0.02857143
[3,]  0.4238095 -0.01904762 -0.20476190  0.05714286
[4,]  0.1142857  0.02857143  0.05714286 -0.08571429
Транспонированная матрица
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    2    7    6    9
[3,]    3    6    3    8
[4,]    4    9    8    2
Сумма диагональных элементов
[1] 13
Определитель матрицы
[1] 210
Алгебраическое дополнение к элементу A[2,3]
[1] -4
>
```

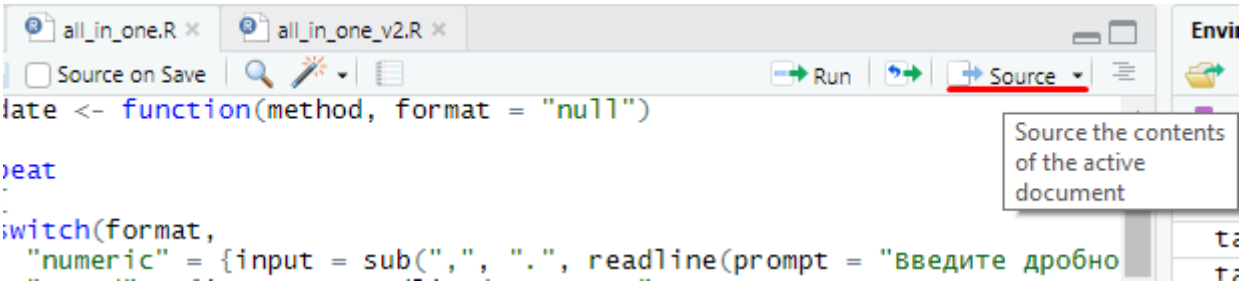
Листинг:

```
task_8 <- function()
{
 Y <- c(1, 50, 1, 75);
 dim(Y) <- c(2, 2);
 Z <- c(100, 6625);
```

```
  dim(Z) <- c(2, 1);
  cat("Решение системы уравнений\n")
  print(solve(Y, Z))
  A <- c(1, 2, 3, 4, 2, 7, 6, 9, 3, 6, 3, 8, 4, 9, 8, 2);
  dim(A) <- c(4, 4);
  cat("Обратная матрица\n")
  print(solve(A))
  cat("Транспонированная матрица\n")
  print(t(A))
  cat("Сумма диагональных элементов\n")
  print(sum(diag(A)))
  cat("Определитель матрицы\n")
  print(det(A))
  Ax <- c(1, 3, 4, 2, 6, 9, 4, 8, 2);
  dim(Ax) <- c(3, 3);
  cat("Алгебраическое дополнение к элементу A[2,3]\n")
  print(det(-Ax))
}
```

Способы запуска:

1) запуск всего кода, чтобы инициализировать функции



2) запуск заданий:
   task_[1]()
   [1] – номер задания