Чтение и запись данных

Вывод данных на экран

Самый простой способ вывести на экран значение переменной или математического выражения – это написать имя переменной или выражения **без точки с запятой** после них.

Пример. Если переменной x присвоить значение 1, то после выполнения такой строки кода на экран выведется значение переменной x:

[ ]:
```
x = 1
```

Т.к. переменная x уже определена, то, просто написав имя этой переменной, вы получите ее значение:

[ ]:
```
x
```

Точно так же на экран выводятся результаты выполнения математических действий:

[ ]:
```
2 + 3
```

[ ]:
```
2 > 1
```

Если после выражения поставить **точку с запятой**, то вывод на экран значения этого выражения будет заблокирован:

[ ]:
```
x = 2;
```

[ ]:
```
4*5;
```

Если в кодовой ячейке содержится несколько выражений, то после выполнения этой ячейки на экран будет выведен результат выполнения только **последнего** выражения. Например, после выполнения следующей ячейки будет выведено значение только переменой `t`:

[ ]:
```
x = 1
y = 2
z = 3; t = 4
```

Если вам нужно вывести данные на экран в произвольном месте кодовой ячейки, то это можно следать с помощью следующих функций:

- `print` – вывод на экран значений переменных, констант, математических или логических выражений. Если нужно вывести несколько значений (`x, y, z`), то они перечисляются через запятую: `print(x, y, z);`
- `println` – то же, что и `print`, но с переходом на новую строку, т.е. следующий вывод будет производиться в новой строке;
- `display` – то же, что и `print`, но у этой функции может быть только один аргумент (константа, переменная или выражение).

- `show` – то же, что и `display`.

Ниже приведены несколько примеров использования функции `print`:

[ ]:
```
print(2*3+5)
```

[ ]:
```
print(true)
```

[ ]:
```
print('A')
```

[ ]:
```
X = 10
print("X = ", X)
```

[ ]:
```
X = [11 12 13]
print("X = ", X)
```

[ ]:
```
print(1>2)
```

Использование функции `println` ничем не отличается от функции `print`.

✏Задание 1

Задайте значение переменной x, равное 5. Затем вычислите значения переменных y и z по формулам: $y = (2x + 1)/7$ и $z = \sin(y)$. После этого выведите на экран значения всех трех переменных x, y и z с помощью функции `println`.

[ ]:
```

```

Подсказка Чтобы при выводе на экран значений нескольких переменных было понятно, что означает каждое из выводимых значений, рекомендуем записывать функцию `println` с двумя аргументами: первый аргумент – это строковая константа, содержащая имя переменной со знаком равенства после нее, а второй аргумент – сама переменная. Например, так: `println("x =  x)`.

Решение

[ ]:
```
x = 5
y = (2x + 1)/7
z = sin(y)
println("x = ", x)
println("y = ", y)
println("z = ", z)
```

Ниже приведены несколько примеров использования функции `display`:

```
[ ]: display("Hello, world!")
```

```
[ ]: display(22/7)
```

```
[ ]: X = 10
display(X)
```

Несколько примеров использования функции `show`:

```
[ ]: show("This is my text")
```

```
[ ]: show(13/17)
```

```
[ ]: a = 3.5
show(a)
```

Функция `summary` возвращает тип переменной. Для массивов возвращается также их размер.

Пример. Выведем на экран информацию о матрице `x`:

```
[ ]: x = [1 2 3; 4 5 6]
summary(x)
```

Форматированный вывод

Вы можете управлять точностью отображения переменных с помощью функции `@printf`. Эта функция будет доступна, если подключить библиотеку `Printf`.

Пример. Выведем на экран значение переменной `x = 12.67835675` с точностью до 4 знаков после запятой:

```
[ ]: using Printf
x = 12.67835675
@printf("%.4f", x)
```

Выражение в кавычках `"%.4f"` задает формат вывода переменной: `.4` означает число знаков после десятичной запятой, буква `f` задает тип данных, который имеет переменная `x`. В данном случае `f` – вещественный тип; также возможны `i` – целый тип, `e` – экспоненциальная запись, `s` – строковый тип и др. В качестве второго аргумента функции `@printf` указывается имя переменной `x`, значение которой нужно вывести на экран. Полную информацию о формате вывода с помощью функции `@printf` вы можете посмотреть в документации.

✏Задание 2

Выведите значение переменной `x = pi^3` с точностью до 6 знаков после запятой.

```
[ ]:
```

Решение

```
[ ]: using Printf
x = pi^3
@printf("%.6f", x)
```

Ввод данных с клавиатуры

Ввести данные с клавиатуры можно с помощью функции `readline()`. Рассмотрим два случая:

1. Если нужно ввести переменную x строкового типа, то используется следующая конструкция:

```
x = readline()
```

2. Если нужно ввести переменную простого числового типа (целого, вещественного или логического), то используется такая запись:

```
x = parse(Type, readline())
```

Функция `readline` предназначена для ввода данных только строкового типа. Чтобы преобразовать введенную переменную в числовой тип, используется функция `parse`. Ее первый аргумент Type задает тип переменной (`Int`, `Float64`, `Float32`, `Bool` и т.д.), в который нужно преобразовать строковую переменную, заданную вторым аргументом`readline()`.

✏Задание 3

Введите с клавиатуры значение переменной x строкового типа. Для этого наберите следующую команду:

```
x = readline()
```

Ниже ячейки с вашим кодом появится поле для ввода значения с клавиатуры. Введите в нем слово Hello (символы вводятся без кавычек!) и нажмите клавишу Enter. На экране будет отображено введенное вами значение переменной x.

```
[ ]:
```

Решение

```
x = readline()

stdin>  Hello
"Hello"
```

✏Задание 4

Введите с клавиатуры значение вещественной переменной x, равное 1.5. Для этого наберите следующую команду:

```
x = parse(Float64, readline())
```

С ее помощью будет выполняться ввод с клавиатуры значения переменной x и преобразование ее в вещественный тип.

Затем в появившемся поле введите число 1.5 и нажмите клавишу Enter.

Следующей командой возведите в квадрат переменную x.

[ ]:

Решение

```
x = parse(Float64, readline())

stdin>  1.5
1.5

x^2

2.25
```

Чтение и запись файлов

Работа с текстовыми файлами

Язык **Julia** предоставляет возможности для чтения данных из файла и для записи в файл.

Чтобы начать работу с файлом, сначала его необходимо открыть. Для этого используется функция `open(filename)`, где `filename` – имя физического файла в локальном окружении **Engee**, взятое в двойные кавычки. Значение функции open присваивается переменной типа `IOStream` (поток ввода-вывода).

Пример. Из текущей папки откроем текстовый файл “data.txt”:

[ ]:
```
f = open("data.txt")
```

Для чтения всех строк из **текстового** файла используется функция `readlines`, для чтения одной строки – функция `readline`. Аргументом этих функций служит переменная типа `IOStream`.

Пример. Прочитаем все строки из файла `data.txt` и присвоим их переменной x:

[ ]:
```
x = readlines(f)
```

Этот файл содержит три строки. Они считываются в вектор x, состоящий из элементов строкового типа. Каждый элемент вектора представляет собой отдельную строку текста.

Пример. Прочитаем одну строку из файла `data.txt`:

[ ]:
```
f = open("data.txt")
x = readline(f)
```

В приведенных выше примерах функция open открывала файл, расположенный в текущей папке (в той же папке, в которой находится данный интерактивный скрипт **04. Чтение и запись данных.ngscript**). Если вы хотите открыть файл, расположенный в другой папке, то в качестве аргумента функции нужно написать полный путь к файлу вместе с его именем. Например, к тому же самому файлу **data.txt** можно обратиться с использованием полного пути:

[ ]:
```
f = open("/user/start/Курсы/Курс «Основы программирования в Engee»/
  ↪data.txt")
```

Обратите внимание на то, что в локальном окружении **Engee** корневая папка называется **/user**, а учебные курсы расположены в папках **/user/start/Курсы/Название курса**.

Julia позволяет сменить текущую папку с помощью команды cd. Например:

[ ]:
```
cd("/user/start/Курсы/Курс «Основы программирования в Engee»")
```

После этого можно обращаться к файлам из этой папки только по имени, без указания пути:

[ ]:
```
f = open("data.txt")
```

✏Задание 5

Откройте текстовый файл **information.txt**. С помощью функции `readline` прочитайте текст из этого файла и выведите его на экран.

[ ]:
```

```

Решение

[ ]:
```
f = open("information.txt")
s = readline(f)
```

Для того чтобы открыть файл для записи данных, используется функция `open(filename, "w")`, где второй агрумент `w` указывает на то, что файл открывается в режиме создания. При этом будет создан новый файл с именем `filename`.

Пример. Создадим и откроем для записи файл **my_file.txt**.

```
[ ]: f = open("my_file.txt","w")
```

После выполнения этой строки кода во вкладке **Файлы** вы увидите новый файл **my_file.txt**.

В таблице перечислены возможные режимы открытия файлов:

Режим	Описание
r	чтение
w	чтение, создание, обрезка
a	чтение, создание, присоединение
r+	чтение, запись
w+	чтение, запись, создание, обрезка
a+	чтение, запись, создание, присоединение

Для записи данных в файл используется функция `write`.

Пример. Запишем в файл строку **This is my text**. Для перехода на новую строку в конце строковой константы добавлено \n.

```
[ ]: write(f, "This is my text\n")
```

Добавим в файл вторую строку с текстом **This is another text**.

```
[ ]: write(f, "This is another text\n")
```

Если нужно записать в файл числовые данные, то их нужно преобразовать в строковый тип с помощью функции `string`.

```
[ ]: write(f, string(8734.354))
```

Если вы просмотрите содержимое файла **my_file.txt** на этом этапе, то вы увидите, что файл пустой. Для того чтобы данные фактически записались, файл должен быть закрыт. Файл закрывается с помощью функции `close`:

```
[ ]: close(f)
```

После выполнения этой команды файл будет содержать три строки с данными.

Для удаления файла используется функция `rm(filename)`.

Пример. Удалим созданный нами файл **my_file.txt**:

```
[ ]: rm("my_file.txt")
```

✏Задание 6

Создайте файл **task.txt** и запишите в него текст **Я программирую на языке Julia**. Просмотрите файл. Затем удалите этот файл с помощью функции `rm`.

[ ]:
```

```

Решение

[ ]:
```
f = open("task.txt","w")
write(f, "Я программирую на языке Julia")
close(f)
```

[ ]:
```
rm("task.txt")
```

Работа с числовыми файлами

Для работы с числовыми данными, оформленными в виде таблиц, широко используются файлы в формате csv. Для доступа к функциям, позволяющим работать с табличными данными, необходимо подключить библиотеки CSV и DataFrames. Функция `CSV.read(filename, DataFrame)` позволяет прочитать данные из файла в переменную табличного типа DataFrame.

Пример. Прочитаем данные из файла **data_table.csv**:

[ ]:
```
using CSV, DataFrames
data = CSV.read("data_table.csv", DataFrame)
```

Мы видим, что каждый столбец таблицы имеет свое имя: Jan, Feb, Mar и Apr. Извлечь столбец из таблицы DataFrame можно при помощи обращения через точку: `таблица.Имя_Переменной`. Например, так:

[ ]:
```
x = data.Jan
```

[ ]:
```
y = data.Feb
```

Если нужно получить не весь столбец, а некоторые его элементы, то можно воспользоваться такой конструкцией:

[ ]:
```
z = data[1, "Mar"]
```

Здесь переменной z присваивается значение первого элемента в столбце Mar таблицы data.

Если нужно получить несколько элементов, идущих подряд в каком-либо столбце, то первый индекс записывается в виде диапазона: `начало:конец`. Например, присвоим переменной t элементы со 2-го по 4-й в столбце Apr таблицы data:

[ ]:
```
t = data[2:4, "Apr"]
```

✏Задание 7

Присвойте переменной v третий элемент столбца Jan таблицы data, а переменной w – со второго по пятый элементы элементы столбца Feb этой же таблицы.

[ ]:

Решение

[ ]:
```
v = data[3, "Jan"]
```

[ ]:
```
w = data[2:5, "Feb"]
```

Больше о работе с данными в **Julia** с помощью библиотеки `DataFrames` можно прочитать в документации.

Работа с графическими файлами

Для работы с графическими файлами необходимо подключить библиотеки `Images` и `ImageShow`. Открыть файл с изображением можно с помощью функции `load(filename)`, а просмотреть изображение – с помощью функции `simshow`.

Пример. Откроем и просмотрим графический файл **image.jpg**:

[ ]:
```
using Images, ImageShow
A = load("image.jpg")
simshow(A)
```

✏Задание 8

Откройте и просмотрите графический файл **landscape.jpg**.

[ ]:

Решение

[ ]:
```
using Images, ImageShow
A = load("landscape.jpg")
simshow(A)
```

Больше о функциях ввода/вывода и работы с файлами в **Julia** вы можете прочитать в документации.

Тест для получения сертификата

Пройти тест по теме **Чтение и запись данных**.