7.2.1.1. Считывание данных

In [1]:
```
# Установка пакетов:
#using Pkg
#for p in ["CSV", "DataFrames", "RDatasets", "FileIO", "DataArrays"]
#    Pkg.add(p)
#end
```

In [2]:
```
using CSV, DataFrames, DelimitedFiles
```

In [3]:
```
#P = download("https://raw.githubusercontent.com/nassarhuda/easy_data/master/progr
```

In [4]:
```
# Считывание данных и их запись в структуру:
P = CSV.File("programminglanguages.csv") |> DataFrame
```

Out[4]: 73 rows × 2 columns

| | year | language |
|---|---|---|
| | Int64 | String |
| 1 | 1951 | Regional Assembly Language |
| 2 | 1952 | Autocode |
| 3 | 1954 | IPL |
| 4 | 1955 | FLOW-MATIC |
| 5 | 1957 | FORTRAN |
| 6 | 1957 | COMTRAN |
| 7 | 1958 | LISP |
| 8 | 1958 | ALGOL 58 |
| 9 | 1959 | FACT |
| 10 | 1959 | COBOL |
| 11 | 1959 | RPG |
| 12 | 1962 | APL |
| 13 | 1962 | Simula |
| 14 | 1962 | SNOBOL |
| 15 | 1963 | CPL |
| 16 | 1964 | Speakeasy |
| 17 | 1964 | BASIC |
| 18 | 1964 | PL/I |
| 19 | 1966 | JOSS |
| 20 | 1967 | BCPL |
| 21 | 1968 | Logo |
| 22 | 1969 | B |
| 23 | 1970 | Pascal |
| 24 | 1970 | Forth |
| 25 | 1972 | C |
| 26 | 1972 | Smalltalk |
| 27 | 1972 | Prolog |
| 28 | 1973 | ML |
| 29 | 1975 | Scheme |
| 30 | 1978 | SQL |
| ⋮ | ⋮ | ⋮ |

In [5]:
```
# Функция определения по названию языка программирования года его создания:
function language_created_year(P,language::String)
    loc = findfirst(P[:,2].==language)
```

```
    return P[loc,1]
end
```

Out[5]:
```
language_created_year (generic function with 1 method)
```

In [6]:
```
# Пример вызова функции и определение даты создания языка Python:
language_created_year(P,"Python")
```

Out[6]:
```
1991
```

In [7]:
```
# Пример вызова функции и определение даты создания языка Julia:
language_created_year(P,"Julia")
```

Out[7]:
```
2012
```

In [8]:
```
#В следующем примере при вызове функции, в качестве аргумента которой указано
#слово julia, написанное со строчной буквы:
language_created_year(P,"julia")
#поэтому выходит ошибка
```

```
MethodError: no method matching getindex(::DataFrame, ::Nothing, ::Int64)
Closest candidates are:
  getindex(::DataFrame, !Matched::Colon, ::Union{AbstractString, Signed, Symbol, U
nsigned}) at C:\Users\Admin\.julia\packages\DataFrames\GtZ1l\src\dataframe\datafra
me.jl:420
  getindex(::DataFrame, !Matched::typeof(!), ::Union{Signed, Unsigned}) at C:\User
s\Admin\.julia\packages\DataFrames\GtZ1l\src\dataframe\dataframe.jl:426
  getindex(::DataFrame, !Matched::Integer, ::Union{Signed, Unsigned}) at C:\Users
\Admin\.julia\packages\DataFrames\GtZ1l\src\dataframe\dataframe.jl:383
  ...

Stacktrace:
 [1] language_created_year(::DataFrame, ::String) at .\In[5]:4
 [2] top-level scope at In[8]:3
 [3] include_string(::Function, ::Module, ::String, ::String) at .\loading.jl:1091
 [4] execute_code(::String, ::String) at C:\Users\Admin\.julia\packages\IJulia\rWZ
9e\src\execute_request.jl:27
 [5] execute_request(::ZMQ.Socket, ::IJulia.Msg) at C:\Users\Admin\.julia\packages
\IJulia\rWZ9e\src\execute_request.jl:86
 [6] #invokelatest#1 at .\essentials.jl:710 [inlined]
 [7] invokelatest at .\essentials.jl:709 [inlined]
 [8] eventloop(::ZMQ.Socket) at C:\Users\Admin\.julia\packages\IJulia\rWZ9e\src\ev
entloop.jl:8
 [9] (::IJulia.var"#15#18")() at .\task.jl:356
```

In [9]:
```
# Функция определения по названию языка программирования
# года его создания (без учёта регистра):
function language_created_year_v2(P,language::String)
    loc = findfirst(lowercase.(P[:,2]).==lowercase.(language))
    return P[loc,1]
end
language_created_year_v2(P,"julia")
```

Out[9]:
```
2012
```

In [10]:
```
# Пример вызова функции и определение даты создания языка julia:
language_created_year_v2(P,"julia")
```

Out[10]:
```
2012
```

In [11]:
```
# Построчное считывание данных с указанием разделителя:
Tx = readdlm("programminglanguages.csv", ',')
```

Out[11]:
```
74×2 Array{Any,2}:
     "year"  "language"
 1951        "Regional Assembly Language"
 1952        "Autocode"
 1954        "IPL"
 1955        "FLOW-MATIC"
 1957        "FORTRAN"
 1957        "COMTRAN"
 1958        "LISP"
 1958        "ALGOL 58"
 1959        "FACT"
 1959        "COBOL"
 1959        "RPG"
 1962        "APL"
    ⋮
 2003        "Scala"
 2005        "F#"
 2006        "PowerShell"
 2007        "Clojure"
 2009        "Go"
 2010        "Rust"
 2011        "Dart"
 2011        "Kotlin"
 2011        "Red"
 2011        "Elixir"
 2012        "Julia"
 2014        "Swift"
```

7.2.1.2. Запись данных в файл

In [12]:
```
# Запись данных в CSV-файл:
CSV.write("programming_languages_data2.csv", P)
```

Out[12]:
```
"programming_languages_data2.csv"
```

In [13]:
```
# Пример записи данных в текстовый файл с разделителем ',':
writedlm("programming_languages_data.txt", Tx, ',')
```

In [14]:
```
# Пример записи данных в текстовый файл с разделителем '-':
writedlm("programming_languages_data2.txt", Tx, '-')
```

In [15]:
```
# Построчное считывание данных с указанием разделителя:
P_new_delim = readdlm("programming_languages_data2.txt", '-')
```

Out[15]:
```
74×2 Array{Any,2}:
     "year"  "language"
 1951        "Regional Assembly Language"
 1952        "Autocode"
 1954        "IPL"
 1955        "FLOW-MATIC"
 1957        "FORTRAN"
 1957        "COMTRAN"
 1958        "LISP"
 1958        "ALGOL 58"
 1959        "FACT"
 1959        "COBOL"
 1959        "RPG"
 1962        "APL"
    ⋮
 2003        "Scala"
 2005        "F#"
 2006        "PowerShell"
 2007        "Clojure"
 2009        "Go"
 2010        "Rust"
 2011        "Dart"
 2011        "Kotlin"
 2011        "Red"
 2011        "Elixir"
 2012        "Julia"
 2014        "Swift"
```

7.2.1.3. Словари

In [16]:
```
# Инициализация словаря:
dict = Dict{Integer,Vector{String}}()
```

Out[16]:
```
Dict{Integer,Array{String,1}}()
```

In [17]:
```
# Инициализация словаря:
dict2 = Dict()
```

Out[17]:
```
Dict{Any,Any}()
```

In [18]:
```
# Заполнение словаря данными:
for i = 1:size(P,1)
    year,lang = P[i,:]

    if year in keys(dict)
        dict[year] = push!(dict[year],lang)
    else
        dict[year] = [lang]
    end
end
```

In [19]:
```
# Пример определения в словаре языков программирования, созданных в 2003 году:
dict[2003]
```

Out[19]:
```
2-element Array{String,1}:
 "Groovy"
 "Scala"
```

7.2.1.4. DataFrames

In [20]:
```
# Подгружаем пакет DataFrames:
using DataFrames
```

In [21]:
```
# Задаём переменную со структурой DataFrame:
df = DataFrame(year = P[:,1], language = P[:,2])
```

Out[21]: 73 rows × 2 columns

| | year | language |
|---|---|---|
| | Int64 | String |
| 1 | 1951 | Regional Assembly Language |
| 2 | 1952 | Autocode |
| 3 | 1954 | IPL |
| 4 | 1955 | FLOW-MATIC |
| 5 | 1957 | FORTRAN |
| 6 | 1957 | COMTRAN |
| 7 | 1958 | LISP |
| 8 | 1958 | ALGOL 58 |
| 9 | 1959 | FACT |
| 10 | 1959 | COBOL |
| 11 | 1959 | RPG |
| 12 | 1962 | APL |
| 13 | 1962 | Simula |
| 14 | 1962 | SNOBOL |
| 15 | 1963 | CPL |
| 16 | 1964 | Speakeasy |
| 17 | 1964 | BASIC |
| 18 | 1964 | PL/I |
| 19 | 1966 | JOSS |
| 20 | 1967 | BCPL |
| 21 | 1968 | Logo |
| 22 | 1969 | B |
| 23 | 1970 | Pascal |
| 24 | 1970 | Forth |
| 25 | 1972 | C |
| 26 | 1972 | Smalltalk |
| 27 | 1972 | Prolog |
| 28 | 1973 | ML |
| 29 | 1975 | Scheme |
| 30 | 1978 | SQL |
| ⋮ | ⋮ | ⋮ |

In [22]:
```
# Вывод всех значения столбца year:
df[!,:year]
```

Out[22]:
```
73-element Array{Int64,1}:
 1951
 1952
 1954
 1955
 1957
 1957
 1958
 1958
 1959
 1959
 1959
 1962
 1962
    ⋮
 2003
 2005
 2006
 2007
 2009
 2010
 2011
 2011
 2011
 2011
 2012
 2014
```

In [23]:
```
# Получение статистических сведений о фрейме:
describe(df)
```

Out[23]: 2 rows × 8 columns

| | variable | mean | min | median | max | nunique | nmissing | eltype |
|---|---|---|---|---|---|---|---|---|
| | Symbol | Union... | Any | Union... | Any | Union... | Nothing | DataType |
| **1** | year | 1982.99 | 1951 | 1986.0 | 2014 | | | Int64 |
| **2** | language | | ALGOL 58 | | dBase III | 73 | | String |

7.2.1.5. RDatasets

In [24]:
```
# Подгружаем пакет RDatasets:
using RDatasets
```

In [25]:
```
# Задаём структуру данных в виде набора данных:
iris = dataset("datasets", "iris")
```

Out[25]: 150 rows × 5 columns

| | SepalLength | SepalWidth | PetalLength | PetalWidth | Species |
|---|---|---|---|---|---|
| | Float64 | Float64 | Float64 | Float64 | Cat... |
| **1** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **2** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **3** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **4** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **5** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| **6** | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| **7** | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| **8** | 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| **9** | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| **10** | 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| **11** | 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| **12** | 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| **13** | 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| **14** | 4.3 | 3.0 | 1.1 | 0.1 | setosa |
| **15** | 5.8 | 4.0 | 1.2 | 0.2 | setosa |
| **16** | 5.7 | 4.4 | 1.5 | 0.4 | setosa |
| **17** | 5.4 | 3.9 | 1.3 | 0.4 | setosa |
| **18** | 5.1 | 3.5 | 1.4 | 0.3 | setosa |
| **19** | 5.7 | 3.8 | 1.7 | 0.3 | setosa |
| **20** | 5.1 | 3.8 | 1.5 | 0.3 | setosa |
| **21** | 5.4 | 3.4 | 1.7 | 0.2 | setosa |
| **22** | 5.1 | 3.7 | 1.5 | 0.4 | setosa |
| **23** | 4.6 | 3.6 | 1.0 | 0.2 | setosa |
| **24** | 5.1 | 3.3 | 1.7 | 0.5 | setosa |
| **25** | 4.8 | 3.4 | 1.9 | 0.2 | setosa |
| **26** | 5.0 | 3.0 | 1.6 | 0.2 | setosa |
| **27** | 5.0 | 3.4 | 1.6 | 0.4 | setosa |
| **28** | 5.2 | 3.5 | 1.5 | 0.2 | setosa |
| **29** | 5.2 | 3.4 | 1.4 | 0.2 | setosa |
| **30** | 4.7 | 3.2 | 1.6 | 0.2 | setosa |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In [26]:
```
# Определения типа переменной:
typeof(iris)
```

Out[26]:
```
DataFrame
```

In [27]:
```
#Пакет RDatasets также предоставляет возможность с помощью description получить
#основные статистические сведения о каждом столбце в наборе данных:
describe(iris)
```

Out[27]: 5 rows × 8 columns

| | variable | mean | min | median | max | nunique | nmissing | eltype |
|---|---|---|---|---|---|---|---|---|
| | Symbol | Union... | Any | Union... | Any | Union... | Nothing | DataType |
| 1 | SepalLength | 5.84333 | 4.3 | 5.8 | 7.9 | | | Float64 |
| 2 | SepalWidth | 3.05733 | 2.0 | 3.0 | 4.4 | | | Float64 |
| 3 | PetalLength | 3.758 | 1.0 | 4.35 | 6.9 | | | Float64 |
| 4 | PetalWidth | 1.19933 | 0.1 | 1.3 | 2.5 | | | Float64 |
| 5 | Species | | setosa | | virginica | 3 | | CategoricalValue{String,UInt8 |

7.2.1.6. Работа с переменными отсутствующего типа (Missing Values)

In [28]:
```
# Отсутствующий тип:
a = missing
typeof(a)
```

Out[28]:
```
Missing
```

In [29]:
```
# Пример операции с переменной отсутствующего типа:
a + 1
```

Out[29]:
```
missing
```

In [30]:
```
# Определение перечня продуктов:
foods = ["apple", "cucumber", "tomato", "banana"]
```

Out[30]:
```
4-element Array{String,1}:
 "apple"
 "cucumber"
 "tomato"
 "banana"
```

In [31]:
```
# Определение калорий:
calories = [missing,47,22,105]
```

Out[31]:
```
4-element Array{Union{Missing, Int64},1}:
    missing
 47
 22
105
```

In [32]:
```
# Определение типа переменной:
typeof(calories)
```

Out[32]:
```
Array{Union{Missing, Int64},1}
```

In [33]:
```
# Подключаем пакет Statistics:
using Statistics
```

In [34]:
```
# Определение среднего значения:
mean(calories)
```

Out[34]:
```
missing
```

In [35]:
```
# Определение среднего значения без значений с отсутствующим типом:
mean(skipmissing(calories))
```

Out[35]:
```
58.0
```

In [36]:
```
# Задание сведений о ценах:
prices = [0.85,1.6,0.8,0.6]
```

Out[36]:
```
4-element Array{Float64,1}:
 0.85
 1.6
 0.8
 0.6
```

In [37]:
```
# Формирование данных о калориях:
dataframe_calories = DataFrame(item=foods,calories=calories)
```

Out[37]: 4 rows × 2 columns

| | item | calories |
|---|---|---|
| | String | Int64? |
| 1 | apple | *missing* |
| 2 | cucumber | 47 |
| 3 | tomato | 22 |
| 4 | banana | 105 |

In [38]:
```
# Формирование данных о ценах:
dataframe_prices = DataFrame(item=foods,price=prices)
```

Out[38]: 4 rows × 2 columns

| | item | price |
|---|---|---|
| | String | Float64 |
| 1 | apple | 0.85 |
| 2 | cucumber | 1.6 |
| 3 | tomato | 0.8 |
| 4 | banana | 0.6 |

In [39]:
```
# Объединение данных о калориях и ценах:
DF = join(dataframe_calories,dataframe_prices,on=:item)
```

Out[39]: 4 rows × 3 columns

| | item | calories | price |
|---|---|---|---|
| | String | Int64? | Float64 |
| 1 | apple | *missing* | 0.85 |
| 2 | cucumber | 47 | 1.6 |
| 3 | tomato | 22 | 0.8 |
| 4 | banana | 105 | 0.6 |

7.2.1.7. FileIO

In [40]:
```
# Подключаем пакет FileIO:
using FileIO
```

In [41]:
```
julialogo = download("https://avatars0.githubusercontent.com/u/743164?s=200&v=4","
```

Out[41]: "julialogo.png"

In [42]:
```
# Подключаем пакет ImageIO:
import Pkg
Pkg.add("ImageIO")
```

```
  Updating registry at `C:\Users\Admin\.julia\registries\General`
 Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [43]:
```
# Загрузка изображения:
X1 = load("julialogo.png")
```

Out[43]:
```
200×200 Array{RGBA{N0f8},2} with eltype ColorTypes.RGBA{FixedPointNumbers.Normed{U
Int8,8}}:
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 ⋮                            ⋱
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)  …  RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
 RGBA{N0f8}(0.0,0.0,0.0,0.0)     RGBA{N0f8}(0.0,0.0,0.0,0.0)
```

In [44]:
```
# Определение типа и размера данных:
@show typeof(X1);
@show size(X1);
```

```
typeof(X1) = Array{ColorTypes.RGBA{FixedPointNumbers.Normed{UInt8,8}},2}
size(X1) = (200, 200)
```

In [ ]:
```

```