**Приложение руководства разработчика к программе, содержащей информацию о известных футболистах**

Разработчики:

Петин Д.

Рыльников А.

Яценко И.

## Листинг interface.py

# -\*- coding: utf-8 -\*-

import sys # модули sys и os для функции перезагрузки программы

import os

sys.path.append('../library')

import base as b # модуль со стандартной базой

import functions as f # модуль с функциями

import tkinter as t # модуль tkinter для создания графического интерфейса

from tkinter import messagebox as mb # messagebox из tkinter для сообщений пользователю

import tkinter.filedialog as fd # tkinter.filedialog для диалоговых окон

import tkinter.colorchooser as cl # tkinter.colorchooser для цветового селектора

import pickle as pk # модуль pickle

par = pk.load(open('parametrs.pic', mode='rb')) # загрузка параметров из файла

def print\_base(fra, database):

"""

Входные параметры - fra - фрейм для записи базы данных, database - база

Выходных параметров нет

Вывод базы на экран

Автор: Петин Д.М.

"""

par = pk.load(open('parametrs.pic', mode='rb'))

k = 0

for i in database.keys():

k = k+1

if k == 0:

bas = t.Label(fra, text=' No items in database', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], '40', 'bold'))

bas.grid(row=0, column=0, sticky=t.NSEW)

else:

# Столбик с номерами

bas = t.Label(fra, text=' Number', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=0, sticky=t.NSEW)

for i in range(len(database)):

bas = t.Label(fra, text=' %s ' % (i + 1), relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=i + 1, column=0, sticky=t.NSEW)

# Названия столбцов

bas = t.Label(fra, text=' Key ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=1, sticky=t.NSEW)

for l in database.keys():

for i, j in enumerate(database[l]):

bas = t.Label(fra, text=' %s ' % j, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=i + 2, sticky=t.NSEW)

break

bas = t.Label(fra, text=' ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=10, columnspan=2, sticky=t.NSEW)

# Вывод данных

lis = [2, 3, 4, 5, 6, 7, 8, 9]

j = 1

val\_sal = 0

val\_aver = 0

sam\_v = 0

sam\_t = 0

av\_ag = 0

sam\_a = 0

for k in database.keys():

bas = t.Label(fra, text=' %s ' % k, relief=t.GROOVE, bg=par['table\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size']))

bas.grid(row=j, column=1, sticky=t.NSEW)

for i, item in zip(lis, database[k].values()):

bas = t.Label(fra, text=' %s ' % item, relief=t.GROOVE, bg=par['table\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size']))

bas.grid(row=j, column=i, sticky=t.NSEW)

# ПОДВЕДЕНИЕ ИТОГОВ

if -1 not in database.keys():

av\_ag = av\_ag + int(database[k]['Age'])

val\_aver = val\_aver + float(database[k]['Transfer Value'])

val\_sal = val\_sal + float(database[k]['Salary'])

deleti = t.Button(fra, text="X", width=1, height=1, bg='red',

fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=lambda db=database, k=k: delet(db, k))

deleti.grid(row=j, column=10, sticky=t.NSEW)

editi = t.Button(fra, text="Edit", width=6, height=1, bg='orange',

fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'),

command=lambda db=database, ke=k, n=j: edit(db, ke, n))

editi.grid(row=j, column=11, sticky=t.NSEW)

j += 1

if -1 not in database.keys():

numbr = ' ' + str(len(database)) + ' '

val\_aver = float(val\_aver) / float(numbr)

val\_sal = float(val\_sal) / float(numbr)

av\_ag = float(av\_ag) / float(numbr)

for k in database.keys():

sam\_v = sam\_v + (float(database[k]['Transfer Value']) - val\_aver)\*\*2

sam\_t = sam\_t + (float(database[k]['Salary']) - val\_sal)\*\*2

sam\_a = sam\_a + (int(database[k]['Age']) - av\_ag) \*\* 2

tot = t.Label(dele\_f, text=' \nTotals\n ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

tot.grid(row=0, column=2, columnspan=2, sticky=t.NSEW)

# КОЛ-ВО ЭЛЕМЕНТОВ

ni = t.Label(dele\_f, text=' Number of items: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

ni.grid(row=1, column=2, sticky=t.NSEW)

ni = t.Label(dele\_f, text='%s' % numbr, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

ni.grid(row=1, column=3, sticky=t.NSEW)

sam\_v = float(sam\_v)/float(numbr)

sam\_t = float(sam\_t) / float(numbr)

sam\_a = float(sam\_a) / float(numbr)

# СРЕДНИЙ ВОЗРАСТ

ava = t.Label(dele\_f, text=' Average age: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

ava.grid(row=2, column=2, sticky=t.NSEW)

ava = t.Label(dele\_f, text='%.2f' % av\_ag, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

ava.grid(row=2, column=3, sticky=t.NSEW)

# ВЫБОРОЧНАЯ ДИСПЕРСИЯ ВОЗРАСТА

sampvar = t.Label(dele\_f, text=' Sample variance of \n'

'age: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sampvar.grid(row=3, column=2, sticky=t.NSEW)

sampvar = t.Label(dele\_f, text='%.2f' % sam\_a, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sampvar.grid(row=3, column=3, sticky=t.NSEW)

# СРЕДНЯЯ ТРАНСФЕРНАЯ ЦЕНА

avt = t.Label(dele\_f, text=' Average transfer \nvalue: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

avt.grid(row=4, column=2, sticky=t.NSEW)

avt = t.Label(dele\_f, text='%.2f' % val\_aver, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

avt.grid(row=4, column=3, sticky=t.NSEW)

# ВЫБОРОЧНАЯ ДИСПЕРСИЯ ТРАНСФЕРНОЙ ЦЕНЫ

sampt = t.Label(dele\_f, text=' Sample variance of \n'

'transfer value: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sampt.grid(row=5, column=2,rowspan=2, sticky=t.NSEW)

sampt = t.Label(dele\_f, text='%.2f' % sam\_v, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sampt.grid(row=5, column=3, rowspan=2, sticky=t.NSEW)

# СРЕДНЯЯ ЗАРПЛАТА

avs = t.Label(dele\_f, text=' Average salary: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

avs.grid(row=7, column=2, sticky=t.NSEW)

avs = t.Label(dele\_f, text='%.2f' % val\_sal, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

avs.grid(row=7, column=3, sticky=t.NSEW)

# ВЫБОРОЧНАЯ ДИСПЕРСИЯ ЗАРПЛАТЫ

sams = t.Label(dele\_f, text=' Sample variance of \n'

'salary: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sams.grid(row=8, column=2,rowspan=2, sticky=t.NSEW)

sams = t.Label(dele\_f, text='%.2f' % sam\_t, relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

sams.grid(row=8, column=3, rowspan=2, sticky=t.NSEW)

ad = t.Label(dele\_f, text='', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

ad.grid(row=10, column=2, columnspan=2, sticky=t.NSEW)

# ЗАПИСЬ В ФАЙЛ

par = pk.load(open('parametrs.pic', mode='rb'))

with open(par['way\_to\_total'], "a") as file:

file.write('Total :\n\n'

'Number of items: ' + str(numbr) + '\n' +

'Average age :' + '%.2f' % av\_ag + '\n' +

'Sample variance of age: :' + '%.2f' % sam\_a + '\n' +

'Average transfer value :' + '%.2f' % val\_aver + '\n' +

'Sample variance of transfer value :' + '%.2f' % sam\_v + '\n' +

'Average salary :' + '%.2f' % val\_sal + '\n' +

'Sample variance of salary :' + '%.2f' % sam\_t + '\n\n')

def print\_butt(\*args):

"""

Входных параметров нет

Выходных параметров нет

Вызов функции печати базы для кнопки интерфейса

Автор: Яценко И.Ю.

"""

print\_base(basafr, database)

def clear\_base(fra):

"""

Входные параметры - fra - фрейм для для удаления

Выходных параметров нет

Очистка фрейма от базы

Автор: Рыльников А.М.

"""

for bas in fra.grid\_slaves():

bas.grid\_forget()

def load(\*args):

"""

Входные параметров нет

Выходных параметров нет

Вызов функции загрзуки базы из библиотеки для кнопки интерфейса

Автор: Яценко И.Ю.

"""

global database

fi = fd.askopenfilename()

if fi == '' or fi[-3:] != 'pic':

mb.showerror("Error",

"File error\nChoose file\nYou can choose only 'pic' file")

else:

par['way\_to\_base'] = fi

pk.dump(par, open('parametrs.pic', mode='wb'))

database = f.load\_file()

prnt.config(state="normal")

clr.config(state="normal")

defa.config(state="normal")

ad.config(state="normal")

sett.config(state="normal")

fltr.config(state="normal")

def save\_b(\*args):

"""

Входные параметров нет

Выходных параметров нет

Вызов функции сохранения базы из библиотеки для кнопки интерфейса

Автор: Рыльников А.М.

"""

f.save\_file(database)

def delet(db,k):

"""

Входные db и k

Выходных параметров нет

Удаление элемента по ключю для кнопки интерфейса

Автор: Яценко И.Ю.

"""

f.delete(db, k)

f.save\_file(db)

database = f.load\_file()

clear\_base(basafr)

print\_base(basafr, database)

def defaults(\*args):

"""

Входные параметров нет

Выходных параметров нет

Возвращение начальных параметров

Автор: Яценко И.Ю

"""

b.defa()

database = f.load\_file()

clear\_base(basafr)

print\_base(basafr, database)

f.save\_file(database)

def add(\*args):

"""

Входные параметров нет

Выходных параметров нет

Добавление элемента для кнопки интерфейса

Автор: Рыльников А.М.

"""

name = nam.get()

nation = nat.get()

age = age1.get()

club = clu.get()

position = pos.get()

value = val.get()

date = exp.get()

salary = sal.get()

s = f.is\_float(value)

s1 = f.is\_float(salary)

s2 = f.is\_int(age)

# проверка полученных данных на соответсвие

if s == False or s1 == False or position == '' or date == '' or s2 == False or float(salary) <0 or float(value) < 0 or int(age) < 0 or age > 50 or age < 16 or value <0 or salary <0:

mb.showerror("Error", "Input error\nAge must be integer from 16 to 50, Salary and value must be floats more than 0\nChoose position and date")

else:

database = f.load\_file()

f.addition(database, name, nation, age, club, position, value, date, salary)

f.save\_file(database)

database = f.load\_file()

clear\_base(basafr)

print\_base(basafr, database)

def edit(database,ke, n):

"""

Входные - database, k

Выходных параметров нет

Редактирование по ключю для кнопки интерфейса

Автор: Рыльников А.М.

"""

name1 = t.Entry(basafr, width=10, bd=8)

name1.grid(row=n, column=2, sticky=t.NSEW)

name1.insert(0, database[ke]['Name'])

nat = t.Entry(basafr, width=10, bd=8)

nat.grid(row=n, column=3, sticky=t.NSEW)

nat.insert(0, database[ke]['Nationality'])

ag = t.Entry(basafr, width=10, bd=8)

ag.grid(row=n, column=4, sticky=t.NSEW)

ag.insert(0, database[ke]['Age'])

clubi = t.Entry(basafr, width=10, bd=8)

clubi.grid(row=n, column=5, sticky=t.NSEW)

clubi.insert(0, database[ke]['Club'])

posi = t.StringVar(basafr)

posi.set(database[ke]['Position'])

posit = t.OptionMenu(basafr, posi, 'Forward', 'Winger', 'Attacking Midfielder', 'Defending Midfielder',

'Left Defender', 'Center Defender', 'Right Defender', 'Goalkeeper')

posit.grid(row=n, column=6, sticky=t.NSEW)

tv = t.Entry(basafr, width=10, bd=8)

tv.grid(row=n, column=7, sticky=t.NSEW)

tv.insert(0, database[ke]['Transfer Value'])

da = t.StringVar(basafr)

da.set(database[ke]['Contract expiry date'])

dax = t.OptionMenu(basafr, da, '30.06.2018', '30.06.2019', '30.06.2020', '30.06.2021',

'30.06.2022', '30.06.2023', '30.06.2024', '30.06.2026', '30.06.2027', '30.06.2028', '30.06.2029')

dax.grid(row=n, column=8, sticky=t.NSEW)

sa = t.Entry(basafr, width=10, bd=8)

sa.grid(row=n, column=9, sticky=t.NSEW)

sa.insert(0, database[ke]['Salary'])

savi = t.Button(basafr, text="Save", width=6, height=1, bg='light blue',

fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'),

command=lambda database=database,k=ke: sav(database, k))

savi.grid(row=n, column=11, sticky=t.NSEW)

def sav(database,k):

name = name1.get()

nation = nat.get()

age = ag.get()

club = clubi.get()

position = posi.get()

value = tv.get()

contract = da.get()

salary = sa.get()

a1 = f.is\_int(age)

v1 = f.is\_float(value)

s1 = f.is\_float(salary)

if (a1 == False or int(age)<16 or int(age)>50 or v1 == False or float(value)<0.1 or float(value)>1000 or s1 == False or float(salary)<0.1 or float(salary)>1000):

mb.showerror("Error",

"Input error\nAge must be integer from 16 to 50\nTransfer Value and Salary must be floats from 0.1 to 1000\nChoose column")

else:

f.edition(database, k, name, nation, age, club, position, value, contract, salary)

f.save\_file(database)

database = f.load\_file()

clear\_base(basafr)

print\_base(basafr, database)

def serch(\*args):

"""

Входные параметров нет

Выходных параметров нет

Поиск элементов для кнопки интерфейса

Автор: Петин Д.М.

"""

def check\_s(v):

"""

Входной ппарметр - s

Возвращает True или False

Проверка на наличие ввода

Автор: Петин Д.М.

"""

if v != '':

return True

else:

return False

# имя

nam\_v = nama.get()

nc = check\_s(nam\_v)

# страна

nat\_v = nata.get()

natc = check\_s(nat\_v)

# возраст

age\_v1 = a11.get()

age\_v2 = a12.get()

a1\_c = check\_s(age\_v1)

a2\_c = check\_s(age\_v2)

ag\_c1 = f.is\_int(age\_v1)

ag\_c2 = f.is\_int(age\_v2)

# клуб

cl\_v = cl3.get()

c2\_c = check\_s(cl\_v)

# клуб

ps\_v = pos12.get()

p2\_c = check\_s(ps\_v)

# стоимость

tv\_v1 = t11.get()

tv\_v2 = t12.get()

t1\_c = check\_s(tv\_v1)

t2\_c = check\_s(tv\_v2)

tv\_c1 = f.is\_float(tv\_v1)

tv\_c2 = f.is\_float(tv\_v2)

# дата

d\_v1 = d11.get()

d\_v2 = d12.get()

d1\_c = check\_s(d\_v1)

d2\_c = check\_s(d\_v2)

# зарплата

s\_v1 = sl11.get()

s\_v2 = sl12.get()

s1\_c = check\_s(s\_v1)

s2\_c = check\_s(s\_v2)

s\_c1 = f.is\_float(s\_v1)

s\_c2 = f.is\_float(s\_v2)

if (a1\_c == True and a2\_c == True and age\_v1 > age\_v2) or (t1\_c == True and t2\_c == True and tv\_v1 > tv\_v2) \

or (d1\_c == True and d2\_c == True and d\_v1 > d\_v2) or (s1\_c == True and s2\_c == True and s\_v1 > s\_v2)\

or s\_c1 == False or s\_c2 == False or tv\_c1 == False or tv\_c2 == False or ag\_c1 == False or ag\_c2 == False:

mb.showerror("Error",

"Input error\nAge must be integer, Salary and Value must be floats\nValue 2 must be more than or Value 1")

elif nc == False and s2\_c == False and s1\_c == False and d2\_c == False and d1\_c == False and t1\_c == False and t2\_c == False and p2\_c == False and c2\_c == False and a1\_c == False and a2\_c == False and natc == False:

mb.showerror("Error",

"Nothing chosen")

else:

database = f.load\_file()

db1 = db2 = db3 = db4 = db5 = db6 = db7 = db8 = False

db1 = f.search(database, 'Name', nam\_v)

if db1 != False:

db2 = f.search(db1, 'Nationality', nat\_v)

else:

db2 = f.search(database, 'Nationality', nat\_v)

if db2 != False:

db3 = f.search\_num(db2, 'Age', age\_v1, age\_v2)

elif db1 != False:

db3 = f.search\_num(db1, 'Age', age\_v1, age\_v2)

else:

db3 = f.search\_num(database, 'Age', age\_v1, age\_v2)

if db3 != False:

db4 = f.search(db3, 'Club', cl\_v)

elif db2 != False:

db4 = f.search(db2, 'Club', cl\_v)

elif db1 != False:

db4 = f.search(db1, 'Club', cl\_v)

else:

db4 = f.search(database, 'Club', cl\_v)

if db4 != False:

db5 = f.search(db4, 'Position', ps\_v)

elif db3 != False:

db5 = f.search(db3, 'Position', ps\_v)

elif db2 != False:

db5 = f.search(db2, 'Position', ps\_v)

elif db1 != False:

db5 = f.search(db1, 'Position', ps\_v)

else:

db5 = f.search(database, 'Position', ps\_v)

if db5 != False:

db6 = f.search\_num(db5, 'Transfer Value', tv\_v1, tv\_v2)

elif db4 != False:

db6 = f.search\_num(db4, 'Transfer Value', tv\_v1, tv\_v2)

elif db3 != False:

db6 = f.search\_num(db3, 'Transfer Value', tv\_v1, tv\_v2)

elif db2 != False:

db6 = f.search\_num(db2, 'Transfer Value', tv\_v1, tv\_v2)

elif db1 != False:

db6 = f.search\_num(db1, 'Transfer Value', tv\_v1, tv\_v2)

else:

db6 = f.search\_num(database, 'Transfer Value', tv\_v1, tv\_v2)

if db6 != False:

db7 = f.search\_date(db6, 'Contract expiry date', d\_v1, d\_v2)

elif db5 != False:

db7 = f.search\_date(db5, 'Contract expiry date', d\_v1, d\_v2)

elif db4 != False:

db7 = f.search\_date(db4, 'Contract expiry date', d\_v1, d\_v2)

elif db3 != False:

db7 = f.search\_date(db3, 'Contract expiry date', d\_v1, d\_v2)

elif db2 != False:

db7 = f.search\_date(db2, 'Contract expiry date', d\_v1, d\_v2)

elif db1 != False:

db7 = f.search\_date(db1, 'Contract expiry date', d\_v1, d\_v2)

else:

db7 = f.search\_date(database, 'Contract expiry date', d\_v1, d\_v2)

if db7 != False:

db8 = f.search\_num(db7, 'Salary', s\_v1, s\_v2)

elif db6 != False:

db8 = f.search\_num(db6, 'Salary', s\_v1, s\_v2)

elif db5 != False:

db8 = f.search\_num(db5, 'Salary', s\_v1, s\_v2)

elif db4 != False:

db8 = f.search\_num(db4, 'Salary', s\_v1, s\_v2)

elif db3 != False:

db8 = f.search\_num(db3, 'Salary', s\_v1, s\_v2)

elif db2 != False:

db8 = f.search\_num(db2, 'Salary', s\_v1, s\_v2)

elif db1 != False:

db8 = f.search\_num(db1, 'Salary', s\_v1, s\_v2)

else:

db8 = f.search\_num(database, 'Salary', s\_v1, s\_v2)

db = {}

if db8 is not False:

db = db8

elif db7 is not False:

db = db7

elif db6 is not False:

db = db6

elif db5 is not False:

db = db5

elif db4 is not False:

db = db4

elif db3 is not False:

db = db3

elif db2 is not False:

db = db2

elif db1 is not False:

db = db1

u = len(db)

filter2 = t.Tk()

filter2.title("Filter")

filter2.geometry('800x750')

def myfunction3(event):

"""

Входные параметров нет

Выходных параметров нет

Функция обозначения размера экрана для скролла

Автор: Рыльников А.М.

"""

canvas.configure(scrollregion=canvas.bbox("all"), width=filter2.winfo\_screenwidth() - 90,

height=filter2.winfo\_screenheight() - 90)

# создание скроллов

myframe = t.Frame(filter2, relief=t.GROOVE, width=50, height=100, bd=2)

myframe.place(x=1, y=1)

canvas = t.Canvas(myframe)

se = t.Frame(canvas)

myscrollbar2 = t.Scrollbar(myframe, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=myscrollbar2.set)

myscrollbar2.pack(side="right", fill="y")

myscrollbar3 = t.Scrollbar(myframe, orient="horizontal", command=canvas.xview)

canvas.configure(xscrollcommand=myscrollbar3.set)

myscrollbar3.pack(side="bottom", fill="x")

canvas.pack(side="left")

canvas.create\_window((0, 0), window=se, anchor='nw')

se.bind("<Configure>", myfunction3)

if u == 0:

bas = t.Label(se, text=' No such items', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], '40', 'bold'))

bas.grid(row=0, column=0, sticky=t.NSEW)

else:

bas = t.Label(se, text=' Number', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=0, sticky=t.NSEW)

for i in range(len(db)):

bas = t.Label(se, text=' %s ' % (i + 1), relief=t.GROOVE, bg=par['table\_colour\_high'],

fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=i + 1, column=0, sticky=t.NSEW)

# Названия столбцов

bas = t.Label(se, text=' Key ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=1, sticky=t.NSEW)

for l in db.keys():

for i, j in enumerate(db[l]):

bas = t.Label(se, text=' %s ' % j, relief=t.GROOVE, bg=par['table\_colour\_high'],

fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

bas.grid(row=0, column=i + 2, sticky=t.NSEW)

break

# Вывод данных

lis = [2, 3, 4, 5, 6, 7, 8, 9]

j = 1

for k in db.keys():

bas = t.Label(se, text=' %s ' % k, relief=t.GROOVE, bg=par['table\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size']))

bas.grid(row=j, column=1, sticky=t.NSEW)

for i, item in zip(lis, db[k].values()):

bas = t.Label(se, text=' %s ' % item, relief=t.GROOVE, bg=par['table\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size']))

bas.grid(row=j, column=i, sticky=t.NSEW)

j += 1

root.bind\_all('<MouseWheel>', lambda event: rollWheel(event))

root.update()

filter2.mainloop()

def settings(\*args):

"""

Входные параметров нет

Выходных параметров нет

Функция настрек интерфейса

Автор: Яценко И.Ю.

"""

# загрузка параметров

par = pk.load(open('parametrs.pic', mode='rb'))

# создание окна

rt = t.Tk()

rt.title("Settings")

rt.geometry('1300x250')

def myfunction2(event):

"""

Входные параметров нет

Выходных параметров нет

Функция обозначения размера экрана для скролла

Автор: Яценко И.Ю.

"""

canvas.configure(scrollregion=canvas.bbox("all"), width=rt.winfo\_screenwidth() - 90,

height=rt.winfo\_screenheight() - 90)

# создание скроллов

myframe = t.Frame(rt, relief=t.GROOVE, width=50, height=100, bd=2)

myframe.place(x=1, y=1)

canvas = t.Canvas(myframe)

se = t.Frame(canvas)

myscrollbar2 = t.Scrollbar(myframe, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=myscrollbar2.set)

myscrollbar2.pack(side="right", fill="y")

myscrollbar3 = t.Scrollbar(myframe, orient="horizontal", command=canvas.xview)

canvas.configure(xscrollcommand=myscrollbar3.set)

myscrollbar3.pack(side="bottom", fill="x")

canvas.pack(side="left")

canvas.create\_window((0, 0), window=se, anchor='nw')

se.bind("<Configure>", myfunction2)

# ВЫБОР ПУТИ К ТЕКСТОВОМУ ФАЙЛУ ЗАПИСИ

puti = t.Label(se, text=' Way to text file for totals: ', font=('12'))

puti.grid(row=0, column=0, columnspan=2, sticky=t.NSEW)

way = t.Label(se, text = '%s' % par['way\_to\_total'] ,font=('arial', '8'), fg='black', width=65, height=2)

way.grid(row=1, column=0, sticky=t.NSEW)

def s1():

"""

Входные параметров нет

Выходных параметров нет

Функция выбора пути к файлу записи итогов

Автор: Яценко И.Ю.

"""

par = pk.load(open('parametrs.pic', mode='rb'))

fi = fd.askopenfilename()

if fi == '' or fi[-3:] != 'txt':

mb.showerror("Error",

"File error\nChoose file\nYou can choose only 'txt' file")

else:

par['way\_to\_total'] = fi

pk.dump(par, open('parametrs.pic', mode='wb'))

par = pk.load(open('parametrs.pic', mode='rb'))

way = t.Label(se, text='%s' % par['way\_to\_total'], font=('arial', '8'), fg='black', width=65, height=1)

way.grid(row=1, column=0, sticky=t.NSEW)

sett = t.Button(se, text="Open..", width=45, height=1,

font=('arial', '9'), command=s1)

sett.grid(row=1, column=1, sticky=t.NSEW)

# ВЫБОР ТИПА ШРИФТА И ЕГО РАЗМЕРОВ

fn = t.StringVar(se)

fn.set(par['font'])

puti = t.Label(se, text=' Font (current font: %s): ' % par['font'], font=('12'))

puti.grid(row=2, column=0, sticky=t.NSEW)

opm7 = t.OptionMenu(se, fn, 'arial', 'times new roman', 'calibri', 'arial black', 'mv boli',

'cambria', 'gothic', 'cursive', 'lucida console', 'segoe script', 'webdings', 'txt', 'stylus bt')

opm7.grid(row=3, column=0, sticky=t.NSEW)

fn1 = t.StringVar(se)

fn1.set(par['letter\_size'])

puti = t.Label(se, text=' Table items font size(current size: %s): ' % par['letter\_size'], font=('12'))

puti.grid(row=2, column=1, sticky=t.NSEW)

opm9 = t.OptionMenu(se, fn1, '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16',

'17', '18', '19', '20', '21', '22')

opm9.grid(row=3, column=1, sticky=t.NSEW)

fn2 = t.StringVar(se)

fn2.set(par['letter\_size\_h1'])

puti = t.Label(se, text=' Font size(current size: %s): ' % par['letter\_size\_h1'], font=('12'))

puti.grid(row=4, column=0, sticky=t.NSEW)

opm11 = t.OptionMenu(se, fn2, '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16',

'17', '18', '19', '20', '21', '22')

opm11.grid(row=5, column=0, sticky=t.NSEW)

fn3 = t.StringVar(se)

fn3.set(par['letter\_size\_h2'])

puti = t.Label(se, text=' Head font size(current size: %s): ' % par['letter\_size\_h2'], font=('12'))

puti.grid(row=4, column=1, sticky=t.NSEW)

opm12 = t.OptionMenu(se, fn3, '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16',

'17', '18', '19', '20', '21', '22')

opm12.grid(row=5, column=1, sticky=t.NSEW)

# ВЫБОР ЦВЕТОВ ФОНА, ТАБЛИЦЫ, ФУНКЦИЙ, КНОПОК И ШРИФТА

def getCol(k, i, j):

"""

Входные k - ключ словаря к данным параметров, i - строка, j - столбец

Выходных параметров нет

Функция выбора цвета

Автор: Рыльников А.М.

"""

rgb, col = cl.askcolor()

par[k] = col

pk.dump(par, open('parametrs.pic', mode='wb'))

puti = t.Label(se, text=' ', font=('12'), bg=par[k], width=5)

puti.grid(row=i, column=j, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'))

puti.grid(row=0, column=2, sticky=t.NSEW)

puti = t.Label(se, text=' Colours ', font=('12'))

puti.grid(row=0, column=3,columnspan=3, sticky=t.NSEW)

puti = t.Label(se, text=' Background colour: ', font=('12'))

puti.grid(row=1, column=3, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'), bg=par['frame\_colour'], width=5)

puti.grid(row=1, column=4, sticky=t.NSEW)

ch = t.Button(se, text="Choose..", width=20, height=1,

font=('arial', '9'), command=lambda k='table\_colour', i=1, j=4: getCol('frame\_colour', 1, 4))

ch.grid(row=1, column=5, sticky=t.NSEW)

puti = t.Label(se, text=' Table colour: ', font=('12'))

puti.grid(row=2, column=3, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'), bg=par['table\_colour'], width=5)

puti.grid(row=2, column=4, sticky=t.NSEW)

ch = t.Button(se, text="Choose..", width=20, height=1,

font=('arial', '9'), command=lambda k='table\_colour', i=2, j=4: getCol('table\_colour', 2, 4))

ch.grid(row=2, column=5, sticky=t.NSEW)

puti = t.Label(se, text=' Functions and table head colour: ', font=('12'))

puti.grid(row=3, column=3, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'), bg=par['table\_colour\_high'], width=5)

puti.grid(row=3, column=4, sticky=t.NSEW)

ch = t.Button(se, text="Choose..", width=20, height=1,

font=('arial', '9'), command=lambda k='table\_colour\_high', i=3, j=4: getCol('table\_colour\_high', 3, 4))

ch.grid(row=3, column=5, sticky=t.NSEW)

puti = t.Label(se, text=' Letter colour: ', font=('12'))

puti.grid(row=4, column=3, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'), bg=par['letter\_colour'], width=5)

puti.grid(row=4, column=4, sticky=t.NSEW)

ch = t.Button(se, text="Choose..", width=20, height=1,

font=('arial', '9'),

command=lambda k='letter\_colour', i=4, j=4: getCol('letter\_colour', 4, 4))

ch.grid(row=4, column=5, sticky=t.NSEW)

puti = t.Label(se, text=' Buttons colour: ', font=('12'))

puti.grid(row=5, column=3, sticky=t.NSEW)

puti = t.Label(se, text=' ', font=('12'), bg=par['button\_colour'], width=5)

puti.grid(row=5, column=4, sticky=t.NSEW)

ch = t.Button(se, text="Choose..", width=20, height=1,

font=('arial', '9'),

command=lambda k='button\_colour', i=5, j=4: getCol('button\_colour', 5, 4))

ch.grid(row=5, column=5, sticky=t.NSEW)

def s2():

"""

Входные параметров нет

Выходных параметров нет

Функция сохранения настроек и перезапуска программы

Автор: Рыльников А.М.

"""

par = pk.load(open('parametrs.pic', mode='rb'))

par['font'] = fn.get()

par['letter\_size'] = fn1.get()

par['letter\_size\_h1'] = fn2.get()

par['letter\_size\_h2'] = fn3.get()

pk.dump(par, open('parametrs.pic', mode='wb'))

python = sys.executable

os.execl(python, python, \*sys.argv)

fnt = t.Button(se, text="Save and Reset", width=45, height=2, bg = 'light grey',

font=('arial', '12', 'bold'), command=s2)

fnt.grid(row=8, column=0, columnspan=6, sticky=t.NSEW)

def myfunction(event):

"""

Входные параметров нет

Выходных параметров нет

Функция обозначения размера экрана для скролла

Автор: Яценко И.Ю.

"""

canvas.configure(scrollregion=canvas.bbox("all"),width=root.winfo\_screenwidth()-90, height=root.winfo\_screenheight()-90)

def rollWheel(event):

"""

Входные параметров нет

Выходных параметров нет

Скролл колесом мыши

Автор: Рыльников А.М.

"""

direction = 0

if event.delta == -120:

direction = 1

if event.delta == 120:

direction = -1

canvas.yview\_scroll(direction, t.UNITS)

''' ИНТЕРФЕЙС Автор: Петин Д.М. '''

root = t.Tk()

root.title("Database") # ЗАГОЛОВОК ОКНА

root.configure(bg=par['frame\_colour'])

root.geometry('1170x670')

myframe=t.Frame(root,relief=t.GROOVE,width=50,height=100,bd=2)

myframe.place(x=1, y=1)

# СОЗДАНИЕ СКРОЛЛОВ

canvas=t.Canvas(myframe , bg=par['frame\_colour'])

frame=t.Frame(canvas, bg=par['frame\_colour'])

myscrollbar=t.Scrollbar(myframe,orient="vertical",command=canvas.yview)

canvas.configure(yscrollcommand=myscrollbar.set)

myscrollbar.pack(side="right",fill="y")

myscrollbar1=t.Scrollbar(myframe,orient="horizontal",command=canvas.xview)

canvas.configure(xscrollcommand=myscrollbar1.set)

myscrollbar1.pack(side="bottom",fill="x")

canvas.pack(side="left")

canvas.create\_window((0,0),window=frame,anchor='nw')

frame.bind("<Configure>",myfunction)

# Вывод базы данных

# ГЛАВНЫЕ ФРЕЙМЫ

f1 = t.Frame(frame, width=250,height=200,bg=par['frame\_colour'], bd=2)

f1.grid(row=0, column=0, sticky=t.NSEW)

fra2 = t.Frame(frame , width=250,height=200,bg=par['frame\_colour'], bd=2)

fra2.grid(row=0, column=1, sticky=t.NSEW)

# ФРЕЙМ С КНОПКАМИ ЗАГРУЗКИ,ПЕЧАТИ,ОЧИСТКИ И НАСТРОЕК

fra1 = t.Frame(f1, width=250,height=200,bg=par['frame\_colour'], bd=2)

fra1.grid(row=0, column=0, sticky=t.NSEW)

# ФРЕЙМ С БАЗОЙ

basafr = t.Frame(f1,width=250,height=200,bg=par['frame\_colour'], bd=2)

basafr.grid(row=1, column=0, sticky=t.NSEW)

# ФРЕЙМ С ФУНКЦИЯМИ

dele\_f = t.Frame(fra2,width=150,height=200,bg=par['frame\_colour'], bd=2)

dele\_f.grid(row=0, column=0, sticky=t.NSEW)

# КНОПКА ЗАГРУЗКИ БАЗЫ

load = t.Button(fra1, text="Load base", width=30, height=1, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=load)

load.grid(row=0, column=0, sticky=t.NSEW)

# КНОПКА ПЕЧАТИ БАЗЫ

prnt = t.Button(fra1, text="Print base", width=29, height=1, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=print\_butt, state='disabled')

prnt.grid(row=0, column=1, sticky=t.NSEW)

# КНОПКА ОЧИСТКИ БАЗЫ

clr = t.Button(fra1, text="Clear base", width=29, height=1, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=lambda fra=basafr: clear\_base(basafr), state='disabled')

clr.grid(row=0, column=2, sticky=t.NSEW)

# КНОПКА НАСТРОЕК

sett = t.Button(fra1, text="Settings", width=29, height=1, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=settings)

sett.grid(row=0, column=3, sticky=t.NSEW)

# КНОПКА ВОЗВРАЩЕНИЯ СТАНДАРТНОЙ БАЗЫ

defa = t.Button(dele\_f, text="Return Defaults", width=20, height=2, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=defaults, state='disabled')

defa.grid(row=0, column=0, columnspan=2, sticky=t.NSEW)

# ВИДЖЕТ ДОБАВЛЕНИЯ ЭЛЕМЕНТОВ

adit = t.Label(dele\_f, text=' \nInput data for new item:\n ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h2'], 'bold'))

adit.grid(row=1, column=0, columnspan=2, sticky=t.NSEW)

# ИМЯ

adit = t.Label(dele\_f, text=' Name: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=2, column=0, sticky=t.NSEW)

nam = t.Entry(dele\_f, width=30, bd=8)

nam.grid(row=2, column=1, sticky=t.NSEW)

# СТРАНА

adit = t.Label(dele\_f, text=' Nation: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=3, column=0, sticky=t.NSEW)

nat = t.Entry(dele\_f, width=20, bd=8)

nat.grid(row=3, column=1, sticky=t.NSEW)

# ВОЗРАСТ

adit = t.Label(dele\_f, text=' Age: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=4, column=0, sticky=t.NSEW)

age1 = t.Entry(dele\_f, width=20, bd=8)

age1.grid(row=4, column=1, sticky=t.NSEW)

# КЛУБ

adit = t.Label(dele\_f, text=' Club: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=5, column=0, sticky=t.NSEW)

clu = t.Entry(dele\_f, width=20, bd=8)

clu.grid(row=5, column=1, sticky=t.NSEW)

# ПОЗИЦИЯ

adit = t.Label(dele\_f, text=' Position: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=6, column=0, sticky=t.NSEW)

pos = t.StringVar()

opm2=t.OptionMenu(dele\_f, pos, 'Forward', 'Winger', 'Attacking Midfielder', 'Defending Midfielder',

'Left Defender', 'Center Defender', 'Right Defender', 'Goalkeeper')

opm2.grid(row=6, column=1, sticky=t.NSEW)

# СТОИМОСТЬ

adit = t.Label(dele\_f, text=' Value: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=7, column=0, sticky=t.NSEW)

val = t.Entry(dele\_f, width=20, bd=8)

val.grid(row=7, column=1, sticky=t.NSEW)

# ДАТА ИСТЕЧЕНИЯ КОНТРАКТА

adit = t.Label(dele\_f, text=' Contract expire date: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=8, column=0, sticky=t.NSEW)

exp = t.StringVar()

opm3 = t.OptionMenu(dele\_f, exp, '30.06.2018', '30.06.2019', '30.06.2020', '30.06.2021',

'30.06.2022', '30.06.2023', '30.06.2024', '30.06.2026', '30.06.2027', '30.06.2028', '30.06.2029')

opm3.grid(row=8, column=1, sticky=t.NSEW)

# ЗАРПЛАТА

adit = t.Label(dele\_f, text=' Salary : ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

adit.grid(row=9, column=0, sticky=t.NSEW)

sal = t.Entry(dele\_f, width=20, bd=8)

sal.grid(row=9, column=1, sticky=t.NSEW)

# КНОПКА ДОБАВЛЕНИЯ

ad = t.Button(dele\_f, text="Add item", width=20, height=2, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=add, state='disabled')

ad.grid(row=10, column=0, columnspan=2, sticky=t.NSEW)

# ВИДЖЕТ ФИЛЬТРА

fil = t.Label(dele\_f, text='\n Filter: \n', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=11, column=0,columnspan=4, sticky=t.NSEW)

fil = t.Label(dele\_f, text='\n Column: \n', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=12, column=0, sticky=t.NSEW)

fil = t.Label(dele\_f, text='\n Operation: \n', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=12, column=1, sticky=t.NSEW)

fil = t.Label(dele\_f, text='\n Value1: \n', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=12, column=2, sticky=t.NSEW)

fil = t.Label(dele\_f, text='\n Value2: \n', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=12, column=3, sticky=t.NSEW)

#NAME

fil = t.Label(dele\_f, text=' Name: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=13, column=0, columnspan=2, sticky=t.NSEW)

nama = t.Entry(dele\_f, width=20, bd=8)

nama.grid(row=13, column=2, columnspan=2, sticky=t.NSEW)

#Nationality

fil = t.Label(dele\_f, text=' Nationality ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=14, column=0, columnspan=2, sticky=t.NSEW)

nata = t.Entry(dele\_f, width=20, bd=8)

nata.grid(row=14, column=2, columnspan=2, sticky=t.NSEW)

#AGE

fil = t.Label(dele\_f, text=' Age: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=15, column=0, columnspan=2, sticky=t.NSEW)

a11 = t.Entry(dele\_f, width=15, bd=8)

a11.grid(row=15, column=2, sticky=t.NSEW)

a12 = t.Entry(dele\_f, width=15, bd=8)

a12.grid(row=15, column=3, sticky=t.NSEW)

#Club

fil = t.Label(dele\_f, text=' Club: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=16, column=0, columnspan=2, sticky=t.NSEW)

cl3 = t.Entry(dele\_f, width=20, bd=8)

cl3.grid(row=16, column=2, columnspan=2, sticky=t.NSEW)

#Position

fil = t.Label(dele\_f, text=' Postion: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=17, column=0, columnspan=2, sticky=t.NSEW)

pos12 = t.StringVar()

pos123=t.OptionMenu(dele\_f, pos12, '', 'Forward', 'Winger', 'Attacking Midfielder', 'Defending Midfielder',

'Left Defender', 'Center Defender', 'Right Defender', 'Goalkeeper')

pos123.grid(row=17, column=2, columnspan=2, sticky=t.NSEW)

#Transfer value

fil = t.Label(dele\_f, text=' Transfer value: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=18, column=0, columnspan=2, sticky=t.NSEW)

tv2 = t.StringVar()

t11 = t.Entry(dele\_f, width=15, bd=8)

t11.grid(row=18, column=2, sticky=t.NSEW)

t12 = t.Entry(dele\_f, width=15, bd=8)

t12.grid(row=18, column=3, sticky=t.NSEW)

#Contract

fil = t.Label(dele\_f, text=' Contract expire date: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=19, column=0,columnspan=2, sticky=t.NSEW)

dt2 = t.StringVar()

d11 = t.StringVar()

data11 = t.OptionMenu(dele\_f, d11, '', '30.06.2018', '30.06.2019', '30.06.2020', '30.06.2021',

'30.06.2022', '30.06.2023', '30.06.2024', '30.06.2026', '30.06.2027', '30.06.2028', '30.06.2029')

data11.grid(row=19, column=2, sticky=t.NSEW)

d12 = t.StringVar()

data12 = t.OptionMenu(dele\_f, d12, '', '30.06.2018', '30.06.2019', '30.06.2020', '30.06.2021',

'30.06.2022', '30.06.2023', '30.06.2024', '30.06.2026', '30.06.2027', '30.06.2028', '30.06.2029')

data12.grid(row=19, column=3, sticky=t.NSEW)

#Salary

fil = t.Label(dele\_f, text=' Salary: ', relief=t.GROOVE, bg=par['table\_colour\_high'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'))

fil.grid(row=20, column=0,columnspan=2, sticky=t.NSEW)

sl11 = t.Entry(dele\_f, width=15, bd=8)

sl11.grid(row=20, column=2, sticky=t.NSEW)

sl12 = t.Entry(dele\_f, width=15, bd=8)

sl12.grid(row=20, column=3, sticky=t.NSEW)

fltr = t.Button(dele\_f, text="Filter", width=20, height=2, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=serch, state='disabled')

fltr.grid(row=21, column=0, columnspan=2, sticky=t.NSEW)

def clean():

"""

Входные параметров нет

Выходных параметров нет

Очистка полей фильтра

Автор: Яценко И.Ю.

"""

nama.delete(0,t.END)

sl11.delete(0, t.END)

sl12.delete(0, t.END)

t11.delete(0, t.END)

t12.delete(0, t.END)

cl3.delete(0, t.END)

a11.delete(0, t.END)

a12.delete(0, t.END)

nata.delete(0, t.END)

d11.set('')

d12.set('')

pos12.set('')

cl1 = t.Button(dele\_f, text="Clear", width=20, height=2, bg=par['button\_colour'], fg=par['letter\_colour'],

font=(par['font'], par['letter\_size\_h1'], 'bold'), command=clean)

cl1.grid(row=21, column=2, columnspan=2, sticky=t.NSEW)

root.bind\_all('<MouseWheel>', lambda event: rollWheel(event))

root.update()

root.mainloop()

# Листинг functions.py

import pickle as pk

def load\_file():

"""

Входных параметров нет

Выхожной параметр - database - база данных из файла

Загрузка базы данных из файла в глобальную переменную database

Автор: Петин Д.М.

"""

par = pk.load(open('parametrs.pic', mode='rb'))

database = pk.load(open(par['way\_to\_base'], mode='rb'))

return database

def save\_file(database):

"""

Входной парметр - database - база для записи в файл

Выходных парметров нет

Сохраняет бызу данных в файл

Автор: Яценко И.Ю

"""

par = pk.load(open('parametrs.pic', mode='rb')) # загрузка параметров из файла

pk.dump(database, open(par['way\_to\_base'], mode='wb'))

def addition(database, name, nation, age, club, position, value, contract\_date, salary):

"""

Входные параметры - name, nation, age, club, position, value, contract\_date, salary - ключи-названия колонок базы

Выходных параметров нет

Добавление новой записи в базу данных

Автор: Петин Д.М.

"""

# Поиск свободного ключа в базе

i = 1

while i in database:

i += 1

# Запись данных по найденному ключу

database[i] = {'Name': name,

'Nationality': nation,

'Age': age,

'Club': club,

'Position': position,

'Transfer Value': value,

'Contract expiry date': contract\_date,

'Salary': salary}

def delete(database, number):

"""

Входные параметры - number - номер колонки к удалению

Выходных параметров нет

Удаление записи по номеру

Автор: Рыльников А.М.

"""

number = int(number)

database.pop(number, None)

def edition(database,k, name, nation, age, club, position, value, contract\_date, salary):

"""

ходные параметры - name, nation, age, club, position, value, contract\_date, salary - ключи-названия колонок базы

Выходных параметров нет

Редактирование записи по номеру и ключу

Автор: Яценко И.Ю.

"""

database[k] = {'Name': name,

'Nationality': nation,

'Age': age,

'Club': club,

'Position': position,

'Transfer Value': value,

'Contract expiry date': contract\_date,

'Salary': salary}

def is\_int(s):

"""

Входные параметры - s - переменная для проверки

Возаращает True - если s - целое, иначе False

Проверка на целочисленность

Автор: Петин Д.М.

"""

if s == '':

return True

try:

int(s)

return True

except ValueError:

return False

def is\_float(s):

"""

Входные параметры - s - переменная для проверки

Возаращает True - если s - число, иначе False

Проверка на число

Автор: Петин Д.М.

"""

if s == '':

return True

try:

float(s)

return True

except ValueError:

return False

def is\_date(s):

"""

Входные параметры - s - переменная для проверки

Возаращает True - если s - корректная дата, иначе False

Проверка на корректную дату

Автор: Рыльников А.М.

"""

a = s[0:2] # число

b = s[3:5] # месяц

c = s[6:10] # год

# проверка на целочисленность числа,месяца и года

a1 = is\_int(a)

b1 = is\_int(b)

c1 = is\_int(c)

d30 = ['04', '06', '09', '11'] # месяцы 30 дней

d29 = ['02'] # месяцы 29 дней

d31 = ['01', '03', '05', '07', '08', '10', '12'] # месяцы 31 день

if len(s) == 10 and a1 and b1 and c1 and s[2] == '.' and s[5] == '.' and int(a) > 0 and int(b) > 0 and \

int(c) >= 2018 and ((b in d30 and int(a) <= 30) or (b in d31 and int(a) <= 31) or

(b in d29 and int(a) <= 29)):

return True

else:

return False

def search(database, key, value):

"""

Входной парметр - database, key, value - база для записи в файл

Возвращает db - словарь словарей - база данных с найденными элементами

Ищет элементы по строковым значениям

Автор: Яценко И.Ю.

"""

if value == '':

return False

else:

db = {}

res = 0

for i in database.keys():

if value.lower() in database[i][key].lower():

db[res] = database[i]

res += 1

return db

def search\_num(database,key, value1, value2):

"""

Входной парметр - database, key, value1, value2 - база для записи в файл

Возвращает db - словарь словарей - база данных с найденными элементами

Ищет элементы по числовым значениям

Автор: Петин Д.М

"""

if value1 == '' and value2 == '':

return False

elif value1 != '' and value2 != '':

db = {}

res = 0

for i in database.keys():

if float(value1) <= float(database[i][key]) <= float(value2):

db[i] = database[i]

res += 1

return db

elif value1 != '' and value2 == '':

db = {}

res = 0

for i in database.keys():

if float(value1) <= float(database[i][key]) :

db[i] = database[i]

res += 1

return db

elif value1 == '' and value2 != '':

db = {}

res = 0

for i in database.keys():

if float(database[i][key]) <= float(value2):

db[i] = database[i]

res += 1

return db

def search\_date(database,key, value1, value2):

"""

Входной парметр - database, key, value1, value2 - база для записи в файл

Возвращает db - словарь словарей - база данных с найденными элементами

Ищет значения по дате

Автор: Рыльников А.М.

"""

if value1 == '' and value2 == '':

return False

elif value1 != '' and value2 != '':

db = {}

value1 = float(str(value1)[-4:])

value2 = float(str(value2)[-4:])

res = 0

for i in database.keys():

if value1 <= float(str(database[i][key])[-4:]) <= value2:

db[i] = database[i]

res += 1

return db

elif value1 != '' and value2 == '':

db = {}

value1 = float(str(value1)[-4:])

#value2 = float(str(value2)[-4:])

res = 0

for i in database.keys():

if value1 <= float(str(database[i][key])[-4:]):

db[i] = database[i]

res += 1

return db

elif value1 == '' and value2 != '':

db = {}

#value1 = float(str(value1)[-4:])

value2 = float(str(value2)[-4:])

res = 0

for i in database.keys():

if float(str(database[i][key])[-4:]) <= value2:

db[i] = database[i]

res += 1

return db

# Листинг base.py

import sys # модули sys и os для функции перезагрузки программы

sys.path.append('../library')

import functions as f

# создание оригинальной базы данных

def defa():

"""

Входные параметров нет

Выходных параметров нет

Возвращение оригинальной базы

Автор: Петин Д.М.

"""

def base\_fotballers():

"""

Входные параметров нет

Выходных параметров нет

Создание базы

Автор: Петин Д.М.

"""

Promes = {'Name': 'Quincy Promes',

'Nationality': 'Netherlands',

'Age': 26,

'Club': 'Spartak',

'Position': 'Winger',

'Transfer Value': 22 ,

'Contract expiry date': '30.06.2021' ,

'Salary': 3.8}

Smolov = {'Name': 'Fedor Smolov',

'Nationality': 'Russia',

'Age': 28,

'Club': 'FC Krasnodar',

'Position': 'Forward',

'Transfer Value': 15,

'Contract expiry date': '30.06.2019',

'Salary': 2.9}

Kokorin = {'Name': 'Alexandr Kokorin',

'Nationality': 'Russia',

'Age': 27,

'Club': 'Zenit',

'Position': 'Forward',

'Transfer Value': 11,

'Contract expiry date': '30.06.2019',

'Salary': 3.3}

Adriano = {'Name': 'Luis Adriano',

'Nationality': 'Brazil',

'Age': 30,

'Club': 'Spartak',

'Position': 'Forward',

'Transfer Value': 6,

'Contract expiry date': '30.06.2019',

'Salary': 3.5}

Miranchuk = {'Name': 'Alexey Miranchuk',

'Nationality': 'Russia',

'Age': 22,

'Club': 'Lokomotiv',

'Position': 'Attacking Midfielder',

'Transfer Value': 10,

'Contract expiry date': '30.06.2021',

'Salary': 1.8}

Akinfeev = {'Name': 'Igor Akinfeev',

'Nationality': 'Russia',

'Age': 32,

'Club': 'CSKA',

'Position': 'Goalkeeper',

'Transfer Value': 10,

'Contract expiry date': '30.06.2019',

'Salary': 2.5}

Popov = {'Name': 'Ivelin Popov',

'Nationality': 'Bulgaria',

'Age': 30,

'Club': 'Rubin',

'Position': 'Attacking Midfielder',

'Transfer Value': 3.5,

'Contract expiry date': '30.06.2019',

'Salary': 1.8}

MarFernandes = {'Name': 'Mario Fernandes',

'Nationality': 'Brazil',

'Age': 27,

'Club': 'CSKA',

'Position': 'Right Defender',

'Transfer Value': 16,

'Contract expiry date': '30.06.2021',

'Salary': 2.2}

Shatov = {'Name': 'Oleg Shatov',

'Nationality' : 'Russia',

'Age': 27,

'Club': 'Zenit',

'Position' : 'Attacking Midfielder',

'Transfer Value': 8.5,

'Contract expiry date': '30.06.2020',

'Salary': 2.5}

Samedov = {'Name': 'Alexander Samedov',

'Nationality': 'Russia',

'Age': 33,

'Club': 'Spartak',

'Position': 'Winger',

'Transfer Value': 1.7,

'Contract expiry date': '30.06.2019',

'Salary': 2.5}

ManFernandes = {'Name': 'Manuel Fernandes',

'Nationality': 'Portugal',

'Age': 32,

'Club': 'Lokomotiv',

'Position': 'Attacking Midfielder',

'Transfer Value': 4.5,

'Contract expiry date': '30.06.2019',

'Salary': 3.8}

Paredes = {'Name': 'Leandro Paredes',

'Nationality': 'Argentina',

'Age': 23,

'Club': 'Zenit',

'Position': 'Defending Midfielder',

'Transfer Value': 18,

'Contract expiry date': '30.06.2019',

'Salary': 5}

Fernando = {'Name': 'Lukas Fernando',

'Nationality': 'Brazil',

'Age': 26,

'Club': 'Spartak',

'Position': 'Defending Midfielder',

'Transfer Value': 14,

'Contract expiry date': '30.06.2021',

'Salary': 2.8}

Wernbloom = {'Name': 'Pontus Wernbloom',

'Nationality': 'Sweden',

'Age': 31,

'Club': 'CSKA',

'Position': 'Defending Midfielder',

'Transfer Value': 7.5,

'Contract expiry date': '30.06.2018',

'Salary': 2}

Criscito = {'Name': 'Dominico Criscito',

'Nationality': 'Italy',

'Age': 31,

'Club': 'Zenit',

'Position': 'Left Defender',

'Transfer Value': 5,

'Contract expiry date': '30.06.2018',

'Salary': 3}

Corluka = {'Name': 'Vedran Corluka',

'Nationality': 'Croatia',

'Age': 32,

'Club': 'Lokomotiv',

'Position': 'Center Defender',

'Transfer Value': 4.2,

'Contract expiry date': '30.06.2020',

'Salary': 4.5}

Ivanovich = {'Name': 'Branislav Ivanovich',

'Nationality': 'Serbia',

'Age': 34,

'Club': 'Zenit',

'Position': 'Center Defender',

'Transfer Value': 4,

'Contract expiry date': '30.06.2019',

'Salary': 4.1}

Glushakov = {'Name': 'Denis Glushakov',

'Nationality': 'Russia',

'Age': 31,

'Club': 'Spartak',

'Position': 'Defending Midfielder',

'Transfer Value': 7.5,

'Contract expiry date': '30.06.2020',

'Salary': 2.6}

ZeLuis = {'Name': 'Ze Luis',

'Nationality': 'Cabo-Verde',

'Age': 27,

'Club': 'Spartak',

'Position': 'Forward',

'Transfer Value': 5.5,

'Contract expiry date': '30.06.2021',

'Salary': 2}

Kombarov = {'Name': 'Dmitry Kombarov',

'Nationality': 'Russia',

'Age': 31,

'Club': 'Spartak',

'Position': 'Left Defender',

'Transfer Value': 3.2,

'Contract expiry date': '30.06.2020',

'Salary': 2}

Tasci = {'Name': 'Serdar Tasci',

'Nationality': 'Germany',

'Age': 30,

'Club': 'Spartak',

'Position': 'Center Defender',

'Transfer Value': 2,

'Contract expiry date': '30.06.2018',

'Salary': 2.2}

Dzagoev = {'Name': 'Alan Dzagoev',

'Nationality': 'Russia',

'Age': 27,

'Club': 'CSKA',

'Position': 'Attacking Midfielder',

'Transfer Value': 15,

'Contract expiry date': '30.06.2019',

'Salary': 2.2}

Guilherme = {'Name': 'Marinato Guilherme',

'Nationality': 'Russia',

'Age': 32,

'Club': 'Lokomotiv',

'Position': 'Goalkeeper',

'Transfer Value': 4,

'Contract expiry date': '30.06.2019',

'Salary': 2}

Zobnin = {'Name': 'Roman Zobnin',

'Nationality': 'Russia',

'Age': 24,

'Club': 'Spartak',

'Position': 'Defending Midfielder',

'Transfer Value': 6,

'Contract expiry date': '30.06.2021',

'Salary': 2.4}

Granquist = {'Name': 'Andreas Granquist',

'Nationality': 'Sweden',

'Age': 32,

'Club': 'FC Krasnodar',

'Position': 'Center Defender',

'Transfer Value': 4.5,

'Contract expiry date': '30.06.2018',

'Salary': 2.4}

Driussi = {'Name': 'Sebastjan Driussi',

'Nationality': 'Argentina',

'Age': 22,

'Club': 'Zenit',

'Position': 'Forward',

'Transfer Value': 12,

'Contract expiry date': '30.06.2021',

'Salary': 2.7}

Melgarejo = {'Name': 'Lorenzo Melgarejo',

'Nationality': 'Paraguay',

'Age': 27,

'Club': 'Spartak',

'Position': 'Winger',

'Transfer Value': 3.5,

'Contract expiry date': '30.06.2020',

'Salary': 2}

Poloz = {'Name': 'Dmitry Poloz',

'Nationality': 'Russia',

'Age': 26,

'Club': 'Zenit',

'Position': 'Winger',

'Transfer Value': 7,

'Contract expiry date': '30.06.2020',

'Salary': 1.8}

Djanaev = {'Name': 'Soslan Djanaev',

'Nationality': 'Russia',

'Age': 31,

'Club': 'Rubin',

'Position': 'Goalkeeper',

'Transfer Value': 2.5,

'Contract expiry date': '30.06.2020',

'Salary': 1.60}

Kardeniz = {'Name': 'Gokdeniz Kardeniz',

'Nationality': 'Turkey',

'Age': 38,

'Club': 'Rubin',

'Position': 'Winger',

'Transfer Value': 0.5,

'Contract expiry date': '30.06.2018',

'Salary': 2.5}

footballers = {

1: Promes,

2: Smolov,

3: Kokorin,

4: Adriano,

5: Miranchuk,

6: Akinfeev,

7: Popov,

8: MarFernandes,

9: Shatov,

10: Samedov,

11: ManFernandes,

12: Paredes,

13: Fernando,

14: Wernbloom,

15: Criscito,

16: Corluka,

17: Ivanovich,

18: Glushakov,

19: ZeLuis,

20: Kombarov,

21: Tasci,

22: Dzagoev,

23: Guilherme,

24: Zobnin,

25: Granquist,

26: Driussi,

27: Melgarejo,

28: Poloz,

29: Djanaev,

30: Kardeniz,

}

return footballers

base = base\_fotballers()

f.save\_file(base)