**Параллельность выполнения потоков**

1. Выолнить программу
2. Прокомментировать назначение переменных и операторов
3. Создайте интерфейс для регулирования шага продвижения потоков

Следующий пример иллюстрирует параллельность выполнения потоков, используя возможности библиотеки графических примитивов Tkinter (она входит в стандартную поставку Python). Несколько потоков наперегонки увеличивают размеры прямоугольника некоторого цвета. Цветом победившего потока окрашивается кнопка Go:

import threading, time, sys

from tkinter import \*

global champion

# Задается дистанция, цвет полосок и другие параметры

distance = 300

colors = ["Red","Orange","Yellow","Green","Blue","DarkBlue","Violet"]

nrunners = len(colors) # количество дополнительных потоков

positions = [0] \* nrunners # список текущих позиций

h, h2 = 20, 10 # параметры высоты полосок

def run(n):

"""Программа бега n-го участника (потока)"""

global champion

while 1:

for i in range(10000): # интенсивные вычисления

pass

graph\_lock.acquire()

positions[n] += 1 # передвижение на шаг

if positions[n] == distance: # если уже финиш

if champion is None: # и чемпион еще не определен,

champion = colors[n] # назначается чемпион

graph\_lock.release()

break

graph\_lock.release()

def ready\_steady\_go():

"""Инициализация начальных позиций и запуск потоков"""

graph\_lock.acquire()

for i in range(nrunners):

positions[i] = 0

threading.Thread(target=run, args=[i,]).start()

graph\_lock.release()

def update\_positions():

"""Обновление позиций"""

graph\_lock.acquire()

for n in range(nrunners):

c.coords(rects[n], 0, n\*h, positions[n], n\*h+h2)

tk.update\_idletasks() # прорисовка изменений

graph\_lock.release()

def quit():

"""Выход из программы"""

tk.quit()

sys.exit(0)

# Прорисовка окна, основы для прямоугольников и самих прямоугольников,

# кнопок для пуска и выхода

tk = Tk()

tk.title("Соревнование потоков")

c = Canvas(tk, width=distance, height=nrunners\*h, bg="White")

c.pack()

rects = [c.create\_rectangle(0, i\*h, 0, i\*h+h2, fill=colors[i])

for i in range(nrunners)]

go\_b = Button(text="Go", command=tk.quit)

go\_b.pack(side=LEFT)

quit\_b = Button(text="Quit", command=quit)

quit\_b.pack(side=RIGHT)

# Замок, регулирующий доступ к функции пакета Tk

graph\_lock = threading.Lock()

# Цикл проведения соревнований

while 1:

go\_b.config(state=NORMAL), quit\_b.config(state=NORMAL)

tk.mainloop() # Ожидание нажатия клавиш

champion = None

ready\_steady\_go()

go\_b.config(state=DISABLED), quit\_b.config(state=DISABLED)

# Главный поток ждет финиша всех участников

while sum(positions) < distance\*nrunners:

update\_positions()

update\_positions()

go\_b.config(bg=champion) # Кнопка окрашивается в цвет победителя

tk.update\_idletasks()