Комп’ютерний практикум 6. Багатопотоковий застосунок із GUI на Python для зображення вимірювальної інформації

Розширити функціонал програми із попередньої практичної розбивши операції генерації даних, їх відображення та збереження на окремі потоки.
Самостійно розібратись з такими термінами, як багатопотоковість та черги в програмуванні. Розглянути особливості створення та застосування потоків та черг в Python. Розглянути модулі threading та queue.

План роботи

1. Приклади застосунків
2. Індивідуальні завдання

1. Приклади застосунків

1.1. Моделювання вимірювальної інформації

Лістинг 1

```
import matplotlib
matplotlib.use('TkAgg')

# Бібліотека для математичних розрахунків
import numpy as np

from time import localtime, strftime

import threading, queue, time

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from matplotlib.figure import Figure

import tkinter as tk

tby = dict(side=tk.TOP, \
           fill=tk.BOTH, \
           expand=tk.YES)

txn = dict(side=tk.TOP, \
           fill=tk.X, \
           expand=tk.NO)

class ChannelThread(threading.Thread):
    def __init__(self, dataQueue, sampleLen, dt):
        self.running = True
        self.dataQueue = dataQueue
        self.sampleLen = sampleLen
```

```
30          self.dt = dt
31          super().__init__()
32          #threading.Thread.__init__(self)
33
34      def run(self):
35          """
36          Override threading.Thread.run(self)
37          Generate data.
38          """
39          t = np.arange(0, self.sampleLen, self.dt)
40          while self.running:
41              ch0 = np.sin(2*np.pi*t) + np.random.randn(t.shape[0])/10
42              ch1 = np.cos(2*np.pi*t) + np.random.randn(t.shape[0])/10
43              ch2 = np.sin(2*np.pi*t) * np.cos(2*np.pi*t) + \
44                                  np.random.randn(t.shape[0])/10
45              self.dataQueue.put([t, ch0, ch1, ch2])
46              print('Thread is working')
47              time.sleep(2)
48
49      def stop(self):
50          self.running = False
51
52  class ChannelThread2(threading.Thread):
53      def __init__(self, dataQueue, saveQueue, sampleLen, dt):
54          self.running = True
55          self.dataQueue = dataQueue
56          self.saveQueue = saveQueue
57          self.sampleLen = sampleLen
58          self.dt = dt
59          super().__init__()
60          #threading.Thread.__init__(self)
61
62      def run(self):
63          """
64          Override threading.Thread.run(self)
65          Generate data.
66          """
67          t = np.arange(0, self.sampleLen, self.dt)
68          while self.running:
69              ch0 = np.sin(2*np.pi*t) + np.random.randn(t.shape[0])/10
70              ch1 = np.cos(2*np.pi*t) + np.random.randn(t.shape[0])/10
71              ch2 = np.sin(2*np.pi*t) * np.cos(2*np.pi*t) + \
72                                  np.random.randn(t.shape[0])/10
73              data = [t, ch0, ch1, ch2]
74              self.dataQueue.put(data)
```

```
75              self.saveQueue.put(data)
76              print('Thread is working')
77              time.sleep(2)
78
79      def stop(self):
80          self.running = False
81
82  class SaveThread(threading.Thread):
83      def __init__(self, saveQueue, fileHandle):
84          self.running = True
85          self.saveQueue = saveQueue
86          self.fileHandle = fileHandle
87          super().__init__()
88          #threading.Thread.__init__(self)
89
90      def run(self):
91          """
92          Override threading.Thread.run(self)
93          Save data to file.
94          """
95          while self.running:
96              try:
97                  data = self.saveQueue.get(block=False)
98              except queue.Empty:
99                  pass
100             else:
101                 for x0, x1, x2 in zip(data[1], data[2], data[3]):
102                         self.fileHandle.write('{:.4f} {:.4f} {:.4f}{}'.format(x0, x1, x2, '\
n'))
103                     self.fileHandle.flush()
104             time.sleep(0.1)
105             print('Saving')
106
107     def stop(self):
108         self.running = False
109
110 class CanvasFrame(tk.Frame):
111     def __init__(self, master):
112         super().__init__(master)
113         self.pack(tby)
114
115         self._canvas = FigureCanvasTkAgg(Figure(figsize=(5, 4), \
116                                                 dpi=100), \
117                                          master=self)
118         self._canvas.get_tk_widget().pack(tby)
```

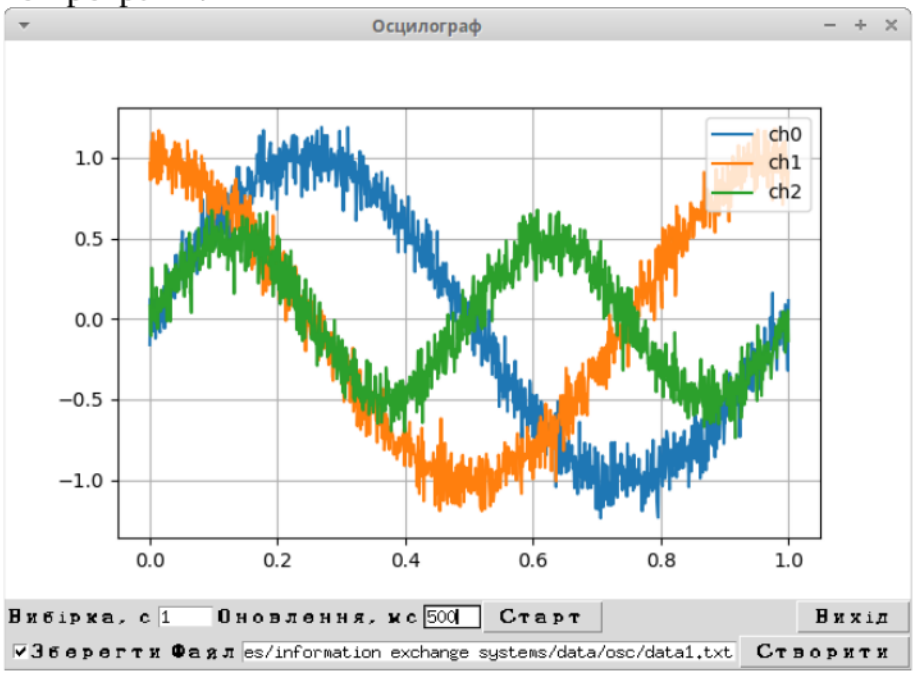
```
119         self._ax = self._canvas.figure.add_subplot(111)
120         self._ax.set_xlabel('t, c')
121         self._ax.grid()
122 
123 class RunFrame(tk.Frame):
124     def __init__(self, master):
125         super().__init__(master)
126         self.pack(txn)
127         tk.Label(self, text='dt, c').pack(side=tk.LEFT)
128         self.dt = tk.Entry(self, width=5)
129         self.dt.pack(side=tk.LEFT)
130         self.dt.insert(0, '0.01')
131         tk.Label(self, text='Вибірка, с').pack(side=tk.LEFT)
132         self.sample = tk.Entry(self, width=5)
133         self.sample.pack(side=tk.LEFT)
134         self.sample.insert(0, '1')
135         tk.Label(self, text='Оновлення, мс').pack(side=tk.LEFT)
136         self.fps = tk.Entry(self, width=5)
137         self.fps.pack(side=tk.LEFT)
138         self.fps.insert(0, '500')
139         tk.Button(master=self,
140                   text='Старт',
141                   command=master.start).pack(side=tk.LEFT)
142         tk.Button(master=self,
143                   text='Вихід',
144                   command=self._quit).pack(side=tk.RIGHT)
145 
146     def _quit(self):
147         if self.master.dataThread:
148             self.master.stop_thread(self.master.dataThread)
149         if self.master.saveThread:
150             self.master.stop_thread(self.master.saveThread)
151         self.quit()
152         self.destroy()
153 
154     def get_sample_size(self):
155         return eval(self.sample.get())
156 
157     def get_fps(self):
158         return eval(self.fps.get())
159 
160     def get_dt(self):
161         return eval(self.dt.get())
162 
163 class SaveOpenFrame(tk.Frame):
```

```
164     def __init__(self, master):
165         super().__init__(master)
166         self.pack(txn)
167         self.fileHandle = None
168         self.checkVar = tk.IntVar()
169                     self.saveCheckbutton  =  tk.Checkbutton(self,  text="Зберегти",
variable=self.checkVar)
170         self.saveCheckbutton.pack(side=tk.LEFT)
171         self.pathVar = tk.StringVar()
172         self.filedialog = tk.filedialog
173         tk.Label(self, text='Файл').pack(side=tk.LEFT)
174         self.file_path = tk.Entry(self, textvariable=self.pathVar, width=50)
175         self.file_path.pack(side=tk.LEFT)
176         tk.Button(master=self,
177                 text='Створити',
178                 command=self._file_save).pack(side=tk.LEFT)
179         tk.Button(master=self,
180                 text='Відкрити',
181                 command=self._file_open).pack(side=tk.RIGHT)
182
183     def _file_save(self):
184         self.pathVar.set(self.filedialog.asksaveasfilename(initialdir = "", \
185                                 title = "Створити файл", \
186                                 filetypes = (("Txt files","*.txt"),("all files","*.*"))))
187
188     def _file_open(self):
189         path = self.filedialog.askopenfilename(initialdir = "", \
190                                 title = "Відкрити файл", \
191                                 filetypes = (("Txt files","*.txt"),("all files","*.*")))
192
193         chanNames = []
194         dt = 0.0
195         with open(path, 'r') as f:
196             f.readline().split()
197             dt = eval(f.readline()[3:])
198             chanNames = f.readline().split()
199         data = np.genfromtxt(path, skip_header=3)
200           self.master.update_canvas([np.arange(0, data.shape[0]*dt, dt), data[:, 0],
data[:, 1], data[:, 2]], chanNames)
201
202   class Application(tk.Frame):
203     def __init__(self, master):
204         super().__init__(master)
205         self.pack(tby)
206         self._timer = None
```

```
207         self.sampleLen = 0
208         self.dt = 0
209         self.isStarted = False
210         self.channels = ['ch0', 'ch1', 'ch2']
211         self.dataQueue = queue.Queue()
212         self.saveQueue = queue.Queue()
213         self.dataThread = None
214         self.saveThread = None
215         self._create_widgets()
216 
217     def _create_widgets(self):
218         self.canvasFrame = CanvasFrame(self)
219         self.canvasFrame.pack(tby)
220         self.runFrame = RunFrame(self)
221         self.runFrame.pack(txn)
222         self.saveFrame = SaveOpenFrame(self)
223         self.saveFrame.pack(txn)
224 
225     def update_canvas(self, data, chanNames):
226         '''
227             Update figure plots
228         '''
229         self.canvasFrame._ax.clear()
230         self.canvasFrame._ax.plot(data[0], data[1], label=chanNames[0])
231         self.canvasFrame._ax.plot(data[0], data[2], label=chanNames[1])
232         self.canvasFrame._ax.plot(data[0], data[3], label=chanNames[2])
233         self.canvasFrame._ax.legend(loc=1)
234         self.canvasFrame._ax.grid()
235         self.canvasFrame._ax.figure.canvas.draw()
236 
237     def _plot(self):
238         try:
239             data = self.dataQueue.get(block=False)
240         except queue.Empty:
241             pass
242         else:
243             self.update_canvas(data, self.channels)
244         print('Plotting')
245 
246     def stop_thread(self, thread):
247         thread.stop()
248         thread.join()
249         thread = None
250 
251     def _flush_queue(self, queue):
```

```
252             while queue.qsize() != 0:
253                 queue.get()
254 
255         def start(self):
256             if self.isStarted:
257                 self._timer.stop()
258                 self.stop_thread(self.dataThread)
259                 self._flush_queue(self.dataQueue)
260                 if self.saveFrame.fileHandle:
261                     self.saveFrame.fileHandle.close()
262                     self.saveFrame.fileHandle = None
263                     self.stop_thread(self.saveThread)
264                     self._flush_queue(self.saveQueue)
265                 self.isStarted = False
266             else:
267                 self.sampleLen = self.runFrame.get_sample_size()
268                 self.dt = self.runFrame.get_dt()
269                 self._timer = self.canvasFrame._canvas.new_timer(
270                                 self.runFrame.get_fps(), \
271                                 [(self._plot, (), {})])
272                 if self.saveFrame.checkVar.get() and self.saveFrame.pathVar.get():
273                     self.saveFrame.fileHandle = open(self.saveFrame.pathVar.get(), 'w')
274                     self.saveFrame.fileHandle.write('{}{}'.format(\
275                         strftime("%a, %d %b %Y %H:%M:%S +0300", localtime()), '\n'))
276                     self.saveFrame.fileHandle.write('dt={}{}'.format(self.dt, '\n'))
277                     self.saveFrame.fileHandle.write('{} {} {}{}'.format('ch0', 'ch1', 'ch2',
'\n'))
278                     self.saveFrame.fileHandle.flush()
279                       self.dataThread = ChannelThread2(self.dataQueue, self.saveQueue,
self.sampleLen, self.dt)
280                                         self.saveThread = SaveThread(self.saveQueue,
self.saveFrame.fileHandle)
281                     self.saveThread.start()
282                 else:
283                         self.dataThread = ChannelThread(self.dataQueue, self.sampleLen,
self.dt)
284                 self._timer.start()
285                 self.isStarted = True
286                 self.dataThread.start()
287 
288     if __name__ == "__main__":
289         root = tk.Tk()
290         root.wm_title("Осцилограф")
291         app = Application(master=root)
292         app.mainloop()
```

Приклад програми на мові Python.
Головне вікно програми.



2. Індивідуальні завдання

2.1. Створити застосунок для відображення змодельованих сигналів за прикладом лістингу 1.

2.2. Використати застосунок із пункту 2.1 та відобразити на графіку сигнали із шумом. Типи сигналів вибрати із таблиці 1.

Таблиця 1

№. п/п	Індивідуальне завдання
1.	Синус, Косинус, Меандр
2.	Трикутний, Косинус, Меандр
3.	Синус, Косинус, Меандр
4.	Пилкоподібний, Синус, Косинус,
5.	Синус, Пилкоподібний,

	Меандр
6.	Трикутний, Косинус, Меандр
7.	Пилкоподібний, Косинус, Меандр