МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| к.т.н., профессор |  |  |  |  | С.Г.Фомичева |
| должность, уч. степень |  | подпись, дата |  |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| **Использование обработчика исключений при работе с массивами** по дисциплине: Распределенные информационные системы |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 3843 |  |  |  | А.П. Конева |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Цель работы**

Получить навыки работы обработки исключений при работе с массивами, формируемыми в различных потоках процесса и режимах.

1. **Задачи**

1) Создать программный проект с защищенным интерфейсом пользователя. Обработанные исключения фиксировать в текстовом файле с указанием даты и времени проявления исключений, сообщения исключения и стека вызовов.

2) Создать программу, выполняющую персональное задание в соответствии с заданным вариантом, причем ввод и вывод массивов производить с использованием компонента GridView. Интерфейс пользователя должен предлагать выбор способа формирования массива (случайно, случайно с заданной частотой, вручную)

3) При формировании длинных массивов в GridView выводить первые N элементов массива, где N задается пользователем.

4) Преобразования элементов массива выполнять параллельно, используя потоки.

**3. Ход работы**

**Алгоритм**

Для реализации п.1 и п.2 данной лабораторной работы необходимо создать окно «РИС3», на котором реализована возможность выбора размера массива, выбора типа формирования массива (случайно, с заданной частотой, вручную). По нажатию на созданную кнопку «Создать массив» будет сформирован массив выбранной размерности. Каждый новый массив добавляется в последующую новую строку GridView. Обработанные исключения должны фиксироваться в текстовом файле с указанием даты и времени проявления исключений, сообщения исключения и стека вызовов. В соответствии с персональным заданием под вариантом 7 необходимо сформировать из элементов массива А массив D той же размерности по следующему правилу: первые 10 элементов - Di=Ai+i, остальные - Di=Ai-i. Заменить минимальный по модулю положительный элемент массива А нулем. Заменить знаки элементов с k1-го по k2-й на обратные.

Для реализации п.2 необходимо создать форму ScrolledText в многострочном режиме для лог-журнала, куда после нажатия на кнопку XOR записываются также исключения из всех форм проекта, а также текущая дата и время выполнения проекта.

Для реализации п.3 будет сформирована форма GridView из п.2.

Для реализации п.4 воспользуемся библиотекой threating.

Дополнительное задание: сравнить время выполнения программы с использованием потоков и без их использования.

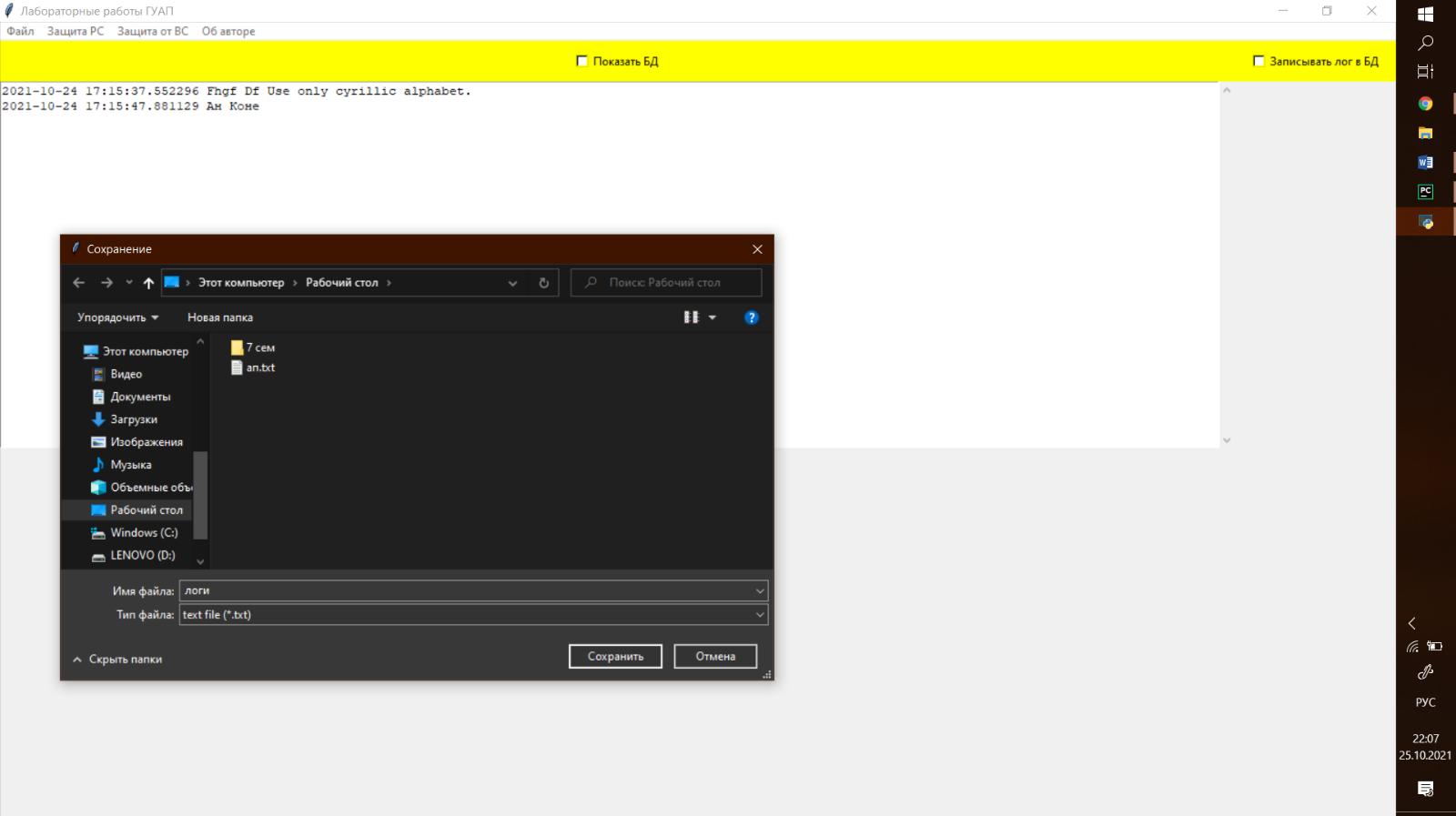
Реализованное в лабораторной работе №1 окно «Лабораторные работы ГУАП» дополнили функционалом сохранения текстового документа с логами(рис.1), открытием содержания выбранного текстового документа в том же окне(рис.2) и созданием нового текстового файла с логами, реализованный путем очищения текущего файла в окне «Лабораторные работы ГУАП».

Рис.1 Функция сохранения текстового файла с логами

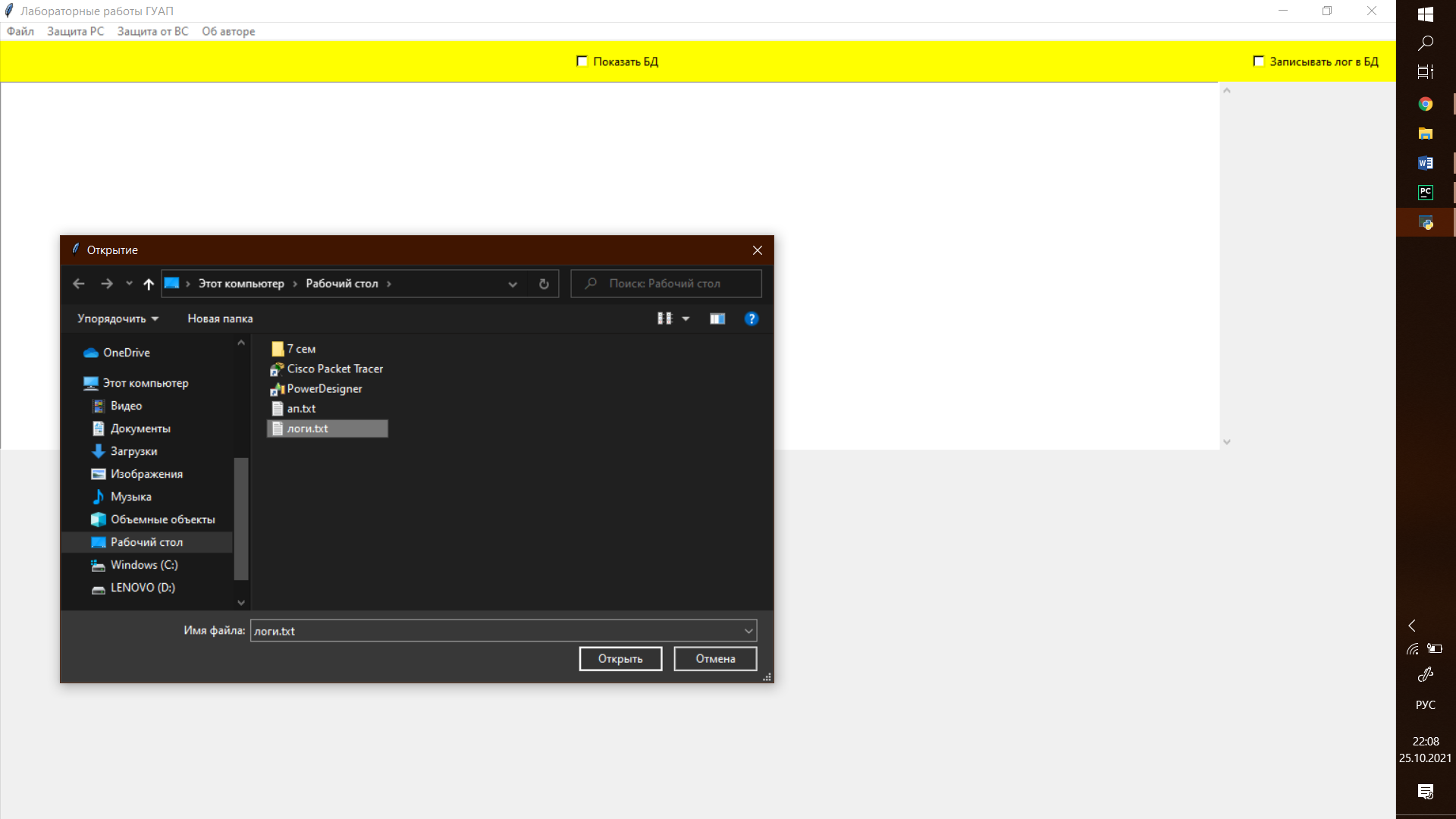


Рис.2 Функция открытия содержания выбранного текстового документа

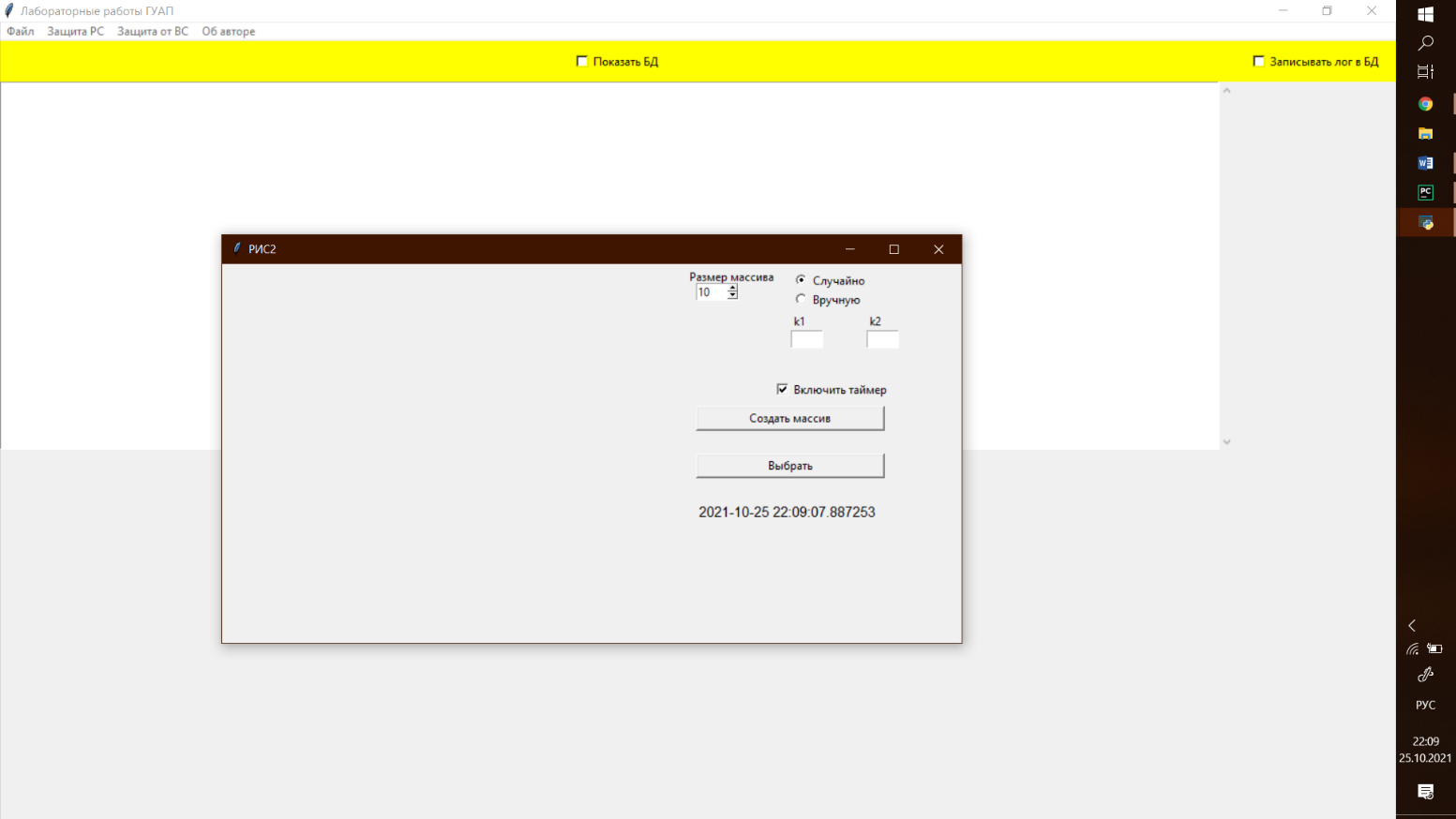
При выборе пункта меню «Защита РС» появляется список Лабораторных работ. При нажатии на РИС2 открывается «РИС2» (рис. 4), содержание которой описано выше в **Алгоритме.**

Рис.4 «РИС2»

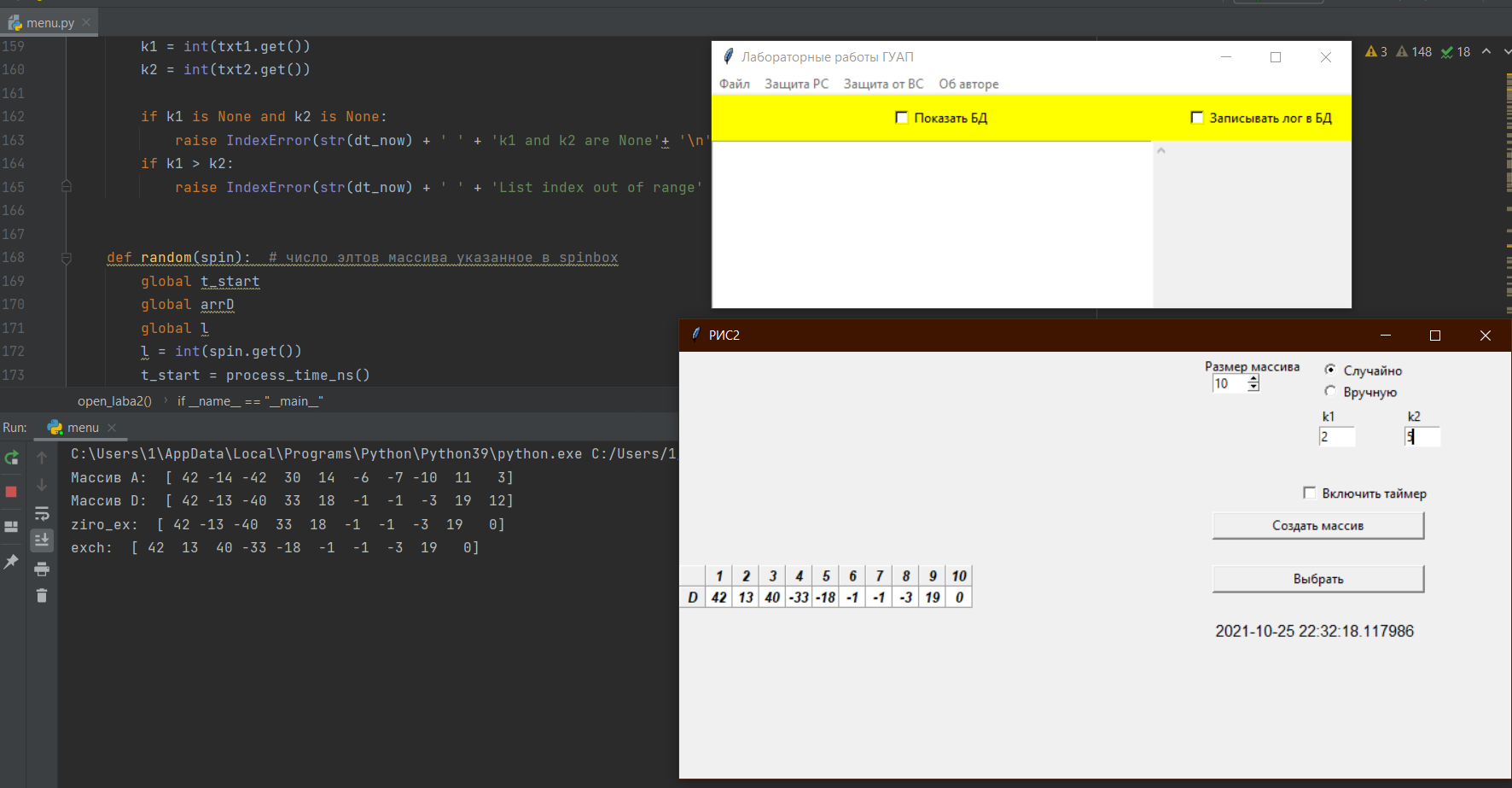
 На рис.5 представлен один из результатов работы программы: сформирован массив заданной размерности из случайных чисел, выполнены соответствующие преобразования по индивидуальному заданию. Убедимся в правильном выполнении, проследив поэтапные преобразования исходного массива в консоли.

Рис.5 Массив из случайных чисел

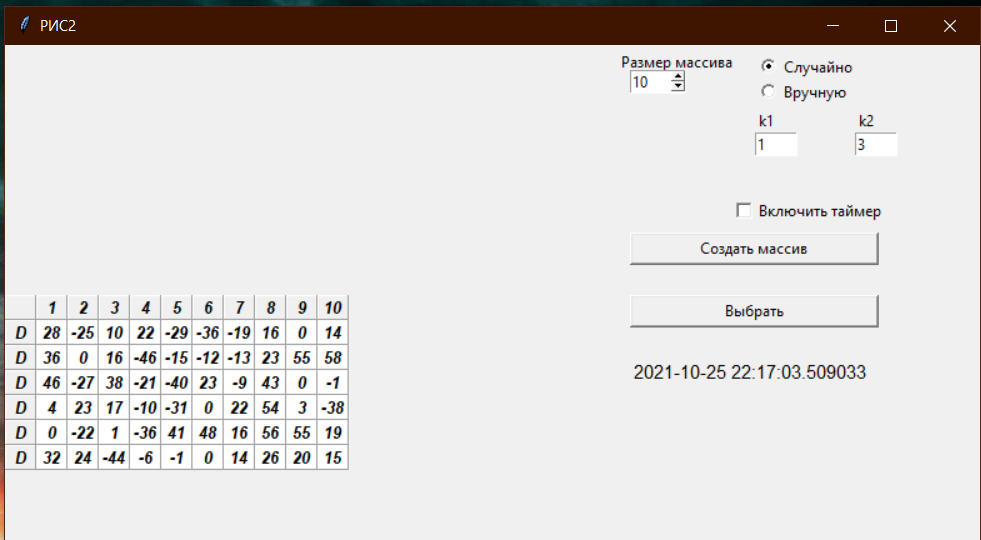
На рис.6 изображен массив, сформированный с заданной частотой (при установленном значении Checkbutton) по умолчанию частотой в 2 c. для каждого нового массива выполнены соответствующие преобразования по индивидуальному заданию.

Рис.6 Массив с заданной частотой

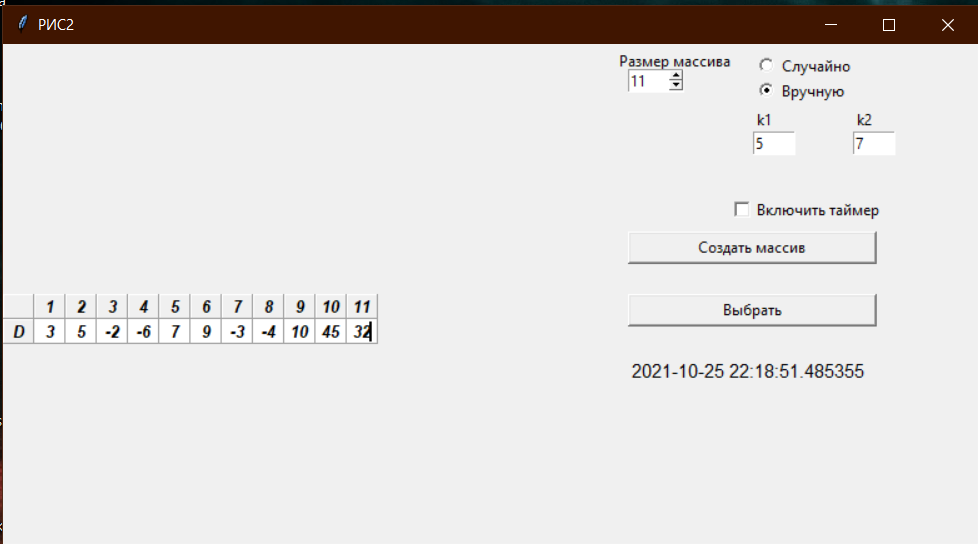
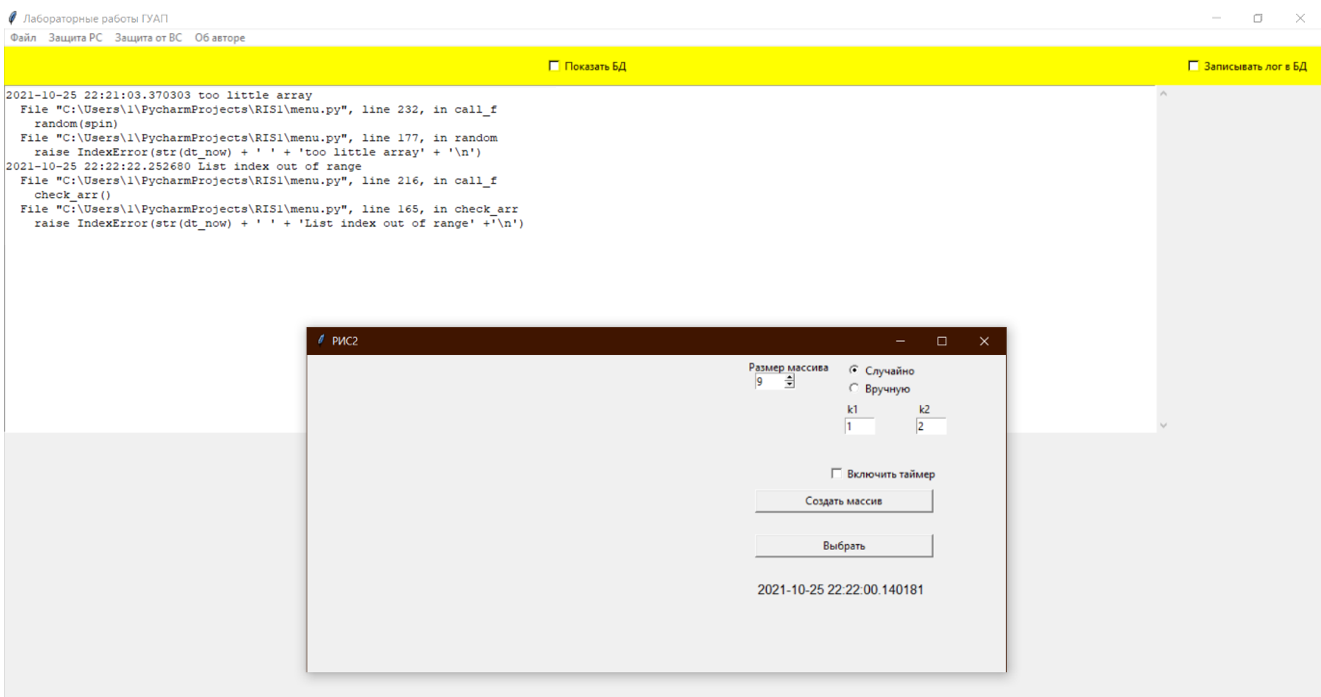
На рис.7 изображен массив, заданный вручную.

Рис.7 Массив, заданный вручную

На рис.8 изображены некоторые логи с вариантами обработки исключений в случае ввода: 1) размера массива, меньше необходимого по заданию (т.е. меньше 10 элементов) - ошибка «too little array»; 2) k1 > k2 для интервала (k1, k2).

~~~~Рис.8 Демонстрация вариантов обработки исключений

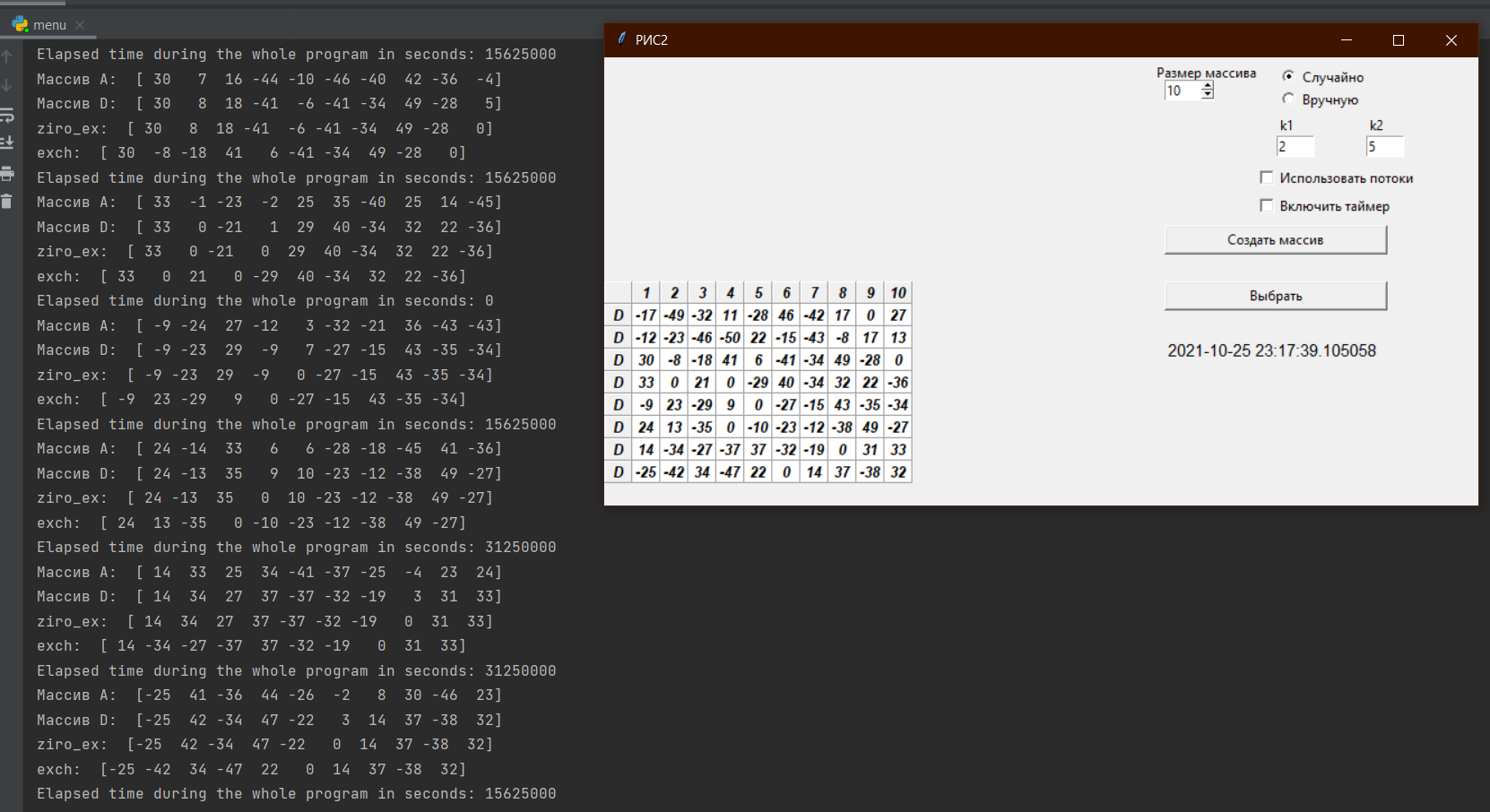
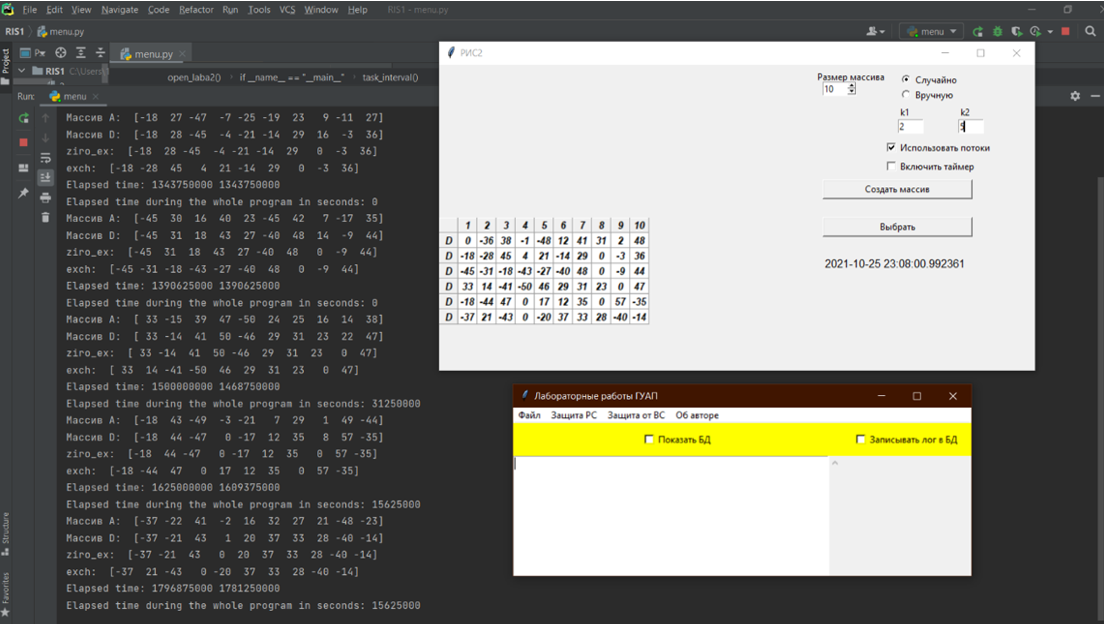
Для сравнения скорости работы программы с использованием потоков и без них добавили на форму «РИС2» дополнительный Checkbutton под названием «Использовать потоки». Был выбран программный модуль, отвечающий за формирование массива с заданной частотой. На рис.9 представлены результаты времени выполнения выбранного модуля без использования потоков.

Рис.9 Результаты времени формирования массивов с заданной частотой без использования потоков

Рис.10 Результаты времени формирования массивов с заданной частотой с использованием потоков

В результате сравнения получили, что использование потоков повышает скорость выполнения программного модуля.

**Листинг программы**

**arr\_laba.py**

def open\_laba2():  
 arr = []  
 def check\_arr():  
 global dt\_now  
 global k1  
 global k2  
 le = int(spin.get())  
 dt\_now = datetime.datetime.now()  
 k1 = int(txt1.get())  
 k2 = int(txt2.get())  
  
 if k1 is None and k2 is None:  
 raise IndexError(str(dt\_now) + ' ' + 'k1 and k2 are None'+ '\n')  
 if k1 > k2:  
 raise IndexError(str(dt\_now) + ' ' + 'List index out of range' +'\n')  
  
  
 def random(spin): # число элтов массива указанное в spinbox  
  
 global arrD  
 global l  
 l = int(spin.get())  
 #t\_start = process\_time\_ns()  
 arrA = np.random.randint(-50, 50, int(spin.get()))  
  
 if l < 10:  
 raise IndexError(str(dt\_now) + ' ' + 'too little array' + '\n')  
 arrD = np.zeros(l, int)  
 for i in range(0, 10):   
 arrD[i] = arrA[i] + i  
 for i in range(10, l):  
 arrD[i] = arrA[i] - i  
 print('Массив А: ', arrA)  
 print('Массив D: ', arrD)  
  
 arr.append(arrD)  
 return (arrA, arrD)  
  
 def emptyArr(spin):  
 arrA = ['' for i in range(int(spin.get()))]  
 arr.append(arrA)  
 print(arrA)  
 return (arrA)  
  
 def ziro\_ex():  
 global t\_start  
 t\_start = process\_time\_ns()  
 min = abs(arrD[0])  
 for i in range(l):  
 if (arrD[i] > 0) and abs(arrD[i]) < min:  
 min = abs(arrD[i])  
 for i in range(l):  
 if arrD[i] == min:  
 arrD[i] = 0  
 print('ziro\_ex: ', arrD)  
 return (arrD)  
  
 def exch():  
 k1= int(txt1.get())-1  
 k2 = int(txt2.get())  
 for i in range(k1, k2):  
 arrD[i] = -(arrD[i])  
 print('exch: ', arrD)  
 return(arrD)  
  
 def call\_f():  
 try:   
 check\_arr()  
 except IndexError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 try:  
 random(spin)  
 except IndexError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
  
 ziro\_ex()  
 exch()  
  
 threadscalc()  
  
  
 textFont1 = ("Arial", 10, "bold italic")  
 textFont2 = ("Arial", 12, "bold")  
 textFont3 = ("Arial", 8, "bold")  
  
 class LabelWidget(Entry):  
 def \_\_init\_\_(self, master, x, y, text):  
 self.text = StringVar(master=master)  
 self.text.set(text)  
 Entry.\_\_init\_\_(self, master=master)  
 self.config(relief="ridge", font=textFont1,  
 justify='center', width=3,  
 textvariable=self.text,  
 state="readonly")  
 self.grid(column=x, row=y)  
  
 class EntryWidget(Entry):  
 def \_\_init\_\_(self, master, x, y, val):  
  
 Entry.\_\_init\_\_(self, master=master)  
 self.value = StringVar(master=master)  
 self.config(textvariable=self.value, width=3,  
 relief="ridge", font=textFont1,  
 justify='center')  
 self.grid(column=x, row=y)  
 self.value.set(val)  
  
 class EntryGrid(Tk):  
 *''' Dialog box with Entry widgets arranged in columns and rows.'''* def \_\_init\_\_(self, root2, colList, rowList, arrD):  
  
 self.root2 = root2  
 self.cols = colList[:]  
 self.colList = colList[:]  
 self.colList.insert(0, "")  
 self.rowList = rowList  
 self.root2.grid()  
 self.make\_header()  
 self.gridDict = {}  
  
 for j in range(len(self.rowList)):  
 for i in range(0, len(arrD[j])):  
 if i > len(arrD[j]):  
 raise IndexError(str(dt\_now) + ' ' + 'List index out of range' + '\n')  
 w = EntryWidget(self.root2, i+1, j + 1, arrD[j][i])  
 self.gridDict[(i, j)] = w.value  
  
 def handler(event, col=i, row=j):  
 return self.\_\_entryhandler(col, row)  
  
 w.bind(sequence="<FocusOut>", func=handler)  
  
 def make\_header(self):  
 self.hdrDict = {}  
 for i, label in enumerate(self.colList):  
 def handler(event, col=i, row=0, text=label):  
 return self.\_\_headerhandler(col, row, text)  
  
 w = LabelWidget(self.root2, i, 0, label)  
 self.hdrDict[(i, 0)] = w  
 w.bind(sequence="<KeyRelease>", func=handler)  
  
 for i, label in enumerate(self.rowList):  
 def handler(event, col=0, row=i + 1, text=label):  
 return self.\_\_headerhandler(col, row, text)  
  
 w = LabelWidget(self.root2, 0, i + 1, label)  
 self.hdrDict[(0, i + 1)] = w  
 w.bind(sequence="<KeyRelease>", func=handler)  
  
 def \_\_entryhandler(self, col, row):  
 s = self.gridDict[(col, row)].get()  
  
 if s.upper().strip() == "EXIT":  
 self.destroy()  
 elif s.upper().strip() == "DEMO":  
 self.demo()  
  
 def demo(self):  
 *''' enter a number into each Entry field '''* for i in range(len(self.cols)):  
 for j in range(len(self.rowList)):  
 sleep(0.25)  
 self.set(i, j, "")  
 self.update\_idletasks()  
 sleep(0.1)  
 self.set(i, j, i + 1 + j)  
 self.update\_idletasks()  
  
 def \_\_headerhandler(self, col, row, text):  
 *''' has no effect when Entry state=readonly '''* self.hdrDict[(col, row)].text.set(text)  
  
 def get(self, x, y):  
 return self.gridDict[(x, y)].get()  
  
 def set(self, x, y, v):  
 self.gridDict[(x, y)].set(v)  
 return v  
  
 def getArrFG(self, rowN, colN):  
 tempArr = []  
 for x in range(colN):  
 tempArr.append(int(self.gridDict[(x+1, len(rowN)-1)].get()))  
  
 arr[-1] = tempArr  
 #threadscalc()  
  
 return arr  
  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 rows = []  
 root2 = Tk()  
 root2.geometry('780x400')  
 root2.title("РИС2")  
 lbl = Label(root2, text="Размер массива")  
 lbl.place(x=490, y=3)  
 lbl = Label(root2, text="k2")  
 lbl.place(x=680, y=50)  
 lbl = Label(root2, text="k1")  
 lbl.place(x=600, y=50)  
 txt1 = Entry(root2, width=5)  
 txt1.place(x=600, y=70)  
 txt2 = Entry(root2, width=5)   
 txt2.place(x=680, y=70)  
  
 datt = Label(root2, font=("helvetica", 11))  
 datt.place(x=500, y=250)  
  
 # create a scrollbar widget and set its command to the text widget  
  
 selected = IntVar(master=root2)  
 selected.set(1)  
 rad1 = Radiobutton(root2, text='Случайно', variable=selected, value=1)  
 rad1.place(x=600, y=5)  
 rad2 = Radiobutton(root2, text='Вручную', variable=selected, value=2)  
 rad2.place(x=600, y=25)  
 spinVar = StringVar(master=root2)  
 spinVar.set("10")  
 spin = Spinbox(root2, from\_=10, to=30, width=5, textvariable=spinVar)  
 spin.place(x=500, y=20)  
 table = Frame(root2)  
  
 def update\_time():  
 datt.config(text=f"{datetime.datetime.now()}")  
 root2.after(100, update\_time)  
  
 def task\_interval():  
 tableview()  
 t\_stop = process\_time\_ns()  
 print("Elapsed time during the whole program in seconds:", t\_stop - t\_start)  
 return arr  
  
 def handle():  
 tableItem.getArrFG(rows, int(spinVar.get()))  
  
  
 def tableview():  
 cols = list(range(1, int(spin.get()) + 1))  
 rows.append('D')  
 global tableItem  
 if int(selected.get()) == 1:  
  
 call\_f()  
 tableItem = EntryGrid(table, cols, rows, arr)  
 elif int(selected.get()) == 2:  
 emptyArr(spin)  
 tableItem = EntryGrid(table, cols, rows, arr)  
 table.place(x=0, y=200)  
 # lArr = arrToList(arr)  
 if chk\_state.get() == True:  
 root2.after(2000, task\_interval)  
 return arr  
 btn = Button(root2, text="Создать массив", width=27, height=1, command=lambda: tableview())  
 btn.place(x=500, y=150)  
 btn = Button(root2, text="Выбрать", width=27, height=1,  
 command=lambda: handle())  
 btn.place(x=500, y=200)  
  
 chk\_state = BooleanVar(master=root2)  
 chk\_state\_thread = BooleanVar(master=root2)  
 chk\_state.set(True)  
 chk\_state\_thread.set(True)# задайте проверку состояния чекбокса  
 chk = Checkbutton(root2, text='Включить таймер', var=chk\_state)  
 chk.place(x=580, y=120)  
 chk = Checkbutton(root2, text='Использовать потоки', var=chk\_state\_thread)  
 chk.place(x=580, y=95)  
 update\_time()  
  
  
 def threadscalc():  
 thread1 = Thread(target=ziro\_ex, args=[arrD])  
 thread2 = Thread(target=exch, args=[arrD])  
 thread1.start()  
 thread2.start()  
 print('da')  
 thread1.join()  
 thread2.join()  
  
 root2.mainloop()

**menu.py**

from arr\_laba import \*  
from time import sleep  
import numpy as np  
from threading import Thread  
from tkinter.ttk import Radiobutton  
from tkinter.ttk import Checkbutton  
from tkinter import filedialog  
from tkinter import messagebox  
from tkinter import scrolledtext  
import datetime  
from tkinter import \*  
import re  
import traceback  
import sys  
from time import process\_time\_ns  
import time  
  
  
def info():  
 messagebox.showinfo("Об авторе", "Конева Анна Павловна\nИнститут инновационных технологий в электромеханике и робототехнике\nГруппа 3843")  
  
window = Tk()  
window.title("Лабораторные работы ГУАП")  
window.geometry('600x200+100+130')  
window.grid\_columnconfigure(0, weight=1)  
  
#чекбоксы  
show\_bd = IntVar()  
show\_bd\_checkbutton = Checkbutton(text="Показать БД", variable=show\_bd, onvalue=1, offvalue=0, padx=15, pady=10, bg = 'yellow')  
show\_bd\_checkbutton.grid(row=0, column=0, sticky=EW )  
  
wr\_logs = IntVar()  
wr\_logs\_checkbutton = Checkbutton(text="Записывать лог в БД", variable=wr\_logs, onvalue=1, offvalue=0, padx=15, pady=10, bg = 'yellow')  
wr\_logs\_checkbutton.grid(row=0, column=1, sticky=EW)  
  
  
logs = scrolledtext.ScrolledText(window, width=500)  
logs.grid(row=1, column=0, sticky=EW )

def save():  
 my\_doc = logs.get("1.0", END) # read from one text box t1  
 fob = filedialog.asksaveasfile(filetypes=[('text file', '\*.txt')],  
 defaultextension='.txt', initialdir='C:\\Users\1\Desktop\7 сем',  
 mode='w')  
 try:  
 fob.write(my\_doc)  
 fob.close()  
 logs.delete('1.0', END) # Delete from position 0 till end  
 logs.update()  
 except:  
 print(" Error initialdir")  
  
def open\_file():  
 file = filedialog.askopenfile()  
 logs.insert(INSERT, file.read())  
  
def create\_file():  
 logs.delete('1.0', END) # Delete from position 0 till end  
 logs.update()  
  
menuBar = Menu(window)  
  
fileMenu = Menu(menuBar, tearoff=0)  
cryptMenu= Menu(fileMenu, tearoff=0)  
cryptMenu.add\_command(label="Эль-Гамаль")  
cryptMenu.add\_command(label="RSA")  
encryptMenu= Menu(fileMenu, tearoff=0)  
encryptMenu.add\_command(label="Эль-Гамаль")  
encryptMenu.add\_command(label="RSA")  
  
fileMenu.add\_command(label="Создать", command = create\_file)  
fileMenu.add\_command(label="Открыть", command = open\_file)  
fileMenu.add\_command(label="Сохранить",command = save)  
fileMenu.add\_separator()  
fileMenu.add\_cascade(label="Шифрование", menu=cryptMenu)  
fileMenu.add\_cascade(label="Дешифрование", menu=encryptMenu)  
fileMenu.add\_separator()  
fileMenu.add\_command(label="Закрыть Alt+X", command=window.destroy)  
  
RSmenu = Menu(menuBar, tearoff=0)  
RSmenu.add\_command(label="РИС1", command=open\_laba1)  
RSmenu.add\_command(label="РИС2", command=open\_laba2)  
RSmenu.add\_command(label="РИС3")  
RSmenu.add\_command(label="РИС4")  
RSmenu.add\_command(label="РИС5")  
  
prMenu = Menu(menuBar, tearoff=0) #protect  
prMenu.add\_command(label="ЗВС1")  
prMenu.add\_command(label="ЗВС2")  
  
menuBar.add\_cascade(label="Файл", menu=fileMenu)  
menuBar.add\_cascade(label="Защита РС", menu=RSmenu) #Raspred Systems  
menuBar.add\_cascade(label="Защита от ВС", menu=prMenu) #protect  
menuBar.add\_cascade(label="Об авторе", command=info)  
  
window.config(menu=menuBar)  
window.mainloop()

**Вывод**

Создали программный проект с защищенным интерфейсом пользователя(рис.4). Обработанные исключения фиксируются в текстовом файле с указанием даты и времени проявления исключений, сообщения исключения и стека вызовов(рис.8)

Реализованный на рис.4 интерфейс предоставляет пользователю выбор способа формирования массива (случайно, случайно с заданной частотой, вручную) и позволяет выполнять задание в соответствие с заданным вариантом, а именно - сформировать из элементов массива А массив D той же размерности по следующему правилу: первые 10 элементов - Di=Ai+i, остальные - Di=Ai-I; заменить минимальный по модулю положительный элемент массива А нулем; заменить знаки элементов с k1-го по k2-й на обратные.

Преобразования элементов массива выполняются параллельно, с использованием потоков.