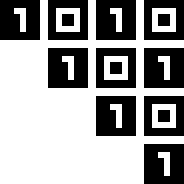
Министерство науки и высшего образования РФ

ФГБОУ ВО «Тамбовский государственный технический университет»

Кафедра

«САПР»



Отчет по лабораторной работе №1

по дисциплине «Алгоритмизация, модели данных и основы ИИ»

Выполнил: студент группы БВТ241

Белов А.К.

Проверил: преподаватель

Патутин К. И.

Тамбов 2025

**Цель работы:**

Освоить основы работы с базовой системой ввода-вывода в языке С++.

Вариант №*2*

**Задание 1**:

Дан текстовый файл. Удалить из него третью строку. Результат записать в другой файл;

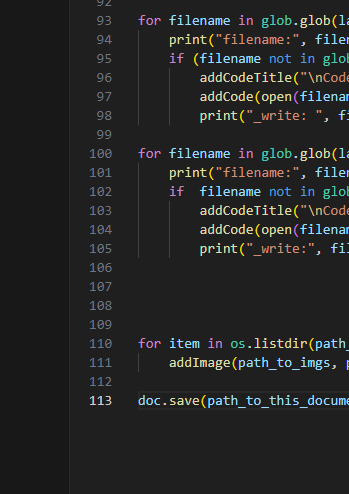
**Решение:**

Для решения использовались 3 функции, помогающие во взаимодействии с файлами (open, read и close будут представлены ниже с другими вспомогательными).

**Листинг программы:**

**Результаты работы программы:**

**Ссылка на весь проект:**

**Цель работы:**1  
 **Заданиe:**1  
 **Решение:  
  
Из файла ...\_Graphics\laba\_logging.py.cpp**import os  
import time  
  
LOG\_FILE = 'loggs.txt'   
inited = []  
def InitLogFile():  
 with open(LOG\_FILE, "w", encoding="utf-8") as f:  
 # fcntl.flock(f, fcntl.LOCK\_EX) # Р‘Р»РѕРєРёСЂРѕРІРєР° С„Р°Р№Р»Р°  
 f.write("=== Start logging ===\n")  
 InitFile(\_\_file\_\_)  
 # fcntl.flock(f, fcntl.LOCK\_UN) # Р Р°Р·Р±Р»РѕРєРёСЂРѕРІРєР°  
  
  
def InitFile(filename):  
 with open(LOG\_FILE, "a+", encoding="utf-8") as f:  
 print("add - "+filename)  
 if filename not in inited:  
 f.write("-init|" + str(filename) + "\n")   
 inited.append(filename)  
 time.sleep(0.1)  
  **Из файла ...\_5\_Graphics\task1\main.py.cpp**import sys  
import os  
  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitLogFile()  
InitFile(\_\_file\_\_)  
  
  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
from graphic import MultiPlotApp  
  
  
def main():  
 root = tk.Tk()  
 app = MultiPlotApp(root, "window", "D:\\projects\\VisualStudioCode\\Laba\_2\_1\_5\_Graphics\\task1\\task1\_data.txt")  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main() **Из файла ...2\_1\_5\_Graphics\graphic.py.cpp**  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import tkinter as tk  
from tkinter import ttk  
  
import sys  
import os  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitFile(\_\_file\_\_)  
  
class MultiPlotApp:  
 def \_\_init\_\_(self, root, name:str, data):  
 self.root = root  
 self.root.title(name)  
 self.root.geometry("1200x800")  
   
 self.categories\_name = 'РєР°С‚РµРіРѕСЂРёРё'  
 self.values\_name = 'Р·РЅР°С‡РµРЅРёСЏ'  
 # РџСЂРёРјРµСЂ РґР°РЅРЅС‹С…  
 self.categories = ['A', 'B', 'C', 'D', 'E']  
 self.values = [23, 45, 56, 78, 33]  
 # self.colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#F9A826', '#6C5CE7']  
   
 self.create\_widgets()  
 self.create\_plots()  
 self.current\_index = 1  
 self.categories\_list = [['q', 'w', 'e', 'e'], ['r', 't', 'y', 'u'], ['i', 'o', 'p', '[]'], ['a', 's', 'd', 'f'], ['g', 'h', 'j', 'k']]  
 self.values\_list = [[12, 32, 64, 15], [48, 63, 15, 36], [48, 95, 62, 51], [42, 51, 54, 36], [75, 35, 42, 63]]  
 self.switch\_data()  
  
  
 #data РёР·СЉСЏС‚РёРµ  
 with open(data, "r", encoding="utf-8") as f:  
 read\_data = f.read().split('-')  
 print(read\_data)  
 self.categories\_list.clear()  
 self.values\_list.clear()  
 for i in range(1, len(read\_data), 2):  
 part\_data\_categories = []  
 part\_data\_values = []  
 print(read\_data[i] + " - " + read\_data[i+1])  
  
 for items in (read\_data[i+1].split('\n')):  
 item = items.split(' ')  
 if(items == ''):  
 continue  
  
 print(item)  
 print(f'|{item[0]}|{item[1]}|')  
  
 part\_data\_categories.append(item[0] + " " + read\_data[i])  
 part\_data\_values.append(int(item[1]))  
 self.categories\_list.append(part\_data\_categories)  
 self.values\_list.append(part\_data\_values)  
 print("all ------------")  
 print(part\_data\_categories)  
 print(part\_data\_values)  
   
  
 def set\_categories(self, data):  
 self.categories = data  
  
 def set\_values(self, data):  
 self.values = data  
  
 def create\_widgets(self):  
 control\_frame = ttk.Frame(self.root, padding="10")  
 control\_frame.grid(row=0, column=0, sticky="ew")  
   
 plot\_frame = ttk.Frame(self.root, padding="10")  
 plot\_frame.grid(row=1, column=0, sticky="nsew")  
   
 self.root.grid\_rowconfigure(1, weight=1)  
 self.root.grid\_columnconfigure(0, weight=1)  
   
 ttk.Label(control\_frame, text="РўРёРї РіСЂР°С„РёРєР°:").grid(row=0, column=0, padx=5)  
   
 self.plot\_type = tk.StringVar(value="linear")  
 plot\_types = [  
 ("Р›РёРЅРµР№РЅС‹Р№ РіСЂР°С„РёРє", "linear"),  
 ("РўРѕС‡РµС‡РЅС‹Р№ РіСЂР°С„РёРє", "scatter"),  
 ("РЎС‚РѕР»Р±С‡Р°С‚Р°СЏ РґРёР°РіСЂР°РјРјР°", "bar"),  
 ("Р“РѕСЂРёР·РѕРЅС‚Р°Р»СЊРЅР°СЏ РґРёР°РіСЂР°РјРјР°", "barh"),  
 ("РљСЂСѓРіРѕРІР°СЏ РґРёР°РіСЂР°РјРјР°", "pie"),  
 ("Р“РёСЃС‚РѕРіСЂР°РјРјР°", "hist")  
 ]  
   
 for i, (text, value) in enumerate(plot\_types):  
 ttk.Radiobutton(control\_frame, text=text, variable=self.plot\_type,   
 value=value, command=self.update\_plot).grid(row=0, column=i+1, padx=5)  
   
 # РљРЅРѕРїРєР° РѕР±РЅРѕРІР»РµРЅРёСЏ  
 ttk.Button(control\_frame, text="<-", command=self.switch\_data\_l).grid(row=0, column=len(plot\_types)+1, padx=5)   
 # РљРЅРѕРїРєР° РѕР±РЅРѕРІР»РµРЅРёСЏ  
 ttk.Button(control\_frame, text="->", command=self.switch\_data\_r).grid(row=0, column=len(plot\_types)+2, padx=5)  
  
 self.figure = plt.Figure(figsize=(10, 6), dpi=100)  
 self.canvas = FigureCanvasTkAgg(self.figure, plot\_frame)  
 self.canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)  
   
 def create\_plots(self):  
 self.update\_plot()  
   
 def switch\_data\_l(self):  
 self.current\_index = max(self.current\_index - 1, 1)  
 self.switch\_data()  
  
 def switch\_data\_r(self):  
 self.current\_index = min(self.current\_index + 1, len(self.categories\_list)-2)  
 self.switch\_data()  
  
 def switch\_data(self):  
 self.set\_categories(self.categories\_list[self.current\_index-1]+self.categories\_list[self.current\_index]+self.categories\_list[self.current\_index+1])  
 self.set\_values(self.values\_list[self.current\_index-1]+self.values\_list[self.current\_index]+self.values\_list[self.current\_index+1])  
 self.update\_plot()  
  
 def update\_plot(self):  
  
  
 plot\_type = self.plot\_type.get()  
 self.figure.clear()  
   
 ax = self.figure.add\_subplot(111)  
   
 if plot\_type == "linear":  
 self.create\_linear\_plot(ax)  
 elif plot\_type == "scatter":  
 self.create\_scatter\_plot(ax)  
 elif plot\_type == "bar":  
 self.create\_bar\_plot(ax)  
 elif plot\_type == "barh":  
 self.create\_barh\_plot(ax)  
 elif plot\_type == "pie":  
 self.create\_pie\_plot(ax)  
 elif plot\_type == "hist":  
 self.create\_hist\_plot(ax)  
   
 self.figure.tight\_layout()  
 self.canvas.draw()  
   
 def create\_linear\_plot(self, ax):  
 x = np.arange(len(self.values))  
 ax.plot(x, self.values, marker='o', linestyle='-', linewidth=2, markersize=8, color='blue')  
 ax.set\_title('Р›РёРЅРµР№РЅС‹Р№ РіСЂР°С„РёРє', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel(self.values\_name, fontsize=12)  
 ax.grid(True, alpha=0.3)  
 ax.set\_xticks(x)  
 ax.set\_xticklabels(self.categories)  
   
 def create\_scatter\_plot(self, ax):  
 x = np.arange(len(self.values))  
 scatter = ax.scatter(x, self.values, s=100, c=self.values, cmap='viridis', alpha=0.7, edgecolors='black')  
 ax.set\_title('РўРѕС‡РµС‡РЅС‹Р№ РіСЂР°С„РёРє', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel(self.values\_name, fontsize=12)  
 ax.grid(True, alpha=0.3)  
 ax.set\_xticks(x)  
 ax.set\_xticklabels(self.categories)  
   
 plt.colorbar(scatter, ax=ax, label=self.values\_name)  
   
 def create\_bar\_plot(self, ax):  
 bars = ax.bar(self.categories, self.values, alpha=0.7, edgecolor='black')  
 ax.set\_title('РЎС‚РѕР»Р±С‡Р°С‚Р°СЏ РґРёР°РіСЂР°РјРјР°', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.categories\_name, fontsize=12)  
 ax.set\_ylabel(self.values\_name, fontsize=12)  
 ax.grid(True, alpha=0.3, axis='y')  
   
 for bar, value in zip(bars, self.values):  
 height = bar.get\_height()  
 ax.text(bar.get\_x() + bar.get\_width()/2., height + 1,  
 f'{value}', ha='center', va='bottom', fontweight='bold')  
   
 def create\_barh\_plot(self, ax):  
 bars = ax.barh(self.categories, self.values, alpha=0.7, edgecolor='black')  
 ax.set\_title('Р“РѕСЂРёР·РѕРЅС‚Р°Р»СЊРЅР°СЏ РґРёР°РіСЂР°РјРјР°', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.values\_name, fontsize=12)  
 ax.set\_ylabel(self.categories\_name, fontsize=12)  
 ax.grid(True, alpha=0.3, axis='x')  
   
 for bar, value in zip(bars, self.values):  
 width = bar.get\_width()  
 ax.text(width + 1, bar.get\_y() + bar.get\_height()/2.,  
 f'{value}', ha='left', va='center', fontweight='bold')  
   
 def create\_pie\_plot(self, ax):  
 wedges, texts, autotexts = ax.pie(  
 self.values,   
 labels=self.categories,   
 autopct='%1.1f%%',  
 startangle=90,  
 shadow=True,  
 explode=[0.05] \* len(self.values)  
 )  
   
 ax.set\_title('РљСЂСѓРіРѕРІР°СЏ РґРёР°РіСЂР°РјРјР°', fontsize=16, fontweight='bold')  
   
 for autotext in autotexts:  
 autotext.set\_color('white')  
 autotext.set\_fontweight('bold')  
   
 def create\_hist\_plot(self, ax):  
 ax.hist(self.values, bins=10, color='skyblue', edgecolor='black', alpha=0.7)  
 ax.set\_title('Р“РёСЃС‚РѕРіСЂР°РјРјР° СЂР°СЃРїСЂРµРґРµР»РµРЅРёСЏ Р·РЅР°С‡РµРЅРёР№', fontsize=16, fontweight='bold')  
 ax.set\_xlabel(self.values\_name, fontsize=12)  
 ax.set\_ylabel("РєРѕР»РёС‡РµСЃС‚РІРѕ", fontsize=12)  
 ax.grid(True, alpha=0.3)  
  
def main():  
 root = tk.Tk()  
 app = MultiPlotApp(root, "window", "D:\\projects\\VisualStudioCode\\Laba\_2\_1\_5\_Graphics\\task1\\task1\_data.txt")  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
Снимок1.PNG:  
Снимок2.PNG: