Студентка группы ИС-23 Васильева А. Н.

**Практическое занятие №17**

**Тема:** составление программ с использованием GUI Tkinter в IDE PyCharm Community, изучение возможностей модуля OS

**Цель**: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с использованием GUI Tkinter в IDE PyCharm Community, изучить возможности модуля OS.

**Постановка задачи №1:** Разработать программу с применением пакета tk, взяв в качестве условия одну любую задачу из ПЗ №№ 2 – 9

**Тип алгоритма**: циклический

**Текст программы:**

import tkinter as tk  
  
def calculate\_distance():  
 v1 = float(entry\_v1.get())  
 v2 = float(entry\_v2.get())  
 s = float(entry\_s.get())  
 t = float(entry\_t.get())  
  
 initial\_distance = abs(s)  
 total\_distance = t \* (v1 + v2)  
 final\_distance = abs(initial\_distance - total\_distance)  
  
 result\_label.config(text=f"Distance between the cars after {t} hours: {final\_distance} km")  
  
  
root = tk.Tk()  
root.title("Car Distance Calculator")  
  
  
label\_v1 = tk.Label(root, text="Speed of Car 1 (km/h):")  
label\_v1.pack()  
entry\_v1 = tk.Entry(root)  
entry\_v1.pack()  
  
label\_v2 = tk.Label(root, text="Speed of Car 2 (km/h):")  
label\_v2.pack()  
entry\_v2 = tk.Entry(root)  
entry\_v2.pack()  
  
label\_s = tk.Label(root, text="Initial Distance (km):")  
label\_s.pack()  
entry\_s = tk.Entry(root)  
entry\_s.pack()  
  
label\_t = tk.Label(root, text="Time (hours):")  
label\_t.pack()  
entry\_t = tk.Entry(root)  
entry\_t.pack()  
  
calculate\_button = tk.Button(root, text="Calculate Distance", command=calculate\_distance)  
calculate\_button.pack()  
  
result\_label = tk.Label(root, text="")  
result\_label.pack()  
  
root.mainloop()

**Протокол программы:**

Process finished with exit code 0

**Постановка задачи №2:** Задание предполагает, что у студента есть проект с практическими работами (№№ 2-13), оформленный согласно требованиям. Все задания выполняются c использованием модуля OS: ⎫ перейдите в каталог PZ11. Выведите список всех файлов в этом каталоге. Имена вложенных подкаталогов выводить не нужно. ⎫ перейти в корень проекта, создать папку с именем test. В ней создать еще одну папку test1. В папку test переместить два файла из ПЗ6, а в папку test1 - один файл из ПЗ7. Файл из ПЗ7 переименовать в test.txt. Вывести в консоль информацию о размере файлов в папке test. ⎫ перейти в папку с PZ11, найти там файл с самым коротким именем, имя вывести в консоль. Использовать функцию basename () (os.path.basename()). ⎫ перейти в любую папку где есть отчет в формате .pdf и «запустите» файл в привязанной к нему программе. Использовать функцию os.startfile(). ⎫ удалить файл test.txt.

**Тип алгоритма**: циклический

**Текст программы:** import os  
import sys  
import subprocess  
  
project\_root = os.path.abspath('../')  
  
paths = {  
 "pz\_6": os.path.join(project\_root, 'ПЗ№6'),  
 "pz\_7": os.path.join(project\_root, 'ПЗ№7', 'pz\_7\_1.py'),  
 "pz\_11": os.path.join(project\_root, 'ПЗ№11'),  
 "test": os.path.join(project\_root, 'test'),  
 "test1": os.path.join(project\_root, 'test', 'test1'),  
 "test\_file": os.path.join(project\_root, 'test', 'test1', 'test.txt'),  
 "reports": os.path.join(project\_root, 'reports'),  
 "report\_pdf": 'PZ\_7 (1).pdf'  
}  
  
  
def open\_file(filename):  
 if sys.platform == "win32":  
 os.startfile(filename)  
 else:  
 opener = "open" if sys.platform == "darwin" else "xdg-open"  
 subprocess.call([opener, filename])  
  
  
def change\_directory(path):  
 if os.path.exists(path):  
 os.chdir(path)  
 return True  
 else:  
 print(f"Каталог {path} не найден")  
 return False  
  
  
def copy\_file(source, destination):  
 if os.path.exists(source):  
 with open(source, 'rb') as f\_src, open(destination, 'wb') as f\_dst:  
 f\_dst.write(f\_src.read())  
 else:  
 print(f"Файл {source} не найден")  
  
  
def list\_files\_in\_directory(path):  
 if os.path.exists(path):  
 os.chdir(path)  
 return [f for f in os.listdir() if os.path.isfile(f)]  
 else:  
 print(f"Каталог {path} не найден")  
 return []  
  
  
def create\_directory(path):  
 os.makedirs(path, exist\_ok=True)  
  
  
def print\_file\_sizes(directory):  
 if os.path.exists(directory):  
 files = [f for f in os.listdir(directory) if os.path.isfile(os.path.join(directory, f))]  
 for file in files:  
 file\_path = os.path.join(directory, file)  
 print(f"Размер файла {file}: {os.path.getsize(file\_path)} байт")  
 else:  
 print(f"Каталог {directory} не найден")  
  
  
os.chdir(project\_root)  
  
files\_in\_pz11 = list\_files\_in\_directory(paths['pz\_11'])  
  
print("Файлы в каталоге PZ\_11:", files\_in\_pz11)  
  
create\_directory(paths['test1'])  
  
files\_to\_copy = ['pz\_6\_1.py', 'pz\_6\_2.py']  
for file in files\_to\_copy:  
 src = os.path.join(paths['pz\_6'], file)  
 dst = os.path.join(paths['test'], file)  
 copy\_file(src, dst)  
  
copy\_file(paths['pz\_7'], paths['test\_file'])  
  
print\_file\_sizes(paths['test'])  
  
if files\_in\_pz11:  
 shortest\_filename = min(files\_in\_pz11, key=len)  
 print("Файл с самым коротким именем:", os.path.basename(shortest\_filename))  
  
  
if change\_directory(paths['reports']) and os.path.exists(paths['report\_pdf']):  
 open\_file(paths['report\_pdf'])  
else:  
 print(f"PDF файл {paths['report\_pdf']} не найден")  
  
  
if os.path.exists(paths['test\_file']):  
 os.remove(paths['test\_file'])  
 print(f"Файл {paths['test\_file']} успешно удален")  
else:  
 print(f"Файл {paths['test\_file']} не найден для удаления")

**Протокол программы:**

Файлы в каталоге PZ\_11: ['PZ\_11.pdf', 'pz\_11\_1.py', 'pz\_11\_2.py']

Размер файла pz\_6\_1.py: 493 байт

Размер файла pz\_6\_2.py: 382 байт

Файл с самым коротким именем: PZ\_11.pdf

Файл C:\Users\User\Documents\PZ\_24\test\test1\test.txt успешно удален

Process finished with exit code 0

**Вывод:** в процессе выполнения практической работы я закрепила усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрела навыки работы с БД в IDE PyCharm Community.

Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.

Готовые программные коды выложены на GitHub.