**Задачи на создание графического интерфейса**

1. Напишите программу для создания графического интерфейса «Радуга». Окно приложения содержит семь кнопок, цвета которых соответствуют цветам радуги. Над кнопками располагаются метка и текстовое поле, которые в начальном состоянии являются пустыми. При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.

Коды цветов в шестнадцатеричной кодировке: #ff0000 – красный, #ff7d00 – оранжевый, #ffff00 – желтый, #00ff00 – зеленый, #007dff – голубой, #0000ff – синий, #7d00ff – фиолетовый.

Примерный образец выполнения программы (нажата красная кнопка) показан рисунке 1.

Программный код:

from tkinter import \*

main\_window = Tk()

color\_name\_label = Label(main\_window, bg='white', fg='black', width=20)

color\_code\_entry = Entry(main\_window, text="цвет", width=20)

color\_name\_label.pack()

color\_code\_entry.pack()

class ChangeButton:

def \_\_init\_\_(self, color, num):

self.button = Button(bg=color, width=20)

self.button.bind('<Button-1>', self.change)

self.button.pack()

self.color = color

self.num = num

def change(self, event):

color\_name\_label.config(text=self.color)

color\_code\_entry.delete(0, 'end')

color\_code\_entry.insert(1, " " \* 13 + self.num)

red = ChangeButton('red', '#ff0000')

orange = ChangeButton('orange', "#ff7d00")

yellow = ChangeButton('yellow', "#ffff00")

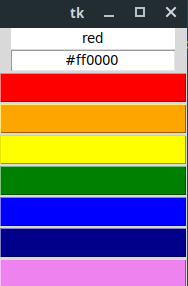
green = ChangeButton('green', "#00ff00")

blue = ChangeButton('blue', "#007dff")

dark\_blue = ChangeButton('dark blue', "#0000ff")

violet = ChangeButton('violet', "#7d00ff")

main\_window.mainloop()



1. Напишите программу для создания графического интерфейса игры «Кирпичи».

Дано *n* кирпичей (генерируется при старте игры каждый раз случайным образом из диапазона от 12 до 20, выводится в область метки). Вы и компьютер ходите поочередно. За ход можно взять 1, 2 или 3 кирпича. Проиграл тот, кому нечего брать. Реализуйте игру с компьютером. Компьютер ходит случайно (без анализа выигрышной стратегии), однако если у него есть ход, который является последним для его выигрыша, то он его совершает. В результате выведите статистику: сколько игр выиграл пользователь. При нажатии на кнопку «Сохранить» данные статистики записываются в файл.

Программный код interface.py

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(777, 502)

MainWindow.setAutoFillBackground(True)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.push\_button\_1 = QtWidgets.QPushButton(self.centralwidget)

self.push\_button\_1.setGeometry(QtCore.QRect(20, 160, 171, 71))

self.push\_button\_1.setFont(QtGui.QFont("Calibri", 25, QtGui.QFont.Bold))

self.push\_button\_1.setMouseTracking(False)

self.push\_button\_1.setObjectName("push\_button\_1")

self.push\_button\_2 = QtWidgets.QPushButton(self.centralwidget)

self.push\_button\_2.setGeometry(QtCore.QRect(20, 260, 171, 71))

self.push\_button\_2.setFont(QtGui.QFont("Calibri", 25, QtGui.QFont.Bold))

self.push\_button\_2.setObjectName("push\_button\_2")

self.push\_button\_3 = QtWidgets.QPushButton(self.centralwidget)

self.push\_button\_3.setGeometry(QtCore.QRect(20, 360, 171, 71))

self.push\_button\_3.setFont(QtGui.QFont("Calibri", 25, QtGui.QFont.Bold))

self.push\_button\_3.setObjectName("push\_button\_3")

self.user\_score = QtWidgets.QLCDNumber(self.centralwidget)

self.user\_score.setGeometry(QtCore.QRect(40, 90, 171, 51))

font = QtGui.QFont()

font.setKerning(True)

self.user\_score.setFont(font)

self.user\_score.setMouseTracking(False)

self.user\_score.setAcceptDrops(False)

self.user\_score.setFrameShape(QtWidgets.QFrame.NoFrame)

self.user\_score.setObjectName("user\_score")

self.comp\_score = QtWidgets.QLCDNumber(self.centralwidget)

self.comp\_score.setGeometry(QtCore.QRect(460, 90, 171, 51))

self.comp\_score.setFrameShape(QtWidgets.QFrame.NoFrame)

self.comp\_score.setObjectName("comp\_score")

self.text\_comp\_step = QtWidgets.QLabel(self.centralwidget)

self.text\_comp\_step.setGeometry(QtCore.QRect(580, 60, 171, 271))

self.text\_comp\_step.setText("Ход компьютера")

self.text\_comp\_step.setFont(QtGui.QFont("Calibri", 13, QtGui.QFont.Bold))

self.comp\_step = QtWidgets.QLabel(self.centralwidget)

self.comp\_step.setGeometry(QtCore.QRect(580, 160, 171, 271))

self.comp\_step.setFrameShape(QtWidgets.QFrame.NoFrame)

self.comp\_step.setTextFormat(QtCore.Qt.AutoText)

self.comp\_step.setAlignment(QtCore.Qt.AlignCenter)

self.comp\_step.setFont(QtGui.QFont("Calibri", 25, QtGui.QFont.Bold))

self.comp\_step.setWordWrap(False)

self.comp\_step.setObjectName("comp\_step")

self.count\_of\_bricks = QtWidgets.QLabel(self.centralwidget)

self.count\_of\_bricks.setGeometry(QtCore.QRect(210, 40, 351, 71))

self.count\_of\_bricks.setTextFormat(QtCore.Qt.AutoText)

self.count\_of\_bricks.setAlignment(QtCore.Qt.AlignCenter)

self.count\_of\_bricks.setFont(QtGui.QFont("Calibri", 13, QtGui.QFont.Bold))

self.count\_of\_bricks.setWordWrap(False)

self.count\_of\_bricks.setIndent(-1)

self.count\_of\_bricks.setObjectName("count\_of\_bricks")

self.user\_icon = QtWidgets.QLabel(self.centralwidget)

self.user\_icon.setGeometry(QtCore.QRect(440, 160, 121, 271))

self.user\_icon.setText("")

self.user\_icon.setPixmap(QtGui.QPixmap("icons8-монитор-100.png"))

self.user\_icon.setObjectName("user\_icon")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(210, 160, 121, 271))

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("icons8-ios-100.png"))

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setEnabled(True)

self.menubar.setGeometry(QtCore.QRect(0, 0, 777, 23))

self.menubar.setAutoFillBackground(False)

self.menubar.setDefaultUp(False)

self.menubar.setNativeMenuBar(True)

Программный код Main.py

from random import randint

from PyQt5 import QtWidgets

from tkinter import \*

import datetime

import interface

list\_logs = ['']

users\_steps = 0

def write\_in\_logs(message):

now = str(datetime.datetime.now())

if message != " ":

list\_logs.append(message + "; " + now)

else:

list\_logs.append(message)

logs = open('logs', 'w')

for line in list\_logs:

logs.write(line + '\n')

print(message)

def write\_in\_stats(user\_steps, comp\_steps, user\_wins, comp\_wins):

stats\_dict = {"Users steps: ": user\_steps, "Computers steps: ": comp\_steps,

"Users wins: ": user\_wins, "Computers wins: ": comp\_wins}

stats = open('stats', 'w')

for key in stats\_dict:

print(key, stats\_dict[key])

stats.write(str(key) + str(stats\_dict[key]) + '\n')

class ExampleApp(QtWidgets.QMainWindow, interface.Ui\_MainWindow):

users\_steps = 0

comp\_steps = 0

comp\_wins = 0

users\_wins = 0

is\_game\_enabled = True

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.counts\_int\_bricks = randint(12, 20)

self.setupUi(self)

self.s = str(self.counts\_int\_bricks)

self.count\_of\_bricks.setText("Количество кирпичей: " + self.s)

self.push\_button\_1.clicked.connect(self.event\_minus\_one)

self.push\_button\_2.clicked.connect(self.event\_minus\_two)

self.push\_button\_3.clicked.connect(self.event\_minus\_three)

self.action\_stats.triggered.connect(self.open\_stats)

self.action\_logs.triggered.connect(self.open\_logs)

@staticmethod

def open\_stats():

stats\_frame = Tk()

listbox = Listbox(stats\_frame, height=100, width=100)

listbox.pack()

listbox.insert(END, "Stats of game")

with open('stats', 'r') as file:

lst = file.readlines()

for item in lst:

listbox.insert(END, item)

mainloop()

@staticmethod

def open\_logs():

frame\_logs = Tk()

listbox = Listbox(frame\_logs, height=100, width=100)

listbox.pack()

listbox.insert(END, "This is logs")

with open('logs', 'r') as file:

lst = file.readlines()

for item in lst:

listbox.insert(END, item)

mainloop()

def game\_again(self):

self.counts\_int\_bricks = randint(12, 20)

self.count\_of\_bricks.setText("Количество кирпичей: " + str(self.counts\_int\_bricks))

self.is\_game\_enabled = True

def make\_step(self, num):

self.users\_steps += 1

if self.is\_game\_enabled:

self.counts\_int\_bricks -= num

self.count\_of\_bricks.setText("")

self.count\_of\_bricks.setText("Количество кирпичей: " + str(self.counts\_int\_bricks))

write\_in\_logs("User makes step")

write\_in\_stats(self.users\_steps, self.comp\_steps, self.users\_wins, self.comp\_steps)

if self.counts\_int\_bricks <= 0:

self.is\_game\_enabled = False

self.users\_wins += 1

self.user\_score.display(str(self.users\_wins))

write\_in\_logs("User win")

write\_in\_logs(" ")

self.game\_again()

if self.counts\_int\_bricks > 0:

self.minus\_comp()

self.comp\_steps += 1

write\_in\_logs("Computer makes step")

if self.counts\_int\_bricks <= 0:

self.is\_game\_enabled = False

self.comp\_wins += 1

self.comp\_score.display(str(self.comp\_wins))

write\_in\_logs("Computer win")

write\_in\_logs(" ")

self.game\_again()

write\_in\_stats(self.users\_steps, self.comp\_steps, self.users\_wins, self.comp\_wins)

def minus\_comp(self):

step = 0

if self.counts\_int\_bricks > 15:

step = randint(1, 3)

else:

if self.counts\_int\_bricks == 15:

step = 3

if self.counts\_int\_bricks == 14:

step = 2

if self.counts\_int\_bricks == 13:

step = 1

if self.counts\_int\_bricks == 12:

step = 2

if self.counts\_int\_bricks == 11:

step = 3

if self.counts\_int\_bricks == 10:

step = 2

if self.counts\_int\_bricks == 9:

step = 1

if self.counts\_int\_bricks == 8:

step = 3

if self.counts\_int\_bricks == 7:

step = 3

if self.counts\_int\_bricks == 6:

step = 2

if self.counts\_int\_bricks == 5:

step = 1

if self.counts\_int\_bricks == 4:

step = randint(1, 3)

if self.counts\_int\_bricks == 3:

step = 3

if self.counts\_int\_bricks == 2:

step = 2

if self.counts\_int\_bricks == 1:

step = 1

self.counts\_int\_bricks -= step

self.comp\_step.setText(str(step))

self.count\_of\_bricks.setText("")

self.count\_of\_bricks.setText("Количество кирпичей: " + str(self.counts\_int\_bricks))

def event\_minus\_one(self):

self.make\_step(1)

def event\_minus\_two(self):

self.make\_step(2)

def event\_minus\_three(self):

self.make\_step(3)

def main():

app = QtWidgets.QApplication(sys.argv) # new QApplication

window = ExampleApp() # Create new object of ExampleApp launch app

window.show()

app.exec()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Интерфейс:

