Государственное бюджетное общеобразовательное учреждение

лицей № 393 Кировского района Санкт-Петербурга

(ГБОУ лицей № 393 Кировского района Санкт-Петербурга)

Руководство программиста

**Генератор задания №8 ЕГЭ**

Автор проекта: Колосов Владимир, ученик 10 «А» класса

Руководитель проекта: Смирнова Надежда Алексеевна, учитель информатики

Санкт-Петербург

2024

Содержание

[Аннотация 3](#_Toc98109664)

[Постановка задачи 4](#_Toc98109665)

[Формализация алгоритма 5](#_Toc98109666)

[Листинг программы 6](#_Toc98109667)

[Тестовые примеры 23](#_Toc98109668)

[Описание размещения 24](#_Toc98109669)

[Требования к программным и аппаратным средствам 25](#_Toc98109670)

# Аннотация

Данный документ представляет собой краткое описание самой программы, её интерфейсов, размещения, требования к аппаратным и программным средствам.

Проект "Генератор заданий номер 8 ЕГЭ по информатике на Python" представляет собой программу, которая генерирует случайные задания из раздела номер 8 ЕГЭ по информатике и выводит их пользователю. Программа написана на языке программирования Python и использует различные математические операции, условные операторы и циклы для создания разнообразных заданий. Проект поможет учащимся подготовиться к сдаче ЕГЭ по информатике, практикуя решение различных задач

# Постановка задачи

Целью моего проекта было создать программу, которая будет автоматически генерировать уникальные задания номер 8 ЕГЭ по информатике, так же необходимо реализовать графический интерфейс с помощью библиотеки tkinter для того, чтобы пользователю было удобнее пользоваться этой программой.

# Формализация алгоритма

1. Библиотека tkinter для реализации графического интерфейса программы.
2. Библиотека random для генерации и перемешивания чисел.
3. Библиотека itertools для генерации задач и алгоритма проверки.
4. Библиотека sys, os для реализации кнопок “Выход”, “Начальное окно”.
5. Библиотека “platform” для возможности запуска на других ОС.
6. Класс Generate, для проверки и генерации задач и функция “start\_win” для открытия начального окна.

# Листинг программы

import random  
from itertools import product  
from itertools import permutations  
import tkinter as tk  
from tkinter import ttk  
import sys  
import os  
import platform  
  
value\_f1=0  
value\_otv=[]  
value\_user\_otv=[]  
def start\_win():  
 global value\_f1  
 root = tk.Tk()  
 root.title("Позорное окно")  
 if platform.system() == 'Windows':  
 root.geometry('200x95')  
 else:  
 root.geometry('260x125')  
  
 label = tk.Label(root, text="Выберите сложность:")  
 label.grid(row=0,column=0)  
  
 def on\_closing():  
 if root.winfo\_exists():  
 print("Окно еще открыто")  
  
 def set\_difficulty(x):  
 root.difficulty = x  
 root.destroy() # Закрыть окно после выбора сложности  
  
 def test():  
 root.flag = True  
 root.destroy()  
  
 def close\_program():  
 if platform.system() == 'Windows':  
 sys.exit()  
 else:  
 os.system('pkill -9 python')  
  
 easy\_button = tk.Button(root, text=" Легко ", command=lambda: set\_difficulty("Easy"))  
 easy\_button.grid(row=1,column=0)  
  
 medium\_button = tk.Button(root, text=" Средне ", command=lambda: set\_difficulty("Medium"))  
 medium\_button.grid(row=2,column=0)  
  
 hard\_button = tk.Button(root, text="Сложно", command=lambda: set\_difficulty("Hard"))  
 hard\_button.grid(row=3,column=0)  
  
  
 test\_button = tk.Button(root, text=" Тест ", command=lambda: test())  
 test\_button.grid(row=1,column=1)  
  
 exit = tk.Button(root, text="Выход", command=close\_program)  
 exit.grid(row=2,column=1)  
  
 root.protocol("WM\_DELETE\_WINDOW", on\_closing)  
  
 root.flag = False  
 root.difficulty = None  
 root.mainloop()  
  
  
 return root.flag, root.difficulty  
  
  
class Generate:  
 def show(self):  
 global value\_f1, value\_otv, count, value\_i  
 if value\_function[0]==True:  
 self.selected\_difficulty = random.choice(['Easy', 'Medium', 'Hard'])  
 Generate.generator(self)  
 if value\_f1==0:  
 value\_f1+=1  
 Generate.test(self)  
 else:  
 self.selected\_difficulty = value\_function[1]  
 Generate.generator(self)  
  
 self.frame = ttk.Frame(notebook)  
 self.text\_widget = tk.Text(self.frame, wrap="word", height=16, width=50)  
 self.text\_widget.insert(tk.END, self.main\_text)  
 self.text\_widget.config(state="disabled")  
 self.text\_widget.grid(row=1,column=0,columnspan=5, stick='nswe')  
 if value\_function[0]==False:  
 self.square = tk.Frame(self.frame, width=400, height=5, bg="white", highlightbackground="black",  
 highlightthickness=1)  
 self.square.grid(row=0, column=0,columnspan=2, stick='nswe')  
 if value\_function[0]==True and count!=9:  
 notebook.add(self.frame, text=f"З {value\_i}")  
 notebook.grid(row=0,rowspan=2,column=0,columnspan=5)  
 value\_otv.append(str(Generate.examinate(self)))  
 elif value\_function[0]==False:  
 notebook.add(self.frame, text=f"Задача {count+1}")  
 notebook.grid(row=0, rowspan=2, column=0, columnspan=5)  
  
 value\_otv.append(str(Generate.examinate(self)))  
  
  
 def get\_entry\_value(self):  
 global current\_timer  
 if len(entry.get())>10:  
 print('введите корректный ответ')  
 else:  
 if value\_function[0]:  
 value\_user\_otv.append(entry.get())  
 if len(value\_user\_otv)==10:  
 count=0  
 for i in range(0,10):  
 notebook.tab(i, state='normal')  
 for i in range(0,10):  
 notebook.forget(notebook.index("current"))  
 win1.after\_cancel(current\_timer)  
 entry.destroy()  
 button\_ex.destroy()  
 button\_exit.destroy()  
 timer.destroy()  
  
 win1.geometry("200x276")  
 for i in range(10):  
 if value\_otv[i]==value\_user\_otv[i]:  
 count+=1  
 pr=tk.Label(text=f'Вы набрали {count} верных ответов')  
 l = tk.Label(text=f'полученный ответ')  
 l.grid(row=0,column=0)  
 l = tk.Label(text=f'верный ответ')  
 l.grid(row=0, column=1)  
 for i in range(10):  
 l=tk.Label(text=f'{value\_user\_otv[i]}')  
 l.grid(row=i+1,column=0)  
 l=tk.Label(text=f'{value\_otv[i]}')  
 l.grid(row=i + 1, column=1)  
 pr.grid(row=i+2,column=0,columnspan=2)  
 button = tk.Button(win1, text="Начальное окно", command=restart\_program)  
 button.grid(row=i+3,column=0,columnspan=2)  
 else:  
 if current\_timer:  
 win1.after\_cancel(current\_timer)  
 countdown(180)  
 notebook.tab(notebook.index("current") + 1, state='normal')  
 notebook.tab(notebook.index("current"), state='disabled')  
 else:  
 if str(entry.get()) == value\_otv[notebook.index("current")]:  
 Generate.change\_color\_right(self)  
 print(entry.get(),value\_otv[notebook.index("current")],"Правильно")  
 else:  
 Generate.change\_color\_false(self)  
 print(entry.get(),value\_otv[notebook.index("current")],"Неправильно")  
 if len(value\_user\_otv)!=10:  
 entry.delete(0, "end")  
  
 def change\_color\_right(self):  
 self.square.config(bg='#00FF00')  
  
 def change\_color\_false(self):  
 self.square.config( bg='#FF0000')  
  
 def generator(self):  
 alph = [ord("А"),ord("У"),ord("О") ,ord("И") ,ord("Э") ,ord("Я"),ord("Ю"),ord("Е")]  
  
 count=[]  
 self.scale = random.randint(4, 6)  
 self.scale\_word=self.scale+random.randint(1,2)  
 self.word=[]  
 self.position=random.choice(['начинается','заканчивается'])  
 self.value=[]  
 self.value1=[]  
 self.value2=[]  
 self.count\_for3\_m=0  
 combinations\_example=''  
 if self.selected\_difficulty=='Easy':  
 self.num=random.randint(1,3)  
 else:  
 self.num=random.randint(1,2)  
 if self.num == 3 and self.selected\_difficulty == 'Medium' or self.num==2 and self.selected\_difficulty=='Medium'or self.num==2 and self.selected\_difficulty=='Hard':  
 self.value1 = random.choice(['гласныe', 'согласныe'])  
 self.value2 = random.choice(['гласные', 'согласные'])  
  
 for i in range(self.scale):  
 if self.num==3 and self.selected\_difficulty=='Medium' or self.num==2 and self.selected\_difficulty=='Medium'or self.num==3 and self.selected\_difficulty=='Hard':  
 if self.value1=='гласныe' and self.count\_for3\_m<self.scale//2 or self.value2=='гласныe' and self.count\_for3\_m<self.scale//2:  
 a=random.choice(alph)  
 self.count\_for3\_m+=1  
 elif self.value1=='согласныe' and self.count\_for3\_m<self.scale//2 or self.value2=='согласные' and self.count\_for3\_m<self.scale//2:  
 a=random.choice([i for i in range(1040,1072) if i not in alph])  
 self.count\_for3\_m += 1  
 # elif self.value1=='гласныe' and self.count\_for3\_m>=self.scale//2:  
 # a=random.choice([i for i in range(1040,1072) if i not in alph])  
 # elif self.value1=='согласныe' and self.count\_for3\_m>=self.scale//2:  
 # a=random.choice(alph)  
 else:  
 a = random.randint(1040, 1071)  
 else:  
 a = random.randint(1040, 1071)  
 if a in count:  
 while a in count:  
  
 a = random.randint(1040, 1071)  
 if a == 1066 or a == 1067 or a == 1068:  
 while a == 1066 or a == 1067 or a == 1068:  
 a = random.randint(1040, 1071)  
 self.word.append(chr(a))  
 self.word = sorted(self.word)  
 count.append(a)  
  
 if self.num in [4, 5, 6]:  
 self.scale\_word = len(self.word)  
 for i, x in enumerate(permutations(''.join(self.word))):  
 if i == self.scale:  
 break  
 combinations\_example += f'\n{i + 1}: {",".join(x)}'  
 else:  
 for i, x in enumerate(product(''.join(self.word), repeat=self.scale\_word)):  
 if i == self.scale:  
 break  
 combinations\_example += f'\n{i + 1}: {",".join(x)}'  
 self.word1 = self.word[:]  
 text = f'Все {self.scale\_word}-буквенные слова, составленные из букв {", ".join(self.word)}, записаны в алфавитном порядке.\n Вот начало списка: {combinations\_example}\n... \n'  
 text1=''  
 if self.selected\_difficulty=='Easy':  
 if self.num==1 :  
 vl=len(self.word)\*\*self.scale\_word  
 self.value=random.randint(10, vl)  
 text1+=f'Запишите слово, которое стоит на {self.value}-м месте от начала списка.'  
 elif self.num==2:  
 self.value = random.choice([i for i in self.word if i!=self.word[0] and i!=self.word[-1]])  
 if self.position=='начинается':  
 text1 += f'Укажите номер последнего слова, которое {self.position} с буквы {self.value}.'  
 else:  
 text1 += f'Укажите номер последнего слова, которое {self.position} на букву {self.value}.'  
 elif self.num==3:  
 self.value = random.choice(self.word)  
 text1 += f'Причём буква {self.value} появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов можно использовать?'  
  
  
 if self.selected\_difficulty=='Medium':  
 self.value = random.choice([i for i in self.word if i!=self.word[0] if i!=self.word[-1]])  
 self.word.pop(self.word.index(self.value))  
 if self.num==1 or self.num==4:  
 self.value1 = random.choice([i for i in self.word if i != self.word[0] if i != self.word[-1]])  
 if self.num==1:  
 text1 += f'Каждую букву можно использовать любое количество раз.Укажите номер первого слова, которое начинается с буквы {self.value} и заканчивается {self.value1}.'  
 elif self.num==2 or self.num==3:  
 text1 += f'Каждую букву можно использовать любое количество раз, при этом нельзя ставить рядом две {self.value1}. Сколько различных последовательностей можно составить?'  
 if self.selected\_difficulty=='Hard':  
 self.value = random.choice(self.word)  
 self.word.pop(self.word.index(self.value))  
 self.value1 = random.choice(self.word)  
 if self.num==1:  
 self.value2 = random.choice(self.word)  
 text1 += f'Буквы {self.value} и {self.value1} нужно обязательно использовать ровно по одному разу, букву {self.value2} можно использовать один раз или не использовать совсем, остальные буквы можно использовать произвольное количество раз или не использовать совсем. Сколько различных кодов можно составить?'  
 if self.num==2:  
 self.position = random.choice(['начинаться', 'заканчиваться'])  
 text1 += f'Каждую букву можно использовать любое количество раз, при этом последовательность не может {self.position} с буквы {self.value}, должна содержать хотя бы одну {self.value1}, а так же две подряд идущие {self.value2}. Сколько различных кодов можно составить?'  
  
 self.main\_text=text+text1  
  
 def examinate(self):  
 alph=['А','У','О','И','Э','Ы','Я','Ю','Е','Ё ']  
  
 self.count=0  
 if self.num in [1,2,3]:  
 for i,x in enumerate(product(self.word1,repeat=self.scale\_word)):  
 if self.selected\_difficulty=='Easy':  
 if self.num==1:  
 if i+1==self.value:  
 return "".join(x)  
 elif self.num==2:  
 if self.position=='начинается':  
 if x[0]==self.value:  
 self.count=i+1  
 if self.position=='заканчивается':  
 if x[-1]==self.value:  
 self.count=i+1  
 elif self.num==3:  
 if x.count(str(self.value))==1:  
 self.count+=1  
 if self.selected\_difficulty=='Medium':  
 if self.num==1:  
 if x[0]==self.value and x[-1]==self.value1:  
 return i+1  
  
 if self.num==2 or self.num==3:  
 local\_count=0  
 for j in range(self.scale\_word-1):  
 if self.value1=="согласныe":  
 if x[j] not in alph and x[j + 1] not in alph:  
 break  
 else:  
 local\_count += 1  
 else:  
 if x[j] in alph and x[j + 1] in alph:  
 break  
 else:  
 local\_count += 1  
 if local\_count == len(x) - 1:  
 self.count += 1  
  
 if self.selected\_difficulty == 'Hard':  
 if self.num == 1:  
 if x.count(self.value) == 1 and x.count(self.value1) == 1 and (x.count(self.value2) == 1 or x.count(self.value2) == 0):  
 self.count+=1  
 elif self.num==2:  
 if self.position=="начинаться":  
 if x[0] != self.value and x.count(self.value1) >= 1:  
 if self.value2=="согласныe":  
 for j in range(len(x) - 1):  
 if x[j] not in alph and x[j + 1] not in alph:  
 self.count += 1  
 else:  
 for j in range(len(x) - 1):  
 if x[j] in alph and x[j + 1] in alph:  
 self.count += 1  
 elif self.position=="заканчиваться":  
 if x[-1] != self.value and x.count(self.value1) >= 1:  
 if self.value2 == "гласные":  
 for j in range(len(x) - 1):  
 if x[j] in alph and x[j + 1] in alph:  
 self.count += 1  
 else:  
 for j in range(len(x) - 1):  
 if x[j] not in alph and x[j + 1] not in alph:  
 self.count += 1  
  
 return self.count  
  
  
  
  
 def exit(self):  
 win1.destroy()  
  
 def next(self):  
 global count  
 count += 1  
 self.new\_tab = Generate()  
 tabs[count] = self.new\_tab  
 self.new\_tab.show()  
  
 def test(self):  
 global count,value\_i  
 for i in range(0,10):  
 self.new\_tab = Generate()  
 value\_i = i+1  
 tabs[i] = self.new\_tab  
 self.new\_tab.show()  
 if i > 0:  
 notebook.tab(i, state='disabled') # Заблокировать вкладку  
 count=i  
  
 def DelTab(self):  
 global count  
 value=notebook.index("current")  
 if count > 0:  
 count -= 1  
 notebook.forget(notebook.index("current"))  
 for i in range(value,len(tabs)-1):  
 tabs[i]=tabs[i+1]  
 else:  
 print('осталась одна вкладка')  
  
  
def countdown(count):  
 global current\_timer  
 if current\_timer:  
 win1.after\_cancel(current\_timer) # Отменяем предыдущий таймер, если он был установлен  
 timer.config(text=str(count))  
 if count > 0:  
 current\_timer = win1.after(1000, countdown, count - 1) # Устанавливаем новый таймер  
 else:  
 Generate.get\_entry\_value(count)  
  
def toggle\_visibility(event):  
 value=notebook.index('current')  
 otv.config(text=f"{value\_otv[value]}")  
  
def restart\_program():  
 python = sys.executable  
 os.execl(python, python, \*sys.argv)  
  
count=0  
value\_i=0  
current\_timer=None  
value\_function=start\_win()  
win1 = tk.Tk()  
win1.title('задача')  
tabs=dict()  
notebook = ttk.Notebook(win1)  
print("Если кнопки 'начальное окно' и 'выход' не работают - виновата школьная ОС, а не разработчик, запускал на эмуляторе убунту, всё работает")  
if value\_function[0]==False:  
 standart=Generate()  
 standart.show()  
 tabs[0]=standart  
 button\_back = tk.Button(win1, text="Следующая задача", command=lambda: tabs[notebook.index("current")].next())  
 button\_back.grid(row=2, column=2, stick='e')  
 button\_back = tk.Button(win1, text="Удалить вкладку", command=lambda: tabs[notebook.index("current")].DelTab())  
 button\_back.grid(row=2, column=4, stick='e')  
 otv = tk.Label(win1, text="смотреть ответ")  
 otv.grid(row=3, column=0)  
 otv.bind("<Button-1>", toggle\_visibility)  
 button = tk.Button(win1, text="Начальное окно", command=restart\_program)  
 button.grid(row=3, column=2, stick='nsew')  
 button\_exit = tk.Button(win1, text="Выйти", command=lambda: tabs[notebook.index("current")].exit())  
 button\_exit.grid(row=3, column=4, stick='nsew')  
else:  
 standart = Generate()  
 standart.show()  
 timer = tk.Label(win1, font=('Arial', 10))  
 timer.grid(row=2, column=2, stick='e')  
 countdown(180)  
 button\_exit = tk.Button(win1, text="Выйти", command=lambda: tabs[notebook.index("current")].exit())  
 button\_exit.grid(row=2, column=4, stick='nsew')  
  
entry = tk.Entry(win1)  
entry.grid( row=2,column=0,stick='e')  
button\_ex = tk.Button(win1, text="Ответить", command=lambda: tabs[notebook.index("current")].get\_entry\_value())  
button\_ex.grid(row=2,column=1,stick='e')  
  
  
  
  
win1.mainloop()

# Тестовые примеры

При выборе сложности “легко” генерируется задача типа:

Все 7-буквенные слова, составленные из букв Н, П, Т, Х, Ш, Я, записаны в алфавитном порядке.

Вот начало списка:

1: Н,Н,Н,Н,Н,Н,Н

2: Н,Н,Н,Н,Н,Н,П

3: Н,Н,Н,Н,Н,Н,Т

4: Н,Н,Н,Н,Н,Н,Х

5: Н,Н,Н,Н,Н,Н,Ш

6: Н,Н,Н,Н,Н,Н,Я

...

Запишите слово, которое стоит на 104545-м месте от начала списка.

При выборе сложности “средне” генерируется задача типа:

Все 8-буквенные слова, составленные из букв Д, М, Н, П, Т, Ф, записаны в алфавитном порядке.

Вот начало списка:

1: Д,Д,Д,Д,Д,Д,Д,Д

2: Д,Д,Д,Д,Д,Д,Д,М

3: Д,Д,Д,Д,Д,Д,Д,Н

4: Д,Д,Д,Д,Д,Д,Д,П

5: Д,Д,Д,Д,Д,Д,Д,Т

6: Д,Д,Д,Д,Д,Д,Д,Ф

...

Каждую букву можно использовать любое количество раз.Укажите номер первого слова, которое начинается с буквы М и заканчивается Н.

При выборе сложности “сложно” генерируется задача типа:

Все 7-буквенные слова, составленные из букв Д, П, С, Ш, Э, записаны в алфавитном порядке.

Вот начало списка:

1: Д,Д,Д,Д,Д,Д,Д

2: Д,Д,Д,Д,Д,Д,П

3: Д,Д,Д,Д,Д,Д,С

4: Д,Д,Д,Д,Д,Д,Ш

5: Д,Д,Д,Д,Д,Д,Э

...

Буквы Ш и Э нужно обязательно использовать ровно по одному разу, букву Д можно использовать один раз или не использовать совсем, остальные буквы можно использовать произвольное количество раз или не использовать совсем. Сколько различных кодов можно составить?

# Описание размещения

Для запуска программы «Генератор заданий номер 8 ЕГЭ по информатике на Python» требуется файл с программой и все используемые модули встроены в Python.

# Требования к программным и аппаратным средствам

Язык программирования: Python

Версия программы для программирования: IDLE (Python 3.9 32-bit)

Версия офисной программы, с помощью которой была подготовлена документация: Microsoft Word

Операционная система: Windows 11 Pro (дата установки – 14.08.2023)

Характеристика компьютера, на котором точно работает программа, программа может работать и на слабых устройствах:

1. Intel i5 12400f
2. RAM 16gb 3200mhz
3. AMD Radeon RX6700