**Проект**

**‘Генератор 14-го задания ЕГЭ по Информатике’**

Руководство оператора

Разработчик: Дейнеко Владимир

Руководитель: Игорь Александрович

Санкт-Петербург, 2024

*Аннотация*

В данном документе содержится описание алгоритмов, по которым работает программа ‘Генератор 14-го задания ЕГЭ по Информатике’, взаимодействует с пользователем, интерфейсом, а также требования к программным и аппаратным средствам.

*Содержание*

[*Постановка задачи* 4](#_Toc158662206)

[*Формализация алгоритма* 5](#_Toc158662207)

[*Листинг файла «main.py»* 7](#_Toc158662208)

[*Описание размещения программы* 15](#_Toc158662209)

[*Требования по программному обеспечению* 16](#_Toc158662210)

# *Постановка задачи*

Данная программа направлена на тренировку 14-го задания ЕГЭ по информатике.

Она позволяет тренировать несколько вариаций 14-го задания ЕГЭ по Информатике следующих типов:

* *Тип 1:*

Значение выражения 329 - 395 записали в системе счисления с основанием 3. Сколько цифр '0' содержится в этой записи?

* *Тип 2:*

Укажите через запятую в порядке возрастания все десятичные натуральные числа, не превосходящие 36, запись которых в семиричной системе счисления оканчивается на две одинаковые цифры.

* *Тип 3:*

Чему равно наименьшее основание позиционной системы счисления x, при котором 445x = 100100101y? Ответ записать в виде целого числа.

* *Тип 4:*

Запись числа 359 в системе счисления с основанием N содержит 3 цифры и оканчивается на 7.Чему равно минимально возможное основание системы счисления?

Каждое задание генерируется рандомно по определённому алгоритму, который представлен далее.

# *Формализация алгоритма*

При запуске программы создаются объекты графического интерфейса и показывается первый из них – приветственное окно. Переключение между окнами происходит нажатием на соответствующие кнопки графического интерфейса, при этом новое окно показывается поверх предыдущего или взамен старого, в зависимости от окна.

После создания интерфейса запускается алгоритм генерации задания, состоящий из следующих этапов:

* Генерация типа задачи из выбранных пользователем и запуск соответствующей функции генерации задачи
* Генерация необходимых констант для условия задачи
* Вычисление ответа на задачу
* Вывод условия задачи в специальное поле главного окна графического интерфейса и запоминание программой ответа

Далее у пользователя есть несколько вариантов действий:

1. Открыть настройки генерации
2. Ввести ответ на задачу и нажать на кнопку проверки
3. Нажать на кнопку генерации следующего задания
4. Посмотреть статистику своих ответов
5. Выйти из программы

В первом случае можно изменить настройки генерации и сохранить их (новые настройки будут сохранены в таблицу для настроек базы данных) либо не сохранять. В этом случае настройки из базы данных будут перенесены в интерфейс.

Во втором случае будет запущен алгоритм, сравнивающий ответ пользователя с ответом, запомненным программой. При совпадении будет выведено «Верно» в строку для вывода вердикта, иначе будет выведено «Неверно».

В третьем случае запустится алгоритм проверки значения в строке для вывода вердикта. Если значение «Верно», то в таблицу статистики ответов базы данных в столбец для вердикта будет занесено значение True (решено), иначе False (не решено). Затем запустится алгоритм генерации, описанный ранее.

В четвёртом случае откроется окно статистики ответов пользователя, изначально только с кнопкой «обновить статистику». При нажатии на неё из таблицы статистики базы данных возьмутся все решённые задачи, все не решённые, и их количество будет выведено в окне статистики в соответствующем поле решённых верно(неверно) задач.

В пятом случае окно интерфейса закроется и работа программы завершится.

# *Листинг файла «main.py»*

import sys

from PyQt5 import uic, QtCore

from PyQt5.QtGui import QPixmap

from PyQt5.QtWidgets import QApplication, QMainWindow

import DataBaseFile

import GenerationQuestionFile

import InformationWindow

import StatisticsWindows

import MoveFunction

# Класс главного окна

class MainQuestionWindow(QMainWindow):

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

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

uic.loadUi('uis\MainQuestionWindow.ui', self)

self.setWindowTitle('Системы счисления')

# Передвижение на центр экрана

MoveFunction.move\_to\_senter(self)

self.initUi()

# Получение настроек

types = DataBaseFile.find('''SELECT \* FROM QuestionSettings''')

b = []

if types[0] == 'True':

b.append(1)

if types[1] == 'True':

b.append(2)

if types[2] == 'True':

b.append(3)

if types[3] == 'True':

b.append(4)

self.types = b

def initUi(self):

self.settingsButton.clicked.connect(self.settings)

self.exitButton.clicked.connect(self.close\_window)

self.statisticsButton.clicked.connect(self.statistics)

self.nextButton.clicked.connect(self.create\_question)

self.answerButton.clicked.connect(self.answer)

self.questionEdit.setReadOnly(True)

def close\_window(self):

self.close()

def settings(self):

sw.show()

def statistics(self):

statw.show()

def answer(self):

# Проверка ответа

if self.answerEdit.text() == str(self.answer):

self.answerLabel.setText('Верно')

# Запись правильно решённой задачи

DataBaseFile.insert('True', 'QuestionAnswers')

else:

self.answerLabel.setText('Неверно')

def create\_question(self, flag=False):

if not flag:

# Запись неправильно решённой задачи

if self.answerLabel.text() != 'Верно':

DataBaseFile.insert('False', 'QuestionAnswers')

self.nextButton.setEnabled(False)

# Генерация задачи

n = GenerationQuestionFile.generation\_function(self.types)

self.answer = n[1]

self.questionEdit.setText(n[0])

self.answerLabel.setText('')

self.answerEdit.setText('')

self.nextButton.setEnabled(True)

# Приветственное окно

class HelloWindow(QMainWindow):

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

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

uic.loadUi('uis\HelloWindow.ui', self)

self.setWindowTitle('Приветствие')

self.startWorkButton.clicked.connect(self.start)

self.pixmap = QPixmap('files\HelloImage.jpg')

self.image.setPixmap(self.pixmap)

# Передвижение на центр экрана

MoveFunction.move\_to\_senter(self)

def start(self):

self.close()

mqw.show()

# Окно настроек задач на системы счисления

class SettingsQuestionWindow(QMainWindow):

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

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

uic.loadUi('uis\SettingsWindow.ui', self)

self.setWindowTitle('Настройки')

# Показывание окна поверх главного

self.setWindowFlag(QtCore.Qt.WindowStaysOnTopHint)

# Передвижение на центр экрана

MoveFunction.move\_to\_senter(self)

self.initUi()

def initUi(self):

self.cancelButton.clicked.connect(self.close\_window)

self.acceptButton.clicked.connect(self.save)

self.informationButton.clicked.connect(self.show\_information)

self.typesBox.clicked.connect(self.choice\_type)

self.clearStatisticsButton.clicked.connect(DataBaseFile.delete\_qa)

# Получение настроек при запуске

a = DataBaseFile.find('''SELECT \* FROM QuestionSettings''')

self.type1Box.setEnabled(False)

self.type2Box.setEnabled(False)

self.type3Box.setEnabled(False)

self.type4Box.setEnabled(False)

if a[0] == 'True':

self.type1Box.setChecked(True)

else:

self.type1Box.setChecked(False)

if a[1] == 'True':

self.type2Box.setChecked(True)

else:

self.type2Box.setChecked(False)

if a[2] == 'True':

self.type3Box.setChecked(True)

else:

self.type3Box.setChecked(False)

if a[3] == 'True':

self.type4Box.setChecked(True)

else:

self.type4Box.setChecked(False)

def close\_window(self):

# Закрытие и возвращение изменений

self.errorLabel.setText('')

a = DataBaseFile.find('''SELECT \* FROM QuestionSettings''')

if a[0] == 'True':

self.type1Box.setChecked(True)

else:

self.type1Box.setChecked(False)

if a[1] == 'True':

self.type2Box.setChecked(True)

else:

self.type2Box.setChecked(False)

if a[2] == 'True':

self.type3Box.setChecked(True)

else:

self.type3Box.setChecked(False)

if a[3] == 'True':

self.type4Box.setChecked(True)

else:

self.type4Box.setChecked(False)

self.close()

def show\_information(self):

iw.show()

def save(self):

# Проверка на наличие хотя бы 1 типа в генерации

if not (self.type1Box.isChecked() or self.type2Box.isChecked() or self.type3Box.isChecked() or

self.type4Box.isChecked()):

self.errorLabel.setText('Вы не выбрали тип задач.')

else:

self.errorLabel.setText('')

# Закрытие и сохранение изменений

types = [self.type1Box.isChecked(), self.type2Box.isChecked(),

self.type3Box.isChecked(), self.type4Box.isChecked()]

DataBaseFile.update(types)

b = []

if types[0]:

b.append(1)

if types[1]:

b.append(2)

if types[2]:

b.append(3)

if types[3]:

b.append(4)

mqw.types = b

self.close()

def choice\_type(self):

# Вкл-выкл выбора типов задач(пользователем)

if self.typesBox.isChecked():

self.type1Box.setEnabled(True)

self.type2Box.setEnabled(True)

self.type3Box.setEnabled(True)

self.type4Box.setEnabled(True)

else:

self.type1Box.setEnabled(False)

self.type2Box.setEnabled(False)

self.type3Box.setEnabled(False)

self.type4Box.setEnabled(False)

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

app = QApplication(sys.argv)

mqw = MainQuestionWindow()

mqw.create\_question(flag=True)

sw = SettingsQuestionWindow()

statw = StatisticsWindows.StatisticsQuestionWindow()

iw = InformationWindow.InformationWindow()

hw = HelloWindow()

hw.show()

sys.exit(app.exec\_())

# *Описание размещения программы*

Программа запускается с помощью любого редактора кода Python.

Файл для запуска: ‘Папка установки \main.py’

Файлы с кодом находятся в корневой папке, которой является папка установки, файлы интерфейса находятся во вложенной папке «uis», прочие файлы в папке «files».

# *Требования по программному обеспечению*

Обязательные требования:

* Наличие Python 3.9
* Модуль PyQt5
* Модуль SQlite3

Программа тестировалась на компьютере с характеристиками:

* Windows 11
* Процессор Intel(R) Core(TM) i3-8100 CPU @ 3.60GHz 3.60 GHz
* Оперативная память 16.00 ГБ