19. Седов В.А. Обоснование применения машин и механизмов для строительства сооружений / В.А. Седов, В.П. Перцев, С.М. Кузнецов // Транспортное строительство. –2004. –№ 2. –С. 12 – 14.
20. Сироткин Н.А. Методика обоснования очередности строительства объектов / Н.А. Сироткин, С.М. Кузнецов, С.Н. Ячменьков // Экономика ж. д. –2006. –№ 10. –С. 75 – 78.
21. Сироткин Н.А. Оценка обоснованности очередности строительства объектов методом имитационного моделирования / Н.А. Сироткин, С.М. Кузнецов // Изв. вузов. Строительство. – 2007. –№ 1. –С. 81 – 86.
22. Экономико-математическая модель работы стреловых кранов / О.В. Демиденко, В.Н. Анферов, С.М. Кузнецов, М.Ю. Серов, С.И. Васильев // Омский научный вестник. –2013. –№ 3 (119). –С.74–80.

Казунин Р.В., Нестеров А.О.

Разработка программы для генерации случайных имён

*УрФУ им. Б.Н. Ельцина, Нижнетагильский технологический институт
(Россия, Нижний Тагил)*

doi: 10.18411/lj-02-2020-22

idsp: ljjournal-02-2020-22

Аннотация

В статье описан процесс разработки приложения, позволяющего генерировать случайные имена с учётом требований пользователя. Программа написана на языке Python. Создание графического интерфейса производилась в среде сборки QtCreator.

Ключевые слова: генерация, программа, случайное имя, алгоритм, программирование, Python.

Abstract

This article describes the process of developing an application that allows you to generate random names based on user requirements. The program is written in Python. The GUI was created in the QtCreator build environment.

Keywords: generation, program, random name, algorithm, programming, Python.

Современный рынок информационно-коммуникационных технологий сложно удивить какими-либо новыми программными продуктами [1]. Однако, в условиях стремительного развития различных гаджетов, перед пользователями возникает задача выбрать псевдоним в игре или социальных сетях. Изучение потребностей рынка позволяет сделать вывод, что большая часть подобных приложений, например «Генератор имен и фамилий онлайн (<https://generator-online.com/names/>) имеют веб-платформы, что делает недопустимым эксплуатацию продукта без доступа к сети Интернет. Таким образом, целью нашего проекта является создание приложения, которое бы могло работать без доступа к сети Интернет.

Для удобства разработки и представления программы были разработаны три взаимосвязанные подсистемы.

1. Алгоритм чтения данных из текстового файла.
2. Алгоритм генерации случайного имени.
3. Графическая оболочка.

Разработка программы, в частности написание кода на Python и создание графического интерфейса, производилась в среде сборки QtCreator.

Программа должна выполнять следующие функции: считывать имена из приложенных текстовых файлов, запрашивать у пользователя параметры имени, генерировать случайное имя, выводить его на форму.

В обязанности графического входят следующие функции: представление активных элементов для удобного визуального управления программой, получение вводимых данных и передача их ядру, вывод сообщений в случае возникновения ошибок исходных данных или обработки, демонстрация результата генерации [2].

Основной задачей системы является генерация случайного имени. Система состоит из двух частей: ядро и графический интерфейс.

Ядро – это основная часть системы, в задачи которого входит выполнение системно зависимых функций и независимость от структуры и реализации графического интерфейса. Ядро собирает информацию с графического интерфейса и, учитывая её, генерирует случайное имя [2, 3].

Основные компоненты ядра системы: модуль чтения данных из файла и модуль генерации случайного имени.

Программа оснащена графическим интерфейсом, обеспечивающим удобную и комфортную работу пользователя с ней.

Алгоритм чтения данных из файла был реализован на языке Python в виде функции, благодаря которой в качестве аргумента передаётся путь к файлу, который требуется прочитать в оперативную память.

Листинг 1

Код чтения данных из файла в оперативную память

```
def read_from_file(path):  
    try:  
        f = open(path, encoding = "utf8")  
    except IOError as e:  
        return "  
    return f.read().replace(u'\uffeff', ")
```

В теле функции производится вызов стандартной функции `open` открытия файла, принимающей путь и кодировку. Функция возвращает объект. На случай, если `open` не найдёт указанный файл, вызов функции `open` осуществляется в блоке `try-except`, который отлавливает исключения. Если оно не возникло, производим чтение данных из файла и их возврат из функции.

Основным алгоритмом в данном проекте является механизм генерации случайных имён, приведённый в листинге 2.

Листинг 2

Код функции чтения данных из файла в оперативную память

```
def random_name_generator(list_of_names, len_name_min, len_name_max, substr):  
    substr_down = substr.lower()  
    list_of_need_len = []  
    for name in list_of_names:  
        if len(name) >= len_name_min and len(name) <= len_name_max and  
substr_down in name.lower():  
            list_of_need_len.append(name)  
    return random.choice(list_of_need_len)
```

Функция `random_name_generator` принимает в качестве аргументов список имён `list_of_names`, минимальную `len_name_min` и максимальную `len_name_max` длину имени и обязательную последовательность `substr`. В теле функции выполняется перевод символов обязательной последовательности в нижний регистр путём вызова метода `lower()`. Затем объявляется пустой список `list_of_need_len` для дальнейшего наполнения подходящими словами. Цикл по коллекции `list_of_names` производит перебор имён, уже извлечённых из файла, на каждой итерации помещая имя в переменную `name`. В теле цикла проверяется удовлетворение текущего имени `name` условиям – параметрам, собранным приложением из элементов формы. Если все условия выполняются и имя подходит, то оно добавляется в список `list_of_need_len` путём вызова метода `append()`. Функция `random_name_generator` возвращает случайно выбранное имя из списка `list_of_need_len`, которое в последствии и отображается как результат генерации на форме.

Для удобства пользователя работы с программой, она оснащена графическим интерфейсом, скрин которого приведён ниже (рис. 1).

Рис. 1. Графический интерфейс программы

Пользователь указывает минимальное и максимальное количество букв посредством соответствующего смещения ползунков. Затем производится выбор пола имени: мужское или женское. По желанию можно задать обязательную последовательность. Это набор символов, который должен обязательно встречаться в имени, которое будет сгенерировано. Например, пользователь в поле обязательная последовательность указал «аш», то имена Саша, Маша, Даша, Паша могут быть сгенерированы, так как содержат сочетание букв «аш». Но имена Коля, Дима, Катя не могут являться результатом генерации в данном случае.

Данное приложение просто в использовании и интуитивно понятно, что позволяет пользователю без особых трудностей им пользоваться. Язык программирования Python и его графический фреймворк PyQt дают огромные возможности для создания оконных приложений на различные операционные системы [4, 5].

В дальнейшем планируется развитие приложения в сторону расширения функциональности: добавление новых видов имён, названий, генерация более сложных конструкций.

1. Бужинская Н.В., Драчев В.В. Проектирование и разработка автоматизированной системы online-мониторинга учебного процесса. Проектирование и технология электронных средств. 2018. №4. С. 15-18.
2. Автостопом по Python/ К. Рейтц, Т. Шлюссер ; пер. Е. Зазноба. СПб.: Питер, 2017. 336 с.
3. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. М.: Издательство Юрайт, 2019. 432 с.
4. PyQt5 Reference Guide. [Электронный ресурс]. URL: <https://www.riverbankcomputing.com/static/Docs/PyQt5> (дата обращения: 12.10.2019).
5. Python 3. [Электронный ресурс]. URL: <https://docs.python.org/3/> (дата обращения: 10.10.2019).