МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ РАЗРАБОТКИ ДИЗАЙНА ВЕБ-САЙТОВ НА ОСНОВЕ CSS**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Акулов

(подпись)

Направление подготовки 02.03.03 — «Математическое обеспечение\_\_\_\_\_\_\_\_

(код, наименование)

и администрирование информационных систем»\_\_\_\_\_\_\_\_\_\_ курс\_\_\_\_\_3\_\_\_\_\_

Направленность (профиль)\_\_\_\_ Технология программирования\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель

канд. техн. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Е. Полупанова

(подпись, дата)

Нормоконтролер

преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Нигодин

(подпись, дата)

Краснодар

2024

**РЕФЕРАТ**

Курсовая работа 34 с., 5 ч., 14 рис., 1 табл., 9 источн., 3 прил.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, PYTHON, CSS, HTML, ВЕБ-ДИЗАЙН, АВТОМАТИЗАЦИЯ

Объектом исследования в данной работе является использование генетического алгоритма для автоматизации дизайна веб-сайтов.

Цель работы: разработка программы на Python, использующей генетический алгоритм для автоматической генерации и оптимизации стилей CSS.

Методологическая основа исследования включает в себя эксперименты, сбор данных, анализ результатов, построение графиков и сравнение полученных данных.

В результате работы была создана программа на языке Python, которая генерирует исходный CSS-код для заданной HTML-структуры, оптимизирует стили с помощью генетического алгоритма и оценивает визуальные качества дизайна на основе заранее заданных метрик.

Новизна работы заключается в том, что для веб-дизайна применён метод искусственного интеллекта, который позволяет автоматически подбирать настройки CSS для получения стильного и удобного дизайна.

В результате решения поставленных задач было определено, что использование генетического алгоритма позволяет значительно улучшить визуальные характеристики дизайна при минимальном участии человека. Разработанная программа демонстрирует высокую эффективность в адаптации параметров CSS для достижения сбалансированного и эстетически привлекательного оформления веб-страниц.

**СОДЕРЖАНИЕ**

[1 Общие сведения о генетическом алгоритме и веб-дизайне 7](#_Toc184052512)

[1.1 Понятие генетического алгоритма 7](#_Toc184052513)

[1.2 Применение генетического алгоритма для оптимизационных задач 7](#_Toc184052514)

[1.3 Основные элементы веб-дизайна 8](#_Toc184052515)

[1.4 Проблемы автоматизации дизайна веб-сайтов 9](#_Toc184052516)

[2 Математическая постановка задачи оптимизации CSS 11](#_Toc184052517)

[2.1 Параметры CSS как пространство поиска 11](#_Toc184052518)

[2.2 Описание функции приспособленности 12](#_Toc184052519)

[3 Применение генетического алгоритма для разработки дизайна веб-сайтов  
 на основе CSS 14](#_Toc184052520)

[3.1 Методы генерации начальной популяции 14](#_Toc184052521)

[3.2 Генетические операторы 14](#_Toc184052522)

[3.2.1 Селекция 14](#_Toc184052523)

[3.2.2 Скрещивание 15](#_Toc184052524)

[3.2.3 Мутация 16](#_Toc184052525)

[3.3 Критерии остановки алгоритма 17](#_Toc184052526)

[3.4 Схема работы генетического алгоритма 17](#_Toc184052527)

[4 Реализация алгоритма на Python 22](#_Toc184052528)

[4.1 Основные сведения о программе 22](#_Toc184052529)

[4.2 Работа с HTML и CSS через Python 22](#_Toc184052530)

[4.3 Пример кода для оценки дизайна 24](#_Toc184052531)

[4.4 Генерация новых стилей CSS 24](#_Toc184052532)

[5 Анализ результатов оптимизации 26](#_Toc184052533)

[5.1 Эволюция CSS параметров по поколениям 26](#_Toc184052534)

[5.2 Сравнение исходного и оптимизированного дизайна 27](#_Toc184052535)

[5.3 Оценка эффективности алгоритма 29](#_Toc184052536)

[Заключение 31](#_Toc184052537)

[Список использованных источников 32](#_Toc184052538)

[Приложение А Основная программа 34](#_Toc184052539)

[Приложение Б CSS стили HTML страницы до работы алгоритма 38](#_Toc184052540)

[Приложение В CSS стили HTML страницы после работы алгоритма 40](#_Toc184052541)

#### **ВВЕДЕНИЕ**

Актуальность данной работы состоит в необходимости разработки новых методов автоматизации проектирования веб-дизайна, которые способны оптимизировать визуальные параметры сайтов с учетом эстетических и функциональных требований. Создание привлекательного и удобного веб-дизайна часто является трудоемким процессом, требующим значительных затрат времени и профессиональных навыков. Применение современных эвристических методов, таких как генетические алгоритмы (ГА), позволяет значительно ускорить этот процесс, обеспечивая поиск оптимальных решений в задачах, где невозможно применить точные методы за приемлемое время.

На сегодняшний день задача автоматизации проектирования веб-дизайна является актуальной в связи с возрастающими требованиями к эстетике и функциональности сайтов, а также необходимостью адаптации дизайна под разные устройства и платформы. Проблема заключается в сложности выбора параметров CSS, таких как цветовые схемы, размеры элементов и структура макета, которые обеспечивают максимальную привлекательность и удобство использования. Генетический алгоритм, основанный на принципах эволюционного отбора, предоставляет возможность эффективно решать эту задачу, адаптируя стили CSS к заданным критериям.

Основная цель работы – разработка системы оптимизации параметров CSS с использованием генетического алгоритма для автоматизированного проектирования веб-дизайна. Для реализации поставленной цели предполагается решить следующие задачи:

* изучить теоретические основы генетического алгоритма и принципы его применения для задач оптимизации;
* определить ключевые параметры CSS, подлежащие оптимизации, и разработать функцию приспособленности, учитывающую эстетические и функциональные критерии;
* реализовать генетический алгоритм на языке программирования Python для автоматической генерации и оценки стилей CSS;
* провести тестирование и анализ эффективности предложенного подхода, включая сравнение исходного и оптимизированного дизайна.

Объектом исследования в данной работе является применение генетического алгоритма для автоматизации проектирования веб-дизайна. Предметом исследования являются параметры CSS, подлежащие оптимизации, и их влияние на визуальное восприятие и удобство использования веб-страниц.

Информационной базой исследования являются результаты современных исследований по теории генетических алгоритмов и проектирования веб-сайтов. Методологическая основа исследования включает:

* графический метод (визуализация изменений параметров CSS и итогового дизайна);
* анализ полученных данных (сравнение исходного и оптимизированного дизайна).

Научная новизна работы заключается в применении генетического алгоритма для автоматизированного проектирования CSS-дизайна, что обеспечивает сочетание оптимизации визуальных и функциональных характеристик.

Теоретическая и практическая значимость работы состоит в применении разработанного подхода для автоматизации дизайна веб-сайтов в различных областях, таких как коммерческие и информационные порталы, игровые сайты, обучающие платформы. Подход может быть полезен при создании адаптивного дизайна для сайтов, работающих на устройствах с различными характеристиками экрана.

# Общие сведения о генетическом алгоритме и веб-дизайне

## Понятие генетического алгоритма

Генетический алгоритм (ГА) — это один из методов эвристического поиска, основанный на моделировании эволюционных процессов, происходящих в природе. ГА относится к классу эволюционных алгоритмов, которые применяются для решения сложных задач оптимизации, в том числе таких, где традиционные методы неэффективны из-за большого объема данных или нелинейной зависимости параметров.

Основная идея ГА заключается в применении принципов естественного отбора: лучшие решения остаются и размножаются, порождая новые комбинации, которые приближаются к оптимальному решению.

Для понимания работы генетического алгоритма важно выделить его ключевые элементы.

Хромосома — представление возможного решения задачи. В контексте веб-дизайна хромосома может быть набором параметров CSS, таких как цвета, размеры шрифтов, интервалы и другие визуальные свойства.

Популяция − совокупность хромосом, которые рассматриваются на каждом этапе (поколении) работы алгоритма.

Функция приспособленности (fitness function) − критерий, по которому оценивается качество хромосомы. Например, в задачах веб-дизайна эта функция может учитывать контрастность, читаемость текста и эстетическую привлекательность.

## Применение генетического алгоритма для оптимизационных задач

Генетический алгоритм используется для решения сложных оптимизационных задач, в которых традиционные методы, такие как линейное программирование или градиентные методы, оказываются недостаточно эффективными. Основное преимущество ГА заключается в его способности находить приближенные решения для задач с высокой размерностью, нелинейными зависимостями и множеством локальных экстремумов.

Применение ГА охватывает различные области:

* инженерные задачи − оптимизация конструкции механизмов, планирование маршрутов и управление производственными процессами;
* экономика и финансы − оптимизация инвестиционного портфеля, прогнозирование рыночных трендов;
* информатика − обучение нейронных сетей, задача коммивояжера, балансировка нагрузки в распределенных системах;
* креативные задачи − генерация музыкальных композиций, текстов и графических элементов.

Особенностью применения ГА является возможность моделировать эвристический подход. Метод не требует точных математических моделей задачи, что делает его универсальным.

Применение ГА для задач веб-дизайна связано с его способностью подбирать такие параметры, как цветовые схемы, размеры шрифтов, интервалы и положение элементов, оптимизируя эстетические и функциональные характеристики страницы. Веб-дизайн предъявляет многокритериальные требования, поэтому ГА может использоваться для нахождения баланса между визуальной привлекательностью и удобством использования.

## 1.3 Основные элементы веб-дизайна

Веб-дизайн включает в себя создание и оформление веб-страниц, направленных на обеспечение удобного взаимодействия пользователя с информацией. Основные элементы веб-дизайна можно классифицировать следующим образом:

* визуальные элементы:

1. цветовая палитра − определяет общую атмосферу сайта и влияет на восприятие страницы;
2. типографика − выбор шрифтов, их размеров, начертания и интервалов между строками;
3. графика и изображения включают логотипы, баннеры и иллюстрации, которые усиливают визуальное воздействие;

* структурные элементы:

1. макет (layout) − организация контента на странице, включая блоки, сетки и размещение элементов;
2. иерархия информации − распределение акцентов с помощью заголовков, подзаголовков и списков;

* интерактивные элементы:

1. кнопки и формы − элементы взаимодействия, обеспечивающие удобство навигации;
2. анимация и переходы − привлекают внимание пользователя и создают современный вид.

Эти элементы задаются с помощью CSS, который позволяет определять стили для HTML-структур. Параметры CSS могут быть оптимизированы для достижения максимальной эстетики и функциональности сайта.

## Проблемы автоматизации дизайна веб-сайтов

Автоматизация дизайна веб-сайтов — это процесс, направленный на создание интерфейсов и стилей без участия человека. Несмотря на очевидные преимущества, связанные с экономией времени и ресурсов, существуют значительные проблемы, связанные с этим процессом:

* многокритериальная оптимизация − дизайн веб-сайта должен одновременно учитывать эстетику, удобство использования, скорость загрузки и адаптивность к различным устройствам, удовлетворение которых затрудняет создание универсального алгоритма;
* оценка качества дизайна − эстетика и удобство использования имеют субъективный характер, а автоматизация требует разработки метрик, которые могли бы количественно оценить субъективные параметры;
* адаптивность − современные сайты должны адаптироваться к различным экранам (смартфоны, планшеты, настольные компьютеры), что усложняет процесс автоматизации, поскольку необходимо учитывать множество комбинаций;
* креативность − машинным алгоритмам сложно моделировать креативные процессы, которые традиционно требуют человеческой интуиции и художественного вкуса.

Применение генетического алгоритма может решить часть этих проблем за счет эвристического подхода. Главная задача − разработка функции приспособленности, которая бы адекватно оценивала качество созданного дизайна.

# Математическая постановка задачи оптимизации CSS

В математической постановке задача формулируется следующим образом: найти такой набор параметров CSS, который максимизирует (или минимизирует) заданную функцию приспособленности при соблюдении ограничений, связанных с корректностью и эстетикой дизайна.

Пространство поиска включает в себя все возможные комбинации значений параметров CSS, таких как цвета, размеры шрифтов, интервалы между элементами, а также их соответствие стилевым и функциональным требованиям.

## 2.1 Параметры CSS как пространство поиска

Каждый параметр CSS представляет собой одну из осей многомерного пространства поиска. В этой задаче хромосома генетического алгоритма — это вектор, содержащий значения конкретных параметров. Пример параметров:

* цветовые характеристики:

1. основной цвет фона − код в формате RGB или HEX;
2. цвет текста − определяет читаемость и контрастность;

* типографика:

1. размер шрифта − целое или дробное число, выраженное в пикселях, em или процентах;
2. межбуквенный интервал (letter-spacing) и межстрочный интервал (line-height);

* геометрические параметры:

1. размеры блоков − ширина и высота (width, height);
2. отступы и поля (margin, padding);

* дополнительные стили:

1. анимации − время выполнения (animation-duration), тип анимации (ease, linear);
2. прозрачность (opacity).

Пространство поиска имеет высокую размерность, а каждая переменная может принимать значения из непрерывного или дискретного диапазона. Это делает задачу сложной для решения традиционными методами оптимизации, но подходящей для генетического алгоритма.

## 2.2 Описание функции приспособленности

Функция приспособленности является ключевым элементом генетического алгоритма. Она определяет, насколько хорошо данная хромосома (набор параметров CSS) соответствует заданным критериям качества дизайна.

Критерии оценки функции приспособленности могут включать:

* эстетические свойства:
  1. контрастность между фоном и текстом, рассчитанная на основе соотношения яркости;
  2. гармоничность цветовой палитры, которая может быть оценена с помощью теорий цветовых сочетаний (например, использование комплементарных цветов);
* читаемость:
  1. размеры шрифтов, интервалы и соотношение этих параметров;
  2. оптимальная длина строки текста;

Математически функция приспособленности F(x) представлена формулой (1.1)

F(x)=w1\*​C(x)+w2\*​R(x) (1.1)

где

C(x)— оценка эстетических свойств;

R(х) — читаемость;

w1​,w2​​ — веса, задающие приоритет каждого критерия.

# Применение генетического алгоритма для разработки дизайна веб-сайтов на основе CSS

## 3.1 Методы генерации начальной популяции

Генерация начальной популяции в генетическом алгоритме — это процесс создания первых хромосом, которые будут использованы для поиска оптимального решения. В контексте CSS это означает генерацию наборов параметров, соответствующих допустимым стилям веб-страницы.

Основные подходы:

* случайная генерация − каждый параметр CSS принимает случайное значение из допустимого диапазона;
  + генерация на основе шаблонов − используются существующие стили, адаптированные для улучшения;
  + эвристическая генерация − значения параметров задаются с учетом известных правил дизайна;
  + комбинированный подход − часть хромосом генерируется случайно, а часть — на основе существующих шаблонов. Это обеспечивает баланс между разнообразием и качеством.

Эффективная начальная популяция позволяет быстрее достичь оптимального решения.

## 3.2 Генетические операторы

### 3.2.1 Селекция

Селекция — это процесс выбора родителей для последующего скрещивания. Основная цель селекции заключается в том, чтобы выделить наиболее приспособленные особи, сохраняя при этом генетическое разнообразие популяции.

Основные методы селекции:

* + пропорциональная селекция (рулетка) − вероятность выбора индивида пропорциональна значению его функции приспособленности;
  + турнирная селекция − из популяции случайным образом выбирается подмножество индивидов, после чего лучший из них становится родителем. Этот метод позволяет контролировать интенсивность отбора, варьируя размер турнира;
  + ранговая селекция − индивиды упорядочиваются по значениям функции приспособленности, после чего вероятность выбора зависит от их ранга, а не абсолютного значения F(x). Этот метод уменьшает вероятность доминирования высоко приспособленных индивидов, способствуя разнообразию;
  + элитарный отбор − лучшие индивиды из текущей популяции автоматически переходят в следующую без изменений.

Селекция задает направление поиска, выбирая решения, которые приближают оптимум, и отбрасывая худшие.

### 3.2.2 Скрещивание

Скрещивание (кроссовер) — процесс обмена генетической информацией между двумя родителями для создания потомков. Этот оператор обеспечивает комбинирование лучших характеристик родителей в новых хромосомах.

Перейдем к рассмотрению основных методов скрещивания:

* + одноточечное скрещивание − родители делятся на две части в случайной точке, после чего потомок получает первую часть от одного родителя, а вторую — от другого;
  + многоточечное скрещивание − обмен происходит в нескольких случайных точках, создавая более разнообразные комбинации генов;
  + однородное скрещивание − для каждого гена случайно выбирается, от какого родителя он будет унаследован;
  + арифметическое скрещивание: значения параметров потомка вычисляются как среднее арифметическое или взвешенное среднее от соответствующих значений родителей.

Скрещивание является основным механизмом передачи полезных характеристик, что делает его центральным этапом алгоритма.

### 3.2.3 Мутация

Мутация — случайное изменение отдельных генов хромосомы. Этот оператор предотвращает преждевременную сходимость алгоритма, добавляя случайность и разнообразие в популяцию.

Рассмотрим основные типы мутаций:

* + точечная мутация − изменяется одно значение в хромосоме;
  + инверсия − перестановка значений внутри хромосомы;
  + гауссовская мутация − добавление случайного числа, сгенерированного из нормального распределения, к числовым параметрам;
  + удаление и добавление генов − удаление параметров, которые больше не нужны, или добавление новых параметров.

Мутация с заданной вероятностью (обычно 1-5%) предотвращает стагнацию популяции, обеспечивая исследование новых областей пространства поиска.

## 3.3 Критерии остановки алгоритма

Критерии остановки определяют, когда алгоритм прекращает выполнение. Использование подходящих критериев важно для достижения баланса между точностью решения и вычислительными затратами.

Выделим следующие основные критерии остановки:

* достижение максимального числа поколений: алгоритм останавливается после заданного количества итераций;
* стабильность функции приспособленности − алгоритм завершается, если среднее значение функции приспособленности популяции не изменяется на протяжении нескольких поколений;
* достижение целевого значения − если значение функции приспособленности достигает заранее заданного порога, алгоритм завершает выполнение;
* отсутствие улучшений − если за определенное количество поколений лучшая хромосома остается неизменной, процесс считается завершенным.

Каждый из этих критериев может быть применен отдельно или в комбинации, в зависимости от сложности задачи и требований к качеству решения.

## 3.4 Схема работы генетического алгоритма

Рассмотрим операторы генетического алгоритма.

Селекция — это первое действие, выполняемое в ГА при имитации эволюционного процесса. Цель этого оператора – с одной стороны оставить в популяции наиболее приспособленных особей, но с другой – сохранить популяционное разнообразие. Например, если оставлять только наиболее приспособленных, то можно потерять важные фрагменты хромосом у менее приспособленных особей. Возможно, оптимальное решение достигается скрещиванием одного из наиболее приспособленного родителя с другим – наименее приспособленным. Если же на этапе отбора второй родитель окажется утерянным, то оптимальное решение, возможно, будет достигаться за большее число шагов работы ГА. А может быть даже потеряется навсегда. Чтобы уменьшить вероятность такого исхода в следующем поколении следует оставлять не только самых лучших, но и давать возможность менее конкурентным оставаться в популяции. Пример селекции для CSS стилей отображен на рисунке 1.

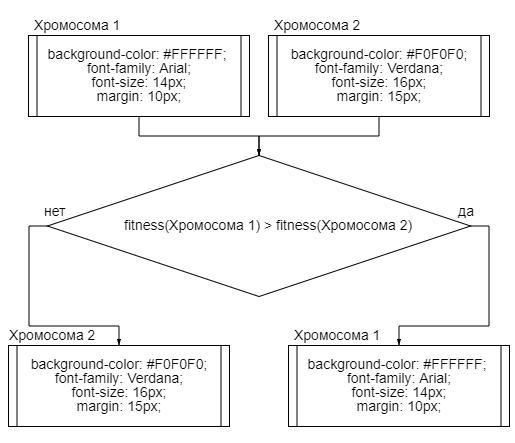


Рисунок 1 – Оператор селекции для CSS стилей

Скрещивание (кроссовер) — на этом этапе происходит обмен данными между частями хромосом родителей. Обычно в популяции перебирают пары родителей, и фрагменты их хромосом перемешивают, получая новый набор генов в хромосомах их потомков. Например, при оптимизации CSS один потомок может унаследовать шрифт от одного родителя, а размер шрифта от другого, что отображено на рисунке 2.

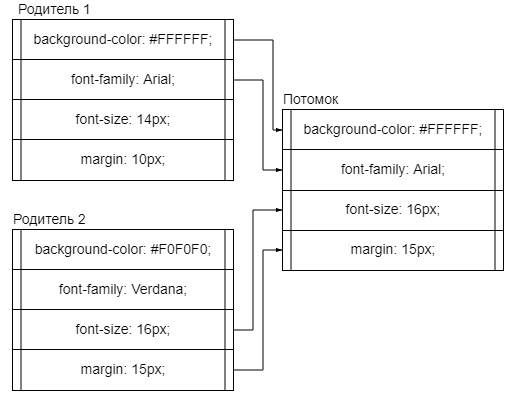


Рисунок 2 – Оператор скрещивания для CSS стилей

Мутация — применяется к полученной популяции и случайным образом с малой вероятностью меняет значения отдельных генов. Этот механизм позволяет расширять область поиска решения задачи и сохранять разнообразие популяции. Возможно, благодаря полезной мутации, особь приобретет новое свойство и станет более конкурентно-способной в своей популяции. В дальнейшем у нее есть все шансы дать потомство и закрепить полезный признак. На рисунке 3 приведен пример мутации, в результате которой был изменен параметр цвета.

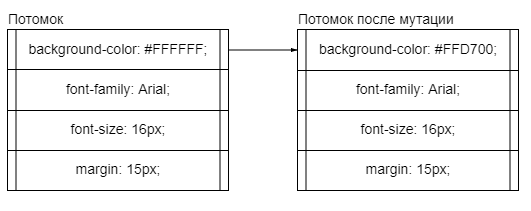


Рисунок 3 – Оператор мутации для CSS стилей

Эти элементы образуют основу работы алгоритма, которая может быть адаптирована для решения различных задач. Более подробно каждый оператор будет рассмотрен далее.

Таким образом, мы можем представить схему работы генетического алгоритма в виде схемы, изображенной на рисунке 4.



Рисунок 4 – Схема работы генетического алгоритма

Операторы генетического алгоритма определяют, как будет происходить эволюция популяции в ходе оптимизации. Основными этапами являются селекция, скрещивание и мутация. Каждый из этих операторов играет ключевую роль в поиске решений, приближающих к глобальному оптимуму.

# Реализация алгоритма на Python

## 4.1 Основные сведения о программе

Программа предназначена для автоматической генерации CSS-стилей с использованием генетического алгоритма. Программа состоит из следующих модулей:

1. модуль генерации CSS −создает начальную популяцию стилей;
2. модуль оценки − анализирует стили и возвращает значение их "качества";
3. модуль генетического алгоритма − управляет эволюцией стилей;
4. запись оптимизированного CSS стиля − происходит в файл с CSS стилями.

Для реализации программы используются библиотеки random (для случайной генерации данных) и jinja2 (для создания HTML-шаблонов).

## 4.2 Работа с HTML и CSS через Python

Python предоставляет возможности для работы с HTML и CSS с помощью библиотек, таких как BeautifulSoup, lxml, а также jinja2 для создания шаблонов.

Для создания CSS-шаблонов и хранения структуры веб-страниц используются CSS-файлы с заполняемыми параметрами стилей. Пример CSS-файла показан на рисунке 5. Значения параметров, такие как ширина (width), высота (heigh) и цвет (color), заполняются с помощью программы на Python.

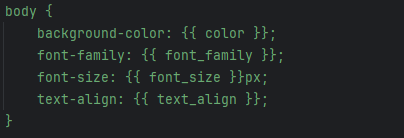


Рисунок 5 – CSS-шаблон с заполняемыми параметрами стилей

Обновление CSS стилей происходит в процессе выполнения программы. Код генерирует CSS-файл, где описываются параметры, соответствующие текущей популяции стилей. Пример кода представлен на рисунке 6. Этот подход позволяет динамически обновлять дизайн в зависимости от параметров генетического алгоритма.

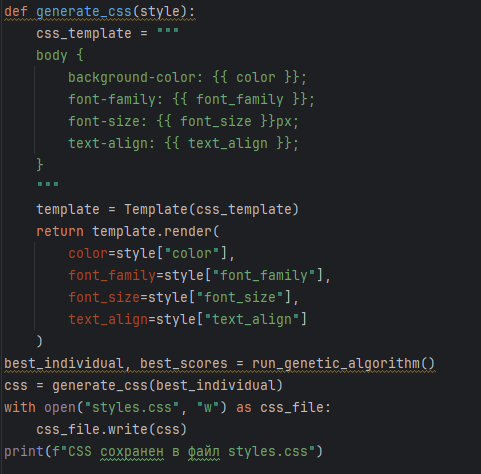
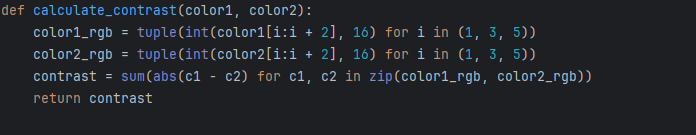


Рисунок 6 – Код обновления и записи CSS стилей в файл

## 4.3 Пример кода для оценки дизайна

Функция приспособленности анализирует визуальные характеристики дизайна, такие как эстетичность, читаемость текста и контрастность. Пример кода для расчета контрастности приведен на рисунке 7.

Рисунок 7 − Код для расчета контрастности

Данная функция оценивает контрастность, размер шрифта и возвращает итоговое значение оценки для конкретного дизайна.

## 4.4 Генерация новых стилей CSS

После выполнения операций генетического алгоритма новые стили CSS создаются на основе лучших решений. Пример кода для генерации и выборки лучших стилей представлен на рисунке 8.

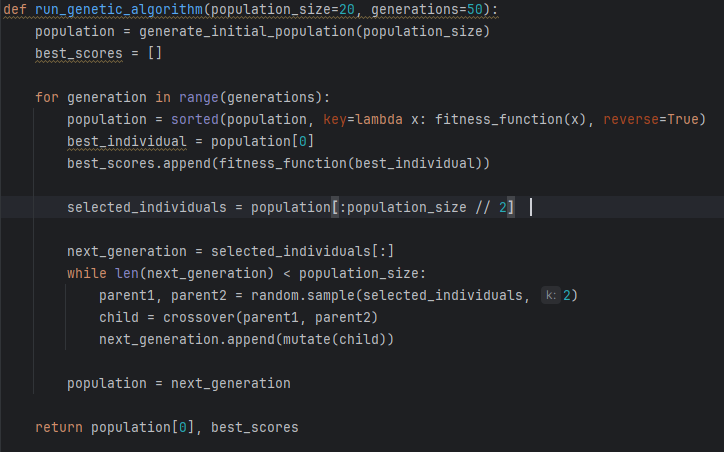


Рисунок 8 – Код генерации и выборки лучших CSS стилей

Сгенерированные стили сохраняются в CSS-файле и применяются к HTML-шаблону, что позволяет визуализировать новые дизайны.

# Анализ результатов оптимизации

## 5.1 Эволюция CSS параметров по поколениям

Для анализа эволюции стилей в генетическом алгоритме отслеживаются изменения параметров CSS по мере прохождения через несколько поколений. Эволюция отображает, как стили, созданные с помощью мутации и скрещивания, с ростом числа итераций становятся более оптимальными, улучшается визуальное качество и соответствие заданным критериям функции оценки.

Произведем исследование зависимости функции оценки от количества поколений (генераций) в генетическом алгоритме. Результаты, полученные при исследовании отражены на рисунке 9.



Рисунок 9 – График прогресса оптимизации в каждой генерации алгоритма

Из данных, полученных из рисунка 9 можно сделать вывод, график показывает, что после 200 поколений улучшения становятся минимальными, это означает, что алгоритм достигает плато, и дальнейшие поколения не дают значительного прогресса. В дальнейшем, при определении временной сложности будем использовать данное число поколений как оптимальное.

## 5.2 Сравнение исходного и оптимизированного дизайна

После нескольких поколений генетического алгоритма создается оптимизированный дизайн. Для сравнения исходного и оптимизированного дизайна используются визуальные и количественные метрики. Оценки качества дизайнов могут включать такие параметры, как читаемость, контрастность, эстетичность и общая пользовательская удовлетворенность.

Увидеть код исходного дизайна можно в приложении Б, сама страница представлена на рисунке 10 и рисунке 11.

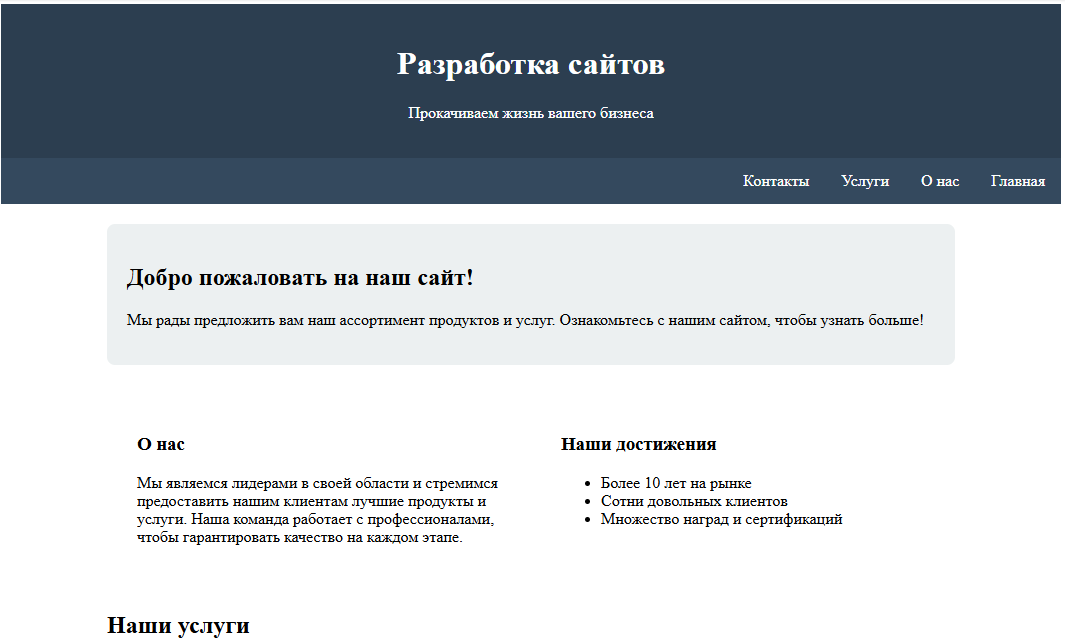


Рисунок 10 – Исходный дизайн веб-страницы (верхняя часть)

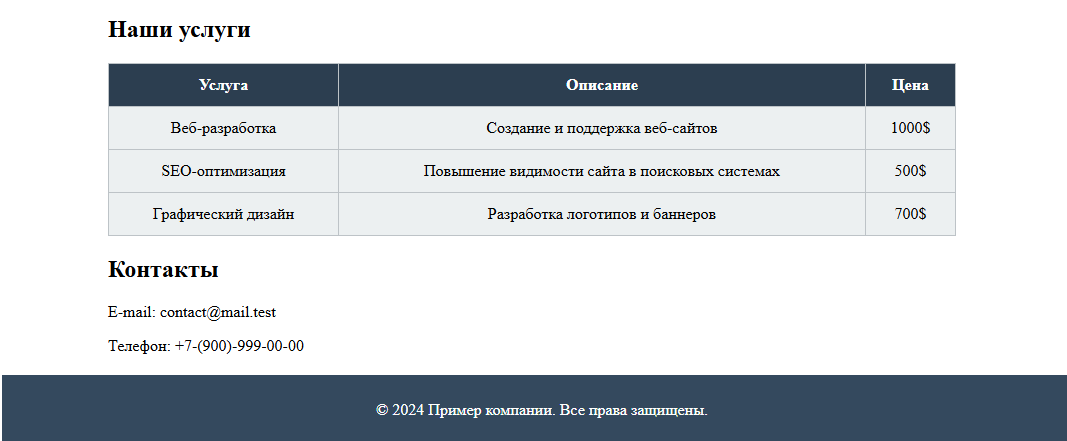


Рисунок 11 – Исходный дизайн веб-страницы (нижняя часть)

Код оптимизированного дизайна представлен в приложении В. Внешний вид веб-страницы с оптимизированными стилями представлен на рисунке 12 и рисунке 13.

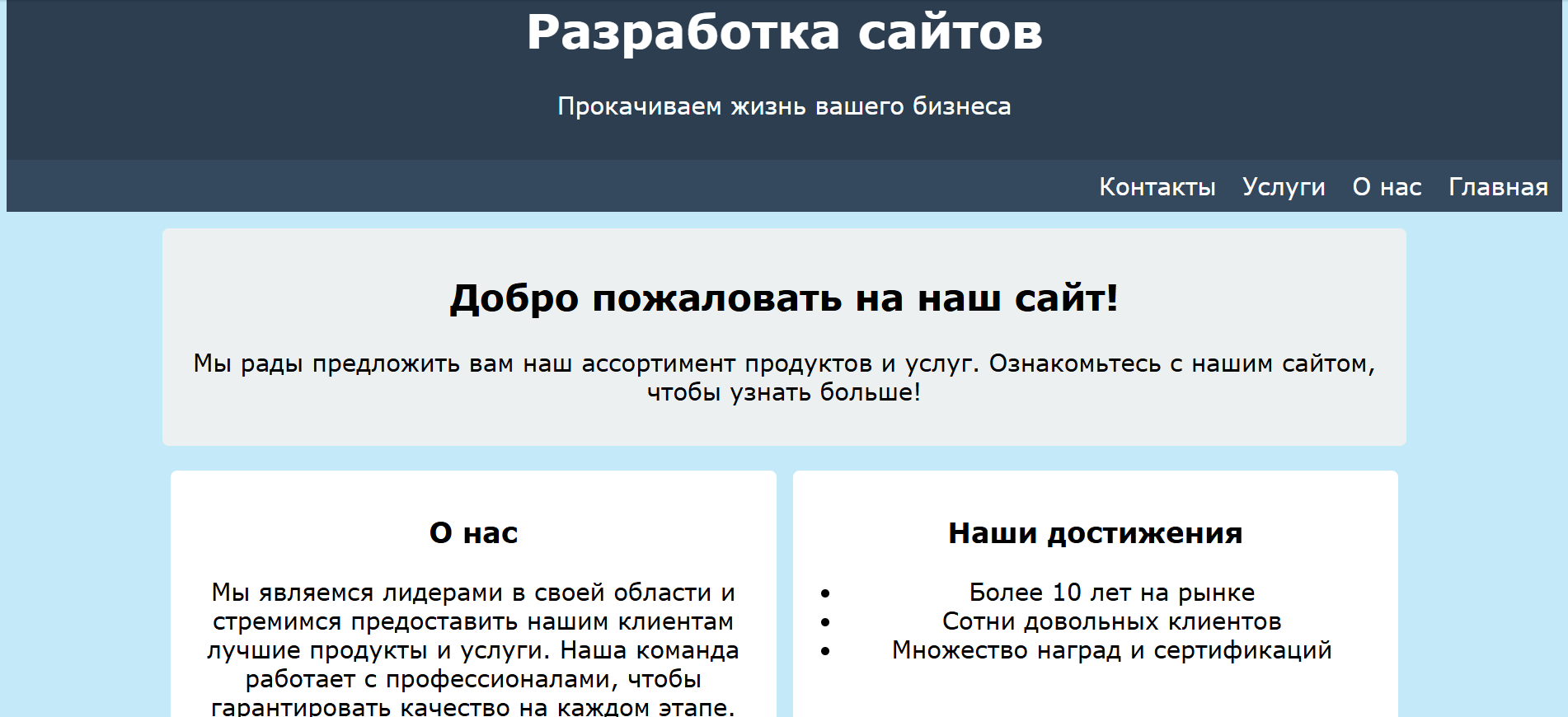


Рисунок 12 – Оптимизированный дизайн веб-страницы (верхняя часть)

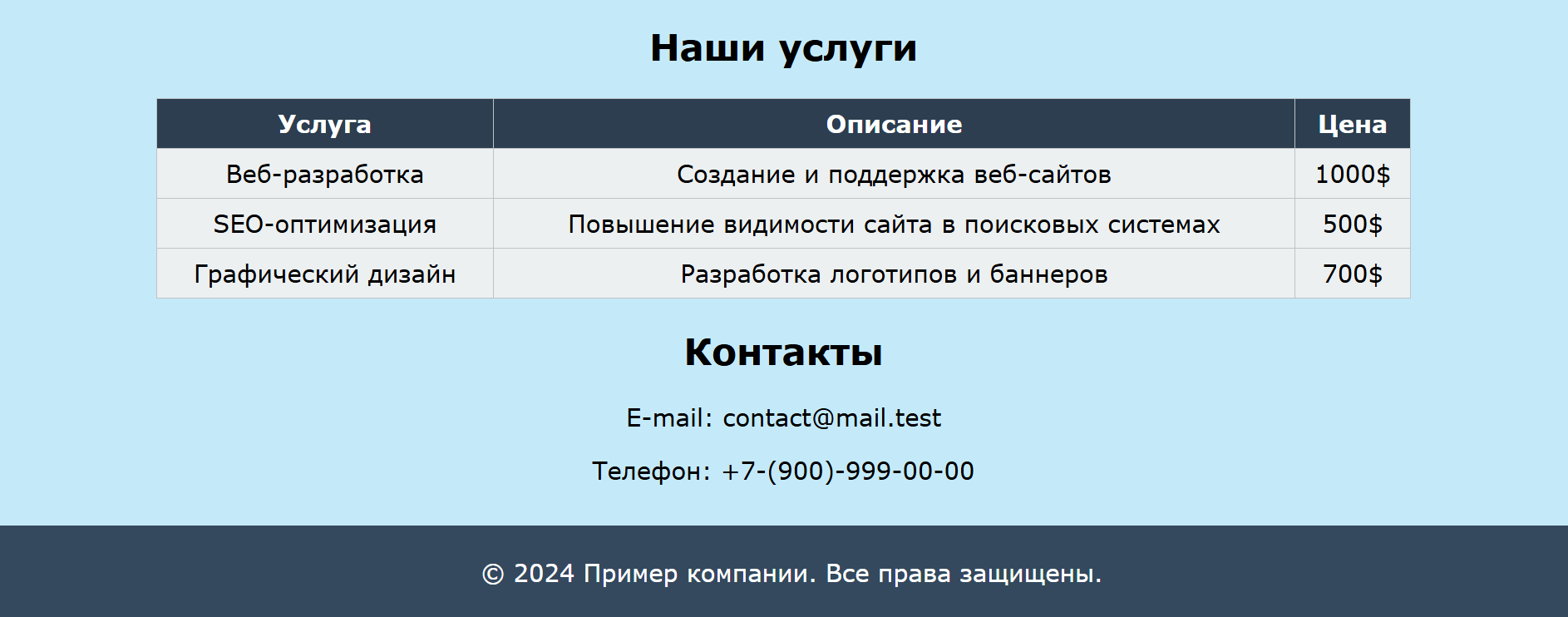


Рисунок 13 – Оптимизированный дизайн веб-страницы (нижняя часть)

## 5.3 Оценка эффективности алгоритма

Для оценки эффективности алгоритма будет исследована зависимость времени выполнения алгоритма от размера начальной популяции. На рисунке 14 представлен график зависимости времени выполнения ГА от размера начальной популяции.

В таблице 1 представлены значения времени выполнения алгоритма при различных значениях начальной популяции.

Таблица 1 – Зависимость времени выполнения ГА от начальной популяции

|  |  |
| --- | --- |
| Начальная популяции | Время выполнения ГА |
| 10 | 16,15 мс |
| 50 | 71,46 мс |
| 100 | 140,94 мс |
| 200 | 278,49 мс |
| 500 | 702,99 мс |

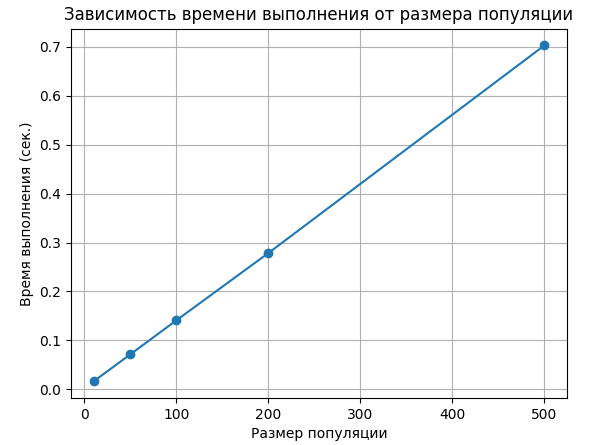


Рисунок 14 – График зависимости времени выполнения ГА от размера начальной популяции

По полученным данным можно сделать вывод, что ГА имеет временную сложность О(n), n – количество особей в начальной популяции. На текущем этапе алгоритм способен справляться с увеличением размеров задач благодаря линейной зависимости времени работы. Однако для задач с ещё более высокой размерностью может потребоваться оптимизация, чтобы сохранить приемлемое время выполнения. Например, применение параллельной обработки.

## ЗАКЛЮЧЕНИЕ

Использование генетического алгоритма для автоматизации веб-дизайна открывает перспективы для дальнейших исследований и практического применения в задачах проектирования интерфейсов и улучшения пользовательского опыта.

В данной работе был реализован генетический алгоритм на Python, позволяющий автоматически генерировать и оценивать стили CSS. Проведено тестирование алгоритма, включая анализ времени выполнения и качества генерируемых решений. Результаты экспериментов показали, что предложенный алгоритм эффективно решает задачу автоматизации дизайна веб-страниц, обеспечивая баланс между эстетикой, функциональностью и производительностью. Экспериментальные исследования продемонстрировали, что алгоритм масштабируется при увеличении размера популяции и количества поколений, сохраняя высокое качество решений. Оптимальное число поколений, определенное экспериментально, составляет 200, что подтверждает способность алгоритма к поиску устойчивых решений за разумное время.

Практическая значимость разработанного подхода заключается в его применении для автоматизации создания веб-дизайна в различных областях, таких как коммерческие сайты, образовательные платформы и информационные порталы. Реализация метода позволяет улучшить процессы проектирования адаптивного дизайна для устройств с разными характеристиками экранов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Голдберг, Д. Э.** Генетические алгоритмы в поиске, оптимизации и машинном обучении / Д. Э. Голдберг ; пер. с англ. А. В. Дьячкова. — Москва : Мир, 1989. — 432 с. — ISBN 5-03-002345-7. — Текст: непосредственный.
2. **Митчелл, М.** Введение в генетические алгоритмы / М. Митчелл ; пер. с англ. О. В. Губарева. — Москва : Лори, 1999. — 208 с. — ISBN 5-85582-091-2. — Текст: непосредственный.
3. Распределенные интеллектуальные информационные системы и среды : монография / А. Н. Швецов, А. А. Суконщиков, Д. В. Кочкин [и др.] ; Министерство образования и науки Российской Федерации, Вологодский государственный университет. — Курск : Университетская книга, 2017. — 196 с. : ил.; 20 см. — Библиогр.: с. 192—196. — 500 экз. — ISBN 978-5-9909988-3-4. — Текст : непосредственный.
4. **Лутц, М.** Изучаем Python. 5-е издание / М. Лутц ; пер. с англ. В. Ястребова. — Санкт-Петербург : Питер, 2013. — 864 с. — ISBN 978-5-496-00160-7. — Текст: непосредственный.
5. Российская книжная палата : [сайт]. - 2018. - URL: http://bookchamber.ru/isbn.html (дата обращения: 22.10.2024). - Текст : электронный.
6. Сайт библиотеки Jinja2 : [сайт]. — URL: https://jinja.palletsprojects.com/en/3.0.x/ (дата обращения: 24.11.2024). — Текст: электронный.
7. Сайт библиотеки Matplotlib : [сайт]. — URL: https://matplotlib.org/ (дата обращения: 27.10.2024). — Текст: электронный.
8. Алгоритмы оптимизации : учебное пособие / Д. Н. Солнцев, М. В. Черепанов ; под ред. А. Г. Турченко. — Новосибирск : НГУ, 2020. — 312 с. — ISBN 978-5-7782-4098-9. — Текст: непосредственный.
9. **Григорьев, А. В.** Алгоритмы и структуры данных на Python / А. В. Григорьев. — Москва : ДМК Пресс, 2018. — 240 с. — ISBN 978-5-94074-899-8. — Текст: непосредственный.

## ПРИЛОЖЕНИЕ А

**Основная программа**

import random  
from jinja2 import Template  
  
def generate\_initial\_population(population\_size):  
 population = []  
 for \_ in range(population\_size):  
 style = {  
 "color": f"#{random.randint(0, 0xFFFFFF):06x}",  
 "font\_size": random.randint(12, 30),  
 "font\_family": random.choice(["Arial", "Verdana", "Times New Roman", "Courier New"]),  
 "text\_align": random.choice(["left", "center", "right"]),  
 }  
 population.append(style)  
 return population  
  
def fitness\_function(style):  
 contrast\_weight = 0.5  
 font\_size\_weight = 0.3  
 alignment\_weight = 0.2  
  
 contrast\_score = calculate\_contrast(style["color"], "#000000")  
 font\_size\_score = max(0, min(1, (style["font\_size"] - 14) / 10)) #от 14-24  
 alignment\_score = 1 if style["text\_align"] == "center" else 0.7 if style["text\_align"] == "right" else 0.5  
  
 return (contrast\_weight \* contrast\_score +  
 font\_size\_weight \* font\_size\_score +  
 alignment\_weight \* alignment\_score)  
  
  
def calculate\_contrast(color1, color2):  
 color1\_rgb = tuple(int(color1[i:i + 2], 16) for i in (1, 3, 5))  
 color2\_rgb = tuple(int(color2[i:i + 2], 16) for i in (1, 3, 5))  
 contrast = sum(abs(c1 - c2) for c1, c2 in zip(color1\_rgb, color2\_rgb))  
 return contrast  
  
def mutate(style):  
 mutation\_choice = random.choice(["color", "font\_size", "font\_family", "width", "height", "text\_align", "display"])  
 if mutation\_choice == "color":  
 style["color"] = f"#{random.randint(0, 0xFFFFFF):06x}"  
 elif mutation\_choice == "font\_size":  
 style["font\_size"] = random.randint(12, 30)  
 elif mutation\_choice == "font\_family":  
 style["font\_family"] = random.choice(["Arial", "Verdana", "Times New Roman", "Courier New"])  
 elif mutation\_choice == "text\_align":  
 style["text\_align"] = random.choice(["left", "center", "right"])  
 return style  
  
def crossover(style1, style2):  
 child\_style = {  
 "color": random.choice([style1["color"], style2["color"]]),  
 "font\_size": random.choice([style1["font\_size"], style2["font\_size"]]),  
 "font\_family": random.choice([style1["font\_family"], style2["font\_family"]]),  
 "text\_align": random.choice([style1["text\_align"], style2["text\_align"]]),  
 }  
 return child\_style  
  
def run\_genetic\_algorithm(population\_size=20, generations=200):  
 population = generate\_initial\_population(population\_size)  
 best\_scores = []  
  
 for generation in range(generations):  
 population = sorted(population, key=lambda x: fitness\_function(x), reverse=True)  
 best\_individual = population[0]  
 best\_scores.append(fitness\_function(best\_individual))  
  
 selected\_individuals = population[:population\_size // 2]  
  
 next\_generation = selected\_individuals[:]  
 while len(next\_generation) < population\_size:  
 parent1, parent2 = random.sample(selected\_individuals, 2)  
 child = crossover(parent1, parent2)  
 next\_generation.append(mutate(child))  
  
 population = next\_generation  
  
 return population[0], best\_scores  
  
def generate\_css(style):  
 css\_template = """  
 body {  
 background-color: {{ color }};  
 font-family: {{ font\_family }};  
 font-size: {{ font\_size }}px;  
 text-align: {{ text\_align }};  
 }  
 """  
 template = Template(css\_template)  
 return template.render(  
 color=style["color"],  
 font\_family=style["font\_family"],  
 font\_size=style["font\_size"],  
 text\_align=style["text\_align"]  
 )  
best\_individual, best\_scores = run\_genetic\_algorithm()  
css = generate\_css(best\_individual)  
with open("styles.css", "w") as css\_file:  
 css\_file.write(css)  
print(f"CSS сохранен в файл styles.css")

## ПРИЛОЖЕНИЕ Б

**CSS стили HTML страницы до работы алгоритма**

header {

background-color: #2c3e50;

color: white;

text-align: center;

padding: 20px;

}

nav {

background-color: #34495e;

overflow: hidden;

}

nav a {

float: left;

display: block;

color: white;

padding: 14px 16px;

text-align: center;

text-decoration: none;

}

nav a:hover {

background-color: #1abc9c;

}

.container {

width: 80%;

margin: 20px auto;

}

.intro {

background-color: #ecf0f1;

padding: 20px;

border-radius: 8px;

margin-bottom: 20px;

}

.content {

display: flex;

justify-content: space-between;

flex-wrap: wrap;

}

.content > div {

flex: 1;

padding: 20px;

margin: 10px;

background-color: #ffffff;

border-radius: 8px;

}

footer {

background-color: #34495e;

color: white;

text-align: center;

padding: 10px;

width: 100%;

bottom: 0;

}

table {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

}

table, th, td {

border: 1px solid #bdc3c7;

}

th, td {

padding: 12px;

text-align: center;

}

th {

background-color: #2c3e50;

color: white;

}

td {

background-color: #ecf0f1;

}

## ПРИЛОЖЕНИЕ В

**CSS стили HTML страницы после работы алгоритма**

body {

background-color: #c4eafa;

font-family: Verdana;

font-size: 29px;

text-align: center;

}

header {

background-color: #2c3e50;

color: white;

text-align: center;

padding: 20px;

}

nav {

background-color: #34495e;

overflow: hidden;

}

nav a {

float: left;

display: block;

color: white;

padding: 14px 16px;

text-align: center;

text-decoration: none;

}

nav a:hover {

background-color: #1abc9c;

}

.container {

width: 80%;

margin: 20px auto;

}

.intro {

background-color: #ecf0f1;

padding: 20px;

border-radius: 8px;

margin-bottom: 20px;

}

.content {

display: flex;

justify-content: space-between;

flex-wrap: wrap;

}

.content > div {

flex: 1;

padding: 20px;

margin: 10px;

background-color: #ffffff;

border-radius: 8px;

}

footer {

background-color: #34495e;

color: white;

text-align: center;

padding: 10px;

width: 100%;

bottom: 0;

}

table {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

}

table, th, td {

border: 1px solid #bdc3c7;

}

th, td {

padding: 12px;

text-align: center;

}

th {

background-color: #2c3e50;

color: white;

}

td {

background-color: #ecf0f1;

}