|  |  |
| --- | --- |
| |  | | --- | | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |

Институт Информационных технологий

Кафедра МОСИТ

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине «Тестирование и отладка программного обеспечения»

**Студент группы** ИКБО-11-17 Алиев Ю.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись студента)*

**Руководитель работы** Басок Б. М. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись руководителя)*

Москва, 2021

# Постановка задачи

Вариант 4

Ввести массив из 10 целых положительных чисел разрядности не более пяти. Вывести количество чисел массива, сумма цифр которых кратна 7 и цифры этих чисел образуют арифметическую прогрессию. Если таких элементов нет, то выдать сообщение об этом.

Для выполнения данного задания взять свой вариант из прилагаемого файла «Варианты заданий для функционального тестирования ПО». Разработать и отладить программу в соответствии с выбранным вариантом задания. Разработать и представить преподавателю исходную документацию для организации тестирования: Техническое задание на разработку и План тестирования программы.

# Техническое задание на разработку программы «Преобразователь массива»

1. Основания для разработки.

Основанием для разработки программы «Преобразователь массива» (в дальнейшем Программы или ППМ) является приобретение практических навыков при организации тестирования и отладки программного продукта, на основании знаний, полученных при изучении дисциплины «Тестирование и отладка программного обеспечения».

2. Назначение разработки.

Автоматическое преобразование введенного пользователем массива положительных чисел разрядности не более пяти и цифры этих чисел образуют арифметическую прогрессию. Если элементы массива не совпадают условию то вывести сообщение пользователю

3. Требование к программным и аппаратным средствам.

3.1. Работа Программы осуществляется на ПЭВМ с любым одноядерным микропроцессором в стандартном окружении.

3.2. Операционная среда Windows 10.

3.3. Язык программирования JavaScript.

3.4. Веб-браузер с поддержкой языка JavaScript.

3.5. Node.js 8.8.1

4. Входные и выходные данные.

4.1. Входной информацией являются вводимые с клавиатуры через запятую разрядности не более пяти и цифры этих чисел образуют арифметическую прогрессию. Количество элементов не ограничено.

4.2. Выходной информацией является результат выполнения операции или же соответствующая ошибка.

5. Требования к основным режимам работы.

5.1. Основные функции Программы.

Программа должна выполнять следующие функции:

5.1.1. Проверка валидации входных данных.

5.1.2 Удаление необходимого элемента.

5.1.3. Выдача информации о проведенной операции.

5.1.4. Реагировать на ошибки ввода пользователя.

5.2. Интерфейс пользователя.

Интерфейс пользователя должен включать следующие элементы:

* Заголовок программы;
* Элемент ввода данных с использованием атрибута placeholder, в котором указан пример валидных данных;
* Кнопку «Получить результат» - предназначенную для проведения основной операции;
* Метку «Результат», после которой должен выводиться результат работы программы.

5.3. Загрузка программы.

Исполнительный модуль программы может быть загружен с диска CDROM или с другой ПЭВМ через локальную сеть.

5.4. Запуск и завершение работы.

5.4.1. Исполнительный модуль программы запускается при помощи расширения VS code Live Server. После выполнения этой операции необходимо открыть веб-браузер и ввести в адресную строку 127.0.0.1:5500/index.html.

5.4.2. Для завершения или прерывания работы нужно просто закрыть страницу браузера.

5.5. Ошибки пользователя.

Программа должна выявлять следующие ошибки пользователя:

* попытку ввода пустой строки;
* попытка ввода больше 10 чисел;
* попытка ввода меньше 10 чисел
* попытку ввода в недопустимой форме обработки элементов;
* попытку ввода чисел, не удовлетворяющих условию задачи.

6. Требования к программной документации.

Программа должна быть описана в следующих документах:

* Техническое задание на разработку;
* Руководство пользователя.

7. Стадии и этапы разработки.

1. Разработка ТЗ - 18.02.21

2. Разработка альфа-версии - 18.02.21

3. Разработка бета-версии - 18.02.21

4. Разработка документации - 19.02.21

5. Тестирование программы и исправление ошибок - 20.02.21

6. Подготовка и проведение приемо-сдаточных испытаний - 20.02.21

8. Порядок контроля и приемки.

Контроль и приемка Программы осуществляется на основании представленных:

* ТЗ на разработку Программы;
* руководства пользователя;
* исполнительного модуля Программы;
* результатов тестирования;
* контрольного (приемо-сдаточного) примера.

# План тестирования программы «Преобразователь массива»

1. Идентификатор тестового плана.

ППМ - ПТ - 1.0.

2. Ссылки на используемые документы.

Данный план разработан на основании стандарта IEEE Std 829-1998 (Стандарт на тестовую документацию) и Технического задания на разработку программы «Преобразователь массива» (в дальнейшем Программы или ППМ).

3. Введение

Целью данного документа является детальная разработка плана тестирования всех возможностей Программы, определение программных и аппаратных средств для реализации этого плана и документов для хранения и отображения результатов тестирования.

4. Тестируемые элементы

4.1. Тестируемая версия – версия 1.0 программы «Преобразователь массива»

4.2. Исправление ошибок - Все найденные и исправленные в ходе тестирования ошибки должны быть проверены.

4.3. Документация - Вся разработанная документация должна соответствовать Техническому заданию и стандартам ЕСПД.

5. Проблемы риска тестируемого продукта.

В связи с тем, что программисты не предоставляют специалистам по тестированию исходные коды программы, практически исключена возможность тестирования программы ППД методом "стеклянного ящика".

6. Свойства, подлежащие тестированию.

6.1. Загрузка исполнительного модуля с носителя.

6.2. Загрузка исполнительного модуля через локальную сеть.

6.3. Запуск Программы.

6.4. Функции интерфейса пользователя:

* ввод и редактирование исходных данных;
* запуск процедуры вычисления результата;
* вывод результатов работы программы.

6.5. Процедуры обработки входных данных в соответствии с принятым алгоритмом и выбранным направлением преобразования.

6.6. Функционирование Программы в операционной среде (ОС) Windows XP, Windows 7 и Windows 10.

6.7. Стабильное многократное выполнение программы без перезапуска.

6.8. Обработка ошибок оператора.

6.9. Завершение программы и выход в ОС.

6.10. Программная документация, включающая:

* ТЗ на разработку ППМ;
* руководство пользователя.

7. Свойства, не подлежащие тестированию

Тестированию не подлежит возможность прерывания работы ППД до ее нормального завершения, поскольку данное свойство Программы реализуется системными средствами и не зависит от конкретной задачи.

8. Подход (Стратегия тестирования)

8.1. При разработке и планировании тестов используется метод "черного ящика", который основывается на проверке интерфейса, ресурсов и основных функций Программы.

8.2. Функции Программы проверяются независимо друг от друга. При этом предполагается, что все функции Программы кроме проверяемой в данный момент функции выполняются без ошибок. Такой подход основывается на принципе редкой компенсации кратных отказов. В этом случае гарантируется, что тесты, выявляющие ошибки при реализации отдельных функций обнаружат их и в том случае, если неверно реализуется любое сочетание этих функций.

8.3. Тестирование первичной приемки.

Смоук - тестирование реализуется каждый раз перед началом полного цикла тестирования.

8.4. Регрессионное тестирование

Регрессионное тестирование осуществляется каждый раз после сборки очередной версии Программы, содержащей исправления ранее обнаруженных ошибок и / или изменения отдельных функций ППД или тестов. В зависимости от сложности ошибки и уровня ее влияния на выполнение программы в качестве регрессионных тестов могут выступать либо тесты полной проверки Программы, либо смоук - тест.

8.5. Тестирование стабильной работы Программы

Для проверки стабильной работы Программы используется пакет Ultimate Visual Studio компании Microsoft версий 2013 и выше, с помощью которого автоматически многократно повторяется выполнение программы без перезапуска, и Диспетчер задач Widows, фиксирующий изменение объемов выделенной памяти во времени.

8.6. Тестирование производительности

Программа выполняется в интерактивном режиме и ее производительность определяется скоростью заполнения наборного поля диалогового окна. Поэтому тестирование производительности не производится.

8.7. Нагрузочное тестирование

Нагрузочное тестирование ППД включает:

* выполнение Программы в мультизадачной среде;
* выполнение Программы на компьютере с минимально возможной тактовой частотой и минимальным объемом памяти;
* выполнение программы на фоне печати текстового файла на принтере.

8.8. Тестирование граничных условий

Тестирование граничных условий включает:

* тестирование обработки граничных значений;
* тестирование обработки данных, не соответствующих граничным условиям.

8.9. Тестирование на "не тронутой" машине (The virgin machine)

Все процедуры тестирования должны быть реализованы также на ПЭВМ с заново установленной операционной системой и необходимым для функционирования и тестирования Программы программным обеспечением. Использование данной конфигурации позволит выявить ошибки, принадлежащие исключительно Программе, а также те ее ошибки, которые возникают только при первом запуске Программы.

8.10. Отслеживание обнаруженных ошибок.

Все обнаруженные тестером ошибки заносятся в базу данных ошибок, доступную также программистам и руководителю проекта. Если программист исправляет ошибку, то он фиксирует это в базе данных ошибок. Тестер проверяет качество исправления ошибки и фиксирует это в базе данных ошибок. Он либо подтверждает факт исправления ошибки, либо открывает ее вновь. Право закрыть проблему предоставляется либо, руководителю проекта, либо руководителю группы программистов.

9. Критерий тестирования при первичной приемке.

Для оперативной оценки качества Программы используется смоук - тест. Только после успешного прохождения смоук - теста данная программа может быть принята для проведения полного цикла тестирования. Данные тест приводится в документе “Программа «Преобразователь массива». Тест первичной приемки".

10.Критерий прохождения тестов.

Критерий успешных и неудачных испытаний для каждого тестового случая описывается через ожидаемые - эталонные результаты. Если после прогона всех тестовых случаев, образующих тест, полученные результаты совпадают с эталонными, то тест считается успешным. Если после прогона, по крайней мере, одного тестового случая ожидаемый результат не получен, считается, что тест потерпел неудачу. Испытания Программы считаются успешными, если успешными оказались все тесты, включенные в систему тестов ППД.

11. Критерий приостановки и возобновления работ

Работы приостанавливаются до решения проблем в следующих случаях:

* при обнаружении тестом катастрофических ошибок, приводящих к аварийному завершению Программы;
* при тестировании обнаруживается, что не совпадает более 50% всех эталонных и полученных результатов;
* при отказе оборудования;
* при изменении отдельных функций Программы.

Тестирование возобновляются по решению руководителя разработки после устранения перечисленных причин приостановки работ.

12.Тестовая документация.

Для тестирования Программы должны быть составлены следующие документы:

* план тестирования;
* тест первичной приемки;
* совокупность проверочных тестовых случаев, обеспечивающих полный цикл тестирования;
* база данных ошибок и отчеты на основе ее содержимого;
* проверочный лист (CheckList);
* итоговый отчет по результатам тестирования.

13. Задачи тестирования.

В ходе подготовки к тестированию и его проведения решаются следующие задачи:

* изучение объекта тестирования (программы «Преобразователь массива»);
* разработка плана тестирования;
* разработка системы проверочных тестов и тестовых случаев;
* разработка документации для проведения процедуры тестирования;
* реализация процедур тестирования;
* ведение базы данных ошибок;
* разработка отчетов о результатах тестирования.

14. Необходимый персонал и обучение.

Для решения задач тестирования требуется один специалист в области тестирования программных средств. Кроме практического опыта, связанного с разработкой контрольных тестов и организацией собственно процесса тестирования, он должен иметь опыт работы со средствами автоматизации тестирования и навыки программиста.

15. Требования окружающей среды.

Для успешного тестирования Программы необходимы следующие аппаратные и программные средства:

* ПЭВМ с микропроцессором, работающем на одном и более ядре (типа Pentium) в стандартном окружении.
* операционная среда: Windows 10.
* программные и аппаратные средства для работы в сети;
* загрузочный модуль тестируемой ППД на носителе;
* сетевая база данных ошибок JIRA компании Atlassian Software. Используемая версия – JIRA Enterprise Edition 3.8 #209;
* Ultimate Visual Studio компании Microsoft версий 2013 и выше, предназначенный для автоматизации отдельных тестовых случаев.

16. Распределение ответственности

Для тестирования данной программы требуется один тестер. В его обязанности входит:

* разработка всей указанной в п. 12 тестовой документации;
* разработка системы тестов Программы;
* реализация тестовых процедур;
* ведение базы данных ошибок;
* подготовка отчетов по результатам тестирования.

17. График работ

Работы по разработке и отладка тестов и тестовой документации производятся в соответствии с данным календарным планом:

Таблица 1. Календарный план тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **N п.п.** | Задача | Дата начала | Дата конца |
| 1 | Изучение технического задания | 20.02.2021 | 20.02.2021 |
| 2 | Разработка и согласование плана тестирования | 20.02.2021 | 20.02.2021 |
| 3 | Разработка спецификации комплекса тестов | 20.02.2021 | 20.02.2021 |
| 4 | Разработка и отладка ручных и  автоматических тестов | 21.02.2021 | 21.02.2021 |
| 5 | Разработка тестовой документации | 21.02.2021 | 21.02.2021 |
| 6 | Реализация тестовых процедур  тестов | 22.02.2021 | 22.02.2021 |
| 7 | Тестирование рабочей документации | 22.02.2021 | 22.02.2021 |
| 8 | Корректировка тестов и тестовой  документации | 22.02.2021 | 22.02.2021 |

18. Риски и непредвиденные обстоятельства

Таблица 2. Риски

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N п.п.** | **Риск** | **Вероятность** | **Влияние** | **Смягчение влияния** |
| 1 | Разработка тестовой документации, а так же процесс реализации тестовых процедур и тестирования рабочей документации могут задержаться из-за возможных задержек при разработке и отладке Программы. | 35% | Основное | Тестер должен находиться в тесном контакте с разработчиком Программы и, по мере получения новой информации корректировать тесты и тестовую документацию. |

19. Утверждение плана

План тестирования утверждает руководитель проекта Васильев И. С.

20. Глоссарий

20.1 Термины.

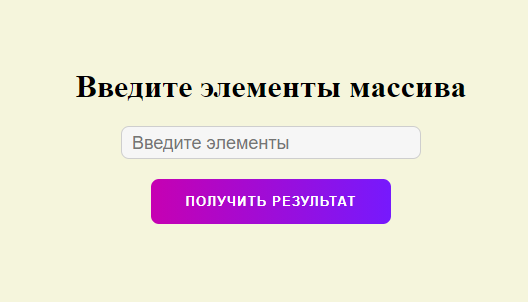
* Тест – это совокупность входных данных, предназначенных для проверки отдельных свойств Программы и выявления дефектов, искажающих эти свойства.
* Тестовый случай – Элементарная составная часть теста, минимальный тестовый модуль, допускающий независимое выполнение от начала до конца. Тестовый случай состоит из одной или нескольких проверок, объединенных одной целью.
* Тестер – участник процесса тестирования программного продукта.

20.2 Условные сокращения.

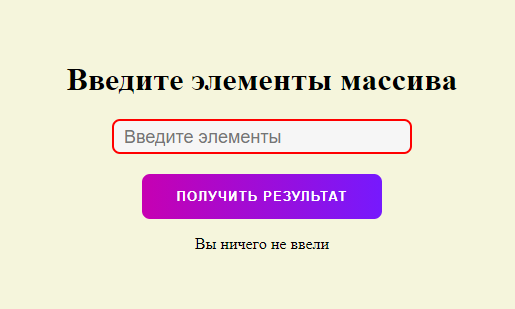
* Программа или ППД – Программа "Преобразование данных".

# Приложение 1

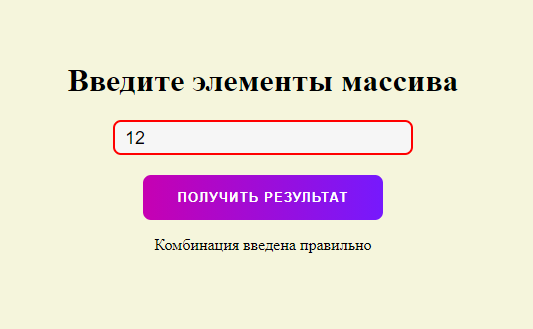
Ниже приводятся рисунки различных состояний интерфейса пользователя программы «Преобразователь массива».



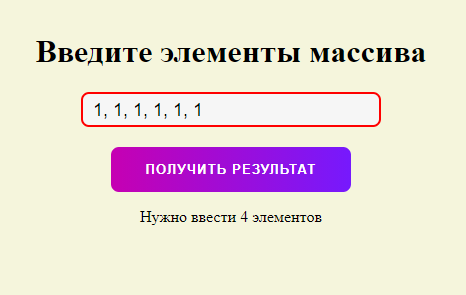
*Рисунок 1. Интерфейс пользователя программы «Преобразователь массива» при открытии программы*



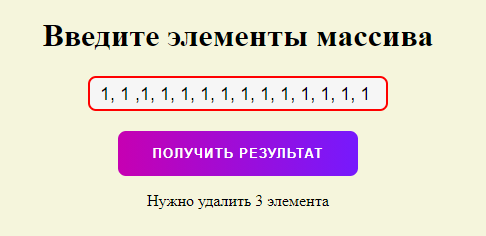
*Рисунок 2. Интерфейс пользователя программы «Преобразователь массива» при заполнении не всех элементов массива и нажатии кнопки «Получить результат»*



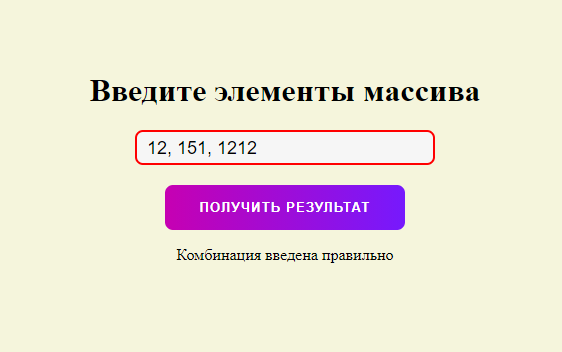
*Рисунок 3. Интерфейс пользователя программы «Преобразователь массива» при заполнении элементов массива в количестве одного элемента*



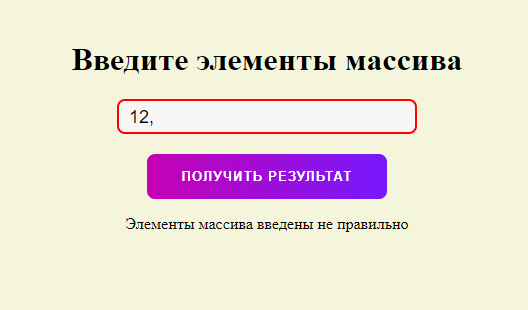
*Рисунок 4. Интерфейс пользователя программы «Преобразователь массива» при длине массива меньше чем 10 элементов*



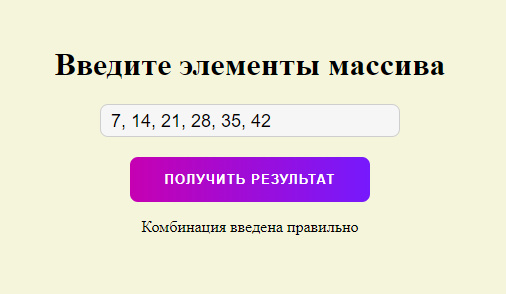
*Рисунок 5. Интерфейс пользователя программы «Преобразователь массива» при длине массива больше чем 10 элементов*



*Рисунок 6. Интерфейс пользователя программы «Преобразователь массива» при заполнении элементов не соответствующих условию задания*



*Рисунок 7. Интерфейс пользователя программы «Преобразователь массива» при заполнении элементов массива в количестве одного элемента с запятой*



*Рисунок 8. Интерфейс пользователя программы «Преобразователь массива» при заполнении элементов массива и нажатии кнопки «Получить результат» при соблюдении всех требований*

# Приложение 2

Ниже приводятся исходный текст программы «Преобразователь массива», и тексты включенных в нее модулей.

**Листинг программы «Преобразователь массива».**

|  |
| --- |
| // Преобразвывает строку в массив и изменяет тип строки на цифры  const addNewArrNum = (str) => {  const newArrayString = str.split(', ');  const newArrayNum = newArrayString.map((el) => Number(el));  return newArrayNum;  }  // Возвращает массив  const numSeven = (arr) => {  const newArr = arr.filter((el) => el % 7 === 0);  return newArr;  }  // Возвращает boolean  const arithmeticProgression = (arr) => {  const step = arr[1] - arr[0];  let valid = false;  arr.map((el, index) => {  if (index !== arr.length - 1) {  valid = el + step === arr[index + 1]  }  });  if(valid) {  inputArr.classList.remove('error')  console.log('Удовлетворяет требованиям')  } else if (!valid) {  inputArr.classList.add('error');  console.log('Таких элементов нет')  }  return (valid && arr) || valid;  } |
|  |

Листинг файла index.js. Исходный код программы.

**Текст модуля №1.**

|  |
| --- |
| const inputArr = document.querySelector('.main\_\_input');  const mainButton = document.querySelector('.main\_\_button');  const textError = document.querySelector('.footer-text\_\_error');  const p = document.getElementsByTagName('p');  const regex = /^((\d\d?\d?\d?\d?)(,\s)?)+$/;  const result = null;  mainButton.addEventListener('click', () => {  const inputValue = inputArr.value;  let str = null;  if (inputValue === '') {  inputArr.classList.add('error');  textError.innerHTML = '';  textError.insertAdjacentHTML('afterbegin', `<p>Вы ничего не ввели</p>`)    } else if (!regex.test(inputValue)) {  inputArr.classList.add('error');  textError.innerHTML = '';  textError.insertAdjacentHTML('afterbegin', `<p>Элементы массива введены не правильно</p>`)  } else {  str = inputValue;  arithmeticProgression(numSeven(addNewArrNum(str)));  textError.innerHTML = '';  textError.insertAdjacentHTML('afterbegin', `<p>Комбинация введена правильно</p>`)  }  });  // Преобразвывает строку в массив и изменяет тип строки на цифры  const addNewArrNum = (str) => {  const newArrayString = str.split(', ');  const newArrayNum = newArrayString.map((el) => Number(el));  return newArrayNum;  }  // Возвращает массив  const numSeven = (arr) => {  const newArr = arr.filter((el) => el % 7 === 0);  return newArr;  }  // Возвращает boolean  const arithmeticProgression = (arr) => {  const step = arr[1] - arr[0];  let valid = false;  arr.map((el, index) => {  if (index !== arr.length - 1) {  valid = el + step === arr[index + 1]  }  });  if(valid) {  inputArr.classList.remove('error')  console.log('Удовлетворяет требованиям')  } else if (!valid) {  inputArr.classList.add('error');  console.log('Таких элементов нет')  }  return (valid && arr) || valid;  } |

Листинг файла index.js. Исходный код программы.

**Текст модуля №2.**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  <link rel="stylesheet" href="./style/index.css">  </head>  <body>  <div class="container">  <header>  <div class="title">  <h1>Введите элементы массива</h1>  </div>  </header>  <main>  <input type="text" class="main\_\_input" placeholder="Введите элементы"/>  <button class="main\_\_button btn btn1">Получить результат</button>  </main>  <footer>  <div class="footer-text\_\_error"></div>  </footer>  </div>  <script src="./index.js" defer></script>  </body>  </html> |

Листинг файла index.html. Исходный код программы.

**Текст модуля №3**

|  |
| --- |
| \* {  box-sizing: border-box;  }  body {  background-color: beige;  }  input {  outline: none;  }  .container {  display: flex;  align-items: center;  justify-content: center;  flex-direction: column;  }  header {  margin-top: 5%;    }  main {  display: flex;  flex-direction: column;  align-items: center;  justify-content: center;  }  .main\_\_input {  width: 300px;  font-size: 13px;  padding: 6px 0 4px 10px;  border: 1px solid #cecece;  background: #F6F6f6;  border-radius: 8px;  font-size: 18px;  margin-bottom: 20px;  }  .error {  border: 2px solid red;  }  .main\_\_button {  }  /\* button \*/  .btn {  margin: 0 20px;  padding: 15px 34px;  overflow: hidden;  font-family: 'Montserrat Alternates', sans-serif;  color: white;  letter-spacing: 1px;  text-transform: uppercase;  font-weight: 600;  border-width: 0;  transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);  outline: none;  cursor: pointer;  border-radius: 8px;  }  .btn1 {  background: linear-gradient(90deg, #c600b2, #7619fd);  }  .btn:before, .btn:after {  content: "";  position: absolute;  z-index: -1;  width: 100%;  height: 100%;  left: -110%;  top: 0;  opacity: 0.1;  background: black;  transform: skewX(25deg);  transition: .5s ease;  }  .btn:hover:before {  left: -10%;  transition: .3s ease;  }  .btn:hover:after {  left: -20%;  } |

Листинг файла style.css. Исходный код программы.