

Создание веб-приложения для открытого урока в рамках дисциплины "Основы алгоритмизации и программирования"

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В данной статье описана реализация веб-ориентированного приложения для демонстрации базовых алгоритмов в программировании. Представлено описание реализации ключевых компонентов на страницах сайта на языке программирования JavaScript и результаты разработки в виде изображений. Описывается работа интерактивного веб-приложения и способы взаимодействия с ним при изучении алгоритмизации и программирования на различных уровнях подготовки.

Ключевые слова: JavaScript, HTML, Python, веб-программирование, разработка сайта, основы алгоритмизации и программирования

Creating a web application for an open lesson as part of the course "Fundamentals of Algorithms and Programming"

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes the implementation of a web-based application for demonstrating basic algorithms in programming. It provides a description of the implementation of key components on the website pages in the JavaScript programming language and the results of the development in the form of images. It describes the operation of the interactive web application and ways of interacting with it when studying algorithmization and programming at various levels of training.

Keywords: JavaScript, HTML, Python, web programming, website development, fundamentals of algorithms and programming

1. Введение

1.1 Актуальность

Изучение программирования на различных уровнях обучения (будь то обучение в вузе или школе) сложно описать без визуальной демонстрации. Такой проблемы не лишены в том числе и различные алгоритмы, такие как сортировки или простые числа.

Правильная реализация визуальной демонстрации подобных алгоритмов является актуальной задачей, ведь в зависимости от предложенного результата будет зависеть и уровень усвоения материала обучающимися: интерактивный материал воспринимать и запоминать будет легче в случае алгоритмов, чем запоминание больших блоков кода (независимо от языков программирования).

1.2 Обзор исследований

Д.А. Кондратенко реализовал алгоритмы сортировки и предложил способ визуализации массива и производимых с ним операций [1]. Программа в рамках работы создана для операционной системы Windows с применением графической подсистемы GDI.

А.О. Гюллинг и Н.В. Воронцова описали алгоритм и программу решения задачи визуализации алгоритма Дейстры [2]. Программа разработана на C# и послужила источником для визуализации рекурсивных алгоритмов.

П.А. Шелкунов в своей работе реализовал программу визуализации алгоритма «Скатерть Улама», который позволяет изучить распределение простых чисел относительно друг друга на числовой спирали [3].

В работе Д.В. Моглан рассмотрен дидактический потенциал использования визуализатора алгоритмов как программного обеспечения, наглядно демонстрирующего работу алгоритмов по обработке набора входных данных [4]. Описаны трудности, возникающие при изучении основ алгоритмизации и программирования, и предложено применять визуализатор на лекционных занятиях для более успешного усвоения механизма работы алгоритмов.

В.Ю. Подставников, Д.В. Фомин и С.Г. Самохвалова рассмотрели существующие криптографические алгоритмы и программы визуализации их работы [5]. В работе обоснована актуальность разработки собственного программного инструмента визуализации криптоалгоритмов и выполнен выбор средств его разработки.

1.3 Цель исследования

Цель – разработать веб-приложение для демонстрации базовых алгоритмов в рамках дисциплины «Основы алгоритмизации и программирования» для обучения студентов второго курса специальности 09.02.07 Информационные системы и программирование.

2 Материалы и методы

Для решения поставленных задач выбран язык программирования JavaScript, язык разметки HTML и для примеров алгоритмов на страницах приложения язык программирования Python

3 Результаты и обсуждения

Продемонстрировать работу алгоритмов с помощью стандартных средств создания презентаций к учебным занятиям во многих случаях тяжело

или невозможно. В таком случае разработка собственного приложения для демонстрации работы алгоритмов является оправданной и полезной задачей. Разработанные материалы можно использовать на любых этапах изучения программирования, так как обучающиеся смогут наблюдать не только программный код, но и видеть его работу в реальном времени.

Решено разработать небольшой сайт с демонстрацией алгоритмов. Разработка сайта имеет ряд преимуществ над приложением для ОС, одним из которых является независимость от платформы, что позволит запустить демонстрацию на любом компьютере, у которого установлен браузер.

На сайте реализованы следующие разделы:

1. Простые числа
2. Длинная арифметика
3. Рекурсивные алгоритмы
4. Алгоритмы сортировок

Во всех разделах присутствуют компоненты демонстрации работы каждого описанного алгоритма. Сам сайт представляет информацию в формате стандартной презентации, что позволяет использовать написанный материал в образовательном процессе (см. рис. 1).

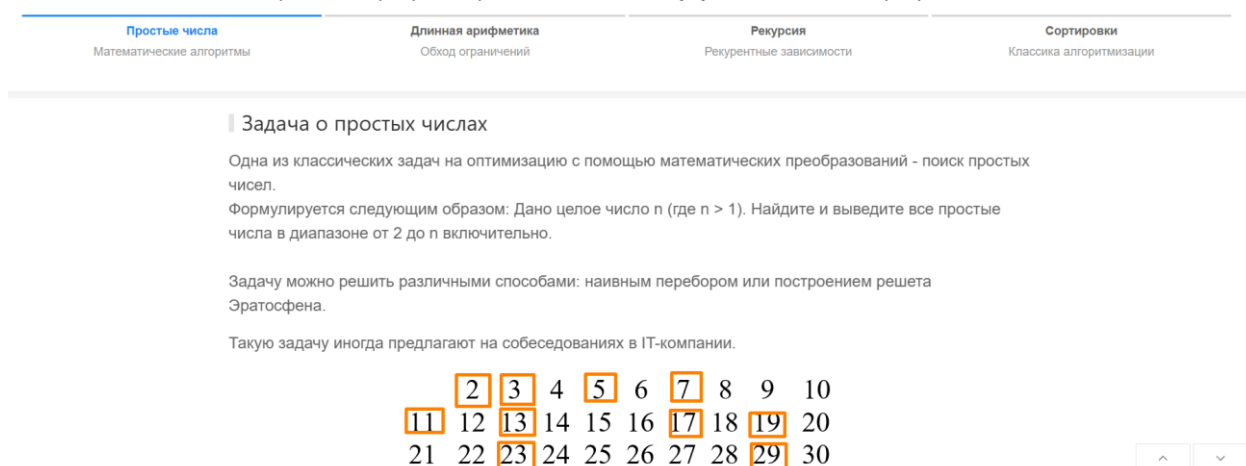


Рисунок 1. Пример страницы с описанием задачи о простых числах

В разделе простых чисел представлены следующие алгоритмы:

1. Наивный способ нахождения простых чисел и его оптимизация.
2. Решето Эратосфена.

На слайдах с описанием алгоритмов есть также и демонстрация реализации алгоритма на языке программирования Python (язык программирования пользуется популярностью в обучении основам программирования, что делает возможным изучение алгоритмов в том числе и школьниками, обучающихся программированию на Python).

■ Наивный способ определения простых чисел

Самая простая реализация поиска простых чисел сводится к перебору диапазона [2:n] (где n - число), в котором производится поиска делителя числа n.

На больших числах такой алгоритм крайне не эффективен и программа не справится с объемом вычислений.

Главный плюс такого метода поиска: простота реализации относительно продвинутых алгоритмов.

```
def is_prime_naive(n):  
    if n < 2:  
        return False  
    for i in range(2, n):  
        if n % i == 0:  
            return False  
    return True  
  
def naive_primes(n):  
    primes = []  
    for num in range(2, n + 1):  
        if is_prime_naive(num):  
            primes.append(num)  
    return primes
```

Код алгоритма

Рисунок 2. Слайд с демонстрацией наивного способа нахождения простых чисел

Для дальнейшего сравнения эффективности каждого из способа нахождения простых чисел написаны соответствующие алгоритмы на JavaScript. Все алгоритмы представлены в виде асинхронных функций, что позволяет замерить скорости выполнения каждого из алгоритмов в реальном времени:

```
async function findPrimesBruteForce(n) {  
    let primes = [];  
    for (let i = 2; i <= n; i++) {  
        let isPrime = true;  
        for (let j = 2; j < i; j++) {  
            if (i % j === 0) {  
                isPrime = false;  
                break;  
            }  
        }  
        await displayNumber(i, isPrime);  
        if (isPrime) primes.push(i);  
    }  
    return primes;  
}
```

Также создан отдельный слайд для демонстрации скорости работы алгоритмов поиска простых чисел. Статистика представлена в виде соответствующей диаграммы, на которой видно, что наивный способ в несколько раз слабее оптимизированного варианта перебора делителей до корня самого числа или решета Эратосфена (см. рис. 3).

Демонстрация алгоритмов поиска простых чисел

Введите n:

10000

Тест скорости

ЗАПУСТИТЬ

ПОКАЗАТЬ/СКРЫТЬ ГРАФИК

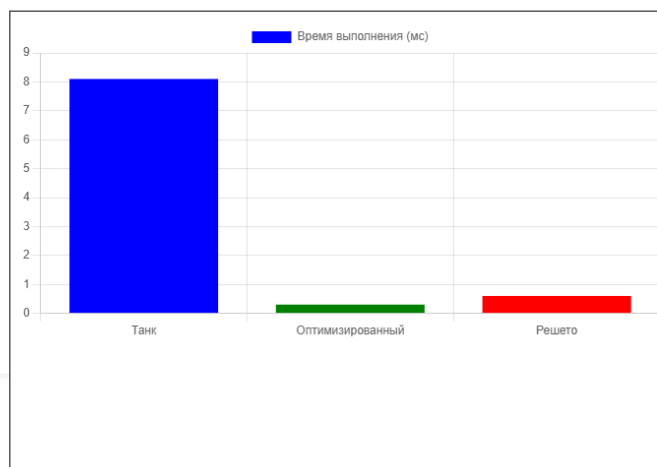


Рисунок 3. Сравнение скорости выполнения алгоритмов поиска простых чисел

Для построения графика написана соответствующая функция drawChart():

```
function drawChart() {
    const ctx =
document.getElementById('chart').getContext('2d');
    if (window.barChart) window.barChart.destroy();
    window.barChart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: chartData.map(d => d.algorithm),
            datasets: [{
                label: 'Время выполнения (мс)',
                data: chartData.map(d => d.time),
                backgroundColor: ['blue', 'green', 'red']
            }]
        }
    });
}
```

Также обучающиеся могут получить список простых чисел до N , который будет выводиться в реальном времени с подсветкой рамки для каждого числа: зелёный цвет обозначает простые числа, а красный все остальные соответственно (см. рис. 4). Для отображения списка целых чисел также написана функция на JavaScript, которая получает пару значений, в котором одно является самим числом, а второе это логическое значение isPrime, определяющее – является ли число простым:

```
async function displayNumber(num, isPrime) {
    return new Promise(resolve => {
```

```

        setTimeout(() => {
            const div = document.createElement('div');
            div.classList.add('number', isPrime ? 'prime' :
'composite');
            div.textContent = num;

document.getElementById('results').appendChild(div);
            resolve();
        }, 50);
    });
}

```

Демонстрация алгоритмов поиска простых чисел

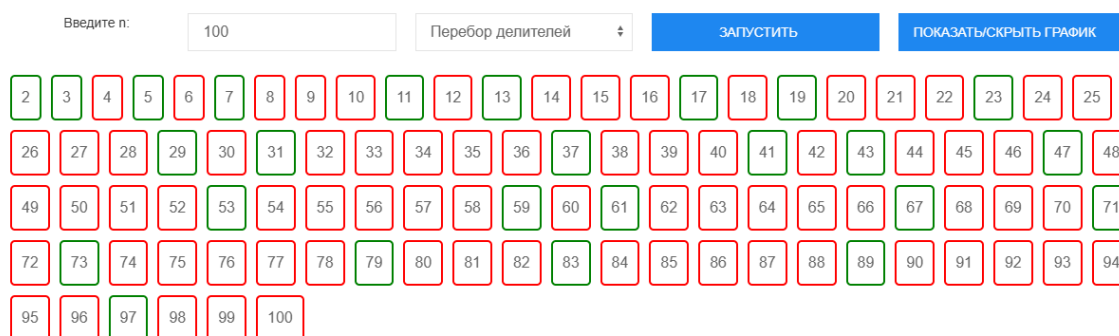


Рисунок 4. Демонстрация простых чисел до введённого числа N

В разделе с длинной арифметикой обучающиеся могут ознакомиться с основными алгоритмами над большими числами, выходящими за допустимые ограничения типов данных: сложение, вычитание, умножение и деление.

Для демонстрации основных алгоритмов написаны функции, визуализирующие принципы работы описанных на слайдах ранее. Для этого написаны функции на JavaScript, подготавливающие числа для отображения и визуализирующие пошаговое выполнение выбранных операций:

```

function createDigitRow(id, number) {
    const row = document.getElementById(id);
    row.innerHTML = "";
    for (let digit of number) {
        let div = document.createElement("div");
        div.classList.add("digit");
        div.textContent = digit;
        row.appendChild(div);
    }
}

function createPartialProductRow(container, value, offset) {
    const rowDiv = document.createElement("div");
    rowDiv.classList.add("row");
    rowDiv.style.marginLeft = (offset * 35) + "px";

    for (let digit of value) {

```

```

        const digitDiv = document.createElement("div");
        digitDiv.classList.add("digit");
        digitDiv.textContent = digit;
        rowDiv.appendChild(digitDiv);
    }

    container.appendChild(rowDiv);
    return rowDiv;
}

function prepareNumbers() {
    let maxLength = Math.max(num1.length, num2.length);
    num1 = num1.padStart(maxLength, '0');
    num2 = num2.padStart(maxLength, '0');
    createDigitRow("num1Row", num1);
    createDigitRow("num2Row", num2);
    createDigitRow("carryRow", "".padStart(maxLength, ' '));
    createDigitRow("resultRow", "".padStart(maxLength, ' '));
});
}

```

Визуализация длинной арифметики

Сложение

											1			
3	5	7	3	5	7	2	7	5	2	4	8	7	1	2
7	3	4	8	3	1	7	5	7	3	4	4	2	3	3

									5	9	2	9	4	5
--	--	--	--	--	--	--	--	--	---	---	---	---	---	---

Выполнить

Рисунок 5. Визуализация алгоритма сложения больших чисел

В разделе с рекурсивными алгоритмами демонстрируется стек вызовов, визуально демонстрирующий принцип выполнения операций в программах, в том числе и в рекурсивных алгоритмах. На слайде с описанием стека вызовов присутствует интерактивная визуализация данной структуры данных, под которой есть две кнопки: PUSH для добавления нового элемента в стек и POP для удаления последнего добавленного элемента.

В JavaScript также реализованы соответствующие функции для взаимодействия со стеком:

```
function pushToStack() {
    const value = nextNumber++;
    stackData.push(value);

    const block = document.createElement('div');
```

```

        block.className = 'stack-item new';
        block.textContent = value;

        stackContainer.appendChild(block);
    }

    function popFromStack() {
        if (stackData.length === 0) {
            alert('Стек пуст!');
            return;
        }
        stackData.pop();

        const lastBlock = stackContainer.lastElementChild;
        if (lastBlock) {
            stackContainer.removeChild(lastBlock);
        }
    }
}

```

Стек вызовов

Рекурсия характеризуется **стеком вызовов**.

Стек вызовов — это структура данных, которая хранит информацию о текущих активных вызовах функций в программе. Каждый раз, когда функция вызывается, в стек добавляется новый "фрейм" с её параметрами, локальными переменными и адресом возврата. После завершения работы функция удаляется из стека, и выполнение продолжается с места вызова.

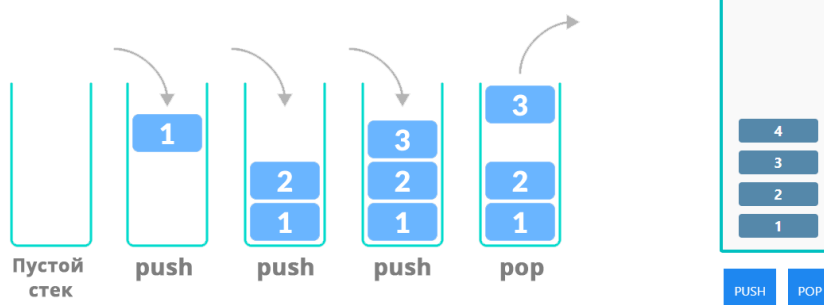


Рисунок 6. Демонстрация работы стека вызовов на сайте

Также для демонстрации алгоритма вычисления чисел Фибоначчи на сайте реализован алгоритм построения дерева вызовов рекурсивной функции, который работает через стандартное полотно HTML.

Для демонстрации сортировок выбран стандартный способ визуализации подобных алгоритмов через перемещение столбиков разной высоты относительно друг друга. На сайте реализовано три алгоритма: сортировки выбором и вставками и быстрая сортировка.

В зависимости от сортировки может отличаться цветовая схема обозначения элементов на экране. Например, в сортировке выбором розовым цветом отмечен элемент, на позицию которого будет поставлен элемент, меньший по значению, а зелёным обозначается элемент с меньшим значением. Во всех сортировках жёлтый цвет обозначает обход этого элемента и сравнение его со значением на позиции элемента с розовым цветом (см. рис. 7).

Для изменения состояния столбиков написаны функции для изменения их расположения и цветов на JavaScript:

```
async function swapBars(index1, index2) {
  return new Promise(resolve => {
    let bar1 = bars[index1].wrapper;
    let bar2 = bars[index2].wrapper;

    let distance = (index2 - index1) * 50;
    bar1.style.transform = `translateX(${distance}px)`;
    bar2.style.transform = `translateX(${distance}px)`;

    setTimeout(() => {
      bar1.style.transform = "";
      bar2.style.transform = "";

      [bars[index1], bars[index2]] = [bars[index2],
bars[index1]];
      [values[index1], values[index2]] =
[values[index2], values[index1]];
      updateBarContainer();
      resolve();
    }, 400);
  });
}

function updateBarContainer() {
  const container =
document.getElementById("barContainer");
  container.innerHTML = "";
  bars.forEach(({ wrapper }) =>
container.appendChild(wrapper));
}

function delay(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
```

Также для демонстрации скорости выполнения сортировки добавлен отдельный слайд, на котором можно загрузить файл с тестовым массивом произвольной длины и измерить скорость выполнения каждой сортировки. Алгоритмы выполняются асинхронно, что позволяет замерить выполнение каждой независимо друг от друга (нет необходимости ждать завершения предыдущей, чтобы начать следующую).

Визуализация сортировки

Сортировка выбором

ВСТАВКАМИ ВЫБОРОМ БЫСТРАЯ

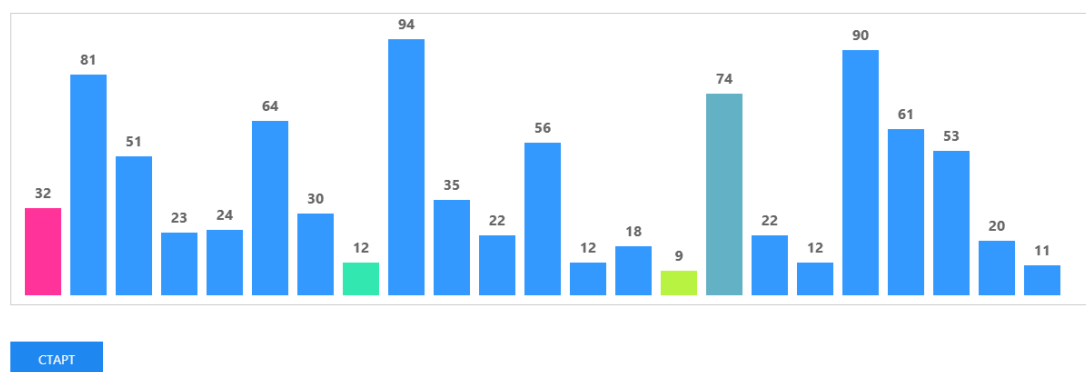


Рисунок 7. Визуализация алгоритма выбором на сайте

4 Выводы

Таким образом, разработано комплексное веб-приложение для демонстрации базовых алгоритмов, которое можно использовать в образовательном процессе независимо от уровня подготовки обучающихся. За счёт возможности визуализации преподаваемого материала уровень усвоения алгоритмов и дальнейшей их реализации может возрасти при правильном использовании.

Планируется апробирование разработки с целью сбора статистики на занятиях в рамках дисциплины «Основы алгоритмизации и программирования» в ПГУ им. Шолом-Алейхема на втором курсе обучающихся по программе СПО.

Библиографический список

1. Кондратенко Д.А. Разработка программы для визуализации алгоритмов сортировки целочисленного массива // Информационные технологии в науке и производстве: Материалы X Всероссийской молодежной научно-технической конференции, Омск, 18 апреля 2023 года / Редколлегия: А.Г. Янишевская (отв. ред.) [и др.]. Омск: Омский государственный технический университет. 2023. С. 59-62. URL: <https://elibrary.ru/item.asp?id=7827346384> (дата обращения: 09.08.2026).
2. Гюллинг А.О., Воронцова Н.В. Визуализация алгоритма Дейкстры // Известия Тульского государственного университета. Технические науки. 2024. № 2. С. 127-132. URL: <https://cyberleninka.ru/article/n/vizualizatsiya-algoritma-deykstry> (дата обращения: 09.08.2026).
3. Шелкунов П.А. Визуализация алгоритма Дейкстры // Проспект Свободный - 2024: Материалы юбилейной XX Международной научной конференции студентов, аспирантов и молодых ученых, Красноярск, 15–20 апреля 2024 года. Красноярск: Сибирский федеральный университет. 2024. С. 723-725. URL: <https://elibrary.ru/item.asp?id=3458274356238> (дата обращения: 09.08.2026).

09.08.2026).

4. Моглан Д.В. Дидактический потенциал использования систем визуализации алгоритмов в процессе обучения программированию // Открытое образование. 2019. Т. 23. № 2. С. 31-41. URL: <https://openedu.rea.ru/jour/article/view/617> (дата обращения: 09.08.2026).
5. Подставников В.Ю., Фомин Д.В., Самохвалова С.Г. Визуализация работы криптографических алгоритмов // Вестник Амурского государственного университета. Серия: Естественные и экономические науки. 2023. № 101. С. 165-171. URL: <https://cyberleninka.ru/article/n/vizualizatsiya-raboty-kriptograficheskikh-algoritmov> (дата обращения: 09.08.2026).