МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Информационная безопасность»

Исполнитель

студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2025

### Алгоритм хеширования MD5

В лабораторной работе был выбран для реализации метод хеширования MD5.

Код реализации представлен на листинге.

|  |
| --- |
| class MD5Hasher {  constructor() {  this.K = new Array(64);  for (let i = 0; i < 64; i++) {  this.K[i] = Math.floor(Math.abs(Math.sin(i + 1)) \* Math.pow(2, 32));  }  }  leftRotate(value, amount) {  return ((value << amount) | (value >>> (32 - amount))) >>> 0;  }  padMessage(bytes) {  const originalLength = bytes.length;  const originalLengthBits = originalLength \* 8;    bytes.push(0x80);    while ((bytes.length % 64) !== 56) {  bytes.push(0);  }    for (let i = 0; i < 8; i++) {  bytes.push((originalLengthBits >>> (i \* 8)) & 0xFF);  }    return bytes;  }  F(x, y, z) { return ((x & y) | (~x & z)) >>> 0; }  G(x, y, z) { return ((x & z) | (y & ~z)) >>> 0; }  H(x, y, z) { return (x ^ y ^ z) >>> 0; }  I(x, y, z) { return (y ^ (x | ~z)) >>> 0; }  md5Operation(func, a, b, c, d, x, s, t) {  return (b + this.leftRotate((a + func(b, c, d) + x + t) >>> 0, s)) >>> 0;  }  hash(message) {  const startTime = performance.now();    let bytes = this.stringToBytes(message);  bytes = this.padMessage(bytes);    let h0 = 0x67452301;  let h1 = 0xEFCDAB89;  let h2 = 0x98BADCFE;  let h3 = 0x10325476;    // Обработка блоков по 512 бит (64 байта)  for (let offset = 0; offset < bytes.length; offset += 64) {  // Разбиваем блок на 16 32-битных слов (little-endian)  const w = new Array(16);  for (let i = 0; i < 16; i++) {  w[i] = bytes[offset + i \* 4] |  (bytes[offset + i \* 4 + 1] << 8) |  (bytes[offset + i \* 4 + 2] << 16) |  (bytes[offset + i \* 4 + 3] << 24);  }    let a = h0, b = h1, c = h2, d = h3;      // Раунд 1  for (let i = 0; i < 16; i++) {  const temp = d;  d = c;  c = b;  b = this.md5Operation(this.F, a, b, c, d, w[i], this.S[i], this.K[i]);  a = temp;  }    // Раунд 2  for (let i = 16; i < 32; i++) {  const temp = d;  d = c;  c = b;  const index = (5 \* (i - 16) + 1) % 16;  b = this.md5Operation(this.G, a, b, c, d, w[index], this.S[i], this.K[i]);  a = temp;  }    // Раунд 3  for (let i = 32; i < 48; i++) {  const temp = d;  d = c;  c = b;  const index = (3 \* (i - 32) + 5) % 16;  b = this.md5Operation(this.H, a, b, c, d, w[index], this.S[i], this.K[i]);  a = temp;  }    // Раунд 4  for (let i = 48; i < 64; i++) {  const temp = d;  d = c;  c = b;  const index = (7 \* (i - 48)) % 16;  b = this.md5Operation(this.I, a, b, c, d, w[index], this.S[i], this.K[i]);  a = temp;  }    h0 = (h0 + a) >>> 0;  h1 = (h1 + b) >>> 0;  h2 = (h2 + c) >>> 0;  h3 = (h3 + d) >>> 0;  }    const endTime = performance.now();  const executionTime = endTime - startTime;  const result = this.toHexString(h0) + this.toHexString(h1) +  this.toHexString(h2) + this.toHexString(h3);  return {  hash: result,  executionTime: executionTime,  messageLength: message.length,  bytesProcessed: bytes.length  };  }  } |

На рисунке 1.1 представлен пример работы хэширования.

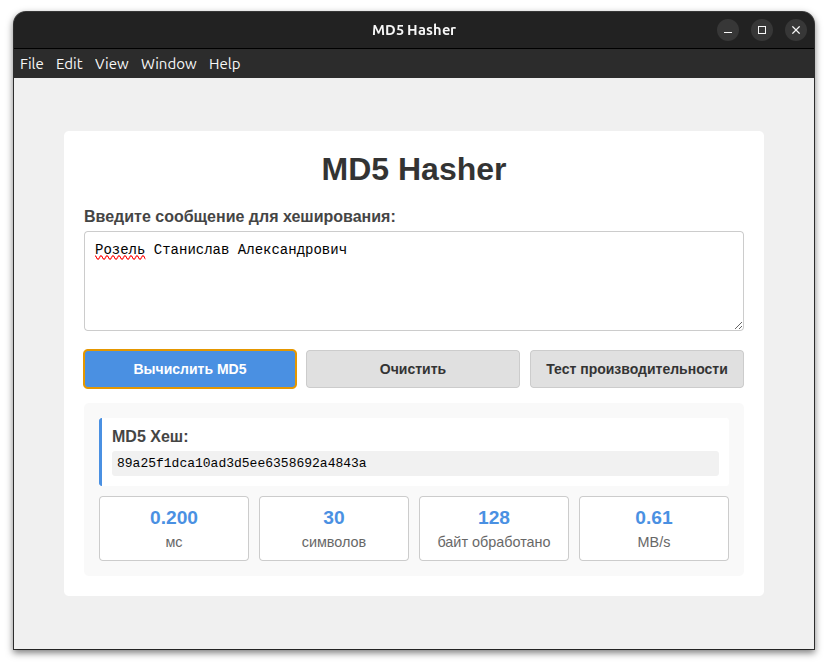


Рисунок 1.1 - Работа алгоритма

Скорость хэширования оказалась 0.61 MB/s/. Так же на рисунке 1.2 продемонстрована работа теста производительности.

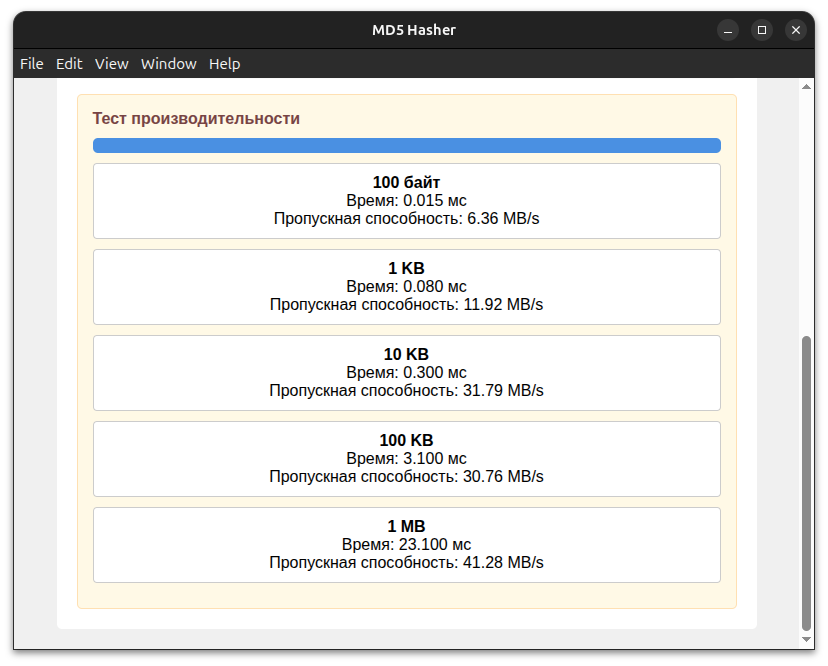


Рисунок 1.2 - Тест производительности