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

### Приложения 1

Приложение реализует алгоритм генерации ПСП BBS с n = 512 бит, p и q должны быть два простых криптостойких числа, а так же соответствовали условию p = q = 3 (mod 4). На листинге представлен код реализации алгоритма.

|  |
| --- |
| class BBSGenerator {  constructor() {  this.p = bigInt('11');  this.q = bigInt('19');  this.n = this.p.multiply(this.q);  if (!this.p.mod(4).equals(3) || !this.q.mod(4).equals(3)) {  throw new Error('P и Q должны быть равны 3 по модулю 4');  }  this.startTime = performance.now();  this.reset();  }    reset(seed = null) {  if (seed === null) {  seed = Date.now().toString();  }  const seedNum = bigInt(Buffer.from(seed.toString()).reduce((acc, val) => acc + val, 0));    this.x = seedNum.mod(this.n);  if (bigInt.gcd(this.x, this.n).notEquals(1)) {  this.x = this.x.add(1);  }    this.sequence = [];  }    nextBit() {  this.x = this.x.pow(2).mod(this.n);  return this.x.isOdd() ? 1 : 0;  }  generateBits(length) {  const bits = [];  const startTime = performance.now();  for (let i = 0; i < length; i++) {  bits.push(this.nextBit());  }  const endTime = performance.now();  this.generationTimes.push({  count: length,  time: endTime - startTime,  bytesPerSecond: length / ((endTime - startTime) / 1000)  });  return bits;  }  generateBytes(length) {  const bytes = new Uint8Array(length);  for (let i = 0; i < length; i++) {  let byte = 0;  for (let j = 0; j < 8; j++) {  byte = (byte << 1) | this.nextBit();  }  bytes[i] = byte;  }  return bytes;  }  generateNumbers(count, max) {  const numbers = [];  for (let i = 0; i < count; i++) {  const bitsNeeded = Math.ceil(Math.log2(max));  let num = 0;  for (let j = 0; j < bitsNeeded; j++) {  num = (num << 1) | this.nextBit();  }  numbers.push(num % max);  }  return numbers;  }  generateSequence(n) {  this.sequence = this.generateBits(n);  return this.sequence;  }  getSequence() {  return this.sequence;  }  analyzeSequence() {  if (this.sequence.length === 0) {  return { zeros: 0, ones: 0, ratio: 0 };  }  const zeros = this.sequence.filter(bit => bit === 0).length;  const ones = this.sequence.filter(bit => bit === 1).length;  return {  zeros,  ones,  ratio: ones / this.sequence.length  };  }  } |

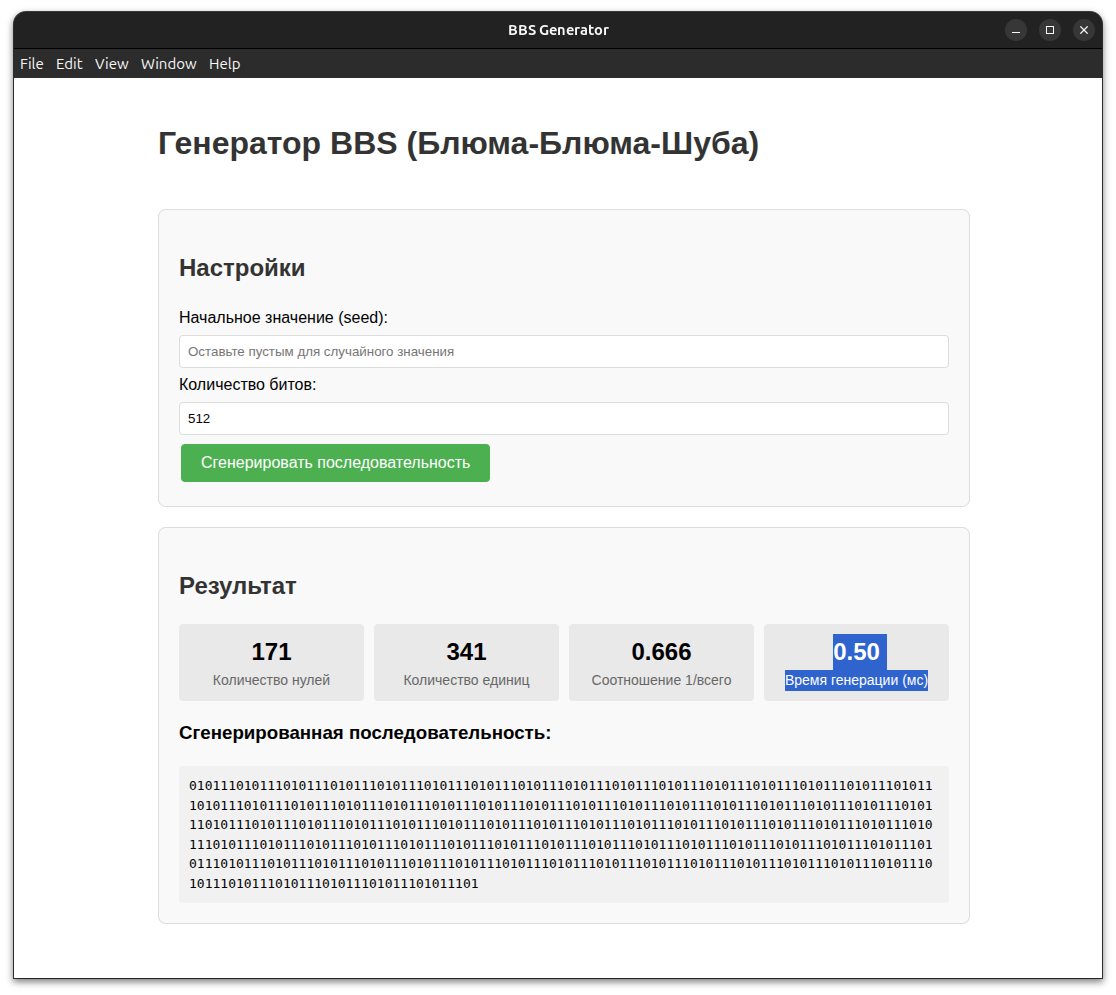
Пример работы приложения 1 представлен на рисунке 1.1.  


Рисунок 1.1 – Пример работы приложения.

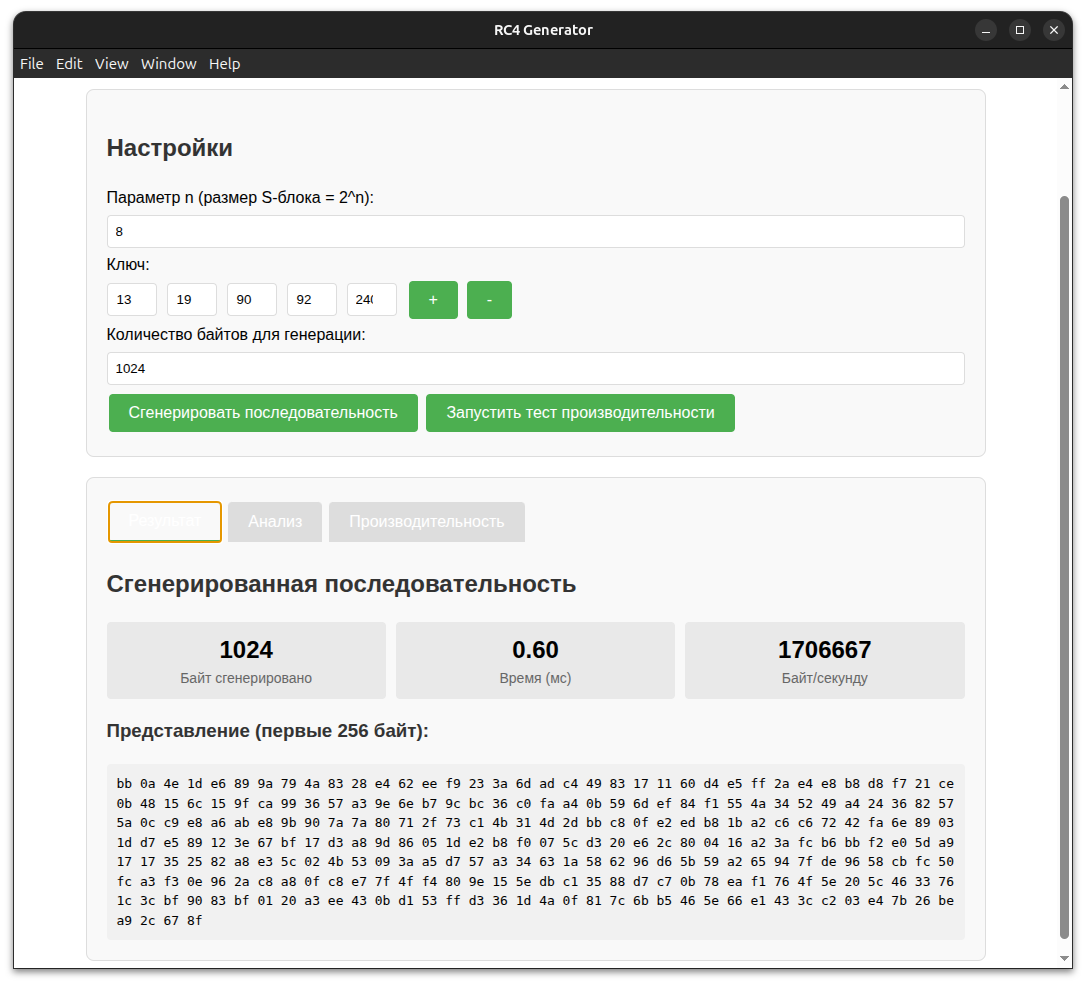
### Приложение 2

Приложение 2 реализует алгоритм RC4 c n = 8 и ключ 13, 19, 90, 92, 240. На листинге 2.1 представлен код реализации алгоритма RC4.

|  |
| --- |
| class RC4Generator {  constructor(n = 8, key = [13, 19, 90, 92, 240]) {  this.n = n;  this.key = key;  this.S = new Array(1 << n);  this.initialize();  }  initialize() {  for (let i = 0; i < (1 << this.n); i++) {  this.S[i] = i;  }    let j = 0;  for (let i = 0; i < (1 << this.n); i++) {  j = (j + this.S[i] + this.key[i % this.key.length]) % (1 << this.n);  [this.S[i], this.S[j]] = [this.S[j], this.S[i]];  }  this.i = 0;  this.j = 0;    this.startTime = performance.now();  this.bytesGenerated = 0;  this.generationTimes = [];  }    nextByte() {  this.i = (this.i + 1) % (1 << this.n);  this.j = (this.j + this.S[this.i]) % (1 << this.n);    [this.S[this.i], this.S[this.j]] = [this.S[this.j], this.S[this.i]];    const t = (this.S[this.i] + this.S[this.j]) % (1 << this.n);  this.bytesGenerated++;  return this.S[t];  }  generateBytes(length) {  const bytes = new Uint8Array(length);  const startTime = performance.now();  for (let i = 0; i < length; i++) {  bytes[i] = this.nextByte();  }  const endTime = performance.now();  this.generationTimes.push({  count: length,  time: endTime - startTime,  bytesPerSecond: length / ((endTime - startTime) / 1000)  });  return bytes;  }  generateBits(length) {  const byteCount = Math.ceil(length / 8);  const bytes = this.generateBytes(byteCount);  const bits = [];  for (let i = 0; i < length; i++) {  const byteIndex = Math.floor(i / 8);  const bitIndex = i % 8;  bits.push((bytes[byteIndex] >> (7 - bitIndex)) & 1);  }  return bits;  }  return { entropy: entropy, uniqueBytes: nonZeroCount, distribution: distribution  };  }  } |

Листинг 2.1 – Реализация приложения 2

Пример работы приложения 2:



Так же по заданию нужно было измерить скорость генерации строки, которая исходя из рисунка 2.1 равна 1024 байта сгенерировано за 0.40 время (мс)