Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский ядерный университет «МИФИ»
Предуниверситарий НИЯУ МИФИ. Университетский лицей №1523.

Кафедра физики.

ПРОЕКТНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

ТЕМА: Исследование методов генерации случайных и псевдослучайных чисел и их использование

Автор проекта: учащийся 10М класса лицея 1523, Клубничкин И.К.
Руководитель: Мареева Е.С.

Москва, 2022

Содержание

Введение…….2-3

Глава I. (Исследование проблемы генерации случайных чисел. Введение понятия ГСЧ и ГПСЧ.).....4-5

Глава II. (Обзор методов генерации случайных и псевдослучайных чисел.).....6-7

Глава III. (Создание экспериментальной установки по генерации случайных чисел.).....8

Глава IV. (Методы определения меры случайности числовых последовательностей.).....9-10

Заключение……..……11

Используемая литература……...12

Приложения……13-17

Введение

Актуальность. В наши дни случайные и псевдослучайные числа широко применяются в криптографии и многих других областях научного знания (случайные числа имеют применение в физике, анализе, программировании, моделировании и т. д.).

Объекты исследования:

1. Генераторы случайных чисел **(**далее - **ГСЧ**) и принципы их работы.
2. Генераторы псевдослучайных чисел (далее - **ГПСЧ**) и принципы их работы.
3. Методы проверки случайности числовой последовательности.

Предметы исследования:

1. ГСЧ, основанные на видеорегистрации хаотичных физических явлений.
2. Наиболее распространенные методы генерации случайных чисел.
3. Пакет статистических тестов NIST (см. ссылку 1).

Практическая значимость проекта. В процессе реализации проекта был создан работающий ГСЧ. Были рассмотрены варианты повышения производительности ГСЧ с помощью ГПСЧ.

Цель. Исследование разных методов генерации случайных и псевдослучайных чисел, воссоздание ГСЧ и его оптимизация.

Задачи:

1. Изучение основных методов генерации случайных и псевдослучайных чисел.
2. Изучение областей применения ГСЧ и ГПСЧ.
3. Создание экспериментальной установки по генерации случайных чисел.
4. Анализ выходных последовательностей, выдаваемых генератором при различных параметрах.

Гипотеза исследования. Допустим, что можно создать ГСЧ на основе некоторого случайного физического явления.

Методы исследовательской работы:

1. Изучение методов генерации случайных и псевдослучайных чисел.

2. Исследование методов определения меры близости заданной последовательности к случайной.
3. Создание экспериментальной установки по генерации случайных чисел и его оптимизация.
4. Проверка случайности последовательности, выдаваемой экспериментальным ГСЧ.

Новизна исследования. На сегодняшний день существуют работы, посвященные изучению отдельных методов генерации случайных чисел. Однако при создание данной работы я подошел к вопросу более комплексно, изучив методы генерации случайных и псевдослучайных чисел, области их применения, создал экспериментальную установку, работоспособность которой была подтверждена тестами, а также изучил методы проверки случайности числовой последовательности с помощью статистических тестов NIST.

Этапы исследования:

1. Изучение методов генерации случайных и псевдослучайных чисел.
2. Исследование методов определения меры близости заданной последовательности к случайной.
3. Создание экспериментальной установки по генерации случайных чисел и его оптимизация.
4. Проверка случайности последовательности, выдаваемой экспериментальным ГСЧ.

Основная часть

Глава 1. Исследование проблемы генерации случайных чисел. Введение понятия ГСЧ и ГПСЧ.

Случайные числа играют важную роль в самых разных областях науки. Их активно применяют в задачах моделирования, численного анализа, теории игр. Генераторы случайных чисел (далее - **ГСЧ**) в качестве механизма получения случайных величин используют разные физические процессы, так как на данный момент не существует математических алгоритмов, которые бы занимались генерацией подлинно случайной и непредсказуемой последовательности. Данная тема крайне обширна и стоит также отметить, что ГСЧ являются ключевым звеном в современной криптографии и информационной безопасности. Таким образом, данная работа посвящена исследованию актуальной области, затрагивающей физику, математику и программирование.

Проблема генерации подлинно случайных чисел заключается в том, что если “случайная” последовательность будет создаваться некоторым математическим алгоритмом, то она будет предсказуема, что, к примеру, в задачах информационной безопасности может скомпрометировать безопасность личных данных. Подобные алгоритмы называют генераторами псевдослучайных чисел (далее - **ГПСЧ**) и их также используют в науке, в задачах, при решении которых не нужна истинная случайность (задачи моделирования, имитация хаотичных действий пользователя ит.д.), а также для повышения производительности ГСЧ. В качестве примера можно рассмотреть один из первых ГПСЧ, который был разработан американским математиком Джоном фон Нейманом, механизм которого называется “методом середины квадрата”. Вот описание данного метода: “*Десятизначное число возводится в квадрат, затем из середины квадрата числа берётся десятизначное число, которое снова возводится в квадрат, и так далее”*(описание данного метода было взято со страницы онлайн энциклопедии “Википедия”, посвященной ГПСЧ (см. ссылку 2)). Очевидным недостатком данного ГПСЧ является ограниченность выдаваемого множества, т.е. если в качестве входного числа будет передано, например, число 1 000 000 000, то ГПСЧ зациклится и начнет выдавать лишь 0 (т.к. серединой $1\,000\,000\,000^2$ является 0, а серединой числа 0^2 является ноль ит.д.). Также рано или поздно середина некоторого числа совпадет с начальным числом, которое вводится человеком и все выходные данные начнут повторяться вновь. Проблемы зацикленности и предсказуемости являются основными проблемами всех ГПСЧ. Зная алгоритм, используемый для создания чисел, и его внутреннее состояние, можно предсказать все числа, возвращаемые при последующих итерациях, тогда как с действительно случайными числами знание одного числа или произвольно длинной последовательности чисел бесполезно.

ГСЧ формируют последовательность случайных чисел в зависимости от текущего значения какого-либо атрибута физической среды, который практически невозможно смоделировать при текущем уровне знаний. В рамках данного проекта я изучил эту область, наиболее распространенные методы генерации случайных чисел, а также методы проверки случайности некоторой последовательности.

Ниже приведена таблица, в которой сравниваются преимущества и недостатки ГСЧ и ГПСЧ (см. табл. 1)

Также была создана таблица, в которой указывается наиболее подходящий генератор для тех или иных ситуаций (см. табл. 2).

Глава 2. Обзор методов генерации случайных и псевдослучайных чисел.

В данной главе будут описаны методы генерации случайных и псевдослучайных чисел, исследованные в рамках данного проекта.

2.1. Тепловые колебания резистора. Данный метод основывается на тепловом шуме резистора, который возникает из-за хаотического теплового движения носителей заряда в проводнике, в результате чего, при измерении напряжения на резисторе, оно будет постоянно флуктуировать. При измерении колебаний напряжения с помощью чувствительного АЦП (аналого-цифрового преобразователя) и усиления сигнала, можно получать случайные числа. Данный метод генерации случайных чисел реализован на логических элементах чипов в современных процессорах Intel. Данный метод генерации случайных чисел является одним из наиболее распространенных, в силу своей высокой производительности и физической компактности финального ГСЧ.

2.2. ГСЧ, основанный на атмосферном шуме. Один из крупнейших сервисов, занимающийся генерацией истино случайных чисел - random.org использует метод, основанный на регистрации атмосферных шумов. В распоряжение сервиса находятся несколько радиоприемников, настроенных на прием определенных частот. Шумы, которые улавливают данные приемники в основном возникают из-за молний, которые ежесекундно ударяют в самых разных точках планеты, создавая радиошумы, которые при смешивании создают некоторый фон. А так как предугадать место и время, в котором ударит следующая молния - невозможно, то изменение этого атмосферного шума является непредсказуемым (данная информация была взята с официального сайта random.org (см. ссылку 3)).

2.3. Квантовые флуктуации вакуума. Хотелось бы также упомянуть метод генерации случайных чисел, основанный на квантовых флуктуациях вакуума. Дело в том, что вакуум в сущности не является пустым. В нем, в силу принципа неопределенности постоянно возникают на короткий промежуток времени пары виртуальных частиц - т.е. возникают непредсказуемые колебания уровня энергии в пространстве.

Команда канадских физиков разработала ГСЧ (см.ссылку 4), основываясь на вышеупомянутом факте. Если определенным образов пустить лазерный луч, то он будет взаимодействовать с виртуальными частицами, которые встречаются у него на пути. А так как возникновение виртуальных частиц является стохастическим, то и поведение луча будет случайным, а регистрация поведения лазера позволяет преобразовать получаемую информацию в случайные числа.

2.4. ГСЧ, основанный на регистрации ионизирующего излучения. HotBits (см. ссылку 5) - еще один сайт, занимающийся генерацией случайных чисел для разных нужд. Случайные числа для этого сайта генерируются с помощью счетчика Гейгера, регистрирующего непредсказуемое ионизирующее излучение, которое вызывается радиоактивным распадом. Минусом данного метода является низкая скорость генерации чисел, в случае с данным сайтом она равна 100 байтам в секунду.

Несмотря на то, что некоторые из вышеописанных методов имеют низкую скорость производства случайных чисел, ее можно увеличить, используя ГПСЧ. Для более ясного объяснения, рассмотрим распространенный метод генерации случайных чисел - линейный конгруэнтный метод (см. ссылку 6). Суть метода сводится к следующей рекуррентной формуле:

$$X_{n+1} = (aX_n + c) mod\ m,$$

где m, a и c - натуральные числа ($m \geq 2$, $0 \leq a < m$, $0 \leq c < m$). Как было сказано выше, данный ГПСЧ, как и все другие рано или поздно зациклится. Например, для m = 10, $X_0 = a = c = 7$ получим последовательность: 7,6,9,0,7,6,9,0…, которая зацикливается (повторяется последовательность 7, 6, 9, 0).

Однако, ограничивая число итераций для данного алгоритма с постоянными коэффициентами (до зацикливания последовательности) и задавая X_0 случайно (взяв значение, полученное с помощью ГСЧ), можно приумножить количество получаемых истинно случайных чисел.

Глава 3. Создание экспериментальной установки по генерации случайных чисел.

В рамках работы над данным проектом был создан ГСЧ, основанный на видеорегистрации движения парафина в лавовой лампе. ГСЧ, основанный на том же физическом механизме, используется в компании CloudFlare.

За движением жидкостей в лавовой лампе наблюдает камера (для демонстрационной установки было использовано записанное видео, которое впоследствии будет заменено на видео с камеры в реальном времени). Для полученного видео устанавливается цветовой режим “оттенки серого”, таким образом, каждый пиксель имеющегося видео можно описать числом от 0 до 255 (где 0 - черный цвет, а 255 - белый (см. рис. 1)), используя серую шкалу, в которой на каждый пиксель изображения приходится 1 байт (8 бит) информации. Разбив видео на кадры, мы можем представить каждый кадр как матрицу, каждый элемент которой характеризует определенный пиксель. По первоначальной задумке планировалось брать числа, полученные при разложении этой матрицы в строку, однако данный вариант имел множество изъянов. Во-первых, возникла бы необходимость брать кадры с некоторой разницей во времени (3-5 секунд), что очень сильно бы уменьшило производительность ГСЧ. А во-вторых, на двух различных кадрах скорее всего были бы идентичные пиксели, так как в кадре оставались неизменные области (края лампы, в которых не плавают капли парафина и т. д.). Поэтому было принято решение пропускать каждое число, характеризующее кадр, через алгоритм хэширования, так как при даже незначительном изменение значения аргумента хэш-функции, значение этой самой функции будет сильно меняться.

Число, получаемое разложением данной матрицы в строку, используется как аргумент для хэш-функции md5, которая возвращает 128-битное число в шестнадцатеричной системе счисления. Так как оно зависит от изменения изображения в кадре и, как следствие, от движения жидкостей в лавовой лампе, а так как современная физика не в состояние полностью описать и предсказать поведение вязкой жидкости в среде, то полученные числа являются случайными.

Для программной реализации данного ГСЧ была написана программа на языке Python с использованием библиотек OpenCV (для работы с изображением) и hashlib (для хеширования входных данных). Ниже представлен полный код программы (см. рис. 2).

После импорта библиотек и загрузки видео программа рассматривает каждый кадр как матрицу, которая представлена как система вложенных массивов. После этого программа путем двойного вхождения в систему массивов извлекает все их элементы, перемещая их в строку, которая будет являться аргументом для хэш-функции. После получения данного числа программа хэширует его используя алгоритм хэширования md5, значение которого после этого переносится в текстовый файл text.txt.

Учитывая, что программа преобразует каждый кадр (видео транслируется с частотой 25 к/с) в число, а алгоритм хэширования md5 возвращает 16 байтное (128 битное) число, несложно посчитать, что производительность данного ГСЧ равняется 400 байтам в секунду (25 к/с * 16 байт = 400).

Используя аналогичный принцип могут быть также созданы ГСЧ, без изменения аппаратного и программного обеспечения, основанные на хаотичных движениях двойного маятника, беспорядочных движениях микроскопических видимых взвешенных частиц твёрдого вещества в жидкости или газе из-за броуновского движения и прочих стохастических макропроцессах.

Глава 4. Методы определения меры случайности числовых последовательностей.

При работе с различными методами генерации случайных чисел возникает вопрос, насколько надежен тот или иной ГСЧ? Для ответа на данный вопрос существует большое количество различных тестов, результаты которых позволяют сказать, является ли последовательность, выдаваемая данным ГСЧ, случайной и насколько она непредсказуема.

Наиболее распространенным методом определения меры случайности двоичных последовательностей является пакет статистических тестов NIST, разработанный Лабораторией информационных технологий. Данный пакет включает в себя 15 различных тестов, которые проверяют разные параметры последовательности, для определения ее случайности. Вот список всех тестов (информация о данных тестах была взята со страницы онлайн энциклопедии “Википедия”, посвященной статистическим тестам NIST (см. ссылку 1)):

1. Частотный побитовый тест.
2. Частотный блочный тест
3. Тест на последовательность одинаковых битов
4. Тест на самую длинную последовательность единиц в блоке
5. Тест рангов бинарных матриц
6. Спектральный тест
7. Тест на совпадение неперекрывающихся шаблонов
8. Тест на совпадение перекрывающихся шаблонов
9. Универсальный статистический тест Маурера
10. Тест на линейную сложность
11. Тест на периодичность
12. Тест приблизительной энтропии
13. Тест кумулятивных сумм
14. Тест на произвольные отклонения
15. Другой тест на произвольные отклонения

В результате прохождения каждого теста, программа возвращает значение коэффициента p. Если значение данного параметра больше 0,01, то тест считается пройденным. На основе всех 15 коэффициентов формируется окончательное значение коэффициента и если $p > 0,01$, то последовательность может считаться случайной.

Вот краткое описание некоторых из указанных выше тестов:

Частотный побитовый тест. Суть данного теста заключается в поиске отношения количества 0 и 1 в двоичной последовательности. Чем ближе данная последовательность к случайной, тем ближе это отношение будет к 1 (или к 0,5 если оценивается отношение 0 или единиц к количеству знаков в последовательности). Данный тест является первым в пакете статистических тестов NIST и все последующие тесты проводятся при условии, что данный тест был пройден.

Тест на последовательность одинаковых битов. Цель данного теста - определить количество последовательностей одинаковых битов в последовательности и сделать вывод о том, действительно ли количество рядов, состоящих из единиц и нулей с различными длинами, соответствует их количеству в случайной последовательности.

Тест на самую длинную последовательность единиц в блоке. Данный тест в чем-то схож с предыдущим (тест на последовательность одинаковых битов). Данный тест заключается в определение длины самого длинного ряда последовательно идущих единиц и выяснить, соответствует ли длина данного ряда ожиданиям длины ряда единиц в абсолютно случайной последовательности.

Спектральный тест. Суть теста заключается в поиске периодических свойств входной последовательности (см. рис. 3) с помощью дискретного преобразования Фурье исходной последовательности.

На заключительном этапе была проверена последовательность, полученная с помощью созданного ГСЧ (см. “Глава 3.”), с помощью тестов NIST. Для данной проверки было установлено соответствующее программное обеспечение с официального сайта NIST (см. ссылку 7), а также выходная последовательность созданного ГСЧ была переведена в двоичную систему счисления с отступами, необходимыми для корректной работы программного обеспечения.

После этого был проведен тест с указанными ниже параметрами (см. рис. 4). После прохождения теста, были получены данные указывающие на то, что последовательность, выдаваемая данным ГСЧ является случайной, так как для всех тестов значение p было выше 0,01 (см. рис. 5 и 6).

Заключение

Все вышесказанное позволяет констатировать, что на основе некого стохастического физического явления можно создать ГСЧ и подтвердить его работоспособность. Следовательно, поставленная гипотеза подтвердилась.

В рамках данного проекта было произведено погружение в большую область, затрагивающую самые разные аспекты современного научного знания. Как мне кажется, при работе над проектом были выполнены все изначально поставленные задачи:

1. Были изучены основные методы генерации случайных и чисел, а также области их применения. Были рассмотрены конкретные методики для создания ГСЧ.
2. В процессе реализации проекта был создан и оптимизирован ГСЧ, основанный на интересном физическом механизме, работоспособность которого подтвердили соответствующие тесты.
3. Были изучены методы проверки случайности некоторой числовой последовательности. Был изучен инструмент, позволяющий сравнить несколько ГСЧ между собой.

Перспективы дальнейшего исследования я вижу в более подробном изучении данной проблемы, поиске новых методов генерации случайных и псевдослучайных чисел и их оптимизации.

На мой взгляд было бы интересно попробовать себя в создание программного обеспечения с использованием генерации случайных и псевдослучайных чисел, таких как:

1. Приложение по шифрованию информации с использованием случайных чисел.
2. Сервис, занимающийся генерацией истинно случайных чисел на основе различных методов.

В процессе написания работы были получены новые знания в самых разных областях: физике, математике, программировании и т.д. Был получен опыт в работе со специальными библиотеками в языке Python (OpenCV и hashlib) при создании экспериментального ГСЧ.

Используемая литература

1. Статистические тесты NIST: https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%8B_NIST
2. Генератор псевдослучайных чисел: https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB
3. https://www.random.org/history/
4. https://opg.optica.org/oe/fulltext.cfm?uri=oe-19-25-25173&id=224867
5. https://www.fourmilab.ch/hotbits/
6. Линейный конгруэнтный метод: https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BD%D0%B3%D1%80%D1%83%D1%8D%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4
7. https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software

Приложения

Таблицы

Характеристика	ГСЧ	ГПСЧ
Производительность	Низкая	Высокая
Детерминизм	Стохастический	Детерминированный
Стоимость (отношение затраченных денег и вычислительных мощностей, необходимых для реализации генератора чисел к его производительности)	Дорогой	Дешевый

табл. 1

Область применения	Наиболее подходящий генератор
Задачи информационной безопасности (генерация ключей и т.д.)	ГСЧ
Моделирование	ГПСЧ
Лотереи, розыгрыши и т.д.	ГСЧ
Дизеринг	ГПСЧ

табл. 2

Рисунки

рис. 1

```python
import cv2
import hashlib
capture = cv2.VideoCapture('video.mp4')
f = open('text.txt', 'a', encoding='utf-8')
while True:
    a = ''
    ret, img = capture.read()
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    for i in range(len(img)):
        for j in range(len(img[i])):
            a += hex(img[i][j])
    f.write((hashlib.md5(a.strip().encode('utf-8'))).hexdigest() + '\n')
```

рис. 2

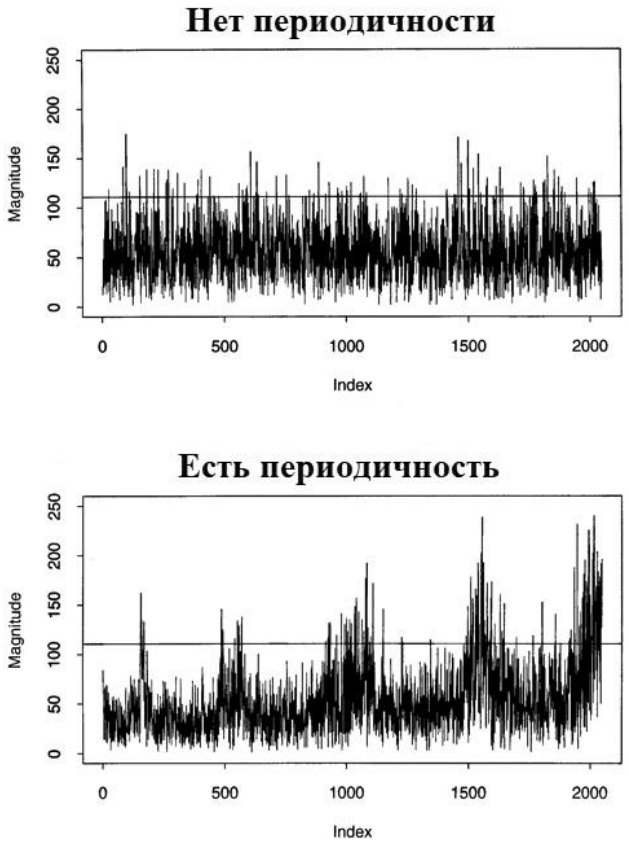

рис. 3

(фото было взято с сайта ru.wikipedia.org (см. ссылку 1))

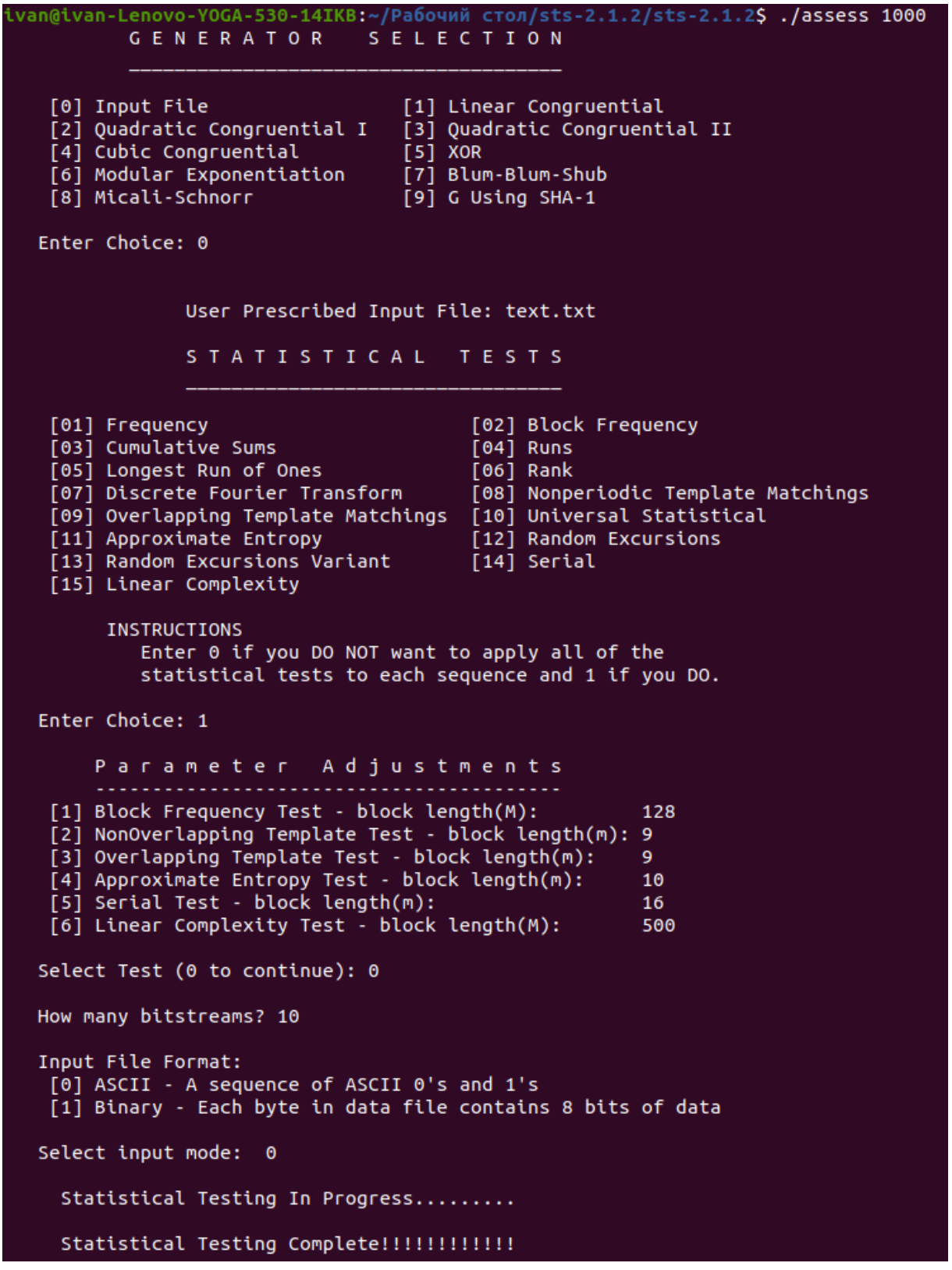
ivan@ivan-Lenovo-YOGA-530-14IKB:~/Рабочий стол/sts-2.1.2/sts-2.1.2$ ./assess 1000
G E N E R A T O R S E L E C T I O N
[0] Input File
[1] Linear Congruential
[2] Quadratic Congruential I
[3] Quadratic Congruential II
[4] Cubic Congruential
[5] XOR
[6] Modular Exponentiation
[7] Blum-Blum-Shub
[8] Micali-Schnorr
[9] G Using SHA-1
Enter Choice: 0
User Prescribed Input File: text.txt
S T A T I S T I C A L T E S T S
[01] Frequency
[02] Block Frequency
[03] Cumulative Sums
[04] Runs
[05] Longest Run of Ones
[06] Rank
[07] Discrete Fourier Transform
[08] Nonperiodic Template Matchings
[09] Overlapping Template Matchings
[10] Universal Statistical
[11] Approximate Entropy
[12] Random Excursions
[13] Random Excursions Variant
[14] Serial
[15] Linear Complexity
INSTRUCTIONS
Enter 0 if you DO NOT want to apply all of the
statistical tests to each sequence and 1 if you DO.
Enter Choice: 1
P a r a m e t e r A d j u s t m e n t s
[1] Block Frequency Test - block length(M): 128
[2] NonOverlapping Template Test - block length(m): 9
[3] Overlapping Template Test - block length(m): 9
[4] Approximate Entropy Test - block length(m): 10
[5] Serial Test - block length(m): 16
[6] Linear Complexity Test - block length(M): 500
Select Test (0 to continue): 0
How many bitstreams? 10
Input File Format:
[0] ASCII - A sequence of ASCII 0's and 1's
[1] Binary - Each byte in data file contains 8 bits of data
Select input mode: 0
Statistical Testing In Progress.....
Statistical Testing Complete!!!!!!!!!!!

рис. 4

```
                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.469000
(b) V_n_obs (Total # of runs) = 516
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 0.804589
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.255179

                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.516000
(b) V_n_obs (Total # of runs) = 509
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 0.425826
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.547035

                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.519000
(b) V_n_obs (Total # of runs) = 493
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 0.281167
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.690903
```

рис. 5

```
                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.532000
(b) V_n_obs (Total # of runs) = 497
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 0.042750
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.951791

                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.499000
(b) V_n_obs (Total # of runs) = 477
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 1.028506
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.145800

                    RUNS TEST
-----------------------------------------
COMPUTATIONAL INFORMATION:
-----------------------------------------
(a) Pi                          = 0.524000
(b) V_n_obs (Total # of runs) = 514
(c) V_n_obs - 2 n pi (1-pi)
    ----------------------    = 0.679183
      2 sqrt(2n) pi (1-pi)
-----------------------------------------
SUCCESS         p_value = 0.336799
```

рис. 6