**Проектная работа**

**на тему:** Алгоритм шифрования на основе элементарных клеточных автоматов

**Автор работы:**

**Обучающийся 8 класса**

**ГБОУ Гимназия №1538**

Чистобаев Андрей Андреевич

**Руководитель работы:**

**Консультант**

Носкова Людмила Николаевна,

Преподаватель математики, высшая категория

**Москва**

**2017 год**

# Содержание

## Постановка проблемы…………………………………………………………3

## Критерии оценки……………………………………………………………….4

## Концепция……………………………………………………………………….5

## Ресурсы…………………………………………………………………………..7

## План выполнения………………………………………………………………8

## Реализация………………………………………………………………………9

## Оценка итогов работы…………………………………………………….….11

## Литература……………………………………………………………………..12

## Приложение. О работе элементарных клеточных автоматов……..……13

# Постановка проблемы

Проблема сохранения конфиденциальности данных пользователей очень важна в современном мире. Данные должны быть защищены всеми возможными способами. От нас этого требует не только мораль, но и закон. И как вы хотите обеспечить их безопасность, пользуясь алгоритмами шифрования, изобретёнными в прошлом веке? Очевидно, что такое невозможно, так как эти шифры имеют по несколько способов взлома. И доступ к самому дорогому, что у вас есть, будет открыт любому желающему. Такое отношение к пользователям безнравственно и незаконно.

Я сам с этим столкнулся. У меня в разработке имеется небольшой самописный коллективный блог. Так как на [сайте](http://www.itgnews.ga/main.php) присутствует форма для регистрации пользователей, из которой я получаю данные о человеке, в свободном доступе должен находиться файл, в котором поясняется причина сбора информации (политика конфиденциальности). И при составлении документа мною был замечен такой пункт, как “защита данных” (защита, которой у меня, конечно же, не было). Эта проблема требовала ~~хлеба и зрелищ~~ скорейшего своего решения. И вот я погрузился в поиск способа этого решения…

# Критерии оценки

Как нам понять, что работа выполнена успешно? Что для нас значит “успех”? Нужно дать определение.

Я считаю, что моя работа будет считаться выполненной, если в её ходе будет создан эффективная и надёжная реализация метода шифрования. Это значит, что алгоритм должен иметь небольшое количество команд и входных данных, быть понятным и легко выполнимым, а так же не иметь известных путей взлома.

Выше было написано, что алгоритм должен оперировать небольшим количеством входных данных. Значит, их стоит ограничить до минимума. Но в любом случае, нам понадобятся данные, которые надо зашифровать. В этом и есть весь смысл нашей работы. Другие данные можно сделать константными или случайными.

Последний пункт нашего определения “успешности” самый сложный для выполнения, так как проверить взломостойкость шифра достаточно сложно, потому что каждый человек имеет свой объём знаний и то, что кто-то расшифровать не сможет, другой расшифрует с лёгкостью. Поэтому мнение о сложности шифра крайне субъективно, и оценить безопасность информации можно будет только с помощью профессионалов, но я попытаюсь их максимально запутать!

# Концепция

И вот мы определились с требованиями к нашему алгоритму. Теперь можно перейти от абстракции к чему-то более конкретному. Для этого ответим на некоторые вопросы:

*Кто будет являться исполнителем алгоритма?*

Конечно же, исполнителем будет компьютер. То есть метод шифрования должен быть записан на языке программирования, на языке, который понятен компьютеру.

*На чём будет основан алгоритм?*

Интрига… На элементарных клеточных автоматах! Да, да, те самые! О работе клеточных автоматов можно писать книги, поэтому подробнее почитайте о них [здесь](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B5%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82), [здесь](https://habrahabr.ru/post/168291/), [здесь](https://habrahabr.ru/post/273393/) и [здесь](http://www.vlad-utenkov.narod.ru/personal2/informat/km/1_ka/1_ka.htm). А в работе я лишь отмечу, что изобретены они были [Джоном фон Нейманом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%94%D0%B6%D0%BE%D0%BD_%D1%84%D0%BE%D0%BD) в сороковых годах, а класс элементарных был выделен из них [Стивеном Вольфрамом](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%BB%D1%8C%D1%84%D1%80%D0%B0%D0%BC,_%D0%A1%D1%82%D0%B8%D0%B2%D0%B5%D0%BD) в его книге [“A new kind of science”](http://www.wolframscience.com/nksonline/toc.html), перевода которой на русский язык я не нашёл. Так же доступно написанное описание их работы находится в приложении.

*Каковы входные данные алгоритма?*

Во-первых, главной информацией, необходимой для работы алгоритма, будет строка символов, которую пользователь желает закодировать. Во-вторых, в соответствии с предыдущим пунктом нам понадобится номер правила для элементарного клеточного автомата. И всё! Обеспечивая безопасность, алгоритм будет иметь небольшой объём входных данных.

*Каков вывод алгоритма?*

Выводом алгоритма будет являться закодированная входная строка. Но при этом, за счёт использования элементарного клеточного автомата, мы сможем добиться неизменности размера этой строки.

Теперь давайте я соберу всё это в единое целое и поясню, к чему именно мы пришли.

Вот, что предстоит сделать. Нам нужно создать программу на языке программирования, которая будет принимать от пользователя строку для закодирования и номер правила для элементарного клеточного автомата. В этот клеточный автомат чудесным образом мы вводим входную строку и в какой-то момент “вынимаем” её оттуда в зашифрованном виде, выводя результат на экран пользователя! Ясно? Считаю, что да. Осталось описать детали.

Вас, конечно же, заинтересовал мой “чудесный” образ ввода строки в поле элементарного клеточного автомата. А реализуется он очень просто. С первых уроков информатики нам известно, что информация в компьютере хранится в виде двоичного кода. А ещё, если вы заглянули в дополнение, то узнали, что поле элементарного клеточного автомата тоже можно представить в виде двоичного кода. И вот мы, в корыстных целях, обманываем наш клеточный автомат, и “скармливаем” ему нашу входную строку! Он просчитывает определённое количество состояний и выдаёт нам готовый результат. Идеально! Ввёл строчку, ввёл число и на тебе - шифр! Но вводить всю строку для меня показалось сложным, поэтому мы передадим в алгоритм каждый её символ отдельно. Это сделает наш алгоритм более универсальным и мы сможем скормить ему даже простое целое число.

Теперь, когда мы определились с работой и внутренним устройством алгоритма, можно перейти к реализации.

# Ресурсы

Решено, что алгоритм будет записан в виде программы, исполняемой на компьютере. Для этого мне понадобится знание языка программирования и компьютер, где программа может быть протестирована. Всё это у меня имеется.

Для воплощения моей идеи в жизнь я выбрал такой знаменитый язык программирования как [C++](https://ru.wikipedia.org/wiki/C%2B%2B). Этот язык удовлетворяет всем требованиям, которые так важны для моего проекта, а это скорость и ясность. Но основной причиной его выбора послужили [статическая типизация](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) и [объектно-ориентированность](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%25D) языка. Это значит, что все мои ошибки, связанные с [типами](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) переменных, будут выявлены на этапе [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80).

Для компиляции мне понадобится [Интегрированная Среда Разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8). Я нашёл 2 IDE и один компилятор. Дело в том, что за компьютером сижу я редко, а смартфон всегда с собой, поэтому я установил одну IDE ([MSVC 2008](https://ru.wikipedia.org/wiki/Microsoft_Visual_C%2B%2B)) на ноутбук с Windows 7, одну IDE ([C4Droid 5.96](http://4pda.ru/forum/index.php?showtopic=246914)) на смартфон и компилятор ([G++](https://ru.wikipedia.org/wiki/G%2B%2B)) на ПК.

Компьютер так же имеется, при этом есть компьютер как на [Linux](http://lubuntu.net/), так и на [Windows](https://ru.wikipedia.org/wiki/Windows_7), поэтому в домашних условиях я смогу провести полноценные тесты на обеих системах. Но компьютеры отличаются ещё и комплектацией, а, следовательно, проверка производительности алгоритма будет ещё более качественной.

Всё оборудование готово ([без единой потраченной копеечки](http://www.utorrent.com/intl/ru/downloads/win)), пора начинать действовать.

# План выполнения

План очень прост и последователен, действия вытекают из всего вышесказанного, и смысла приводить его не было, но формальность требует от меня этого, и отказать я не смею:

1. Реализация алгоритма на языке C++
2. Тестирование алгоритма на 3 разных системах
3. Переписывание алгоритма на языке [PHP](https://ru.wikipedia.org/wiki/PHP)
4. Тестирование алгоритма на [сервере](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5))
5. ???
6. PROFIT!

Но в реальности всё было не так просто, и до пункта 3 ещё так далеко, как до Луны пешком. Какие появились проблемы в ходе реализации? Рассмотрим далее.

# Реализация

Как я уже сказал выше, реализовать алгоритм оказалось не так-то просто. Первые трудности подстерегали меня уже в самом начале, но всё по порядку.

**Проблема №1**

Во-первых, как поступим с многобайтовыми кодировками? Ведь во всём мире уже давно используют в программах символы национальных алфавитов, а на данный момент реализация поддерживает лишь кодировку ASCII. Это недопустимо! И что делать? Оставлять как есть? Нет, но в стандартной библиотеке нет типа, который точно поддерживал бы расширенный набор символов (не считая wchar\_t, но и он не даёт гарантий – это может быть лишь псевдоним типа char), а лишние зависимости ухудшают переносимость, поэтому стоит подумать перед тем, как их вносить.

**Возможное решение:** в будущих версиях я просто заменю символьный тип на целочисленный, имеющий больший размер. При этом вызывающей стороне придётся самой конвертировать символ в его целочисленный эквивалент.

**Проблема №2**

Во-вторых, нужно разбить объявление и определение класса по разным файлам. Сейчас для наглядности всё находится в одном файле, напоминающем адскую мешанину. Также имеются и вполне реальные проблемы: по умолчанию, все функции-члены класса объявляются как inline, но, я надеюсь, вы заметили, что функции моего класса достаточно велики для подстановки. Поэтому их определения нужно вынести из класса не только по этическим соображениям.

**Решение:** я вынес классы в отдельные файлы и собрал в одну папку, из которой файлы и подключаются.

**Проблема №3**

Лень (комментариев нет, как и решения).

**Проблема №4**

Я не совсем понимаю, что происходит. С самого начала я кинулся строчить код, не задумываясь о смысле написанного. Теперь я об этом очень жалею. Из-за возникшего глубокого недопонимания между мной и кодом было решено, что стоит переписать программу с нуля. А можно было просто сначала подумать…

Но, несмотря на все эти проблемы, список которых очень даже неполный, программа работает и выполняет поставленные задачи, что вызывает у меня неописуемую радость и искреннее недоумение.

# Оценка итогов работы

Хотя моя реализация метода шифрования и имеет все вышеописанные недостатки, но я с полной уверенностью считаю, что её можно назвать успешной. Вспомните определение. Алгоритм и быстр, и ~~не~~ясен для понимания, и оперирует небольшим количеством входных данных и т.д. Не рассматривайте этот абзац, как попытку оправдаться. Моя программа выполняет свои обязанности, а если вы ждали от неё большего, то вы не правы, и требовать чего-то ещё не смеете, так как она не должна соответствовать вашим ожиданиям. Ваши требования завышены лишь потому, что вы неправильно поняли задачи проекта, вернитесь и перечитайте.

Самостоятельно оценить выполненную мною работу вы сможете по [ссылке](http://www.itgnews.ga/CA.html). А я продолжу разрабатывать и улучшать свой алгоритм, буду пользоваться им в своих дальнейших проектах.

С советами и предложениями обращайтесь по адресу [kukushechkaebushechka@gmail.com](mailto:kukushechkaebushechka@gmail.com)

# Литература

Ссылки на источники информации прилагаются в виде ссылок в тексте работы по мере появления неизвестных терминов. Здесь я укажу лишь основные источники этих ссылок, а именно:

1. Википедия – ru.wikipedia.org
2. Хабрахабр™ – habrahabr.ru

# Приложение. О работе элементарных клеточных автоматов

Здесь я постараюсь объяснить об этом наиболее понятно, никакой научной речи здесь нет, её вы найдёте, перейдя по ссылкам в тексте.

Начнём с элементарных клеточных автоматов. Что это такое, клеточный автомат? Это бесконечная лента, поделённая на клеточки, при этом каждая клетка может иметь одно из определённого заранее множества состояний. Обычно, то, что клетка в данный момент имеет какое-то состояние, обозначают цветом. То есть в итоге у нас получается цветная лента. Она называется полем клеточного автомата. В элементарном клеточном автомате имеется лишь два состояния: 1 или 0, чёрный или белый, включено или выключено. Ясно?

Такое поле реализовать на практике невозможно, потому что ресурсы человека всегда ограничены. Сможете ли вы запомнить состояния бесконечного множества клеток? Конечно, нет. Поэтому в реальности ленту либо ограничивают в размерах, получая отрезок, либо соединяют концы, создавая кольцо.

Самая важная и интересная особенность поля в том, что клетки его могут менять состояние в определённый момент времени. Как часто следующее состояние клетки будет просчитываться, выбирает человек. И так, просчитав следующее состояние каждой клетки поля, мы получаем уже вторую ленту клеток. Каждая такая лента называется поколением клеточного автомата. Просчитали ленту, сохранили, просчитали, сохранили и т.д. Если выводить поколения в хронологическом порядке, то получится узор необычайной красоты, называемый эволюцией клеточного автомата.

Так как же нам просчитать следующее состояние клетки? Для этого предопределены правила перехода состояний, в зависимости от состояний клеток-соседок. Состояния клеток-соседок и клетки посередине называется окрестностью. Всего возможно 8 случаев: 111 110 101 100 011 010 001 000. Если предыдущие предложение вам непонятно, то напомню, что клетки одного состояния обозначают 1, а клетки другого – 0. Правило перехода должно рассмотреть все эти случаи окрестности. При этом, правило определяет, как в таком случае окрестности изменится клетка посередине. То есть, чтобы рассчитать следующее ПОКОЛЕНИЕ (не СОСТОЯНИЕ) нужно перебрать все клетки нашего поля, узнать, к какому случаю окрестности они подходят и заменить текущее состояние состоянием, которое определено правилом перехода для данной окрестности. Состояния эти соответствуют окрестностям, записанных в том же порядке, как и было написано выше:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| окрестность | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
| состояние | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

Это пример правила перехода. Но, чтобы не чертить таких таблиц, принято записывать правила в виде десятичного числа. Вот как это делается:

1. Составляют таблицу, описанную выше
2. Выписывают строку “состояние” -> 10011010
3. Представляют, что выписанная строка – двоичное число, и переводят его в десятичную систему счисления -> 154
4. Готово!

Такая запись гораздо удобнее громоздкой таблицы.

Вот примерно так они и работают.