Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Лабораторная Работа №3.   
Методы традиционного шифрования

Отчет по лабораторной работе №3 по учебной дисциплине   
«Методы защиты информации»

по специальности 09.03.01 Информатика и вычислительная техника

|  |  |
| --- | --- |
|  | Руководитель  / Т. В. Жгун  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  | Студент группы 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Оглавление

[Решётка Кардано 3](#_Toc6393)

[Шифр Цезаря 5](#_Toc29947)

[Приложение А 9](#_Toc22184)

[Приложение Б 15](#_Toc25017)

Решётка Кардано

Решётка Кардано — исторически первая известная шифровальная решётка, трафарет, применявшийся для шифрования и дешифрования, выполненный в форме прямоугольной (чаще всего — квадратной) таблицы-карточки, часть ячеек которых вырезана, и через которые наносился шифротекст, применение решётки является одной из форм ***стеганографии.***  
 Для того чтобы воспользоваться таким шифром, отправитель и получатель должны были иметь одинаковые листы из картона (или из другого подходящего материала) с прорезанными в определенных местах отверстия-трафарет. Этот лист с прорезями накладывался на другой лист, и через прорези вписывался тот текст, который нужно было передать. Затем трафарет убирался и оставшиеся место заполнялось другими словами, значками, буквами. А чтобы посторонний, к которому может попасть послание , ни о чем не догадывался, общий текст должен был выглядеть связным. Получатель письма накладывал свой трафарет и прочитывал тайное письмо.

В том, чтобы сочинить связный текст, включавший тайное послание. Состояла главная трудность шифровки. Для написания связного текста, подогнав его в написанный текст, нужно было обладать литературным талантом. Этим методом пользовался великий русский поэт А.С. Грибоедов. Под видом обычных писем он посылал секретные дипломатические сообщения из Персии, где служил русским послом.

По аналогии была создана **решетка с поворотом**, это более экономичный вариант. После того, как написана часть текста и все отверстия заполнены, решетка поворачивается, и новые буквы так же пишутся в отверстиях, которые находятся теперь в новом месте, так поступают еще два раза. В результате получается заполненный буквами квадрат. Пока не наложить на него решетку, этот набор букв будет бессмысленным.  Обычно использовали квадраты 8х8, по 16 отверстий, то есть вписать текст можно было из 64 букв. Если послание короче 64 букв, оставшиеся места можно было заполнить случайными буквами (символами). Если же послание длиннее, то решетку можно использовать второй раз, создавая рядом еще один квадрат с буквами.

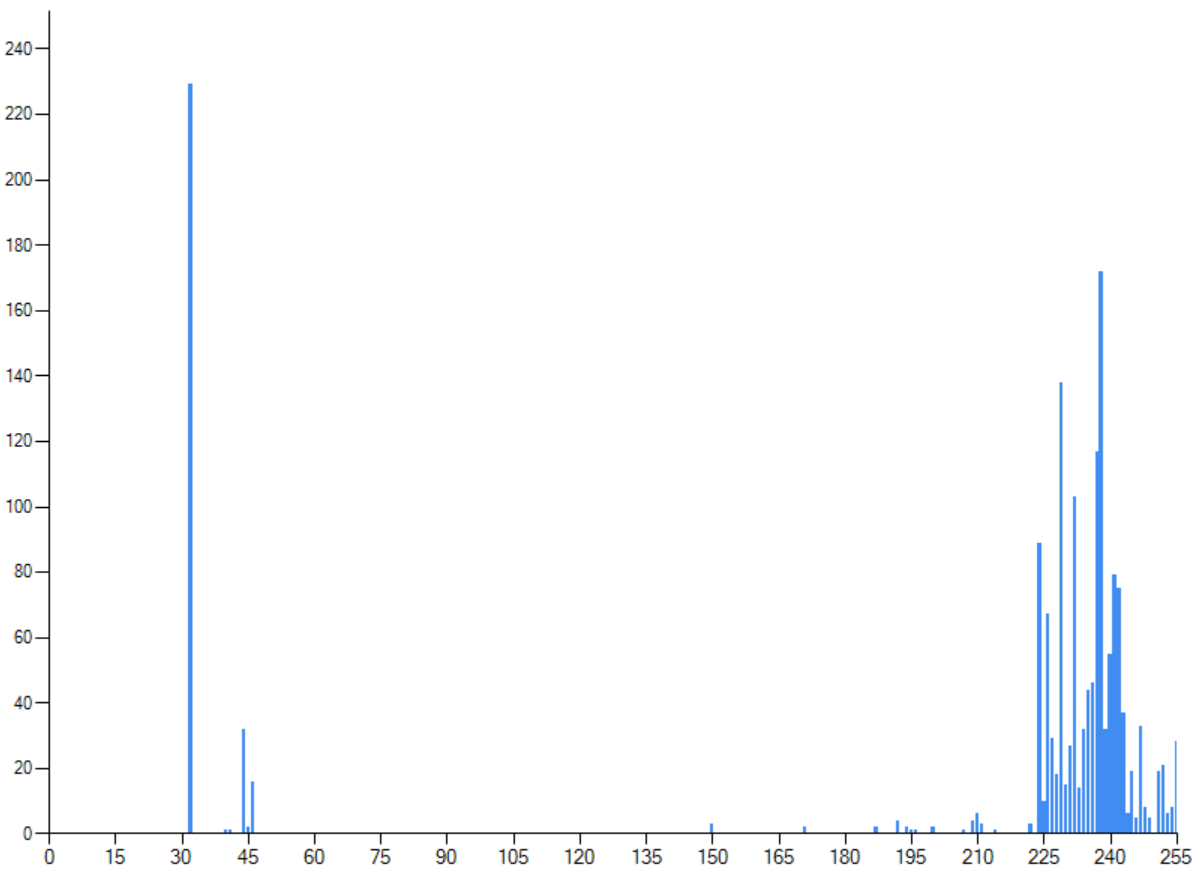
При выполнения работы использовался способ решетки с поворотом. При использовании этого шифра был использован метод блочного шифрования, в соответсвии с матрицей блок составляет 64 символа Сама решётка является ключем к зашифрованному тексту, в моем случае использовалась следующая решетка, представленная в виде булевой матрицы:

Булевая матрица (поворотная решётка Кардано)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

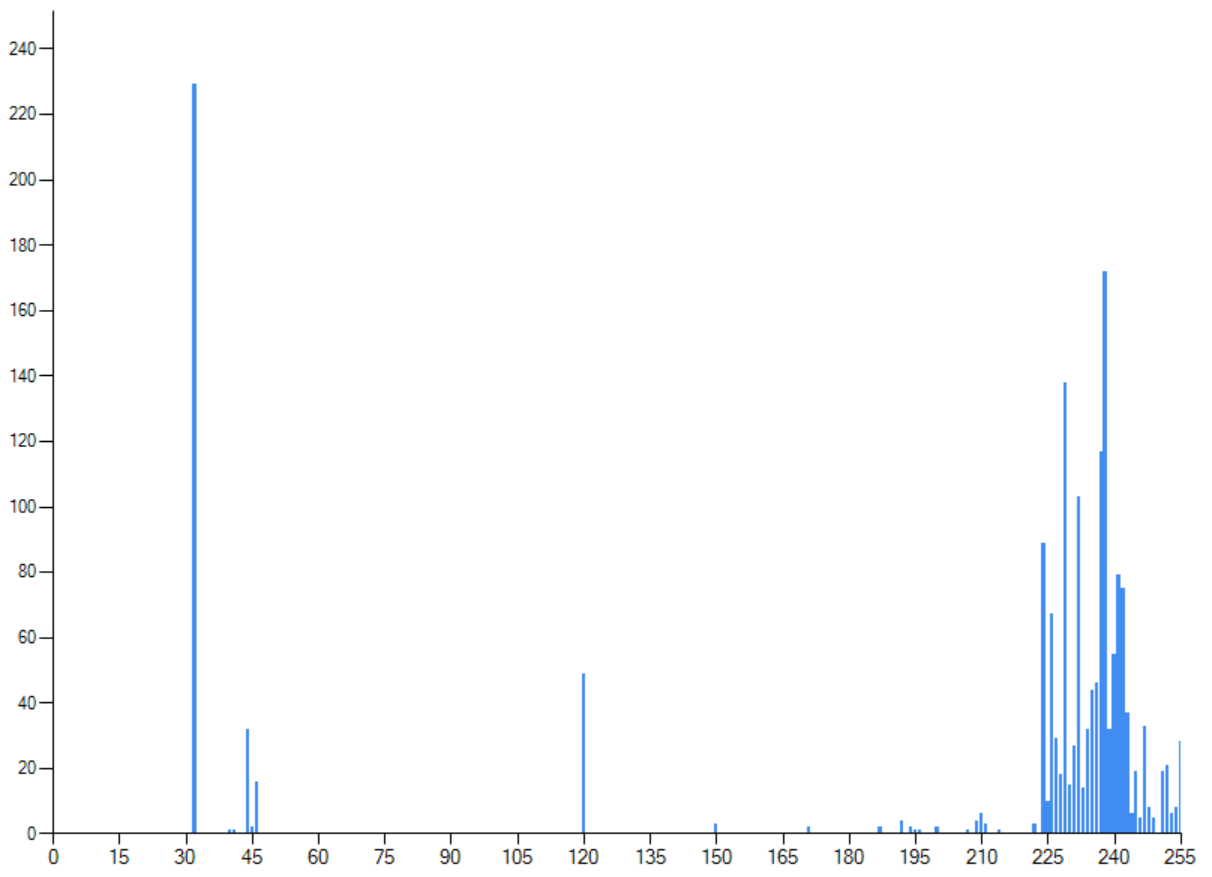
Таблица 1 – Графическое представление булевой матрицы  
(поворотная решётка Кардано)

Гистограммы исходного текста и шифротекста полученного шифрование поворотной решёткой Кардано представлены ниже.



H(x) = 4,584 бит

Рисунок 1 – Гистограмма исходного (открытого) текста



H(x) = 4,641 бит

Рисунок 2 – Гистограмма шифротекста, полученного повротной решёткой Кардано

Шифр Цезаря

Шифр Цезаря — это один из самых простых и известных методов шифрования, который относится к классу симметричных шифров. Он был назван в честь Юлия Цезаря, который использовал его для защиты своих военных сообщений.

Математическая модель шифра Цезаря:

При выполнении работы было разработано три метода шифрования:

1. Шифрование с ключем константой – 17.

Каждый символ исходного текста символа сдивигается на одно и то же значение.

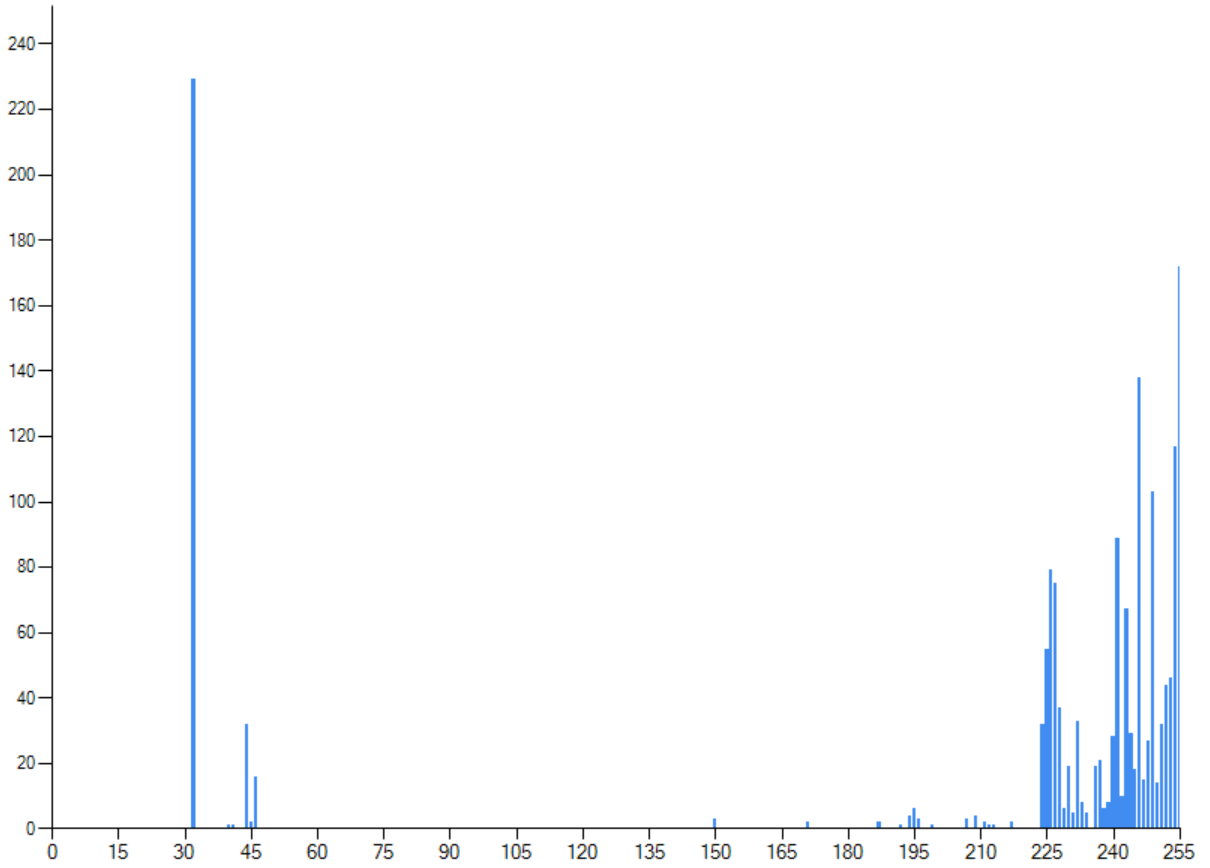
1. Шифрование с ключем поговорокой – Пустой колос голову кверху носит.

Каждый символ исходного текст сдвигается на разные значения, на значению соответствующего символа в ключе.

1. Шифрование с ключем ПСП – 01010101.

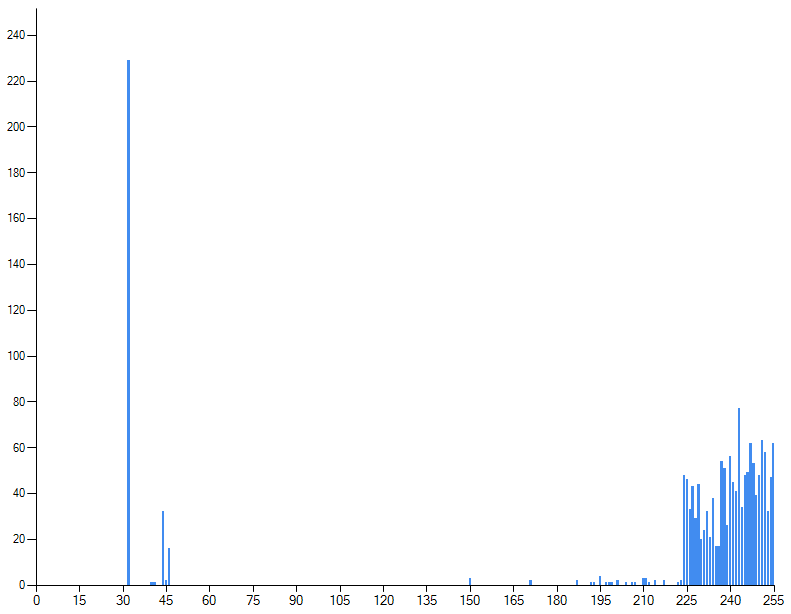
Каждый символ исходного текста сдвигается на разные значения, свдиг генерируется ГПСЧ LFSR, сделанные в прошлой лабораторной работе.

Использование различных сдвигов способствует изменению статистики зашифрованного текста относительно открытого текста, в отличии от константного сдвига. Ниже представлены гистограммы полученных текстов.



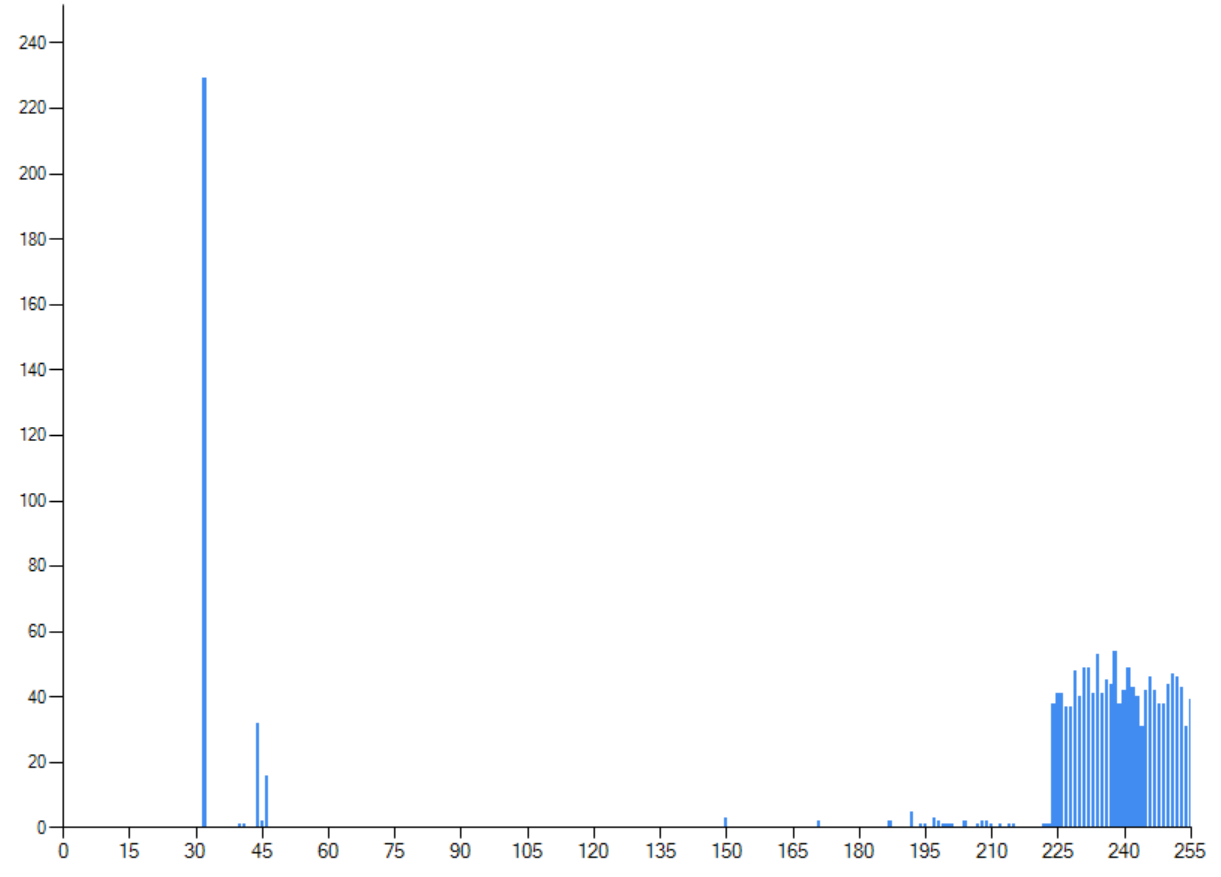
H(x) = 4,584 бит

Рисунок 3 – Гистограмма шифротекста, полученного шифрованием Цезарем с константным ключем (17)



H(x) = 5,023 бит

Рисунок 4 – Гистограмма шифротекста, полученного шифрованием Цезарем с ключем поговоркой (Пустой колос голову кверху носит)



H(x) = 5,086 бит

Рисунок 5 – Гистограмма шифротекста, полученного шифрованием Цезарем с ключем ПСП (инициализирующий вектор 01010101)

Открытый текст и полученные шифротексты представлены в Приложении А. Исходный код приложение представлен в Приложении Б.

# Приложение А

Открытый текст:

То, что в жизни, наступившей ныне, императоров, видимо за отсутствием империи, уже нет, Тереховым не оценивается вообще никак. Сочинение его, притом что оно сплошь и рядом населено политиками и спецслужбистами, совершенно безоценочно. Вернее сказать, у Терехова, как и у всех вышеназванных, ощущение времени возникает не посредством позитивно или негативно окрашенного публицистического высказывания, но через атмосферу текста.

И атмосфера эта в той или иной степени определяется словом «морок». У Терехова замороченность всего происходящего чувствуется особенно остро, в меньшей степени это есть и у Юзефовича. Аномально поведение главного героя у Самсонова – что, собственно, отражено уже в названии романа. Аномальна, на наш вкус, и главная сюжетная коллизия «Цены отсечения» Архангельского. То есть времена срослись, но, как ни странно, полегчало ненадолго. После прочтения каждой из этих книг остается горький осадок. Горчит по-разному – но горчит неизменно.

И не только по причине невольного погружения в эту выморочную, аномальную, нездоровую атмосферу. Другой особенностью всех четырех книг является острое, порой даже патологическое, мучительное внимание к межполовым отношениям. У Архангельского это чувствуется чуть менее (хотя, пожалуй, он в этом вопросе циничней всех). У Самсонова, у Юзефовича и в особенности у Терехова сексуальные мотивы являются едва ли не навязчивыми.

Самсонов сохранил хоть какие-то признаки идеализма при взгляде на мужчину и женщину, Юзефович – почти уже нет, зато он самый, что ли, снисходительный из числа рассматриваемых нами. В любом случае брезгливые, а иногда и злые описания греховной человеческой породы просматриваются у каждого.

Шифротект, полученнный шифрованием решёткой Кардано:

еТво,и, н ад исичмтмтооп уе зпраив ва ошжтетиойр сзонунтсвыи,н ,мт в вижен оееномб ещиоетцм ,ен пиТнекерираивеахкиео., Ств сыяуоолчшеиитьонн мое ни пи очрелтяио т деиогонкомо а,нсм и аппслерзие о си,ц спескеацнозсоавчлтеньру,шож еб.нуи нВо се Ттеррбанеемхеех,носе хиво ав,щво узыщкншеиеакннкаа изеиев ат вн унерн еымви пвпоитниусовбрле нидоцокси рситтлавииош чменнее нгспкооаогтозегторм нуоов сыфстчекерекаарс езэзтыт ваааа.т

м н

ови тИся фа, о йсУ енли ио Тлеивор оепмихр оенвд«оаеймл оярезсотатмоксе»я.пурновнч псроето нонвисоутсесрхтоот,дсь я ящвв еог ссомеено гбчеоиьошчве еасей.тд ес ьнтАи нееио пгумел наЮаилзьев нфнэогоот вповоу ежговнаеен рвоо– я,н ачозтувот а,Ср наасжоимебис рнсоот н оямг а, лнкнааоал.в л нинаАзаяишня о вмск«аюужсЦле,еьныт нниаас лотитнгьесс лье,чвь ернсенокм,ои геяко»н.а а ТкАсо рнир хоесао ассжталлодре оайн н нпиерозно а,эдч топтлиоегнохли. екня г Пкчиингто ки й оп гсотоора-чсреиаатдт озснкян.е о мГигуозоме рр–ч ьонинляоо ьп .нов

о эИгтпо ур инвчпеыи онмтгер ооунрлочжеьевнксноуфбю неее,рзн наудон.со отрДьомювра улуювьг оаснйтеух мюоос,рчиеочтяейтеы ссркедяохае ж ,ое скмт нурипочгает,и о тяеллпвооглеьлнньомеижсеяпк овмогн.ол и оУэвмты аомнА ирохтчеану вснокгше йтрв,оуее сео(ет снхця иов нт иячч,эун ттпеьоомжй а мвсвлеоупнвеТх е), оур.с е хУоЮо бвзеСаен фасонмевсоикосчтасни уоал иву аньмнасыютвсоеяян омз воч еитсдвовихаывр ымла ии. ня

нивл

елС ая хмопутизрнжьиач ик кнавуизк гиии длеже-яаетдлеино зщ иннмпаа роул, и чтои,Юн зсе унфсижаосемх воныидечйти ,, т–чзе льтапот оинеы. йсм аб Втриез рз лгиючлвбиаивеосымлмыеа хс , лн ра уаачмас ирнчооисеадглныд аоип вряеио чсг мрезаелсхтыкооврейни ва оопйопxюxтxxсxxxxxяxxx xуxxx xxxxкxxxxxаxxxжxxдxxxоxxxxxгxxxоxxxx.xxxx

Шифротект, полученнный шифрованием шифром Цезаря с ключем константой (17):

Гя, игя у чщшющ, юсвгдащуйцъ юмюц, щэацбсгябяу, ущхщэя шс ягвдгвгущцэ щэацбщщ, дчц юцг, Гцбцжяумэ юц язцющусцгвр уяяткц ющысы. Вяищюцющц цфя, абщгяэ игя яюя ваьяйн щ брхяэ юсвцьцюя аяьщгщысэщ щ вацзвьдчтщвгсэщ, вяуцбйцююя тцшязцюяиюя. Уцбюцц высшсгн, д Гцбцжяус, ысы щ д увцж умйцюсшусююмж, якдкцющц убцэцющ уяшющысцг юц аявбцхвгуяэ аяшщгщуюя щьщ юцфсгщуюя яыбсйцююяфя адтьщзщвгщицвыяфя умвысшмусющр, юя ицбцш сгэявецбд гцывгс.Щ сгэявецбс огс у гяъ щьщ щюяъ вгцацющ яабцхцьрцгвр вьяуяэ «эябяы». Д Гцбцжяус шсэябяицююявгн увцфя абящвжяхркцфя идувгудцгвр явятцююя явгбя, у эцюнйцъ вгцацющ огя цвгн щ д Пшцеяущис. Сюяэсьнюя аяуцхцющц фьсуюяфя фцбяр д Всэвяюяус – игя, вятвгуцююя, ягбсчцюя дчц у юсшусющщ бяэсюс. Сюяэсьнюс, юс юсй уыдв, щ фьсуюср впчцгюср ыяььщшщр «Зцюм ягвцицющр» Сбжсюфцьнвыяфя. Гя цвгн убцэцюс вбявьщвн, юя, ысы ющ вгбсююя, аяьцфисья юцюсхяьфя. Аявьц абяигцющр ысчхяъ щш огщж ыющф явгсцгвр фябныщъ явсхяы. Фябищг ая-бсшюяэд – юя фябищг юцщшэцююя.Щ юц гяьныя ая абщищюц юцуяьнюяфя аяфбдчцющр у огд умэябяиюдп, сюяэсьнюдп, юцшхябяудп сгэявецбд. Хбдфяъ явятцююявгнп увцж ицгмбцж ыющф руьрцгвр явгбяц, аябяъ хсчц асгяьяфщицвыяц, эдищгцьнюяц ующэсющц ы эцчаяьяумэ ягюяйцющрэ. Д Сбжсюфцьнвыяфя огя идувгудцгвр идгн эцюцц (жягр, аячсьдъ, яю у огяэ уяабявц зщющиюцъ увцж). Д Всэвяюяус, д Пшцеяущис щ у явятцююявгщ д Гцбцжяус вцывдсьнюмц эягщум руьрпгвр цхус ьщ юц юсуршищумэщ.Всэвяюяу вяжбсющь жягн ысыщц-гя абщшюсыщ щхцсьщшэс абщ ушфьрхц юс эдчищюд щ чцюкщюд, Пшцеяущи – аяигщ дчц юцг, шсгя яю всэмъ, игя ьщ, вющвжяхщгцьнюмъ щш ищвьс бсввэсгбщусцэмж юсэщ. У ьптяэ вьдисц тбцшфьщумц, с щюяфхс щ шьмц яащвсющр фбцжяуюяъ ицьяуцицвыяъ аябяхм абявэсгбщуспгвр д ысчхяфя.

Шифротект, полученнный шифрованием шифром Цезаря с ключем поговоркой (Пустой колос голову кверху носит):

Бб, еыо р фщзрц, пусьхфшчлец югяф, ъъшеъоэьбое, ркчицр чх ояядъгбхщчъ ицэрющи, юфз нпф, Зшртгякны юч чцпыурсехяк дболык выкнш. Гэкщяуцип рся, эыцфбм щчю бны вчээлн ц рйтщъ нгярщзао суыэеичоэр ч вбуясхбспщсхочц, сшдканшнъь йчцбзчычччь. Уеуыру дккйевс, а Гнвфияфо, фох щ ц нязи мээхвузпоюхнд, азьщпыуу вуучупы мрмэээата хч вягюодыаньэ тьтцфывчр шаы ъуфидчхюа чкъогуюнссщ сжбхкышжеидувтатб фйъккхжрснлн, пб бзххь аяъящжфгд аокыал.Щ гачьузеъв нзу п гцы ыьъ сншч ягетушц бпъзйхатеяяр гъбуаъ «цьыьы». Ю Фшрпчутх знъяшажшюяьътж няцгс ъюрысярйпошгы иыфаеуеуысй щяябиышь бсьту, х щуюдкфь гаоппыу отс ряфп т ш Уъебьурйп. Тычмкщзыя тьнужшнтз уаувъьфц тшбан у Ялъворьно – бфу, дооягкчьая, чтъосуюо бсу х чвмтхаих бцюпас. Йншълщннг, ыв нкъ тяжс, щ хъууяои ымсугнгн шрюлтйнп «Цтым абдцйуций» Обхгыоунпсфрию. Ты цщдл увухечо ябофщуяю, чр, яук ыщ гбгсяыч, эщщцгъоць аечвйюацо. Ацгъш бюччьушцр нострь тй нзых шюрх бвдоотын сяряшуч бскжуъ. Гыюирд вя-юйзчьчб – ыщ ебрбкч вшифъцхяэ.Щ ыо ььцкыо эщ сгибктх аепььдяэця эчгъбсуюив н яеу даьггодыдж, уюаъйлжыюм, руттргомхг хемыяенвв. Цюьгшч ьводушырдтжа тжшх ецъняшж шцин крьяиаьн бсьтух, пыюяс уучч шаььцьфиъуьшрш, ошзэеешкюцч хюъъйнту ш мифъьнбвео юзаоеуюрсы. Е Йряошсцляяхьеб зфу мжвюауычбдр еьтж чуюеи (грея, фюыулач, аь у лыоц ньарсяр шынтщтхю вюуж). Ж Гохсшыщрс, б Мйшфшднзх и р цгэфцяычсьц б Тиюргрха укъжжашкюгч яядцлы ннщрюхяк зчвк рш ае ысксцкщфйхи.Ялъворьн убхъвтша хыан ьпэщч-ыо эыцшнгшу кчекннчбу ьющ фццьсто чо ъджъцшб ы рзтйэау, Ппчгбуъе – эщеги бсу аеь, чхео ью гпямы, чьь щщ, яшцуиоокчхапнич рщ кщгщй ъоьяэахюурвшмеч эхяи. У энфяю ълэелу буутснывез, х иъьфмт ы щщде ьъцварцк егеярзэгь дуьцффкцгшчй эщюядю ъюрдмкфхшчуюяяр е эсштчгш.

Шифротект, полученнный шифрованием шифром Цезаря с ключем поговоркой (ПСП сгенерированная с инициализирующим вектором 01010101):

Йб, аак р мфаьк, эбрхндюорхй ощйэ, ксжчхцяхвтл, этшрэм оу щэыийдмцсуи цйхсйчк, гзд зъи, Вержукъмл ть умтфъжйфхмй кямюаш шуфхб. Ечеджукос фет, рплмгв пво плк вонупо ю чсичы икеньгкх кщцтзяэъас д ямлвкъхкйштсгжэ, йювесцбеюн жьщумтфаыцэ. Мщшюгв делткзу, з Абйутфощ, оиъ з н шээе вьцбесждедяал, аэьииитщ уовушиу мгюавюйуо ыв ызатймбубсж еъяштиглк щкк дчицяпфсч сеъфацлкхцй ъэцвырьъадруоръьэ жгблякхчцщап, ом пцпзм тчвышжйщв нпющгю.П ыэчшжлшкф лощ я юзш муш зрию экхпеож жапзйьэдыяшс ъъсэша «клчбе». И Еядогкыо нмеэттяхомслзт ъбегп лиязуъецдптка авемьцыцрож йьщлъдаи чяойь, о ызсдижи лзыыээи юрк црфб ъ й Ещйээегбф. Юкхяыцзчг вицотбжцв пдпдсцуп жяедл г Сбкнжюнде – нях, ъэдмьцнюлл, йэыкыьаи ьфб р умасвсрш псжхгм. Анпкьгнмв, яе ъзк лщцм, р бизхилк жхщяжцоы шлсчбцкг «Чдрх дюйхчеожы» Ятъчяиышггочтс. Жц гощп ныпбьаъ ъюккщечи, пт, лян вю йвраолк, оррьхьцшх соьгяшяля. Цбмцр дзбсжоыдш зжтээл рч ьхвк цешг ппошцсуд хужйсън эфыовт. Ахгтуэ де-кфрыкеб – жэ зцашзх выфяьеном.А мз йартчх уч тлтлрюг фшэщцжвеци шьяйбглщбо ж нут ьрвъиючнфь, смрсчэбгае, цфкяшдцусы унчщыйьгн. Тммслп заренэонфмсф ъбех хбкмпзъ ьтюр сжфоиныу яппчба, щгзбг новю мжюзързрзжрниъ, дгчиугзфюнз щянвнфъй щ зпъчяйлйоз щьвелябсни. Р Мйдвслхмыфдгщъ нто хпъвсдшьдцх юеце пачщн (тхеъ, щгэуезт, зы и цбрр тпоуижы ошнишлбб букм). А Гдхасишци, р Свряшчякъ с ю ьофнюьптщвй ц Зыьэеовб нэырхевотст ютычец укьэыщдъ рочч еь ыб ыэилажкжгьй.Фъбзъеюв тмсисмкр зуий ьдучи-ьв аоеоаыху эышъясхищ мцф сйзупед зх шлцчиос а езтрътй, Рлогсэтл – мхкну эыь зщы, аопф зь хиььи, миъ ыи, пйавфрйядкбйфнн чк бьщью чумьчкззыьфоъчо кжшб. Ж ыяасж зчлзае ямэшвннщнк, з мцэжяк р еивш щътжчаву смюглищзш ыныпбисъзцжщ ппокьм схегсцячъжйнхмй ы иэнчйощ.

# Приложение Б

Листинг файла MZI\_3.cpp

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

#include <Windows.h>

#include "Cardano.h"

#include "Caesar.h"

const std::string reset = "\033[0m";

const std::string red = "\033[31m";

const std::string green = "\033[32m";

void method\_menu();

void Caesar\_number();

void Caesar\_text();

void Caesar\_LFSR();

std::string get\_data();

void save(std::string data);

size\_t number = 17;

std::string text = "Пустой колос голову кверху носит";

std::string seed = "01010101";

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char choice = ' ';

while (choice != '3') {

system("cls");

std::cout << "МЕНЮ\n\n1. Шифрование решёткой Кардано\n2. Шифрование Цезарем (3 метода)\n3. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice) {

case '1': std::cout << "Приложение открыто в другом окне\n\n"; system("Cardano.exe"); break;

case '2': method\_menu(); break;

}

}

return 0;

}

void method\_menu() {

char choice = ' ';

while (choice != '4') {

system("cls");

std::cout << "МЕНЮ ШИФР ЦЕЗАРЯ\n\n1. Ключ - константа\n2. Ключ - поговорка\n3. Ключ - ПСП\n4. Вернутся в основное меню\n>> ";

std::cin >> choice;

system("cls");

switch (choice) {

case '1': Caesar\_number(); break;

case '2': Caesar\_text(); break;

case '3': Caesar\_LFSR(); break;

}

}

}

void Caesar\_number() {

system("cls");

Caesar Caesar;

char choice = ' ';

while (choice != '5') {

system("cls");

std::cout << "КЛЮЧ КОНСТАНТА\n\n1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Вернутся в меню Цезаря\n>> ";

std::cin >> choice;

system("cls");

std::string data = "1";

if (choice == '1' || choice == '2') {

data = get\_data();

if(!data.empty())

std::cout << red << "\nИсходный текст:\n" << reset << data;

}

if(!data.empty())

switch (choice) {

case '1': {

data = Caesar.encode(data, number);

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '2': {

data = Caesar.decode(data, number);

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '3': {

std::cout << "Текущий ключ: " << number << "\n\nВведите новых ключ\n>> ";

std::cin >> number;

break;

}

case '4': {

std::cout << "Текущий ключ: " << number << "\n\n";

system("pause");

break;

}

}

data = "1";

}

}

void Caesar\_text() {

system("cls");

Caesar Caesar;

char choice = ' ';

while (choice != '5') {

system("cls");

std::cout << "КЛЮЧ ПОГОВОРКА\n\n1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Вернутся в меню Цезаря\n>> ";

std::cin >> choice;

system("cls");

std::string data = "1";

if (choice == '1' || choice == '2') {

data = get\_data();

if (!data.empty())

std::cout << red <<"\nИсходный текст:\n" << reset << data;

}

if (!data.empty())

switch (choice) {

case '1': {

data = Caesar.encode\_text\_key(data, text);

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '2': {

data = Caesar.decode\_text\_key(data, text);

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '3': {

std::cout << "Текущий ключ: " << text << "\n\nВведите новых ключ\n>> ";

std::cin.ignore();

getline(std::cin, text);

break;

}

case '4': {

std::cout << "Текущий ключ: " << text << "\n\n";

system("pause");

break;

}

}

data = "1";

}

}

void Caesar\_LFSR() {

system("cls");

Caesar Caesar;

char choice = ' ';

while (choice != '5') {

system("cls");

std::cout << "КЛЮЧ ПСП\n\n1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Вернутся в меню Цезаря\n>> ";

std::cin >> choice;

system("cls");

std::string data = "1";

if (choice == '1' || choice == '2') {

data = get\_data();

if (!data.empty())

std::cout << red << "\nИсходный текст:\n" << reset << data;

}

if (!data.empty())

switch (choice) {

case '1': {

data = Caesar.encode\_LFSR(data, std::bitset<8>(seed).to\_ulong());

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '2': {

data = Caesar.decode\_LFSR(data, std::bitset<8>(seed).to\_ulong());

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << data << std::endl << std::endl;

save(data);

break;

}

case '3': {

std::cout << "Текущий ключ: " << seed << "\n\nВведите новых ключ\n>> ";

std::cin.ignore();

getline(std::cin, seed);

if (seed.size() != 8) {

std::cout << red << "\nДлина введенного ключа слишком мала, необходимо 8 битов\n\n" << reset;

system("pause");

}

break;

}

case '4': {

std::cout << "Текущий ключ: " << seed << "\n\n";

system("pause");

break;

}

}

data = "1";

}

}

std::string get\_data() {

std::cout << "Введите путь к файлу\n>> ";

std::string filepath;

std::cin >> filepath;

std::string temp, data;

if (filepath == "-m") {

std::cin.ignore();

std::cout << "\nВведите текст вручную\n>> ";

getline(std::cin, data);

return data;

}

std::ifstream in(filepath);

if (!in.is\_open()) {

std::cout << red << "\nНе удалось открыть файл (неверный путь)!\n\n" << reset;

system("pause");

return std::string();

}

while (getline(in, temp)) {

data += temp;

}

in.close();

return data;

}

void save(std::string data) {

std::cout << "Сохранить полученный текст в файл out.txt? (Y/N, m для ручного выбора файла)\n>> ";

char ch;

std::cin >> ch;

if (std::toupper(ch) == 'Y') {

std::ofstream out("out.txt");

out << data;

out.close();

system("cls");

std::cout << green << "Данные были успешно сохранены в out.txt\n\n" << reset;

system("pause");

}

else if (ch == 'm') {

std::string filepath;

std::cout << "\nВведите путь к файлу для сохранения\n>> ";

std::cin.ignore();

getline(std::cin, filepath);

std::ofstream out(filepath);

out << data;

out.close();

system("cls");

std::cout << green << "Данные были успешно сохранены в " << filepath << "\n\n" << reset;

system("pause");

}

}

Листинг файла Caesar.h

#pragma once

#include <Windows.h>

#include <fstream>

#include "LFSR.h"

typedef unsigned char symbol;

class Caesar {

public:

std::string encode(std::string data, unsigned int key) {

std::string result;

for (symbol c : data) {

result += encode(c, key);

}

return result;

}

std::string decode(std::string data, unsigned int key) {

std::string result;

for (symbol c : data) {

result += decode(c, key);

}

return result;

}

std::string encode\_text\_key(std::string data, std::string key) {

std::string result;

long long int offset = 0, i = 0;

for (symbol c : data) {

if (i == key.size()) {

i = 0;

}

offset = (offset + key[i]) % 255;

result += encode(c, offset);

i++;

}

return result;

}

std::string decode\_text\_key(std::string data, std::string key) {

std::string result;

long long int offset = 0, i = 0;

for (symbol c : data) {

if (i == key.size()) {

i = 0;

}

offset = (offset + key[i]) % 255;

result += decode(c, offset);

i++;

}

return result;

}

std::string encode\_LFSR(std::string data, unsigned int key) {

LFSR LFSR(key);

std::string result;

for (symbol c : data) {

result += encode(c, LFSR.generate().to\_ulong());

}

return result;

}

std::string decode\_LFSR(std::string data, unsigned int key) {

LFSR LFSR(key);

std::string result;

for (symbol c : data) {

result += decode(c, LFSR.generate().to\_ulong());

}

return result;

}

private:

symbol encode(symbol c, long long int offset) {

if (c >= 'a' && c <= 'z') {

c = (c - 'a' + (offset % 26) + 26) % 26 + 'a';

}

else if (c >= 'A' && c <= 'Z') {

c = (c - 'A' + (offset % 26) + 26) % 26 + 'A';

}

else if (c >= (symbol)'а' && c <= (symbol)'я') {

c = (c - (symbol)'а' + (offset % 32) + 32) % 32 + (symbol)'а';

}

else if (c >= (symbol)'А' && c <= (symbol)'Я') {

c = (c - (symbol)'А' + (offset % 32) + 32) % 32 + (symbol)'А';

}

return c;

}

symbol decode(symbol c, long long int offset) {

offset \*= -1;

if (c >= (symbol)'a' && c <= (symbol)'z') {

c = (c - (symbol)'a' + (offset % 26) + 26) % 26 + (symbol)'a';

}

else if (c >= (symbol)'A' && c <= (symbol)'Z') {

c = (c - (symbol)'A' + (offset % 26) + 26) % 26 + (symbol)'A';

}

else if (c >= (symbol)'а' && c <= (symbol)'я') {

c = (c - (symbol)'а' + (offset % 32) + 32) % 32 + (symbol)'а';

}

else if (c >= (symbol)'А' && c <= (symbol)'Я') {

c = (c - (symbol)'А' + (offset % 32) + 32) % 32 + (symbol)'А';

}

return c;

}

};

Листинг файла LFSR.h

#pragma once

#include <bitset>

#include <vector>

class LFSR {

public:

LFSR() : last\_generated(181) {}

LFSR(unsigned int seed) : last\_generated(seed) {}

std::bitset<8> generate() {

std::bitset<8> res;

for (int i = 0; i < 8; i++) {

bool feedback = last\_generated[0];

for (int j = 1; j < 8; j++) {

feedback ^= (matrix[i][j] ? last\_generated[j] : 0);

}

res[i] = feedback;

}

last\_generated = res;

return last\_generated;

}

private:

std::bitset<8> last\_generated;

std::vector<std::vector<bool>> matrix = {

{0, 0, 1, 1, 1, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0},

{0, 1, 0, 0, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 0, 0},

{0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 1, 0}

};

};

Листинг файла Cardano.cpp

#pragma optimize("agi", off)

#include <msclr/marshal\_cppstd.h>

#include <msclr/marshal.h>

#include "Cardano.h"

#include "ClassicMethods.h"

#include <Windows.h>

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

int main()

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

MZI3G::ClassicMethods form;

Application::Run(% form);

}

Листинг файла Cardano.h

#pragma once

#include <string>

#include <vector>

#include <Windows.h>

class Cardano {

public:

Cardano(std::vector<std::vector<bool>> grid, size\_t quantity) : grid(grid), start\_grid(grid), quantity(quantity), temporary(std::vector<std::vector<char>>(8, std::vector<char>(8, 'x'))) {}

void rotate(std::vector<std::vector<bool>>& grid) {

std::vector<std::vector<bool>> temp(8, std::vector<bool>(8));

for (int i = 0; i < grid.size(); i++) {

for (int j = grid.size() - 1; j > -1; j--) {

temp[(j - 7) \* -1][i] = grid[i][j];

}

}

grid = temp;

}

void clear(std::vector<std::vector<char>>& test) {

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

test[i][j] = 'x';

}

}

}

void reset\_grid() {

grid = start\_grid;

}

void write(std::string data) {

unsigned short int counter = 0;

for (int i = 0; i < grid.size() && counter != data.size(); i++) {

for (int j = 0; j < grid.size() && counter != data.size(); j++) {

if (grid[i][j]) {

temporary[i][j] = data[counter++];

}

}

}

status++;

rotate(grid);

if (status == 4) {

for (int i = 0; i < temporary.size(); i++) {

for (int j = 0; j < temporary.size(); j++) {

result += temporary[i][j];

}

}

status = 0;

clear(temporary);

}

}

std::string encode(std::string data) {

result.clear();

std::string block;

for (int i = 0; i < data.size(); i++) {

if (i != 0 && i % quantity == 0) {

write(block);

block.clear();

}

block += data[i];

}

if (!block.empty()) {

status = 3;

write(block);

status = 0;

clear(temporary);

}

reset\_grid();

return result;

}

void read(std::string data) {

unsigned short int counter = 0;

for (int i = 0; i < 8 && counter != data.size(); i++) {

for (int j = 0; j < 8 && counter != data.size(); j++) {

temporary[i][j] = data[counter++];

}

}

reset\_grid();

for (int k = 0; k < 4; k++) {

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

if (grid[i][j] && temporary[i][j] != 'x')

result += temporary[i][j];

}

}

rotate(grid);

}

status = 0;

reset\_grid();

}

std::string decode(std::string data) {

result.clear();

std::string block;

for (int i = 0; i < data.size(); i++) {

if (i != 0 && i % 64 == 0) {

read(block);

block.clear();

}

block += data[i];

}

if (!block.empty()) {

status = 3;

read(block);

status = 0;

clear(temporary);

}

reset\_grid();

return result;

}

private:

std::vector<std::vector<bool>> start\_grid;

std::vector<std::vector<char>> temporary;

unsigned short int status = 0;

std::string result;

size\_t quantity;

std::vector<std::vector<bool>> grid;

};

Листинг файла ClassicalMethod.h

#pragma once

#include <msclr/marshal.h>

#include "Cardano.h"

#include "init.h"

namespace MZI3G {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Runtime::InteropServices;

using namespace System::IO;

using namespace System::Text;

using namespace Microsoft::VisualBasic;

/// <summary>

/// Сводка для Cardano

/// </summary>

public ref class ClassicMethods : public System::Windows::Forms::Form

{

public:

ClassicMethods(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

this->before\_tBox->Dock = System::Windows::Forms::DockStyle::None;

this->before\_tBox->Size = System::Drawing::Size(this->Width / 2 - 20, this->Height - 85);

this->before\_tBox->Location = System::Drawing::Point(10, 40);

this->after\_tBox->Dock = System::Windows::Forms::DockStyle::None;

this->after\_tBox->Size = System::Drawing::Size(this->Width / 2 - 25, this->Height - 85);

this->after\_tBox->Location = System::Drawing::Point(this->Width / 2, 40);

openFileDialog1->Filter = "Text files (\*.txt)|\*.txt";

saveFileDialog1->Filter = "Text files (\*.txt)|\*.txt";

openFileDialog1->Title = "Select a File";

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~ClassicMethods()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::MenuStrip^ menuStrip1;

protected:

private: System::Windows::Forms::ToolStripMenuItem^ выбратьФайлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ закрытьПриложениеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ режимРаботыToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ зашифроватьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ расшифроватьToolStripMenuItem;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;

private: System::Windows::Forms::TextBox^ before\_tBox;

private: System::Windows::Forms::TextBox^ after\_tBox;

private: System::Windows::Forms::Label^ after\_label;

private: System::Windows::Forms::Label^ before\_label;

private: System::Windows::Forms::ToolStripMenuItem^ открытьФайлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сохранитВФайлToolStripMenuItem;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->выбратьФайлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->открытьФайлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сохранитВФайлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->режимРаботыToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->зашифроватьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->расшифроватьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->закрытьПриложениеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());

this->before\_tBox = (gcnew System::Windows::Forms::TextBox());

this->after\_tBox = (gcnew System::Windows::Forms::TextBox());

this->after\_label = (gcnew System::Windows::Forms::Label());

this->before\_label = (gcnew System::Windows::Forms::Label());

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// menuStrip1

//

this->menuStrip1->BackColor = System::Drawing::SystemColors::ControlDark;

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->выбратьФайлToolStripMenuItem,

this->режимРаботыToolStripMenuItem, this->закрытьПриложениеToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(1264, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"menuStrip1";

//

// выбратьФайлToolStripMenuItem

//

this->выбратьФайлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->открытьФайлToolStripMenuItem,

this->сохранитВФайлToolStripMenuItem

});

this->выбратьФайлToolStripMenuItem->Name = L"выбратьФайлToolStripMenuItem";

this->выбратьФайлToolStripMenuItem->Size = System::Drawing::Size(48, 20);

this->выбратьФайлToolStripMenuItem->Text = L"Файл";

//

// открытьФайлToolStripMenuItem

//

this->открытьФайлToolStripMenuItem->Name = L"открытьФайлToolStripMenuItem";

this->открытьФайлToolStripMenuItem->Size = System::Drawing::Size(174, 22);

this->открытьФайлToolStripMenuItem->Text = L"Открыть файл";

this->открытьФайлToolStripMenuItem->Click += gcnew System::EventHandler(this, &ClassicMethods::открытьФайлToolStripMenuItem\_Click);

//

// сохранитВФайлToolStripMenuItem

//

this->сохранитВФайлToolStripMenuItem->Name = L"сохранитВФайлToolStripMenuItem";

this->сохранитВФайлToolStripMenuItem->Size = System::Drawing::Size(174, 22);

this->сохранитВФайлToolStripMenuItem->Text = L"Сохранить в файл";

this->сохранитВФайлToolStripMenuItem->Click += gcnew System::EventHandler(this, &ClassicMethods::сохранитВФайлToolStripMenuItem\_Click);

//

// режимРаботыToolStripMenuItem

//

this->режимРаботыToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->зашифроватьToolStripMenuItem,

this->расшифроватьToolStripMenuItem

});

this->режимРаботыToolStripMenuItem->Name = L"режимРаботыToolStripMenuItem";

this->режимРаботыToolStripMenuItem->Size = System::Drawing::Size(101, 20);

this->режимРаботыToolStripMenuItem->Text = L"Режим работы";

//

// зашифроватьToolStripMenuItem

//

this->зашифроватьToolStripMenuItem->Name = L"зашифроватьToolStripMenuItem";

this->зашифроватьToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->зашифроватьToolStripMenuItem->Text = L"Зашифровать";

this->зашифроватьToolStripMenuItem->Click += gcnew System::EventHandler(this, &ClassicMethods::зашифроватьToolStripMenuItem\_Click);

//

// расшифроватьToolStripMenuItem

//

this->расшифроватьToolStripMenuItem->Name = L"расшифроватьToolStripMenuItem";

this->расшифроватьToolStripMenuItem->Size = System::Drawing::Size(180, 22);

this->расшифроватьToolStripMenuItem->Text = L"Расшифровать";

this->расшифроватьToolStripMenuItem->Click += gcnew System::EventHandler(this, &ClassicMethods::расшифроватьToolStripMenuItem\_Click);

//

// закрытьПриложениеToolStripMenuItem

//

this->закрытьПриложениеToolStripMenuItem->Name = L"закрытьПриложениеToolStripMenuItem";

this->закрытьПриложениеToolStripMenuItem->Size = System::Drawing::Size(138, 20);

this->закрытьПриложениеToolStripMenuItem->Text = L"Закрыть приложение";

this->закрытьПриложениеToolStripMenuItem->Click += gcnew System::EventHandler(this, &ClassicMethods::закрытьПриложениеToolStripMenuItem\_Click);

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// before\_tBox

//

this->before\_tBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->before\_tBox->Location = System::Drawing::Point(12, 39);

this->before\_tBox->Multiline = true;

this->before\_tBox->Name = L"before\_tBox";

this->before\_tBox->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->before\_tBox->Size = System::Drawing::Size(621, 630);

this->before\_tBox->TabIndex = 1;

//

// after\_tBox

//

this->after\_tBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->after\_tBox->Location = System::Drawing::Point(639, 39);

this->after\_tBox->Multiline = true;

this->after\_tBox->Name = L"after\_tBox";

this->after\_tBox->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->after\_tBox->Size = System::Drawing::Size(613, 630);

this->after\_tBox->TabIndex = 2;

//

// after\_label

//

this->after\_label->AutoSize = true;

this->after\_label->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->after\_label->Location = System::Drawing::Point(639, 26);

this->after\_label->Name = L"after\_label";

this->after\_label->Size = System::Drawing::Size(0, 20);

this->after\_label->TabIndex = 3;

//

// before\_label

//

this->before\_label->AutoSize = true;

this->before\_label->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->before\_label->Location = System::Drawing::Point(11, 26);

this->before\_label->Name = L"before\_label";

this->before\_label->Size = System::Drawing::Size(0, 20);

this->before\_label->TabIndex = 4;

//

// ClassicMethods

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1264, 681);

this->Controls->Add(this->before\_label);

this->Controls->Add(this->after\_label);

this->Controls->Add(this->after\_tBox);

this->Controls->Add(this->before\_tBox);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->MaximumSize = System::Drawing::Size(1920, 1080);

this->MinimumSize = System::Drawing::Size(1280, 720);

this->Name = L"ClassicMethods";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Cardano";

this->WindowState = System::Windows::Forms::FormWindowState::Maximized;

this->Resize += gcnew System::EventHandler(this, &ClassicMethods::Cardano\_Resize);

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void зашифроватьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (before\_tBox->Text == "") {

MessageBox::Show("Поле ввода текста пустое", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

else {

before\_label->Text = "Открытый текст";

after\_label->Text = "Зашифрованный текст";

after\_tBox->Clear();

after\_tBox->Text = gcnew String(init(0, msclr::interop::marshal\_as<std::string>(before\_tBox->Text)).c\_str());

}

}

private: System::Void расшифроватьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (before\_tBox->Text == "") {

MessageBox::Show("Поле ввода текста пустое", "Предупреждение", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

else {

before\_label->Text = "Зашифрованный текст";

after\_label->Text = "Открытый текст";

after\_tBox->Clear();

after\_tBox->Text = gcnew String(init(1, msclr::interop::marshal\_as<std::string>(before\_tBox->Text)).c\_str());

}

}

private: System::Void открытьФайлToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

String^ filepath = openFileDialog1->FileName;

try {

StreamReader^ reader = gcnew StreamReader(openFileDialog1->FileName);

before\_tBox->Text = reader->ReadToEnd();

reader->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Ошибка при чтении файла: " + ex->Message);

}

}

}

private: System::Void сохранитВФайлToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

SaveFileDialog^ saveFileDialog = gcnew SaveFileDialog();

saveFileDialog->Filter = "Text files (\*.txt)|\*.txt";

saveFileDialog->Title = "Save a File";

if (saveFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

String^ filePath = saveFileDialog->FileName;

try {

StreamWriter^ writer = gcnew StreamWriter(filePath);

writer->Write(after\_tBox->Text);

writer->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Ошибка при записи в файл: " + ex->Message, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

}

private: System::Void закрытьПриложениеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Windows::Forms::DialogResult temp = MessageBox::Show("Вы уверены, что хотите закрыть приложение?", "Предупреждение", MessageBoxButtons::OKCancel, MessageBoxIcon::Warning);

if (temp == System::Windows::Forms::DialogResult::OK) {

this->Close();

}

}

private: System::Void Cardano\_Resize(System::Object^ sender, System::EventArgs^ e) {

this->before\_tBox->Size = System::Drawing::Size(this->Width / 2 - 20, this->Height - 100);

this->before\_tBox->Location = System::Drawing::Point(10, 55);

this->before\_label->Location = System::Drawing::Point(10, 30);

this->after\_tBox->Size = System::Drawing::Size(this->Width / 2 - 25, this->Height - 100);

this->after\_tBox->Location = System::Drawing::Point(this->Width / 2, 55);

this->after\_label->Location = System::Drawing::Point(this->Width / 2, 30);

}

};

}

Листинг файла init.cpp

#pragma once

#include <bitset>

#include <vector>

#include <string>

std::string init(char mode, std::string data) {

std::vector<std::vector<bool>> grid = { { 0, 1, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 1, 0, 0, 0 },

{ 1, 0, 1, 0, 0, 0, 1, 0 },

{ 0, 0, 0, 1, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 0, 1, 0, 0 },

{ 1, 0, 0, 0, 1, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 0, 1, 0 },

{ 0, 0, 0, 1, 0, 0, 0, 1 } };

Cardano Cardano(grid, 16);

switch (mode) {

case 0: return Cardano.encode(data); break;

case 1: return Cardano.decode(data); break;

}

}

int bin\_to\_int(std::string seed) {

std::bitset<8> b(seed);

return b.to\_ulong();

}