МІНІСТЕРСТВО ОСВІТИ, НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

СИМЕТРИЧНА КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3
Криптоаналіз афінної біграмної підстановки

Підготували:
студенти 3 курсу
групи ФІ-84
Ковальчук Ольга Миронівна
Коломієць Андрій Юрійович

Викладач:

Київ – 2021

Криптоаналіз афінної біграмної підстановки

Мета роботи

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Порядок виконання роботи

1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв'язуванням лінійних порівнянь. При розв'язуванні порівнянь потрібно коректно обробляти випадок із декількома розв'язками, повертаючи їх усі.
2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп'ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).
3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п'яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ шляхом розв'язання системи конгуренцій.
4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.
5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

Результати роботи програми

П'ятьма найчастішими біграмами російської мови (в порядку спадання частот) є біграми «ст», «но», «то», «на», «ен».

Olga Kovalchuk

Variant#3

Process sorting statistics bigram…

-bigram: тд equel 77;

-bigram: рб equel 53;

-bigram: во equel 52;

-bigram: щю equel 45;

-bigram: кд equel 42;

Text:
отцеубийствокакизвестнооснованоеиизначалыноягрестнглениечеловечестваиотделыноц
очеловекавовсякомслучаеонфплавньйисточникчувствавиньнеизвестноединственньйлии
сследованиямнеудалосыещеустановитыдушевноепроисхыждениевиньипотребностиискнгл
енияноотнюдынесзщественноединствебньйлиэтоисточнидгсихологическоеположениесло
жноинуждаетсявобясненияхотношениемалычикакоткукакмвповоримамбивалентнопомимон
енавистииззакоторойхотелосыбьотцакаксигерникаустранитысзществуетобьчноншкотор
аядолянежностикнемуобаотношениясливаютсявидентификациюсотцомхотелосыбьзанятым
естоотцщготомучтоонвьзьваетвосхищениехотелосыббьтыканипотомучтохочетсялпоу
странитывсеэтонаталкиваетсянакрупноепрягтствиевигределебньймоментребенокначи
наемгониматычтопопьткаустранитыотцакаксигерникавстретилабьсостороньотцанаказа
ниечерезкастрациюизстрахакастрациитоестывинтересахсовранениясвоеймужественнос
тиребенокотказьваетсяотжеланияобладатыматерыюиотустраненияотцщгосколыцуэтыжел

аниеостаетсявобластибессознателыноцоонояоляетсяосновойдляобразованиячувствави
ньнамкажетсячтомьигисалинормалыньягроцессьобьчнуюсудыбутакназьваемоцоэдвговак
омплшкюаследуетоднаковнестиважноедигвлнениевозникаютдалынейшиеосложненияеслиу
ребенкасилынееразвитконституциобньйфакторназьваемьйнамибиееексуалыностыютогдщг
одщпрозогготеримужественностфчерезкастрациюукрягляетсятенденцияуклонитысявсто
ронуженствебностиболеетфпотенденцияпоставитысебянаместоматеривгеренятыеервлык
акобшкталюбвиотцаодналишзбоязныкастрацииделаетэтуразвязкуневозможнойребенодго
нимаетчтоондолженвзятынасебяикастрированисеслионхочетбьтылюбимьмотцомкакженжи
натакобршкаютсянавьтеснениеобапорьваненавистыкоткуивлюбленностывотцаизвестнаш
гсихологическаяразницаусматриваетсявтомчтоотненавистикотцуотказьваютсявследст
виестрахщгередвнешнейопасностыюкастрацией влюбленностожевотцаволгринимаетсякак
внутренняяопасностыпеэвфчноцопозьвакоторашгосутисвоейсновавозхращаетсяктойжев
нешнейигасностистрахпередотцамделаетненавистыкоткунеприемлемойкастрацияужасна
каквкачествшкарьтакиценьлюбвиизобоихфакторовввьтесняющихненавистыкоткупеэвьйне
посредствебньйстрахнаказанияикастрацииследуетназватынормалыньмпатогеническоеу
силениепривноситсякаккажетсялишыдругиафакторомбоязныюженственнойустановкиярко
вьражебнаябиееексуалынаясклонностыстановитсятакимобразомоднимизусловийилиподтв
ерждений неврозаэтусклонностыочевидноследуетпризнатыиудостоевскоцоионалатентна
епомосшксуалыностыпроояоляетсявдозввленномвидевтомзначениикакоеимелавлпыжизнид
ружбасмужчинамивецодострабностинежномотношенииксоперникамолюбвиивлпигрекрасно
тгониманииположенийобяснимьхлишывьтесненноппомосшксуалыностыюкакнаэтоуказьваю
тмнфпочислебньепримерьизецопроизведенийсожалеюнонфчлпонемфпуизменитыеелвгодро
нностионенавистиилюбвикотцуиобихвидоизмененияхподвлияниемугрозькастрациинесве
дзщемувпсихоанализечитателюпокажутсябезвкусньмиималовероятньмвгреюгвлжпаючтои
мебнокомплшкскастрациибудетотклоненсилынсевслпоносмеюуверитычтигсихоаналитфче
скийопьтставитимебноэтиявленияврневсякфпосомненияинаходитвнихключклюбомунехроз
уилгьтаемжсецовслучаетакназьваемойцгилягсиинашлпигисателянонашемусознаниютакч
уждьтеяоленияввооластикоторьхнаходитсянашабессознателынашгсихическаящизныуказа
бньмвьшенеисчекгьваютсявэдвговомкотглексягоследствиявьтесненияненавистикоткун
овьмяоляетсяточтовконцеконцовотыждестолениесотцомзавоевьваетвнашемяпостоябное
местоэтоотыждестолениеволгринимаетсянашимянопредставляетсобойвнемособуюинстан
циюпротивостоящуюосталыномусодержаниюнашецоямьназьваемтфпдаэтуинстанциюнашимс
верхяипрвгисьваемейнаследницеродителыскоцоолияниянаиважнейшиефункцииеслиотецб
ьчсуровнасилыственжестокнашесверхшгеренимаетотнлпоэтикачестваивлпоотношениикя
сновавозникаетпассивностыкоторойкакразнадлежалобьбьтывьтеснебнойсверхясталоса
дистическимястановитсямазохистскимтоестывосновесвоейженствебнигассивньмвнашем
явозникаетболышашготренностывнаказанииияотчастиотдаетсебякактаковоевралгоряже
ниесудыбьотчастиженаходитудовлетворениевжестокомобращенииснимсверхясознаниеви
нькаждаякараявляетсяведывосновесвоейкастрациейикактаковаяосуществолениемизнача
лынфпигассивнфпоотношениякотцуисудыбавконцеконцовлишыдалынейшаяпроекцияотцано
рмалыньеявленишгроисходяжиягриформированиисовестидвлжнхгоходитынаопиюабньезде
сыанормалыньенамещенеудалосыустановитыразгранфчениямеждунимизамечаетсячтонаиб
влышаяролыздесывконечномитфпягрипись ваетсяпассивньмэлементамвьтесненнойженств
ебностиищекакслучайньйфакторимсетзначениеяоляетсяливнушающийстрахотецивдейст
вителыностиособебнонасилыственньмэтоотносится кдостоевскомуфактлпоисключителын
оцочувствавиньравнокакимазохистскфпообразажизнимьсводимкецоособенноярковьраже
бномукомпонентуженствебностидостоевскоцоможноопределитыследующимобразомособеб
носилынаябиееексуалынашгредрасположебностыиспосонностысособойсилойзажищатысяот
зависимостиотчрезвьчайносуровоцоотцаэтотхарактербиееексуалыностимпдобавляемкра
нсеузнабньмкотгонентамецосуществарабнийситгтотгрипадковсмертиможнорассматрива
тыкакотождествлениесволпоясотцомдопущебноевкачественаказаниясостороньсверхять
захотелубитыотцадабьстатыотгомсамомутягерытьотецноотецмертвьйобьчньймеханизми
стерфческихсимптомовиктомужетеперытебяубиваетотецдлянашецоясимптомсмертиявляе
тсяудовлетворениеафантазиимужскфпыжеланияиодновремебномазохистскитгосредством
наказанияторстысадистическимудовлетворениемобаяисверхяопраютролыотцаидалышево
бщемотношениемеждулфчностыюиобектомотцаприсохраненииецосодержанияперешловотно
шениемеждуяисверхяноваяинсценировканавторойсценетакиеинфантилыньереакцииэдвго
вакомплшкюамогутзжплохнутыеслидействительностынедаетимвдалынейшемпижинохаракт
еротцаостаетсятемжесамьмнетонухудшаетсясцодамитакимобразомпродвлжаетоставатыс
яиненавистыдостоевскоцокотцужеланиесмертиэтомузломуоткустановитсяигасньмеслит
акиевьтесненньежеланияосзществляютсянаделефантазиясталареалыностыювсемерьзащи
тьтягерыа

Key: (199,700)

Andrew Kolomiets

Variant#4

Process sorting statistics bigram…

-bigram: еш equel 67;

-bigram: еы equel 49;

-bigram: ск equel 47;

-bigram: шя equel 47;

-bigram: до equel 46;

Text:
еслиправдачтодостоевскийвсибирщебьлподверженприпадкммтоэтолишьподтверждаетто
чтоегоприпадкибьлиегокаройонболеерщихненуждалсякогдабьлкараемцщьмобразоыщодок
узатыэтонехозможноскорееэтойнеобводимостзюрщакузощфидляпсихическойэкэщомфидос
тоевскогообясняетсяточтоонпрошещщесломлущньмчерезэтфгодьбедствийивщижущийосуж
дениедостоевскоговкачествеполитическогопреступникабьлонесправедливьмиондолжен
бьлэтознатыноэщпрцщллэтэщезаслужеьщоенаказаниеожбатюшкицарякакзммущвщакузощия
заслужущногоимзасвойгрехпоотношениюксвоемусобствущномуотцувместоюамонаказания
ондалсебянаказатызаместителюотцаэтодаетнаыщекотороепредставлениеопсихологичес
комоправданицщакузощийприсуждаемьхобществхомэтонаюамомделетакыщогиеизпрестующи
ковжаждутнаказанияеготребуетихсверхяизбавляясебятакимобразомотсммэщакузощиято
тктохщаетслофщоефизмущчихоезначениеистерическихсимптомовпойметчтомьздесынепьт
аемсядобитысясмьслаприпадковвдостоевскогововсейполнотедостаточнотогочтоможнопр
едположитычтоихперхоначалынаясенностыосталасйщеизмущночщесморянавсепоследупни
енаслоенияможноскузатычтодостоевскийтакникогдаинеосхободилсяотугрьзущийсовест
ивсвязиснамерениемубитыотцаэтолежащеущасовестибремяопределилотакжеегообщышущи
екдоумдругишсферампокоющишснщаобщышущфикотцукгосударствущномуавторитетуиквере
вбогавпервойэщпришелкполномуподчущущиюбатюшкшцарюотщаждьразьгравшемуащимкомед
июубийствавдействителйщостинаводившуюстолыкорузотражениевегоприпадкахздесывер
хвзялопоканщиеболышесвободьоставалосыунегохобластирелигиознойпэщедопускапнимс
оыщущийсведущиямондопослетщейминутьсвоейжизнивсеколебалсямеждзверойибезбожием
еговьсокийумнепоуволллемвщезммечатытетрутщостиосмьсливощияцкоторьмприводитвер
авиндивидуалыномповторениимирового гоисторического раувития эщнаделлсявидеалехрист
ощайтивьводиосхобождущиеотгреховфисполызоватысхоисобствеьщьестраданиячтобьпри
тязатйщаролыхристаеслионвконежщомсчетущепришелксвободеисталреакциэщеромтоэтоо
бяащяетсятемчтоотнечеловеческаяскщовняявцщощакоторойстроитсярелигиозноечзвстх
одостфглавщегосверхцщдивидуалйщойсильцщемоглабьтыпреодоленадажеегоьсокойцщте
ялектуалйщостыюздесынаскузалосыбьможноупрешщутывтомчтомьотказьваемсяотбесприс
трастностипсивоощализаиподвергаемдостоевскогооцущюеимеющейпрахонасущестховани
елишыспристрасбщойточкизренияопределущногомирохоззрениякэщсерваторсталбьнаточ
казрениявеликогоинквизитораиоценивалбьдостоевскогоиначеупрексправедливдляегос
мягчущиямофщолишысказатычторешущиедостоевскоговьувощоочевитщозатрутщущностзюе
гомьшлениявследствиущеврозаедвалипростойслучачщостыюможнообяснитычтотришедевр
ммировойлитературьвсехвремущтрактуютоднуитужетемутемуотцеубийствацарыэдипсофо
клагамлетшекспираибратыякарммузовьдостоевскогововсехтрехраскрьваетсяимотивдея
ниясексуалйщоесопеящичествоиззажущщиньпрямеевсегокэщечноэтопредставлущовдрмме
оащовощночщагреческомскузощфиздесыдеяниесовершаетсяцнесммимгероемнобезсмягчущ
ияизавуалированияпоэтическаяобработкощевозмофщаоткровеьщоепризнаниевнамерении
убитыотцакакогомьдобиваемсяприпсивоощализекажетсянепереносимьмбезощалитическо
йподготовкивгреческоэдраменеобводимоесмягчущиеприсохранениисенностимастерскид
остфгаетсятемчтобессохщателйщьймотивгерояпроецируетсявдействителыностыкакчужд
оеемупринуждущиенавязаьщоесурыбойгеройсовершаетдеяниущепретщммерущноиповсейви
димостибеувлинщияжеянцщьивсежеэтостечущиеобстоятелыствпринимаетсяврасчеттакка
конможетзахоеватыцарицуматытолыкопослеповторущиятогожедействияхобщышущфичудов
ищасимволизирующегоотцапослетогокакобнаруживаетсяиоглашаетсяеговинанеделается
никакихпопьтокащятыееессебявзвалитыеенапрцщуждениесостороньсурыбьнаоборотвинап
ризнаетсяикаквсщелаявинанаказьваетсячторассудкуможетпоказатысянесправедливьм
нопсихологическиабсолюбщоправилыновангличскоэдрамеэтоизображущоболеекосвущноп

оступоксовершаетснщесммимгероемадругимдлякоторогоэтотпоступокнеявляетсяотцеуб
ийстхомпоэтомупредосудителйщьймотивсексуалйщогосоперничествaужущщиньненуждает
сявзаоуалировощфирарщоийдиповкомплексгероямьвидимкакбьвотражеьщомсветеtаккакм
ьвидимлишытокакоедействиепроизходибщагерояпоступокдругогоондолженбьлбьзаэтотп
оступокотомститйщострощньмобрузомневсилахэтосделатымьхщаемчтоегорасслабляетсо
бствущноечзвстховиньвсоответствиисхарактеромневротическихявлущийпроисходитсдв
игичувствовцщьпереходитхосохщощиесвоечщеспосогщостивьпощщитыэтозаданиепоявляю
тсяприхщакитогочтогеройвоспрцщимаетэтувинукаксверхцщдирщдуалйщуюэщпрезираетдр
угидщемущеечешсебяеслиобходитысяскаждьмпозаслугммктвуэдетотпоркивэтомнаправле
нииромощрусскогопиюателяуходитнашагдалошеиздесыубийствосовершенодругимчеловек
омотщакочеловекомсвязощньшсубитьмтакимижееьнорщимиобщышущиямикакигероэдмитрий
укоторогомотивсексуалыногосопеящичествaоткровеьщоприхщаетсясовершенодругимбра
томкоторомукакибтереснозаметитыдостоевскийпередалсвоюсобствущнуюболезныякобьэ
пилепсиютешсммьмкакбьжелаясделатыприхщощиечтомолэпилептишщевротиквомнеотцеуби
йцаивотвречизснитниканасудетажеизвестнанщасмешкощадпсихологиейонамолпалкаодоу
хкэщцахзавуалировановеликолеющотаккакстоитвсеэтоперевеящутыинаводишыглубочайш
уюсенностывосприятиядостоевскогозаслуживаетнасмешкиотнюрынепсихологиясудебнь
йпроцессдохщощиясовершущнобезрузлижщоктоэтотпоступоксовершилнаюамомделепсивол
огияибтересуетсллишытемктоеговсхоемсердцежелаликтопоегосовершущфиегоприветств
овалипоэтомувплотыдокобтрастнойффгурьалешивсебратыяравновинорщьдвижимьйпервиж
щьмипозьвммфискателйщаслаждущийпощщьйскепсиюациникиэпилептическийпрестующиквб
ратыяхкарамазовьхестысценаввьсшейстепенихарактернаядлядостоевскогоизразговора
сдмитриемстаршцпостфгаетчтодмитричщоситвсебеготовностыкотцеубийстоуиброюается
перетщимнаколениэтэщеможетявлятысявьражениемхосхщнущияадолфщоохщачатычтосвято
йотстраняетотсебяискушениеисполнитысяпрезрениемкубийцеилиимпосщушатысяипоэтом
упереднишсмиряетсясимпатиядостоевскогокпреступникудействителйщобезгрощичнаэща
далековьходитзапредельсострадощиянакотороущесчастньйимеетправоонанапоминаетбл
агоговущиескоторьмвдрерщостиотносилисыкэпилептикуидушерщоболйщомупреступникдл
янегопочтиспасителывзявшичщасебявцщукоторуювдругошслучаущеслибьдругиеаа

Key: (390,10)

В якості розпізнавача мови в програми реалізовано дві функції:

- перша функція використовує індекс відповідності мови;
- друга функція виконує розпізнавання тексту шляхом підрахунку заборонених біграм, котрі спостерігаються в мові.

Висновок

Після виконання комп’ютерного практикуму, ми набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки, опанували прийомами роботи в модулярній арифметиці. В якості розпізнавача мови, тобто змістовного тексту, ми використовували метод заборонених біграм, та індекс відповідності мови. Кожен метод працює по різному, за допомогою індексу відповідності текст можна віднайти відразу, в порівнянні з методом заборонених біграм, котрий потребує декількох переборів кандидатів на змістовний текст.

Код програми

```
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>
#include <Windows.h>
#include <cmath>
#include <iomanip>
#include <ctime>
#include <cctype>
#include <vector>

using namespace std;

#define type int

/// function for work with bigrams block
start//////////////////////////////////////////////////////////////////////////////////////////

struct bigram_tools //ok
{
	string bigram;
	type value;
```

```
};

struct key//ok
{
	type first;
	type second;
	bool system_solution;
};

struct possible_permutation//ok
{
	bigram_tools first_;
	bigram_tools second_;
};


string remove_all_spaces(string str) //ok
{
	str.erase(remove(str.begin(), str.end(), ' '), str.end());
	return str;
}

string filtration_the_text_without_space(string text) //ok
{
```

```
string upper_alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯ";
string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщьыэюя";

for (type i = 0; i < text.length(); i++)
{
	for (type j = 0; j < 31; j++)
	{
		if (text[i] == lower_alphabet[j])
		{
			text[i] = lower_alphabet[j];
			break;
		}
		if (text[i] == upper_alphabet[j])
		{
			text[i] = lower_alphabet[j];
			break;
		}
		if (text[i] == 'Ё' || text[i] == 'ё')
		{
			text[i] = 'e';
			break;
		}
		if (text[i] == 'Ъ' || text[i] == 'ъ')
		{
```

```
                        text[i] = 'ь';
                        break;
                  }
                  if (j == 30 && text[i] != lower_alphabet[j])
                  {
                        text[i] = ' ';
                        break;
                  }
            }

      }

      return remove_all_spaces(text);
}


type** matrix_frequancy_with_step_two(type** matrix, type row, type column, string text) //ok
{
      string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

      for (type t = 0; t < text.length() - 1; t = t + 2)
      {
            type temp_i = 0;
            type temp_j = 0;
```

```
	for (type i = 0; i < row; i++)
	{
		if (text[t] == lower_alphabet[i])
		{
			temp_i = i;
			break;
		}
	}
	for (type j = 0; j < column; j++)
	{
		if (text[t + 1] == lower_alphabet[j])
		{
			temp_j = j;
			break;
		}
	}

	matrix[temp_i][temp_j] = matrix[temp_i][temp_j] + 1;

	temp_i = 0;
	temp_j = 0;
}
```

```
	return matrix;
}


string get_all_text_in_one_string(ifstream &text_link, string text, string text_string) //ok
{

	text_link.seekg(0, std::ios::beg);

	text = " ";

	while (!text_link.eof())
	{
		getline(text_link, text_string);

		text_string = filtration_the_text_without_space(text_string);

		text = text + text_string;
	}

	text = remove_all_spaces(text);

	return text;
}
```

```
string we_make_the_text_of_even_length(string text)//ok
{

	if (text.length() % 2 != 0)
	{
		text = text.substr(0, text.size() - 1);
	}

	return text;
}



type* copy_vector(type* vector_first, type* vector_second, type size) //ok
{
	for (type i = 0; i < size; i++)
	{
		vector_second[i] = vector_first[i];
	}

	return vector_second;
}

type** copy_matrix(type** matrix_first, type** matrix_second, type size) //ok
```

```
{
	for (type i = 0; i < size; i++)
	{
		for (type j = 0; j < size; j++)
		{
			matrix_second[i][j] = matrix_first[i][j];
		}
	}

	return matrix_second;
}

type* null_vector(type* vector, type size)//ok
{
	for (type i = 0; i < size; i++)
	{
		vector[i] = 0;
	}

	return vector;
}

type** null_matrix(type** matrix, type size)//ok
{
```

```
	for (type i = 0; i < size; i++)
	{
		for (type j = 0; j < size; j++)
		{
			matrix[i][j] = 0;
		}
	}
	return matrix;
}


void bubbleSort(bigram_tools* bigrams_statistics_sorting, type length_array)//ok
{
	type temp_ = 0;
	string _temp;
	bool exit = false;

	while (!exit)
	{
		exit = true;
		for (type int_counter = 0; int_counter < (length_array - 1); int_counter++)

			if (bigrams_statistics_sorting[int_counter].value < bigrams_statistics_sorting[int_counter + 1].value)
			{
```

```
                        temp_ = bigrams_statistics_sorting[int_counter].value;
                        bigrams_statistics_sorting[int_counter].value = bigrams_statistics_sorting[int_counter +
1].value;
                        bigrams_statistics_sorting[int_counter + 1].value = temp_;

                        _temp = bigrams_statistics_sorting[int_counter].bigram;
                        bigrams_statistics_sorting[int_counter].bigram = bigrams_statistics_sorting[int_counter +
1].bigram;
                        bigrams_statistics_sorting[int_counter + 1].bigram = _temp;

                        exit = false;
                  }
      }
}

bigram_tools* statistics_start_initil(bigram_tools* array_sorting_bigram)//ok
{
      string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

      type count = 0;

      for (type i = 0; i < lower_alphabet.length(); i++)
      {
            for (type j = 0; j < lower_alphabet.length(); j++)
```

```
		{
			string temp = " ";

			temp = temp + lower_alphabet[i] + lower_alphabet[j];
			temp = remove_all_spaces(temp);

			array_sorting_bigram[count].bigram = temp;
			array_sorting_bigram[count].value = 0;

			count++;
		}
	}

	return array_sorting_bigram;
}

void output_statistics_bigram(bigram_tools* array_sorting_bigram) //ok
{
	cout << endl << "Process sorting statistics bigram.. " << endl;

	cout << endl;

	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";
```

```
	for (type i = 0; i < 5/*pow(lower_alphabet.length(), 2)*/; i++)
	{
		cout << " \t -bigram: " << array_sorting_bigram[i].bigram << " equel " << array_sorting_bigram[i].value << ";" << endl;
	}
}

bigram_tools* sorting_matrix_in_descending(bigram_tools* array_sorting_bigram, string text)//ok
{
	type count = 0;

	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";


	type** matrix = new type*[lower_alphabet.length()];

	for (type i = 0; i < lower_alphabet.length(); i++)
	{
		matrix[i] = new type[lower_alphabet.length()];
	}

	null_matrix(matrix, lower_alphabet.length());


	matrix = matrix_frequancy_with_step_two(matrix, lower_alphabet.length(), lower_alphabet.length(), text);
```

```
for (type i = 0; i < lower_alphabet.length(); i++)
{
      for (type j = 0; j < lower_alphabet.length(); j++)
      {
            string temp = " ";
            temp = temp + lower_alphabet[i] + lower_alphabet[j];
            temp = remove_all_spaces(temp);

            array_sorting_bigram[count].bigram = temp;
            array_sorting_bigram[count].value = matrix[i][j];

            count++;
      }
}

bubbleSort(array_sorting_bigram, pow(lower_alphabet.length(), 2));



for (type i = 0; i < lower_alphabet.length(); i++)
{
      delete matrix[i];
}
```

```
	delete[] matrix;

	return array_sorting_bigram;
}

/// function for work with bigrams block
end/////////////////////////////////////////////////////////////////////////////////////////////


///algorithmic part block
start//////////////////////////////////////////////////////////////////////////////////////////////////////////


//......................mathematic block function start.....................//

type mod(type number, type module)//ok
{
	type numb_by_module;

	if (number < 0)
	{
		if (number / module != 0)
		{
			numb_by_module = module - (abs(number) - module * (abs(number) / module));
```

```
            }
            else
            {
                  numb_by_module = module - abs(number);
            }

            //numb_by_module = module - abs(number);
      }
      else if (number / module != 0)
      {
            numb_by_module = number - module * (number / module);
      }
      else
      {
            numb_by_module = number;
      }

      //numb_by_module = number;

      return numb_by_module;
}

type gcd(type first_num, type second_num)//ok
{
```

```
type gcd_value;

type r_1 = first_num;
type r_2 = second_num;

if (first_num == 0)
{
      if (second_num == 0)
      {
            return 0;
      }
      return second_num;
}
else if (second_num == 0)
{
      if (first_num == 0)
      {
            return 0;
      }
      return first_num;
}
else
{
      type q_1;
```

```
	type q_2 = -1;

	while (q_2 != 0)
	{
		if (r_1 < r_2)
		{
			type temp;
			temp = r_1;
			r_1 = r_2;
			r_2 = temp;
			gcd(r_1, r_2);
		}
		else
		{
			q_1 = r_1 / r_2;
			q_2 = r_1 - (q_1 * r_2);
			r_1 = r_2;
			r_2 = q_2;
		}
	}
	gcd_value = r_1;

	return gcd_value;
}
```

```
}

type inverted_element(type number, type moduls)//ok
{
	number = mod(number, moduls);

	if (gcd(number, moduls) == 1 )
	{
		type b0 = moduls, t, q;
		type x0 = 0, x1 = 1;
		if (moduls == 1) return 1;
		while (number > 1) {
			q = number / moduls;
			t = moduls, moduls = number % moduls, number = t;
			t = x0, x0 = x1 - q * x0, x1 = t;
		}
		if (x1 < 0) x1 += b0;

		return x1;
	}
	if (gcd(number, moduls) > 1)
	{
		return -1; //code number if roots more or does`t exist
	}
```

```
}

key tools(type first, type second, type system_solution)//ok
{
	key keys;

	keys.first = first;
	keys.second = second;
	keys.system_solution = system_solution;

	return keys;
}

vector<type>  congruence(vector<type> solution, type number_a, type number_b, type module)
{
	number_a = mod(number_a, module);   number_b = mod(number_b, module);

	if (number_a == 0 && number_b == 0)
	{
		solution.push_back(-2);
	}
	if (gcd(number_a, module) == 1 && inverted_element(number_a, module) != -1)  //code -1 seems, that exist invers
element with gcd more than one
	{
	   type inverted_numer_a = inverted_element(number_a, module);
```

```
   solution.push_back(mod(inverted_numer_a * number_b, module));
}
if (number_b % gcd(number_a, module) != 0&& inverted_element(number_a, module) == -1)
{
      solution.push_back(-2);
}
if (number_b % gcd(number_a, module) == 0 && inverted_element(number_a, module) == -1)
{
      type  d = gcd(number_a, module);

      type _number_a = number_a / gcd(number_a, module);
      type _number_b = number_b / gcd(number_a, module);
      type _module = module / gcd(number_a, module);

      type inverted_numer_a = inverted_element(_number_a, _module);

      type temp = mod(inverted_numer_a * _number_b, _module);

      for (type i = 0; i < d; i++)
      {
                  solution.push_back(mod(temp + i * _module, module));
      }
```

```
	}



	return solution;
}

vector<key> solving_system(vector<key> keys, type X_second_most_freq_in_language, type  X_first_most_freq_in_language,
type  Y_second_most_freq_in_ciphertext, type Y_first_most_freq_in_ciphertext, type moduls)
{

	type numb_a_for_congurency = mod((X_first_most_freq_in_language-X_second_most_freq_in_language ), moduls);
	type numb_b_for_congurency = mod((Y_first_most_freq_in_ciphertext-Y_second_most_freq_in_ciphertext ), moduls);

	vector<type> solution;	solution = congruence(solution, numb_a_for_congurency, numb_b_for_congurency, moduls);

	for (type i = 0; i < solution.size(); i++)
	{
		if (solution[i] != -2/*&& solution[i] == 13*/)
		{
			type temp = mod(Y_first_most_freq_in_ciphertext-solution[i] * X_first_most_freq_in_language,
moduls);
			keys.push_back(tools(solution[i], temp, true));
		}

	}
```

```
	return keys;

}

//.....................mathematic block function end.....................//


//.....functionality tools for work with coding and decoding text start.....//

type search_number_representation_of_bigram(char first, char second) //ok
{
	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

	type temp_first, temp_second;


	for (type i = 0; i < lower_alphabet.length(); i++)
	{
		if (first == lower_alphabet[i])
		{
			temp_first = i;
			break;
		}
```

```
	}
	for (type j = 0; j < lower_alphabet.length(); j++)
	{
		if (second == lower_alphabet[j])
		{
			temp_second = j;
			break;
		}

	}

	return temp_first * lower_alphabet.length() + temp_second;
}

string search_symbols_representation_of_bigram(type number_representation_bigrams) //ok
{
	string search_bigram = " ";

	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

	for (type i = 0; i < lower_alphabet.length(); i++)
	{
```

```
		for (type j = 0; j < lower_alphabet.length(); j++)
		{
			if (number_representation_bigrams == i * lower_alphabet.length() + j)
			{
				search_bigram = search_bigram + lower_alphabet[i] + lower_alphabet[j];
				search_bigram = remove_all_spaces(search_bigram);
				break;
			}

		}

	}


	return search_bigram;
}

type* convert_number_representation_of_bigram(string text_in_symbols, type* text_in_number)// ok
{
	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

	for (type t = 0; t < (text_in_symbols.length() / 2); t++)
	{
		text_in_number[t] = search_number_representation_of_bigram(text_in_symbols[2 * t], text_in_symbols[2 * t +
1]);
```

```
	}


	return text_in_number;
}


string convert_string_representation_of_bigram(type* text_in_number, string text_in_symbols, type
size_text_in_number)//ok
{
	text_in_symbols = " ";


	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";


	for (type i = 0; i < size_text_in_number; i++)
	{
		text_in_symbols = text_in_symbols + search_symbols_representation_of_bigram(text_in_number[i]);
	}


	text_in_symbols = remove_all_spaces(text_in_symbols);


	return text_in_symbols;
}


string affine_encryption(string text, key key_)//ok
{
	string cipher_text = " ";
```

```
	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";


	type* cipher_text_number_representation = new type[text.length() / 2]; null_vector(cipher_text_number_representation, text.length() / 2);

	type* text_number_representation = new type[text.length() / 2];		null_vector(text_number_representation, text.length() / 2);


	text_number_representation = convert_number_representation_of_bigram(text, text_number_representation);


	for (type i = 0; i < text.length() / 2; i++)

	{

		type moduls = pow(lower_alphabet.length(), 2);

		cipher_text_number_representation[i] = (key_.first * text_number_representation[i] + key_.second) % moduls;

	}


	cipher_text = remove_all_spaces(convert_string_representation_of_bigram(cipher_text_number_representation, cipher_text, text.length() / 2));


	delete[] text_number_representation;

	delete[] cipher_text_number_representation;


	return cipher_text;

}
```

```
string affine_decryption(string cipher_text, key _key)//ok
{
	string text = " ";

	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

	type* cipher_text_number_representation = new type[cipher_text.length() / 2];
null_vector(cipher_text_number_representation, cipher_text.length() / 2);
	type* text_number_representation = new type[cipher_text.length() / 2];
null_vector(text_number_representation, cipher_text.length() / 2);

	cipher_text_number_representation = convert_number_representation_of_bigram(cipher_text,
cipher_text_number_representation);

	type moduls = pow(lower_alphabet.length(), 2);
	type invert = inverted_element(_key.first, moduls);

	for (type i = 0; i < cipher_text.length() / 2; i++)
	{
		if ((cipher_text_number_representation[i] - _key.second) < 0)
		{
			type temp = (moduls - (_key.second - cipher_text_number_representation[i])) % moduls;
			text_number_representation[i] = (invert * temp) % moduls;
		}
		else
```

```
            {
                  text_number_representation[i] = (invert*(cipher_text_number_representation[i] - _key.second)) % moduls;
            }
      }

      text = remove_all_spaces(convert_string_representation_of_bigram(text_number_representation, text,
cipher_text.length() / 2));

      delete[] text_number_representation;
      delete[] cipher_text_number_representation;

      return text;

}

//.....functionality tools for work with coding and decoding text end...../



//.......................crack key block start........................//

bool check_the_content_of_the_text(string text)//ok
{

      bool verification = false;
```

```
string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

string set_of_forbidden_bigram[] =
{ "ыы","ьы","ыь","ьь","щщ",
   /*"аы","бй","вй","гй","гц",
   "гщ","еы","жй","жф","жх",
   "жщ","жы","жю","зй","зщ",
   "иы","иь","йй","кй","кы",
   "кь","лй","мй","нй","оы",
   "оь","пф","пэ","рй","сй",
   "фж","фз","фй","фх","фц",
   "фш","фщ","фэ","фю","хй",
   "хы","хь","цй","цф","цш",
   "цщ","ць","цю","чй","чф",
   "чщ","чы","чю","шд","шж",
   "шщ","щб","щж","эо","эе",
   "уы","уь","фг","фд","юы"*/
};


type count_errors_combination_in_text = 0;

for (type i = 0; i <5; i++)
{
```

```
            if (text.find(set_of_forbidden_bigram[i]) != std::string::npos)
            {
                  count_errors_combination_in_text++;
            }
      }

      if (count_errors_combination_in_text == 0)
      {
            verification = true;
      }


      return verification;
}

bool check_the_content_of_the_text_modified(string text)
{
      bool verification = false;

      string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

      type * array = new type[lower_alphabet.length()]; null_vector(array, lower_alphabet.length());

      for (type i = 0; i < text.length(); i++)
```

```
{
	for (type j = 0; j < lower_alphabet.length(); j++)
	{
		if (text[i] == lower_alphabet[j])
		{

			array[j]++;
			break;
		}

	}
}

long double N, n = text.length(), index = 0.0;

for (int i = 0; i < lower_alphabet.length(); i++)
{
	N = array[i];
	index += N * (N - 1);
}

index /= (n * (n - 1));

if (index > 0.05&&index < 0.06) //0.0553
```

```
	{
		verification = true;
	}
	else
	{
		verification = false;
	}

	delete[] array;

	return verification;
}

possible_permutation* permutation_all_bigrams_with_each_other_theoretical_and_practice(possible_permutation*
combination_bigram, string* theoretical_statistic, string* practical_statistic, type size_of_array_frequent_bigrams)//ok
{
	type counter = 0;

	cout << endl << "All possible combination of bigrams..." << endl;

	cout << endl << "\t" << "Number" << "\t" << "First " << "\t" << "Code " << "\t" << "Second " << "\t" << "Code " <<
endl;

	for (type i = 0; i < size_of_array_frequent_bigrams; i++)
	{
```

```
            for (type j = 0; j < size_of_array_frequent_bigrams; j++)
            {
                  combination_bigram[counter].first_.bigram = theoretical_statistic[i];
                  combination_bigram[counter].first_.value =
search_number_representation_of_bigram(combination_bigram[counter].first_.bigram[0],
combination_bigram[counter].first_.bigram[1]);

                  combination_bigram[counter].second_.bigram = practical_statistic[j];
                  combination_bigram[counter].second_.value =
search_number_representation_of_bigram(combination_bigram[counter].second_.bigram[0],
combination_bigram[counter].second_.bigram[1]);

                  cout << endl << "\t" << counter << "\t" << combination_bigram[counter].first_.bigram << "\t" <<
combination_bigram[counter].first_.value << "\t" << combination_bigram[counter].second_.bigram << "\t" <<
combination_bigram[counter].second_.value << endl;

                  counter++;
            }
      }

      return  combination_bigram;

}

vector<key> build_all_possible_key(possible_permutation* combination_bigram, vector<key> keys)
{
```

```
	string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";


	for (type i = 0; i < 25; i++)
	{
		for (type j = 0; j < 25; j++)
		{
			keys=solving_system(keys,
(combination_bigram[j]).first_.value,(combination_bigram[i]).first_.value,(combination_bigram[j]).second_.value,(combinat
ion_bigram[i]).second_.value,  pow(lower_alphabet.length(), 2));
		}
	}


	return keys;
}


key choose_text_decryption(string cipher_text, vector<key> keys)//ok
{
	key cracked_key;		cracked_key.first = 0; cracked_key.second = 0; cracked_key.system_solution = true;


	for (type i = 0; i < keys.size(); i++)
	{
		if (keys[i].system_solution == true)
		{
			string text = affine_decryption(cipher_text, keys[i]);
```

```
if (check_the_content_of_the_text_modified(text) && text.length() == cipher_text.length())
{

	cout << endl << "Text: " << text << endl;

	cout << endl << "The text is normal?(yes/no)" << endl;

	string answer;
	cin >> answer;

	if (answer == "yes")
	{
		cracked_key.first = keys[i].first;
		cracked_key.second = keys[i].second;

		break;
	}
	else
	{
		continue;
	}

}
else
```

```
                {
                        continue;
                }



        }
        else
        {
                continue;
        }


    }


    return cracked_key;
}


key crack_key(bigram_tools* theoretical_bigram_sorted, string cipher_text)
{


    string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";


    string theoretical_statistic[5] ={"ст", "но", "то" ,"на",  "ен"};


    string practical_statistic[5] ={ theoretical_bigram_sorted[0].bigram , theoretical_bigram_sorted[1].bigram
,theoretical_bigram_sorted[2].bigram,theoretical_bigram_sorted[3].bigram ,theoretical_bigram_sorted[4].bigram};
```

```
	possible_permutation* combination_bigram = new possible_permutation[pow(5, 2)];



	permutation_all_bigrams_with_each_other_theoretical_and_practice(combination_bigram, theoretical_statistic,
practical_statistic, 5);


	vector<key> keys;  keys = build_all_possible_key(combination_bigram, keys);



	//cout << endl << "All possible keys of systems with bool flugs:" << endl;


	//cout <<endl << "\t" <<"Number"<<"\t"  << "First" << "\t" << "Second" << "\t" << "Flag" << endl;


	//for (type i = 0; i < keys.size(); i++)

	//{

	//		cout << "\t" << i << "\t" << keys[i].first << "\t" << keys[i].second << "\t" << keys[i].system_solution
<< endl;

	//}



	key cracked_key = choose_text_decryption(cipher_text, keys);
```

```
	delete[] combination_bigram;


	return cracked_key;


}


//.....................crack key block end.........................//



///algorithmic part block
end//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////



int main()

{

	SetConsoleCP(1251);			SetConsoleOutputCP(1251);



	ifstream fin_for_decryption_text_1;	string path_for_decryption_text_1 = "text_for_decryption_1.txt";
fin_for_decryption_text_1.open(path_for_decryption_text_1);

	ifstream fin_for_decryption_text_2;	string path_for_decryption_text_2 = "text_for_decryption_2.txt";
fin_for_decryption_text_2.open(path_for_decryption_text_2);


	fstream fout_for_decryption_text;	string _path_for_decryption_text = "decrypted.txt";
fout_for_decryption_text.open(_path_for_decryption_text, ios::out);
```

```
	if (!fin_for_decryption_text_1.is_open() && !fin_for_decryption_text_2.is_open())
	{
		cout << endl << "Error of open files!" << endl;
	}
	else
	{

		cout << endl << "File open seccsesfull." << endl;

		unsigned int start_time = clock();

	//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////

		/// common variables block start

		string lower_alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя";

		string one_string_decryption_text;	string decryption_text;

		/// vommon variables block end

		cout << endl << "Start program..." << endl;
```

```
            {   // block with text decryption  process

                  key key_decryption;   key_decryption.first = 0;  key_decryption.second = 0;

                  decryption_text = we_make_the_text_of_even_length(get_all_text_in_one_string(fin_for_decryption_text_1,
decryption_text, decryption_text));

                  bigram_tools* theoretical_bigram_sorted = new bigram_tools[pow(lower_alphabet.length(), 2)];
statistics_start_initil(theoretical_bigram_sorted);

                  sorting_matrix_in_descending(theoretical_bigram_sorted, decryption_text);

                  output_statistics_bigram(theoretical_bigram_sorted);

                  key_decryption = crack_key(theoretical_bigram_sorted, decryption_text);

                  string _temp = affine_decryption(decryption_text, key_decryption);

                  fout_for_decryption_text << endl << "Text: " << _temp << endl;

                  fout_for_decryption_text << endl << "Key: (" << key_decryption.first << "," << key_decryption.second <<
")" << endl;

                  delete[]  theoretical_bigram_sorted;
```

```
            }

            {   // block with text decryption  process

                  key key_decryption;   key_decryption.first = 0;  key_decryption.second = 0;

                  decryption_text = we_make_the_text_of_even_length(get_all_text_in_one_string(fin_for_decryption_text_2,
decryption_text, decryption_text));

                  bigram_tools* theoretical_bigram_sorted = new bigram_tools[pow(lower_alphabet.length(), 2)];
statistics_start_initil(theoretical_bigram_sorted);

                  sorting_matrix_in_descending(theoretical_bigram_sorted, decryption_text);

                  output_statistics_bigram(theoretical_bigram_sorted);

                  key_decryption = crack_key(theoretical_bigram_sorted, decryption_text);

                  string _temp = affine_decryption(decryption_text, key_decryption);

                  fout_for_decryption_text << endl << "Text: " << _temp << endl;

                  fout_for_decryption_text << endl << "Key: (" << key_decryption.first << "," << key_decryption.second <<
")" << endl;
```

```
                delete[]  theoretical_bigram_sorted;
            }

            cout << endl << "End program..." << endl;

      ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////

            unsigned int end_time = clock();

            unsigned int search_time = end_time - start_time;

            cout << endl << "Time of work: " << search_time << " miliseconds;" << endl;

      }

      fout_for_decryption_text.close();

      fin_for_decryption_text_2.close();
      fin_for_decryption_text_1.close();
      return 0;
}
```