Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №3
з дисципліни
«Криптографія»
на тему:
«Криптоаналіз афінної біграмної підстановки»

Виконали:
студенти 3 курсу ФТІ
групи ФБ-71
Рейценштейн Кирило і Таран Вікторія
Перевірили:
Чорний О.
Савчук М. М.
Завадська Л. О.

Київ 2019

Мета роботи :

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

Порядок виконання роботи:

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.
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.
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.

Критерій відбору ключів:
Проаналізувавши результати 1 лабораторної роботи, ми обрали для трьох найчастіших монограм ліміти частот. Тобто, якщо в розшифрованому тексті за допомогою деякої пари ключів частота якоїсь із літер менша за ліміт частоти цієї літери, то текст не пройшов перевірку і пара ключів не розглядається як шукана.
a - 0.055
e - 0.06
o - 0.085
Частоти набагато менші за статистичні, але вони перевірялися протягом виконання 3 лабораторної роботи.

Результати:
Розшифрування тексту за варіантом 13:
Використавши основи модулярної арифметики, критерій відбору ключів та частотний аналіз, було знайдено таку пару ключів:
a = 99, b = 60
Серед 5 найчастіших біграм, що зустрілися в запропонованому розшифрованому тексті, були біграми но і то, що входять до п'ятірки найчастіших біграм російської мови.

Найчастіші біграми ШТ і відповідні їм біграми ВТ та частоти надані в табличці:

Найчастіші біграми зашифрованого тексту	Відповідні біграми розшифрованого тексту	Частота
аф	но	0.0143395
яф	то	0.0131808
нф	по	0.0115875
эв	ер	0.0110081
дю	ра	0.0108633

Зашифрований текст:

дюэорэдюэорэтнтфоэлкшэунскынайцбюовыюежэмюшыафткьэапжнечсюкэфэгябаейпблрщкбсяфийпкдчаясюлюхэитрэшуафюэмпсьфэпбщзскынафвфюэбэеыыпыфьркэяфщюхэнфмюмфыфтфрюсюьэфээбжооафэьякюфььчтвлэйзцндютккхмфяюкпнюузнонмынюбжалюхэзящпызхбврэвьдоанфхэкюжчеыжфдюжгзкжхяюзимощюяпэомоииаьэвынрихбйюзшюхыэфпкйэвунххяюннубякецякзгевкерйыфафйртфмыпйрчяюыннцнюифмфяйожхыфжюкесзпукоынмфвэкуафвфакжнйьрюэоаизююивыйжфщюесюхфяюяндюткдчлзоэфэфпрбнфьфуфкебывыапщфкзацлоечяфьойьнбрзапубюаеюеориээхэлеуисфцовыюежэпвякафмювэьэафйбаилещыгмхблыххвпьоедкрефкзнфьяэпвьрзхбддзчафкьаулпвэубунжотнхxзешфнфабнцацхбзктэипукьэсфнфxддцяфеющюафэьзавэчкьэубгоуыбсцпхрэвьдрикйлроцдмлегйтьзааихэыошайбпйьиэвмпфндюткяхуклооньцэврфяюбихээооарчмюжпывыжбфгнжотндчьэтфкьхбпюсыххгуяюыфжюяфвфйьцнэпйьцнзвкрнфяюкюынночгунфцншыяьэвкэзяыфжюкезфжнмоякчвнювфгьрзфэфпжкцбрюеющкбзвоывнчыфнфцыфлупмюипкэяьяпжешожхейлюызхбххоэрмэнщпшувщзпюйвонаижзиижвомвтщпацзкюфтыщкьэвфжкцбрюеююуэюфэдзяюлпапаячцхбзктэяьыдтфкюапйндюмулывыжвщпгоонвыбфжнжвлззвфгэньэуфшоыдевшэфиьпдьыдфгжвуыжбйинюгфмпдюгзевлзуьпйевлоригвдтзкриепкззягиюоуыяфхошавэыьтфепроуывыххгвэнрилзбэаняпзцлоыфжюкекьэвмгчвоэрмэнюусюмэьяяюгзгпмфюэбндюткяхщюьяжэмфбэмпжнриывыжйийбаинфукыыдчукгонфилсзяюмюсмцбапюущюкнаишэкээьрзхэеорхфэсзунюэлпбняпзцлошвицвкоэфэеоедкрйшгээшефнфсюбклргоаиубякецедэвстынджюткдчвэжукэийикжбтьххэрюйныдпзйилойфвьдютвсыотьлоепыкуняющпмфяфкумфяюрюфншзнотьрзхэифжкьэпдшзгвэнафрзяфтфсожфзбпкнфэьякхбрунинцрукбчыьчсытндютккямюлыяпоэапфбтнрзюэпоывйьлзпкдччэлыгоммрумйевлоцсоэхвппкющфгбиижоикьэпдеваискынчййихбуфяфьянфжбацкэмпкэлпгвнвщккрцуфпипбэьяафбиртюобыдцэнецжвбыйьгошзнфюмсэщфоббылотьрзхэифрьхбшзубрзчвдццкхээожжшушвщзщюяюугынвыцдчйяфвфжзсьуфсфээхэафбиртюомдэвстлоынрэмюоэйвщкующюцэюоуыырэунпущювззеукэввтэпщфюоякшэбээолзлеыншщаплгшзрюафрчйзузякьэдзрююежпцдкртнжзиинцхээорэьинцхээорэяюггчбэоэщяжушувэыоынвэщфтуяфмпьддюжяжбйинюемяфмпвфыфяпикяфжузляфвэжбйинюзфмпфужпыоынвыфэьяшфмувэвфвэвьтфыфифшоынгонплыюевэдюрддюденюынибевьзщьшзнфцосьрзщьсэюцыпркынвокргоынвыххкйыфтымоындьецафзыефэошзэулысэрююэпяюэпойьецккэнчвпюйвлзжбзаесмвнзщфжбынгыврщюжбйинюсзэулысэвьецафзыиьзчщфуюмигоуыэнунтпгорзяюмютцбреюжнтнгодьяпжажвюовыгоммругфжнмоякюуяьтфаэдзхбддзчлехбевьзайедтфжээошэыфьэцооцйюжяхэыошйоцчэжюапабнцшаэвщфьцжвстзккенчпюкэрчжяшфюцшзцнаивфшужбйинюиичйфпапсюмювфтфеошэубврывщпщмшзыэгьцбрьяпкэнщквртфцошэнфыйюобзубяккйгэсзбэфьнбфэьядерющпрзрюапаяшгикыньзфэпсфниякпумнаигпмфыфдюкпнюиисюмютзяюмюсмцбаппумнунгпзтцднчьилзбэаняпзцломэжырумкйыеспюбэхэ

люнюяпдоврврлохбэннбтьиэгюапжбйинюжцхбрyюзузнфьячоывмффэьякюжпрзрюипяьпйоцэрнбецлыгэшыдююощьшыэнчвнчшокйьцэвюоефнняkнвынжвсюшурпжншэмюапкешзэвьж
идужющыонгоюбавмфхнлолзкенцхбзкечбцсьзауфйбцдйионйилзгинфхэжьхбшзбыжвибгмюхщпшфрощэубрзюэкпнюуфлыыфвяапжнмукэфэгязкяфьорзжкпйщйевлоиэжнбыуфн
ювфнншагогыыднбщфпюлнунйьтфьэкчлыэвщйчвмфдуyюкуьэжолешцуыьидьецшфроэайиьиьиеццикафмюэбзтлолеызрюьжрхпзщьмолеызуфшошыгонэмнлзрююежпжбйинюдорицдчв
ынйинпжзхэсйжвшзлеббгмтэкужнфэрвавнчтвынмпевшзэндюткяхгощюцэююооынмфрзыдлоафкувфыфяптцбреюьэщцчвввккпяювфсюикяфжукювубэдывыххяфоуеоххмюююяндютк
яхаэфэцммбыэявлоафсффпйзяюмюнилзбэаняпзцлоыфяфмпшаунцнэужнонуьврнбеццюйныдпзгпфэкччкцыэвтнмгшзрюручкпйонйкикяфжукюяндюткяхейипцрруапрзрюафипюзц
няпзцлоифжнжвскхбвунфыйюопзубяккйгэобддцникрзяфшглоафмюыфшпапщкыэжншзмпэьунушчвуынбзпрусюжбиктндцлркэгуаэакюющюмкщфсмжвщкиэпйхбпйкэаявэнфбцтьрзрюи
папззжбоцдцичнбпэжндюафбэоэщягощзякьэббщздеююыннчьдюсфмпепкэапсюжбикяугвынжвлзмобкямлзюээмюоххскфгцсдвьдмпюухчеыгэмвшзучцпындмафыфлыякшэтьевь
зюогпжбацкэзбщзлышйпкжвяюзиафвфхэыюшйоцпувьякммхбйисюдюлуфбртлоафхлжндюкээрдюуфеышзжбкужеяюннгпьфжзжаепбнгувпапхинцлеынгубеяфюзбэзящпжцибзгхб
ыдапнюузуьунаипзхбеонбшйлюююпнюцьжвдмнфакзешфнффндюювэффчапаяпббрмюпббраифэкчлпцынпыэьяяюапскыничрытюмфифыюкеуирзсыйимфжнчыщкдчяюмвчыйиццдцяфэрдю
жовыепуфлфвэдюрахгщкжвацйипкявюолефьеюжябаскбзшыьилзяфзмьищкнбунмпяюжпьилгкплейнвышабэхооасюфбзалюхэээооаыфтфыоямяюпмяфяфвфэричфпмюифоотннчыфяф
мпрзрюсыпллплирзсыякяфьянфмюифпргэвфуеифлылзяочвнпубзкдюлпапаяоывыиэяфкэфэрчэоиэыэепыфпйхбэвфгжваиэвпюапыфкрэвьдиэцдчйочбэсоиэюезюыэявлоафгвшзяю
еюаясдхэжнншшаяюеехюгюнювфафыфжюкеакьэвфыфяпапвфыфяпапкэучмышйжврзомяфгодьхблошзапаяоывыиэяфюоиэгбиихпжпжнхбжбацкэкзрюмкпббрсюфпыфщфуфифш
юшвшйаувэвфвфзчсывывфдубеэлхэьняпаэлыгэеюэьдмщпкэзяубйьяенююзгоафчэжнфэыфсогоуыяфмпюущювэыьптчвшфжзжажнньгошаыджниьжвхэепщюцэшфрощэпвякафрзрюафип
уфаясюецафуишзбэлрэврзжбацкэсюмфеоьижвнчдьоцжвлзжббыпйбзафнюапфкпйшууэяпйутйэвуншэвэипфпееяуьфюекмевшйеддюеюаяцжюэяфщпеуьгонеуьгонжбйинюсзяфавщфс
мжвщкжхяфьопйвыэвифнюэожхнбуюгыдхэлыщкдчжзэвввэьооадупюшбддцнсэапыевшэипботюсюгуяювумпккьэтшнцхбжююзгинцлерйтьзажбйинюхпнфбцэнсьяеакынхбшыжэюэа
промошаецфгшуунвэынмгофжнтфююивундюткеэтквэчэлыяпнфэьтфхэбэеюфпяющусывыяфяфейпбврруроиьнбоцаиххунгещюлюбэяпаэвфифэбврйьецяибсэмяфвкщзчыдюмкбэцоаи
жбйинюддневуншатпубюаеюеорзякьэмывынющнбыкркэяфорцсшинбмглориублзжпфээтгэфэкчрюмфюэпоывкйртмбхюэтгэгвдтзкшэепкэзяхппзевлзьзшпафщылпсфкупйхбчымюэбзтл
оафхлжндюкэацкшэнрищфжняюмыуннбшэзежнчыщккбсввецафуюжпукгоюувцхбзкжоружьшзлоюушзапацлпмфмпткифьячэфэуюиплешзжбмохцзпзямфюдртлюнфзняпзцлошйрчкоывукго
шыондюткяхмфчмжвнчсюмюжмхбчйдумзюкицрпжпркынйибфсыдцауцэжтуфшфрощэаэфэсвшзэрнбецонтнщпхьямщзэюьялпкэфэыфпвякафжнлзруапкррзякьэызяфтфнняпзцломюяф
мипйчвщкщцьпзлптцынюхсюбэмпжнцсывыжгмцбапацбреюыьрзхэеоьибфжнмоякчвлзбуунфбднххбкхбдьхбнцбэмывыжниьонноoщздвоцружьецафзыефмпжюеоякучопнюапафщц
зчжпыфщцэаубгокймолехлхэьняпаэжныфжпвэскецчвэреюезоэрмэншыжбацкэсюсюыэеоххгунфцнпэшушвщзмфшвоцчвээхэлеэдцбгюыожхщфызацлоечтшхбшйвричкэюзхбддтфыэпя
бэцошыдюхожхяюжквтфчмфяфуфлмяфуцрзипщюзягилзбэаняпзцложзкрякйиифэбврнбддцнсэмэюкацьинчжпжнхбжбацкэвэмэюкацьинчдюшщжвнчвэйбаищпакщцюакнпйвыюееюю
аябкхбуьецыддюьцапэожхгуумэнжхтййылыеюеоххмфеомгщпжбйинюбинцхээойьйьяемкмфпвсьэвдцлолерйафычрмшзэвьжикшзрфдюжгзкжхяюзимощюяпкзюэеыбсцпрйэвуна
игунфцншыыжфщпоедшызаскхбруэйоцжвнорхбрэвлзфбнвнчубйьжвкйшвшйдюукпйонстнфьямпнффкикяфжукюфьфэвбоневнощщюбкоывпюйввтлрдчнфхфкпвэжбунцюдюпупункцыэв
воуывывэдцщзйзйфмфщцэнуьяпкэийэныдххубгишычкиэкуфбнвлжюкбпчоюоуыдцэнсьхбноофмфщцсэляыфегтнпцбркеяфьянфмфшулюдоецбреюгзяфавсюдужющыыпузафипкэс
мяфжукэийикжбцбеддмгэмюьярнфэьядевфзчкптkцбаидьунаимпьнецхглонцвроцсэйззмяфнфбцэнпйкэаяжэбндюткяхафвэмнсьхбэпиисnкньифэкчрюыэмэцовраиафвфхэыочйед
дняляфсзхбддзчафжцнбмгщплыэвгыгокйжвщкяхфэьяденютфцощзрювэмнвыиэтнтфрзловсапзпщфупапюэхохху брзвэлянчгэпвюосьецххлпнфкоцюрзрюжянчйнсьуфчурнафкьрзбэ
апвэубиилоафпзяерпляяюйркэяфщюхэнфоужяпвчтафчэбллпуклоонжбйинюупипцрцдяужбловэчуапгесюрптфнфэьтфчйкэыэуцмбяклежпзчщюжпгвэнмоякрчьфипдвлоякйнвтюос
ьхбнцбэмывысюьоыннчеозксюпгкйзяпзяерплямюифщфгйсюнфеечймущппвякафмюлпяюрнпюдепвякафмфяюлпйзузйбюаунаикэнчяфугевчйлезддмгонфжзуфшопйэнфэцпсыюеьягиюю
шйцбддэвстйшдчубддэвстйшдчжнфэнфмюпвякафяюгуьэьяпвякафлюхэзяяючлхэьняпаэлыдчжузчпвякафвфдузюгиэвмпгиевгывэлплзаэчоынмюуклоонукгонфзндюткяхкйртдьоня
фьорзфщгогзецякххйьпзяюхпшоякуьецмплюмьецафзыспшоякгынмвэйзузгонювфгьйьяеюэйзыфьиэубэткьэщюьяжэацжвстсьрзыфьимфрюлолешщикафранбгололеббевьзйиаывыххяфс
киэкьунсээоххщюшорзсфтфвээоххщэапмфпвяубпсюмфвффчииочизубрзчвпюйвхчхожфчкынчвшушвщзузафмфвффчиибю

Розшифрований текст:

раннеераннееутропервыеотсветызаринакрышезаокномвселистьянадеревьяхвздрагиваютотзываясьнамалейшеедуновениепредрассветноговетеркаивотгдетодалекоиззаповоротанасе
ребряныхрельсахпоявляетсятрамвайпокачиваясьначетырехмаленькихсероголубыхколесахяркооранжевыйкакмандаринnанемэполетымерцающеймедиизолотойкантпроводовижелт
ыйзвонокгромкозвякаетедвапотопныйвожатыйстукнетпонемуногойвстоптанномбашмакецифрынабокахтрамваяиспередияркозолотыеклимонсиденьяточнопорослипрохл
адnымзеленыммхомнакрышесловнозанесеногромныйкучерскойбичнабегуонскользитпосеребрянойпаутинепротянутойвысокосредидеревьевизвсехоконбудтоладаномпахнетвсепрон
икающимголубымизагадочнымзапахомлетнихгрозимолнийтрамвайзвенитвдольокаймленныхвязамиулициобтянутаясеройперчаткойрукавожатогоопятьиопятьлегкокасаетсярукоято
кволденьвожатыйостановилвагонпосредикварталаивысунулсявокошкоэйизавидевпризывныйвзмахсеройперчаткидугласчарлитомвсемальчишкиидевчонкивсегокварталаку
баремскатилисьсдеревьевпобросаливтравускакалкионитакиосталисьлежатьсловнобелыезмеиипобежаликтрамваюонирасселисьпозеленымплюшевымсиденьяминиктоснихн
еспросилникакойплатымистертридденвожатыйположилперчаткунащелькассыиповелтрамвайдальшепотенистымулицамгромкозвякаязвонкомэйсказалчарликудаэтомыедемпо
следнийрейсответилтридденглядявпереднабегущиевысоконадвагономпроводабольшетрамваянебудетзавтрапойдетавтобусаменяотправляютнапенсиюВоткакипотомупокатайтесьн
апоследоквсембесплатноосторожноонрывкомповернулмеднуюрукояткутрамвайзаскрипеликрутосвернулописываябесконечнуюзеленуюпетлюисамовремянавсембеломсветеза
мерлотольkотридденидетиплыливегоудивительноймашинекудатодалекопонесконочаемойрекенапоследокпереспросилудивленныйдугласдакакжетакибезтоговсеплохозеленоймаш
иныбольшенетеезаперливгаражеиникакееоттуданевызволишьимоиновыетеннисныетуфлиужестановятсясовсемстарымиибегутвсемедленнееимедленнеекакжеятеперьбудунет
нетнемогутониубратьтрамвайчтонироворавтобусэтонетрамвайонишумитнетакрельсовунегонетпроводовнетониискрынеразбрасываетирельсыпескомнепосыпаетдаицветунегонетако
йзвонканетиподножкуоннеспускаетаведьверноподхватилчарлистрахлюблюсмотретькогдатрамвайспускаетподножкупрямогармоникатотоионосказалдугластутониприехалинакон
ечнуюостановкувпрочемсеребряныерельсызаброшенныевосемнадцатьлетназадбежалисредихолмовдальшевтысячадевятьсотдесятомгодутрамваемездилиназагородныепро
гулкивчесменпаркприхватывогромныекорзиныспровизиейстехпоррельсытакиосталисьржаветьсредихолмовтуттомыиповорачиваемназадсказалчарлитуттотыиошибсяимисте
ртридденщелкнулвыключателемаварийногогенератораповехалитрамвайдернулсяскользнулпорельсамиоставивпозадигородскиеокраиныпокатилсявнизвдолинуонтовылеталнаду
шистыезалитыесолнцемлужайкитонырялподтенистыедеревьягдепахлогрибамитамисямколеюпересекалиручейкисолнцепросвечивалосквозьлиствудеревьевточносквозьзеленоес
текловагонтихонькобормочачтотопросебяскользилполугамусеяннымдикимиподсолнухамимимодавнозаброшенныхстанцийусыпанныхсловноконфеттистарымитрамвайнымибил
етамиивследзалеснымручьемустремлялсявлетниелесатрамвайондажепахнетпоособенномуговорилдугласездилявчикагонаавтобусахутехкакойточуднойзапахтрамвайчересчурме
дленноходитсказалмистертридденвотониихотятпуститьпогородуавтобусыиребятвшколутожестанутвозитьвавтобусахтрамвайвзвизгнулиостановилсятриддендосталсверхукор
зинуспровизиейребятавосторженнозавопилиивместеснимпотащиликорзинунатравутудагдеручейвпадалвмолчаливоеозероздесьнекогдапоставилиэстрадудляоркестранотеперьон
асовсемрассыпаетсявпрахонисиделинатравеуплеталисандвичисветчинойсвежуюклубникуияркиеблестящиеточновосковыеапельсиныитридденрассказывалкакмноголетназадтут
повечерамнаразукрашеннойэстрадеигралоркестрмузыкантыизовсехсилтрубиливмедныетрубытолстенькийдирижерobливаясьпотомусерднoразмахивалпалочкойввысокойтр
авегонялисьдругзадругомребятишкиимелькалисветлячкиaподощатыммосткампостукиваякаблукамибудтоиграянаксилофонерасхаживалидамывдлинныхплатьяхсвысокимист
оячимиворотникамиимужчинывтакихтесныхнакрахмаленныхворотничкахчтотогоиглядизадохнутсяВотониостаткиэтихмостковтолькозадолгиегодыдоскисгнилиипревратилисьвкако
етодеревянноемесивоозеролежаломолчаливоеголубоеибезмятежноерыбамедленноплескаласьвблестящихкамышахавагоновожатыйвсеговорилиговорилидетямказалосьчтоониперен
еслисьвкакоетоиноевремяимистертридденсталвдругнадивомолодойаглазаунегогорятнакголубыеэлектрическиелампочкиденьпроплывалсонноБестревожноникточикуданеспеш
илсовсехсторонихобступаллесидажесолнцесловноостановилосьнаодномместеаголостридденаподнималсяипадалистрекозысноваливвоздухерисуязолотыеневидимыеузорыпчелаза
бралaсьвцветокижужжитжужжиттрамвайстоялмолчаливыйточнозаколдованныйорганпоблескиваявсолнечныхлучахребятаелиспелыевишниаруки хунихвсеещедержалсямедныйзапа
хтрамваяикогдатеплыйветерокшевелилнаниходеждуотнеетожеостропахлотрамваемвнебескрикомпролетеладикаяуткактотовздрогнулнупорадомойотцыдаматеричегодоброгоподума
ютчтоявасукралвтемномтрамваебылотихоипрохладносовсемкаквaптекегдеторгуютмороженымприсмиревшиеребятаповернулизашуршавшиеплюшевыесиденьяиуселисьспин
ойктихомуозеруkзаброшеннойэстрадеидощатыммосткамкоторыевыстукиваютподногамизвонкуюдеревяннуюпесенкуеслиидтипонимвдольберегавиныестраныдзиньподбаш
макомтридденазвякнулзвонокитрамвайпомчалсяназадчерезлугсувядшимицветамиоткудауже ушлосолнцечерезлесигородитуткирпичасфальтидеревословностиснулиегосовсе
хсторонтриддензатормозиливыпустилдетейнатенистуюулицучарлиидугласпоследнимиостановилисьуоткрытойдверипередтемкакступитьнаскладнуюподножкуонижадновтягивалино
здрямивоздухпронизанныйэлектричествоминесводилиглазсперчатоктридденанамеднойрукояткедугласпогладилзеленыйбархатныймохсиденьяещеразоглянулсясеребромедьт
емнокрасныйкаквишняпотолокчтождосвиданьямистертридденвсеговамдоброгоребятаещеувидимсямистертридденещеувидимсяраздалсянегромкийвздохэтозакрыласьдвер
ьподобравдлинныйрубчатыйязыкскладнойподножкитрамваймедленноповлылвпослеполуденныйзнойярчесолнцавесьоранжевыйкакмандаринсверкающийзолотомрукояток
ицифрнабокахсвернулзадальнийуголискрылсяпропализглазразвозитьшкольниковвавтобусахпрезрительнофыркнулчарлишагаякобочинетротуаратутужвшколуникакнеудастся
опоздатьпридетзатобойпрямоктвоемукрыльцувжизнииникудатеперьнеопоздаешьвотжутьдугтытолькоподумайнодугласстоялналужайкеиясновиделчтобудетзавтрарабочиезал
ьютрельсыгорячимваромипотомниктодаженедогадаетсячтокогдатоздесьшелтрамвайнонеттеперьиемуиэтимребятамещемногомноголетнезабытьэтойсеребрянойдорожкиск
ольконизливайрельсыварomнастанеттакоеутроосеньюилизимойиливеснойпроснешьсяиеслинеподойтикокнуаостатьсявтеплойуютнойпостелинепременноуслышишькакгдетод
алекочутьслышнобежитизвениттрамвайивизгибеутреннейулицынаширокомпроездемеждуровнымирядамиплатановвязовикленоввтишинепередтемкакначнетсядневнаяжиз
ньуслышишьзадомомзнакомыезвукисловнозатикаютчасысловнопокатитсясгрохотомдесятокжелезныхбочонковсловнозатрещиткрыльяминазаребольшущаяпребольшущаястрекозасл
овнокарусельсловномаленькаяэлектрическаябурясловноголубаямолниямелькнетиисчезнетзазвенитзвонкомтрамвайизашипитточносифонссодовойопускаяивновьподнимаяп
одножкуивновькачнетсяонвагонпоплыветсвоимпутемвседальшеидальшепосвоимпотаеннымдавносхороненнымрельсамккакойтосвоейпотаеннойдавносхороненнойцелипослеужин
апогоняеммячспросилчарлиясноответилдугласяснопогоняемаа

Код:

Source.cpp

```
#include "Includes.h"
#include "TextFormatting.h"
#include "AfiniCrypto.h"
#include "AttackAfini.h"

int main()
{
	// setting console output to 1251 (russian ascii)
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	//

	// gathering info what to do
	std::cout << "Выберите что вы хотите сделать:\n1 - Зашифровать текст (Только UTF-8)\n2 - Расшифровать текст (Только Windows-1251)\n3 - Атака на Аффинный шифр (Только Windows-1251)\nВведите значение: ";
	std::string action;
	std::cin >> action;
	std::string a_string, b_string;
	if (action != "3")
	{
		std::cout << "Введите a: ";
		std::cin >> a_string;
		std::cout << "Введите b: ";
		std::cin >> b_string;
	}
	std::cout << "Введите название файла: ";
	std::string file_name;
	std::cin >> file_name;
	//

	// reading the file
	int64_t file_size = std::filesystem::file_size(file_name);
	char* file_data = (char*)malloc(file_size + 1);
	ZeroMemory(file_data, file_size + 1);
	FILE* file_ptr = NULL;
	fopen_s(&file_ptr, file_name.c_str(), "rb");
	fread(file_data, 1, file_size, file_ptr);
	fclose(file_ptr);
	//

	// doing actions
	if (action == "1")
	{
		// working with data
		std::wstring data_unicode = toUNICODE(file_data, file_size);
		std::string data_string = to1251(data_unicode);
		data_string = clear_the_text(data_string);
		std::ofstream cleared("cleared.txt");
		cleared << data_string;
		cleared.close();
		//

		// encrypting
		int a = atoi(a_string.c_str());
		int b = atoi(b_string.c_str());
		std::string data_encrypted = EncryptDecryptAfini(data_string, a, b, true);
		std::ofstream encrypted("encrypted.txt");
		encrypted << data_encrypted;
		encrypted.close();
		//

		// showing results
		std::cout << data_encrypted << std::endl;
		//
	}
	else if (action == "2")
	{
		// working with data
		std::string data_string(file_data);
		data_string = clear_the_text(data_string);
		std::ofstream cleared("cleared.txt");
		cleared << data_string;
		cleared.close();
		//

		// decrypting
		int a = atoi(a_string.c_str());
		int b = atoi(b_string.c_str());
		std::string data_decrypted = EncryptDecryptAfini(data_string, a, b, false);
		std::ofstream decrypted("decrypted.txt");
		decrypted << data_decrypted;
		decrypted.close();
		//

		// showing results
		std::cout << data_decrypted << std::endl;
		//
	}
	else if (action == "3")
	{
		// working with data
		std::string data_string(file_data);
		data_string = clear_the_text(data_string);
		//

		// attacking
		AttackAfini(data_string);
		//
	}
	//

	system("pause");
	return true;
}
```

Includes.h

```
#include <Windows.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <filesystem>
#include <boost/algorithm/string.hpp>

// constants
//std::string rus_alpha = "абвгдежзийклмнопрстуфхцчшщыьэюя";
std::string rus_alpha = "абвгдежзийклмнопрстуфхцчшщыьэюя";
const int alpha_size = 31;
const int m = 31 * 31;
const int magic_pairs[5] = {
545,436,417,324, 572};
```

TextFormatting.h

```
#pragma once

	wchar_t* unicode_data_ptr = (wchar_t*)malloc(unicode_buffer_size);
	ZeroMemory(unicode_data_ptr, unicode_buffer_size);
```

```
std::string clear_the_text(std::string data)
{
	std::string data_ptr;

	for (int i = 0; i < data.length(); i++)
	{
		if (data[i] >= 'a' && data[i] <= 'я' || data[i] >= 'A' && data[i] <= 'Я')
		{
			data_ptr += data[i];
		}
	}

	boost::to_lower(data_ptr, std::locale(".1251"));
	boost::replace_all(data_ptr, "ё", "е");
	boost::replace_all(data_ptr, "ъ", "ь");

	return data_ptr;
}

std::wstring toUNICODE(void* buffer, int64_t buffer_size)
{
	int64_t unicode_buffer_size = MultiByteToWideChar(CP_UTF8, 0, (LPCCH)buffer, buffer_size, 0, 0) * sizeof(wchar_t) + 2;

	MultiByteToWideChar(CP_UTF8, 0, (LPCCH)buffer, buffer_size, unicode_data_ptr, unicode_buffer_size);

	std::wstring unicode_wstring(unicode_data_ptr);

	return unicode_wstring;
}

std::string to1251(std::wstring unicode_data)
{
	int64_t mb_buffer_size = WideCharToMultiByte(1251, 0, unicode_data.c_str(), unicode_data.length(), 0, 0, 0, 0) + 1;

	char* mb_data_ptr = (char*)malloc(mb_buffer_size);
	ZeroMemory(mb_data_ptr, mb_buffer_size);

	WideCharToMultiByte(1251, 0, unicode_data.c_str(), unicode_data.length(), mb_data_ptr, mb_buffer_size, 0, 0);

	std::string mb_string(mb_data_ptr);

	return mb_string;
}
```

AfiniCrypto.h

```
#pragma once
int mod(int a, int b)
{
	return (a % b + b) % b;
}

struct euclidean_data
{
	long x;
	long y;
	long gcd;
};

long extended_euclidean(int a, int b, long* x, long* y)
{
	if (a == 0)
	{
		*x = 0;
		*y = 1;
		return b;
	}

	long x_temp = 0, y_temp = 0;
	long gcd = extended_euclidean(b % a, a, &x_temp, &y_temp);

	*x = y_temp - (b / a) * x_temp;
	*y = x_temp;

	return gcd;
}

euclidean_data* euclidean_struct(int a, int b)
{
	euclidean_data* result = new euclidean_data();

	result->gcd = extended_euclidean(a, b, &result->x, &result->y);

	return result;
}

int euclidean(int a, int b, bool is_obernene)
{
	auto result = euclidean_struct(a, b);

	if (is_obernene)
	{
		return mod(result->x, b);
	}
	else
	{
		return result->gcd;
	}
}
std::string encryptFunction(std::string pair, int a, int b)
{
	std::string encrypted_pair;
	int firstpos = rus_alpha.find(pair[0]);
	int secondpos = rus_alpha.find(pair[1]);
	int decrypted_number = firstpos * alpha_size + secondpos;
	int encrypted_number = mod(a * decrypted_number + b, m);

	for (int i = 0; i < alpha_size; i++)
	{
		for (int j = 0; j < alpha_size; j++)
```

```
std::string decryptFunction(std::string pair, int a, int b)
{
	std::string decrypted_pair;
	int a_reversed = euclidean(a, m, true);
int firstpos = rus_alpha.find(pair[0]);
int secondpos = rus_alpha.find(pair[1]);
	int encrypted_number = firstpos * alpha_size + secondpos;
	int decrypted_number = mod(a_reversed * (encrypted_number - b), m);

	for (int i = 0; i < alpha_size; i++)
	{
		for (int j = 0; j < alpha_size; j++)
		{
			if ((i * alpha_size + j) == decrypted_number)
			{
				decrypted_pair += rus_alpha[i];
				decrypted_pair += rus_alpha[j];
				return decrypted_pair;
			}
		}
	}

	return "";
}

std::string EncryptDecryptAfini(std::string data, int a, int b, bool is_encryptmode)
{
	std::string final_data;

	for (int i = 0; i < data.length(); i += 2)
	{
		std::string pair = data.substr(i, 2);

		if (pair.length() < 2)
		{
			final_data += pair;
			break;
		}

		if (is_encryptmode)
		{
			final_data += encryptFunction(pair, a, b);
		}
		else
		{
			final_data += decryptFunction(pair, a, b);
		}
	}

	return final_data;
}
```

```
                {
                        if ((i * alpha_size + j) == encrypted_number)
                        {
                                encrypted_pair += rus_alpha[i];
                                encrypted_pair += rus_alpha[j];
                                return encrypted_pair;
                        }
                }
        }

        return "";
}
```

AttackAfini.h

```
#pragma once
int64_t find_pair(std::string pair_data, std::string pair)
{
        for (int i = 0; i < pair_data.length(); i +=2)
        {
                std::string pair_to_cmp = pair_data.substr(i, 2);
                if (pair_to_cmp == pair)
                {
                        return i;
                }
        }
        return std::string::npos;
}

int get_occurrences(std::string data, std::string keyword)
{
        int counter = 0;
        int64_t position = 0;

        while ((position = data.find(keyword, position)) != std::string::npos)
        {
                counter++;
                position += keyword.length();
        }

        return counter;
}

void CheckKeys(std::string encrypted_data, int* a_ptr, int* b_ptr, int arr_size)
{
        for (int i = 0; i < arr_size; i++)
        {
                if (a_ptr[i] != 0 && b_ptr[i] != 0)
                {
                        std::string decrypted_data =
EncryptDecryptAfini(encrypted_data, a_ptr[i], b_ptr[i], false);

                        double data_size = decrypted_data.length();
                        double a_occur =
get_occurrences(decrypted_data, "a") / data_size;
                        double e_occur =
get_occurrences(decrypted_data, "e") / data_size;
                        double o_occur =
get_occurrences(decrypted_data, "o") / data_size;

                        // 0.055 0.06 0.085 - found during deeeeeep tests
                        if (a_occur >= 0.055 && e_occur >= 0.06 &&
o_occur >= 0.085)
                        {
                                std::cout << "Возможная пара
ключей (a,b): " << a_ptr[i] << " " << b_ptr[i] << std::endl;
                        }
                }
        }
}

void FindKeys(std::string pair, std::string pair2, int open_pair, int open_pair2, int**
a_ptr, int** b_ptr, int* arr_size)
{
        int encrypted_number = mod((rus_alpha.find(pair[0]) * alpha_size +
rus_alpha.find(pair[1])) - (rus_alpha.find(pair2[0]) * alpha_size +
rus_alpha.find(pair2[1])), m);
        int decrypted_number = mod(open_pair - open_pair2, m);

        int array_size = 0;
        int* a_array = new int[m];
        int gcd = euclidean(decrypted_number, m, false);

        if (gcd == 1)
        {
                a_array[array_size++] = mod(encrypted_number *
euclidean(decrypted_number, m, true), m);
        }
        else if (decrypted_number % gcd == 0)
        {
                int* occurrence_ptr = new int[data.length() / 2];
                //

                // getting all bigramms and its occurrences value
                for (int i = 0, j = 0; i < data.length(); i += 2)
                {
                        std::string pair = data.substr(i, 2);
                        if (pair.length() < 2)
                        {
                                break;
                        }

                        if (find_pair(bigramms_data, pair) == std::string::npos)
                        {
                                bigramms_data += pair;
                                occurrence_ptr[j++] = get_occurrences(data, pair);
                        }
                }
                //

                // getting max bigramms
                std::string* max_bigramms = new std::string[bigramms_count];
                std::string max_birgamm;

                for (int i = 0; i < bigramms_count; i++)
                {
                        for (int j = 0, max_value = 0; j < bigramms_data.length() / 2;
j++)
                        {
                                if (occurrence_ptr[j] > max_value)
                                {
                                        max_value = occurrence_ptr[j];
                                        max_birgamm =
bigramms_data.substr(j * 2, 2);
                                }
                        }
                        max_bigramms[i] = max_birgamm;
                        int pos = find_pair(bigramms_data, max_birgamm);
                        occurrence_ptr[pos / 2] = 0;
                }
                //

                return max_bigramms;
}

void AttackAfini(std::string encrypted_data)
{
        // first stage, getting n max bigramms
        std::cout << "Атакуем Афины..." << std::endl;
        std::cout << "Ищем 5 самых повторяемых биграмм в зашифрованном
тексте..." << std::endl;

        std::string* max_bigramms = find_max_bigramms(encrypted_data, 5);
        for (int i = 0; i < 5; i++)
        {
                std::cout << i+1 << " повторяющаяся биграмма: "
<< max_bigramms[i] << std::endl;
        }
        //

        // second stage, find all possible keys and trying to find the best keys xD (10
is the best for the magic_pairs, was found during the tests)
        std::cout << "Ищем всевозможные ключи используя статистику русского
алфавита и немного магии..." << std::endl;
        std::cout << "Если выпадает много пар, в большинстве случаев нужно
выбрать ту, которая повторилась больше всех, но так же не забывайте проверить и
другие пары..." << std::endl;

        for (int i = 0; i < 5; i++)
        {
                for (int j = 0; j < 5; j++)
                {
                        if (i == j)
                        {
                                continue;
```

```
                int encrypted_gcd = encrypted_number / gcd;
                int decrypted_gcd = decrypted_number / gcd;
                int m_gcd = m / gcd;
                int x_0 = mod(encrypted_gcd * euclidean(decrypted_gcd,
m_gcd, true), m_gcd);
                a_array[array_size++] = x_0;

                for (int i = 1; i < gcd; i++)
                {
                        a_array[array_size++] = x_0 + (i * m_gcd);
                }
        }

        int* b_array = new int[array_size];
        int one_encrypted_pair = rus_alpha.find(pair[0]) * alpha_size +
rus_alpha.find(pair[1]);

                for (int i = 0; i < array_size; i++)
        {
                b_array[i] = mod(one_encrypted_pair - a_array[i] * open_pair,
m);
        }

        *a_ptr = a_array;
        *b_ptr = b_array;
        *arr_size = array_size;
}

std::string* find_max_bigramms(std::string data, int bigramms_count)
{
        // variables initialization
                std::string bigramms_data;
```

```
                }
                for (int k = 0; k < 5; k++)
                {
                        for (int l = 0; l < 5; l++)
                        {
                                if (k == l)
                                {
                                        continue;
                                }
                                int array_size = 0;
                                int* a_array = NULL;
                                int* b_array = NULL;

                FindKeys(max_bigramms[i], max_bigramms[j], magic_pairs[k],
                magic_pairs[l], &a_array, &b_array, &array_size);
                                // Checking our keys and
showing them if found

        CheckKeys(encrypted_data, a_array, b_array, array_size);
                                //
                        }
                }
        }
        }
        //

        std::cout << "Атака на Афины завершена, надеюсь вы нашли ключ :)" <<
std::endl;
                }
```

Висновок:

Під час данного комп’ютерного практикуму, ми опанували прийомами роботи в модулярній арифметиці та набули навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки.