Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Лабораторна робота

із Криптографії №3

Криптоаналіз афінної біграмної підстановки

Виконали:

Топчій Микита ФБ - 74

Височанська Вікторія ФБ - 71

Перевірено_____

Київ 2019

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3
Криптоаналіз афінної біграмної підстановки

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 (a,b) шляхом розв'язання системи (1).
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.

Варіант 18

Шифрований текст:

юетруожсвеызцэзыфшойызмбибсйкврбсйэффшщшвожкмчюетруожсвекзюегшшоакжжябсйцсвещтюср
оауцезохюдбйэйаяэчэьогбйэжзмеьэнецеяэйгвекзсйютэфейцшгшимюетруожсвейбмомчьогбчуткаучзэз
щмжзвзгефхмафыяэюэрелфауимчмгембуйвецщкэмоцыэбьекзмафыяэбьшемхдшчюфтчймеацжзвзгефх
мафыяэюэрелфауимчмгелецшвимореиыфемэиоялшоуйфбяшмаокжыцзбкжжябсйцсвещтюсроауцезох
юдбркшомэршйаябйолрхэдаючетжжгтифдзтшттфычсведаетсчлехввамчглмоцтябмчжзвзгефхмаллэзио
аущтюсроаучмэзршоюжеоэщедаюцюеютвючавэыфдшгегшчэмэдэдамчвенеттючмочажсвершоюжеоэщ
ечмгецехйызребэлхывмыгеузызталлэффшщшьэвемаэфлшщтсэиеабзеьаллжсвелешжфехййэезбйютяшл
фнекцршчмхвсйызвееосэткгйреткммвеуоцыэбьекзюсвершгеюкоюдбкзиэезнжцышоларемангмэяшбша
шксййгшвозшашткшощсревкэбюсейчффыяэюэчкэбюсейчфокаагтсдвзбэхэвейюейюэббморенеозюбюс
брвеушйжфшьэаыфшойызмбибсйвйэффшщшвожкмчрещбдоуйюевебкэфлеишеьэщлэыфэфлшщшнеою
ючялфеузхосйлхлецшвиффкшненрлечммэээиожкэщлэыфялфыфедааууййютциимайсмббщтюсрофгмэщ
тнвдаяферфанрршдедаважсимдэщбдоуйршюетруожсвечмдэшбяеимцщдшшдяэдашзццкэрзнэвччсфбсв
ьцавьцззюевеацмашоцыэбьечюдбмбуйвецщкэрзгшцйьоатчфцтсыэбчажкгзфлнэвдчаозглцтэенемхдшч
юфтчйцэхекэсшворбфжаышолаякркмщбфпжвэщшфаюаяекйвеларбвйчфуйезсмонрквеацкэайчфуэкзхэ
нэаыжсючофббнвдабэуаллжсвеншеьллоюнпмощмнрюэцщдшайатэлллокозтшттфыфцкэрзнэьтжжгтиб
фбовууочэбмоткаувчсймебймещлаюцтьэаэчмнэаюффйшсмвчсверкзшыаакчмеиоеэдагтжкэфршофгшу
швосдщтчьейледэамткьобрмыаакчгйршвзвшломоимцжатяфсчочшйгедйджцтцзцявдлечммэиыяэюэимя
лфыфшойызмбжщмйифршййюэббвчкчменэушдацшушбюэбеиючфцгбаюерючнрбэамткьояшщтюсров
шбюэбеимомчвдааитдфйшмйээбйщклраадшмхреужлалрщтюсроуэнээшьмстевщтюсрощтнвдаяферфан
ркзщеоэкзмецешыозцшщбффсчйймчмысйвюеэщвроркэфэбхпимдэюебфзшзоюсвегосймэбйозиомойэп
лфеузхосйлхжйоюзабряеоэкзмецегегосчмышонвуудабэвшмхредаййцеууашоцнздабрбйюдатэлвдледыя
ффэуыяэюэюыфеыжцшдаййэфцтфэдиьшнрлечммэуаркаусшщтюсроаужзмейкжжкзюегшшойечфеймэб
фсэцеайжзвзбэббгэааюзлхкзвзилцтэечйчфземхдшчюфтчймеиыфшойызмбтшмйдомчнпрбьояшфбройп
рбьояшюшкфллынозбьхжткллокцыозэзюедажычайояфвеяэеомщббзпжцжюзшсшнэчмшоршвзжевшнээ
ароуййцшюгэршвзжевшкюбвьцнеаэютуачучашйссимэкцыознэлехввакздаэбяккчмоьшцщдшэюдбоюдф
цеаймшкзюегшшоуэцезоаоючллгечмчмлхйпрбжюаоцыэбьевоьеткгткэнйлафшаерзьэючаадажычафжяф
ййййцрорфрбрбкздажыванфауткмчршчмгечмцезоаозшсшнэчмбйейцтлзйааоиыхпрбябмысйициэлхкзпаг
тюсццжййцшюсахпрфауимршщегбоюхюзарахпмоючллгечмчмлхцезоошсшнэчмршчмгероютгльогбйэ
ровюдбоюыыяэбьйемхдшчюфтчйменеюсофдарзсймеыюгшьэифябвкезгвфшровюдбоюшймчщвцтлзиоа
умбуймэттсйхэйэршчмгечмлтьогцтйместрфэбмьйэмысйгшдерздивдоэеавагтдарзйэзыцтйэффлшневене
мафыфеыжцшгелтьогцтйтцчтрфэбмьтщчаркюзгшвючжщвреаэогмэуыяэюэвоюсялшоючллгечмчмлхда
ябоюзаеыяэюэимюаффлшдэшбшззвыыяэюэьйэфчзнэщблаллжсвечмщтсэюфеймэвемаоюдбсэббнвэфва
рбйошоюсвершчмгеййвбьсчарбйоаущтюсроаулечммэмзлечммэнвэбгооюдбсйэфцтфэлхнрвемхмеюеве
ацмазередеспречмгелецшвичавйевэбгохюдбвчфбнэшодбовууаафшщиййчемйцевдушгецшуштквдщцбй
чфййцеялгтжжычуээарояфсйвецйвзчзюзецкэрзлхлшашюедесосдвееоаарксшжзюоеймемхмецшмепакчк
ожжывбфсйвежзкзвзчзсинэверыдбовууаюффлрлшашюедесосддшайчфбркзюегшшоуэябвкезгвозааббэ
фркаагтсдвзбэлшгежзвзгефхмаоюшйоюзаяшлецшвифбсйвэнэршлецшвифбсйяюдбчарксшузгещтсэюф
эбйохкрксвуудабэвкчпденелехввацинвкшнедыкдщвюентвютщфбмышорюгтягвершщегбоюзавэкзушю
евеацмащвервенемаоюдфэбдоцыэбьекзшзгшщйьоатваюдцщдшхейкаагтсджзвзгефхвзбэлшгелечммэвю
дбуэшыэбршкзршоюжеоэщечмгеледыывмояшашксййгшвосдоюдайрвшмхдаюючжщвренеыжмевеммр
йоюзаяшршщегбоюшйэфцтфэлхнрршоюжеоэщечмгемаоюсчфбнэшодбовууттакмдмевеммнюдбывууда
бэвшмхкзэйфбуйэбюзщммевеммейютрбибффчжнэщтэфшзйамыжицэкздечмгемаоюоеьэлегбуйфбййвэ
шыэбьэлхюаофэбышоэьэеосдййэзчмыбовуужюдбмыяэцшдшимтгвеацашксййгшвосчменэушдацедйгег
осдозщозеытжжимуыяэюэенваябыжпфййвеяэроеэпюйпмощбтеврэомобрршозоцлосдифгшьэоазшдара
юдозоцлосйлхмевеммйчаоиышовчйшшсючаегшючшйызозючнрозоцлозеврэочаюгмэифгшьэифййвеяэ
чаючшоййызрешрмйеверкзэоцтжжычршеэршюетруоьшждвзуушгецшьпффуащбэйвюаыэбфжшшмйэ
йцеаедеивердеданпмобрбфсэыбюсимцьрбууюсофдаэцуабйючсцбшдацедйцевчмазеучхпуалшфсуйювр
бвавэнэршюврбвараркзшведшайчфяшрежзцймефешоюврбйшвшнеызявуудабэбгвелецшвиморерфыжц
шдаимшчкчмочффшщшьэдаьоэфчанррщеюфозмэсмдэдшэзуанрдедайрьэючййвеяэффмевефехеревчь
шуоейцеаехозшцыфшдещтнврбморечжкзушьэьшдаквцшдагтшоэпыжьэюкчпдешсрбэфвкйпыжщхнэео

лрмевеозлаючаегшейцеаехоисючаамбщсморшьшеэйацтейшэыэщовймэрэзолрферфлравщсрбэфвкршы
жашгосэшыэбтруащтнвиэезвоошючхжркшонсрбэфвклещоварбйохпффуащбэйвюдосййцыэйюейлещов
арбйосчгтлвяшмвердеданпмобрййвеяэуштквдоюбшушгецшуштквдэшгшайэфцткшвозуюедажыокршц
еюфозмэдэдшайчфяшялсйэфчзюзифэффшщшьэочфбнэшомчсвуудабэивьцчэыбовууюейцнзуаййвеяэу
шткзшщоисмояшушпфыиейвечаушгецшюойеьяейчфморелбейуопьщсючтюэбеивчфбовьчяеьйшрвзйа
ючаемхкшмбуймэшовчьшуоывьцчэыбовуувшщеушбэхоьшщтнвфыфшдшгшвосйлхроеэпюйпредаякчп
денеялфысйицекуйцтфьльетрфэфьцэшаеьхууцзюхейлхозкэдагтйэййвеяэлазшдаяюейлещоварбйолрою
ьзотзшрзбпыжщхушгецшушдарзбйщкаудэццведпыжьэцзфшгыэбфжшшмйеверлшвржзмеьцоэючщбэй
цевдэшгшзоцыэбьевоейщеютдоважсвеючаюэбьшушбкллюзьэжцнхвшюшюхьлщвакжжябсйцсвеааршн
евзбэлшгефбровкркдысйгшхоючиэфхщшиэезгшацябвкезгвжожкллюзаегшвкрквшюшюхршюещбйпяб
вшлоеэпюйпимцжркшомчмеваммчжуйызмбибяшнедшайчфьрцеуофимойэлшвржзмшацжзвзгефхюебф
зшацщтсэюфэблврбвакверюобшнедшэзщбээдагтйэвемащвэбгооюдбсйэфцтфэлхнрцщоэвзчзюзщфгляб
йййояфсйвеявервенемаоюфцкэрзлхатэлатэлуэюевзмеццледыывмобрщтюсроюецврбябййвймчййоюьщ
ййэфчзнэсгмэдйоюзаяшташйссвчрбцэуыяэюэгверюаллщвбэсшьэээщбифауимакгтлжьоейоюцщдшьэвк
шомэршйаябйоаухэдаючшэншеьллэбквможцшлткчусвбэсшьэрфауткдыьцхйтщщтнвщтролазшжзукжж
ябсййоцыэбьецщщблаллааэбякжсвечмдедаглццлхцегтзпморенедэозфбгюэфвэщтффнсжзлтьогцтйменев
чюшозвеозвфауршледыяфдзэйозшыэбхпмояшщшнекзфеюзофгшушьэялозгвяещосйлхйыяэневембмоч
уьтуйжеымюещтглааэфгшьэжэюеютвюредаеыяэоюфшвшьэларквдомнелкаагтошчмокжыфбегмэозодщ
вшэщвфтузьэдияшюеушдаффатчждшршсвмыгеузмэуштквшюшфшнецемммевеммиыэфмлеркцлеттючм
моьэмеююдбффнвмошййитмоьэипрбьояшмвсймэсшьэхчьтуэмевеммуаэбякнцпжмбибййчэбгюяшве
аэатэлаудагтшошэдэшбяеимдэшбяерейвьцчэуыяэюэюыщвдшэздивднрюеоэщшцяьшлецшвичавйэйоюз
аяшледыяффэрйьэбкллюзледтрбйрхбоксэчзюзуаааушнешофыывйшвеьаллэбюсведрневшуайжючгшаа
вененэмаджуйсдйшмйвшгпмочеьаллэбюсвесцфэмйэйэббьючмаааабпрбьшбсвецшвшнацрлшвшнеаавен
енэмасйвеняючофьэоцузмхюэтцжеушмйэйейлхялшоюсвершщтюсрояфлшюаааэбяксйэфчздшьэуаюдгй
гебэгыяэмевеммрйэфцтдзуанрбйщкзшыэщомореьешльоюсеймееймэрэзозшлечммэеэвейюааьожкмеве
ммзовудшбсвещюгемечзюзуаааvenемаокгшбсважзозюбнлйшмйээвеацюедесозшжцдшьарксшщтяфчар
ксшршщтяфжзцэвеютвюмоокдайрлэлфьэоцмевеммайжзсшбсвещюгемежыябзеьаллжсвеючыдаарбсйв
йейлхялсэмазеучэбфбнроююакчызьпиыозбэршщтяфжцдшьаааеасйжзсшбсвекзюегшшоюекшшфщеткэ
цюевеацмазенжмдаарбюаллсэаавенеюечэясвеябвкезгвсжмдаарбялфыллжюрбжкркщтяфроиыозююлву
авейкветкчпуамаджяфябффдзэйифдэшбяеюылвэфюелкфыьокзцлгтпцчзэзаавенемафысйгшхочуозаеуо
жклветфбюехоьшюеббнвьэстжжцэыкьщбэдагтюсейцэрйоюзабряарккзамрзючошцшнэцэрбовййсащркб
хксвэщлэмоуэщшрерзьэайоюзатщмевеммуарксшжзоркбхквчмоьшбйялсэщшфабфпжййезсмдйоюзаяш
бкчпденвуудабэеэнкчпуяьшнрвчщтсэюфэбйосчщбэймчвдлшврбэдагтюсеймедйсмббмоиыфеыжцемхре
ьомдщвчанрбйшртэлебйючжededэдагтйэяещтнврбмофыфшщфщсрегз

Розшифрований текст:

понятночтотакимпредставлялосьделосовременникампонятночтонаполеонуказалосьчтопричинойвойн
ыбылиинтригианглиикаконигоборилэтонаостровесвеленыпонятночточленаманглийскойпалатыказало
сьчтопричинойвойныбыловластолюбиенаполеоначтопринцуольденбургскомуказалосьчтопричинойво
йныбылосовершенноепротивнегонасилиечтокупцамказалосьчтопричинойвойныбылаконтинентальная
системаразорявшаяевропучтостарымсолдатамигенераламказалосьчтоглавнойпричинойбыланеобходи
мостьупотребитьихвделолегитимистамтоговремениточчтонеобходимобыловосстановитьадипломатамт
оговремениточчтовсепроизошлооттогочтосоюзроссиисавстриейвгодунебылдостаточноискусноскрытот
наполеонаичтонеловкобылнаписанзапонятночтоэтиещебесчисленноебесконечноеколичествопричинк
оличествокоторыхзависитотбесчисленногоразличияточекзренияпредставлялосьсовременникамнодлян
аспотомковсозерцающихвовсемегообемегромадностьсовершившегосясобытияивникающихвегопросто
йистрашныйсмыслпричиныэтипредставляютсянедостаточнымидлянаснепонятночтобымиллионылюде
йхристианубивалиимучилидругдругапотомучтонаполеонбылвластолюбивалександртвердполитикаанг
лиихитраигерцогольденбургскийобиженнельзяпонятькакуюсвязьимеютэтиобстоятельствассамымфакт
омубийстваинасилияпочемувследствиетогочтогерцогобиженгысячилюдейсдругогокраяевропыубивал
иираздорялилюдейсмоленскойимосковскойгубернийибылиубиваемыимидлянаспотомковнеисториковн
еувлеченныхпроцессомизысканияипотомуснезатемненнымздравымсмысломсозерцающихсобытиепри
чиныегопредставляютсявнеисчислимомколичествечембольшемыуглубляемсявизысканиепричинтембо
льшенамихоткрываетсяивсякаяотдельновзятаяпричинаилицелыйрядпричинпредставляютсянамодинак
овосправедливымисамипосебеиодинаковоложнымипосвоейничтожностивсравнениисгромадностьюсо
бытияиодинаковоложнымипонедействительностисвоейбезучастиявсехдругихсовпавшихпричинпроизв
естисовершившееясобытиетакойжепричинойкакотказнаполеонаотвестисвоивойсказавислуиотдатьназ
адгерцогствоольденбургскоепредставляетсянамижеланиеилинежеланиепервогофранцузскогокапралап
оступитьнавторичнуюслужбуибоежелибыоннезахотелидтинаслужбуинезахотелбыдругойитретийитыс
ячныйкапралисолдатнастолькоменеелюдейбылобыввойскенаполеонаивойнынемоглобыбытьежелибын
аполеоннеоскорбилсятребованиемотступитьзавислуиневелелнаступатьвойскамнебылобывойныноеже
либывсесержантынепожелалипоступитьнавторичнуюслужбутожевойнынемоглобыбытьтоженемоглоб
ыбытьвойныежелибынебылоинтриганглиииинебылобыпринцаольденбургскогоичувстваоскорблениява
лександреинебылобысамодержавнойвластивроссииинебылобыфранцузскойреволюцииипоследовавши
хдиктаторстваимперииивсеготогочтопроизвелофранцузскуюреволюциюитакдалеебезоднойизэтихпр
ичинничегонемоглобыбытьсталобытьпричиныэтивсемиллиардыпричинсовпалидлятогочтобыпроизвес
титочтобылоисследовательноничтонебылоисключительнойпричинойсобытияасобытиедолжнобылосов
ершитьсятолькопотомучтоонодолжнобылосовершитьсядолжныбылимиллионылюдейотрекшисьотсво
ихчеловеческихчувствисвоегоразумаидтинавостоксзападаиубиватьсебеподобныхточнотакжекакнеско
льковековтомуназадсвостоканазападшлитолпылюдейубиваясебеподобныхдействиянаполеонаиалекса
ндраотсловакоторыхзависелоказалосьчтобысобытиесовершилосьилинесовершилосьбылитакжемалопр
оизвольныкакидействиекаждогосолдаташедшеговпоходпожребиюилипонаборуэтонемоглобытьиначеп
отомучтодлятогочтобыволянаполеонаиалександратехлюдейоткоторыхказалосьзависелособытиебылаи
сполненанеобходимобылосовпадениебесчисленныхобстоятельствбезодногоизкоторыхсобытиенемогл
обысовершитьсянеобходимобылочтобымиллионылюдейврукахкоторыхбыладействительнаясилаолда
тыкоторыестреляливезлипровиантипушкинадобылочтобыонисогласилисьисполнитьэтуволюединичны
хислабыхлюдейибылиприведеныкэтомубесчисленнымколичествомсложныхразнообразныхпричинфат
ализмвисториинеизбежендляобъясненияразумныхявленийтоестьтехразумностькоторыхмынепонимае
мчемболеемыстараемсяразумнообъяснитьэтиявленияисториитемонистановятсядлянасразумнееинеп
онятнеекаждыйчеловекживетдлясебяпользуетсясвободойдлядостижениясвоихличныхцелейичувствует
всемсуществомсвоимчтоонможетсейчасделатьилинесделатьтакоетодействиенокакскороонсделаетего
такдействиеэтосовершенноевизвестныймоментвременистановитсяневозвратимымиделаетсядостояние
мисториивкоторойоноимеетнесвободноеапредопределенноезначениеестьдвестороныжизнивкаждомче
ловекежизньличнаякотораятемболеесвободначемотвлеченнееееинтересыижизньстихийнаяроеваягдече
ловекнеизбежноисполняетпредписанныеемузаконычеловексознательноживетдлясебянослужитбессозн
ательныморудиемдлядостиженияисторическихобщечеловеческихцелейсовершенныйпоступокневозвр
атимидействиеегосовпадаявовременисмиллионамидействийдругихлюдейполучаетисторическоезначен
иечемвышестоитчеловекнаобщественнойлестницечемсбольшимилюдьмионсвязантембольшевластион

имеетнадругихлюдейтемочевиднеепредопределенностьинеизбежностькаждогоегопоступкасердцецаре
вовруцебожьейцарьестьрабисториииисториятоестьбессознательнаяобщаяроеваяжизньчеловечествавсяк
ойминутойжизницарейпользуетсядлясебякакорудиемдлясвоихцелейнаполеоннесмотрянаточтоемубол
еечемкоганибудьтеперьвгодуказалосьчтоотнегозависелоилинекаквпоследнемписьмеписалемуалекса
ндрникогдаболеекактеперьнеподлежалтемнеизбежнымзаконамкоторыезаставлялиегодействуявотноше
ниисебякакемуказалосьпосвоемупроизволуделатьдляобщегоделадляисторииточтодолжнобылосоверш
итьсялюдизапададвигалисьнавостокдлятогочтобыубиватьдругдругаипозаконусовпаденияпричинподде
лалисьсамисобоюисовпалисэтимсобытиемтысячимелкихпричиндляэтогодвиженияидлявойныукорыза
несоблюдениеконтинентальнойсистемыигерцогольденбургскийидвижениевойсквпруссиюпредпринят
оекакказалосьнаполеонудлятоготолькочтобыдостигнутьвооруженногомираилюбовьипривычкафранцу
зскогоимператораквойнесовпавшаясрасположениемегонародаувлечениеграндиозностьюприготовлени
йирасходыпоприготовлениюипотребностьприобретениятакихвыгодкоторыебыокупилиэтирасходыиод
урманившиепочестивдрезденеидипломатическиепереговорыкоторыеповзглядусовременниковбыливед
енысискреннимжеланиемдостижениямираикоторыетолькоуязвлялисамолюбиетойидругойстороныими
ллионымиллионовдругихпричинподделавшихсяподимеющеесовершитьсясобытиесовпавшихснимкогд
асозрелояблокоипадаетотчегоонопадаетоттоголичтотяготееткземлеоттоголичтозасыхаетстерженьотто
голичтосушитсясолнцемчтотяжелеетчтоветертрясетегооттоголичтостоящемувнизумальчикухочетсясе
стьегоничтонепричинавсеэтотолькосовпадениетехусловийприкоторыхсовершаетсявсякоежизненноео
рганическоестихийноесобытиеитотботаниккоторыйнайдетчтояблокопадаетоттогочтоклетчаткаразлага
етсяитомуподобноебудеттакжеправитакженеправкакитотребенокстоящийвнизукоторыйскажетчтоябл
окоупалооттогочтоемухотелосьсестьегоичтоонмолилсяобэтомтакжеправинеправбудеттоткткоскажетчт
онаполеонпошелвмосквупотомучтоонзахотелэтогоиоттогопогибчтоалександрзахотелегопогибеликакп
равинеправбудеттоткткоскажетчтозавалившаясявмиллионпудовподкопаннаягораупалаоттогочтопослед
нийработникударилподнеепоследнийразкиркоювисторическихсобытияхтакназываемыевеликиелюдис
утьярлыкидающиенаименованийсобытиюкоторыетакжекакярлыкименеевсегоимеютсвязиссамымсобы
тиемкаждоедействиеихкажущеесяимпроизвольнымдлясамихсебявисторическомсмысленепроизвольно
анаходитсявсвязисовсемходомисторииопределенопредвечноаа

КЛЮЧ: (425, 100)

Найчастіші біграми шифрованого тексту:

1. ве
2. да
3. эб
4. ге
5. ме

Розпізнавач російської мови:

Для визначення того, чи являється текст інформативним використовувався підхід на основі індексу відповідності та ентропії. Для кожного набору ключів аналізувався розшифрований текст, у випадку коли ентропія та індекс відповідності були у допустимих межах, то текст вважається коректним. Межі допустимості були визначені емпіричним шляхом.

Труднощі:

При виконанні даного практикуму виникли деякі труднощі. По-перше, некоректно було дане пояснення щодо необхідного алфавіту, а саме заміна твердого та м’якого знака.
Окрім цього деякі труднощі викликав алгоритм перебору можливих наборів біграм.

Висновок: у ході виконання практикуму було набуто знань з використання афінного шифру та методів його криптоаналізу. Було набуто навичок аналізу тексту на його інформативність за допомогою статистичних даних, розглянуто декілька моделей на основі яких проводився аналіз.

Програмна реалізація

```
#include <iostream>
#include <string>
#include <map>
#include <set>
#include <fstream>
#include <limits>
#include <cmath>
#include <algorithm>
//--------------------------------------------
#define PRECISION 0.01
#define CI  0.0553
#define ENTROPY 4.459
//--------------------------------------------
using namespace std;

set<char> Alphabet = { 'а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и',    'й', 'к', 'л', 'м', 'н','о', 'п',
                                                                                    'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч','ш', 'щ', 'ы', 'ь', 'э', 'ю', 'я' };

map<char, int> CharIntMap = { {'а', 0}, {'б', 1}, {'в', 2}, {'г', 3}, {'д', 4}, {'е', 5}, {'ж', 6}, {'з', 7}, {'и', 8},
 {'й', 9}, {'к', 10}, {'л', 11}, {'м', 12}, {'н', 13}, {'о', 14}, {'п', 15}, {'р', 16},
 {'с', 17}, {'т', 18}, {'у', 19}, {'ф', 20}, {'х', 21}, {'ц', 22}, {'ч', 23}, {'ш', 24},
 {'щ', 25}, {'ь', 26}, {'ы', 27}, {'э', 28}, {'ю', 29}, {'я', 30} };

map<int, char> IntCharMap = { {0, 'а'}, {1, 'б'}, {2, 'в'}, {3, 'г'}, {4, 'д'}, {5, 'е'}, {6, 'ж'}, {7, 'з'}, {8, 'и'},
 {9, 'й'}, {10, 'к'}, {11, 'л'}, {12, 'м'}, {13, 'н'}, {14, 'о'}, {15, 'п'}, {16, 'р'},
 {17, 'с'}, {18, 'т'}, {19, 'у'}, {20, 'ф'}, {21, 'х'}, {22, 'ц'}, {23, 'ч'}, {24, 'ш'},
 {25, 'щ'}, {26, 'ь'}, {27, 'ы'}, {28, 'э'}, {29, 'ю'}, {30, 'я'} };

map<int, string> MostFreqBigram = { {0, "ст"}, {1, "но"}, {2, "то"}, {3, "на"}, {4, "ен"} }; // Most frequent bigram in russian language

bool AlmostEqual(double, double); // Comparing two double values function with precision value PRECISION;
void initMonogramMap(string, map<char, double>&, set<char>, double&); // Initialization map with monogram and their amount
void initBigramMap(string, map<string, double>&, set<char>, double&); // Initialization map with bigram and their amount
double MonoEntropy(map<char, double>, double); // Get monogram entropy; It is used in TextAnalyzer
double MonoCI(map<char, double>, double); // Get coincidence index; It is used in TextAnalyzer
void ShowMap(map<char, double>, double); // Show map content
void ShowMap(map<string, double>, double);
void ShowMap(map<int, string>);
bool TextAnalyzer(map<char, double>, double); // Function, that check text for it's pithiness (via statistics);
                                                                                                                        //return 1 - text is
informative, 0 - not informative
int GCD(int a, int b); // Great common divisor
int ExtendedGCD(int a, int b, int& x, int& y);
int BigramToInt(string, map<char, int>); // Function that returns bigram value converted in digital type
void IntToBigram(int, string&, map<int, char>); // Fuction that convert Int value to Bigram
void TopNValue(map <string, double>, map<int, string>&, int); // Get top N value map
void EqSolution(int, int, int, map<int, int>&, int, int);
void EncryptAffine(string, int, int, map<char, int>, map<int, char>); // Encryption function for Affin cipher
void DecryptAffine(string, int, int, map<char, int>, map<int, char>); // Decryption dunction for Affin ciphers
void FindKeys(string, map<int, string>, map<int, string>, map<char, int>, map<int, int>); // Find key (value a and b) for affine cipher

int main() {
        setlocale(LC_ALL, "rus");
        string FilePath = "..\\..\\18.txt";
        map<string, double> BiMap;
        double BigramAmount = 0;
        initBigramMap(FilePath, BiMap, Alphabet, BigramAmount);

        cout << "\n\n\n";
        map<int, string> TopMap;
        TopNValue(BiMap, TopMap, 5);
        ShowMap(TopMap);
        map<int, int> Solution;
        FindKeys(FilePath, TopMap, MostFreqBigram, CharIntMap, Solution);
        system("pause");
        return 0;
}

bool AlmostEqual(double a, double b) {
        if (fabs(a - b) < PRECISION) return 1;
        else return 0;
```

```
}

void initMonogramMap(string FilePath, map<char, double> & Letters, set<char> Alphabet, double& LetterAmount) {
	ifstream fin(FilePath);
	string buffer;
	if (fin.is_open()) {
		while (fin.peek() != EOF) {
			getline(fin, buffer);
			fin.seekg(fin.tellg());
			for (int i = 0; i < buffer.length(); i++) {
				if (Letters.count(char(tolower(buffer[i])))) {
					Letters.at(char(tolower(buffer[i])))++;
					LetterAmount++;
				}
				else {
					Letters.emplace(char(tolower(buffer[i])), 1);
					LetterAmount++;
				}
			}
		}
	}
	else cout << "File opening error" << endl;
	fin.close();
};

void initBigramMap(string FilePath, map<string, double> & BiMap, set<char> Alphabet, double& BigramAmount) {
	ifstream fin(FilePath);
	string buffer;
	string TempBigram;
	if (fin.is_open()) {
		while (fin.peek() != EOF) {
			getline(fin, buffer);
			fin.seekg(fin.tellg());
			for (int i = 0; i < buffer.length() - 1; i += 2) {
				TempBigram.push_back(buffer[i]);
				TempBigram.push_back(buffer[i + 1]);
				//cout << TempBigram << endl;
				if (BiMap.count(TempBigram)) {
					BiMap.at(TempBigram)++;
					BigramAmount++;
				}
				else {
					BiMap.emplace(TempBigram, 1);
					BigramAmount++;
				}
				TempBigram.clear();
			}
		}
	}
	else cout << "File opening error" << endl;
	fin.close();
}

double MonoEntropy(map<char, double> MonoMap, double LetterAmount) {
	double Entropy = 0;
	for (auto it = MonoMap.cbegin(); it != MonoMap.cend(); it++) {
		Entropy += -(it->second / LetterAmount) * log2(it->second / LetterAmount);
	}
	return Entropy;
}

double MonoCI(map<char, double> MonoMap, double LetterAmount) {
	double CoincidenceIndex = 0;
	for (auto it = MonoMap.cbegin(); it != MonoMap.cend(); it++)
	{
		CoincidenceIndex += it->second * (it->second - 1);
	}
	return CoincidenceIndex / (LetterAmount * (LetterAmount - 1));
}

void ShowMap(map<char, double> MonoMap, double LetterAmount) {
	for (auto it = MonoMap.cbegin(); it != MonoMap.cend(); it++) {
		cout << it->first << "   " << it->second / LetterAmount << endl;
```

```
		}
}

void ShowMap(map<string, double>BiMap, double BigramAmount) {
		for (auto it = BiMap.cbegin(); it != BiMap.cend(); it++) {
				cout << it->first << " " << it->second / BigramAmount << endl;
		}
}

void ShowMap(map<int, string> Map) {
		for (auto it = Map.cbegin(); it != Map.cend(); it++) {
				cout << it->first << " " << it->second << endl;
		}
}

bool TextAnalyzer(map<char, double> MonoMap, double LetterAmount) {
		// If text is informative return value = 1, else - 0
		double CoincidenceIndex = MonoCI(MonoMap, LetterAmount);
		double Entropy = MonoEntropy(MonoMap, LetterAmount);
		if (AlmostEqual(CI, CoincidenceIndex) || AlmostEqual(ENTROPY, Entropy)) {
				cout << CoincidenceIndex << "		" << Entropy << endl;
				return 1; // Text is informative
		}
		else return 0;
}

int GCD(int a, int b)
{
		if (b == 0)
				return a;
		return GCD(b, a % b);

}

int ExtendedGCD(int a, int b, int& x, int& y) {
		if (a == 0) {
				x = 0;
				y = 1;
				return b;
		}
		int x1, y1;
		int d = ExtendedGCD(b % a, a, x1, y1);
		x = y1 - (b / a) * x1;
		y = x1;
		return d;
}

int BigramToInt(string Bigram, map<char, int> CharIntMap) {
		int BigramValue;
		if (Bigram.size() != 2) return 0;
		else {
				BigramValue = CharIntMap.find(Bigram[0])->second * CharIntMap.size() + CharIntMap.find(Bigram[1])->second;
		}
		return BigramValue;
}

void IntToBigram(int BigramValue, string & Bigram, map<int, char> IntCharMap) {
		Bigram.clear();
		int AlphabetSize = IntCharMap.size();
		Bigram.push_back(IntCharMap.find(BigramValue / AlphabetSize)->second);
		Bigram.push_back(IntCharMap.find(BigramValue % AlphabetSize)->second);
}

void TopNValue(map<string, double> BiMap, map<int, string> & TopMap, int Amount) {
		string Bigram;
		map<string, double> CopyBiMap = BiMap;
		//ShowMap(CopyBiMap, 1);
		double Value = 0;
		for (int i = 0; i < Amount; i++) {
				for (auto it = CopyBiMap.cbegin(); it != CopyBiMap.cend(); it++) {
						if (TopMap.count(i)) {
								if (it->second > Value) {
										Value = it->second;
```

```
                                                Bigram = it->first;
                                                TopMap.at(i).clear();
                                                TopMap.at(i) = Bigram;
                                                Bigram.clear();
                                        }
                                        else continue;
                                }
                                else {
                                        TopMap.emplace(i, it->first);
                                }
                        }
                        CopyBiMap.erase(TopMap.find(i)->second);
                        //cout << CopyBiMap.size() << endl;
                        Value = 0;
                }
        }

void EqSolution(int diffY, int diffX, int AlphabetSize, map<int, int> & Solution, int X, int Y) {
        // ax = b(mod m);
        // diffY = a * diffX (mod |Alphabet|^2); =>
        // => a = diffX^(-1) * diffY (mod |Alphabet|^2);
        //int m = pow(AlphabetSize, 2);
        int m = AlphabetSize;
        if (diffX < 0) diffX += m;
        if (diffY < 0) diffY += m;
        if (GCD(diffX, m) == 1) {
                int x = 0, y = 0; // ExtendedGCD(a, b, &x, &y);
                ExtendedGCD(diffX, m, x, y);
                if (x < 0) x += m; // x is reverse element of diffX
                int a = (x * diffY) % m;
                //int b = (BigramToInt(IteratorY->second, CharIntMap) - a * BigramToInt(IteratorX->second, CharIntMap)) % m;
                int b = (Y - a * X) % m;
                if (b < 0) b += m;
                cout << " a: " << a << " b: " << b << endl;
                Solution.emplace(a, b);
                return;
        }
        else if (GCD(diffX, m) != 1) {
                if (diffY % GCD(diffX, m) != 0) {
                        cout << "Equatation hasn't solution" << endl;
                }
                else if (diffY % GCD(diffX, m) == 0) {
                        // if diffY % d = 0 => Exist d solution;
                        int d = GCD(diffX, m);
                        int a1 = diffX / d;
                        int b1 = diffY / d;
                        int n1 = m / d;
                        int x = 0, y = 0;
                        ExtendedGCD(a1, n1, x, y); // x - a1^(-1) mod n1
                        int x0 = (b1 * x) % n1;
                        for (int i = 0; i < d; i++) {
                                int a = x0 + i * n1;
                                //int b = (BigramToInt(IteratorY->second, CharIntMap) - a * BigramToInt(IteratorX->second,
CharIntMap)) % m;
                                int b = (Y - a * X) % m;
                                if (b < 0) b += m;
                                cout << " a: " << a << " b: " << b << endl;
                                Solution.emplace(a, b);
                        }


                }
                return;
        }
}

void EncryptAffine(string FilePath, int a, int b, map<char, int> CharIntMap, map<int, char> IntCharMap) {
        ifstream fin(FilePath);
        ofstream fout("..\\..\\EncryptedFile.txt");
        string buffer;
        string TempBigram;
        int AlphabetSize = CharIntMap.size();
        int BigramIntValue; // Variable that containt integer value of bigram;
```

```
	int EncBigramIntValue;
	string EncBigram;
	//char EncChar; // Encrypted char
	if (fin.is_open()) {
		while (fin.peek() != EOF) {
			getline(fin, buffer);
			fin.seekg(fin.tellg());
			for (int i = 0; i < buffer.length() - 1; i += 2) {
				TempBigram.push_back(buffer[i]);
				TempBigram.push_back(buffer[i + 1]);
				BigramIntValue = BigramToInt(TempBigram, CharIntMap);
				EncBigramIntValue = (a * BigramIntValue + b) % (AlphabetSize * AlphabetSize);
				IntToBigram(EncBigramIntValue, EncBigram, IntCharMap);
				//cout << /*BigramIntValue << "		" <<*/ EncBigram;
				fout << EncBigram;
				EncBigram.clear();
				TempBigram.clear();
			}

		}
	}
	else cout << "File opening error" << endl;
	fin.close();
	fout.close();
}

void DecryptAffine(string FilePath, int a, int b, map<char, int> CharIntMap, map<int, char> IntCharMap) {
	ifstream fin(FilePath);
	ofstream fout("..\\..\\Decrypted Files\\DecryptedFile " + to_string(a) + " " + to_string(b) + ".txt");
	string buffer;
	int AlphabetSize = CharIntMap.size();
	int EncBigramIntValue;  // Integer variable that contain char transforemed in int form
	int  DecBigramIntValue; // Decrypted bigram in digital form
	string EncBigram; //Encrypted bigram
	string DecBigram; //Decrypted bigram
	int x, y; // Variables that are used for ExtendedGCD
	ExtendedGCD(a, pow(AlphabetSize, 2), x, y); // X is reverse element of a modulo AlphabetSize^2
	if (x < 0) x += pow(AlphabetSize, 2);
	/*if (a < 0) a += pow(AlphabetSize, 2);
	if (b < 0) b += pow(AlphabetSize, 2);*/
	if (fin.is_open()) {
		while (fin.peek() != EOF) {
			getline(fin, buffer);
			fin.seekg(fin.tellg());
			for (int i = 0; i < buffer.length() - 1; i += 2) {
				EncBigram.push_back(buffer[i]);
				EncBigram.push_back(buffer[i + 1]); // Filling string with a bigram
				EncBigramIntValue = BigramToInt(EncBigram, CharIntMap); // Encrypted bigram in digital form
				DecBigramIntValue = (x * (EncBigramIntValue - b)) % (AlphabetSize * AlphabetSize); // Xi = a^(1-) *
(Yi - b) mod m^2
				// x is reverse element of a modulo AlphabetSize^2
				if (DecBigramIntValue < 0) DecBigramIntValue += pow(AlphabetSize, 2);
				IntToBigram(DecBigramIntValue, DecBigram, IntCharMap);
				fout << DecBigram;
				DecBigram.clear();
				EncBigram.clear();
			}
		}
	}
	else cout << "File opening error" << endl;
	fin.close();
	fout.close();
}

void FindKeys(string FilePath, map<int, string> TopMap, map<int, string> MostFreqBigram, map<char, int> CharIntMap, map<int, int>  Solution) {
	int DecBigramsArr[5] = { 0 };
	int MostFreqBigramsArr[5] = { 0 };
	int i = 0;
	for (auto it = TopMap.cbegin(); it != TopMap.cend(); it++) {
		int a = BigramToInt(it->second, CharIntMap);
		DecBigramsArr[i] = a;
		i++;
	}
```

```
	cout << endl << endl;
	i = 0;
	for (auto it = MostFreqBigram.cbegin(); it != MostFreqBigram.cend(); it++) {
		int a = BigramToInt(it->second, CharIntMap);
		MostFreqBigramsArr[i] = a;
		i++;
	}
	int AlphabetSize = IntCharMap.size();
	int s = 0;
	while (s != 5) {
		int k = 0;
		while (k != 5) {
			int i = 0;
			if (s == k) {
				if (k == 4) break;
				else k++;
			}
			while (i != 5) {
				int j = 0;
				while (j != 5) {
					if (i == j) {
						if (j == 4) break;
						else j++;
					}
					string Y, Y1, X, X1;
					int diffY = DecBigramsArr[s] - DecBigramsArr[k];
					IntToBigram(DecBigramsArr[s], Y, IntCharMap);
					IntToBigram(DecBigramsArr[k], Y1, IntCharMap);
					IntToBigram(MostFreqBigramsArr[i], X, IntCharMap);
					IntToBigram(MostFreqBigramsArr[j], X1, IntCharMap);
					cout << "Y = " << Y << " Y1 = " << Y1 << endl;
					int diffX = MostFreqBigramsArr[i] - MostFreqBigramsArr[j];
					cout << "X = " << X << " X1 = " << X1 << endl;
					if (diffX < 0) diffX += pow(AlphabetSize, 2);
					if (diffY < 0) diffY += pow(AlphabetSize, 2);
					EqSolution(diffY, diffX, pow(AlphabetSize, 2), Solution, MostFreqBigramsArr[i],
DecBigramsArr[s]);
					j++;
				}
				i++;
			}
			k++;
		}
		s++;
	}
	map<char, double> MonoMap;
	double LetterAmount = 0;
	int a = 0, b = 0;
	for (auto it = Solution.cbegin(); it != Solution.cend(); it++) {
		cout << it->first << "		" << it->second << endl;
		a = it->first;
		b = it->second;
		DecryptAffine(FilePath, it->first, it->second, CharIntMap, IntCharMap);
		initMonogramMap("..\\..\\Decrypted Files\\DecryptedFile " + to_string(a) + " " + to_string(b) + ".txt", MonoMap, Alphabet,
LetterAmount);
		if (TextAnalyzer(MonoMap, LetterAmount)) {
			cout << "Correct key: " << " a = " << it->first << " b = " << it->second << endl;
			//return;
		}
		else {
			cout << "Incorrect key: a = " << it->first << " b = " << it->second << endl;
			cout << "Because of Coincidence Index or Entropy differ" << endl;
		}
		LetterAmount = 0;
		MonoMap.clear();

	}
}
```