**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД**

**«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»**

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт

з дисципліни

**«Безпека програм та даних»**

для студентів напрямів підготовки

6.050101 «Комп'ютерні науки»

6.050102 «Комп'ютерна інженерія»

6.050103 «Програмна інженерія»

усіх форм навчання

Красноармійськ-2016

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД**

**«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»**

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт

з дисципліни

**«Безпека програм та даних»**

для студентів напрямів підготовки

6.050101 «Комп’ютерні науки»

6.050102 «Комп’ютерна інженерія»

6.050103 «Програмна інженерія»

усіх форм навчання

Затверджено на Ухвалено на засіданні

засіданні навчально- кафедри прикладної

видавничої ради ДонНТУ математики та інформатики

протокол № \_\_\_\_\_ протокол № 4

від «\_\_\_\_» \_\_\_\_\_\_\_\_\_2016 р. від «24» листопада 2016 р.

Красноармійськ-2015

ББК

32.973.2-018

Методичні вказівки для самостійної роботи на тему «Основи алгоритмізації» до курсу «Програмування». для студентів напрямів підготовки 6.050101 «Комп’ютерні науки», 6.050102 «Комп’ютерна інженерія», 6.050103 «Програмна інженерія» / Укл. Н.О.Маслова, О.А.Золотухіна, О.В.Курило, О.А.Тихонова. - Красноармійськ: ДонНТУ, 2015. - 50 с.

Матеріал, наведений у методичних вказівках, включає відомості з алгоритмізації обчислювальних процесів, що є необхідною складовою здобуття професійних навичок з програмування. Розглянуті основні поняття алгоритмізації, класифікація алгоритмів, надано теоретичний матеріал та приклади рішення типових задач.

Методичні вказівки призначені для самостійного опрацювання бакалаврами напрямів підготовки 6.050101 «Комп’ютерні науки», 6.050102 «Комп’ютерна інженерія», 6.050103 «Програмна інженерія» очної, очно-заочної, заочної форм навчання.

Материал, вошедший в методические указания, включает сведения по алгоритмизации вычислительных процессов, которые являются необходимой составляющей получения профессиональных навыков по программированию. Рассмотрены основные понятия алгоритмизации, классификация алгоритмов, представлен теоретический материал и примеры решения типовых задач.

Методические указания предназначены для самостоятельной работы бакалавров, что обучаются по направлениям подготовки 6.050101 «Компьютерные науки», 6.050102 «Компьютерная инженерия», 6.050103 «Программная инженерия» очной, очно-заочной и заочной форм обучения.

*Рекомендовано Навчалъно-видавничою радою ДонНТУ*

Укладачі:

Н.О.Маслова, доцент, к.т.н., доцент каф.ПМІ,

Т.В. Скрипник, асистент

Відповідальний за випуск:

О.А. Дмитрієва, проф., д.т.н., зав. каф. ПМІ

Рецензент: Є.Є. Федоров, завідувач кафедрою комп’ютерних наук, д.т.н.

**©** Донецький національний

технічний університет, 2016

**Зміст**

[ВСТУП 5](#_Toc445138261)

[Лабораторна рОбота №1. ФормУВАННЯ таблицЬ простИх чисел 6](#_Toc445138262)

[Мета роботи 6](#_Toc445138263)

[Порядок виконання 6](#_Toc445138264)

[Вимоги до звіту: 6](#_Toc445138265)

[Алгоритми пошуку простих чисел 7](#_Toc445138266)

[Варіанти завдань до лабораторної роботи №1 15](#_Toc445138267)

[Лабораторна рОбота №2. АлгоритмИ ОБчисленНя НСК І НСД 16](#_Toc445138268)

[Мета роботи 16](#_Toc445138269)

[Порядок виконання 16](#_Toc445138270)

[Вимоги до звіту: 17](#_Toc445138271)

[Варіанти завдань до лабораторної роботи №2 18](#_Toc445138272)

[Лабораторна робота №3. Алгоритми шифрування 19](#_Toc445138273)

[Мета роботи 19](#_Toc445138274)

[Порядок віконання 19](#_Toc445138275)

[Вимоги до звіту: 19](#_Toc445138276)

[Теоретичні відомості про найпростіші алгоритми шифрування 21](#_Toc445138277)

[Варіанти завдань до лабораторної роботи№3 29](#_Toc445138278)

[Лабораторна робота №4. Криптографічні бібліотеки 30](#_Toc445138279)

[Мета роботы 30](#_Toc445138280)

[Порядок виконання роботи 30](#_Toc445138281)

[Приклади криптографічнихх бібліотек: 31](#_Toc445138282)

[Питання для самоконтролю 31](#_Toc445138283)

[СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 32](#_Toc445138284)

# ВСТУП

Діяльність фахівця в області програмної інженерії – це особливий вид діяльності, спрямований на колективну розробку і супровід складних програмних систем, контакти з замовником, користувачем й постачальником

Особливість роботи сучасного ІТ-спеціаліста є нематеріальність результатів, колективна у рамках проекту й у той же час індивідуальна робота над окремими розділами.

Спеціаліст у галузі програмної інженерії повинен мати навички спілкування з партнерами, замовниками, клієнтами та користувачами. Він має мати діловий вигляд, переконливо вести бесіду, володіти навичками вербального та невербального спілкування.

Одним з інструментів комунікації є візитна картка та бедж, які широко застосовуються на сучасних підприємствах як у друкованому, так і в електронному вигляді.

Спілкування за допомогою Інтернет, вміння знайти й застосувати потрібну інформацію, провести консультацію за допомогою засобів телекомунікації - навички, які потрібні як розробнику, так і персоналу по супроводу або технічному консультанту.

Сучасний спеціаліст повинен вміти грамотно скласти резюме, написати та оформити діловий папір, будь то заява на придбання оргтехніки, або лист діловому партнеру, надати відгук на роботу, що виконувалась по замовленню, оформити технічний звіт та документацію на програмний продукт.

На здобуття перелічених навичок й спрямований лабораторний курс предмету «Групова динаміка та комунікації».

Пропоновані «Методичні вказівки до виконання лабораторних робіт з дисципліни «Групова динаміка і комунікації» рекомендовано застосовувати разом з «Методичним посібником для самостійної роботи студентів з дисципліни «Групова динаміка і комунікації», у якому викладені теоретичні основи для виконання лабораторних робіт курсу.

# Лабораторна рОбота №1. ФормУВАННЯ таблицЬ простИх чисел

## Мета роботи

Освоїти сучасні методи і базові алгоритми формування таблиць простих чисел.

## Порядок виконання

1. Повторити основні терміни арифметики і теорії чисел, що пов’язані з поняттям простого, складеного та взаємно-простого числа, розкладанням числа на множники, кратністю.
2. Провести пошук в Інтернет і в літературних джерелах, знайти поняття та опис способів отримання таблиць простих чисел. В якості базових пропонуються методи простого перебору (простого ділення); алгоритми Ератосфена; Сундарама і Аткіна. Можна запропонувати інший сучасний або самостійно розроблений алгоритм.
3. Скласти таблицю простих чисел для заданого інтервалу, підрахувати кількість цих чисел в інтервалі і визначити обчислювальну складність базового алгоритму (кількість операцій додавання (віднімання), множення (ділення) або зсуву, необхідних для отримання результату), час виконання розрахунку.
4. З метою підвищення ефективності метода, запропонувати модифікацію базового алгоритма, орієнтуючись на найпощиреніші критерії (швидкодія, пам’ять, кількість обчислювальних операцій, надійність).
5. Підтвердити ефективність модификації та відобразити це у звіті.

## Вимоги до звіту:

1. Титульний лист
2. Вступ (про застосування простих чисел у захисті інформації);
3. Основний розділ, в якому:

* дати визначення всім перерахованим у п.1 поняттям (просте число, складене число, число, що кратне заданому і т.д.);
* описати алгоритм і відобразити блок-схему формування таблиці простих чисел базовим методом (у відповідності до варіанта);
* надати код базового алгоритму (вибір мови - вільний);
* показати результати роботы програми (таблиця простих чисел у заданому інтервалі; їх кількість; дані, що підтверджують обчислювальну складність методу) – таблиці, копії екранів;
* описати сутність пропонованої модификації та схему нового алгоритму;
* програмний код модифікованого алгоритму та результати, що підтверджують його ефективність;

1. Сформулювати висновки.

## Алгоритми пошуку простих чисел

Решето Ератосфена

Цілком ймовірно, що алгоритм, придуманий більше 2000 років тому грецьким математиком Эратосфеном Киренским, був першим у своєму роді. Його єдина задача – знаходження всіх простих чисел до деякого заданого числа N. Термін «решето» передбачає фільтрацію, а саме фільтрацію всіх чисел за винятком простих. Так, обробка алгоритмом числової послідовності залишить лише прості числа, всі ж складені числа відсіються.

Розглянемо у загальних рисах роботу методу. Дана упорядкована за зростанням послідовність натуральних чисел. Дотримуючись методу Ератосфена, візьмемо деяке число P спочатку дорівнююче 2 – першому простому числу, і викреслимо з послідовності всі числа кратні P: 2P, 3P, 4P, ..., iP (iP≤N). Далі, з отриманого списку в якості P береться наступне за двійкою число – трійка, викреслюються всі кратні їй числа (6, 9, 12, ...). За таким принципом алгоритм продовжує виконуватися для решти послідовності, відсіваючи всі складені числа в заданому діапазоні.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |  | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |  | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 |  | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |  | 96 | 97 | 98 | 99 | 100 |

У наведеній таблиці записані натуральні числа від 2 до 100. Червоним кольором позначені ті, які видаляються в процесі виконання алгоритму «Решето Ератосфена».

Програмна реалізація алгоритму Ератосфена вимагає:

1. організувати логічний масив і надати його елементам з діапазону від 2 до N значення логічної одиниці;
2. у вільну змінну P записати число 2, що є першим простим числом;
3. видалити з масиву всі числа кратні P2, ступаючи з кроком по P;
4. записати в P наступне за ним не закреслене число;
5. повторювати дії, описані у двох попередніх пунктах, поки це можливо.

Зверніть увагу: на третьому кроці ми виключаємо числа, починаючи відразу з P2, це пов'язано з тим, що всі складені числа менші за P будуть вже закреслені. Тому процес фільтрації слід зупинити, коли P2 стане перевищувати N. Це важливе зауваження дозволяє поліпшити алгоритм, зменшивши число виконуваних операцій.

Так буде виглядати псевдокод алгоритму:

P←2  
поки P2≤N виконувати

{  
i←P2  
якщо B[P]=true то

поки i≤N виконувати

{  
B[i]←false  
i←i+P  
}  
P←P+1  
}

 Він складається з двох циклів: зовнішнього і внутрішнього. Зовнішній цикл виконується доти, поки P2 не перевищить N. Саме ж P змінюється з кроком P+1. Внутрішній цикл виконується лише в тому разі, якщо на черговому кроці зовнішнього циклу виявиться, що елемент з індексом P не закреслений. Саме у внутрішньому циклі відбувається відсіювання всіх складених чисел.

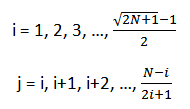
Решето Ератосфена для виявлення всіх простих чисел в заданій послідовності обмеженою деяким N потребує O(Nlog (log N) операцій. Тому доречніше використовувати даний метод ніж, наприклад, найбільш тривіальний і витратний перебір дільників.

Решето Сундарама

Решето Сундарама – алгоритм пошуку всіх простих чисел в деякому заданому діапазоні. Він був розроблений в 1934 році нині невідомим студентом з Індії С. П. Сундарамом.

Принцип роботи алгоритму Сундарама зводиться, як і в його знаменитого попередника, до послідовного відсіювання всіх непотрібних чисел. Але у нього є одна невелика особливість: результатом роботи алгоритму буде послідовність простих чисел з діапазону від 2 до подвоєного значення граничного числа. Припустимо необхідно отримати всі прості числа до деякого N, тоді вихідними даними будуть всі прості числа від 2 до 2N+1.

Решето Сундарама з ряду натуральних чисел, що не перевищують N, виключає числа виду 2ij+i+j. Результат даного виразу, ні при яких значеннях вхідних у нього змінних, що не перевищує N (2ij+i+j≤N). Дотримуючись цієї умови, а також того, що i завжди менше або дорівнює j, змінні i та j пробігають всі натуральні значення з множин:

[](http://kvodo.ru/wp-content/uploads/SieveOfSundaram.png)

Після виключення всіх непотрібних чисел необхідно збільшити кожне залишене число в два рази і додати одиницю (2i+1). У підсумку множина міститиме числа: 2, 3, …, 2N+1.

Решето Аткіна.

Алгоритм складається з наступних кроків:

1. Виписуються всі натуральні числа з діапазону від **1** до **n**.

2. Перебираються всі можливі пари чисел **x**, **y**, де **x<=sqrt(n)** і **y<=sqrt(n)**. Тобто (1,1), (1,2),…, (1,sqrt(n)), (2,1), (2,2),…, (sqrt(n),sqrt(n)).

3. Для кожної пари чисел розраховуються значення наступних трьох рівнянь:

a) **4\*x^2+y^2**;

b) **3\*x^2+y^2**;

c) **3\*x^2-y^2**, значення обчислюється тільки при **x>y**.

4. Для кожного обчисленого значення рівнянь розраховуються залишки від ділення на **12**, причому

a) якщо залишок дорівнює **1** або **5** для значення першого рівняння;

b) якщо залишок дорівнює  **7** для значення другого рівняння;

с) якщо залишок дорівнює  **11** для значення третього рівняння.

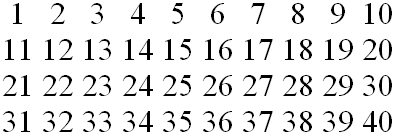
То у вхідному ряду чисел від **1** до **n** число позначається як просте.

***Зауваження****: якщо якесь число Z присутнє в декількох значеннях рівнянь (припустимо a і b), і залишок від ділення на 12 цього числа задовольняє умовам обох груп, то число позначається двічі: спочатку як просте, а потім як складне.*

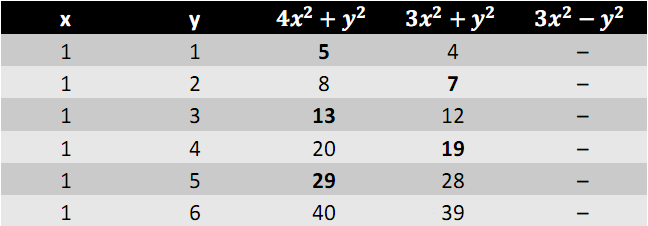
5. На останньому етапі перевіряється кратність позначених чисел квадратам простих чисел з діапазону від 5 до sqrt(n). Якщо число кратне квадрату, то воно позначається як складене.

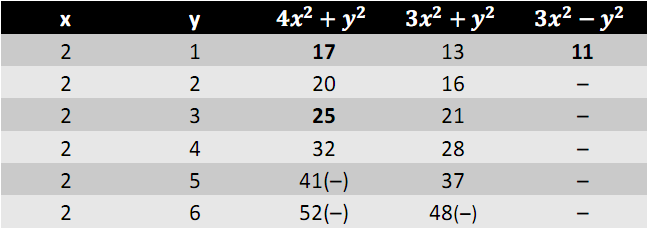
Приклад роботи алгоритму для **n = 40**.

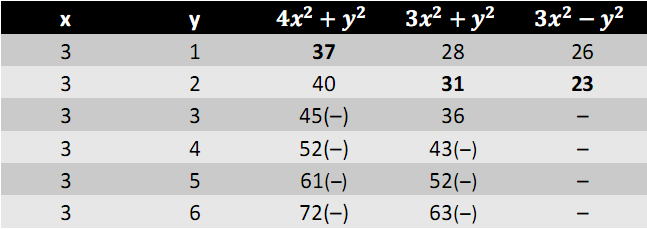
1. Виписуємо всі натуральні числа з діапазону від **1** до **40**.

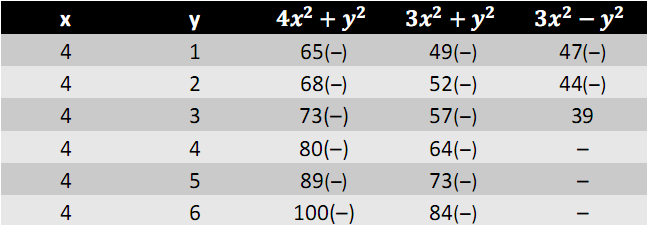
[](http://cybern.ru/atkin.html/ris1-4)

1. Перебираємо всі можливі пари чисел від (1,1) до (6,6) (Тому що sqrt(40) ~ 6).
2. Обчислюємо значення рівнянь для кожної пари чисел x та y.

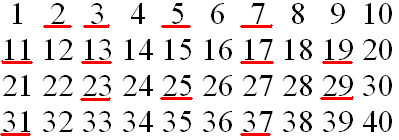
[](http://cybern.ru/atkin.html/pary-1-6)

[](http://cybern.ru/atkin.html/pary-2-6)

[](http://cybern.ru/atkin.html/pary-3-6)

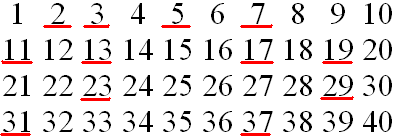
[](http://cybern.ru/atkin.html/pary-4-6)

1. Обчислюємо залишок від ділення на **12** і позначаємо прості числа.

[](http://cybern.ru/atkin.html/ris2-4)

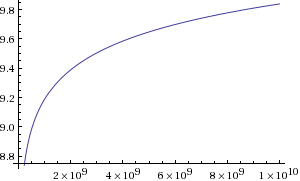
1. Перевіряємо кратність позначених чисел квадратам простих з діапазону від 5 до 6.

5 — просте число, 6 — складене, значить перевіряємо на кратність 52=25 позначені числа. У результаті 25 — потрібно викреслити. У підсумку залишаються тільки прості числа від 1 до 40.

[](http://cybern.ru/atkin.html/ris3-3)

Аналіз алгоритма:

Алгоритм має асимптотичну складність O(N/ln(ln(N)) і потребує біт пам’яті. На вхідних значеннях порядку 109 решето Аткіна швидче решета Ерастофена у 9.2 рази. Наведемо графік зростання переваги алгоритму Аткіна на числах від 2 до 109:



В результаті можна спостерігати наступну швидкість виконання:

|  |  |
| --- | --- |
| 10 000 000 | 0.15 сек. |
| 100 000 000 | 2.16 сек. |
| 1 000 000 000 | 48.76 сек. |

Алгоритм простого ділення

*Метод простого ділення* **(**<http://algolist.manual.ru/maths/teornum/factor/trial.php>). Практично при будь-якому алгоритм факторизації оптимально використовувати цей спосіб до певної межі B, щоб прибрати малі дільники. Найбільш наївний підхід - просто ділити на всі числа підряд до кореня з N.

Нехай потрібно поділити N на всі прості числа до кореня з N. Для цього ми можемо мати або не мати у своєму розпорядженні досить велику таблицю простих чисел. Якщо це не так, то, очевидно, ми можемо ділити числа N на числа d у заданих класах еквівалентності, наприклад, 1 і 5 за модулем 6, або 1,7,11,13,17,19,23,29 по модулю 30. Тоді ми зробимо багато безглуздих ділень (на складені числа), однак результат залишиться вірним. Таким чином, можна скласти наступний алгоритм:

Припустимо, що в нас вже є таблиця простих чисел: p[1] = 2, p[2] = 3, ... , p[k], де k > 3, масив t := [6,4,2,4,2,4,6,2], та індекс j, такий що якщо p[k] mod 30 дорівнює 1,7,11,13,17,19,23 або 29, то j := 0,1,2,3,4,5,6 або 7 відповідно.

Оберемо деяку верхню грань B, таку що B >= p[k], щоб не витрачати на цей алгоритм надто багато часу.

Отримавши на вході додатне ціле число N, цей алгоритм намагається розкласти його на множники, і, якщо це не вдається, то N - число без простих дільників, менших або дорівнюючих B.

1. [Ініціалізація]

Якщо N <= 5, вивести розкладання 1 = 1, 2 = 2, 3 = 3, 4 = 22, 5 = 5 в залежності від N та вийти. Інакше, i := -1, m := 0, L := [ N1/2]

1. [Наступне просте]

Нехай m := m + 1. Якщо m > k, то i : = j - 1 та йти до кроку 5, інакше d=p[m].

1. [Просте ділення]

Нехай r : = N mod d. Якщо r = 0, то вивести d як нетривіальний дільник N та вийти ( або N := N / d, L : = [ N1/2 ] і повторити крок 3, якщо хочемо продовжити знаходження дільників N).

1. [Просте?]

Якщо d >= L, то у випадку N > 1 вивести повідомлення, що частка N, що лишилася – просте число і вийти. Інакше, якщо i < 0 - перейти на крок 2.

1. [Наступний дільник]

Нехай i := i + 1 mod 8, d := d + t[i]. Якщо d > B, то вивести повідомлення про те, що прості дільники N, що лишилися більше B, інакше перейти на крок 3.

Слід зазначити, що i = -1 поки використовується таблиця простих чисел, та i >= 0 якщо ні.

Метод дуже корисний для видалення малих множників, проте його не слід використовувати для повної факторизації, окрім тих випадків, коли N - дуже маленьке ( наприклад, N < 108).

Особливості практичної реалізації

Припустимо, що ми отримуємо таблицю простих до 500'000 ( 41'538 простих чисел ).Просте ділення при такому обмеженні ніколи не займає більше кількох секунд на сучасних комп'ютерах. Причому зберігати можна тільки різниці між простими (або їхні половини), а не самі числа, тому що p[k] - p[k-1] можна помістити в 1 байті замість чотирьох, якщо p[k] <= 1'872'851'947 ( а половину цієї різниці - якщо p[k] <= 1'999'066'711'391).

Крім того, не потрібно робити більше ділень після того, як скінчиться таблиця простих, бо є найкращі методи для видалення малих дільників.

І, нарешті, помітно, що немає необхідності в обчисленні L := [ N1/2] під час ініціалізації, тому що перевірка d >= L на кроці 4 може бути замінена на q <= L, де q - евклідова частка при діленні N на d : N = q \* d + r, зазвичай обчислюється одночасно з залишком на кроці 3.

## Варіанти завдань до лабораторної роботи №1

Таблиця 1.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п\п** | **m** | **n** | **Базовий алгоритм** |
| 1 | 500 | 894 | перебору |
| 2 | 700 | 1063 | Ератосфена |
| 3 | 550 | 1241 | Сундарама |
| 4 | 900 | 1387 | Аткіна |
| 5 | 1000 | 2384 | перебору |
| 6 | 1250 | 1625 | Ератосфена |
| 7 | 750 | 1141 | Сундарама |
| 8 | 1500 | 2131 | Аткіна |
| 9 | 1350 | 1289 | перебору |
| 10 | 700 | 1368 | Ератосфена |
| 11 | 1650 | 2453 | Сундарама |
| 12 | 1050 | 1765 | Аткіна |
| 13 | 1150 | 1717 | перебору |
| 14 | 600 | 1659 | Ератосфена |
| 15 | 750 | 1561 | Сундарама |
| 16 | 1900 | 2386 | Аткіна |
| 17 | 850 | 2745 | перебору |
| 18 | 1800 | 2821 | Ератосфена |
| 19 | 1100 | 1466 | Сундарама |
| 20 | 1550 | 1666 | Аткіна |
| 21 | 650 | 777 | перебору |
| 22 | 1200 | 1555 | Ератосфена |
| 23 | 1750 | 2461 | Сундарама |
| 24 | 1400 | 2357 | Аткіна |
| 25 | 300 | 999 | перебору |
| 26 | 800 | 1221 | Ератосфена |
| 27 | 1450 | 1723 | Сундарама |
| 28 | 1300 | 1611 | Аткіна |
| 29 | 1600 | 2555 | перебору |
| 30 | 950 | 1745 | Ератосфена |

# Лабораторна рОбота №2. АлгоритмИ ОБчисленНя НСК І НСД

## Мета роботи

Засвоїти сучасні методи і базові алгоритми знаходження найбільшого спільного дільника (НСД) і найменшого спільного кратного (НСК)

## Порядок виконання

1. Повторити основні поняття теорії чисел, що пов’язані із знаходженням найменшого спільного кратного і найбільшого спільного дільника.
2. Знайти матеріал на тему «Алгоритми обчислення НСК і НСД та їх застосування при захисті інформації».

В материале:

- дати визначення всім переліченим в п.1 поняттям

- показати використання НСК і НСД в захисті інформації;

- знайти і проаналізувати не менш двох методів (алгоритмів) обчислення НСК і НСД;

- розглянути способи підвищення ефективності роботи алгоритма обчислення НСД за одним з критеріїв (швидкодія, пам’ять, кількість обчислювальних операцій).

3. Для алгоритму визначення НСД, що вказаний у Вашому варіанті завдання виконати наступне:

* описати схему визначення НСД і скласти алгоритм та програму обчислення НСД базовим методом, передбачивши в якості вихідного параметра вивід значення критерію, за яким передбачається проведення оптимізації;
* запропонувати і обґрунтувати модифікацію методу, вказавши переваги цієї модифікації.
* скласти програму для модифікованого алгоритму, передбачивши в якості вихідного параметра вивід значення критерію, за яким проводилася оптимізація;
* знайти значення НСК і НСД для зазначених у варіанті завдань значень n, m і показати на числовому прикладі ефективність отриманих результатів.

## Вимоги до звіту:

1. Титульний лист
2. Вступ
3. Матеріал на тему «Алгоритми обчислення НСК і НСД та їх застосування при захисті інформації».
4. Блок-схему і опис базового алгоритму обчислення НСД (у відповідності до варіанта завдання)
5. Блок-схему і опис модифікованого алгоритму
6. Розраховані значення НСК і НСД для заданих чисел
7. Матеріали (таблицю), що підтверджують ефективність модификації
8. Висновки

## Варіанти завдань до лабораторної роботи №2

Таблиця 2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п\п** | **m** | **n** | **Базовий алгоритм** |
| 1 | 500 | 894 | Алгоритм Евкліда відніманням |
| 2 | 700 | 1063 | Алгоритм Евкліда діленням |
| 3 | 550 | 1241 | Алгоритм Евкліда для цілих чисел |
| 4 | 900 | 1387 | Геометричний алгоритм Евкліда |
| 5 | 1000 | 2384 | Розширений алгоритм Евкліда |
| 6 | 1250 | 1625 | Бінарний алгоритм обчислення НСД |
| 7 | 750 | 1141 | Алгоритм Евкліда відніманням |
| 8 | 1500 | 2131 | Алгоритм Евкліда діленням |
| 9 | 1350 | 1289 | Алгоритм Евкліда для цілих чисел |
| 10 | 700 | 1368 | Геометричний алгоритм Евкліда |
| 11 | 1650 | 2453 | Розширений алгоритм Евкліда |
| 12 | 1050 | 1765 | Бінарний алгоритм обчислення НСД |
| 13 | 1150 | 1717 | Алгоритм Евкліда відніманням |
| 14 | 600 | 1659 | Алгоритм Евкліда діленням |
| 15 | 750 | 1561 | Алгоритм Евкліда для цілих чисел |
| 16 | 1900 | 2386 | Геометричний алгоритм Евкліда |
| 17 | 850 | 2745 | Розширений алгоритм Евкліда |
| 18 | 1800 | 2821 | Бінарний алгоритм обчислення НСД |
| 19 | 1100 | 1466 | Алгоритм Евкліда відніманням |
| 20 | 1550 | 1666 | Алгоритм Евкліда діленням |
| 21 | 650 | 777 | Алгоритм Евкліда для цілих чисел |
| 22 | 1200 | 1555 | Геометричний алгоритм Евкліда |
| 23 | 1750 | 2461 | Розширений алгоритм Евкліда |
| 24 | 1400 | 2357 | Бінарний алгоритм обчислення НСД |
| 25 | 300 | 999 | Алгоритм Евкліда відніманням |
| 26 | 800 | 1221 | Алгоритм Евкліда діленням |
| 27 | 1450 | 1723 | Алгоритм Евкліда для цілих чисел |
| 28 | 1300 | 1611 | Геометричний алгоритм Евкліда |
| 29 | 1600 | 2555 | Розширений алгоритм Евкліда |
| 30 | 950 | 1745 | Бінарний алгоритм обчислення НСД |

# Лабораторна робота №3. Алгоритми шифрування

## Мета роботи

Засвоїти найпростіші алгоритми шифрування і отримати навички їхнього використання.

## Порядок віконання

1. Ознайомитися з теоретичним матеріалом до лабораторної роботи.
2. Використовуючи запропонований в таблиці варіантів алгоритм, скласти блок-схему і програму шифрування та дешифрування повідомлень одним з базових методів.
3. Запропонувати модифікацію алгоритма, що покращує його криптографічну стійкість (наприклад, додавання ключа, використання «складеного» алфавіту, іншої кодової таблиці і др.) або підвищує ефективність (швидкодія, пам’ять, кількість обчислювальних операцій).
4. Виконати шифрування довільного, обраного студентом повідомлення.
5. Виконати дешифрування отриманого в п.3. повідомлення, порівняти з вхідним.
6. Проаналізувати основні характеристики базового і модифікованого алгоритму.

## Вимоги до звіту:

1. Титульний лист
2. Вступ (про основні алгоритми шифрування);
3. Основний розділ, в якому:

- словесно описати схему отримання зашифрованого повідомлення, пояснити основні особливості;

- скласти алгоритм (відобразити блок-схему та програму роботи базового алгоритму (вибір мови - довільний);

- описати запропоновану модифікацію алгоритму, відобразити алгоритмічно і програмно;

- показати результати порівняння алгоритмів і роботи програми, зашифрувавши, а потім і розшифрувавши повідомлення;

1. Формулювати висновки.

## Теоретичні відомості про найпростіші алгоритми шифрування

Наука про шифри отримала назву криптологія, слово утворене з двох грецьких: "criptos" - таємний і "logos" - повідомлення (слово). З самого початку криптологія включала дві взаємодоповнюючі гілки: криптографію, в якій вивчалися методи шифрування повідомлень, та криптоаналіз, де розроблялися методи розкриття шифрів.

криптографія (розробка шифрів)

КРИПТОЛОГІЯ

криптоаналіз (розкриття шифрів)

Основоположною роботою сучасної криптології вважається робота американського вченого Клода Шеннона "Теорія зв'язку в секретних системах", що опублікована в 1949 році. К.Шеннон розглядає шифрування як відображення вхідного повідомлення в зашифроване

С= FiM,

де С – криптограма, Fi – відображення, М – початковий стан. Індекс i відповідає конкретному використовуваному ключу. Для того, щоб була можливість однозначного дешифрування повідомлення відображення Fi  повинно мати зворотнє відображення. Тоді

M=Fi-1С.

Шифр - це безліч зворотних перетворень форми повідомлення з метою його захисту від несанкціонованого прочитання.

Шифри, орієнтовані на операції з машинними словами називаються блоковими, а ті, які передбачають послідовну посимвольную обробку інформації – потоковими шифрами.

Початкове повідомлення, яке підлягає шифруванню, називається відкритим текстом, а результат, отриманий через застосування перетворення шифру до вихідного повідомлення, називається шифротекстом або криптограммой.

Принцип побудови перетворення шифру завжди припускає безліч варіантів його реалізації, а для конкретних випадків використання шифру вибирається цілком певний варіант. Сукупність даних, що визначають конкретне перетворення шифру з безлічі можливих, називається ключем.

Стійкість шифру - це здатність протистояти спробам сторонньої особи відновити (дешифрувати) відкритий текст за перехваченим шифротекстом. В цих спробах криптоаналітик спочатку намагається вгадати принцип побудови шифру, а потім визначити ключ.

Стійкість шифрів оцінюється за необхідним противнику часом, який озброєний сучасними засобами обчислювальної техніки будь яким чином (наприклад, повним перебором варіантів) намагається дешифрувати повідомлення. Чим більше варіантів ключів можливо, тим більш важким для дешифрування є шифр.

Спочатку криптографія вивчала методи шифрування інформації — зворотного перетворення відкритого (вхідного) тексту на основі секретного алгоритму або ключа в шифрований текст (шифротекст). Традиційна криптографія утворює розділ симетричних криптосистем, в яких зашифрування і розшифрування проводиться з використанням одного і того ж секретного ключа. Окрім цього розділу сучасна криптографія включає в себе асиметричні криптосистеми, системи електронного цифрового підпису (ЕЦП), хеш-функції, керування ключами, отримання прихованої інформації, квантову криптографію.

Базові алгоритми шифрування

До механізмів найпростіших шифрів відносяться заміна, перестановка і дроблення.

Заміна. При цьому кожна літера відкритого тексту замінюється на якусь букву алфавіту (можливо, на ту ж саму). Для цього відправник повідомлення повинен знати, на яку букву в шифротексті слід замінити кожну літеру відкритого тексту. Часто це робиться шляхом зведення потрібних відповідностей літер у вигляді двох алфавітів. Шифрограма виходить шляхом заміни кожної букви відкритого тексту на записану безпосередньо під нею букву шифрувального алфавіту.

Наприклад, наступне повідомлення отримано шифром заміни (тільки літер)

**О н р ъ к я э \_ 1й в \_ б ж п ъ б ц я с й з.**

Сказати про нього щось однозначне важко, бо замало інформації. Але ключ тут дуже простий, досить побачити відкритий текст.

**П о с ы л а ю \_1кг \_ в з р ы в ч а т к и.**

Кожна буква тут замінена на її попередницю в алфавіті ("о" замість "п", "н" замість "o", "p" замість "c", "к" замість "л", "я" замість "а" тощо). Це один з варіантів так званого шифру Цезаря. Достовірно відомо, що Ю. Цезар застосовував шифр заміни зі зсувом на три букви вперед при шифруванні.

Більш формальний підхід до дешифрування заснований на використанні середньої частоти появи літер у текстах. Вперше метод був запропонований наприкінці 15-го століття італійським математиком Леоном Баттіста Альберті. Він використовував властивість нерівномірності зустрічальності різних букв алфавіту. Пізніше були визначені середні частоти використання літер мови в текстах. Деякі з них наведені в таблиці 3.1.

Таблиця 3.2. Частота використання літер (%)

|  |  |
| --- | --- |
| Англійська мова | **Е** (12,9) **Т** (9,7) **А** (8,0) **I**  (7,5) **N** (7,0) **R** (7,0) |
| Німецька мова | **E**  (19,2) **N** (10,2) **I** (8,2) **S** (7,0) **R** (7,0) **T** (5,9) |
| Російська мова | **O** (11,0) **И** (8,9) **Е** (8,3) **А** (7,9) **Н** (6,9) **Т** (6,0) |

Тепер, маючи шифротексти, можна було провести в них частотний аналіз використання символів та на його основі одержати (при необмеженій кількості шифроповідомлень) точні значення всіх букв. Але оскільки матеріалу, як правило, не дуже багато, то частотний аналіз дає приблизні результати, тому можна лише з високою часткою ймовірності припустити буквене значення найчастіше вживаних символів, а далі - необхідно підставляти їх в шифротексти і намагатися вгадувати слова, однак результати з'являються досить швидко. На слабкість шифрів однозначної заміни звернули увагу ще в 15-му столітті. Випадкові припущення - хто і кому пише, назви міст і селищ, часто вживані слова, начебто прийменників, могли призвести до майже миттєвого розкриття шифру. Спроби модифікації ґрунтувалися на багатозначній заміні букв відкритого тексту з використанням ключової послідовності (ключового слова або ключа).

В найбільш чистому вигляді цей підхід можна викласти так. Нехай ми хочемо отримати 10 варіантів (0, 1, ..., 9) заміни кожної букви вхідного тексту (в таблиці 2 варіант заміни визначає величину зсуву за алфавітом).

Придумаємо ключову послідовність цифр 0...9 довільної довжини (наприклад, 190 277 321 856 403). Для відкритого тексту надпишемо над літерами цифри ключа (періодично) і виконаємо зашифрування, вибираючи варіант заміни за цифрою ключа. Добре видно, що одні і ті ж букви замінюються по різному, а різні літери можуть бути представлені однаково:

1 9 0 2 7 7 3 2 1 8 5 6 4 0 3 1 9 0 2 7 7 3 2 1 8 5 6 ...

Н а ш а \_ Т а н я \_ г р о м к о \_ п л а ч е т , урон и л а ...

Ойшв \_ щ з р б \_ д шутоо \_ т м кчзщ, ыуроррж ...

###### Таблиця 3.3 Таблиця варіантів заміни

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Варіант  Буква | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| А | а | б | в | г | д | е | ж | з | и | й |
| Б | б | в | г | д | е | ж | з | и | й | к |
| В | в | г | д | е | ж | з | и | й | к | л |
| … | … | … | … | … | … | … | … | … | … | … |
| Я | я | а | б | в | г | д | е | ж | з | и |

Ідея фактично була запропонована в 16-му столітті французьким дипломатом Блезом де Вижинером. Замість цифр їм використовувалися літери, і ключова послідовність являла собою слово.

Легко бачити, що алгоритм визначає сукупність перетворень шифру, що відрізняються параметром - ключовою послідовністю шифрування (ключем). Це дозволяє будувати надійну криптосистему на підставі фіксованого (несекретного) алгоритму шифрування, але секретного ключа, який регулярно змінюється. Теоретично такий шифр піддається дешифрованию на основі частотного аналізу вживання букв, але для цього потрібно, щоб довжина шифроповідомлень, зроблених з цим ключем, значно перевищувала довжину самого ключа.

Перестановка. У шифрі перестановки всі букви відкритого тексту залишаються без змін, але переставляються за заздалегідь обумовленим правилом. Тут також може використовуватися ключ, керуючий процедурою шифрування. Ключове слово може бути використане для отримання шифрованої числової послідовності шляхом нумерації букв ключового слова (відносно одна одної) в порядку їхнього проходження зліва направо в стандартному алфавіті. Далі під числовою послідовністю в рядках, рівних за довжиною ключовому слову, записаний відкритий текст. У процесі шифрування текст виписується вже за окремими стовпцями в порядку, визначеному даною числовою послідовністю..

Метод перестановки частіше полягає в перестановці стовпців, але можна обрати й інші «маршрути» перестановки, наприклад виписувати шифротекст слідуючи по діагоналі (зліва направо або справа наліво, або ж почергово ліве і праве напрями) або по спіралі і т. п. Крім того, букви шіфротекста можуть записуватися у вигляді різних геометричних фігур або будь-якими іншими способами. Один з них полягає в подвійному шифруванні шляхом повторної перестановки стовпців. При цьому і в першому, і в другому блоках перестановки може бути використано одне і те ж ключове слово, хоча краще використовувати різні ключові слова (подвійна перестановка).

Найпростіший варіант перестановки - прямокутна таблиця з секретним розміром стовпця (рис. 3.1), куди вхідний текст записується по стовпцях, а шифроповідомлення зчитується за рядками.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| П | л | к | З | ч | и |
| о | а | Г | Р | а | # |
| с | ю | - | Ы | т | # |
| ы | - | В | В | к | # |

Рис. 3.1. Шифр табличної перестановки

Відкритий текст: Посылаю \_ кг \_ взрывчатки###.

Шифроповідомлення:**Плкзчиоагра#сю\_ыт#ы\_ввк#.**

‘#’ - довільні символи

Для розшифрування треба довжину повідомлення розділити на довжину стовпчика, щоб визначити довжину рядка, вписати шифроповідомлення в таблицю по рядках, а потім прочитати відкритий текст.

Інший варіант - кодування перестановкою по групах символів, використовуючи деякі зигзагоподібні шаблони, наприклад, як показано на рисунку 3.2.

1 7 9 3 5 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| П | ю | к | с | л | \_ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \_ | а | о | г | в | ы |

8 6 2 10 12 4

Рис. 3.2. Зигзагоподібний шифр перестановки

Відкритий текст: Посылаю\_кг\_взрывчатки###

Шифроповідомлення:**Пюксл\_ывгоа\_ зтиыч#в##рак**

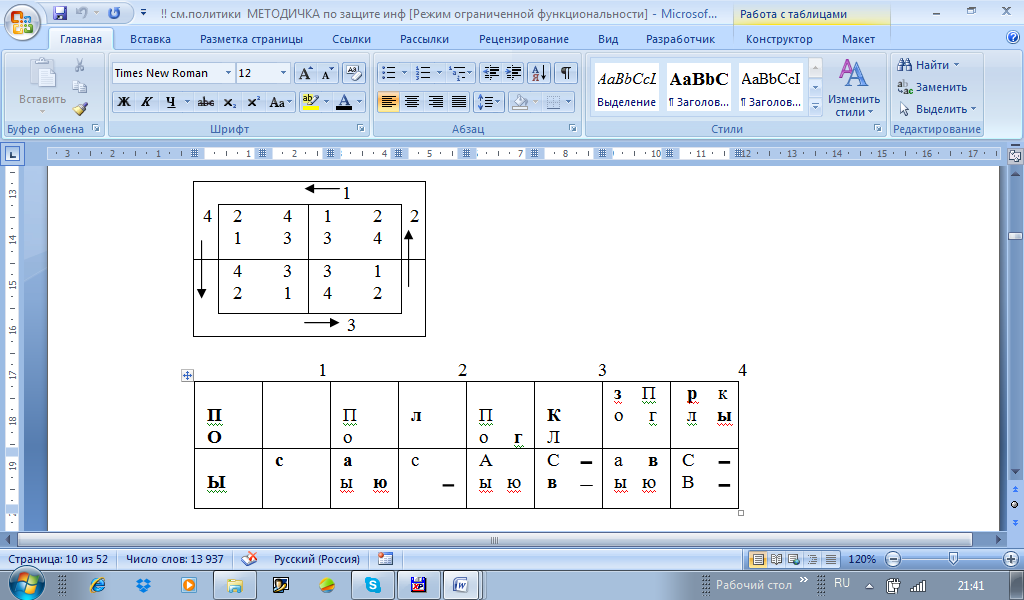
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

группа 1 группа2

‘#’- довільні символы.

Варто записати символи відкритого тексту за зигзагом, а прочитати по колу (або навпаки) - і шифоповідомлення готове, якщо здається ненадійним, то можна ввести додаткові ускладнення.

Використовуються і більш складні (ручні) системи, а також групові. Наприклад, у квадраті (мал. 4), що складається з 4 малих квадратів з певною нумерацією клітин, вирізують 4 клітини під різними номерами. Квадрат кладеться в початкове положення («1» - згори) і в отвори (зліва направо/зверху вниз) вписуються букви відкритого повідомлення. Потім квадрат повертається проти годинникової стрілки на 90 градусів («2» - згори) і знов вписуються наступні літери, потім повторюємо процес для положення «3» і «4». Якщо залишаються вільні клітини - вони заповнюються довільними символами. Шифр повідомлення отримують, прочитавши по стовпцях або рядках послідовність записаних у прямокутнику букв.



Посылаю\_кг\_взрывчатки

ЗоаыПгвюрлСВкы\_ \_

группа\_1 группа\_2

Рис.4. Шифрувальний квадрат

Фактично, розглянуті вище та інші шифри перестановки з сучасної точки зору абсолютно подібні, бо представляють собою послідовність елементарних процедур перестановки групи символів виду

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| П | О | с | ы | л | а | ю |  | к | Г |  | в | з | р | ы | в |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 2 | 6 | 4 | 1 | 10 | 16 | 7 | 14 | 5 | 3 | 12 | 9 | 15 | 11 | 8 |
| з | О | а | ы | п | г | в | Ю | р | Л | с | в | к | ы | \_ | \_ |

Дешифрування повідомлень, отриманих шифром перестановки, значно важче, ніж при використанні шифрів заміни. Теоретичної передумови, крім перебору варіантів, не існує, хоча окремі здогадки можуть спростити завдання.

Дроблення. При цьому кожній букві відкритого тексту зіставляється більше одного символу шифротексту, після чого символи перемішуються (переставляються) у певному порядку. Нижче наведена система, що демонструє процедуру дроблення з використанням знаменитого шифру Bifid, авторство якого приписується французькому криптографу Феліксу Марі Деластеллю.

Спочатку складається шифрувальна таблиця розміром 5x5 (т. зв. полібіанський квадрат), куди порядково вписується шифрувальний алфавіт з ключовою фразою, причому заради того, щоб загальне число букв алфавіту не перевищувало 25, літера J опускається (оскільки ця буква, з одного боку, маловживана в англійських текстах, а з іншого – цілком може бути замінена буквою I, без будь-якої шкоди для змісту).

Далі в процесі шифрування під кожною буквою відкритого тексту записуються в стовпчик її табличні координати – номер рядка, а нижче номер стовпця, а потім отримана цифрова послідовність перекладається за допомогою тієї ж таблиці назад в буквену форму, але на цей раз вона читається вже в рядок. При такому шифруванні координата рядка і координата стовпця кожної букви виявляються роз'єднаними, що характерно саме для роздроблюючого шифру.

## Варіанти завдань до лабораторної роботи№3

Таблиця 3.1.

|  |  |  |
| --- | --- | --- |
| **№ п\п** | **Базовий алгоритм** | **Особливості ключа, метод** |
| 1 | Однозначна заміна | зсув на дві букви вперед |
| 2 | Перестановка | довільна числова послідовність |
| 3 | Дроблення | полібіанський квадрат 6\*6 |
| 4 | Багатозначна заміна | кількість стовпчиків=7 |
| 5 | Перестановка | Маршрут перестановки-діагональ справа наліво |
| 6 | Дроблення | з використанням шифру Bifid на 5 символів |
| 7 | Однозначна заміна | зсув на попередню букву |
| 8 | Перестановка | За довільним шаблоном |
| 9 | Дроблення | шифр Bifid на 6 символів |
| 10 | Однозначна заміна | Схема Цезаря |
| 11 | Перестановка | Таблична 9\*4 |
| 12 | Дроблення | полібиіанський квадрат 5\*5 |
| 13 | Багатозначна заміна | кількість стовпчиків=8 |
| 14 | Однозначна заміна | зсув на 5 символів вперед |
| 15 | Перестановка | Шифр – таблична перестановка |
| 16 | Однозначна заміна | «зворотній алфавіт» |
| 17 | Перестановка | Шифрувальний квадрат |
| 18 | Однозначна заміна | з урахуванням середньої частоти появи букв |
| 19 | Перестановка | Перестановка за заданою групою символів |
| 20 | дроблення | Розмір шифрувальної таблиці – 6\*6 |
| 21 | Багатозначна заміна | Перестановка за групами символів |
| 22 | Перестановка | Зигзагоподібний шифр |
| 23 | Однозначна заміна | зсув на попередню букву |
| 24 | Дроблення | полібіанський квадрат 6\*6 |
| 25 | Перестановка | Шифрувальний квадрат (нумерація клітин – задається довільно) |
| 26 | Однозначна заміна | зсув «через п’ять» |
| 27 | Багатозначна заміна | «маршрут» перестановки – ліво-право |
| 28 | Перестановка | Таблична 8\*4 |
| 29 | дроблення | Три символи замість одного |
| 30 | Однозначна заміна | зсув «через п’ять» |

# Лабораторна робота №4. Криптографічні бібліотеки

## Мета роботы

Ознайомитися з існуючими криптографічними бібліотеками. Навчитися використовувати сторонні криптографічні бібліотеки при розробці власних програмних додатків.

## Порядок виконання роботи

1. **Знайти в мережі** Інтернет повнофункціональну бібліотеку криптографічних алгоритмів (або вибрати криптографічну бібліотеку, вбудовану в мову програмування, яку планується використовувати).

Вимоги до функцій бібліотеки - можливість виконувати наступні чотири типи крипто - перетворення:

-шифрування й дешифрування симетричними алгоритмами;

- шифрування і дешифрування асиметричними алгоритмами;

- обчислення і перевірку електронного цифрового підпису;

- хэширування.

1. **Вивчити** функції, можливості та основні режими роботи з обраною бібліотекою. Оформити реферат на тему «Функціональні можливості пакету \_\_\_\_\_\_\_\_\_\_\_\_\_\_ і його застосування у криптографічному захисті».

**У рефераті розкрити:**

- загальне призначення бібліотеки, її склад, з якими технологіями і середовищами програмування працює;

- вказати відомості про розробника (компанія, сайт, рік створення, тип ліцензії, номер доступної версії, вартість (для комерційних продуктів);

- описати функціональні можливості;

- розглянути принципи роботи з бібліотекою

- привести скан-копії екранів, що підтверджують наявність режимів, що дозволяють виконувати криптографічні функції.

1. Підготуватися до використання бібліотеки у власних розробках, для чого написати програму виклику 1-2-х довільних функцій бібліотеки і продемонструвати роботу програмного додатка.

Результати роботи програми включити реферат.

## Приклади криптографічнихх бібліотек:

1. PyCripto ( Python)
2. Cripto API (С#)
3. MCrypt ( PHP)
4. OpenSSL, Cryptography ( C++)
5. Java Cryptography Extension JCE, Bouncy Castle ( Java)
6. Cripto C++ (C++)
7. [Cryptlib](http://www.cs.auckland.ac.nz/~pgut001/cryptlib/)
8. [libgcrypt](http://directory.fsf.org/security/libgcrypt.html)
9. botan.randombit.net

# Питання для самоконтролю

1. Розкрийте поняття «Криптологія»
2. Дайте визначення поняттю «Шифр» і вкажіть різницю між блоковимии і потоковими шифрами.
3. Як оцінити стійкість шифрів
4. Перелічіть механізми найпростіших шифрів
5. Охарактеризуйте механізм «Заміна»
6. Охарактеризуйте механізм «Перестановка»
7. Охарактеризуйте механізм «Дроблення»

# СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

|  |  |
| --- | --- |
|  |  |
| 1 | Белов Е.Б., Лось В.П., Мещеряков Р.В., Шелупанов А.А. Основы информационной безопасности: Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2006. – 544с  <http://www.alleng.ru/d/comp/comp51.htm> |
| 2 | Варфоломеев А.А. Основы информационной безопасности: Учеб. пособие. – М.: РУДН, 2008. – 412 с. |
| 3 | Галатенко В.А. Основы информационной безопасности: Курс лекций. – М.: ИНТУИТ. РУ, 2006. – 205 с. // эл.ресурс - <http://www.intuit.ru/studies/courses/10/10/info> |
| 4 | А.В.Домашев, М.М. Грунтович, В.О.Попов, Д.И.Правиков, А.Ю.Щербаков Программирование алгоритмов защиты информации. - - М.: Изд-во "Нолидж", 2001. – 552c. |
| 5 | С.Мазаник Безопасность компьютера: защита от сбоев, вирусов и неисправностей / С.Мазаник . – М.: Эксмо, 2014. – 256с.  tfile.me.700211.torrent |
| 6 | Проскурин В.Г. Защита программ и данных: учеб.пособие для студ.учреждений высш.проф.образования /В.Г.Проскурин. – М.: Издательский центр «Академия», 2012. – 208с.  [BiGTORRENT.ORG]v.g.-prskurin.-zashchita-programm-i-dannyih.torrent |
| 7 | Шаньгин В.Ф. Защита компьютерной информации. Эффективные методы и средства/ Шаньгин В.Ф. – М.: ДМК Пресс, 2008. – 544 с. |
| 8 | Гайкович В.Ю., Ершов Д.В. Основы безопасности информационных тех­нологий. - М.: МИФИ, 1995. - 96с. |
| 9 | Галатенко В.А. Информационная безопасность: практический подход.- М.: Наука, 1998. - 301с |
| 10 | Мельников В.В. Защита информации в компьютерных системах. – М., 1997. – 368с. |
| 11 | Введение в специальность «Организация и технология защиты информации». сост. Л.К.Бабенко. Таганрог: Изд-во ТРТУ, 1999.54с |
| 12 | Інформаційна безпека України // Электр.ресурс. <http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B0_%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D0%B8> |
| 13 | Список нормативних документів щодо інформаційної безпеки в Україні // Электр.ресурс. <http://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B8%D1%85_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%96%D0%B2_%D1%89%D0%BE%D0%B4%D0%BE_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%BE%D1%97_%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B8_%D0%B2_%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%96> |
| 14 | Про рішення Ради національної безпеки і оборони України від 28 квітня 2014 року «Про скасування деяких рішень Ради національної безпеки і оборони України» // Электр.ресурс. <http://zakon1.rada.gov.ua/laws/show/514/2009> |

НАВЧАЛЬНО – МЕТОДИЧНЕ ВИДАННЯ

**МЕТОДИЧНІ ВКАЗІВКИ**

до виконання лабораторних робіт

з дисципліни

**«Безпека програм та даних»**

для студентів напрямів підготовки

6.050101 «Комп’ютерні науки»

6.050102 «Комп’ютерна інженерія»

6.050103 «Програмна інженерія»

усіх форм навчання

Комп’ютерний набір і верстка:

Редактор:

Донецький національний технічний університет

м. Красноармійськ, пл. Шибанкова, 2.