МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсової роботи з дисципліни

"ПРОГРАМУВАННЯ"

для студентів напряму підготовки 6.170101
“Безпека інформаційних і комунікаційних систем ”

Міністерство освіти і науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсової роботи з дисципліни

"ПРОГРАМУВАННЯ"

для студентів напряму підготовки 6.170101
“Безпека інформаційних і комунікаційних систем ”

Вінниця
ВНТУ
2010

Рекомендовано до друку Методичною радою Вінницького національного технічного університету Міністерства освіти і науки України (Протокол № 10 від 18.06.2009 р.)

Рецензенти:

А. М. Пєтух, доктор технічних наук професор

Г. Б. Ракитянська, кандидат технічних наук доцент

Методичні вказівки до виконання курсової роботи з дисципліни "Програмування" для студентів напряму підготовки 6.170101 “Безпека інформаційних і комунікаційних систем” / Уклад. В. А. Каплун, О. П. Войтович. – Вінниця: ВНТУ, 2010. – 62 с.

Методичні вказівки призначені для надання допомоги при виконанні курсової роботи з дисципліни "Програмування". У роботі зроблено акцент на застосуванні об’єктно-орієнтованого програмування, як нової технології програмування, на необхідності використання API-функцій, знання яких має неабияке значення при захисті програмного забезпечення, та на тих базових конструкціях, які доцільно використовувати при реалізації завдання курсової роботи. Крім того, у методичних вказівках дано рекомендації з коректного оформлення пояснювальної записки до курсової роботи, наведено ряд прикладів подання текстової інформації та графічної частини.

Методичні вказівки призначені для студентів напряму підготовки 6.170101 “Безпека інформаційних і комунікаційних систем” усіх форм навчання.

ЗМІСТ

1 ТЕМАТИКА ТА ЗМІСТ КУРСОВОЇ РОБОТИ

Курсова робота (КР) з дисципліни «Програмування» – це самостійна робота, яка охоплює весь матеріал, викладений під час вивчення дисципліни «Програмування», і містить елементи (задачі) навчального, аналітично-розрахункового та науково-дослідницького характеру.

В курсовій роботі з дисципліни «Програмування» студент повинен показати знання мов програмування, розуміння основних концепцій нових технологій програмування, вміння самостійно розробити схему роботи програми в цілому та алгоритми складових поставленої задачі, підібрати засоби його програмної реалізації та подати розробку у вигляді, зручному для його використання сторонніми користувачами.

Тематика курсової роботи пов’язана з майбутньою спеціальністю студентів. Для програмної реалізації даної курсової роботи пропонуються найпростіші традиційні шифри для криптографічного захисту інформації: шифри перестановок, заміни, гамування тощо. Крім того, в якості об’єкту програмування можуть бути розробки ігрових програм, реалізація тестових програм, розробка лабораторних практикумів для інших дисциплін тощо.

Під час виконання курсової роботи студенти повинні використати всі знання, отримані ними під час вивчення дисципліни “Програмування”: робота з файлами, робота з масивами, різноманітні види операторів (умовні оператори, оператори вибору, оператори циклу і т.д.), застосування принципів об’єктно-орієнтованого програмування, робота у візуальних середовищах програмування.

Зміст курсової роботи визначається завданням, яке видається на консультації викладачем кожному студенту. Завдання видається не пізніше 6 днів з початку семестра. Курсове проектування включає декілька послідовних етапів, які, в загальному випадку, пов'язані зі змістовною постановкою задачі, розробкою індивідуального технічного завдання, вибором форми подання задачі, розробкою математичної моделі, вибором оптимального алгоритму реалізації задачі, проведенням досліджень режимів роботи програми та формулюванням обґрунтованих висновків щодо отриманих в роботі результатів. Кожен етап роботи обов’язково має знайти своє відображення в пояснювальній записці, що містить вхідні, вихідні та пояснювальні матеріали, які пов'язані з виконанням курсової роботи.

Завдання для курсових робіт визначаються викладачем із загального списку завдань на курсову роботу (Додаток А). Заохочуються пропозиції студентів щодо самостійного, за узгодженням з викладачем, вибору теми КР поза межами запропонованого в методичних вказівках переліку. Самостійний вибір предметної області, в якій доцільно використовувати сучасні методи програмування та оригінальні алгоритми, дозволяє зробити висновок щодо рівня творчої активності студента, його вміння самостійно здійснити попередній аналіз предметної області і розробити технічне завдання.

2 ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

Програма, яка є результатом виконання курсової роботи, повинна бути повноцінним додатком операційної системи Windows. Розроблена програма обов'язково повинна мати такі складові.

1. *Застосування основних принципів об'єктно-орієнтованого програмування*: абстрактні типи даних, інкапсуляція, успадкування класів, поліморфізм. Студент повинен добре розуміти, з яких причин він використав у програмі той чи інший принцип, довести доцільність його використання, вміти пояснити це під час захисту роботи.

   Дана вимога пояснюється тим, що саме технологія об'єктно-орієнтованого програмування є на даному етапі найсучаснішою і найуживанішою, і знання основних її концепцій обумовлює розуміння побудови інших технологій програмування.

2. *Дотримання технології модульного програмування.* Бажано фрагменти коду, що мають певне самостійне значення, оформляти у вигляді процедур та функцій, з яких формуються відповідні файли заголовків. Структура головної програми та додаткових програмних модулів повинні бути зрозумілими, змістовними. Програми повинні бути читабельними (розташування операторних дужок, структурність вкладених операторів тощо) і мати відповідні коментарі, що пояснюють певні фрагменти коду. Це полегшить розуміння програми, продемонструє вміння студентів коректно і грамотно користуватись основними прийомами програмування, допоможе при захисті курсової роботи.

3. *Реалізація дружнього інтерфейсу*: використання багаторівневого меню, діалогових вікон, різноманітних елементів керування роботою програми, можлива графічна інтерпретація результатів, попередження про можливі помилки при введенні інформації, підказки під час інтерактивного режиму роботи і т.д. Меню програми обов'язково повинно містити пункти з інформацією про розробника програми, короткі теоретичні відомості (наприклад, пояснення принципу шифрування, формалізований опис алгоритму тощо), перегляд вихідного тексту програми (можливо з розділенням на підпункти, які відповідають окремим підпрограмам). Реалізація інтерфейсу та використання елементів керування повинні базуватися на API-функціях, оскільки саме ці функції є складовою ядра операційної системи і основою для побудови програмних засобів багатьох мов програмування і використання більшістю сучасних програмних середовищ.

4. *Використання файлів для зберігання та зчитування інформації*. Це може бути або введення початкової інформацій з файлу (файлів) і виведення результуючої інформації у файл (файли), або зберігання

ключової інформації, або зберігання необхідних таблиць та алфавітів для шифрування. Причому, якщо задача передбачає різні варіанти вхідних даних, для кожного з випадків необхідно підготувати свій набір вхідних даних. Це повинно знайти своє відображення і при розробці математичної моделі, і при програмуванні, і при аналізі результатів.

Дана вимога при виконанні курсової роботи зумовлена тим, що робота з файлами є надзвичайно необхідною у будь-якій галузі програмування: при зберіганні і використанні інформації різноманітного характеру (числової, текстової, графічної тощо), при розробці і супроводженні баз даних, при передачі і отриманні повідомлень і т.д.

5. *Перевірка цілісності даних* на рівні перевірки правильності введеної інформації (числової, символьної, великі літери, малі літери, належність до алфавіту тощо), перевірка існування потрібних файлів і т.д. Ця вимога є необхідною, оскільки програми, в основному, створюються для пересічних користувачів, які не обізнані з тонкощами програмування і особливостями комп'ютерних систем. І тому при розробці програм необхідно враховувати можливість неправильного введення даних; допомогу при появі помилок та при відсутності необхідної для програми інформації.
6. *Подання інформації* (як вхідної, так і результуючої) повинно буди зрозумілим, мати необхідні пояснення. Всі результати вхідних, проміжних, результуючих дій повинні бути виведені на екран у вигляді, зручному для розуміння та аналізу стороннім користувачем, з поясненнями, допоміжними вікнами повідомлень.
7. *Розробка супроводжувальної документації* до програмної розробки. Необхідність виконання цієї вимоги пояснюється тим, що кожен товар (а програма також є товаром) повинен супроводжуватись детальним описом, що включає наведення його характеристик і параметрів, розробників, правила користування і т.д. Для курсової роботи супроводжувальна документація являє собою пояснювальну записку, яка має повністю описувати процес виконання задачі: побудову схем і алгоритмів, послідовність розробки процедур і функцій, їх опис та підключення до основної програми, перевірку правильності роботи, тестування на предмет ефективності роботи, інструкції для користування нею. Про правила оформлення пояснювальної записки йтиметься далі.

3 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Даний підрозділ методичних вказівок не дає повного обсягу теоретичних відомостей про всі можливі засоби програмування мовою С++. Його метою є – звернути увагу на деякі аспекти програмування при реалізації завдань курсової роботи: оптимальний підбір базових конструкцій мови програмування, особливості об'єктно-орієнтованого підходу до написання програм, доцільність і необхідність використання API-функцій тощо, обґрунтування використання тих або інших програмних засобів. Нижче наведені основні рекомендації, яких слід дотримуватись при виконанні курсової роботи.

3.1 Обґрунтування вибору мови програмування та програмного середовища

Основним завданням курсової роботи є розробка програми, що реалізує поставлену перед студентом задачу. Для реалізації поставленої задачі рекомендується використовувати мову програмування С++. Цю мову програмування справедливо називають продовженням мови С і будь-яка роботоздатна програма мовою С буде підтримуватись компілятором мови С++, але С++ дозволяє перейти до нового способу концептуалізації і розв'язання багатьох проблем програмування, які не можна було реалізувати за допомогою мови С.

Програми, написані різними мовами програмування, перекладаються на машинні мови інтерпретаторами або компіляторами [1, 2]. Різниця між ними в тому, що інтерпретатор в міру зчитування програми послідовно перетворює її команди (або код) у команди машинної мови, а компілятор повністю переводить програмний код (лістинг програми) в деяку проміжну форму – об'єктний файл (це і є етап компіляції), а лише після цього компілятор викликає програму компонування, яка перетворює об'єктний файл у виконуваний файл програми.

З інтерпретаторами працювати простіше, оскільки команди програми виконуються у тій послідовності, в якій вони записані, що полегшує контроль за виконанням програми. Компілятор вносить додаткові етапи компіляції і компонування програм, в результаті чого утворюється виконуваний файл, недоступний для аналізу і редагування. Але скомпільовані програми виконуються швидше, оскільки переведення команд програми на машинну мову вже здійснився на етапі компіляції, а отже, отримана програма може виконуватись на комп'ютерах без компілятора. Мова С++ якраз і відноситься до компільованих мов. При роботі ж з інтерпретованими мовами для виконання готової програми треба обов'язково мати відповідну програму-інтерпретатор. В деяких мовах (наприклад, Visual Basic), роль інтерпретатора виконує динамічна бібліотека, а

інтерпретатором мови Java є віртуальна машина (Virtual Machine).

Протягом багатьох років основними характеристиками програм вважались їх розмір і швидкість виконання. Програму намагались зробити якомога меншою, оскільки пам'ять була досить дорогою, а зацікавленість високою швидкістю виконання пояснювалась високою вартістю процесорного часу. Але внаслідок здешевлення комп'ютерів і його складових пріоритети змінювались. Сьогодні вартість робочого часу програміста набагато перевищує вартість більшості комп'ютерів. Великим попитом користуються професійно написані і легкі в експлуатації програми. А простота експлуатації означає, що при змінах вимог, пов'язаних з розв'язанням конкретних задач, програма легко переналаштовується без великих додаткових витрат.

Саме з цих міркувань для програмної реалізації завдань курсової роботи вибрано мову програмування C++ [3, 4]. А для розробки програми пропонується використовувати програмне середовище Visual C++. Саме це програмне середовище є тим зручним інструментом, який оснащений різноманітними допоміжними засобами для створення зручних і зрозумілих програмних додатків для операційної системи сімейства Windows і для побудови зрозумілого графічного інтерфейсу програм [5, 6].

3.2 Використання основних базових конструкцій мови C++

Мовою програмування для реалізації завдань курсової роботи вибрано мову C++. Отже, при розробці програми студенти повинні дотримуватись усіх правил і погоджень, які прийняті при програмуванні цією мовою. Наведемо лише основні рекомендації.

Змінні і константи [1, 2, 7]**.** У мові C++ змінні використовуються для зберігання інформації у комірках (в одній або декількох) пам'яті комп'ютера, а назву змінної можна подати як напис на комірці пам'яті, за яким її можна знайти. Тому назви змінних повинні бути такими, щоб вони свідчили про сутність і призначення цих змінних. При цьому слід пам'ятати, що мова C++ чуттєва до регістру літер, тобто великі і малі літери вважаються різними. При програмуванні для Windows прийнято ідентифікувати змінні за певними правилами. Ввів ці правила угорський програміст Чарльз Симонаі, і надалі ця система іменування змінних стала називатися "угорською нотацією". Ось основні правила:

- кожне слово в імені змінної пишеться з великої літери, наприклад, MyVariable, BigWind;
- кожному ідентифіктору передує декілька малих букв, що визначають його тип. Наприклад, nMyVar – змінна цілого типу; cYourVar – змінна символьного типу.

Ці правила дозволяють спростити процес читання і розуміння програм – ім'я змінної визначається її типом. Саме ці правила

використовуються в іменах змінних, констант та у назвах API-функцій.

Константи також являють собою комірки пам'яті для зберігання даних, але, на відміну від змінних, їх не можна змінювати. Створювану константу необхідно ініціалізувати, оскільки пізніше їм не можна буде присвоїти нове значення. Зробити це можна за допомогою директиви #define або за допомогою ключового слова const.

При програмній реалізації задачі студент повинен дотримуватись усіх цих правил.

Типи даних. При визначенні змінної у мові С++ необхідно надати компілятору інформацію про її тип [3, 4, 7]. Звичайно, кожен з базових типів доцільно використовувати для зберігання певного виду інформації:

int, short, long, unsigned int, unsigned short, unsigned long – для визначення змінних для лічильників, підрахунку кількості, змінних циклу;

bool – для визначення логічних змінних, які можуть приймати одне з двох значень: true (1) або false (0);

float, double – для визначення змінних, які можуть приймати дійсні значення;

char – для визначення символьних змінних (символи, рядки символів).

Окрім базових типів даних, при програмуванні під Windows введені нові типи даних. Всі вони визначені в заголовочних файлах Win32, і базовим для них є <windows.h>. Це, наприклад, такі типи даних:

HANDLE – ціле число, яке використовується як дескриптор, тобто для позначення якогось ресурсу (вікно, меню, ікона, курсор, ...);

HWND – дескриптор вікна;

HINSTANCE– дескриптор додатка;

HBRUSH, HPEN– дескриптор пензля і олівця;

HFONT – дескриптор шрифта;

HBITMAP – дескриптор растрового зображення;

HDC – дескриптор контексту пристрою;

BOOL – бульовий тип;

UINT – ціле 32-розрядне беззнакове число;

BYTE – 8-розрядне ціле число;

WORD – слово;

DWORD – подвійне слово;

LONG – ціле число;

LPSTR – покажчик на рядок;

LPCSTR – константний покажчик на рядок.

При введенні і використанні тієї або іншої змінної і константи студент повинен обґрунтувати тип даних, який він обрав для неї.

Вирази і оператори [1, 2, 4, 7]. Програма являє собою набір команд, що виконуються у певній послідовності. Кожна команда являє собою сукупність певних операторів:

– *оператори присвоювання* = (коли значення операнда зліва заміняється значенням виразу справа);
– *математичні оператори* (п’ять основних математичних операцій: +, –, *, / , %);
– *оператори інкременту і декременту* (постфіксні і префіксні): ++, --, які збільшують або зменшують значення змінної на одиницю після або до операції;
– *оператори відношень* (==, >, >=, <, <=, !=), які використовуються для порівняння двох значень і завжди повертають значення true або false;
– *логічні оператори* (&&, ||, !), які використовуються, коли виникає необхідність перевірити не одну умову, а відразу декілька.

Використовуючи ці основні оператори, студенти повинні бути уважними: дотримуватись пріоритетів операторів, правильно розставляти операторні дужки, що суттєво впливає на хід виконання програми і часто є джерелом незрозумілих помилок.

Крім основних операторів, існує ще ряд операторів, які дозволяють змінювати послідовний хід виконання програми, спрощувати її написання.

Умовний оператор if…else, який дозволяє змінювати процес виконання програми, направляючи його по шляху, який буде залежати від певної умови. Причому допускається використання вкладених блоків умовних операторів, що дозволяє реалізовувати більш складний механізм керування ходом програми.

Оператор вибору switch використовується тоді, коли треба перевірити відразу декілька умов і коли використання умовного оператора може привести до виникнення конструкцій з великою кількістю вкладень, що утруднює як написання, так і сприйняття програми. Наприклад, при обробці меню, яке складається з 25 пунктів, доцільніше використовувати саме оператор вибору switch, а не перевіряти 25 разів умову оператором if.

Оператори циклів використовуються тоді, коли необхідно виконати одні й ті самі дії багаторазово. Студенти повинні правильно підбирати той або інший оператор циклу. Це свідчитиме про те, що студент глибоко розуміє призначення кожного з них, оскільки кожний з них є оптимальним лише при виконання певних правил:
– оператор циклу while (<умова>) організовує цикл, в якому виконання послідовності операторів продовжується, поки умова виконується, тобто умова перевіряється перед виконанням тіла циклу;
– оператор циклу do…while (<умова>) використовується тоді, коли умова перевіряється лише після виконання тіла циклу;
– оператор циклу for(…) виконує декілька дій: встановлює початкове значення змінних циклу, а потім на кожній ітерації контролює виконання умов продовження циклу і змінює значення змінних циклу.

Операторні дужки, коментарі. Для покращення читабельності програми настійливо рекомендується використовувати операторні дужки та

коментарі. Операторні дужки застосовуються для того, щоб об'єднати в один блок групу операторів, які виконують якусь певну функцію, або послідовність операторів, які виконують певний крок в алгоритмі. Це зручно, це доцільно і це зрозуміло. Крім того, операторні дужки ({ і }) бажано розташовувати так, щоб їх було видно, а тіло блока, розташованого в межах цих дужок, бажано дещо зміщувати вправо. Якщо виконувати ці прості правила, програма буде набагато зрозумілішою, прозорою, легше піддаватись корегуванню і пошуку помилок.

Коментарі також спрощують сприйняття програми, дають потрібні пояснення, оскільки і сам розробник програми з часом забуває, що, як і з якою метою він реалізував у певному фрагменті програми. Використання операторних дужок і коментарів вважається ознакою хорошого тону при програмуванні.

Функції [1, 2, 3]. Складні задачі слід розбивати на декілька більш простих, кожна з яких виконує конкретну і цілком зрозумілу задачу. Це і є функції програми. Кожна функція за своєю суттю – це підпрограма, яка може маніпулювати власними даними і повертати деяке значення. Кожна функція має власне ім'я, і, коли воно зустрічається у програмі, керування переходить до тіла даної функції. Цей процес називається викликом функції (або зверненням до функції). Після виконання функції виконання програми відновлюється з оператора, що слідує за викликом функції.

Розрізняють два види функцій: *нестандартні* – ті, що визначені і розроблені користувачем, і *вбудовані* – ті, які є складовою частиною пакета компілятора і надаються фірмою-розробником [4]. Вбудовані функції несуть велике навантаження у програмах і виконують величезну кількість стандартних дій (наприклад, обчислюють значення тригонометричних функцій, виводять на екран текст, примітивні геометричні фігури, встановлюють кольори, параметри, певні характеристики тощо).

Нестандартні функції створюються самим програмістом. Їх використання у курсовій роботі є бажаним і є показником глибоких знань у галузі програмування. Наприклад, студенти можуть розробляти власні функції для здійснення процесу шифрування або розшифрування, для знаходження певного файлу на диску, для виведення певного зображення, для виведення певного тексту певним шрифтом, для обробки певних пунктів меню або інших елементів керування, для визначення опцій і режимів роботи програми і т.д. Як правило, головна функція програми повинна, в основному, викликати в певній послідовності певні функції. У такому випадку вона буде зрозумілою, простою. Крім того, кожну функцію окремо відлагоджувати набагато простіше, а шукати помилки у невеличкому фрагменті коду набагато легше, ніж копирсатись у довгих роздруківках або перегортати одну за одною сторінки екрана.

Глобальні і локальні змінні [1, 2, 3, 4]. Кожна змінна характеризується своєю областю дії, що визначає термін життя і доступність змінної у програмі. *Локальні змінні* – це змінні, які об'явлені в середині деякого блоку програми і мають область дії, обмежену лише цим блоком. До них можна отримати доступ лише в межах цього блоку, і після того, як виконання програми вийде за межі, всі його локальні змінні автоматично вилучаються з пам'яті. Наприклад, змінну циклу у функції не варто об'являти глобально, вона повинна працювати лише в цій функції, або навіть у якомусь маленькому блоці.

Глобальні змінні мають глобальну область дії і доступні з будь-якої точки програми. Вони необхідні у тих випадках, коли програмісту слід зробити дані доступними для багатьох функцій, а передавати дані з функції у функцію як параметри досить проблематично.

Так, наприклад, у додатках під Windows більшість API-функцій мають використовують дескриптор додатка, але не передають його в якості параметра. Отже змінну для дескриптора додатка слід об'явити глобально. А от дескриптор вікна більшість API-функцій мають в якості параметра, тому змінну для дескриптора вікна можна оголошувати локально.

Отож, при розробці програми студентам треба ретельно аналізувати область дії оголошених змінних і правильно використовувати їх, враховуючи всі особливості, виконуючи всі правила застосування локальних і глобальних змінних. Досить часто неправильне розуміння цих особливостей призводить до помилок, при яких програма не видає помилок, але працює не так, як треба, або видає несподіваний результат.

Таким чином, обираючи той або інший програмний засіб або програмну конструкцію, студент повинен вміти обґрунтувати і пояснити свій вибір під час проведення захисту курсової роботи.

3.3 Основні концепції об'єктно-орієнтованого програмування

Об'єктно-орієнтоване програмування ("object-oriented programming") – новий підхід до програмування. В міру розвитку обчислювальної техніки і ускладнення задач виникали різні моделі програмування.

Перші компілятори (наприклад, FORTRAN) підтримували *процедурну модель програмування*, в основі якої лежить використання функцій [7]. Наступний етап розвитку пов'язаний з переходом до *структурної моделі програмування* (до неї відносяться компілятори ALGOL, Pascal, C), в основі якого лежать такі положенні: програми подаються у вигляді сукупності взаємопов'язаних процедур і даних, якими ці процедури (або блоки) оперують.

При першому і другому підходах дані і процедури їх оголошуються, обробляються та готуються окремо. При використанні об'єктно-орієнтованого програмування (ООП) дані і підпрограми обробки цих даних

об’єднані в єдиній сутності, яка називається класом.

Модель ООП основана на декількох концепціях [8].

Абстракція даних (або типи даних, визначені користувачем). Ця концепція полягає у можливості визначати нові типи даних, з якими можна працювати так само, як і з основними типами даних. Абстракція має місце і при застосуванні шаблонів, тобто введення абстрактних типів даних, які в залежності від умов їх застосування приймають той або інший тип.

Інкапсуляція – об’єднання даних і функцій роботи з цими даними для обробки об’єктів певного типу. Для реалізації абстракцій і інкапсуляцій використовуються класи. Тобто, клас визначає тип даних. Кожний представник класу називається об’єктом. Призначення класу визначається можливими діями над об’єктами класу, які задаються функціями-членами. Створення об’єктів даного класу виконується конструкторами, а знищення – деструкторами.

Успадкування (наслідування) – процес, за допомогою якого об’єкт може набувати властивостей іншого об’єкта. Це означає, що в ООП на основі вже існуючих класів можна будувати похідні (роС. производные) класи. При цьому похідний клас, або клас-нащадок успадковує елементи (дані і функції-члени) свого батьківського класу (класу-пращура, базового класу), додаючи до нього свої, які дозволяють йому реалізовувати тільки йому характерні функції.

Крім того, у похідному класі успадковані функції можуть бути перевизначені. Таким чином можна побудувати ієрархії класів, пов’язаних між собою. Якщо об’єкт наслідує свої атрибути (дані-члени і функції-члени) від одного базового класу, це буде просте успадкування. Якщо об’єкт успадковує атрибути від кількох батьків, це буде множинне успадкування.

Поліморфізм (грецькою ”polymorphos”– множинність форм) – це властивість програмного коду поводитись по-різному в залежності від ситуації, що виникає в момент виконання. Проявляється поліморфізм у можливості використання одного імені функції-члена для функцій, що мають різні типи аргументів. Це називається перевантаженням функцій. Поліморфізм застосовується як до функцій, так і до окремих операцій, тобто виконувані операцією дії також можуть залежати від типу операторів – це перевантаження операторів.

Як зауважено вище, саме створення і використання класів і складає основу усіх концепцій ООП.

Клас – це абстрактний тип даних, що визначається користувачем.

Змінні, які об’являються в середині класу, називаються *данимичленами*_класу, а функції називаються *функціями-членами* або методами даного класу.

За замовчуванням всі члени класу є закритими, тобто вони доступні тільки для членів цього класу. Але можна задати різні режими доступу, що визначається такими специфікаторами:

– private – дані-члени і функції-члени доступні тільки для функцій-членів цього класу;
– protected – дані-члени і функції-члени доступні для функцій даного класу і похідних від нього;
– public – дані-члени і функції-члени класу доступні для функцій цього класу і інших функцій програми, в якій є представник даного класу.

Структури та об'єднання є частинними випадками класів.

Структура (struct) – це тип класу, в якому всі члени за замовчуванням "public," хоча при необхідності можна використовувати private.

Об'єднання (union) – це також тип класу, в якому всі члени за замовчуванням мають специфікатор "public" і цей специфікатор не може бути змінений. Крім того, об'єднання не підтримують ієрархію класів, тобто вони не можуть успадковувати і самі не можуть бути базовими класами.

Серед функцій-членів класу є такі, які визначають особливості створення об'єктів, ініціалізацію змінних, копіювання та знищення об'єктів даного класу. Це методи класу, які мають назви конструктор та деструктор.

Конструктор – функція-член класу, основна мета якої - ініціалізація об'єктів даного типу, виділення пам'яті для їх зберігання. Особливості цих методів полягають в тому, що:

– конструктор обов'язково має те саме ім'я, що і клас;
– конструктор не має повертати ніякого значення, а отже немає типу;
– конструктор не успадковується, але може перевантажуватись.

Якщо в класі не визначено конструктор, тоді компілятор сам генерує конструктор за замовчуванням, без параметрів. Конструктор викликається завжди, коли створюється об'єкт. Таким чином, конструктор викликається явно при створенні об'єкта і неявно при застосуванні оператора new. Отже, якщо не треба визначити якісь певні дії при створенні об'єктів даного класу, програміст не повинен писати код, що визначає конструктор, компілятор робить це сам (створює конструктор без параметрів і без тіла).

Деструктор – це особливий метод класу, що знищує об'єкт явно (за допомогою оператора delete, або неявно) при завершенні програми. Клас може мати тільки один деструктор. Як правило, у тілі деструктора описуються ті дії, які слід виконати при знищенні об'єкта даного класу. Деструктор повинен задовольняти такі умови:

– деструктор не може мати параметрів;
– деструктор не може повертати значення;
– деструктор не успадковується.

Таким чином, основним поняттям ООП є класи, які становлять основу концепцій абстракції, інкапсуляції, успадкування і поліморфізму у програмуванні.

а) До переваг ООП відносять такі:
б) при використанні технології ООП можна повторно використовувати код програми і таким чином економити час на розробку;

в) програми з використанням ООП добре структуровані, що дозволяє добре розуміти, які функції виконують окремі підпрограми;
г) програми з використанням ООП легко тестувати. Можна розбити програму на компоненти і тестувати роботу кожної з них;
д) програми легко модифікувати.

3.4 Використання API-функцій при написанні програми

Метою програми, яка повинна бути створена в результаті виконання курсової роботи, є створення повноцінного GUI-додатка для Windows. Однією з вимог при розробці курсової роботи є використання API-функцій. API-функції – це функції прикладного програмного інтерфейсу Windows (Application Programming Interface – API) [9]. Основною рисою Windows-додатків є те, що вони підтримують віконний інтерфейс, використовуючи при цьому множину стандартних елементів керування (кнопки, лінійки, шкали, списки і т.п.). Ці елементи підтримуються за допомогою динамічних бібліотек (DLL), які є частиною операційної системи і базуються на API-функціях ядра операційної системи. Знання API-функцій і вміння їх використовувати є суттєвим при розробці систем захисту програмного забезпечення, при побудові захисту ресурсів операційної системи, для забезпечення протидії засобам статичного дослідження (дизасемблерам, декомпіляторам) та засобам динамічного дослідження і зламу (налагоджувачам реального та захищеного режимів) захищених програмних засобів.

Модель програмування Windows в повній мірі реалізує основну ідею об'єктно-орієнтованого програмування: кожне вікно являє собою об’єкт (кнопка, поле вибору, меню, смуги прокрутки тощо), а керування об'єктами здійснюється за допомогою повідомлень [10].

Програмування на рівні API-функцій – найнижчий рівень програмування на Сі в середовищі Windows, на відміну від використання бібліотеки MFC. Не зважаючи на збільшення об’єму програм при написанні прграм з API-функціями, все ж програмування на API-функціях має ряд переваг:

- в бібліотеці MFC не реалізовані всі можливості API;
- більшість методів класів бібліотеки MFC побудовані на використанні API-функцій;
- до API-функцій можна звернутися із підпрограм, бібліотек, написаних в різних середовищах: Visual C++, Borland C++, C Bilder тощо.

3.5 Обробка повідомлень Windows

Всі Windows-програми є програмами, що керуються подіями – це відрізняє їх від традиційних програм. Більшу частину часу додаток, що керується подіями, очікує на подію, точніше на повідомлення про

подію (програми ДОС є активними, тобто весь час тримають необхідні їм ресурси) [9, 10, 11].

В Windows прикладна програма посилає всі свої повідомлення системі, а система, визначаючи з дескриптора додатка, до якої програми належить це повідомлення, розташовує це повідомлення в черзі повідомлень даного додатка, далі повідомлення надсилається на обробку віконній функції, яка його і обробляє. Тобто, існують дві черги: системна черга повідомлень та черга повідомлень кожного додатка (головного вікна програми). І програми, і система з деякою періодичністю звертаються до черги і перевіряють, чи немає в черзі повідомлень. Таким чином, у кожної програми, а також у кожної системи повинні існувати цикли, під час яких обпитується черга і вибирається інформація про повідомлення, тобто цикл опитування черги – обов'язкова частина віконної функції.

Вся програма для Windows зводиться до обробки повідомлень, які надходять, тобто Windows-програми є пасивними (весь час очікують повідомлень).

Повідомлення Windows мають стандартні імена, більшість з яких починається з префікса WM_ (скорочено від слів Windows Message).

Всі повідомлення Windows (їх близько 1000) можна розбити на декілька класів (близько 15), основні з яких такі.

1. Системні повідомлення – WM_SYSCOMMAND.
2. Повідомлення керування вікнами: WM_CREATE – надходить при створенні вікна; WM_DESTROY – при знищенні вікна; WM_SIZE – при зміні розмірів вікна; WM_MOVE – при переміщенні вікна.
3. Повідомлення від клавіатури: WM_KEYDOWN – при натисканні на клавішу клавіатури; WM_KEYUP – при відпусканні клавіші.
4. Повідомлення від миші: WM_LBUTTONDOWN – при натисканні лівої кнопки миші; WM_RBUTTOMDOWN – при натисканні правої кнопки миші; WM_LBUTTONUP – при відпусканні лівої кнопки миші; WM_RBUTTONUP – при відпусканні правої кнопки миші; WM_MOUSEMOVE – при переміщенні курсора миші.
5. Повідомлення для реакції на вибір пунктів і підпунктів меню або від елементів керування у діалогових вікнах WM_COMMAND.
6. Повідомлення від таймера WM_TIMER.
7. Повідомлення для перерисовування вікна WM_PAINT.

Це основні групи повідомлень, але існує ще велика кількість інших повідомлень. Кожне повідомлення, яке передбачається використати у програмному додатка, повинно знайти своє відображення у віконній функції того вікна додатка, яке генерує це повідомлення [9-11]. Тобто, якщо у якомусь певному вікні додатка передбачається здійснювати якість дії при натисканні на кнопки миші, на клавіші клавіатури, на пункти меню, то при написанні віконної функції для цього вікна обов'язково треба передбачити обробку відповідних повідомлень.

3.6 Використання вікон у програмах для Windows

Найпростіший додаток Windows містить два модулі:

- функція *WinMain ()*;
- функція *WndProc ()*.

WinMain() (ім'я зарезервовано) – головна функція додатка аналогічна функції *main()* в звичайній програмі на С. Вона виконує такі дії:

а) визначає (описує) клас вікна. Для цього необхідно заповнити структуру атрибутів класу вікна типу *WNDCLASS* (опис цієї структури знаходиться у файлі *<windows.h>)*. Це робиться для того, щоб на основі стандартного класу для опису параметрів вікна W*NDCLASS* створити (успадкувати) свій клас, який надає вікну деяких особливостей (власна іконка, власний курсор, колір, фон, меню і т.д.);

б) реєструє клас вікна за допомогою функції *RegisterClass( )*. Це робиться для того, щоб проінформувати систему, що такий клас існує;

в) сворює вікно даного класу (тобто створює об'єкт типу клас вікна) за допомогою функції *CreateWindow( )*. На цьому етапі вказуються конкретні параметри створюваного вікна (розташування, розміри, стиль відображення тощо);

г) відображає вікно і посилає йому команду перерисуватися. Для цього необхідно викликати функції *ShowWindow(hWnd,nCmdShow)* та *UpdateWindow(hWnd)*;

д) запускає цикл обробки повідомлень. Для обробки черги додатка в функції *WinMain()* організовується нескінчений цикл обробки черги повідомлень, при виконанні якого виконуються такі дії: повідомлення витягуються з черги повідомлень і розміщуються в структурі типу MSG; віртуальні коди повідомлень перетворюються у символи ASCII; отримані повідомлення передаються відповідній віконній процедурі;

е) завершує роботу додатка при надходженні з черги повідомлення WM_QUIT.

WndProc() – функція обробки повідомлень вікна або віконна функція (ім'я функції може бути довільним). Виконання Windows програми починається з функції WinMain(), яка створює головне вікно додатка і відображає його на екрані, а потім запускає цикл очікування і обробки повідомлень. Функція вікна "просіює" всі повідомлення і обробляє ті з них, які вибрав розробник програми. Типова структура віконної процедури – це swith-блок, кожна гілка якого містить обробку якогось одного певного повідомлення. Деякі повідомлення Windows можуть попадати в функцію вікна безпосередньо із системної черги. Наприклад, WM_DESTROY – повідомлення про закриття вікна. Якщо у віконній процедурі відсутня гілка для обробки якогось повідомлення, то вікно доручає його обробку системі шляхом виклику функції *DefWindowProc()*. Таким чином,

фактично вся віконна процедура складається з одного великого оператора switch(), в якому розташовані багаточисельні case, всередині яких можуть бути знову розташовані свої switch(). Тобто, віконна процедура складається з блоків, кожний з яких обробляє певне повідомлення.

Для роботи з вікнами використовується ряд додаткових функцій.

Для виведення текстових повідомлень і отримання відповіді користувача використовується вікно повідомлень *MessageBox()*.

IsWindow() – функція повертає false (0), якщо вікно вже створене і існує, і true (1) – у протилежному випадку.

MoveWindow() – пересуває вікно у вказану позицію і змінює його параметри (розміри, прапорець перерисовування).

ShowWindow () – змінює вигляд вікна (приховане, мінімізоване і т.п.)

GetWindowRect() – задає координати прямокутника, що обрамляє вікно.

GetClientRect() – задає координати прямокутника, що обрамляє робочу область вікна (без заголовку).

InvalidateRect() – перерисовує вікно або вказану частину вікна.

ValidateRect() – не перерисовує вказану частину вікна.

Детальний опис функцій і їх використання знаходиться у джерелах, наведених у переліку рекомендованої літератури.

3.7 Виведення інформації у вікно

Особливості виведення у вікна додатка. При розробці додатків для операційної системи Windows виведення інформації на екран має деякі особливості.

1. Не можна користуватись функціями бібліотеки компілятора, оскільки вони виводять в одне єдине вікно, а у Windows додатки виводять інформацію одразу у декілька вікон.
2. Інтерфейс графічних пристроїв надає багато функцій виведення. І всі ці функції працюють не з фізичними пристроями, а з логічними, тобто опис виклику функцій не залежить від фізичного способу відображення, тобто додатки працюють з будь яким пристроєм виведення. Параметри виведення встановлюються в контексті відображення, тобто в структурі даних, яка містить характеристики пристроїв та покажчики на вибрані інструменти пристрою або на так звані логічні об’єкти графіки:
   - систему координат;
   - колір фону та виведення тексту (змінні типу COLORREF);
   - шрифти для виведення тексту (змінні типу HFONT);
   - пера для виведення графічних зображень (змінні типу HPEN);
   - пензлі для зарисовування графічних зображень (змінні типу HBRUSH).
3. Дескриптор контексту (змінна типу HDC) є першим аргументом виклику всіх функцій, пов’язаних з виведенням у вікно.

4. Прикладна програма не знає заздалегідь розміри вікна, куди буде виводитись інформація (для цього і встановлюється контекст відображення).
5. Прикладна програма не знає заздалегідь часу, коли відбуватиметься виведення інформації.
6. Система Windows не запам'ятовує вміст вікна прикладної програми. Збереження вмісту покладено на програму. Тому, коли виникає необхідність виведення (відновлення, перерисовування, ...), система посилає програмі повідомлення WM_PAINT, а ми повинні його обробити.

Виведення будь-якої інформації у вікно (тексту, графічних зображень) здійснюють за допомогою повідомлення WM_PAINT. Для цього необхідно заповнити структуру типу PAINTSTRUCT, створити контекст пристрою за допомогою функції *BeginPaint()*, а далі вже здійснювати будь-яке виведення інформації. Не слід забувати після завершення виведення звільнити зайняті ресурси, викликавши для цього функцію *EndPaint()*.

Слід зауважити, що виведення інформації може здійснюватись при обробці не лише повідомлення WM_PAINT, але й інших. Але тоді контекст пристрою повинен створюватись і знищуватись іншими методами:

GetDC() – створення тимчасового контексту пристрою;
ReleaseDC() – знищення тимчасового контексту пристою.

Крім того, використовуються такі функції:

SelectObject() – для вибору будь-якого логічного об'єкта в контекст пристрою;
DeleteObject() – знищення непотрібного логічного об'єкта графіки.

Виведення тексту. Для виведення тексту у вікно додатка необхідно підготувати параметри цього виведення (за замовчуванням вирівнювання по лівому краю, шрифт – прийнятий за замовчуванням у системі, колір тексту – чорний і т.д.). Якщо ж не задовольняють параметри за замовчуванням, можна їх змінити, використавши такі функції.

SetBkColor() – встановити колір фону вікна;
SetTextColor() – встановити колір тексту;
RGB() – формування кольору;
CreateFontIndirect() – створити шрифт із заданими параметрами;
SetTextAlign() – вибрати режим вирівнювання;
SetBkMode() – задати режим фону;
TextOut() – безпосереднє виведення тексту у зазначені координати;
GetTextMetrics() – отримати метрики шрифту.

Виведення графічних примітивів. Для рисування ліній використовуються пера. За замовчуванням перо має товщину рисування 1 і чорний колір. Для рисування замкнених фігур використовуються пензлі. За замовчуванням зарисовування замкнутих фігур відбувається білим кольором. Всі пера повинні мати тип HPEN, а пензлі повинні мати тип HBRUSH.

CreatePen() – створення пера;

CreateSolidBRUSH() – створення пензля суцільного кольору;
CreateHatchBrush() – створення штрихованого пензля із вказаним стилем та кольором;
MoveToEx() – установлює будь-яку позицію пера;
GetCurrentPositionEx() – знаходить поточну позицію пера;
LineTo() – рисує лінію поточним пером, починаючи з поточної позиції точки до вказаної точки;
SetPixel() – виводить точку заданим кольором;
GetPixel() –повертає колір точки заданої координати.
Rectangle() – рисує прямокутник заданим пером і заповнює пензлем;
RoundRect() – рисує прямокутник з закругленими краями;
FillRect() – зарисовує прямокутник з координатами, заданими в структурі типу RECT;
FrameRect() – рисує прямокутну рамку заданим пензлем (товщина – одна стандартна одиниця);
Ellipse() – рисує еліпс, ніби вписаний у прямокутник;
Arc() –рисує дугу еліпса або кола вибраним пером;
Chord() – виводить сегмент еліпса вибраним пером;
Pie() – виводить сектор еліпса, обводячи по краях заданим пером;
Poliline() – рисує ламану лінію заданим пером – з'єднує лініями точки, координати яких вказані у масиві структур типу POINT ;
Polygon() – рисує замкнуту фігуру – многокутник.

Іноді буває зручно працювати одночасно із сукупністю фігур. У цьому випадку у разі необхідності змінити положення або виконати інші дії над цією сукупністю прийшлось би перерисовувати всі фігури окремо. Існує можливість об'єднати всі ці фігури в єдиний об'єкт, і потім працювати з ним як з однією одиницею. Для цього можна об'явити змінні типу HRGN і здійснити над ними такі дії:

CreateRectRgn() – створити прямокутний регіон;
CreateEllipticRgn() – створити еліптичний регіон;
CombineRgn() – об'єднати ці регіони в єдиний об'єкт;
CreatePolygonRect() – створити багатокутний регіон;
PaintRgn() – відобразити регіон на екрані і заповнити його поточним пензлем;
FillRgn() – нарисувати регіон, заповнюючи його вказаним пензлем;
OffsetRgn() – змінити координати регіону на певну величину.

Виведення растрових зображень. Звичайні, різноманітні графічні зображення прикрашають інтерфейс будь-якої програми. Для виведення растрових зображень слід виконати такі дії.

1. Отримати хендл (створити змінну типу HDC) контексту пристрою (за допомогою методів *BeginPaint()* або *GetDC()*).

2. Отримати хендл графічного зображення (змінна типу HBITMAP), який буде відображуватись у вікні, за допомогою функції

   LoadImage().

   Наведена функція дозволяє завантажувати графічні образи як з ресурсів, записаних у виконуваному файлі, так і з файлів, що містять зображення. Графічним образом може бути bitmap, курсор, іконка. Крім того, наведена функція дозволяє керувати параметрами зображення і завантаженням образів.
3. Отримати сумісний (з фізичним) контекст у пам'яті для зберігання зображення, яке потім буде виводитись на пристрій виведення (перехідник між програмою і драйвером пристрою виведення):

   CreateCompatibleDC().
4. Завантажити хендл нашого bitmap'а у той сумісний контекст, який ми створили у пам'яті.

   SelectObject().
5. Скопіювати наш bitmap з контексту в пам'яті у контекст пристрою. Це робиться однією з функцій:

   StretchBlt() – масштабує зображення, приводячи ("натягуючи") його розмір до розміру робочої області;

   BitBlt() – копіює зображення, але не масштабує;

   PatBlt() – зафарбовує вказаний прямокутник вибраним пензлем.
6. Наприкінці слід знищити сумісний контекст, сам об'єкт і контекст пристрою за допомогою функції *Delete()*.

Обробка повідомлень таймера. Для того, щоб зображення у вікні рухалось, можна використати системний таймер, тобто, звертаючись через певні проміжки часу до системи, повідомляти її, що ми хочемо відтворити або змінити при виведенні зображення. Обробка таймера відбувається при посиланні повідомлення WM_TIMER.

SetTimer() – встановити величину інтервалу, через який будуть відбуватися події, описані в повідомленні WM_TIMER;

KillTimer() – знищити таймер.

3.8 Діалогові вікна

Основним засобом спілкування користувача з програмою є діалогові вікна. Діалогові вікна можуть бути:

а) системні діалогові вікна (при застосуванні яких користувач не зможе працювати з жодним іншим додатком і з жодним іншим вікном до тих пір, поки не дасть відповіді на поставлене питання);

б) модальні діалогові вікна. Вони використовуються частіше. Ці вікна не дають користувачу можливості працювати з іншими вікнами тільки

цього додатка (з іншими додатками працювати можна) до тих пір, поки він не дасть відповіді на поставлене питання, але дозволяють переключатись на інші додатки;

в) немодальні діалогові вікна – можна паралельно працювати з декількома вікнами цього додатка (саме для цього ці вікна і використовуються).

Діалогові вікна дозволяють вводити і отримувати інформацію, яку складно ввести за допомогою меню. Діалогове вікно має в своєму складі елементи (об'єкти) більш низького рівня. Ці елементи також являють собою вікна специфічного вигляду і наперед обумовленого призначення.

У програмному середовищі Visual C++ повинен бути витриманий порядок створення діалогового вікна:

а) створення шаблону діалогового вікна за допомогою засобів програмного середовища. В результаті на екрані з'являється шаблон діалогового вікна, в яке потім будуть додаватися елементи керування;

б) Написання програмного коду, що (викликає) зв'язує якусь певну дію з відкриттям цього діалогового вікна, за допомогою функції DialogBox();

в) Створення функції обробки діалогового вікна, прототип якої має такий самий вигляд, як і для будь-якого іншого вікна має деякі особливості:

- функція вікна діалогової панелі не обробляє повідомлення WM_CREATE, WM_PAINT та WM_DESTROY;
- безпосередньо перед відображення будь-якого діалогового вікна функція отримує повідомлення WM_INITDIALOG (замість повідомлення WM_CREATE для звичайних вікон). В цьому повідомленні можна ініціалізувати змінні для роботи з даними, з елементами керування (встановити початковий статус елементів керування, заповнити списки, текстові поля тощо);
- інформація від органів керування повинна оброблятися у повідомленні WM_COMMAND (це ж повідомлення використовується при обробці пунктів меню). Слід пам'ятати, що серед повідомлень, які обробляє діалогове вікно, обов'язково повинно бути передбачено завершення роботи з діалоговим вікном. Це може бути здійснено, наприклад, при обробці кнопок OK і CANCEL, які завжди пропонуються при створенні шаблону діалогового вікна;
- завершення роботи з діалоговою панеллю відбувається при виконанні виклику функції *EndDialog()*;
- функція вікна діалогової панелі не містить виклик функції *DefWindowProc()* для тих повідомлень, які вона не обробляє.

У програмі, яка реалізовуватиме завдання курсової роботи, діалогові вікна можна використати для введення параметрів шифрування, для введення паролю доступу, для введення параметрів гри, вибору параметрів шрифту (колір, гарнітура) тощо.

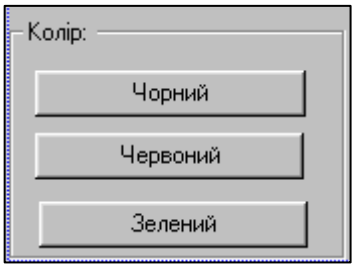
3.9 Елементи керування

Елементи керування – це вікна більш низького відносно дочірнього вікна рівня [9, 10]. Вони ніколи не можуть бути використані самостійно, а завжди на тлі якогось вікна (батьківського). Кожний елемент керування має свій ідентифікатор. Програміст сам повинен вибирати елементи керування і обґрунтувати свій вибір.

При створенні шаблону діалогового вікна за допомогою програмного середовища Visual C++ з'являється панель Control, в якій можна вибрати і розмістити потрібний елемент і задати його характеристики. Дане елементу ім'я (ідентифікатор) надалі використовується при вказанні імені при обробці повідомлення WM_COMMAND.

При додаванні будь-якого елемента керування у діалогове вікно при використанні програмного середовища Visual C++ автоматично здійснюються зміни у двох файлах: у файлі "*.rc" – файлі, що описує фізичні параметри певного елементу керування такі, наприклад, як координати розташування, розміри, стилі, первинний стан і т.д. Ці параметри елементів керування підбираються при оформленні його за допомогою візуального середовища; у файлі "resource.h" – заголовочному файлі, в якому для кожного елемента керування з'являється рядок, що вказує на ідентифікатор, який ми надали елементу керування при його створенні, і значення, яке надала йому операційна система.

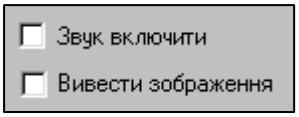
Серед основних елементів керування такі.



Статичні елементи – StaticText та GroupBox – це вікна класу static. Вони не посилають батьківському вікну повідомлень, тобто і обробляти їх немає потреби.

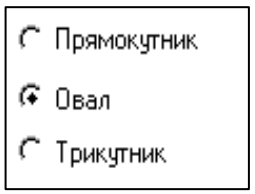
Звичайні кнопки – Button. Кнопки створюються на базі класу button. Діалогове вікно вже передбачає дві стандартні кнопки: "OK" і "CANCEL", яким в системі присвоєно ідентифікатори IDOK та IDCANCEL. Замість них ми можемо ввести і інші кнопки, з іншими заголовками, з іншими ідентифікаторами, і обробити їх по-своєму.

Кнопки з незалежною фіксацією (прапорці) – CheckBox. Це різновид кнопок, які використовують для вибору режимів роботи додатка. Ці кнопки також створюються на базі класу button. Прапорці відносяться до перемикачів з незалежною фіксацією. У групі з декількох перемикачів одночасно можуть бути включені декілька перемикачів.

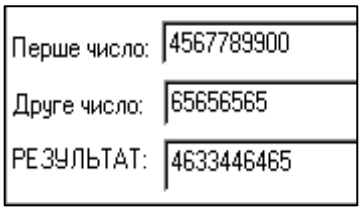


Для обробки повідомлень від прапорців і посилання їм повідомлень використовується функція *SendDelItemMessage()*. Третій параметр функції якраз і відповідає за те, яке повідомлення посилається даному елементу керування. Це ідентифікатор з іменем, яке починається префіксом BM_.

Кнопки із залежною фіксацією (селекторні кнопки) – Radio Button. Селекторні кнопки, як і звичайні прапорці, також створюються на базі класу button. Але це специфічний вид кнопок, який також використовується для вибору режимів роботи додатків. У додатках ці перемикачі використовуються аналогічно кнопкам перемикання діапазонів у радіоприймачі. В одній групі розташовують декілька таких радіоперемикачів, причому включеним може бути лише один. Включення одного перемикача в групі викликає відключення решти. Обробка повідомлень від радіоперемикачів (встановлення статусу, отримання статусу) відбувається таким саме чином, як і для кнопок з незалежною фіксацією (прапорців).



Текстові вікна *(вікна введення)* – Edit Box. Текстове вікно – це також вікно, але специфічного призначення, це однорядковий текстовий редактор. Створюються текстові вікна на базі класу edit. Його використовують для введення значень символьних або числових змінних, а також для створення і редагування текстових файлів. Цей редактор вміє виконувати функції виділення тексту, працювати з буфером обміну. Ось лише деякі функції роботи з текстовими вікнами.

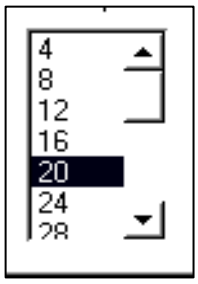


GetDlgItemText() – прочитати вміст текстового вікна,

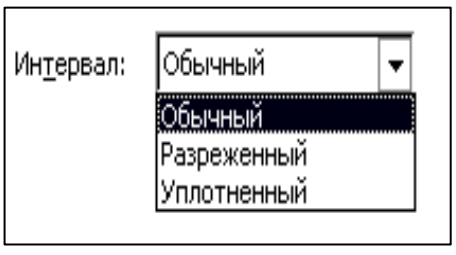
GetDlgItem() – отримати дескриптор текстового вікна,

SetWindowText() – посилання текстового повідомлення вікну.

Списки рядків – ListBox – створюються на базі класу listbox. Рядки у списку нумеруються, починаючи з 0. Списки можуть бути розташовані в одну колонку і в багато колонок, з вертикальною (для однієї колонки) і горизонтальною (для багатьох колонок) смугами прокручування. Послати повідомлення списку і "взяти" у нього інформацію можна за допомогою функції *SendDlgItemMessage(),* третій параметр якої задає код повідомлення і починається з префікса LB_.



Комбінований список – ComboBox – створений на базі класу combobox. Він є комбінацією однорядкового редактора тексту і звичайного списку. Робота з комбінованим списком багато в чому схожа на роботу із звичайним списком, хоча і є деякі відмінності. Для керування комбінованим списком використовується набір повідомлень, аналогічний повідомленням від списку, але префікс цих повідомлень CB_.



Зазвичай використання діалогових вікон з елементами керування в них свідчать про розуміння розробниками конкретних задач, на які розбивається виконання завдання курсової роботи.

3.9 Використання додаткових елементів керування

Кожний програмний додаток для операційної системи Windows має привабливий вигляд, якщо в ньому використовуються додаткові можливості для організації зручного інтерфейсу:

- стандартні діалогові вікна,
- акселератори (гарячі клавіші),
- панелі інструментів,
- рядки стану і т.д.

Стандартні діалогові панелі. При створенні додатків часто виникає необхідність використовувати стандартні діалогові панелі, до яких відносяться: панель для відкриття і збереження файлів; панель для вибору кольору; панель для вибору шрифтів.

Панелі для відкриття та збереження файлів використовуються майже у всіх додатках для Windows [12]. Оскільки така можливість вже реалізована, то немає необхідності створювати її заново. Слід лише правильно використовувати цю можливість користувацького інтерфейсу. Це здійснюється за допомогою таких функцій:

GetOpenName() – відкриття панелі для вибору імен файлів;

GetSaveName() – відкриття панелі збереження файлів.

Обидві ці функції використовують структуру типу, в якій містяться дані для ініціалізації панелі, – OPENFILENAME. Після завершення роботи функцій ця структура доповниться даними про вибрані імена файлів.

Стандартна панель для вибору кольорів – зручний інструмент, з яким користувачі зустрічаються майже у будь-якому текстовому або графічному редакторі [9-11]. Для її створення викликають функцію:

ChooseColor ().

Параметр функції – покажчик на структуру типу CHOOSECOLOR, в якій знаходяться дані для ініціалізації панелі кольорів, а після виклику панелі буде знаходитись результуючий колір.

Для виклику стандартної панелі вибору атрибутів логічного шрифту використовують функцію [9-11]:

ChooseFont().

Параметр функції типу CHOOSEFONT вказує на структуру типу, яка ініціалізує панель і повертає в своїх полях вибрані значення.

Акселератори. Для швидкого доступу до елементів керування програми використовують акселератори. Усі використовувані акселератори записують в одну таблицю (масив структур типу ACCEL) і працюють з дескриптором цієї таблиці. Для цього використовуються такі функції:

CreateAcceleratorTable() – створення таблиці акселераторів;

TranslateAcceleratorTable() – формування відповідних повідомлень при натисканні клавіш-акселераторів.

Панель інструментів і рядок стану. Ці елементи керування є вікнами додаткових класів API Win32, і тому для їх використання у програмі слід підключити заготовочний файл <commctrl.h>, функції якого знаходяться у динамічній бібліотеці comctl32.dll.

Кнопки панелі так само, як і акселератори, пов'язані з командами. Так само створюється таблиця дескрипторів кнопок – масив структур типу TBUTTON, де описуються і самі кнопки, і їх вигляд, і пункти меню, з якими вони пов'язані, і стиль відображення і інші параметри. Різниця полягає в структурі кнопок, функції створення і формі відображення панелі. Для створення панелі інструментів викликають функцію

CreateToolBar().

Рядок стану використовують для відображення текстів про поточний стан додатка.

Для роботи з рядком стану використовують функції:

CreateStatusWindow() – створення рядка стану і розділення його на декілька частин, кожна з яких являє собою вікно;

SendMessage() – посилання повідомлення і одне з вікон рядка стану.

Таким чином, мова C++ і програмне середовище Visual C++ надає розробникам велику кількість програмних засобів як для реалізації основної задачі програми, так і для формування зручного інтерфейсу.

4 ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

4.1 Загальні правила оформлення

При оформленні пояснювальної записки необхідно дотримуватись вимог до КР за ДСТУ 3008-95. Текст пояснювальної записки повинен бути набраний на комп'ютері та роздрукований на принтері.

Шрифт і відступи. Текст пояснювальної записки повинен бути набраний у будь-якому текстовому редакторі шрифтом Times New Roman розміром 14 з інтервалом між рядками 1.5. Шрифт та міжрядковий інтервал у додатках можуть бути довільними, але такими, щоб можна було прочитати і зрозуміти. Відступи: зліва – 2.5 см, справа – 1 см, решта – 2.0 см.

Нумерація сторінок. Сторінки повинні бути пронумеровані, починаючи з третьої (зміст), у правому верхньому кутку сторінки. Нумерація додатків продовжує основну нумерацію.

Оформлення розділів і підрозділів. Структурними елементами основної частини ПЗ є розділи, підрозділи, пункти, підпункти, переліки.

Розділ – головна ступінь поділу тексту, позначена номером і має заголовок. *Підрозділ* – частина розділу, позначена номером і має заголовок. *Пункт* – частина розділу чи підрозділу, позначена номером і може мати заголовок. *Підпункт* – частина пункту, позначена номером і може мати заголовок. Заголовки структурних елементів необхідно нумерувати тільки арабськими числами.

Кожен розділ рекомендується починати з нової сторінки. Заголовок розділу записують посередині (ДСТУ 3008-95) великими літерами з більш високою насиченістю.

Заголовки розділів, підрозділів, пунктів та підпунктів (при наявності заголовка) записують з абзацу малими літерами, починаючи з великої. Перед заголовком і після нього пропускають один рядок.

Розділи нумерують порядковими номерами в межах всього документа (1, 2, і т.д.). Підрозділи нумерують в межах кожного розділу, пункти – в межах підрозділу і т.д. за формою (3.1, 3.2, 3.2.1, 3.2.2, 3.2.2.1 і т.д.). Цифри, які вказують номер, не повинні виступати за абзац. Після номера крапку не ставлять, а пропускають один знак.

Допускається розміщувати текст між заголовками розділу і підрозділу, між заголовками підрозділу і пункту. Посилання в тексті на розділи виконується за формою: “...наведено в розділі 3”.

Оформлення таблиць. Таблицю розміщують симетрично до тексту після першого посилання на даній сторінці або на наступній, якщо на даній вона не уміщується і таким чином, щоб зручно було її розглядати без

повороту або з поворотом на кут 90°. Таблиці у тексті пояснювальної записки набираються основним шрифтом, в деяких випадках розмір шрифту може бути зменшений до 10-12. Підписи таблиць розташовуються над таблицею з вказанням її номера і назви, вирівнявши по лівому краю таблиці. Наприклад,

Таблиця 1 – Основні типи даних

Номер	Тип даних	Опис
1	Float	Дійсні числа з плаваючою точкою
2	Integer	Цілі числа
…	**… … …**	**… … …**

На всі таблиці мають бути посилання за формою “ ... в табл. 1 або в дужках по тексту (табл. 1). Посилання на раніше наведену таблицю дають зі скороченим словом ”дивись” (див. табл. 1) за ходом чи в кінці речення.

При перенесенні частин таблиці на інші сторінки, повторюють або продовжують найменування граф. Допускається виконувати нумерацію граф на початку таблиці і при перенесенні частин таблиці на наступні сторінки повторювати тільки нумерацію граф. У всіх випадках найменування (при його наявності) таблиці розміщують тільки над першою частиною, а над іншими частинами зліва пишуть “Продовження таблиці 1” без крапки в кінці, наприклад,

Продовження таблиці 1

1	2	3
25	String	Рядок символів
26	Char	Символьні літерали
…	**… … …**	**… … …**

Оформлення рисунків. Розміщують рисунки в тексті або в додатках. В тексті ілюстрацію розміщують симетрично до тексту після першого посилання на неї або на наступній сторінці, якщо на даній вона не уміщується без повороту. На всі рисунки мають бути посилання за формою: “ ... на рис. 3–5”, або в дужках по тексту (рис. 3.6). Посилання на раніше наведений рисунок дають зі скороченим словом ”дивись” (див. рис. 4) за ходом чи в кінці речення.

Кожен рисунок повинен мати номер і підпис, розташовані під рисунком по центру. Крапку в кінці не ставлять, знак переносу не використовують. Якщо найменування рисунка довге, то його продовжують у наступному рядку, починаючи від найменування. Наприклад,

Рисунок 12 – Вигляд екрана при обробці пункту меню “Файл”

Між ілюстрацією і текстом пропускають один рядок.

Нумерують ілюстрації в межах розділів, вказуючи номер розділу і порядковий номер ілюстрації в розділі, розділяючи крапкою. Дозволяється нумерувати рисунки в межах всього документа.

Оформлення формул. Кожну формулу записують з нового рядка, симетрично до тексту, курсивом. Між формулою і текстом пропускають один рядок. Умовні літерні позначення в формулі наводять в тексті або зразу ж під формулою. Для цього після формули ставлять кому і записують пояснення до кожного символу з нового рядка в тій послідовності, в якій вони наведені у формулі, розділяючи крапкою з комою. Перший рядок повинен починатися з абзацу зі слова “де” і без будь-якого знака після нього.

Всі формули нумерують в межах розділу арабськими числами. Номер вказують в круглих дужках з правої сторони, в кінці рядка, на рівні закінчення формули. Номер формули складається з номера розділу і порядкового номера формули в розділі, розділених крапкою. Дозволяється виконувати нумерацію в межах всього документа.

Формула є частиною речення, тому до неї застосовують такі ж правила граматики, як і до інших членів речення. Якщо формула знаходиться в кінці речення, то після неї ставлять крапку. Формули, які йдуть одна за одною і не розділені текстом, відокремлюють комою.

4.2 Структура пояснювальної записки

Пояснювальна записка повинна відповідати індивідуальному завданню, а її оформлення – чинним державним стандартам, які слід враховувати на момент виконання розробки з врахуванням всіх офіційних змін, введених в дію.

Пояснювальна записка повинна мати таку структуру:

1. *Вступна частину,* яка містить:
   - титульний аркуш;
   - індивідуальне завдання;
   - анотацію;
   - зміст.
2. *Основна частина*, яка складається із:
   - вступу;
   - суті курсової роботи;
   - висновків;

- список використаних джерел.

3. *Додатки*, які розміщуються після основної частини пояснювальної записки курсової роботи.

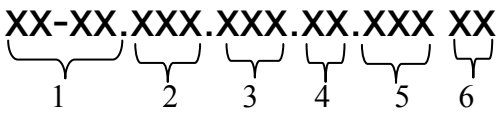
4.3 Вміст вступної частини пояснювальної записки

4.3.1 Титульний аркуш

Титульний аркуш є першою сторінкою КР, яка не нумерується. Згідно з діючим стандартом ДСТУ 3008-95 титульний аркуш виконується за встановленим зразком. Зразок титульного аркушу пропонується у додатка Б. Для курсової роботи титульний аркуш виконується без рамки.

На титульному аркуші для курсових робіт подаються: тема курсової роботи; запис „Пояснювальна записка ...” із зазначенням спеціальності, умовне позначення згідно з прийнятою системою (див. далі); перераховується науковий ступінь та звання керівника. Підписи керівника та студента із зазначенням термінів обов’язкові.

Для курсових робіт доцільною є предметна система умовних позначень, яка має таку структуру:



де 1 (XX-XX) – числовий шифр кафедри, прийнятий у ВНТУ (08-24);

2 (XXX) – умовне скорочення для дисципліни (ПРГ – Програмування, ЗПЗ – Захист програмного забезпечення і т.д.);

3 (XXX) – перша цифра 0, якщо це проект або 1, якщо робота, друга і третя цифри означають рік, наприклад, 08 – 2008 рік);

4 (XX) – варіант завдання на курсову роботу (наприклад, 01, 02, ...);

5 (XXX) – перший символ – номер групи (1 або 2), наступні два символи задають номер студента за списком у журналі академічної групи;

6 (XX) – код документа (ПЗ – пояснювальна записка).

Слід зазначити, що робота, яка подається у вигляді копії, до захисту не приймається.

4.3.2 Індивідуальне завдання

Конкретний зміст кожної КР, етапи виконання визначає керівник на підставі індивідуального завдання, затвердженого завідувачем кафедри і затвердженого на засіданні кафедри.

Попередньо керівник видає індивідуальне завдання до курсової роботи. Індивідуальне завдання в перелік змісту не вноситься і має бути другою сторінкою після титульного аркуша. Зразок індивідуального

завдання до курсової роботи наведено в додатка В. Обов'язковим в індивідуальному завданні є наведення вхідних і вихідних даних.

Керівник роботи пропонує зміст пояснювальної записки, як правило, в розроблених методичних вказівках. У навчальних цілях зміст може висвітлюватись в індивідуальному завданні. В залежності від специфіки індивідуального завдання керівник курсової роботи може пропонувати тему, яка підлягає конкретному обґрунтуванню та розробці індивідуального завдання, яке дещо різниться від типового.

Індивідуальне завдання до курсової роботи має містити термін видачі, підписи керівника та студента.

Завдання на курсову роботу повинно бути підготовлено студентом не пізніше другого тижня з початку навчального семестру, підписано викладачем, що видав завдання і студентом, що прийняв його до виконання.

4.3.3 Анотація

Анотація призначена для ознайомлення з текстовим документом курсової роботи. Анотація повинна коротко характеризувати мету роботи, засоби, використані для досягнення поставленої задачі, коротку інформацію про досягнуті результати. Розмір анотації повинен становити приблизно ⅓ частину сторінки (не перевищувати ½ сторінки).

Анотацію розміщують безпосередньо за аркушем з індивідуальним завданням, починаючи з нової сторінки (третьої), нумерація якої не зазначається і в зміст не входить.

4.3.4 Зміст

Зміст розташовують безпосередньо після анотації, починаючи з нової сторінки. До змісту включають: вступ; послідовно перелічені назви всіх розділів, підрозділів, пунктів і підпунктів (якщо вони мають заголовки) суті роботи; висновки; перелік використаних джерел; назви додатків і номери сторінок, які містять початок матеріалу. Зміст не включає титульний лист, індивідуальне завдання на курсову роботу та анотацію. Нумерація у змісті починається зі ВСТУПУ (відповідно до нумерації у пояснювальній записці). Сам зміст за нумерацією пояснювальної записки є четвертою сторінкою. Нумерація сторінок повинна бути наскрізною, включаючи додатки.

Назви заголовків змісту повинні однозначно відповідати назвам заголовків пояснювальної записки за текстом. Формування змісту у текстовому документі бажано здійснювати автоматично, використовуючи засоби обраного текстового редактора.

Приклад оформлення змісту:

ВСТУП
1 РОЗРОБКА ...

1.1 Варіанти ...
1.1.1 ...
…….
2 ЗАГОЛОВОК ДРУГОГО РОЗДІЛУ ….
2.1 Заголовки підрозділів
2.1.1 ...
…..
3 ЗАГОЛОВОК ТРЕТЬОГО РОЗДІЛУ
3.1 Заголовки підрозділів
3.1.1 ...
…..
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТКИ
Додаток А. Назва першого додатка
Додаток Б. Назва другого додатка
…..

4.4 Вміст і оформлення основної частини

4.4.1 Вступ

Вступ пишуть з нової пронумерованої сторінки із заголовком ВСТУП посередині (ДСТУ 3008-95) великими літерами з більш високою насиченістю (жирністю) шрифту.

Текст вступу повинен бути коротким і висвітлювати питання актуальності, значення, сучасний рівень і призначення курсової роботи. У вступі і далі за текстом не дозволяється використовувати скорочені слова, терміни, крім загальноприйнятих. Якщо у вступі і далі за текстом використовується деяке загальновживане поняття у вигляді абревіатури, то при першій появі цього поняття воно наводиться повністю, а поруч у дужках наводиться скорочення. При повторному використанні введеного поняття можна наводити лише скорочення у вигляді абревіатури.

Вступ висвітлює:

- стан розвитку проблеми в даній галузі, до якої має відношення розробка (важливість нових технологій програмування, програмних середовищ, API-програмування, мов програмування тощо);
- галузь використання та призначення даної розробки;
- мету та загальну постановку задачі;
- актуальність, яка повинна подаватись в останньому абзаці вступу з метою стислого викладання суті обраної розробки.

Обсяг вступу не повинен перевищувати 1-2 сторінок.

4.4.2 Розробка і обґрунтування структури програми

В даному розділі повинна бути детально описана мета роботи, алгоритм досягнення результату, передбачена розробка загальної схеми функціонування програми.

В цьому розділі формуються і обґрунтовуються основні вимоги до технічних і програмних показників розробки, які надалі у стислому вигляді будуть подані у вигляді інструкцій, вимоги до характеристик програми, яка розробляється для розв'язання поставленої задачі. Тут наводиться математична модель (якщо необхідно), проводяться дослідження використовуваних методів, аналіз роботи методів і алгоритмів на простих прикладах. Результатом такого аналізу повинні бути розроблені блок-схеми алгоритмів, схеми взаємозв’язків між підпрограмами.

Підготовка інтерфейсу програми, що розробляється, також повинна знайти відображення у даному розділі, які види вікон треба використати і чому, які пункти меню доцільно включити і для чого, які елементи керування слід передбачити і для чого, – все це повинно знайти відображення в даному розділі. В результаті повинна бути розроблена схема роботи програми.

Кожна програма повинна мати певні вхідні і вихідні дані, як вони будуть готуватись, звідки вводитись у програму, де зберігатись, що буде вихідними даними і у якому вигляді – це також повинно бути описано у цьому розділі або подано у вигляді схеми даних.

Крім того, в даному розділі слід розробити послідовність основних кроків, необхідних для створення конкретного програмного продукту. Якщо використовуються якісь особливі методи або логічні розв’язування поставленої задачі, їх теж потрібно описати. Таким чином, даний розділ повинен бути підготовкою для наступного етапу – етапу програмування.

4.4.3 Використання схем

Окремими підпунктами основного розділу або завершенням певних підрозділів бажано розробити і описати різні схеми:

- схеми даних;
- схеми програм;
- схеми роботи системи;
- схеми взаємодії програм;
- схеми ресурсів системи.

Розробник програми сам повинен вирішувати, які саме схеми доцільно розробляти у своїй роботі.

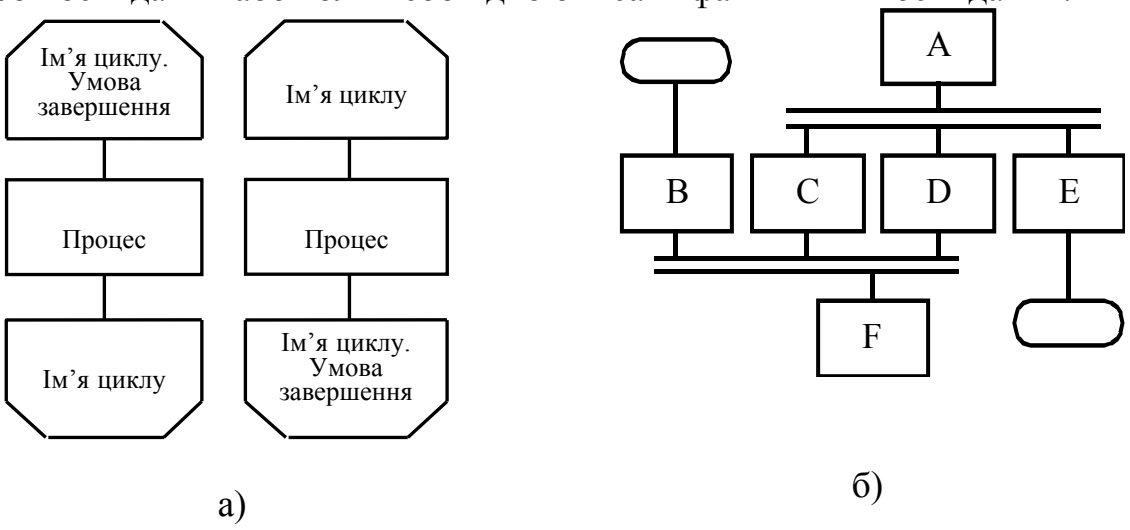
Схеми можуть використовуватися на різних рівнях деталізації, причому кількість рівнів залежить від розмірів і складності задачі оброблення даних.

Схеми алгоритмів, програм, даних і систем складаються з символів, що мають задане значення, короткого тексту пояснення і з'єднувальних ліній. Усі символи поділяються на такі підгрупи:

- символи даних (додаток Г, табл. Г. 1);
- символи процесів (додаток Г, табл. Г. 2). Приклад наведено на рис.1.;
- спеціальні символи (додаток Г, табл. Г. 3). Приклад наведено на рис.2;
- символи ліній (додаток Г, табл. Г. 4). Приклад наведено на рис.3.

Символи поділяються на:

- основні, для випадків, коли точний вигляд процесу або носія даних невідомий або відсутня необхідність в описі фактичного носія даних;
- специфічні, використовувані тоді, коли відомий точний вигляд процесу або носія даних або коли необхідно описати фактичний носій даних.



а) б)

Рисунок 1 – Приклад застосування символів процесу (а – використання сиволів меж циклу; б – використання символів паралельних дій)

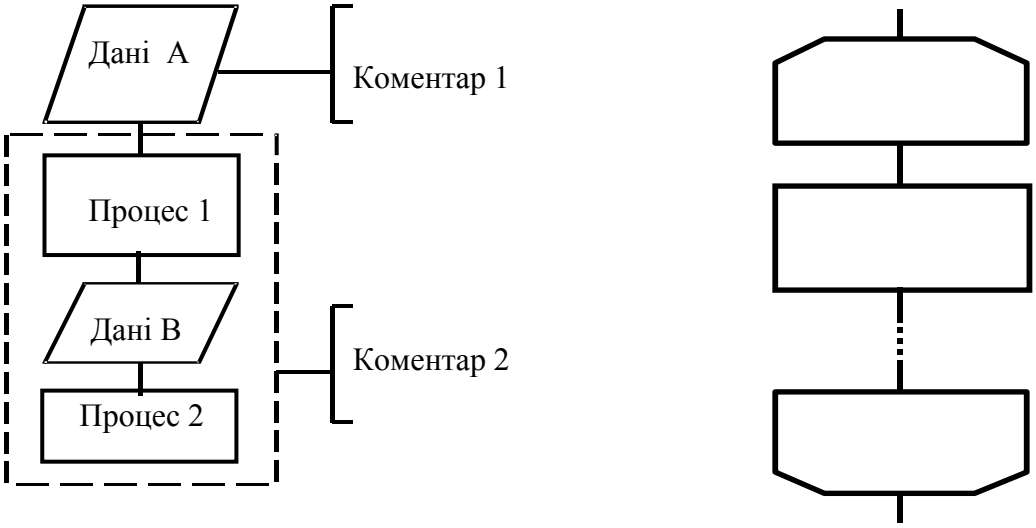


Рисунок 2 – Приклади використання спеціальних символів

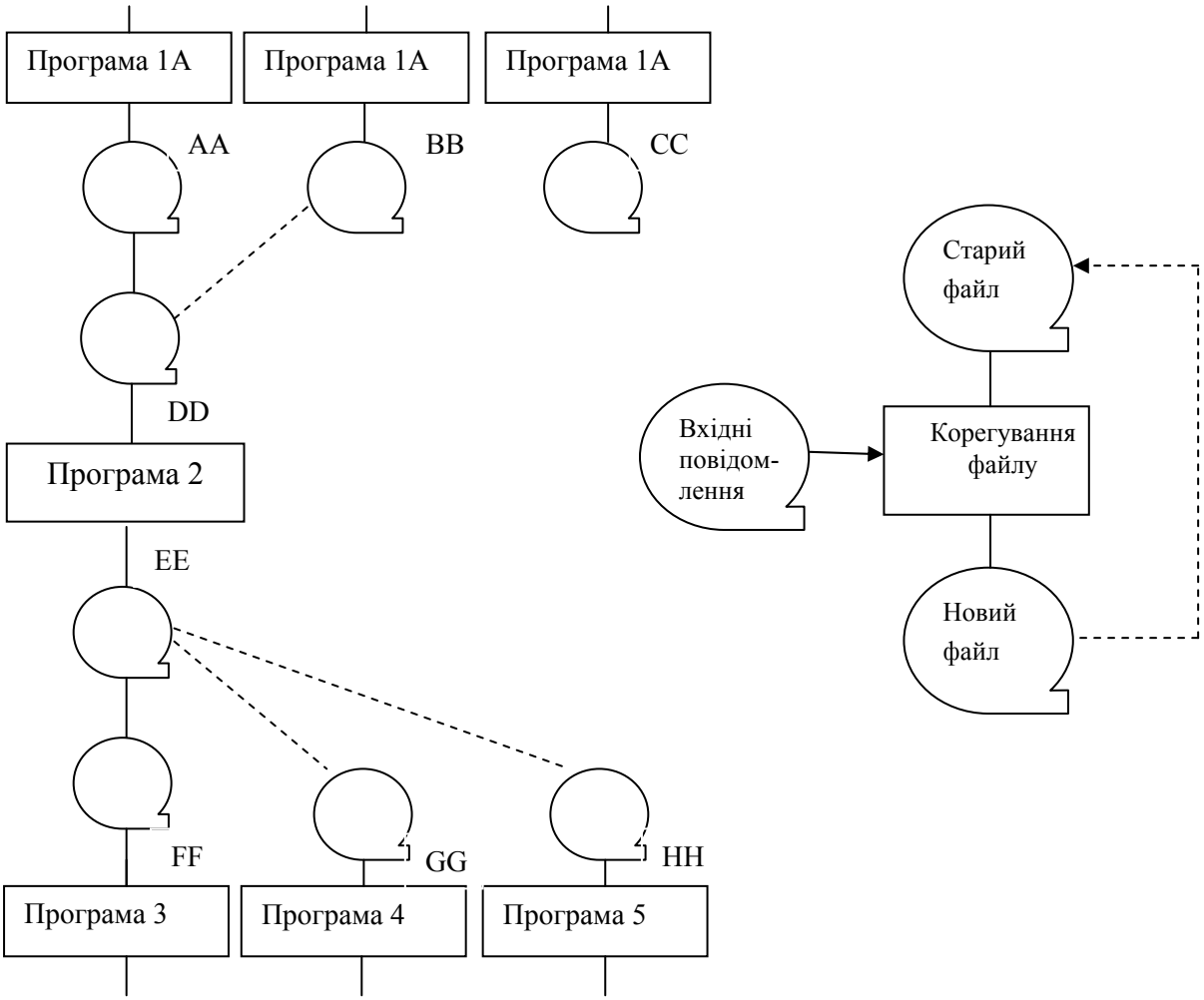


Рисунок 3 – Приклади використання символів ліній

Схеми даних відображають шлях даних при розв'язанні задач і визначають етапи обробки, а також різні використовувані носії даних. Схема даних складається з:

- символів даних, які можуть також вказувати вид носія даних;
- символів процесу, який виконується над даними (можуть також вказувати функції, виконувані обчислювальною машиною);
- символів ліній, які вказують потоки даних між процесами і (або) носіями даних;
- спеціальних символів для полегшення написання і читання схеми.

Символи даних чергуються з символами процесу. Схема даних починається і закінчується символами даних (за винятком спец. символів).

Схеми програм відображають послідовність операцій в програмі. Схема програми складається з:

- символів процесу, що вказують фактичні операції обробки даних (включаючи символи, що визначають шлях, якого слід дотримуватися з урахуванням логічних умов);
- лінійних символів, що вказують потік управління;
- спеціальних символів, використовуваних для полегшення написання і читання схеми.

Схеми роботи системи відображають управління операціями і потік даних в системі. Схема роботи системи складається з:

- символів даних, що вказують на наявність даних (символи даних можуть також вказувати на вид носія даних);
- символів процесу, що вказують операції, які слід виконати над даними, а також визначають логічний шлях, якого слід дотримуватися;
- лінійних символів, що вказують на потоки даних між процесами і (або) носіями даних, а також потік управління між процесами;
- спеціальних символів, використовуваних для полегшення написання і читання блок-схеми.

Схема взаємодії програм відображає шлях активацій програм і взаємодій з відповідними даними. Кожна програма в схемі взаємодії програм показується тільки один раз (у схемі роботи системи програма може зображатися більше, ніж в одному потоці управління). Схема взаємодії програм складається з:

- символів даних, що вказують на наявність даних;
- символів процесу, що вказують на операції, які виконують над даними;
- лінійних символів, що відображають потік між процесами і даними, а також ініціації процесів;
- спеціальних символів, використовуваних для полегшення написання і читання схеми.

Схема ресурсів системи відображає конфігурацію блоків даних, блоків обробки цих даних, яка потрібна для розв’язання задачі або набору задач. Схема ресурсів системи складається з:

- символів даних, що відображають вхідні, вихідні і запам'ятовують пристрої обчислювальної машини;
- символів процесу, що відображають процесори (центральні процесори, канали і т.д.).
- лінійних символів, що відображають передачу даних між пристроями введення-виведення і процесорами, а також передачу управління між процесорами;
- спеціальних символів, використовуваних для полегшення написання і читання схеми.

4.4.4 Програмна реалізація задачі

Необхідно дати обґрунтування того, чому ті або інші засоби програмування доцільно використати для реалізації певних цілей завдання. Проводиться покроковий опис програмної реалізації алгоритму поставленої задачі, додержуючись принципів структурного програмування. Описуються основні структури мови програмування, які використані в даній роботі. Наводяться фрагменти (не більше в 5-10 операторів, а не

програма цілком!). Описуються всі основні змінні, а також допоміжні масиви, якщо вони використовуються при розв'язанні задачі. Всі допоміжні модулі, класи, процедури і функції, основні і заголовочні файли, які використовуються при розробці програмного забезпечення, також повинні бути описані коротко, якщо вони є стандартними, і детально, якщо вони є продуктом роботи студента.

Якщо розроблено власні підпрограми або функції, вони повинні бути описані і дано посилання на ті додатки, де знаходиться лістинг програми. Якщо використовуються стандартні функції, навести їх доцільність та мету використання і внутрішнє наповнення цих функцій (також з посиланням на відповідні додатки).

Оскільки програма розробляється для її використання в операційному середовищі Windows, то слід звернути особливу увагу на повідомлення від операційної системи та їх обробку і на використання API-функцій.

Крім того, оскільки розроблена програма має бути повноцінним додатком Windows, то необхідно докладно описати програмні засоби для реалізації користувацького інтерфейсу: які діалогові вікна використані, які елементи керування за що відповідають, які засоби використані для реалізації підказок, надання допомоги, виведення повідомлень.

Даний розділ також має бути дуже змістовним, конкретним, зрозумілим, оскільки саме він демонструє знання та навички у галузі програмування. Рекомендований обсяг даного розділу – 8-10 сторінок пояснювальної записки.

4.4.5 Тестування програми і розробка інструкцій

Цей розділ повинен бути присвячений тестуванню розробленої програми для формування інструкцій та рекомендацій для роботи з нею. В ньому необхідно продемонструвати весь хід виконання програми на всіх режимах її роботи. Для цього слід підготувати різні комплекти вхідних даних, правильні і такі, що призводять до помилок. При здійсненні і описі аналізу роботи програми можна наводити ілюстрації (вигляд екрана), що демонструють основні режими і можливості функціонування програми. Ілюстрації можна виносити у додатки, а в тексті пояснювальної записки посилатись на ці додатки.

Оскільки темою даної курсової роботи є розробка програмного продукту, то необхідно передбачити розробку конкретних інструктивних матеріалів для роботи з програмою. Даний підрозділ так і може називатись: "Інструкції для роботи з програмою ...", або "Розробка інструктивних документів з ...", або "Розробка рекомендацій для ...".

Таким чином, слід передбачити розробку рекомендацій для роботи з програмою: інструкцію програмісту, інструкцію системному програмісту,

інструкцію оператору (користувачу), інструкцію з технічного обслуговування. Інструкції для роботи з програмою можна виносити в додатки.

При описі вказаних підпунктів рекомендується наводити безпосередньо по тексту або винести у додатки вигляд діалогових вікон, образи екранів і т.д., що пояснюють наведений текст.

Інструкція з технічного обслуговування. Інструкція з технічного обслуговування повинна містити такі підпункти (відповідно до ГОСТ 19.508-79):

- *вступ* (призначення інструкцій, перелік експлуатаційних документів, якими повинні додатково до інструкцій користуватися при технічному обслуговуванні і експлуатації);
- *загальні вказівки* (порядок технічного обслуговування, вказівки щодо організації і особливостей його проведення);
- *вимоги до технічних засобів* (вказують мінімальний склад технічних засобів, що забезпечують роботу програми);
- *опис функцій* (максимальний перелік функцій, що здійснюються цією програмою; опис сумісного функціонування технічних засобів і програми з вказанням методу обробки помилок; опис організації вхідних і вихідних даних для перевірки роботоздатності; опис взаємодій пристроїв з програмою, результатів взаємодій, висновки із результатів роботи програми).

Інструкція системного програміста. Інструкція системного програміста повинна відповідати ГОСТ 19.503-79 і містити наступні підпункти:

- *загальні відомості про програму* (призначення і функції програми і зведення про технічні і програмні засоби, що забезпечують виконання даної програми);
- *структура програми* (відомості про структуру програми, її складові частини, про зв'язки між складовими частинами і про зв'язки з іншими програмами);
- *настроювання програми* (опис дій для налаштування програми на умови конкретного застосування – налагодження на склад технічних засобів, вибір функцій і ін. При необхідності наводять пояснювальні приклади.);
- *перевірка програми* (опис способів перевірки, що дозволяють дати загальний висновок про роботоздатність програми – контрольні приклади, методи прогону, результати);
- *додаткові можливості* (опис додаткових функціональних можливостей програми і способів їх вибору);
- *повідомлення системному програмісту* (тут повинні бути вказані тексти повідомлень в ході виконання налаштування, перевірки програми, а також в ході виконання програми, опис їх змісту і дій, які необхідно виконати після аналізу цих повідомлень).

Інструкція програміста. Інструкція програміста повинне містити такі підрозділи (відповідно до ГОСТ 19.504-79):

- *призначення і умови застосування програми* (тут слід вказати призначення і функції, виконувані програмою, умови, необхідні для виконання програми – об'єм оперативної пам'яті, вимоги до складу і параметрів пристроїв, вимоги до програмного забезпечення і т.п.);
- *характеристика програми* (опис основних характеристик і особливостей програми – тимчасові характеристики, режими роботи, засоби контролю правильності виконання і самовідновлення програми і т.п.);
- *звернення до програми* (опис процедур виклику програми (способи передачі управління, параметрів і ін.);
- *вхідні і вихідні дані* (опис організації використовуваної вхідної і вихідної інформації і, при необхідності, її кодування);
- *повідомлення* (тексти повідомлень, видаваних програмісту або оператору в ході виконання програми, опис їх змісту і дій, які слід виконати після аналізу цих повідомлень).

Інструкція оператора. Інструкція оператора повинне містити такі підрозділи (відповідно до ГОСТ 19.505-79):

- *призначення програми* (відомості про призначення програми і інформація, достатня для розуміння функцій програми і її експлуатації);
- *умови виконання програми* (умови, необхідні для роботи програми: мінімальний і/або максимальний склад апаратних і програмних засобів і т. п.);
- *виконання програми* (послідовність дій оператора, що забезпечують завантаження, запуск, виконання і завершення програми, опис функцій, формату і можливих варіантів команд, за допомогою яких оператор здійснює завантаження і управління ходом виконання програми);
- *повідомлення оператору* (тексти повідомлень в ході виконання програми, опис їх змісту і відповідні дії оператора: у разі збою, можливості повторного запуску програми і т. п.).

4.4.6 Висновки

Висновки оформляють з нової пронумерованої сторінки посередині (ДСТУ 3008-95) великими літерами більш високої насиченості.

У висновках наводяться основні результати роботи над курсовою роботою. На основі проведених досліджень результатів роботи надаються обґрунтовані висновки щодо переваг та недоліків застосування тієї чи іншої мови програмування, того чи іншого засобу програмування, недоліки та переваги даного програмного продукту, труднощі при розробленні програми та причини, що їх обумовили і можливі шляхи їх подолання, можливі рекомендації прикладного застосування та шляхи (перспективи) удосконалення розробленого програмного забезпечення.

4.4.7 Оформлення переліку використаних джерел

Список містить перелік літературних джерел, на які повинні бути обов'язкові посилання в тексті пояснювальної записки. Література (книги, статті, патенти, журнали) в загальний список записується в порядку посилання на неї в тексті. В даному переліку дається оформлений відповідно до вимог державних стандартів, список тих джерел (книги, підручники, журнали, електронні адреси), які було використано в процесі виконання роботи, і на яку є посилання в тексті пояснювальної записки. Кожне джерело повинно бути вказано разом з видавництвом, роком видання, кількістю сторінок. Літературу записують мовою оригіналу. В списку кожну літературу записують з абзацу, нумерують арабськими цифрами, починаючи з одиниці. Правильне оформлення певного джерела інформації можна переглянути у переліку літературних джерел у будь-якому навчальному посібнику. Якщо у списку використаних джерел є посилання на Інтернет-сторінки, слід наводити разом з назвою Інтернет-сторінки.

Приклад оформлення переліку використаних джерел різного характеру.

Посилання на книги

1. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С. 2-е изд. - СПб.: Вильямс, 2000. - 789 С.
2. Бабаш А. В., Шанкин Г. П. Криптография. Под ред. Шерстюка В. П., Применко Э. А. / Шанкин Г .П.. - М.: СОЛОН-Р, 2002. - 512 С.
3. Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В. Основы криптографии. Учебное пособие. - М.: Гелиос АРВ, 2001. - 480 С.

Посилання на журнали

4. Ершов А. А. Стабильные методы оценки параметров // Автоматика и телемеханика. – 1978. – №8. – С. 86-91.

Посилання на ГОСТ і ДСТУ

5. ГОСТ 7.9-77. Реферат и аннотация. – М.: Издательство стандартов, 1981. – 6 с.

Посилання на патенти

6. Пат. 3818311, США, МКИ НОЗК 17/60. Схема защиты полупроводникового переключателя. – Опубл. 04.05.84.

Посилання на web-сторінки

7. Інформаційна безпека // http://www.domarev.ru/.
8. Научная библиотека // www.abris.crimea.ua/index.php.

4.5 Оформлення додатків

Додатки повинні містити матеріал, який не увійшов в основні розділи пояснювальної записки: лістинги програм, підпрограм та функцій, результати тестування програми у вигляді образів екранів, таблиць, графіків, схеми роботи програм, блок-схеми алгоритмів, схеми ресурсів і даних, схеми взаємодії програм тощо.

Кожен додаток необхідно починати з нової сторінки, вказуючи зверху посередині рядка слово “Додаток” і через пропуск – його позначення. Додатки позначають послідовно великими українськими літерами, за винятком літер Ґ, Є, З, І, Ї, Й, О, Ч, Ь, наприклад, Додаток А, Додаток Б.

Кожен додаток повинен мати тематичний (змістовний) заголовок, який записують посередині рядка малими літерами з першої великої.

Сторінки додатків нумеруються, продовжуючи загальну нумерацію у пояснювальній записці.

Всі додатки включають у зміст, вказуючи номер додатка, заголовок і номер сторінки, з яких вони починаються.

Приклад оформлення додатків можна переглянути у додатках до даних методичних вказівок.

5 ГРАФІК ВИКОНАННЯ КУРСОВОЇ РОБОТИ І ПОРЯДОК ЇЇ ЗАХИСТУ

Рекомендується такий графік виконання курсової роботи, який враховує самостійну роботу студентів під час 4-го триместру (14 тижнів).

Зміст розділу	Термін виконання
Отримання завдання на курсову роботу, розробка і оформлення індивідуального завдання	1-2 тижні
Розробка структури програмного забезпечення: дослідження алгоритму задачі, виконання вручну контрольних прикладів, розробка інтерфейсу, обґрунтування необхідності додаткових засобів, розробка структури вхідних і вихідних даних, підбір необхідних	3-4 тижні
Розробка програмного забезпечення і налагоджування його: програмування та тестування основних процедур та функцій, програмна реалізація інтерфейсу, програмна реалізація роботи з файлами, з елементами керування, реалізація захисту та перевірки цілісності	5-11 тижні
Тестування розробленого програмного продукту та виправлення виявлених недоліків. Підготовка контрольних прикладів.	11 тиждень
Оформлення пояснювальної записки до курсової роботи, розробка рекомендацій для роботи з розробленою програмою	12-13 тижні
Здача курсової роботи на попередню перевірку: демонстрація роботи програми та чернетки пояснювальної записки (бажано тверда копія, але можливий електронний варіант)	13 тиждень
Корегування і доповнення (при необхідності) програми згідно із зауваженнями керівника курсової роботи, врахування і виправлення	13-14 тижні
Захист курсової роботи	13-14 тижні

Готовність до захисту курсової роботи визначає керівник за результатами попередньої перевірки якості пояснювальної записки та роботоздатності програми (згідно із графіком попереднього захисту). Записка повинна бути здана керівнику на перевірку не менше, як за тиждень до визначеного терміну захисту роботи. Якщо робота виконана в повному обсязі і не має принципових помилок, керівник допускає студента до захисту. В іншому випадку робота повертається студенту на доопрацювання протягом вказаного терміну. Після позитивного висновку про готовність курсової роботи студент повинен захистити її перед комісією у складі двох викладачів, які призначені кафедрою.

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Либерти ДжесС. Освой самостоятельно С++ за 21 день. – Москва-Санкт-Петербург-Киев: «Вильямс», 2001. - 815 с.
2. Павловская Т .А. Программирование на языке высокого уровня С/С++. – С.-П.: «Питер», 2002. - 460 с.
3. Круподьорова Л. М., Пєтух А. М. Технологія програмування мовою Сі. Частина 1. Навчальний посібник. – Вінниця: ВНТУ, 2006. – 206 С.
4. Герберт Шилдт, Полный стправочник по С. – Москва-Санкт-Петербург-Киев:: «Вільямс», 2003. – 800 с.
5. Грегори К. Использование Visual С++ 6. – Москва-Санкт-Петербург-Киев: «Вильямс», 2000. – 850 с.
6. Глушаков С. В., Коваль А. В., Черепнин С. А. Программирование на Visual С++ 6.0. – Харьков: Фолио, 2002. – 726 с.
7. Глушаков С. В., Коваль А. В., Смирнов С.В.. Язык прграммирования С++. – Харьков: «Фолио», 2002. - 500 с.
8. Гради Буч. Объектно-ориентированное программирование с примерами применения.: Пер. с англ.. – М.: Конкорд, 1992. - 519 с.
9. Ганеев Р. М. Проектирование пользовательского интерфейса средствами Win32API. – М.: Горячая линия - Телеком, 2001. - 334 с.
10. Румянцев П. В. Азбука программирования Win32API. – М.: Горячая линия - Телеком, 2001. - 310 с.
11. Семеренко В. П. Програмування мовами С та С++ в середовищі Windows. Навчальний посібник. – Вінниця: ВДТУ, 2002. – 128 с.
12. Румянцев П. В. Работа с файлами в Win32API. – М.: Горячая линия - Телеком, 2002. - 216 с.
13. Бабаш А. В., Шанкин Г. П. История криптографии. Ч.1. – М.: Гелтос АРВ, 2002. – 240 с.
14. Кухар В. М., Тадіян С. І. Математика. Множини. Логіка. Цілі числа. Практикум. – Київ: Вища школа, 1989. – 333 с.

Додаток А

Варіанти завдань на курсову роботу

Варіант 1. Реалізувати **шифрування Даніеля Дефо** [13, С.168]. Сутність його у тому, що у зашифрованому тексті значення мали лише літери, що стоять на парних (або непарних) місцях. Наприклад, фраза "КУРСОВА РОБОТА" після зашифрування може виглядати так:

"КЙУУРЕСИОЛВІАПРЮОЛБФОКТРАС".

Варіант 2. Реалізувати **шифрування "слободское письмо"** [13, С.46]. Ідея використання літер "чужого алфавіту" для засекречування повідомлень отримала розвиток у XVII-XVIII ст.. Вона полягає в написанні українських (російських) слів латинськими літерами.

Варіант 3. Реалізувати шифрування **методом прямої перестановки**, для якої ключем слугує розмір таблиці. Наприклад, повідомлення “ЦЕ МОЯ ПЕРША КУРСОВА РОБОТА” записується у таблицю по стовпцях. Результатом заповнення таблиці розміром 5×5 буде:

Ц	П	К	В	О
Е	Е	У	А	Т
М	Р	Р	Р	А
О	Ш	С	О	Ц
Я	А	О	Б	Е

Після заповнення таблиці текстом повідомлення по стовпцях для формування шифротексту зчитують вміст таблиці по рядках. Якщо шифротекст записувати групами по 5 літер (ключ – розмір таблиці), виходить таку шифроване повідомлення:

ЦПКВО ЕЕУАТ МРРРА ОШСОЦ ЯАОБЕ.

Варіант 4. Реалізувати шифрування **методом прямої перестановки** для випадку, коли запис повідомлення у таблицю відбуватиметься по рядках (див. варіант № 3).

Варіант 5. Реалізувати **шифр маршрутної перестановки (шифр "Сцитала")** [13, С.33]. Відкритий текст записується у прямокутну таблицю з n рядків і m стовпців. Вважається, що довжина тексту t дорівнює $n \cdot m$ (в іншому випадку залишкова частина тексту шифрується окремо з тим самим шифром). Якщо $t<n \cdot m$, то решта пустих клітинок заповнюється довільним чином літерами з алфавіту. Шифротекст записується по цій таблиці за заздалегідь обумовленим "маршрутом" – шляхом, що проходить через усі клітинки таблиці. Ключем шифру є числа n, m та вказаний маршрут.

Варіант 6. Реалізувати шифрування **методом поодинокої перестановки за ключем (система шифрування Фальконета)** [13, С.176]**.**

Даний метод базується на методі, наведеному у варіанті 3, але відрізняється тим, що стовпці таблиці переставляються за ключовим словом, фразою або набором чисел довжиною в рядок таблиці. Наприклад, візьмемо як ключ слово “ДИСКЕТА”, а текст повідомлення візьмемо з попереднього варіанта. Наведемо дві таблиці: одна відповідає заповненню до перестановки, друга – після перестановки.

До перестановки:

Д	И	С	К	Е
1	3	5	4	2
Ц	П	К	В	О
Е	Е	У	А	Т
М	Р	Р	Р	А
О	Ш	С	О	Ц
Я	А	О	Б	Е

Після перестановки:

Д	Е	И	К	С
1	2	3	4	5
Ц	О	П	В	К
Е	Т	Е	А	У
М	А	Р	Р	Р
О	Ц	Ш	О	С
Я	Е	А	Б	О

У верхньому рядку лівої таблиці записаний ключ, а номери під літерами ключа визначені відповідно до природного порядку літер ключа в алфавіті. У разі, якщо в ключі зустрічаються однакові букви, вони були б пронумеровані зліва направо. У правій таблиці стовпці переставлені за порядком номерів літер ключа. При зчитуванні вмісту правої таблиці по рядках отримуємо повідомлення:

ЦОПВКЕТЕАУМАРРРОЦШОСЯЕАБО.

Ключем для даного методу шифрування служать розміри таблиці та ключова фраза. Для розшифрування дії виконуються у зворотному порядку.

Варіант 7. Реалізувати шифрування **методом подвійної перестановки.** Даний метод схожий на метод з варіанта 6, але тут перестановки визначаються окремо для рядків і для стовпців. Спочатку у таблицю записується текст повідомлення, а потім по черзі переставляються стовпці, а потім рядки. При розшифруванні порядок перестановки повинен бути зворотним. Приклад виконання шифрування повідомлення “ЦЕ МОЯ КУРСОВА РОБОТА” методом подвійної перестановки наведений у таблицях:

Початкова таблиця:

	3	1	2
3	Ц	Е	М
1	О	Я	К
6	У	Р	С
4	О	В	А
2	Р	О	Б
5	О	Т	А

Таблиця після перестановки стовпців:

	1	2	3
3	Е	М	Ц
1	Я	К	О
6	Р	С	У
4	В	А	О
2	О	Б	Р
5	Т	А	О

Після перестановки рядків:

	1	2	3
1	Я	К	О
2	О	Б	Р
3	Е	М	Ц
4	В	А	О
5	Т	А	О
6	Р	С	У

Зчитуючи шифротекст з правої таблиці по рядках, отримуємо: ЯКООБРЕМЦВАОТАОРСУ. Ключем до шифру подвійної перестановки служить послідовність номерів стовпців і номерів рядків початкової таблиці (у нашому прикладі 312 та 316425, відповідно).

Варіант 8. Реалізувати **шифрування за допомогою магічних квадратів** [13, С.50]. Магічними квадратами називають квадратні таблиці з вписаними в їх клітинки послідовними натуральними числами, починаючи з 1, які дають в сумі по кожному стовпцю, кожному рядку і кожній діагоналі одне й те саме число. Повідомлення для шифрування вписується у магічні квадрати відповідно до нумерації їх клітинок. Якщо потім виписати вміст такої таблиці по рядках, то отримаємо шифротекст, сформований завдяки перестановці літер початкового повідомлення. В давні часи вважалось. що за допомогою магічних квадратів шифротекст береже не тільки ключ, але й магічна сила.

Приклад магічного квадрата і його заповнення повідомлення “МОЯ КУРСОВА РОБОТА” наведений нижче:

Магічний квадрат:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Заповнення магічного квадрату:

А	Я	О	Б
У	А	Р	О
В	Р	С	О
К	Т	О	М

Шифротекст, що отримуємо при зчитуванні вмісту правої таблиці по рядках, має цілком загадковий вигляд: АЯОБ УАРО ВРСО КТОМ.

Варіант 9. Реалізувати **систему шифрування Цезаря** [13, С.25]**.** Шифр Цезаря є окремим випадком шифру простої заміни (одноалфавітна підстановка). Свою назву цей шифр отримав за іменем римського імператора Гая Юлія Цезаря, який використовував його при переписці з Цицероном (приблизно за 50 р. до н.е.). При шифруванні початкового тексту кожна літера замінювалась на іншу літеру цього ж самого алфавіту за таким правилом. Замінювальна літера визначалась шляхом зміщення за алфавітои початкової літери на К літер. При досягненні кінця алфавіту виконувався циклічний перехід на його початок. Цезар використовував шифр заміни при зміщенні на К=3.

А - Ю	Д - В	И - Ж	Н - Л	С - П	Х - У	Щ - Ч	Э - Ы
Б - Я	Е - Г	К - З	О - М	Т - Р	Ц - Ф	Ъ - Ш	Ю - Ь
В - А	Ж - Д	Л - И	П - Н	У - С	Ч - Х	Ы - Щ	Я - Э
Г - Б	З - Е	М - К	Р - О	Ф - Т	Ш - Ц	Ь - Ъ	

Наприклад, якщо використовувати одноалфавітну підстановку, як показано в таблиці, повідомлення МОЯ ПЕРША КУРСОВА РОБОТА перетвориться у повідомлення КМЭ НГОЦЮ ЗСОПМАЮ ОМЯМРЮ.

Варіант 10. Реалізувати **афінну систему підстановок Цезаря.** Сутність цієї системи полягає в тому, що літера, яка відповідає числу t, замінюється на літеру, що відповідає числовому значенню (at+b) за модулем m, де m – кількість літер алфавіту; a, b – цілі числа, причому a>=0, b<m, НОД(a, m) = 1. Нехай m=31, a=3, b=5. Тоді очевидно, що НОД(3, 31) = 1, і ми отримуємо таку відповідність між числовими кодами букв:

Початковий алфавіт	А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3t+5	5	8	11	14	17	20	23	26	29	1	4	7	10	13	16	19	22
Алфавіт підстановки	Е	И	М	П	Т	Х	Ш	Ы	Ю	Б	Д	З	Л	О	С	Ф	Ч

Початковий алфавіт	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я			
t	17	18	19	20	21	22	23	24	25	26	27	28	29	30			
3t+5	25	28	0	3	6	9	12	15	18	21	24	27	30	2			
Алфавіт підстановки	Ъ	Э	А	Г	Ж	К	Н	Р	У	Ц	Щ	Ь	Я	В			

Якщо візьмемо для шифрування слово “ПРОГРАМУВАННЯ”, за допомогою даної системи підстановки отримаємо зашифроване повідомлення: “СФОПФЕЗЄЬЕЛЛВ”. Позитивною рисою афінної системи є зручне управління ключами – ключі шифрування і розшифрування подаються у компактній формі у вигляді пари (a, b).

Варіант 11. Реалізувати **систему шифрування Цезаря з ключовим словом.** Ця система шифрування є одноалфавітною системою підстановки. Особливістю її є використання ключового слова для зміщення та зміни порядку символів в алфавіті підстановки. Виберемо деяке число k (0≤k≤25) і слово або коротку фразу як ключове слово. Бажано, щоб всі букви ключового слова були різними. Нехай вибрано слово “КУРСОВА” як ключове слово і число k=5. Ключове слово записується під буквами алфавіту, починаючи з букви, числовий код якої збігається з вибраним числом k, а решта літер алфавіту підстановки записуються після ключового слова в алфавітному порядку:

0	1	2	3	4	**5**					10					15					20
А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ы	Ь	Э	Ю	Я	**К**	**У**	**Р**	**С**	**О**	**В**	**А**	Б	Г	Д	Е	Ж	З	И	Л	М

21				25					30
Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Н	П	Т	Ф	Х	Ц	Ч	Ш	Щ	Ъ

Тепер ми маємо підстановку для кожної літери довільного повідомлення. Так, повідомлення Я ЗАХИСТИВ КУРСОВУ шифрується як ЪРЫМСЖЗСЭОИЕЖГЭИ.

Слід зауважити, що вимоги щодо різних літер ключового слова не є обов'язковими. Можна записати ключове слово або фразу без повторення однакових літер.

Варіант 12. Реалізувати шифрування за допомогою **шифруючої таблиці Трисемуса.** У 1508 р. абат з Германії Иоганн Трисемус написав друковану роботу з криптології, в якій вперше систематично описав застосування шифруючих таблиць, заповнених алфавітом у випадковому порядку. Для отримання такого шифру заміни зазвичай використовувалась таблиця для запису літер алфавіту і ключове слово (або фраза). У таблицю спочатку вписувалось по рядках ключове слово, причому літери, що повторювались, відкидалися. Потім ця таблиця заповнювалась літерами з алфавіту, які не увійшли у ключову фразу. Оскільки ключову фразу досить легко зберігати у пам'яті, то такий підхід спрощував процеси шифрування і розшифрування. Наведемо приклад. Для російського алфавіту шифруюча таблиця може мати розмір 4×8. Оберемо ключем слово "БАНДЕРОЛЬ". Шифруюча таблиця з цим ключем буде такою:

Б	А	Н	Д	Е	Р	О	Л
Ь	В	Г	Ж	З	И	Й	К
М	П	С	Т	У	Ф	Х	Ц
Ч	Ш	Щ	Ы	Ъ	Э	Ю	Я

При шифруванні знаходять в цій таблиці чергову літеру відкритого тексту і записують у шифротекст літеру, що розташована на один рядок нижче (або першу літеру цього стовпця, якщо літера розташована у нижньому рядку). Наприклад, при шифруванні повідомлення КУРСОВАЯ РАБОТА отримуємо шифротекст ЦЪИЩЙПВЛИВЬЙЪВ. Такі табличні шифри є монограмними, оскільки шифрування виконується по одній літері.

Варіант 13. Реалізувати застосування **біграмного шифру Плейфейра.** Основою шифру є шифротаблиця з випадково розташованими літерами алфавіту початкового тексту. Для зручності запам'ятовування шифротаблиці відправник і одержувач повідомлення можуть використовувати ключове слово або фразу при заповненні початкових рядків таблиці. В цілому структура шифруючої таблиці системи Плейфейра повністю аналогічна структурі шифруючої таблиці Трисемуса (див. варіант 12). Для пояснення використаємо саме її. Процедура шифрування включає в себе такі кроки.

1. Відкритий текст повідомлення розбивається на пари літер (біграми). Текст повинен мати парну кількість літер і в ньому не повинно бути біграм, що містять однакові літери. Якщо ці вимоги не витримані, то текст модифікується, навіть припускаються незначні помилки.
2. Послідовність біграм відкритого тексту перетворюється за допомогою шифруючої таблиці в послідовність біграм шифротексту за такими правилами:
   - якщо обидві літери біграми відкритого тексту не попадають на один рядок або стовпець, тоді знаходять літери в кутках прямокутника, що визначається даною парою літер(наприклад, пара літер АЙ відображається в пару ОВ);
   - якщо обидві літери біграми відкритого тексту належать одному стовпці, то літерами шифротексту вважаються літери, що лежать під ними (наприклад, біграма НС відображається в ГЩ). Якщо при цьому літера відкритого тексту знаходиться в нижньому рядку, то для шифротексту береться відповідна буква з верхнього рядка одного й того ж самого стовпця (наприклад, ВШ відображається в ПА);

- якщо обидві літери біграми належать одному рядку, то літерами шифротексту вважаються літерами, що лежать справа від них (наприклад, НО відображається ДЛ). Якщо при цьому літера знаходиться у крайньому правому стовпці, то для шифру беруть відповідну літеру з лівого стовпця того ж рядка (наприклад, ФЦ відображається в ХМ).

Зашифруємо, наприклад, текст ВСЕ ТАЙНОЕ СТАНЕТ ЯВНЫМ. Розбиваємо його на біграми: ВС ЕТ АЙ НО ЕС ТА НЕ ТЯ ВН ЫМ. Дана послідовність біграм відкритого тексту перетворюється у таку послідовність: ГП ДУ ОВ ДЛ НУ ПД ДР ЦЫ ГА ЧТ.

При розшифруванні застосовується зворотний порядок дій.

Варіант 14. Реалізувати **багатоалфавітне шифрування.** Багатоалфавітний шифр відноситься до шифрів складної заміни. Для шифрування кожного символу початкового повідомлення застосовують свій шифр простої заміни. Багатоалфавітна заміна послідовно і циклічно змінює використовувані алфавіти. При r-алфавітній підстановці символ x_0 початкового повідомлення заміняють символом y_0 з алфавіту B_0, символ x_1 – символом y_1 з алфавіту B_1, і т.д., символ x_{r-1} заміняють символом y_{r-1} з алфавіту B_{r-1}, символ x_r – символом y_r знову з алфавіту B_0. Наведемо загальну схему багатоалфавітної підстановки для випадку r=4.

Символи початкового повідомлення	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Алфавіт підстановки	B_0	B_1	B_2	B_3	B_0	B_1	B_2	B_3	B_0

Ефект використання багатоалфавітної підстановки полягає в тому, що забезпечується маскування природної статистики початкової мови, оскільки конкретний символ може бути перетворений в декілька різних символів шифрувальних алфавітів.

Варіант 15. Реалізувати **систему шифрування Віжинера** [13, С.97]**.** Система Віжинера (за іменем французького дипломата XVI ст. Блеза Віжинера) вперше була опублікована в 1586 р. і є однією з найстаріших та найбільш відомих багатоалфавітних систем. Даний шифр багатоалфавітної заміни можна описати таблицею шифрування, яка носить назву таблиці (квадрата) Віжинера. Для російського алфавіту вона має вигляд, показаний у додатка Д. Таблиця Віжинера використовується для шифрування і розшифрування. Таблиця має два входи:

– верхній рядок підкреслених символів, що застосовується для зчитування чергової літери початкового відкритого тексту;
– крайній лівий стовпець ключа.

Послідовність ключів зазвичай отримують з числових значень літер ключового слова. При шифруванні початкового повідомлення його вписують в рядок, а під ним записують ключове слово або фразу. Якщо ключ виявився коротшим, ніж повідомлення, то його циклічно повторюють. В процесі шифрування знаходять у верхньому рядку таблиці чергову букву початкового тексту і в лівому стовпці чергове значення ключа. Чергова літера шифротексту знаходиться на перетині стовпця, що визначається літерою, яка шифрується, і рядка, що відповідає значенню ключа.

Розглянемо приклад отримання шифротексту за допомогою таблиці Віжинера. Нехай ми вибрали ключове слово ОЦЕНКА. Зашифруємо повідомлення МОЯ КУРСОВАЯ РАБОТА. Випишемо наше повідомлення в рядок і запишемо під ним ключове слово з повторенням. В третій рядок будемо вписувати літери шифротексту за таблицею Віжинера.

Повідомлення	М	О	Я	К	У	Р	С	О	В	А	Я	Р	А	Б	О	Т	А
Ключ	О	Ц	Е	Н	К	А	О	Ц	Е	Н	К	А	О	Ц	Е	Н	К
Шифротекст	Ь	Е	Д	Ч	Ю	Р	А	Е	З	Н	Й	Р	О	Ч	У	А	К

Варіант 16. Реалізувати шифр **“подвійний квадрат” Уітстона**. Шифр “подвійний квадрат” використовує відразу дві таблиці, розміщені по одній горизонталі, а шифрування здійснюється біграмами, як у шифрі Плейфейра (див. варіант 13). Нехай є дві таблиці з випадково розташованими в них алфавітами.

Ж	Щ	Н	Ю	Р
И	Т	Ь	Ц	Б
Я	М	Е	.	С
В	Ы	П	Ч	
:	Д	У	О	К
З	Э	Ф	Г	Ш
Х	А	,	Л	Ъ

И	Ч	Г	Я	Т
,	Ж	Ь	М	О
З	Ю	Р	В	Щ
Ц	:	П	Е	Л
Ъ	А	Н	.	Х
Э	К	С	Ш	Д
Б	Ф	У	Ы	

Перед шифруванням початкове повідомлення розбивають на біграми. Кожна біграма шифрується окремо. Першу літеру знаходять у лівій таблиці, а другу – у правій таблиці. Потім подумки будують прямокутник таким чином, щоб літери біграм належали його протилежним вершинам. Інші дві вершини цього прямокутника дають букви біграми шифротексту. Припустимо, що шифрується біграма ИЛ. Літера И знаходиться у стовпці 1 і у рядку 2 лівої таблиці. Літера Л знаходиться у стовпці 5 і у рядку 4 правої таблиці. Це означає, що прямокутник утворений рядками 2 і 4 та стовпцями 1 лівої таблиці і 5 правої таблиці. Отже, в біграму шифротексту входять літери О (стовпець 5 і рядок 2 правої таблиці) та літера В (стовпець 1 і рядок 4 лівої таблиці), і ми отримуємо біграму ОВ. Якщо обидві літери біграми повідомлення лежать в одному рядку, то і букви шифротексту беруть з цього ж самого рядка. Першу літеру біграми шифротексту беруть з лівої таблиці у стовпці, що відповідає другій літері біграми повідомлення. Друга літера біграми шифротексту береться з правої таблиці у стовпці, що відповідає першій літері біграми повідомлення. Тому біграма повідомлення ТО перетворюється у біграму шифротексту ЖБ.

Наприклад, зашифруємо повідомлення КУРСОВА РОБОТА. Розбиваємо повідомлення на біграми: КУ РС ОВ АР ОБ ОТ А. Отримуємо шифротекст: НЪ ГШ .. УМ ЪЛ ХЮ ЫД.

Варіант 17. Реалізувати **одноразову систему шифрування.** Одноразова система шифрування винайдена у 1917 р. американцями Дж. Моборном та Г. Вернамом. Дана система шифрує початковий відкритий текст: $X = (X_0, X_1, ..., X_{n-1})$ у шифротекст: $Y = (Y_0, Y_1, ..., Y_{n-1})$ за допомогою підстановки Цезаря: $Y_i = (X_i + K_i) \bmod m, (0 \le i < n)$, де K_i – i-й елемент ключової послідовності, m – кількість літер алфавіту. Процедура розшифрування описується співвідношенням: $Y_i = (X_i - K_i) \bmod m, (0 \le i < n)$.

Для реалізації цієї системи підстановки іноді використовувався одноразовий блокнот. Цей блокнот складений з відривних сторінок, на кожній з яких надрукована таблиця з випадковими ключами. Блокнот виконується у двох екземплярах: один використовується відправником, а другий – одержувачем. Для кожного символу повідомлення використовується свій ключ лише один раз. Після того, як таблиця використана, вона повинна бути видалена з блокнота і знищена. Шифрування нового повідомлення починається з нової сторінки.

Варіант 18. Реалізувати **шифр “квадрат Полібія”** [13, С.36]. Квадрат Полібія – це винахід древніх греків (Полібій – грецький державний діяч, полководець, історик, III ст. до н.е.). Сутність цього шифрування відносно латинського алфавіту з 26 літер (вважалось, що I=J) полягала у тому, що у квадрат розміром 5×5 клітинок вписувались літери алфавіту:

	A	**D**	**C**	**D**	**E**
A	A	B	C	D	E
D	F	G	H	I	K
C	L	M	N	O	P
D	Q	R	S	T	U
E	Y	W	X	Y	Z

Літера, що шифрується, замінювалась на координати квадрату, в якому вона записана. Наприклад, B замінялась на AB, F на BA, R на DB і т.д. При розшифруванні кожна така пара визначала відповідну букву повідомлення. У даному випадку ключ відсутній, оскільки використовується фіксований порядок літер. Все виходить занадто просто. Ускладнений варіант шифру Полібія полягає у запису літер алфавіту у довільному порядку. Цей довільний порядок і є ключем шифру. Але довільний порядок складно запам'ятати, користувачам шифру постійно приходиться тримати при собі ключ-квадрат. Як компроміс було запропоновано ключ–пароль. Пароль виписувався без повторів у квадрат, в решту клітинок вписувались в алфавітному порядку літери алфавіту, що не увійшли у парольну фразу. Такий квадрат вже немає необхідності тримати при собі. Досить лише запам'ятати пароль.

Варіант 18а. Реалізувати **шифр "квадрат Полібія"** [13, С.36] для українського алфавіту (ускладнений варіант) (див. варіант 18).

Варіант 19. Реалізувати **шифр Чейза** [13, С.39]**.** В середині XIX століття американець П. Е. Чейз запропонував таку модифікацію шифру Полібія. У прямокутник 3×10 вписуються літери алфавіту. Ключем шифру є порядок розташування літер у таблиці. Як і у шифрі Полібія з ключовим словом (див. варіант 18) порядок літер у таблиці можна зробити не зовсім випадковим (щоб не тримати при собі ключ-квадрат), а задати ключову фразу. Нехай. наприклад, ключем буде слово "ПРОГРАМА".

	1	2	3	4	5	6	7	8	9	0
1	П	Р	О	Г	А	М	Б	В	Д	Е
2	Ж	З	И	К	Л	Н	С	Т	У	Ф
3	Х	Ц	Ч	Ш	Щ	Ь,Ъ	Ы	Э	Ю	Я

Ключем (вже другим) шифру є порядок розташування літер у таблиці. При шифруванні координати літер виписуються вертикально. Наприклад, слово КУРСОВА прийме вигляд:

2 2 1 2 1 1 1
4 9 2 7 3 8 5

Чейз запропонував ввести третій ключ: заздалегідь обговорене правило перетворення нижнього (верхнього) ряду цифр. Наприклад, число, утворене цим рядом, множиться на 9:

$$4927385 \times 9 = 44346465.$$

Отримуємо новий дворядковий запис:

2 2 1 2 1 1 1
4 4 3 4 6 4 6 5

Тепер цей дворядковий запис знову переводиться у літери згідно з таблицею; при цьому перше число (4) нижнього рядка визначає літеру першого рядка. Шифротекст набуває вигляду:

Г К И Г Н Г М А

Можуть бути використані і інші перетворення координат. Цей шифр сильніший за шифр Полібія, він вже не є шифром простої заміни. При розшифруванні отримана послідовність переводиться у дворядковий запис:

(1) 2 2 1 2 1 1 1

4 4 3 4 6 4 6 5

Нижній ряд ділиться на 9 (44346465 : 9 = 4927385), утворюється дворядковий запис і за ним згідно з таблицею читається відкритий текст.

Варіант 20. Реалізувати **шифрування за допомогою таблиць Тритемія** [13, С.57]**.** Реалізація таблиці Тритемія не потребувала використання якихось механічних застосувань. Таблиця складалася з рядків, кожний з яких являв собою літери алфавіту, зсунуті з кожним рядком на одиницю вліво (додаток Е). При шифруванні перша літера відкритого повідомлення шифрується по першому рядку (перший рядок є одночасно і рядком літер відкритого тексту), друга літера – по другому рядку і т.д. Після використання останнього рядка знову повертаються до першого. Так, наприклад, слово ПРОГРАММА відкритого повідомлення перетвориться у послідовність ПСРЖФЕТЗ шифротексту.

Варіант 21. Реалізувати **шифр Порта** [13, С.87]**.** Цей шифр являє собою прямокутну таблицю з літер алфавіту у такому порядку:

1	**А**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Б	р	с	т	у	ф	х	Ц	ч	ш	щ	ъ	ы	ь	э	ю	я
2	**В**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Г	с	т	у	ф	х	ц	Ч	ш	щ	ъ	ы	ь	э	ю	я	р
3	**Д**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Е	т	у	ф	х	ц	ч	Ш	щ	ъ	ы	ь	э	ю	я	р	с
4	**Ж**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	З	у	ф	х	ц	ч	ш	Щ	ъ	ы	ь	э	ю	я	р	с	т
5	**И**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Й	ф	х	ц	ч	ш	щ	Ъ	ы	ь	э	ю	я	р	с	т	у
6	**К**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Л	х	ц	ч	ш	щ	ъ	Ы	ь	э	ю	я	р	с	т	у	ф
7	**М**	а	б	в	г	д	е	Ж	з	и	й	к	л	м	н	о	п
	Н	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	р	с	т	у	ф	х
8	**О**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	П	ч	ш	щ	ъ	ы	ь	э	ю	я	р	с	т	у	ф	х	ц
9	**Р**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	С	ш	щ	ъ	ы	ь	э	ю	я	р	с	т	у	ф	х	ц	ч
10	**Т**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	У	щ	ъ	ы	ь	э	ю	я	р	с	т	у	ф	х	ц	ч	ш
11	**Ф**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Х	ъ	ы	ь	э	ю	я	р	с	т	у	ф	х	ц	ч	ш	щ
12	**Ц**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Ч	ы	ь	э	ю	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ
13	**Ш**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Щ	ь	э	ю	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы
14	**Ъ**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Ы	э	ю	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь
15	**Ь**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Э	ю	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
16	**Ю**	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	Я	я	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю

Шифрування здійснюється за допомогою секретного ключа, так званого лозунгу. Лозунг періодично виписується над відкритим текстом. За першою літерою лозунгу

розшукується алфавіт (великі літери на початку рядка), у верхньому або нижньому півалфавіті відшукується перша літера відкритого тексту і заміняється відповідною літерою з верхнього або нижнього рядка. Аналогічно шифруються і інші літери (інтерваліи між словами не враховуються). Нариклад, зашифруємо за допомогою ключового слова «ШИФР» послідовність «КУРСОВАЯ РАБОТА».

Ключова фраза	Ш	И	Ф	Р	Ш	И	Ф	Р	Ш	И	Ф	Р	Ш	И
Відкритий текст	К	У	Р	С	О	В	А	Я	Р	А	Б	О	Т	А
Шифротекст	Ц	П	Ж	Й	Ъ	Ц	Ъ	З	Е	Ф	Ы	Ц	Ж	Ф

Варіант 22. Реалізувати **шифр Белазо** [13, С.87]. Даний спосіб шифрування винайдений італійцем Жованом Белазо. У 1553 р. виходить у світ його книжка "Шифр синьйора Белазо". В цьому шифрі ключем є так званий пароль – фраза або слово, що легко запам'ятовуються. Пароль записується періодично над літерами відкритого тексту. Літера паролю, що стоїть над відповідною літерою відкритого тексту, вказує номер рядка у таблиці Тритемія (див. варіант 20, додаток Е), за якою слід проводити заміну (шифрування) цієї літери (літера відкритого тексту знаходиться у першому рядку таблиці). Наприклад, якщо взяти як пароль слово "ШИФР", то при шифруванні слова "ПРОГРАМА" отримуємо:

Ключове слово	Ш	И	Ф	Р	Ш	И	Ф	Р
Текст повідомлення	П	Р	О	Г	Р	А	М	А
Шифротекст	И	Ш	Г	У	Й	И	Б	Р

Варіант 23. Реалізувати **шифр "Атбаш"**[13, С.142]. В Біблії є натяки на шифрування і дешифрування текстів. Сутність їх полягає у тому, що древні євреї використовували декілька систем шифрування за принципом простої заміни. Шифр «Атбаш» задавався таким чином:

А	Б	В	… … …	Э	Ю	Я
Я	Ю	Э	… … …	В	Б	А

Зауважимо, що в цьому шифрі заміна має симетричний вигляд: (А-Я, Я-А), (Б-Ю, Ю-Б) і т. д. Тому як і при шифруванні, так і при розшифруванні літери відкритого і шифрованого текстів беруться з одного й того самого верхнього рядка.

Варіант 24. Реалізувати **шифр "Альбам"**. Шифр "Альбам" полягає в розбитті алфавіту на дві частини і підписуванні однієї частини під другою:

А Б В Г ... Н О П
Р С Т У ... Є Ю Я

Тут заміна має, як і в шифрі "Атбаш", симетричний характер:

(А-Р, Р-А), (Б-С, С-Б), ..., (П-Я, Я-П)

Як і при шифруванні, так і при розшифруванні літери відкритого і шифрованого текстів беруться з одного й того самого верхнього рядка.

Варіант 25. Реалізувати **книжковий шрифт Енея** [13, С.34-36, С.62]. Існує немало можливостей використовувати книжки для таємного обміну повідомленнями. Наприклад, якщо адресати заздалегідь домовились про використання дублікатів однієї і тієї ж книжки як ключа шифру, то їх таємні послання могли б складатися з таких елементарних одиниць: n/m/t, n – номер сторінки книги, m – номер рядка, t – номер літери в рядку. Так само і читається таємне послання. Ключем такого шифру є книга і використовувана в ній сторінка. Замість книг можуть бути використані окремі файли.

Варіант 26. Реалізувати **омофонну заміну** [13, С.204]**.** Омофонна заміна аналогічна

простій заміні, але має єдину відмінність: кожній букві відкритого тексту ставиться у відповідність декілька символів шифротексту. Наприклад, літера «А» заміняється на цифру 5, 13, 25 або 57; літера «Б» - на 7, 19, 12, 41 і т. д.

Варіант 27. Реалізувати шифрування за допомогою **блочних замін,** в якій шифрування відкритого тексту здійснюється блоками. Наприклад, блоку літер «АБА» може відповідати блок «РТК», а блоку літер «ВАЯ» - блок «АСС» і т. д.

Варіант 28. **Решето Ератосфена.** Розробити програму, яка дозволяє скласти "решето Ератосфена" на будь-якому інтервалі чисел. *([14], С.284-285).*

Варіант 29. **Історія систем числення.** Розробити програму, яка надає історичні відомості про різні системи числення та демонструє їх. *([14], С.261-265).*

Варіант 30. **Подільність чисел.** Розробити програму, яка надає теоретичні відомості про загальну ознаку подільності (ознаку Паскаля) та демонструє визначення подільності чисел на 3, 5, 7, 9 і т.д. *([14], С. 277-280)*.

Варіант 31. **НСК і НСД.** Розробити програму, яка за допомогою алгоритма Евкліда дозволяє визначити НСК і НСД для будь-яких довільних чисел. *([14], С.288-290).*

Варіант 32. **Сортування.** Розробити програму, яка надає теоретичні відомості про основні методи сортування, демонструє їх застосування і наводить основні статистичні дані результатів сортування. *([15], С.438-451).*

Варіант 33. **Календар.** Розробити програму, яка виводить на екран календар на вказаний рік, де користувач зможе вибирати, якою мовою має бути виведений календар.

Варіант 34. **Генератори випадкових чисел.** Розробити програму, яка надає теоретичні відомості про генератори випадкових чисел і демонструє їх роботу (генерує вказану кількість випадкових чисел на вказаному інтервалі).

Варіант 35. **Генератори простих чисел.** Розробити програму, яка надає теоретичні відомості про генератори простих чисел і демонструє їх роботу (генерує вказану кількість простих чисел на вказаному інтервалі).

Варіант 36. Розробити програму для забезпечення лабораторної роботи на тему: «**Графіки тригонометричних функцій**». Передбачити можливість керування параметрами *a, b, A, B* функції: $y = A \cdot f(ax + b) + B$. Як *f(x)* передбачити функції *sin(x), cos(x), tg(x), ctg(x), ln(x), exp(x).*

Варіант 37. **Системи числення.** Розробити програму для реалізації переведення будь-яких чисел з однієї системи числення в іншу. Розглянути десяткову, двійкову, вісімкову та шістнадцяткову системи числення.

Варіант 38. Розробити програму для реалізації **гри «Тетрис»**. Гра повинна бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 39. Розробити програму для реалізації **гри «Пінг-понг»**. Гра повинна бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 40. Розробити програму для реалізації **гри «Змійка»**. Гра повинна бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 41. Розробити програму для реалізації **гри «Lines»**. Гра повинна бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 42. Розробити програму для реалізації **гри «Хрестики-нолики»**. Гра повинна

бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 43. Розробити програму для **демонстрації пор року і часу доби** (ранок, день, вечір, ніч), використовуючи для цього різні ефекти.

Варіант 44. Розробити програму для **демонстрації пішоходного та автомобільного руху** по вулиці, використавши керування світлофором.

Варіант 45. Розробити програму для реалізації **спектру можливих кольорів**, виводячи його або у вигляді кругової, стовпцевої діаграми, або у вигляді клітинок на екрані і т. і.

Варіант 46. Розробити програму для реалізації **гри «Попадання в мішень»**. Гра повинно бути, по можливості, дещо спрощеною, але оригінальною.

Варіант 47. **Конвертор валют.** Розробити програму, яка підраховує для вказаної користувачем суми в одній валюті її еквівалент у іншій валюті.

Варіант 48. **Обмін повідомленнями**. Розробити програму обміну повідомленнями по мережі.

Варіант 49. **Академічна група**. Розробити програму для подання інформації про свою академічну групу.

Варіант 50. **Блокнот студента**. Розробити програму для подання розкладу занять і нагадувань про події на поточний день.

Примітка. Варіанти завдань можуть змінюватись та доповнюватись викладачем.

Додаток Б

Приклад оформлення титульного аркуша

Міністерство освіти і науки України
Вінницький національний технічний університет
Інститут інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

РОЗРОБКА ПРОГРАМИ ДЛЯ РЕАЛІЗАЦІЇ ШИФРУВАННЯ МЕТОДОМ ПРЯМОЇ ПЕРЕСТАНОВКИ

Пояснювальна записка
до курсової роботи з дисципліни "Програмування"
за напрямом підготовки
6.170101 – " Безпека інформаційних і комунікаційних систем "
08-20.ПРГ.110.13.107 ПЗ

Керівник курсової роботи
ст. викл. каф. ЗІ _____ Каплун В. А.

Розробив студент гр. 1БС-07
_____ Холодов І. В.

Курсову роботу захищено
з оцінкою _____

_____ _____ 2010 р.
(Підпис керівника) (Дата)

_____ _____
(Підпис асистента)

Вінниця 2010

Додаток В

Приклад оформлення індивідуального завдання

Вінницький національний технічний університет
Інститут інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д.т.н., проф.
_____В. А. Лужецький
"___" _____ 20__ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу з дисципліни "Програмування"
студенту групи 1БС-07 факультету КСМ Холодову І. В.

Тема: «Розробка програми для реалізації шифрування методом прямої перестановки»

Вимоги до курсової роботи.

1. Дослідити і засвоїти метод шифрування повідомлень.
2. Розробити загальну структурну схему функціонування програми в цілому і алгоритмів шифрування і розшифрування.
3. Підібрати програмні засоби для реалізації розробленого алгоритму, підготувати складові користувацького інтерфейсу.
4. Програмно реалізувати поставлену задачу і підготувати детальний опис послідовності розробки програмного засобу.
5. Протестувати програму на предмет ефективності і правильності її роботи і розробити рекомендації для роботи з нею.

Вихідні дані: об'єкт шифрування – повідомлення з клавіатури або текстові файли; метод шифрування – пряма перестановка символів; мова програмування – C++; програмні засоби – API-функції; програмне середовище – Visual C++; інтерфейс – віконно-графічний; операційна система – сімейство Windows.

Дата видачі _____ 20__ р.

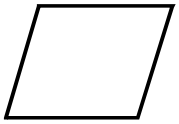
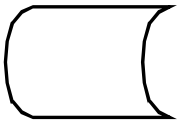
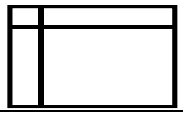
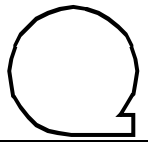
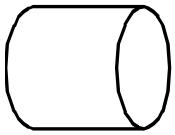
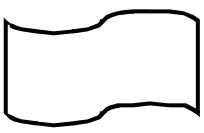
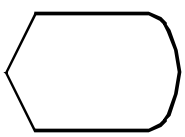
Керівник _____ Каплун В. А.

Завдання отримав _____ Холодов І. В.

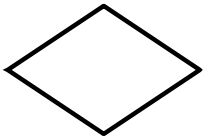
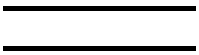
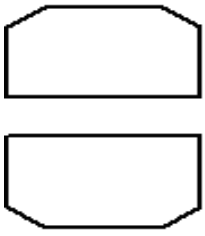
Додаток Г

Символи даних, процесів і ліній

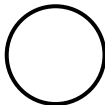
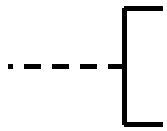
Таблиця Г.1 – Символи даних

Основні символи даних		
Дані		Символ відображає дані, носій даних невизначений
Дані, що запам'ятовуються		Символ відображає дані, що зберігаються, у вигляді, придатному для обробки, носій даних невизначений
Специфічні символи даних		
Оперативний запам'ятовувальний пристрій		Символ відображає дані, що зберігаються в оперативному запам'ятовувальному пристрої
Запам'ятовувальний пристрій з послідовним доступом		Символ відображає дані, що зберігаються в запам'ятовувальному пристрої з послідовним доступом (магнітна стрічка, касета з магнітною стрічкою, магнітофонна касета)
Запам'ятовувальний пристрій з прямим доступом		Символ відображає дані, що зберігаються в запам'ятовувальному пристрої з прямим доступом (магнітний диск, магнітний барабан, гнучкий магнітний диск)
Документ		Символ відображає дані, подані на носії в легкій для читання формі (документ для оптичного або магнітного зчитування, мікрофільм, рулон стрічки з підсумковими даними, бланки введення даних)
Ручне введення		Символ відображає дані, що вводяться вручну під час оброблення з пристроїв будь-якого типу (клавіатура, перемикачі, кнопки, світлове перо, смужки зі штриховим кодом)
Карта		Символ відображає дані, подані на носії у вигляді карти (перфокарти, магнітні карти, карти зі зчитуваними мітками, карти з відривним ярликом, карти зі сканованими мітками).
Паперова стрічка		Символ відображає дані, подані на носії у вигляді паперової стрічки
Дисплей		Символ відображає дані, подані у візуальній людиночитабельній формі на носії у вигляді пристрою відображення (екран для візуального спостереження, індикатори введення інформації)

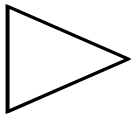
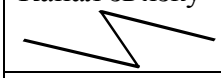
Таблиця Г. 2 – Символи процесу

Основні символи, процесу		
Процес		Символ відображає функцію обробки даних будь-якого вигляду (виконання певної операції або їх групи, що приводить до зміни значення, форми інформації).
Специфічні символи процесу		
Підпорядкований процес		Символ відображає підпорядкований процес, що складається з однієї або декількох операцій або кроків програми, які визначені у іншому місці
Ручна операція		Символ відображає будь-який процес, виконуваний людиною
Підготовка		Символ відображає модифікацію команди або групи команд з метою дії на деяку подальшу функцію (установлення перемикача, модифікація індексного регістра або ініціалізація програми)
Умова або вибір		Символ відображає умову, вибір або функцію типу перемикача, що має один вхід і ряд альтернативних виходів, один і лише один з яких може бути активізований після обчислення умов, визначених усередині цього символу
Паралельні дії		Символ відображає синхронізацію двох або більше паралельних операцій
Межа циклу		Символ, що складається з двох частин, відображає початок і кінець циклу. Обидві частини символу мають один і той самий ідентифікатор. Умови для ініціалізації, прирости, завершення поміщаються усередині символу на початку або в кінці залежно від розташування операції, що перевіряє умову

Таблиця Г. 3 – Спеціальні символи

З'єднувач 	Символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми і використовується для обривання лінії і продовження її у іншому місці. Відповідні символи-з'єднувачі повинні містити одне і те ж унікальне позначення
Термінатор 	Символ відображає вихід в зовнішнє середовище і вхід із зовнішнього середовища (початок або кінець схеми програми, зовнішнє використовування і джерело або пункт призначення даних)
Коментар 	Символ використовують для додавання описових коментарів або записів пояснень з метою пояснення або приміток. Пунктирні лінії в символі коментаря пов'язані з відповідним символом або можуть окреслювати групу символів. Текст коментарів або приміток повинен бути поміщений біля обмежуючої фігури
Пропуск 	Символ (три крапки) використовують в схемах для відображення пропуску символу або групи символів, в яких не визначені ні тип, ні число символів. Символ використовують тільки в символах лінії або між ними. Він застосовується головним чином в схемах, що зображають загальні результати вибору з невідомим числом повторень

Таблиця Г. 4 – Символи ліній

Основний символ ліній	
Лінія	Символ відображає потік даних або управління. У разі необхідності або для підвищення легкості читання можуть бути додані стрілки-покажчики
Специфічні символи ліній	
Передача управління 	Символ відображає безпосередню передачу управління від одного процесу до іншого, іноді з можливістю прямого повернення до ініціювального процесу після того, як ініційований процес завершить свої функції. Тип передачі управління повинен бути названий усередині символу (наприклад, запит, виклик, подія)
Канал зв'язку 	Символ відображає передачу даних по каналу зв'язку
Пунктирна лінія	Символ відображає альтернативний зв'язок між двома або більшою кількістю символів, а також використовується для обведення ділянки

Додаток Д

Шифрувальна таблиця Віжинера

Ключ	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	Ь	Ы	Ъ	Ю	Я
0	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	Ь	Ы	Ъ	Ю	Я
1	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	Ь	Ы	Ъ	Ю	Я	А
2	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б
3	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В
4	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г
5	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д
6	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е
7	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж
8	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
9	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
10	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
11	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
12	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
13	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
14	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
15	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
16	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
17	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
18	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
19	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
20	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
21	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
22	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
23	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
24	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
25	Щ	Ь	Ы	Ъ	Ю	Я	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
26	Ь	Ы	Ъ	Ю	Я	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
27	Ы	Ъ	Ю	Я	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	Ь
28	Ъ	Ю	Я	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	Ь	Ы
29	Ю	Я	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	Ь	Ы	Ъ
30	Я	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	Ь	Ы	Ъ	Ю

Додаток Е

Шифрувальна таблиця Тритемія

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	Ь	Ы	Ъ	Ю	Я
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	Ь	Ы	Ъ	Ю	Я	А
В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б
Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В
Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г
Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д
Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е
З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж
И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Щ	Ь	Ы	Ъ	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Ь	Ы	Ъ	Ю	Я	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
Ь	Ы	Ъ	Ю	Я	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	Ъ	Ю	Я	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	Ь
Ъ	Ю	Я	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	Ь	Ы
Ю	Я	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	Ь	Ы	Ъ
Я	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	Ь	Ы	Ъ	Ю

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсової роботи з дисципліни
"ПРОГРАМУВАННЯ"
для студентів напряму підготовки 6.170101
“Безпека інформаційних і комунікаційних систем”

Редактор В. Дружиніна
Коректор З. Поліщук

Укладачі: Валентина Аполлінаріївна Каплун
Олеся Петрівна Войтович

Оригінал-макет підготовлено В. Каплун

Підписано до друку
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк.
Наклад прим. Зам. №

Вінницький національний технічний університет,
науково-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб’єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп’ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-81-59.
Свідоцтво суб’єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.