Зміст

АНОТАЦІЯ

Описано об’єктно-орієнтований підхід розробки програмних продуктів мовою програмування С++ й застосовано уніфіковану мову моделювання (UML) для графічного відображення програми. Результатом роботи є програма для шифрування і дешифрування тексту на основі алгоритму RSA.

ВСТУП

Програмування - це створення програмних продуктів, які написані на мові програмування. Мова програмування - це формальна знакова система, яка призначена для написання програм і зрозумілою для компютера. З розвитком програмування виникла ідея поєднати в межах однієї сутності дані і код, що безпосередньо опрацьовує ці дані. Така сутність отримала назву об’єкт, а відповідний підхід до створення програм називають об’єктно-орієнтованим програмуванням.

Об’єктно-орієнтоване програмування (ООП) – це парадигма програмування, яка розглядає програму як сукупність гнучко пов’язаних між собою об’єктів.

Основні переваги концепції ООП:

- можливість створювати користувацькі типи даних (класи);
- приховування деталей реалізації (інкапсуляція);
- можливість повторного використання коду (наслідування);
- інтерпретація викликів процедур та функцій на етапі виконання (поліморфізм).

Клас – це спеціальна конструкція мови програмування, що використовується для групування пов'язаних змінних та функцій.

Інкапсулювання – це механізм в програмуванні, який пов’язує в одне ціле функції і дані, якими вони маніпулють, а також захищає їх від зовнішнього доступу і неправильного застосування.

Успадкування – це властивість, з допомогою якої один об’єкт може набувати властивостей іншого. При цьому підтримується концепція ієрархічної класифікації.

Поліморфізм дозволяє писати більш абстрактні програми і підвищити коефіцієнт повторного використання коду. Разом з інкапсуляцією і успадкуванням поліморфізм також являє собою одну із важливих концепцій ООП. Застосування цієї концепції дозволяє значно полегшити розробку складних програм.

Метою даної роботи є розробка програми для шифрування текстових повідомлень за допомогою алгоритму RSA. Шифрування – це процес застосування шифру до захищеної інформації, перетворення захищеної інформації в шифроване повідомлення за допомогою певних правил, що містяться в шифрі[3,8]. Шифрування поділяється на симетричне і асиметричне. В симетричному шифруванні один і той самий ключ (що зберігається в секреті) використовується як для шифрування, так і для розшифрування. Асиметричні криптосистеми — ефективні системи криптографічного захисту даних, які також називають криптосистемами з відкритим ключем. В таких системах для зашифровування даних використовується один ключ, а для розшифровування — інший ключ (звідси і назва — асиметричні). Перший ключ є відкритим і може бути опублікованим для використання усіма користувачами системи, які шифрують дані. Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних отримувач зашифрованої інформації використовує другий ключ, який є секретним. Зрозуміло, що ключ розшифровування не може бути визначеним з ключа зашифровування.

Дешифрування(розшифрування) – це процес, оберненого шифрування, перетворення шифрованого повідомлення в зашищену інформацію за допомогою певних правил, що містяться в шифрі[3,8]. В асиметричному шифруванні є два пов'язаних ключа – пара ключів.

Найбільш широко використовуваною і перевіреної криптосистемою з відкритим ключем, яка була придумана Рівестом, Шамір і Ед-Леманом (Rivest, Shamir, Adleman), є система RSA. Вона основана на дивно простий теоретко-числовий (можна сказати, навіть арифметичної) ідеї і ще в змозі чинити опір всім криптоаналітіческім атакам. Ідея полягає в майстерному використанні того факту, що легко перемножіть два великих простих числа, однако вкрай важко розкласти на множники їх произведений. Таким чином, добуток може бути відкрито і використано як ключ зашифрування. Вихідні прості числа не можуть бути відновновлено з їх добутків. З іншого боку, ці прості складі не-

обхідні для дешифруванні. Отже, ми маємо чудовий каркас для криптосистеми з відкритим ключем.

Тепер розглянемо алгоритм RSA більш докладно. Нехай p і q - два різних великих випадково вибраних простих числа (що мають зазвичай 100 розрядів в їх десятковому поданні). Позначимо

$$n = pq \text{ і } \varphi(n) = (p - 1)(q - 1),$$

де φ - функція Ейлера. Випадково виберемо велике число $d > 1$, таке, що $(d, \varphi(n)) = 1$, і обчислимо e, $1 < e < \varphi(n)$, що задовольнить порівнянні

$$ed \equiv 1 \pmod{\varphi(n)}.$$

Числа n, e і d називаються *модулем*, *експонентою зашифрування* і *дешифрування* відповідно. Числа n і e утворюють *відкритий ключ зашифрування*, тоді як решта числа p, q, $\varphi(n)$ і d формують *закритий ключ*. Очевидно, що він включає в себе взаємозалежні величини. Наприклад, знаючи p, неважко обчислити решту три величини.

При *шифруванні* вихідний текст зводиться до степеня e по модулю n. При *дешифруванні* криптотекст зводиться до степеня d по модулю n [7,4].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Розробляємо програму для кодування тексту. Користувач має можливість зашифрувати і розшифрувати текст: ввівши його з клавіатури, при цьому вказавши ключі для шифрування або ж згенерувати їх програмно, зчитати з файлів текст і ключі чи зашифрувати дані тимчасово збережені у програмі. Аналогічні дії можливі для дешифрування тексту.

Додатковими можливостями програми є зберегти і зчитати дані з файлів.

На базі ТЗ було побудовано діаграму прецедентів, яка відображає вимоги замовника до системних [1,129]. (на рис. 1.1).

Рисунок 1.1. Діаграма прецедентів

2 ОБГРУНТУВАННЯ АЛГОРИТМУ І СТРУКТУРИ ПРОГРАМИ

В програмі створено декілька класів. Основний з них *RSA* – клас, який реалізує шифрування й дешифрування. Він утворений наслідування від класу *asymmetric_encryption*, який в свою чергу унаслідується від *encryption*.

Інші класи реалізують додаткові функції при роботі програми. Клас *Text*– реалізує збереження даних у зашифрованому і дешифрованому вигляді, і ключі шифрування. Клас *MyFile* – реалізує роботу з файлами : збереження даних і ключів, і їх завантаження.

RSA — криптографічна система з відкритим ключем. RSA став першим алгоритмом такого типу, придатним і для шифрування і для цифрового підпису. Алгоритм використовується у великій кількості криптографічних застосунків. Безпека RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

3 РОЗРОБКА ПРОГРАМИ

Програма шифрування реалізована за допомогою об’єктно-орієнтованого підходу на мові C++ і на основі декількох класів зображених на діаграмі класів (на рис. 3.1).

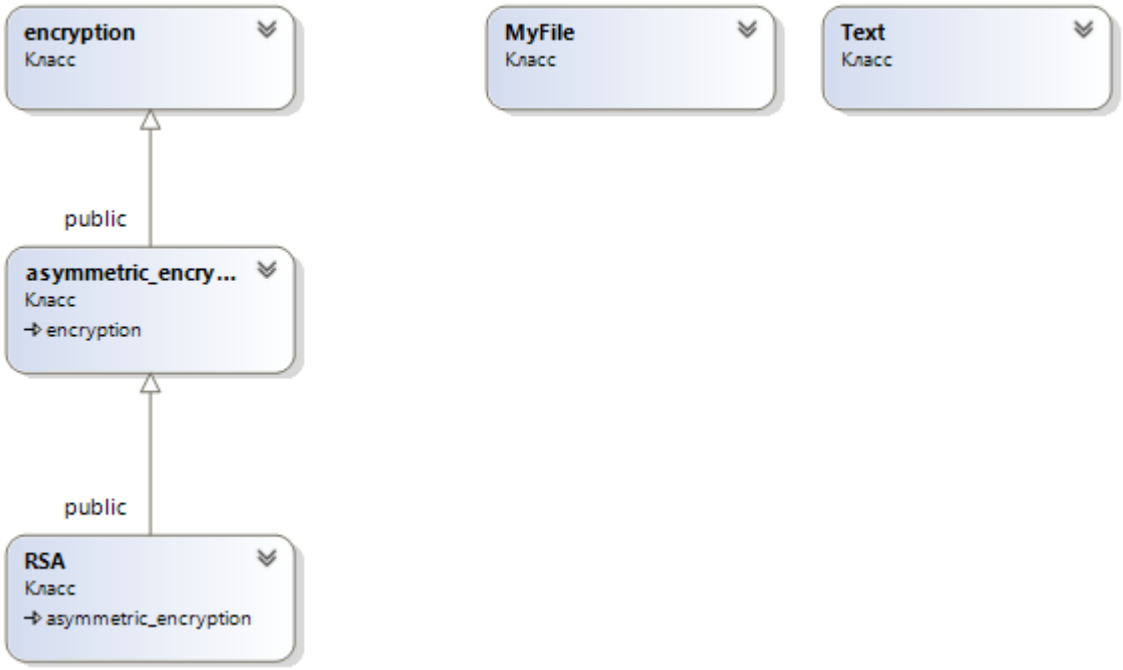

Рисунок 3.1. Діаграма класів

Основний алгоритм — шифрування — реалізований у класів RSA (рис. 3.4), який створений на основі класу asymmetric_encryption (рис. 3.3), який у свою чергу — encryption (рис. 3.4), за допомогою наслідування.

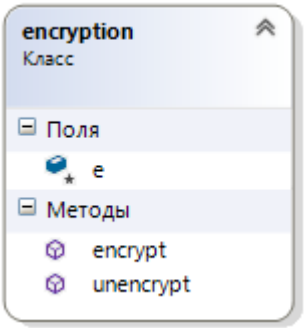

Рисунок 3.2. Клас *encryption*

Клас *encryption* містить поле *e* — ключ шифрування, і 2 методи *encrypt* і *unencrypt* для *шифрування* і *дешифрування* даних.

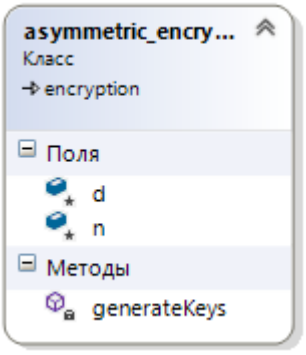

Рисунок 3.3. Клас *asymmetric_encryption*

Клас *asymmetric_encryption* – клас асиметричного шифрування, наслідується з *encryption*, при цьому у ньому присутні додаткові поля *d* і *n* – параметри закритого і відкритого ключа, і метод *generateKeys*, який генерує ключі для шифрування.

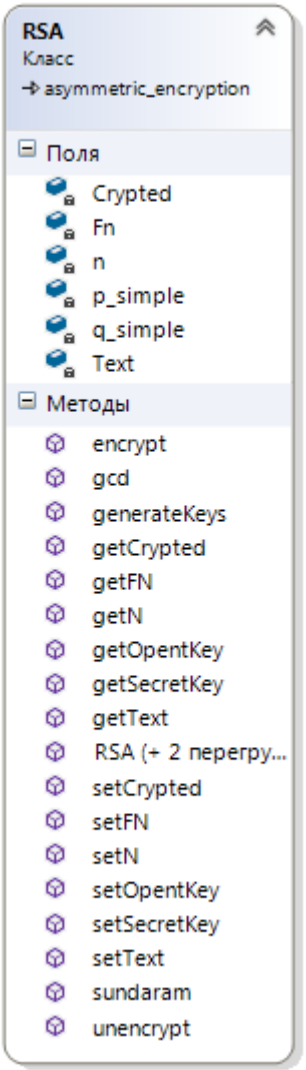

Рисунок 3.5. Клас *RSA*

Клас *RSA* успадкований від класу *asymmetric_encryption* й інкапсулює у собі усі необхідні методи і поля для реалізації алгоритму шифрування RSA :

1) унаслідовані :

- поле *e* – експонента відкритого ключа
- поле *d* – експонента закритого ключа
- метод *enctrypt* – реалізує шифрування
- метод *unencrypt* – реалізує дешифрування
- метод *generateKeys* – реалізує генерування ключів

2) нові :

- поля *p_simple* і *q_simple* – параметри шифрування, частина закритого ключа
- поле *n* – модуль числа, частина відкритого ключа
- поле *Fn* – функція Ейлера, параметр потрібний для генерування експоненти закритого ключа
- поле *Text* – містить дані для шифрування
- поле *Crypted* – містить зашифровані дані
- методи *getCrypted, getFN, getN, getOpenKey, getSecretKey, getText* – асесори доступу до полів, які реалізують можливість отримання даних цих полів
- методи *setCrypted, setFN, setN, setOpenKey, setSecretKey, setText* – асесори доступу до полів, які реалізують можливість запису даних у ці поля
- метод *gcd* – найбільший спільний дільник, метод потрібний для генерування ключів
- метод *sundaram* – алгоритм "Решето Сундарама", реалізує пошук простого числа, метод потрібний для генерації ключів

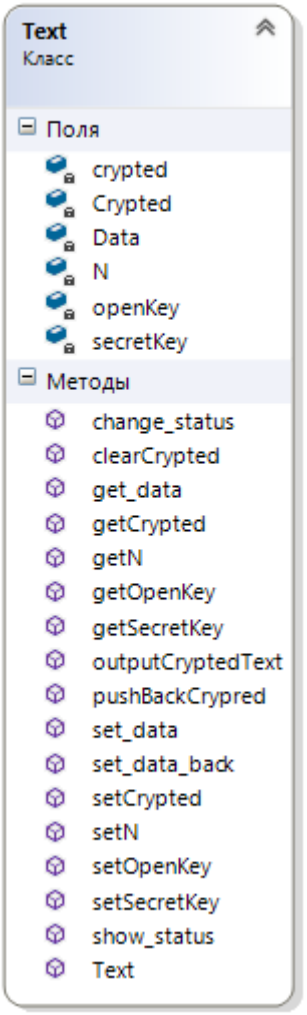

Рисунок 3.6. Клас Text

Клас Text (рис. 3.6) реалізує роботу із даними як зашифрованому вигляді так і у звичайному: зберігання даних у двох видах і зберігання параметрів шифрування – відкритого і закритого ключів. Цей клас інкапсулює такі поля і методи як :

- поле *crypted* – поле-індикатор, вказує чи є дані зашифрованими чи у дешифрованому вигляді
- поле *Crypted* – містить зашифрований текст у вигляді цілих чисел
- поле *Data* – містить дані у незашифрованому вигляді
- поле *N* – модуль, параметр відкритого ключа
- поле o*penKey* – експонента відкритого числа
- поле *secretKey* – експонента закритого числа
- методи *getCrypted*,*getN*, *getOpenKey*, *getSecretKey*, *get_data*, *show_status*, *outputCryptedText* – методи-аксесори доступу до полів, за допомогою яких реалізується отримання даних із цих полів
- методи *set_data*, *set_data_back*, *setCrypted*, *setN*, *setOpenKey*, *setSecretKey*, *change_status*, *pushBackCrypted* – методи-

аксесори доступу до полів, що реалізують запис даних у ці поля

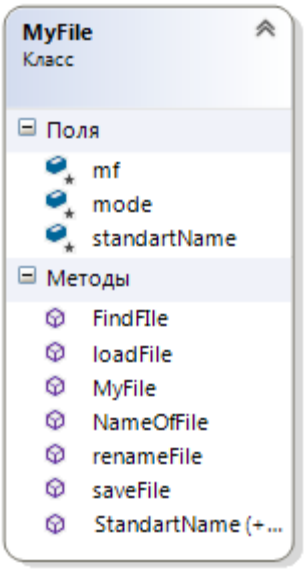

Рисунок 3.7. Клас *MyFile*

Клас *MyFile* (рис. 3.7) реалізує роботу із файлами. Він інкапсулює у собі такі поля і методи як :

- поле *mf* – вказівник на файл
- поле *mode* – режим із яким буде відкрито файл
- поле *standartName* – ім’я файлу
- методи *StandartName* – метод-аксесор доступу до поля, що містить ім’я
- метод *renameFile* – метод для перейменування файлу
- матод *FindFile* – метод, що реалізує перевірку чи існує файл із вказаним ім’я у директорії із виконуваним файлом
- метод *saveFile* – метод, що реалізує збереження даних у файл
- метод *loadFile* – метод, що реалізує завантаження даних із файлу

4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

При запуску програми відображається основне вікно меню зображено на рис. 4.1.

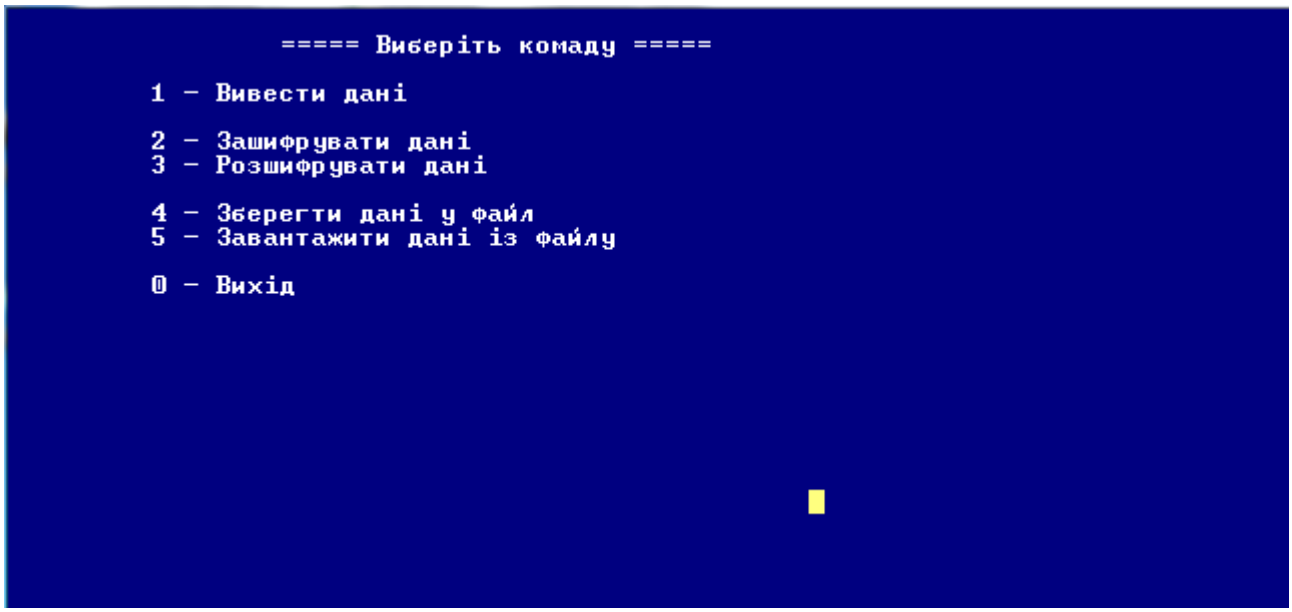

Рисунок 4.1. Меню програми

Нажимання кнопок «1» виводиться інформацію про введені дані : текст і, якщо текст зашифрований . Якщо у програму нічого не введено, вікно буде таким, як зображено на рис.4.2.

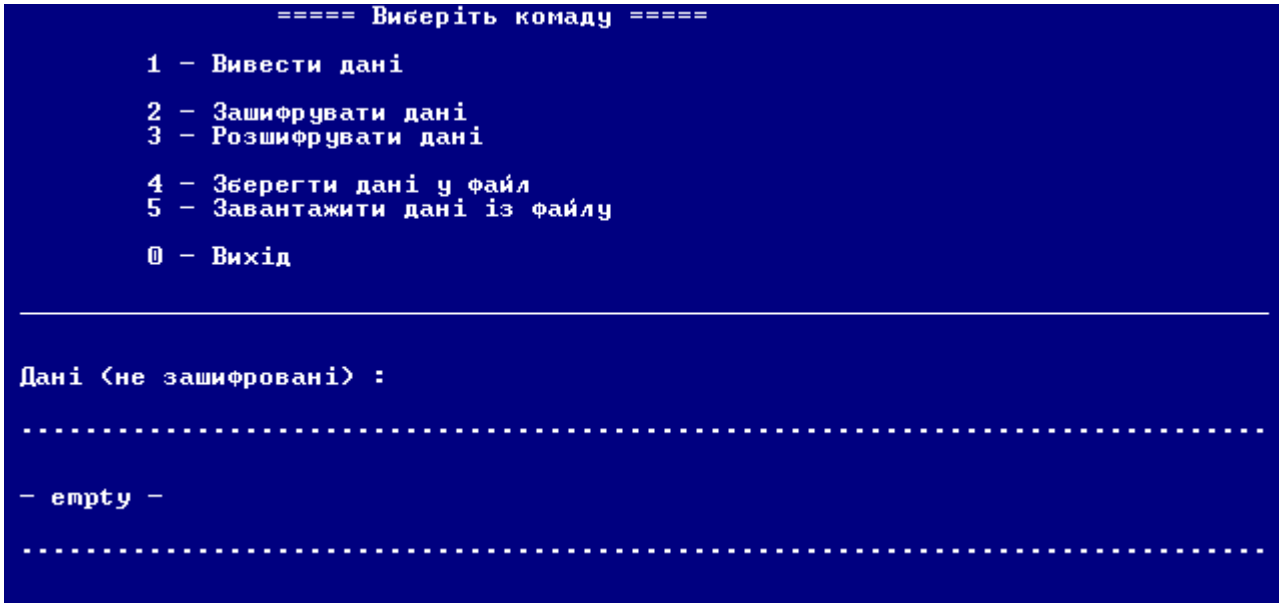

Рисунок 4.2. Вікно виводу введених даних

Нажимаючи «2» у головному вікні програми виводиться підменю *шифрування* (рис. 4.3), де пропонується три метода вводу тексту : використати дані, що уже є у програмі , ввести з клавіатури дані і ключі, зчитати дані з файликів – дані будуть завантажені із файлу «temp.dat», ключі із «settings.key», які знаходяться у тій же директорії що і виконуваний файл. Формат файлу «temp.dat» – символ, що вказує що чи є дані зашифрованими чи ні : «0» – дані у звичайному вигляді, «1» – дані завантажені і зображені у вигляді цілих чисел записаних через пробіли. Формат файлу «settings.key» – 3 числа, перше експонента відкритого ключа, друге – експонента закритого ключа, третє – модуль.

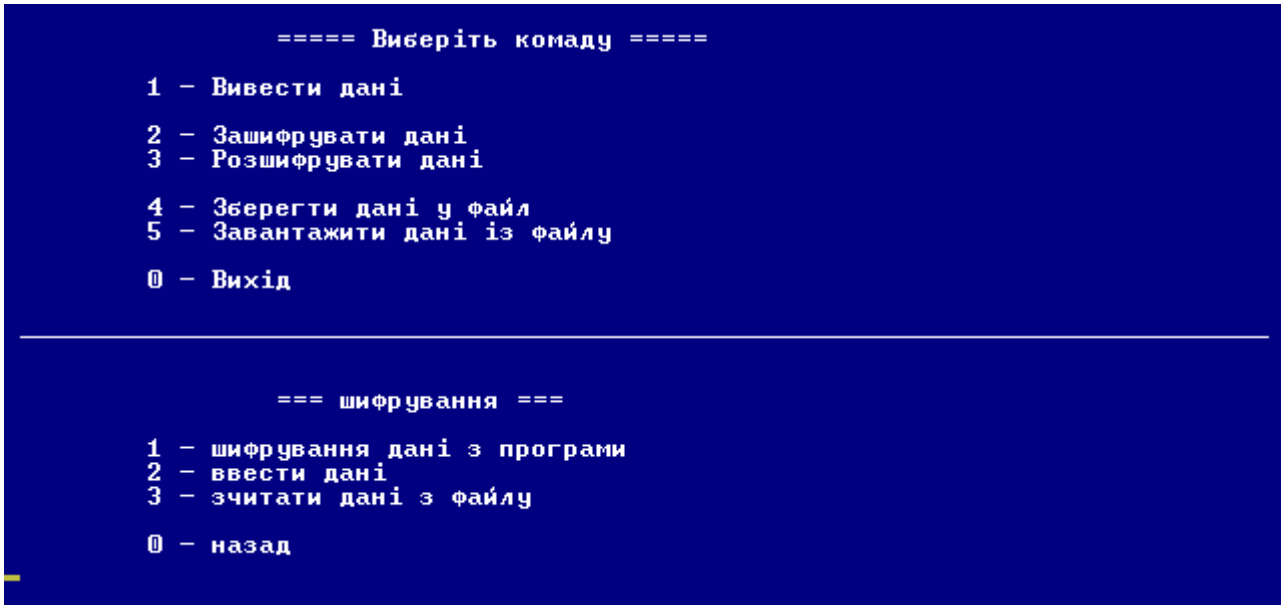

Рисунок 4.3. Меню вибору введення даних

При виборі «2», програма приймає дані для шифрування і пропонує ввести ключі, в іншому ж випадку генерує їх самостійно (рис. 4.4).

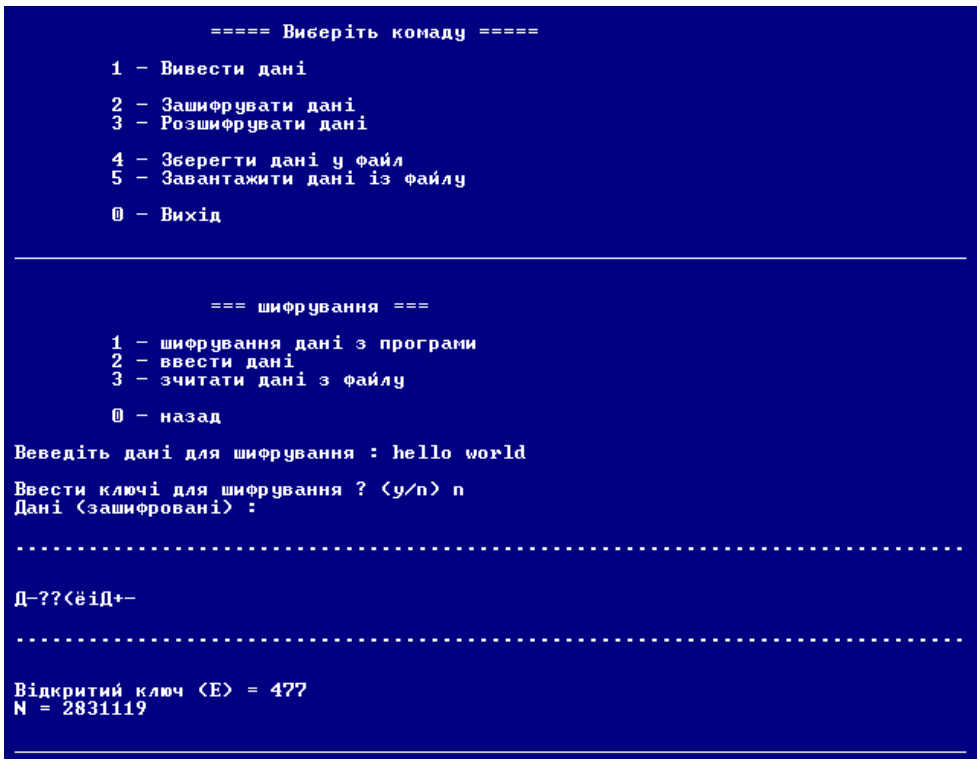

Рисунок 4.4. Введені дані зашифровано

Натиснувши «3» у головному меню програми, програма виводить підменю *дешифрування* аналогічне підменю шифруванню, де можна вибрати протилежні дії до вище згаданих : дешифрування і вивід результату (рис. 4.5).

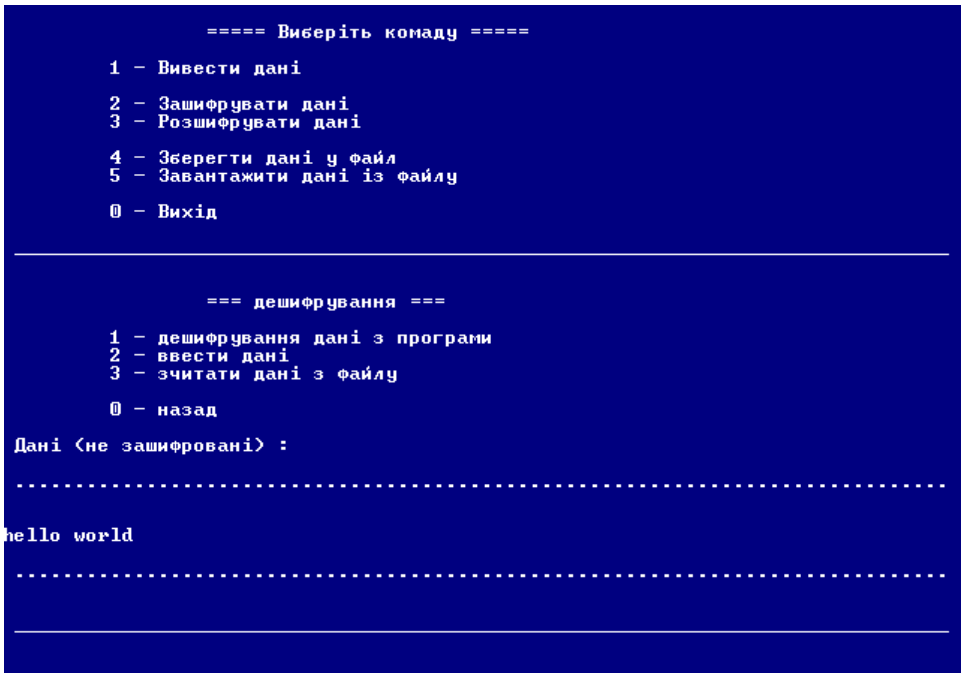

Рисунок 4.5. Дешифрування даних і вивід результату

Також в головному меню можна завантажити дані із файлів, аналогічно вище згаданому способу завантаженні файлу при шифруванні, і зберегти дані із ключами у файли, у вище згаданому форматі. Також програма робить автоматичне збереження даних у файли після кожної операції над даними – шифруванні чи дешифруванні.

На рисунку 4.6 зображено вихідний файл «temp.dat» після шифрування фрази «hello world», а на рисунку 4.7 – файл із збереженими ключами.

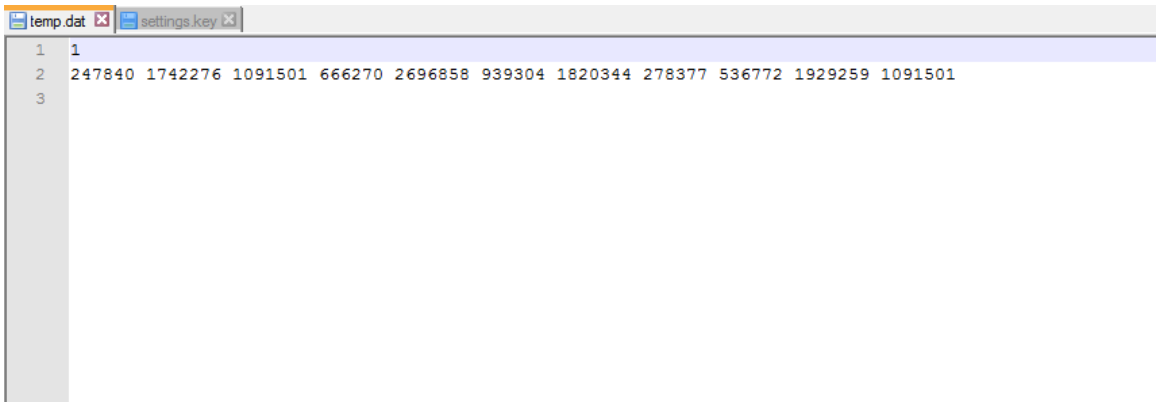

Рисунок 4.6. Файл «temp.dat»

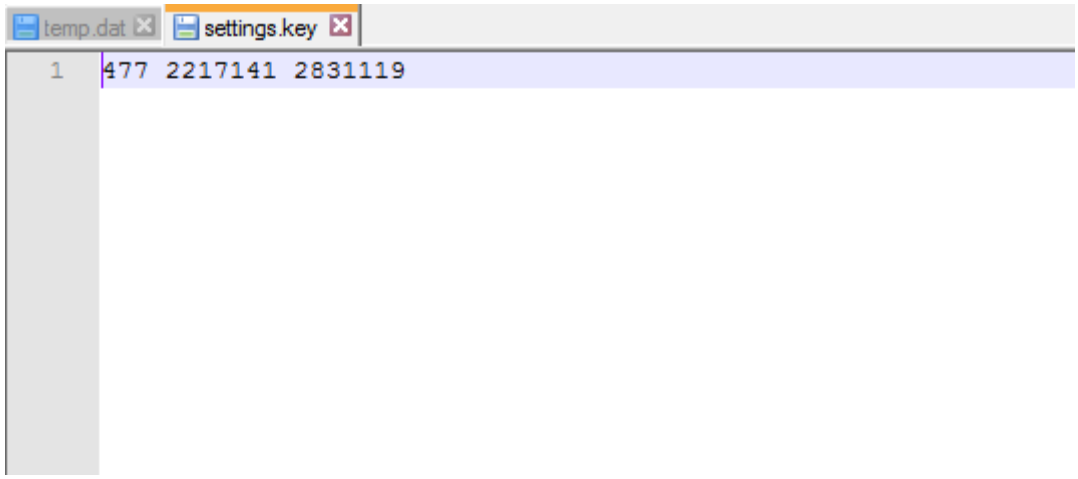

Рисунок 4.7. Файл «settings.key»

ВИСНОВКИ

В результаті виконання роботи було розроблено програму, яка шифрує і дешифрує текстові повідомлення.

Було використано мову програмування C++. Також в результаті роботи було використано ООП, а саме інкапсуляція і наслідування. Забезпечена модульність.

Організовано роботу з файлами: створення файлу, зчитування файлу, вивід даних на екран.

Було розроблено діаграму прецедентів. Після визначення усіх вимог до системи змодельовано діаграму класів і на основі цієї діаграми було розроблено класи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Орлов С. А. Технологии разработки программного обеспечения: Учебник. - СПб.: Питер, 2002. – 464 с. ISBN: 5-94723-145-X(рус.).

2. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд. / Стивен Прата : Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2012. – 1248 с. ISBN 978-5-8459-1778-2 (рус.).

3. В.В.Ященко - Введение в Криптографию. /:Издательство ЧеРо, 1999. – 271 с. ISBN: 5-900916-40-5 (рус.).

4. Страуструп Б. Язык программирования С++: Специальное издание. / Бьерн Страуструп. Пер. с англ. – М.: Издательство Бином, 2011. – 1136 с. ISBN 978-5-7989-0425-9 (рус.).

5. Дейтел Х.М. Как программировать на С++: 5-е издание. / Х.М. Дейтел, П. Дж. Дейтел : Пер. с англ. – М.: ООО «Бином-Пресс», 2008. – 1456 с. ISBN 978-5-9518-0224-8 (рус.).

6. Р. ЛАФОРЕ «Объеткно-ориентированное программирование в С++» 4-е издание.

7. А. Саломаа «Криптография с открытым ключом». / И.А. Вихлянцева, А.Е. Андреева и А.А. Болотова. Перевод с англ. - Москва ”Мир” 1995

ДОДАТОК А

Лістинг «encryption.h»

```
#ifndef encryption_h
#endif encryption_h

#include <vector>
#include <string>

class encryption
{
protected:
	unsigned long e; // key
public:
	virtual std::vector<unsigned long> encrypt() =0;
	virtual std::string unencrypt() = 0;
};
```

Лістинг «asymmetric_encryption.h»

```
#ifndef asymmetric_h
#endif asymmetric_h

#include "encryption.h"

class asymmetric_encryption : public encryption
{
protected:
	unsigned long d; // secret key
	unsigned long n;
private:
	virtual void generateKeys(unsigned long, unsigned long) = 0;
   };
```

Літсинг «RSA.h»

```
#ifndef RSA_H
#define RSA_H

#include <iostream>
#include <string>
#include <cstdlib> // Для rand()
#include <time.h> // Для time
#include <cmath>
#include <vector>

#include "asymmetric_encryption.h"

using namespace std;

class RSA : public asymmetric_encryption
{
private:
	string Text;
	vector<unsigned long> Crypted;
	unsigned long p_simple;
	unsigned long q_simple;
	unsigned long n;
	unsigned long Fn;
public:
	RSA(string _data = "")
	{
		Text = _data;
```

```
			generateKeys();
		}
		RSA(vector<unsigned long> _crypted, unsigned long openK = 0, unsigned long secretK =
0, unsigned long _N = 0 , unsigned long _Fn = 0)
		{
			Crypted.resize( _crypted.size() );
			Crypted = _crypted;
			e = openK;
			d = secretK;
			n = _N;
			Fn = _Fn;
		}
		RSA(string _data,  unsigned long ok,  unsigned long sk,  unsigned long N)
		{
			Text = _data;
			d =  sk;
			e = ok;
			n = N;
		}
		// Методи-аксесори {
		void setSecretKey(unsigned long  sk)
		{
			d = sk;
		}
		unsigned long  getSecretKey()
		{
			return d;
		}
		void setOpentKey(unsigned long  ok)
		{
			e = ok;
		}
		unsigned long  getOpentKey()
		{
			return e;
		}
		unsigned long getN()
		{
			return n;
		}
		void setFN( unsigned long fn)
		{
			Fn = fn;
		}
		void setN( unsigned long _n)
		{
			n = _n;
		}
		 unsigned long getFN()
		{
			return Fn;
		}
		string getText()
		{
			return Text;
		}
		void setText(string _text)
		{
			Text = _text;
		}
		vector<unsigned long> getCrypted()
		{
			return Crypted;
		}
		void setCrypted(vector<unsigned long> crypted)
```

```
{
	Crypted = crypted;
}
// } Методи-аксесори

// Методи для генерації ключів
unsigned long sundaram(unsigned long n) // Алгоритм "Решето Сундарама"
{
	unsigned long *a = new unsigned long [n], i, j, k;
	memset(a, 0, sizeof(unsigned long) * n);
	for(i = 1; 3*i+1 < n; i++)
	{
		for(j = 1; (k = i+j+2*i*j) < n && j <= i; j++)
			a[k] = 1;
	}
	// Повертання найближчого числа до заданого
	for(i = n-1; i >= 1; i--)
		if(a[i] == 0)
		{
			return (2 * i + 1);
			break;
		}
		delete [] a;
}
void generateKeys(unsigned long p = 0, unsigned long q = 0)
{
	srand( (unsigned)time( NULL ) );
	if (!p && !q)
	{
		p = rand()%1000;
		q = rand()%1000;
	}
	p_simple = sundaram(p);
	q_simple = sundaram(q);
	//Находимо число n.
	n = p_simple*q_simple;
	Fn = ((p_simple-1)*(q_simple-1));
	////Генерація числа d і провірка його на взаємопростоту
	////з числом Fn.
	unsigned long e_simple = 0;
	while (e_simple !=1)
	{
		e = rand()%1000;
		e_simple = gcd (e, ((p_simple-1)*(q_simple-1)));
	}
	//Визначення числа e, для якого є істинним
	//відношення (e*d)%Fn=1.
	unsigned long d_simple = 0;
	d = 0;
	while (d_simple !=1)
	{
		d += 1;
		d_simple = (e*d)%((p_simple-1)*(q_simple-1));
	}
}
unsigned long gcd(unsigned long a, unsigned long b) // Алгоритм Евклида
{
	unsigned long c;
	while (b)
	{
		c = a % b;
		a = b;
		b = c;
	}
	return a;
```

```
	}
	// -------------------------

	// Методи шифрування - дишифрування
	vector<unsigned long> encrypt() // Метод шифрування
	{
		if (n == 0 || e == 0 || d == 0)
			generateKeys();

		Text.c_str();
		Crypted.resize( Text.length());
		unsigned long long c;
		unsigned long b =301;
		for (unsigned long j = 0; j < Text.length(); j++)
		{
			c = 1;
			unsigned long i = 0;
			unsigned long ASCIIcode = (static_cast<unsigned long>(Text[j]))+b;
			while (i<e)
			{
				c = c*ASCIIcode;
				c = c%n;
				i++;
			}
			Crypted[j] = c;
			b+=1;
		}
		return Crypted;
	}
	string unencrypt() // Метод для розшифрування
	{
		unsigned long b = 301;
		unsigned long long m ;
		for(unsigned long j = 0; j < Crypted.size(); j++)
		{
			m=1;
			unsigned long i = 0;
			while (i<d)
			{
				m = m * Crypted[j];
				m = m%n;
				i++;
			}
			m -= b;
			b+=1;
			Text.push_back(static_cast<char>(m));
		}
		return Text;
	}
	//---------------------------------

};
  #endif
```

Лістинг «text.h»

```
#ifndef TEXT_H
#define TEXT_H

#include <iostream>
#include <string>
#include <vector>

using namespace std;
```

```
// Клас тексту з розширеними властивостями
class Text
{
private:
	string Data;
	vector<unsigned long> Crypted;
	unsigned long openKey;
	unsigned long secretKey;
	unsigned long N;
	bool crypted;
public:
	Text( string new_data = "", unsigned long new_OKey = 0, unsigned long new_SKey = 0,
bool new_h = 0)
	{
		openKey = new_OKey;
		secretKey = new_SKey;
		Data = new_data;
		crypted = new_h;
		Crypted.resize( Data.length());
		N = 0;
	}
	void setCrypted(vector<unsigned long> crypted)
	{
		Crypted = crypted;
	}
	void pushBackCrypred(unsigned long temp)
	{
		Crypted.push_back(temp);
	}
	unsigned long clearCrypted()
	{
		Crypted.clear();
		return 1;
	}
	vector<unsigned long> getCrypted()
	{
		return Crypted;
	}
	void outputCryptedText()
	{
		for(unsigned long i=0; i<Crypted.size(); i++)
			cout << (char)Crypted[i];
	}
	void setN( unsigned long n)
	{
		N = n;
	}
	 unsigned long getN()
	{
		return N;
	}
	bool show_status()
	{
		if (crypted) return true;
		else return false;
	}
	void change_status()
	{
		if (crypted) crypted = 0;
		else crypted = 1;
	}
	void set_data( string new_data )
	{
```

```
		Data = new_data;
	}
	void set_data_back(char _data)
	{
		Data.push_back(_data);
	}
	void setOpenKey(  unsigned long new_key )
	{
		openKey = new_key;
	}
	void setSecretKey( unsigned long new_key)
	{
		secretKey = new_key;
	}
	 unsigned long getOpenKey()
	{
		return openKey;
	}
	 unsigned long getSecretKey()
	{
		return secretKey;
	}
	string get_data()
	{
		return Data;
	}
};

#endif
```

Лістинг «Filefunct.h»

```
#ifndef _FileFunct_H_
#define _FileFinct_H_

#include <string>
#include <cstdio>

using namespace std;

class MyFile
{
protected:
	FILE *mf;
	string standartName; // Ім'я файла
	string mode;		// Режим відкриття файла
public:
	MyFile(string Name = "temp.dat", string Mode = "rt+")
	{
		standartName = Name;
		mode = Mode;
	}
	bool FindFIle(string name = "") // Медот пошуку файла: якщо файл із вказаним ім'ям
існує то він поверає true (1), якщо ні - false (0)
	{
		if( name == "" ) name = standartName;

		if ((mf = fopen( name.c_str() , "rt" )) != NULL)
		{
			fclose(mf);
			return true;
		}
		else
		{
			return false;
```

```
	}
}
string StandartName() const { return standartName; }
void StandartName(string val) { standartName = val; }
string NameOfFile() // Метод для зміни імені файлу
{
	setlocale(LC_ALL, "rus");

	cout << "\n Введіть назву файлу ( назва.розширення )"
		<< "\n (Нажміть Enter для вводу імені \""
		<< standartName << "\" ) : \n\t";

	string _NameOfFile("");
	getline(cin, _NameOfFile);

	// Якщо нічого не було введено то назва файлу залишається попередньою
	// (стандартною, заданою при ініціалізації об'єкта)
	if (_NameOfFile == "")  _NameOfFile = standartName;
	// ----------------------------------------------------------------


	// Перевірки назви файлу на пробіли і заміна їх на символ '_'
	_NameOfFile.c_str();
	unsigned long i;
	for (i=0; i<_NameOfFile.length(); i++)
	{
		if (_NameOfFile[i] == ' ' ) _NameOfFile[i] = '_';
	}
	// ------------------------------------------------------

	return _NameOfFile;
}
unsigned long loadFile(Text& text, string temp = "")
{
	setlocale(LC_ALL, "rus");

	if (temp == "settings")
	{
		if(FindFIle())
		{
			SetFileAttributes( TEXT("settings.key") ,FILE_ATTRIBUTE_NORMAL);
   // Функція змінює атрибути файлу, забирає атрибут 'hide'

			mf = fopen(standartName.c_str(),"rt");
			unsigned long temp;
			fscanf(mf,"%lu",&temp);
			text.setOpenKey(temp);
			fscanf(mf,"%lu",&temp);
			text.setSecretKey(temp);
			fscanf(mf,"%lu",&temp);
			text.setN(temp);

			SetFileAttributes( TEXT("settings.key") ,FILE_ATTRIBUTE_HIDDEN);
   // Функція змінює атрибути файлу на скритий (сховує файл із ключами)

			return 0;
		}
		else
			return -1;

	}
	else
	{
		while (!FindFIle())
		{
```

```
                        cout << "\n\n Файл \"" << standartName <<"\" не знайдено! \n";
                        renameFile();
                        cout << "\n\t Продовжити пошук файлу? (y/n) ";
                        char answ = _getche();
                        if (answ == 'n') return -1;
                }

                mf = fopen(standartName.c_str(), "rt");

                char temp;
                unsigned long temp2;
                fscanf(mf,"%c",&temp);
                if ( temp == '1')
                {
                        text.clearCrypted();
                        if (!text.show_status()) text.change_status();
                        while(!feof(mf))
                        {
                                fscanf(mf,"%lu",&temp2);
                                text.pushBackCrypred(temp2);
                        }
                }
                else if( temp == '0')
                {
                        text.set_data("");
                        if (text.show_status()) text.change_status();
                        while(!feof(mf))
                        {
                                fscanf(mf,"%c",&temp);
                                text.set_data_back(temp);
                        }
                }
                else return -1;
        }
        return 0;
}
unsigned long saveFile(Text& text, string atrubute = "auto")
{
        setlocale(LC_ALL,NULL);
        if( atrubute == "settings")
        {
                SetFileAttributes( TEXT("settings.key") ,FILE_ATTRIBUTE_NORMAL);
// Функція змінює атрибути файлу, забирає атрибут 'hide'
                mf = fopen( standartName.c_str(), "wt");
                fprintf(mf, "%i %i %i ",text.getOpenKey(), text.getSecretKey(),
text.getN());
                fclose(mf);
                SetFileAttributes( TEXT("settings.key") ,FILE_ATTRIBUTE_HIDDEN);
// Функція змінює атрибути файлу на скритий (сховує файл із ключами)
        }
        else
        {
                if (atrubute == "saveAs" ) renameFile();

                mf = fopen( standartName.c_str(), "wt");
                string toWrite = text.get_data();
                if (!text.show_status())
                {
                        fprintf(mf, "0\n%s",toWrite.c_str());
                }
                else
                {
                        vector <unsigned long> a ;
                        a = text.getCrypted();
                        fprintf(mf,"1\n");
```

```
                                for (unsigned long i =0; i<a.size() ; i++ )
                                        fprintf(mf, "%lu ",a[i]);
                        }
                        fprintf(mf,"\n");

                        fclose(mf);
                }
                return 0;
        }
        void renameFile()
        {
                string temp =standartName;
                standartName = NameOfFile();
                rename(temp.c_str() , standartName.c_str() );
        }
};



#endif
```

Лістинг «MyFunct.h»

```
#ifndef MyFunct_H
#define MyFunct_H

#include <string>
#include <conio.h>
#include <windows.h>

#include "FileFunct.h"
#include "RSA.h"


// Прототипи функцій
void printLine(char whatPrint = '_',unsigned long length = 80);     // Функція виводить
стмвол '_' 78 раз, малює умовну лінію
void exitProg(Text& t, MyFile& f);                                          // Функція ре-
алізує завершення програми і зберігання дані у файли
void outputText(Text& text);                                                // Функція, яка
виводить інформацію про введені дані (текст)
void encrypt(Text& text);                                                   // Функція шифру-
вання тексту
void unencrypt(Text& text);                                                 // Функція дешиф-
рування тексту
void saveSattings(Text& t);                                                 // Функція
зберігає ключі в файл
void loadFile(Text& text,MyFile& file);                                 // Фукнція для заванте-
ження даних (тексту) з файлу
void saveFile(Text& text, MyFile& file);                            // Фукнція для збереження даних
(тексту) в файлу
void inputKey(Text& text);                                                  // Функція для
введення ключів
void menu (Text& text, MyFile& file, string whatToDo);// Підменю
////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////


// Опис функцій
void MainMenu(Text& _text, MyFile& _file)  //  Функція головного меню
{
        char answ = '0';
        setlocale(LC_ALL, "rus");
```

```
	do
	{
		system("color 1f");
		cout << "\n\t\t ===== Виберіть комаду ===== \n"
		     << "\n\t 1 - Вивести дані"
		     << "\n\n\t 2 - Зашифрувати дані"
		     << "\n\t 3 - Розшифрувати дані"
		     <<     "\n\n\t 4 - Зберегти дані у файл"
		     << "\n\t 5 - Завантажити дані із файлу"
		     << "\n\n\t 0 - Вихід \n";

		answ = _getch();

		switch(answ)
		{
		case '1':
			{
				printLine();
				outputText(_text);
				printLine();
				_getch();break;
			}
		case '2':
			{
				printLine();
				menu(_text,_file,"шифрування");
				printLine();
				_getch();break;
			}
		case '3':
			{
				printLine();
				menu(_text, _file,"дешифрування");
				printLine();
				_getch();break;
			}
		case '4':
			{
				printLine();
				saveFile(_text,_file);
				cout << "\n Дані збережені в файл";
				printLine();
				_getch();break;
			}
		case '5':
			{
				printLine();
				loadFile(_text,_file);
				outputText(_text);
				printLine();
				_getch();break;
			}
		case '0':
			{
				printLine();
				exitProg( _text, _file);
				return;
			}
		}
		system("cls");
	} while (1);
}

void menu(Text& text, MyFile& file, string whatToDo)
{
```

```
cout << "\n \t\t === "<< whatToDo <<" ===\n"
       << "\n\t 1 - "<<whatToDo<<" дані з програми"
       << "\n\t 2 - ввести дані"
       << "\n\t 3 - зчитати дані з файлу"
       << "\n\n\t 0 - назад \n";

char answ = '9';

while(!(answ =='1' || answ =='2' || answ == '0'  || answ == '3'))
       answ = _getch();

if (text.get_data() == " - empty - " && answ == '1')
{
       answ = '2';
       cout << "\n Нічого не введено\n";
}

switch (answ)
{
case '1':
       {
              if(whatToDo == "шифрування" && !text.show_status())
              {
                     encrypt(text);
                     saveFile(text,file);
              }
              else if (whatToDo == "дешифрування" && text.show_status())
              {
                     unencrypt(text);
                     saveFile(text,file);
              }
              else
                     cout << "\n Неможливо \n";

              outputText(text);

              break;
              _getch();
       }
case '2':
       {
              cout << "\n Веведіть дані для "<<whatToDo<<" : " ;
              string temp;
              getline(cin,temp);
              text.set_data(temp);

              cout << "\n Ввести ключі для "<<whatToDo<<" ? (y/n) ";

              char answ = '\0';
              answ = _getche();


              if (answ == 'y')
              {
                     inputKey(text);
              }
              if(whatToDo == "шифрування")
              {

                     encrypt(text);
                     saveFile(text,file);
              }
              else if (whatToDo == "дешифрування")
              {
```

```
                        unencrypt(text);
                        saveFile(text,file);
                }

                outputText(text);

                break;
                _getch();
        }
    case '3':
        {
                loadFile(text,file);

                if(whatToDo == "шифрування" && !text.show_status())
                {
                        encrypt(text);
                        saveFile(text,file);
                }
                else if (whatToDo == "дешифрування" && text.show_status())
                {
                        unencrypt(text);
                        saveFile(text,file);
                }

                outputText(text);

                break;
                _getch();
        }
    case '0':
         return;

    default:;
    }

}

void encrypt(Text& text)
{
    RSA rsa ( text.get_data(),text.getOpenKey(),text.getSecretKey(),text.getN());

    rsa.encrypt ();                                                    // Шифру-
вання
    text.setCrypted( rsa.getCrypted() );                  // Передача даних в об'єкт
"Текст"
    text.setOpenKey( rsa.getOpentKey());                  //
    text.setSecretKey( rsa.getSecretKey());               //
    text.setN( rsa.getN());                                            // --------------
--------------
    if (!text.show_status()) text.change_status();  // Вказання, що стан "Тексту" захифро-
ваний
}
void unencrypt(Text& text)
{
    RSA rsa ( text.getCrypted(),text.getOpenKey(),text.getSecretKey(),text.getN());
    text.set_data(  rsa.unencrypt() );
    if (text.show_status()) text.change_status();
}

void outputText(Text& text)
{
    setlocale(LC_ALL,NULL);
    if (!text.show_status())
    {
            cout << "\n Дані (не зашифровані) : \n" ;
```

```
                printLine('.');
                cout << endl << text.get_data() << endl;
                printLine('.');
        }
        else
        {
                cout << "\n Дані (зашифровані) : \n" ;
                printLine('.');
                cout << endl;
                text.outputCryptedText();
                cout << endl;
                printLine('.');
                cout << "\n Відкритий ключ (E) = " << text.getOpenKey()
                        << "\n N = " << text.getN() << endl;
        }
}
void saveFile(Text& text, MyFile& file)
{
        file.saveFile( text);
        saveSattings(text);
}
void printLine(char whatPrint,unsigned long length)
{
        unsigned long i=0;
        cout << "\n ";
        while(i++ < length - 2)
                cout << whatPrint;
        cout << " \n";
}
void exitProg(Text& t, MyFile& f)
{
        f.saveFile(t);

        saveSattings(t);
        cout<< "\n Exit ... \n";
        exit(1);
}
void saveSattings(Text& t)
{
        MyFile settings("settings.key","wt");
        settings.saveFile(t,"settings");
}
void loadFile(Text& text, MyFile& file)
{
        MyFile settings("settings.key","rt");
        if (file.loadFile(text)    )
                cout << "\n Дані не завантажено!\n";
        else
                cout << "\n Дані завантажено\n";

        if (settings.loadFile(text,"settings"))
        {
                if (text.show_status())
                {
                        cout << "\n Файл ( settings.key ) із ключами не знайдено. Ввести ключі?
(y/n) ";
                        char answer = '\0';

                        while(answer != 'n' && answer != 'y')
                                answer = _getche();


                        if(answer == 'y')
                                inputKey(text);
```

```
                    else if (answer == 'n')
                    {
                           cout << "\n Дешифрування без ключів не можливе\n";
                           return;
                    }
             }

      }
}
void inputKey(Text& text)
{
      unsigned long temp;
      cout << "\n Введіть відкритий ключ (E) ";
      cin >> temp;
      text.setOpenKey(temp);
      cout << "\n Введіть закритий ключ (D) ";
      cin >> temp;
      text.setSecretKey(temp);
      cout << "\n Введіть число N ";
      cin >> temp;
      text.setN(temp);
}


                                          #endif
```

Лістинг «main.cpp»

```
#include <iostream>

#include "text.h" // Підключення класу 'Text'-- тексту із розширеними властивостями
#include "MyFunct.h" // Бібліотека із деякими моїми функціями
#include "RSA.h"

#include <conio.h>

using namespace std;

int main (int argc, char* argv[])
{
      Text new_text(" - empty - ");
      MyFile data_file("temp.dat", "rt");

      MainMenu(new_text, data_file);

      _getch();
      return 0;
}
```

ДОДАТОК Б

Діаграма класів