Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 1

з дисципліни

“ООП”

Тема: «Класи С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко Максим Миколайович |  | студент 2-го курсу  гр. ІП-52 ФІОТ  Набоков Едуард Максимович |

Київ – 2016

Зміст:

[1 Мета роботи 3](#_Toc462796421)

[2 Постановка задачі (варіант 10, рівень В) 4](#_Toc462796422)

[3 Покроковий алгоритм 5](#_Toc462796423)

[4 Діаграма класів 8](#_Toc462796424)

[5 Код програми 9](#_Toc462796425)

[6 Приклади виконання програми 17](#_Toc462796426)

[7 Висновок 19](#_Toc462796427)

# Мета роботи

Мета роботи – вивчити основні концепції обєктно-орієнтованого програмування. Вивчити особливості використання класів та об’єктів, а також особливості використання конструкторів та деструкторів.

# Постановка задачі (варіант 10, рівень В)

Определить класс «Криптография», с закрытым элементом: t – который представляет собой некоторый текст. Предусмотреть в классе:

– конструктор инициализации, для ввода текста с клавиатуры (задать некоторый текст по умолчанию).

Класс должен содержать методы для шифровки и расшифровки текста по трем алгоритмам или более. Сымитировать атаку криптоаналитика (имеется зашифрованный текст, делается предположение о методе, которым он зашифрован, подбирается ключ и расшифровывается текст). Сравнить время затраченное на подбор ключей для разных методов.

Определить деструктор класса.

# Покроковий алгоритм

* Метод Цезаря(шифрування)

1. ПОЧАТОК
2. Обирається числовий ключ (**key**) від 1 до нескінченності.
3. Знаходимо остачу від ділення на 26 (к-сть літер в англійскому алфавіті)
4. ЦИКЛ по всім буквеним символам рядка (**sentence**)
   1. Зміщуємо кожну літеру вперед по алфавіту на довжину ключа (**key**)
   2. ЯКЩО поточна літера більша за останню літеру «z»(«Z») , то присвоюємо поточній літері значення першої літери у алфавіті та зміщуємо її на різницю попередньої літери та літери «z» («Z»).
5. КІНЕЦЬ

* Метод Цезаря(дешифрування)

1. ПОЧАТОК
2. Використовується той самий ключ, що і при шифруванні.
3. ЦИКЛ по всім буквеним символам рядка (**sentence**)
   1. Зміщуємо кожну літеру назад по алфавіту на довжину ключа (**key**)
   2. ЯКЩО поточна літера менша за першу літеру «a»(«A») , то присвоюємо поточній літері значення останньої літери у алфавіті та зміщуємо її ще на різницю попередньої літери та літери «a» («A»).
4. КІНЕЦЬ.

* Метод Френсіса Бекона(шифрування)

1. ПОЧАТОК
2. Кожна літера англ. алфавіту замінюється на індекс (0, 25) та переводиться у бінарний код (довжина у 5 бітів), де нуль – «А», одиниця – «В».
3. ЦИКЛ по кожній літері речення (**sentence**)
   1. Знаходимо еквівалент поточній літері у бінарному коді.
4. КІНЕЦЬ

* Метод Френсіса Бекона(дешифрування)

1. ПОЧАТОК
2. Записуємо шифроване речення у поточний файл (**input.txt**) замінюючи «А» та «В» на нуль та одиницю відповідно.
3. ЦИКЛ поки не скінчиться речення у файлі
   1. Зчитуємо по 5 «бітів»(літер) у масив (**code**)
   2. Конвертуємо символьний рядок у число та зберігаємо (**bin\_code**)
   3. Переводимо поточне число з бінарного коду у десятичне
   4. Знаходимо літеру під поточним знайденим індексом та записуємо до нового файлу (**output.txt**)
4. Кінець.

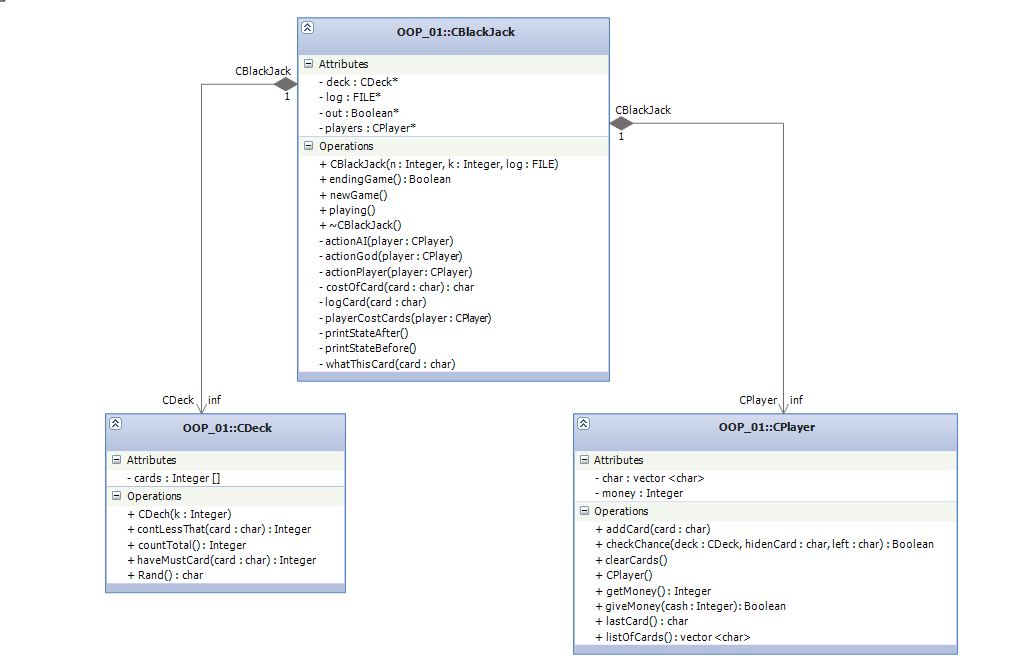
* Метод матриці (або Полібія)(шифрування)

1. ПОЧАТОК
2. Заповнюємо символьну матрицю алфавітом зміщену на ключ (**key**) від початку.
3. ЦИКЛ по кожній літері символьного рядка (**sentence**)
   1. ЦИКЛ по кожній літері символьній матриці
      1. ЯКЩО літера знайдена в алфавіті ТО записуємо до файлу координати літери у матриці (рядок та стовпчик) (**enMatr.txt**)
4. Зчитуємо у масив (**text**) дані з поточного файлу.
5. КІНЕЦЬ

* Метод матриці (або Полібія)(дешифрування)

1. ПОЧАТОК
2. Використовуємо ту саму символьну матрицю.
3. Розбиваємо поточний шифрований текст по два символи та знаходимо відповідну літеру у матриці
   1. ЦИКЛ по всім символам рядка з кроком (**i += 2**)
      1. Записуємо кожну першу літеру у (**row**) та кожну другу у (**col**)
      2. Знаходимо літеру у матриці.
      3. Записуємо у поточний файл (**deMatr.txt**)
4. Зчитуємо з поточного файла у символьний рядок (**text**)
5. КІНЕЦЬ

# Діаграма класів



# Код програми

“main.cpp”

#include <iostream>

#include <stdio.h>

#include <ctime>

#include <string>

#include <fstream>

#include "Support.h"

#pragma warning(disable : 4996)

using namespace std;

typedef char Matrix[25][25];

int main()

{

char sentence[1000], initial[100];

CDecode x; //creation of object

strcpy(sentence, x.getT()); //copying from objects' string to simple string

strcpy(initial, sentence);

Matrix matrKey;

int choice;

int key;

cout << "Choose one of the following methods:\n[1] -> Caesar\n[2] -> Fransis Backon(Binary)\n[3] -> Matrix\n[4] -> Break\n Your choice: "; cin >> choice;

getchar(); //to catch '\n'

switch (choice)

{

case 1: cout << "Enter key(number): "; cin >> key; key %= 26; x.enCaesar(sentence, key); cout << endl << "Encoded: " << sentence << endl; x.deCaesar(sentence, key); break;

case 2: x.enFransisBackon(sentence); cout << endl << "Encoded: " << sentence << endl; x.deFransisBackon(sentence); cout << "Decoded: " << sentence << endl; break;

case 3: cout << "Enter key(number): "; cin >> key; key %= 26; keyGenerator(matrKey, key); x.enMatr(sentence, matrKey); cout << "Encoded: " << sentence << endl; x.deMatr(sentence, matrKey); cout << "Decoded: " << sentence; break;

case 4: break;

}

cout << "\nDecoded!\n";

cout << "Result of decoding: " << sentence << endl;

getchar(); //to catch '\n'

cout << "\n\n\nCRYPTOATTACK!\n";

cout << "Enter encoded text: ";

char encoded[100];

gets\_s(encoded, 100);

cout << "Make some predictions to decrypt this " << encoded << " encoded text.\n[1] -> Caesar\n[2] -> Fransis Backon(Binary)\n[3] -> Matrix\n[4] -> break\nYour choice: ";

cin >> choice;

switch (choice)

{

case 1: analyzeCaesar(encoded); cout << endl; break; //calling Caesar analyzer

case 2: analyzeFB(encoded); cout << endl; break; //calling Fransis Backon analyzer

case 3: analyzeMatr(encoded); cout << endl; break; //calling Matrix analyzer

case 4: break;

}

system("pause");

}

“Support.h”

#pragma once

#pragma warning (disable : 4996)

typedef char Matrix[25][25];

using namespace std;

class CDecode

{

private:

char t[1000];

public:

//constructor by default

CDecode()

{

strcpy(t, "empty");

}

//get function of char\*

char \* getT()

{

cout << "Enter the sentence: ";

gets\_s(t, 100);

return t;

}

// methods of encryption

void enCaesar(char \*text, int key);

void enFransisBackon(char \*text);

void enMatr(char \*text, Matrix);

//methods of decryption

void deCaesar(char \*text, int);

void deFransisBackon(char \*text);

void deMatr(char \*text, Matrix);

//destructor of class

~CDecode() {}

};

//analyzing of different methods

void analyzeCaesar(char\*);

void analyzeFB(char \*);

void analyzeMatr(char \*);

//generator for key for shifting of Matrix creation

void keyGenerator(Matrix &, int);

char \*get\_code(char); //convert letter to binary code

int bin\_to\_dec(int); //func that implemented for coverting binary number to decimal

“Support.cpp”

#include <iostream>

#include <stdio.h>

#include <ctime>

#include <string>

#include <fstream>

#include "Support.h"

using namespace std;

void CDecode::enCaesar(char \*text, int key)

{

for (int i = 0; i < strlen(text); i++)

{

if (text[i] == ' ')

i += 1;

if (text[i] >= 'A' && text[i] <= 'Z')

{

text[i] = text[i] + (char)key;

if (text[i] > 'Z')

text[i] = 'A' + (text[i] - 'Z') - 1;

}

if (text[i] >= 'a' && text[i] <= 'z')

{

text[i] = text[i] + (char)key;

if (text[i] > 'z' || text[i] < 'A')

text[i] = 'a' + (text[i] - 'z') - 1;

}

}

}

void CDecode::deCaesar(char \*text, int key)

{

for (int i = 0; i < strlen(text); i++)

{

if (text[i] >= 'A' && text[i] <= 'Z')

{

text[i] = text[i] - (char)key;

if (text[i] < 'A') text[i] = 'Z' - ('A' - text[i]) + 1;

}

if (text[i] >= 'a' && text[i] <= 'z')

{

text[i] = text[i] - (char)key;

if (text[i] < 'a') text[i] = 'z' - ('a' - text[i]) + 1;

}

}

}

void CDecode::enFransisBackon(char \*text)

{

char code[10] = "";

FILE \*s;

s = fopen("in.txt", "w+");

for (int i = 0; i < strlen(text); i++)

{

if (!isdigit(text[i]) && isalpha(text[i]) && text[i] != ' ')

{

text[i] = tolower(text[i]);

strcpy(code, get\_code(text[i]));

cout << endl << code;

fprintf(s, "%s", code);

}

}

rewind(s);

fgets(text, 1000, s);

fclose(s);

remove("in.txt");

}

void CDecode::deFransisBackon(char \*text)

{

char ABC[] = "abcdefghijklmnopqrstuvwxyz";

FILE \*f = fopen("input.txt", "w+");

FILE \*s = fopen("output.txt", "w+");

//fwrite(text, sizeof(text), 100, f); all of string

char bin\_bit;

for (int i = 0; i < strlen(text); i++)

{

if (text[i] == 'A')

bin\_bit = '0';

if (text[i] == 'B')

bin\_bit = '1';

fprintf(f, "%c", bin\_bit);

}

char code[10] = "";

rewind(f);

for (int i = 0; i < strlen(text) - 4; i += 5)

{

fgets(code, 6, f);

int bin\_code;

sscanf(code, "%d", &bin\_code); //conver string to int

int decimal = 0;

cout << "Binary: " << bin\_code << " ";

decimal = bin\_to\_dec(bin\_code);

cout << "Decimal: " << decimal << endl;

fprintf(s, "%c", ABC[decimal]);

}

fclose(s);

s = fopen("output.txt", "r");

fscanf(s, "%s", text);

fclose(s);

fclose(f);

/\*remove("input.txt");

remove("output.txt"); \*/

}

void CDecode::enMatr(char \*text, Matrix key)

{

char b[100], a[100];

FILE \*p;

p = fopen("enMatr.txt", "w+");

int flag = 0;

for (int i = 0; i < strlen(text); i++)

{

if (text[i] == ' ')

i += 1;

flag = 0;

if (isalpha(text[i]) || text[i] == ' ')

for (int j = 0; j < 6; j++)

{

for (int z = 0; z < 6; z++)

if (tolower(text[i]) == key[j][z])

{

///p << char(j + 1) << char(z +

sprintf(a, "%d", j + 1);

sprintf(b, "%d", z + 1);

fprintf(p, "%s", a);

fprintf(p, "%s", b);

flag = 1;

if (flag)

break;

}

if (flag)

break;

}

}

rewind(p);

fgets(text, strlen(text) \* 36, p);

//cout << "Encrypted text: " << text << endl;

fclose(p);

remove("enMatr.txt");

}

void CDecode::deMatr(char \*text, Matrix key)

{

char a;

FILE \*p;

p = fopen("deMatr.txt", "w+");

int row, col;

for (int i = 0; i < strlen(text) - 1; i += 2)

{

row = text[i] - 48;

col = text[i + 1] - 48;

char a = key[row - 1][col - 1];

fprintf(p, "%c", a);

}

rewind(p);

fgets(text, strlen(text) \* 36, p);

fclose(p);

remove("deMatr.txt");

}

void keyGenerator(Matrix &Mkey, int key)

{

char initial = 'a' + (char)key;

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

initial += (char)1;

Mkey[i][j] = initial;

if (Mkey[i][j] > 'z')

{

initial = 'a';

Mkey[i][j] = initial;

}

}

}

Mkey[6][6] = ' ';

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

cout << Mkey[i][j] << " ";

cout << endl;

}

}

char \*get\_code(char ch)

{

switch (ch)

{

case 'a': return "AAAAA"; break; //0

case 'b': return "AAAAB"; break; //1

case 'c': return "AAABA"; break; //2

case 'd': return "AAABB"; break;//3

case 'e': return "AABAA"; break;//4

case 'f': return "AABAB"; break;//5

case 'g': return "AABBA"; break;//6

case 'h': return "AABBB"; break;//7

case 'i': return "ABAAA"; break;//8

case 'j': return "ABAAB"; break;//9

case 'k': return "ABABA"; break;//10

case 'l': return "ABABB"; break;//11

case 'm': return "ABBAA"; break;//12

case 'n': return "ABBAA"; break;//13

case 'o': return "ABBBA"; break;//14

case 'p': return "ABBBB"; break;//15

case 'q': return "BAAAA"; break;//16

case 'r': return "BAAAB"; break;//17

case 's': return "BAABA"; break;//18

case 't': return "BAABB"; break;//19

case 'u': return "BABAA"; break;//20

case 'v': return "BABAB"; break;//21

case 'w': return "BABBA"; break;//22

case 'x': return "BABBB"; break;//23

case 'y': return "BBAAA"; break;//24

case 'z': return "BBAAB"; break;//25

}

return "0";

}

int bin\_to\_dec(int bin\_code)

{

int rem = 0;

int decimal = 0;

int index = 0;

while (bin\_code != 0)

{

rem = bin\_code % 10;

bin\_code /= 10;

decimal += rem \* pow(2, index);

++index;

}

return decimal;

}

void analyzeCaesar(char \*encodedText)

{

cout << endl;

clock\_t start\_time = clock();

int key = 0;

while (key != 26)

{

for (int i = 0; i < strlen(encodedText); i++)

{

if (!isalpha(encodedText[i]))

i += 1;

if (encodedText[i] == ' ')

i += 1;

encodedText[i] += (char)1;

if (tolower(encodedText[i]) == 'z')

encodedText[i] = 'a';

}

key += 1;

//output time

cout << key << ": " << encodedText << " --- [time of executing: " << (double)(clock() - start\_time) / CLOCKS\_PER\_SEC << " s]" << endl;

}

}

void analyzeFB(char \*encodedText)

{

char Abc[] = "abcdefghijklmnopqrstuvwxyz";

cout << endl;

clock\_t start\_time = clock();

FILE \*f = fopen("input.txt", "w+");

FILE \*o = fopen("output.txt", "w+");

//recording the whole row to the file

for (int j = 0; j < strlen(encodedText); j++)

{

if (toupper(encodedText[j]) != 'A' && toupper(encodedText[j]) != 'B')

{

cout << "\nWrong method for such decryption!\n";

return;

}

if (toupper(encodedText[j]) == 'A')

encodedText[j] = '0';

if (toupper(encodedText[j]) == 'B')

encodedText[j] = '1';

fputc(encodedText[j], f);

}

rewind(f); int flag = 0;

//make prediction that that is binary system but 0 is A, 1 - B

//check the correctness of char row

char code[100] = "";

int bin\_code;

if (strlen(encodedText) % 5 == 0)

{

flag = 1;

for (int i = 0; i < strlen(encodedText) - 4; i += 5)

{

fgets(code, 6, f);

sscanf(code, "%d", &bin\_code);

int decimal = 0;

cout << "Binary: " << bin\_code << " ";

decimal = bin\_to\_dec(bin\_code);

cout << "Decimal: " << decimal << endl;

fprintf(o, "%c", Abc[decimal]);

}

fclose(o);

o = fopen("output.txt", "r");

fscanf(o, "%s", encodedText);

fclose(o);

fclose(f);

//output of time

cout << "Result: " << encodedText << " --- [time of executing: " << (double)(clock() - start\_time) / CLOCKS\_PER\_SEC << " s]" << endl;

return;

}

if (!flag)

{

fclose(o);

fclose(f);

}

}

void analyzeMatr(char\* encodedText)

{

FILE \*s; //= fopen("smth.txt", "w+");

Matrix abc;

int key = 0;

clock\_t start\_time = clock();

int row, col;

char a;

char result[100];

for (int i = 0; i < strlen(encodedText); i++)

{

if (isalpha(encodedText[i]))

{

cout << "\nWrong method for such encryption\n";

return;

}

}

if (strlen(encodedText) % 2 == 0)

{

while (key != 26)

{

s = fopen("smth.txt", "w+");

cout << key + 1 << ": " << endl;

keyGenerator(abc, key);

row = 0, col = 0;

for (int i = 0; i < strlen(encodedText) - 1; i += 2)

{

row = encodedText[i] - 48;

col = encodedText[i + 1] - 48;

a = abc[row - 1][col - 1];

fputc(a, s);

}

rewind(s);

fgets(result, strlen(encodedText) \* 36, s);

//output time

cout << result << " --- [time of executing: " << (double)(clock() - start\_time) / CLOCKS\_PER\_SEC << " s]" << endl << endl;

key += 1;

fclose(s);

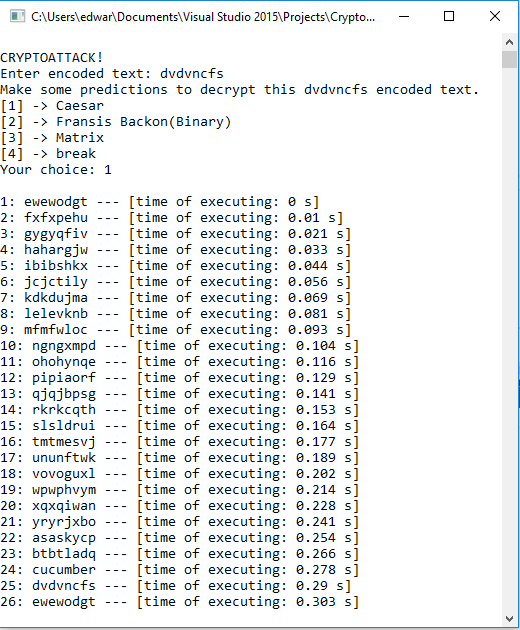
//remove("smth.txt");

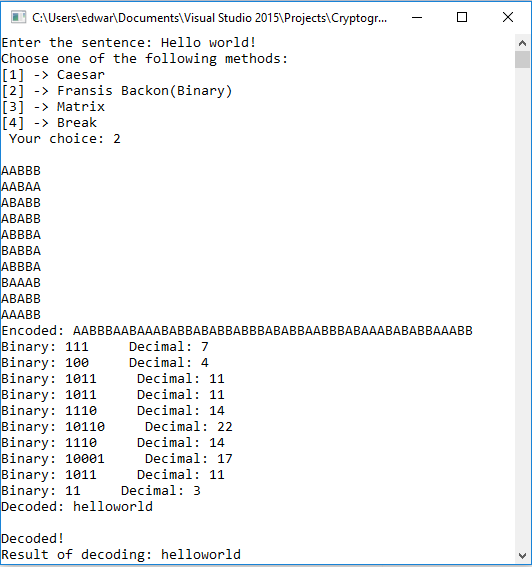
}

}

}

# Приклади виконання програми





# Висновок

Отже, дана програма створює об’єкт класа CDecode, яка реалізує популярні та перші методи шифрування інформації, а також їх декодування за допомогою ключа. Також було реалізовано декодування без ключа(brute force) та оцінено час, який потрібен для взлому.