Лабораторная работа №XX

по предмету «**Информационная безопасность**»

**Выполнила:** XXXX.

**Группа:** XXXX

**Проверила:** др.,Чербу О.

Кишинев, 2021

Задача:

Написать программу для шифрования и дешифровки сообщения по методу TEA.

# Теоретическая часть:

Tiny Encryption Algorithm (TEA) — [блочный алгоритм шифрования](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) типа «[Сеть Фейстеля](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F" \o "Сеть Фейстеля)». Алгоритм шифрования TEA основан на [битовых операциях](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8) с 64-битным блоком, имеет 128-битный [ключ шифрования](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)).

1. **Исходный текст разбивается на блоки по 64 бита каждый.**
2. **128-битный ключ К делится на четыре 32-битных подключа K[0], K[1], K[2] и K[3].**

На этом подготовительный процесс заканчивается, после чего каждый 64-битный блок шифруется на протяжении 32 циклов (64 раундов) по нижеприведённому алгоритму.[[5]](https://ru.wikipedia.org/wiki/TEA#cite_note-differential-5)

1. Предположим, что на вход n-го раунда (для 1 ≤ n ≤ 64) поступают правая и левая часть (Ln, Rn), тогда на выходе n-го раунда будут левая и правая части (Ln+1, Rn+1), которые вычисляются по следующим правилам:

**Ln+1 = Rn.**

Если n = 2 \* i — 1 для 1 ≤ i ≤ 32 (нечётные раунды), то

**Rn+1 = Ln \boxplus ({ [ Rn \ll 4 ] \boxplus K[0] } \oplus { Rn \boxplus i \* δ } \oplus { [ Rn \gg 5 ] \boxplus K[1] })**

Если n = 2 \* i для 1 ≤ i ≤ 32 (чётные раунды), то

**Rn+1 = Ln \boxplus ({ [ Rn \ll 4 ] \boxplus K[2] } \oplus { Rn \boxplus i \* δ } \oplus { [ Rn \gg 5 ] \boxplus K[3] })**

Где

* *X \boxplus Y — операция сложения чисел X и Y по модулю 232.*
* *X \oplus Y — побитовое* [*исключающее «ИЛИ» (XOR)*](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2) *чисел X и Y, которое в* [*языке программирования Си*](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) *обозначается как X ^ Y*
* *X \ll Y и X \gg Y — операции* [*побитового сдвига*](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B4%D0%B2%D0%B8%D0%B3) *числа X на Y бит влево и вправо соответственно.*
* *Константа δ была выведена из* [*Золотого сечения*](https://ru.wikipedia.org/wiki/%D0%97%D0%BE%D0%BB%D0%BE%D1%82%D0%BE%D0%B5_%D1%81%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)***δ = (\sqrt{5} - 1) \* 231 = 2654435769 = 9E3779B9h.*** *В каждом раунде константа умножается на номер цикла i.*

В нечётных раундах используются подключи К[0] и К[1], в чётных — К[2] и К[3].

Так как это блочный шифроалгоритм, где длина блока 64-бит, а длина данных может быть не кратна 64-битам, значения всех байтов дополняющих блок до кратности в 64-бит устанавливается в 0x01 .

# Листинг приложения:

#include <stdint.h>

#include <iostream>

using namespace std;

void encrypt (uint32\_t\* LR\_mes, uint32\_t\* key) {

uint32\_t L=LR\_mes[0], R=LR\_mes[1], sum=0;

uint32\_t delta=0x9e3779b9;

for (int i=0; i < 32; i++){

sum += delta;

L += ((R<<4) + key[0]) ^ (R + sum) ^ ((R>>5) + key[1]);

R += ((L<<4) + key[2]) ^ (L + sum) ^ ((L>>5) + key[3]);

}

LR\_mes[0] = L;

LR\_mes[1] = R;

}

void decrypt (uint32\_t\* LR\_mes, uint32\_t\* key) {

uint32\_t L=LR\_mes[0], R=LR\_mes[1], sum=0xC6EF3720;

uint32\_t delta=0x9e3779b9;

for (int i=0; i < 32; i++){

R -= ((L<<4) + key[2]) ^ (L + sum) ^ ((L>>5) + key[3]);

L -= ((R<<4) + key[0]) ^ (R + sum) ^ ((R>>5) + key[1]);

sum -= delta;

}

LR\_mes[0] = L;

LR\_mes[1] = R;

}

int main(){

char message[256];

char crypt[256];

char decrypt\_line[256];

cout << "\nPlease, enter the message for encryption: \n";

fflush(stdin);

gets(message);

cout << "Your message is : \n" << message << endl;

uint32\_t \*arr, \*k;

uint64\_t \*message\_blocks, \*crypt\_blocks, \*decrypt\_blocks;

char key[16];

int block\_number;

if (strlen(message)%8 != 0)

block\_number = strlen(message)/8+1;

else

block\_number = strlen(message)/8;

message\_blocks = new uint64\_t[block\_number];

crypt\_blocks = new uint64\_t[block\_number];

decrypt\_blocks = new uint64\_t[block\_number];

arr = new uint32\_t[2\*block\_number];

k = new uint32\_t[4];

//uint64\_t long\_type = 2100005695555;

if((strlen(message)%8) != 0){

int new\_strlen = 8\*((strlen(message)/8)+1);

for(int i=strlen(message); i < new\_strlen; ++i){

message[i] = 0;

}

message[new\_strlen] = '\0';

cout << "\n\nModified message = " << message << " strlen = " << strlen(message) <<endl;

}

for(int i=0;i < block\_number; ++i){

message\_blocks[i] = ((uint64\_t)(message[i\*8]) << 56) | ((uint64\_t)(message[i\*8+1]) << 48) |

((uint64\_t)(message[i\*8+2]) << 40) | ((uint64\_t)(message[i\*8+3]) << 32) |

((uint64\_t)(message[i\*8+4]) << 24) | ((uint64\_t)(message[i\*8+5]) << 16) |

((uint64\_t)(message[i\*8+6]) << 8) | ((uint64\_t)(message[i\*8+7]));

cout << "\nmessage\_blocks[" << i << "] = ";

for (int j = (8\*sizeof(message\_blocks[i])-1); j > -1; j--)

(((message\_blocks[i] >> j) & 1) == 1) ? cout << 1 : cout << 0;

}

cout << "\nPlease, enter the key: \n";

fflush(stdin);

gets(key);

for(int i=strlen(key); i < 16; ++i){

key[i] = 0;

}

for(int i=0; i < 4; i++){

k[i] = ((key[i\*4] << 24) | (key[i\*4+1] << 16) | (key[i\*4+2] << 8) | (key[i\*4+3]));

cout << "\nk[" << i << "] = ";

for (int j = (8\*sizeof(k[i])-1); j > -1; j--)

(((k[i] >> j) & 1) == 1) ? cout << 1 : cout << 0;

}

for(int i=0; i < block\_number; ++i){

arr[0] = (message\_blocks[i] & 0xFFFFFFFF00000000ULL) >> 32;

arr[1] = message\_blocks[i] & 0xFFFFFFFF;

cout << "\n\nmessage\_blocks[" << i << "] = " << message\_blocks[i]

<< "\narr[0] = " << arr[0] << " arr[1] = " << arr[1] << endl;

for (int j = (8\*sizeof(arr[0])-1); j > -1; j--)

(((arr[0] >> j) & 1) == 1) ? cout << 1 : cout << 0;

cout << endl;

for (int j = (8\*sizeof(arr[1])-1); j > -1; j--)

(((arr[1] >> j) & 1) == 1) ? cout << 1 : cout << 0;

encrypt(arr,k);

cout << "\nEncrypted arr"

<< "\narr[0] = " << arr[0] << " arr[1] = " << arr[1] << endl;

for (int j = (8\*sizeof(arr[0])-1); j > -1; j--)

(((arr[0] >> j) & 1) == 1) ? cout << 1 : cout << 0;

cout << endl;

for (int j = (8\*sizeof(arr[1])-1); j > -1; j--)

(((arr[1] >> j) & 1) == 1) ? cout << 1 : cout << 0;

crypt\_blocks[i] = ((arr[0] & 0x00000000FFFFFFFFULL) << 32) | arr[1];

cout << "\nEncrypted message\_block : " << crypt\_blocks[i] << " : \n";

for (int j = (8\*sizeof(crypt\_blocks[i])-1); j > -1; j--)

(((crypt\_blocks[i] >> j) & 1) == 1) ? cout << 1 : cout << 0;

for(int j=0; j < 8; ++j){

int tmp = crypt[i\*8+j] = (crypt\_blocks[i] >> (56-8\*j)) & 0x000000FF;

cout << "\n" << crypt[i\*8+j] << "\n";

for (int j = (8\*sizeof(tmp)-1); j > -1; j--)

(((tmp >> j) & 1) == 1) ? cout << 1 : cout << 0;

}

}

crypt[block\_number\*8] = '\0';

cout << "\n===========================================================\n";

cout << "The encryption is : " << crypt;

for(int i=0; i < block\_number; ++i){

arr[0] = (crypt\_blocks[i] & 0xFFFFFFFF00000000ULL) >> 32;

arr[1] = crypt\_blocks[i] & 0xFFFFFFFF;

decrypt(arr,k);

decrypt\_blocks[i] = ((arr[0] & 0x00000000FFFFFFFFULL) << 32) | arr[1];

for(int j=0; j < 8; ++j){

int tmp = decrypt\_line[i\*8+j] = (decrypt\_blocks[i] >> (56-8\*j)) & 0x000000FF;

}

}

decrypt\_line[block\_number\*8] = '\0';

cout << "\n===========================================================\n";

cout << "The decryption is : " << decrypt\_line;

cin.get();

delete [] arr;

delete [] k;

delete [] message\_blocks;

delete [] crypt\_blocks;

delete [] decrypt\_blocks;

return 0;

}

# Результат:

Please, enter the message for encryption:

I like cryptography

Your message is :

I like cryptography

Modified message = I like cryptography strlen = 19

message\_blocks[0] = 010010010010000001101100011010010110101101100101001000000110

0011

message\_blocks[1] = 011100100111100101110000011101000110111101100111011100100110

0001

message\_blocks[2] = 011100000110100001111001000000000000000000000000000000000000

0000

Please, enter the key:

apple

k[0] = 01100001011100000111000001101100

k[1] = 01100101000000000000000000000000

k[2] = 00000000000000000000000000000000

k[3] = 00000000000000000000000000000000

message\_blocks[0] = 5269330764052635747

arr[0] = 1226861673 arr[1] = 1801789539

01001001001000000110110001101001

01101011011001010010000001100011

Encrypted arr

arr[0] = 3288083938 arr[1] = 3656864325

11000011111111000010110111100010

11011001111101110101001001000101

Encrypted message\_block : 14122212983869755973 :

1100001111111100001011011110001011011001111101110101001001000101

├

11111111111111111111111111000011

№

11111111111111111111111111111100

-

00000000000000000000000000101101

т

11111111111111111111111111100010

┘

11111111111111111111111111011001

ў

11111111111111111111111111110111

R

00000000000000000000000001010010

E

00000000000000000000000001000101

message\_blocks[1] = 8248747837893341793

arr[0] = 1920561268 arr[1] = 1869050465

01110010011110010111000001110100

01101111011001110111001001100001

Encrypted arr

arr[0] = 2376237686 arr[1] = 2980394827

10001101101000101000001001110110

10110001101001010011011101001011

Encrypted message\_block : 10205863151873111883 :

1000110110100010100000100111011010110001101001010011011101001011

Н

11111111111111111111111110001101

в

11111111111111111111111110100010

В

11111111111111111111111110000010

v

00000000000000000000000001110110

▒

11111111111111111111111110110001

е

11111111111111111111111110100101

7

00000000000000000000000000110111

K

00000000000000000000000001001011

message\_blocks[2] = 8099856970732797952

arr[0] = 1885894912 arr[1] = 0

01110000011010000111100100000000

00000000000000000000000000000000

Encrypted arr

arr[0] = 2869217433 arr[1] = 1258175252

10101011000001001100100010011001

01001010111111100011101100010100

Encrypted message\_block : 12323195041106246420 :

1010101100000100110010001001100101001010111111100011101100010100

л

11111111111111111111111110101011

♦

00000000000000000000000000000100

╚

11111111111111111111111111001000

Щ

11111111111111111111111110011001

J

00000000000000000000000001001010

■

11111111111111111111111111111110

;

00000000000000000000000000111011

¶

00000000000000000000000000010100

===========================================================

The encryption is : ├№-т┘ўREНвВv▒е7Kл♦╚ЩJ■;¶

===========================================================

The decryption is : I like cryptography