

**Министерство науки и высшего образования
Российской Федерации**

**Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Тульский государственный университет»**

Интернет-институт ТулГУ

Кафедра ИБ

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

«Методы и средства защиты компьютерной информации»

Семестр 7

Вариант № 5

Выполнил:

**студент группы ИБ262521-ф
Артемов Александр Евгеньевич**

Проверил:

**канд. техн. наук, доц.
Сафронова Марина Алексеевна**

Тула, 2025

Лабораторная работа № 1.

Название работы: Симметричные криптографические системы.

Цели работы: Освоение теоретических методов создания симметричных криптографических систем, получение практических навыков в преобразовании информации с помощью ЭВМ.

Задание:

1. Составить алгоритм для выполнения прямого, а затем обратного преобразования текста в соответствии с вариантом. Номер вариант соответствует последней цифре номера зачетки.
2. Написать, отладить и выполнить программу для пункта 1.

Выполнение лабораторной работы.

Изучены теоретические сведения к выполнению работы.

1. Составление алгоритма для выполнения прямого, а затем обратного преобразования текста.

Номер зачетки — 220085: решаем вариант № 5.

Написать программу шифрования и дешифрования сообщения, используя древнеспартанский шифр «скитала».

Спартанцы заметили, что если полоску пергамента намотать спирально на палочку и написать на нем вдоль палочки текст сообщения, то после снятия полоски буквы в ней расположатся хаотично. Этот эффект можно получить, если записывать буквы не подряд, а через условленное число по кольцу до тех пор, пока весь текст не будет исчерпан. Сообщение **ВЫСТУПАЙТЕ** при окружности палочки в 3 буквы даст шифровку **ВУТЫПЕСАТЙ**.

Решение:

Для шифрования сообщения (в нашем случае **ВЫСТУПАЙТЕ**) используем следующую последовательность действий:

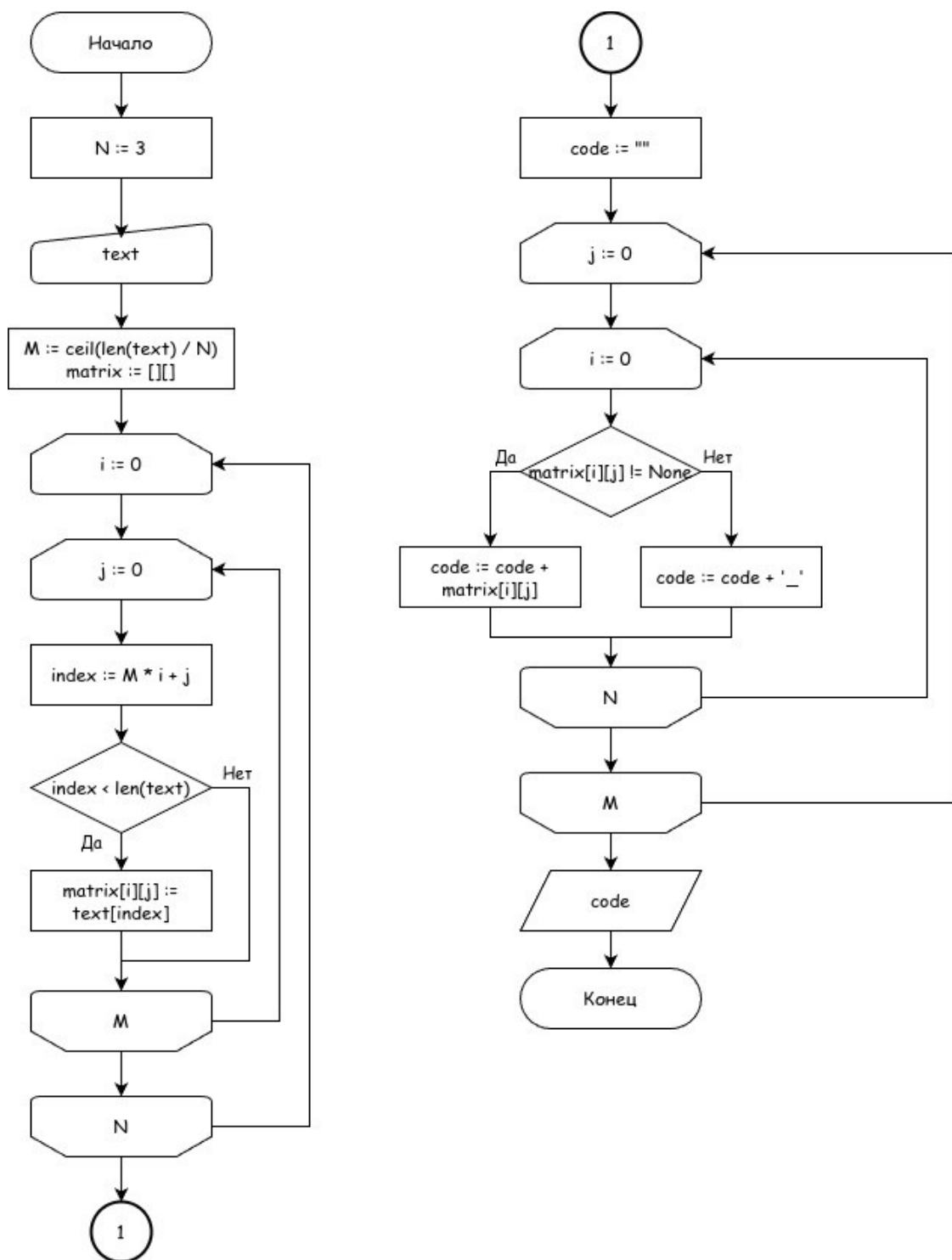
получим размерность необходимой матрицы: N — количество букв в окружности палочки (в нашем случае это 3) — строки, а M — длина сообщения разделенная на N и округленная вверх до целого — столбцы. Сообщение **ВЫСТУПАЙТЕ** имеет длину 10 букв, $M = \text{ОКРВВЕРХ}(10 / 3) = 4$;

внесем буквы текста в матрицу по строкам слева направо сверху вниз, в пустые ячейки запишем « — »:

В	Ы	С	Т
У	П	А	Й
Т	Е	—	—

вычитаем из матрицы код по столбцам сверху вниз слева направо, получим ВУТЫПЕСА_ТЙ_.

Оформим данный алгоритм в блок-схему.



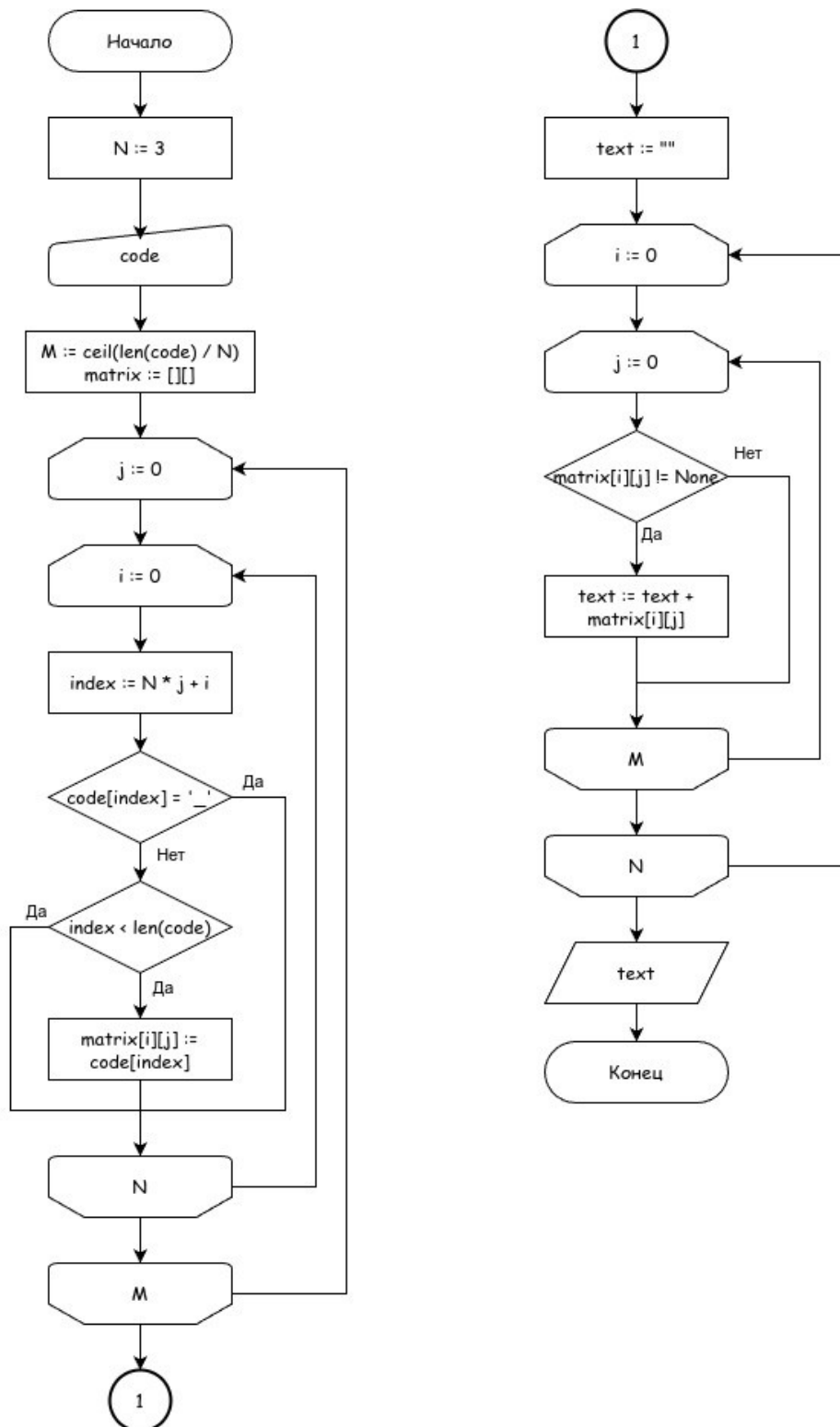
Для дешифровки кода (в нашем случае ВУТЫПЕСА_ТЙ_) используем следующую последовательность действий:

получим размерность необходимой матрицы аналогично процедуры шифрования;

внесем код в матрицу по столбцам сверху вниз слева направо:

В	Ы	С	Т
У	П	А	Й
Т	Е	—	—

вычитаем из матрицы текст сообщения по строкам слева направо сверху вниз, пропуская символ «_», получим сообщение ВЫСТУПАЙТЕ.
Оформим данный алгоритм в блок-схему.



2. Написание, отладка и выполнение программы для пункта 1.

Для создания программы шифрования и дешифрования сообщений методом древнеспартанского шифра «скитала» используем язык программирования Python версии 3.12 при помощи кроссплатформенного редактора кода Visual Studio Code. Создадим файл lab1.py содержащий следующий код:

```
from math import ceil

# в окружность палочки помещается N букв
N = 3

"""
шифруем текст
разносим текст в матрицу N * M где N - строки = число букв помещающихся в
окружность
M - столбцы = длина строки / N и округленная вверх до целого
В матрицу записываем текст по строкам, результат выводим по столбцам
"""

def encode(text):
    # получаем количество столбцов
    M = ceil(float(len(text)) / N)
    # создаем матрицу N * M
    matrix = [[None] * M for _ in range(N)]
    # заполняем матрицу текстом по строкам
    for i in range(N):
        for j in range(M):
            index = M * i + j
            if index < len(text):
                matrix[i][j] = text[index]
    # читаем матрицу по столбцам => получаем код
    code = ""
    for j in range(M):
        for i in range(N):
            if matrix[i][j] != None:
                code = code + matrix[i][j]
            else:
                code = code + '_'
    return code

"""
расшифровываем код
разносим код в матрицу N * M где N - строки = число букв помещающихся в окружность
M - столбцы = длина строки / N и округленная вверх до целого
В матрицу записываем код по столбцам, декодированный текст вычитываем по строкам
"""

def decode(code):
    # получаем количество столбцов
    M = ceil(float(len(code)) / N)
    # создаем матрицу N * M
    matrix = [[None] * M for _ in range(N)]
    # загоняем в матрицу код по столбцам
    for j in range(M):
        for i in range(N):
            index = N * j + i
```

```

        if code[index] == '_':
            continue
        if index < len(code):
            matrix[i][j] = code[index]
    text = ""
    # вычитываем из матрицы текст по строкам
    for i in range(N):
        for j in range(M):
            if matrix[i][j] != None:
                text = text + matrix[i][j]
    return text

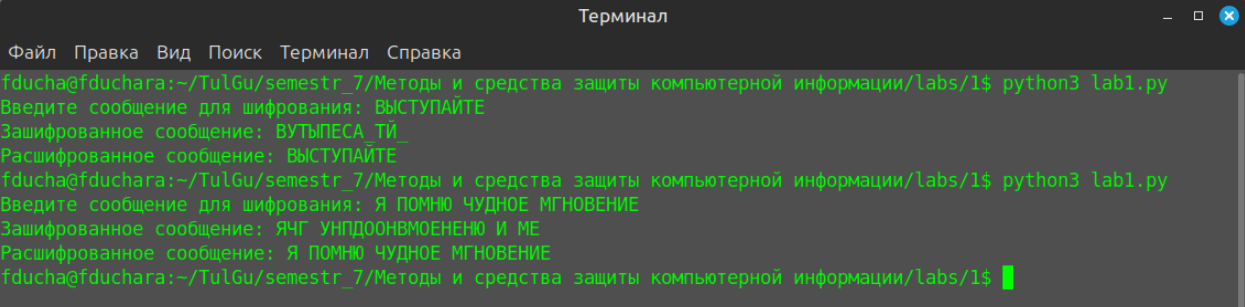
mes = input("Введите сообщение для шифрования: ")
encode_mes = encode(mes)
print("Зашифрованное сообщение:", encode_mes)

decode_mes = decode(encode_mes)
print("Расшифрованное сообщение:", decode_mes)

```

В коде импортируем функцию `ceil()` из модуля `math` для округления до целого, определяем глобальную переменную `N = 3`, хранящую количество букв на одном витке полосы. Далее определяем функции: `encode(text)` — шифрования сообщения и `decode(code)` — дешифровки кода.

Для тестирования шифрования просим пользователя ввести сообщение, шифруем его, выводим зашифрованное сообщение, и дешифруем его обратно, выводя результат.



The screenshot shows a terminal window titled "Терминал" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays the following commands and outputs:

```

fducha@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/1$ python3 lab1.py
Введите сообщение для шифрования: ВЫСТУПАЙТЕ
Зашифрованное сообщение: ВУТЫПЕСА ТИ
Расшифрованное сообщение: ВЫСТУПАЙТЕ
fducha@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/1$ python3 lab1.py
Введите сообщение для шифрования: Я ПОМНЮ ЧУДНОЕ МГНОВЕНИЕ
Зашифрованное сообщение: ЯЧГ УНПДООНВМОЕНЕЮ И МЕ
Расшифрованное сообщение: Я ПОМНЮ ЧУДНОЕ МГНОВЕНИЕ
fducha@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/1$

```

Контрольные вопросы.

1. В чем отличие симметричных и ассиметричных систем шифрования?

В симметричных криптосистемах для шифрования и для дешифрования используется один и тот же ключ.

В системах с открытым ключом используются два ключа – открытый и закрытый, которые математически связаны друг с другом. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

2. Какие виды симметричных криптосистем вы знаете?

Моно- и многоалфавитные подстановки, перестановки, гаммирование, блочные шифры.

3. В чем сущность подстановки Цезаря?

При шифровании исходного текста каждая буква заменяется на другую букву того же алфавита по следующему правилу. Заменяющая буква определяется путем смещения по алфавиту от исходной буквы на K букв. При достижении конца алфавита выполняется циклический переход к его началу. Цезарь использовал шифр замены при смещении $K = 3$. Такой шифр замены можно задать таблицей подстановок, содержащей соответствующие пары букв открытого текста и шифртекста.

4. Каким образом выполняется обратное преобразование текста, зашифрованного с помощью подстановки Цезаря?

Исходная буква определяется путем обратного смещения по алфавиту от зашифрованной буквы на K букв. При достижении начала алфавита выполняется циклический переход к его концу.

5. Назвать условия, которыми необходимо руководствоваться при выборе ключа в симметричных криптосистемах, для повышения криптоустойчивости системы шифрования.

Криптоустойчивость симметричных криптосистем увеличивается с увеличением длины ключа. Значение ключа должно оставаться конфиденциальным.

Лабораторная работа № 2.

Название работы: Парольные системы.

Цели работы: Получение практических навыков по использованию парольной аутентификации при разработке программного обеспечения.

Задание:

1. Отладить программу «Светофор», исходный текст которой приведен в приложении.
2. При запуске данной программы использовать парольную аутентификацию. В качестве пароля для запуска программы использовать слово длиной до 16 символов. В файл паролей записать результат выполнения функции (в соответствии с вариантом) над исходным паролем. Включить в программу защиты следующее условие: после трех неверных попыток введения пароля закрыть выполняемую программу и зафиксировать попытку входа в систему в журнале безопасности (текстовый файл, в котором фиксируется для неудачной попытки дата и время). При этом номер вариант соответствует последней цифре номера зачетки.

Выполнение лабораторной работы.

Изучены теоретические сведения к выполнению работы.

1. Отладить программу «Светофор», исходный текст которой приведен в приложении.

В указанном приложении дан исходный код программы, выполняющей шифрование — дешифрование при помощи алгоритма блочного шифрования DES, на языке программирования C. Данный код содержит множество ошибок и опечаток, после исправления которых программа успешно компилируется и выполняется. Для более удобного использования в дальнейшем, данный код был разделен на файлы `des.cpp` и `main.cpp`, а для компиляции был использован компилятор C++. В файл `des.cpp` перенесены функции и структуры исходного кода алгоритма, S-блоки и остальные данные. Демонстрация работы алгоритма перенесена в функцию `demo_DES_crypto()` в файле `main.cpp`.

Исходный код файла `des.cpp`:

```
#define EN0 0 /* MODE == encrypt */
#define DE1 1 /* MODE == decrypt */
typedef struct
{
    unsigned long ek[32];
    unsigned long dk[32];
} des_ctx;
void deskey(unsigned char *, short);
/* hexkey[8] MODE
 * Sets the internal key register according to the hexadecimal
 * key contained in the 8 bytes of hexkey, according to the DES,
 * for encryption or decryption according to MODE.
```



```

*/
void usekey(unsigned long *);
/* cookedkey[32]
 * Loads the internal key register with the data in cookedkey.
 */
void cpkey(unsigned long *);
/* cookedkey[32]
 * Copies the contents of the internal key register into the storage
 * located at &cookedkey[0].
 */
void des(unsigned char *, unsigned char *);
/* from[8] to[8]
 * Encrypts/Decrypts (according to the key currently loaded in the
 * internal key register) one block of eight bytes at address 'from'
 * into the block at address 'to'. They can be the same.
 */
static void scrunch(unsigned char *, unsigned long *);
static void unscrunch(unsigned long *, unsigned char *);
static void desfunc(unsigned long *, unsigned long *);
static void cookey(unsigned long *);
static unsigned long KnL[32] = {0L};
static unsigned long KnR[32] = {0L};
static unsigned long Kn3[32] = {0L};
static unsigned char Df_Key[24] = {
    0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef,
    0xfe, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10,
    0x89, 0xab, 0xcd, 0xef, 0x01, 0x23, 0x45, 0x67};
static unsigned short bytebit[8] = {
    0200, 0100, 040, 020, 010, 04, 02, 01};
static unsigned long bigbyte[24] = {
    0x800000L, 0x400000L, 0x200000L, 0x100000L,
    0x80000L, 0x40000L, 0x20000L, 0x10000L,
    0x8000L, 0x4000L, 0x2000L, 0x1000L,
    0x800L, 0x400L, 0x200L, 0x100L,
    0x80L, 0x40L, 0x20L, 0x10L,
    0x8L, 0x4L, 0x2L, 0x1L};
/* Use the key schedule specified in the Standard (ANSI X3.92-1981). */
static unsigned char pc1[56] = {
    56, 48, 40, 32, 24, 16, 8, 0, 57, 49, 41, 33, 25, 17,
    9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59, 51, 43, 35,
    62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37, 29, 21,
    13, 5, 60, 52, 44, 36, 28, 20, 12, 4, 27, 19, 11, 3};
static unsigned char totrot[16] = {
    1, 2, 4, 6, 8, 10, 12, 14, 15, 17, 19, 21, 23, 25, 27, 28};
static unsigned char pc2[48] = {
    13, 16, 10, 23, 0, 4, 2, 27, 14, 5, 20, 9,
    22, 18, 11, 3, 25, 7, 15, 6, 26, 19, 12, 1,
    40, 51, 30, 36, 46, 54, 29, 39, 50, 44, 32, 47,
    43, 48, 38, 55, 33, 52, 45, 41, 49, 35, 28, 31};
void deskey(unsigned char *key, short edf) /* Thanks to James Gillogly & Phil Karn!
*/
{
    int i, j, l, m, n;
    unsigned char pc1m[56], pcr[56];
    unsigned long kn[32];
    for (j = 0; j < 56; j++)
    {
        l = pc1[j];
        m = l & 07;
        pc1m[j] = (key[l >> 3] & bytebit[m]) ? 1 : 0;
    }
    for (i = 0; i < 16; i++)
    {
        if (edf == DE1)
            m = (15 - i) << 1;
    }
}

```

```

else
    m = i << 1;
    n = m + 1;
    kn[m] = kn[n] = 0L;
    for (j = 0; j < 28; j++)
    {
        l = j + totrot[i];
        if (l < 28)
            pcr[j] = pc1m[l];
        else
            pcr[j] = pc1m[l - 28];
    }
    for (j = 28; j < 56; j++)
    {
        l = j + totrot[i];
        if (l < 56)
            pcr[j] = pc1m[l];
        else
            pcr[j] = pc1m[l - 28];
    }
    for (j = 0; j < 24; j++)
    {
        if (pcr[pc2[j]])
            kn[m] |= bigbyte[j];
        if (pcr[pc2[j + 24]])
            kn[n] |= bigbyte[j];
    }
    }
    }
    cookey(kn);
    return;
}

static void cookey(unsigned long *raw1)
{
    unsigned long *cook, *raw0;
    unsigned long dough[32];
    int i;
    cook = dough;
    for (i = 0; i < 16; i++, raw1++)
    {
        raw0 = raw1++;
        *cook = (*raw0 & 0x00fc0000L) << 6;
        *cook |= (*raw0 & 0x00000fc0L) << 10;
        *cook |= (*raw1 & 0x00fc0000L) >> 10;
        *cook++ |= (*raw1 & 0x00000fc0L) >> 6;
        *cook = (*raw0 & 0x0003f000L) << 12;
        *cook |= (*raw0 & 0x0000003fL) << 16;
        *cook |= (*raw1 & 0x0003f000L) >> 4;
        *cook++ |= (*raw1 & 0x0000003fL);
    }
    usekey(dough);
    return;
}

void cpkey(unsigned long *into)
{
    unsigned long *from, *endp;
    from = KnL, endp = &KnL[32];
    while (from < endp)
        *into++ = *from++;
    return;
}

void usekey(unsigned long *from)
{
    unsigned long *to, *endp;
    to = KnL, endp = &KnL[32];
    while (to < endp)

```

```

        *to++ = *from++;
    return;
}
void des(unsigned char *inblock, unsigned char *outblock)
{
    unsigned long work[2];
    scrunch(inblock, work);
    desfunc(work, KnL);
    unscrunch(work, outblock);
    return;
}
static void scrunch(unsigned char *outof, unsigned long *into)
{
    *into = (*outof++ & 0xffL) << 24;
    *into |= (*outof++ & 0xffL) << 16;
    *into |= (*outof++ & 0xffL) << 8;
    *into++ |= (*outof++ & 0xffL);
    *into = (*outof++ & 0xffL) << 24;
    *into |= (*outof++ & 0xffL) << 16;
    *into |= (*outof++ & 0xffL) << 8;
    *into |= (*outof & 0xffL);
    return;
}
static void unscrunch(unsigned long *outof, unsigned char *into)
{
    *into++ = (*outof >> 24) & 0xffL;
    *into++ = (*outof >> 16) & 0xffL;
    *into++ = (*outof >> 8) & 0xffL;
    *into++ = *outof++ & 0xffL;
    *into++ = (*outof >> 24) & 0xffL;
    *into++ = (*outof >> 16) & 0xffL;
    *into++ = (*outof >> 8) & 0xffL;
    *into = *outof & 0xffL;
    return;
}
static unsigned long SP1[64] = {
    0x01010400L, 0x00000000L, 0x00010000L, 0x01010404L,
    0x01010004L, 0x00010404L, 0x00000004L, 0x00010000L,
    0x00000400L, 0x01010400L, 0x01010404L, 0x00000400L,
    0x01000404L, 0x01010004L, 0x01000000L, 0x00000004L,
    0x00000404L, 0x01000400L, 0x01000400L, 0x00010400L,
    0x00010400L, 0x01010000L, 0x01010000L, 0x01000404L,
    0x00010004L, 0x01000004L, 0x01000004L, 0x00010004L,
    0x00000000L, 0x00000404L, 0x00010404L, 0x01000000L,
    0x00010000L, 0x01010404L, 0x00000004L, 0x01010000L,
    0x01010400L, 0x01000000L, 0x01000000L, 0x00000400L,
    0x01010004L, 0x00010000L, 0x00010400L, 0x01000004L,
    0x00000400L, 0x00000004L, 0x01000404L, 0x00010404L,
    0x01010404L, 0x00010004L, 0x01010000L, 0x01000404L,
    0x01000004L, 0x00000404L, 0x00000404L, 0x01010400L,
    0x00000404L, 0x01000400L, 0x01000400L, 0x00000000L,
    0x00010004L, 0x00010400L, 0x00000000L, 0x01010004L};
static unsigned long SP2[64] = {
    0x80108020L, 0x80008000L, 0x00008000L, 0x00108020L,
    0x00100000L, 0x00000020L, 0x80100020L, 0x80008020L,
    0x80000020L, 0x80108020L, 0x80108000L, 0x80000000L,
    0x80008000L, 0x00100000L, 0x00000020L, 0x80100020L,
    0x00108000L, 0x00100020L, 0x80008020L, 0x00000000L,
    0x80000000L, 0x00008000L, 0x00108020L, 0x80100000L,
    0x00100020L, 0x80000020L, 0x00000000L, 0x00108000L,
    0x00008020L, 0x80108000L, 0x80100000L, 0x00008020L,
    0x00000000L, 0x00108020L, 0x80100020L, 0x00100000L,
    0x80008020L, 0x80100000L, 0x80108000L, 0x00008000L,
    0x80100000L, 0x80008000L, 0x00000020L, 0x80108020L,
    0x00108020L, 0x00000020L, 0x00008000L, 0x80000000L,

```

```

    0x00008020L, 0x80108000L, 0x00100000L, 0x80000020L,
    0x00100020L, 0x80008020L, 0x80000020L, 0x00100020L,
    0x00108000L, 0x00000000L, 0x80008000L, 0x00008020L,
    0x80000000L, 0x80100020L, 0x80108020L, 0x00108000L};
static unsigned long SP3[64] = {
    0x000000208L, 0x08020200L, 0x00000000L, 0x08020008L,
    0x08000200L, 0x00000000L, 0x00020208L, 0x08000200L,
    0x00020008L, 0x08000008L, 0x08000008L, 0x00020000L,
    0x08020208L, 0x00020008L, 0x08020000L, 0x00000208L,
    0x08000000L, 0x00000008L, 0x08020200L, 0x00000200L,
    0x00020200L, 0x08020000L, 0x08020008L, 0x00020208L,
    0x08000208L, 0x00020200L, 0x00020000L, 0x08000208L,
    0x00000008L, 0x08020208L, 0x00000200L, 0x08000000L,
    0x08020200L, 0x08000000L, 0x00020008L, 0x00000208L,
    0x00020000L, 0x08020200L, 0x08000200L, 0x00000000L,
    0x00000200L, 0x00020008L, 0x08020208L, 0x08000200L,
    0x08000008L, 0x00000200L, 0x00000000L, 0x08020008L,
    0x08000208L, 0x00020000L, 0x08000000L, 0x08020208L,
    0x00000008L, 0x00020208L, 0x00020200L, 0x08000008L,
    0x08020000L, 0x00000008L, 0x00000208L, 0x08020000L,
    0x00020008L, 0x00000008L, 0x08020008L, 0x00020200L};
static unsigned long SP4[64] = {
    0x00802001L, 0x00002081L, 0x00002081L, 0x00000080L,
    0x00802080L, 0x00800081L, 0x00800001L, 0x00002001L,
    0x00000000L, 0x00802000L, 0x00802000L, 0x00802081L,
    0x00000081L, 0x00000000L, 0x00800080L, 0x00800001L,
    0x00000001L, 0x00002000L, 0x00800000L, 0x00802001L,
    0x00000080L, 0x00800000L, 0x00002001L, 0x00002080L,
    0x00800081L, 0x00000001L, 0x00002080L, 0x00800080L,
    0x00002000L, 0x00802080L, 0x00802081L, 0x00000081L,
    0x00800080L, 0x00800001L, 0x00802000L, 0x00802081L,
    0x00000081L, 0x00000000L, 0x00000000L, 0x00802000L,
    0x00002080L, 0x00800080L, 0x00800081L, 0x00000001L,
    0x00802001L, 0x00002081L, 0x00002081L, 0x00000080L,
    0x0080001L, 0x00000081L, 0x00000001L, 0x00002000L,
    0x00800001L, 0x00002001L, 0x00802080L, 0x00800081L,
    0x00002001L, 0x00002080L, 0x00800000L, 0x00802001L,
    0x00000080L, 0x00800000L, 0x00002000L, 0x00802080L};
static unsigned long SP5[64] = {
    0x00000100L, 0x02080100L, 0x02080000L, 0x42000100L,
    0x00080000L, 0x00000100L, 0x40000000L, 0x02080000L,
    0x40080100L, 0x00080000L, 0x02000100L, 0x40080100L,
    0x42000100L, 0x42080000L, 0x00080100L, 0x40000000L,
    0x02000000L, 0x40080000L, 0x40080000L, 0x00000000L,
    0x40000100L, 0x42080100L, 0x42080100L, 0x02000100L,
    0x42080000L, 0x40000100L, 0x00000000L, 0x42000000L,
    0x02080100L, 0x02000000L, 0x42000000L, 0x00080100L,
    0x00080000L, 0x42000100L, 0x00000100L, 0x02000000L,
    0x40000000L, 0x02080000L, 0x42000100L, 0x40080100L,
    0x02000100L, 0x40000000L, 0x02000000L, 0x42080000L,
    0x40080100L, 0x00080100L, 0x42000000L, 0x42080100L,
    0x02080000L, 0x00000000L, 0x40080000L, 0x42000000L,
    0x00080100L, 0x02000100L, 0x40000100L, 0x00080000L,
    0x00000000L, 0x40080000L, 0x02080100L, 0x40000100L};
static unsigned long SP6[64] = {
    0x20000010L, 0x20400000L, 0x00004000L, 0x20404010L,
    0x20400000L, 0x00000010L, 0x20404010L, 0x00400000L,
    0x20004000L, 0x00404010L, 0x00400000L, 0x20000010L,
    0x00400010L, 0x20004000L, 0x20000000L, 0x00004010L,
    0x00000000L, 0x00400010L, 0x20004010L, 0x00004000L,
    0x00404000L, 0x20004010L, 0x00000010L, 0x20400010L,
    0x20400010L, 0x00000000L, 0x00404010L, 0x20404000L,
    0x00004010L, 0x00404000L, 0x20404000L, 0x20000000L,
    0x20004000L, 0x00000010L, 0x20400010L, 0x00404000L,

```

```

    0x20404010L, 0x00400000L, 0x00004010L, 0x20000010L,
    0x00400000L, 0x20004000L, 0x20000000L, 0x00004010L,
    0x20000010L, 0x20404010L, 0x00404000L, 0x20400000L,
    0x00404010L, 0x20404000L, 0x00000000L, 0x20400010L,
    0x00000010L, 0x00004000L, 0x20400000L, 0x00404010L,
    0x00004000L, 0x00400010L, 0x20004010L, 0x00000000L,
    0x20404000L, 0x20000000L, 0x00400010L, 0x20004010L};
static unsigned long SP7[64] = {
    0x00200000L, 0x04200002L, 0x04000802L, 0x00000000L,
    0x00000800L, 0x04000802L, 0x00200802L, 0x04200800L,
    0x04200802L, 0x00200000L, 0x00000000L, 0x04000002L,
    0x00000002L, 0x04000000L, 0x04200002L, 0x00000802L,
    0x04000800L, 0x00200802L, 0x00200002L, 0x04000800L,
    0x04000002L, 0x04200000L, 0x04200800L, 0x00200002L,
    0x04200000L, 0x00000800L, 0x00000802L, 0x04200802L,
    0x00200800L, 0x00000002L, 0x04000000L, 0x00200800L,
    0x04000000L, 0x00200800L, 0x00200000L, 0x04000802L,
    0x04000802L, 0x04200002L, 0x04200002L, 0x00000002L,
    0x00200002L, 0x04000000L, 0x04000800L, 0x00200000L,
    0x04200800L, 0x00000802L, 0x00200802L, 0x04200800L,
    0x00000800L, 0x00000000L, 0x00000002L, 0x04200802L,
    0x00000000L, 0x00200802L, 0x04200000L, 0x00000800L,
    0x04000002L, 0x04000800L, 0x00000800L, 0x00200002L};
static unsigned long SP8[64] = {
    0x10001040L, 0x00001000L, 0x00040000L, 0x10041040L,
    0x10000000L, 0x10001040L, 0x00000040L, 0x10000000L,
    0x00040040L, 0x10040000L, 0x10041040L, 0x00041000L,
    0x10041000L, 0x00041040L, 0x00001000L, 0x00000040L,
    0x10040000L, 0x10000040L, 0x10001000L, 0x00001040L,
    0x00041000L, 0x00040040L, 0x10040040L, 0x10041000L,
    0x00001040L, 0x00000000L, 0x00000000L, 0x10040040L,
    0x10000040L, 0x10001000L, 0x00041040L, 0x00040000L,
    0x00041040L, 0x00040000L, 0x10041000L, 0x00001000L,
    0x00000040L, 0x10040040L, 0x00000040L, 0x10040000L,
    0x10040040L, 0x10000000L, 0x00040000L, 0x10001040L,
    0x00000000L, 0x10041040L, 0x00040040L, 0x10000040L,
    0x10040000L, 0x10001000L, 0x10001040L, 0x00000000L,
    0x10041040L, 0x00041000L, 0x00041000L, 0x00001040L,
    0x00001040L, 0x00040040L, 0x10000000L, 0x10041000L};
static void desfunc(unsigned long *block, unsigned long *keys)
{
    unsigned long fval, work, right, leftt;
    int round;
    leftt = block[0];
    right = block[1];
    work = ((leftt >> 4) ^ right) & 0xf0f0f0fL;
    right ^= work;
    leftt ^= (work << 4);
    work = ((leftt >> 16) ^ right) & 0x0000ffffL;
    right ^= work;
    leftt ^= (work << 16);
    work = ((right >> 2) ^ leftt) & 0x33333333L;
    leftt ^= work;
    right ^= (work << 2);
    work = ((right >> 8) ^ leftt) & 0x00ff00ffL;
    leftt ^= work;
    right ^= (work << 8);
    right = ((right << 1) | ((right >> 31) & 1L)) & 0xffffffffL;
    work = (leftt ^ right) & 0xaaaaaaaaL;
    leftt ^= work;
    right ^= work;
    leftt = ((leftt << 1) | ((leftt >> 31) & 1L)) & 0xffffffffL;
    for (round = 0; round < 8; round++)

```

```

{
    work = (right << 28) | (right >> 4);
    work ^= *keys++;
    fval = SP7[work & 0x3fL];
    fval |= SP5[(work >> 8) & 0x3fL];
    fval |= SP3[(work >> 16) & 0x3fL];
    fval |= SP1[(work >> 24) & 0x3fL];
    work = right ^ *keys++;
    fval |= SP8[work & 0x3fL];
    fval |= SP6[(work >> 8) & 0x3fL];
    fval |= SP4[(work >> 16) & 0x3fL];
    fval |= SP2[(work >> 24) & 0x3fL];
    leftt ^= fval;
    work = (leftt << 28) | (leftt >> 4);
    work ^= *keys++;
    fval = SP7[work & 0x3fL];
    fval |= SP5[(work >> 8) & 0x3fL];
    fval |= SP3[(work >> 16) & 0x3fL];
    fval |= SP1[(work >> 24) & 0x3fL];
    work = leftt ^ *keys++;
    fval |= SP8[work & 0x3fL];
    fval |= SP6[(work >> 8) & 0x3fL];
    fval |= SP4[(work >> 16) & 0x3fL];
    fval |= SP2[(work >> 24) & 0x3fL];
    right ^= fval;
}
right = (right << 31) | (right >> 1);
work = (leftt ^ right) & 0xaaaaaaaaL;
leftt ^= work;
right ^= work;
leftt = (leftt << 31) | (leftt >> 1);
work = ((leftt >> 8) ^ right) & 0x00ff00ffL;
right ^= work;
leftt ^= (work << 8);
work = ((leftt >> 2) ^ right) & 0x33333333L;
right ^= work;
leftt ^= (work << 2);
work = ((right >> 16) ^ leftt) & 0x0000ffffL;
leftt ^= work;
right ^= (work << 16);
work = ((right >> 4) ^ leftt) & 0x0f0f0f0fL;
leftt ^= work;
right ^= (work << 4);
*block++ = right;
*block = leftt;
return;
}
/* Validation sets:
*
* Single-length key, single-length plaintext -
* Key : 0123 4567 89ab cdef
* Plain : 0123 4567 89ab cde7
* Cipher : c957 4425 6a5e d31d
*
*****/
void des_key(des_ctx *dc, unsigned char *key)
{
    deskey(key, EN0);
    cpkey(dc->ek);
    deskey(key, DE1);
    cpkey(dc->dk);
}
/* Encrypt several blocks in ECB mode. Caller is responsible for
short blocks. */
void des_enc(des_ctx *dc, unsigned char *data, int blocks)

```

```

{
    unsigned long work[2];
    int i;
    unsigned char *cp;
    cp = data;
    for (i = 0; i < blocks; i++)
    {
        scrunch(cp, work);
        desfunc(work, dc->ek);
        unscrunch(work, cp);
        cp += 8;
    }
}

void des_dec(des_ctx *dc, unsigned char *data, int blocks)
{
    unsigned long work[2];
    int i;
    unsigned char *cp;
    cp = data;
    for (i = 0; i < blocks; i++)
    {
        scrunch(cp, work);
        desfunc(work, dc->dk);
        unscrunch(work, cp);
        cp += 8;
    }
}

```

Исходный код файла main.cpp:

```

#include <stdio.h>
#include "des.cpp"

void demo_DES_crypto() {
    des_ctx dc;
    int i;
    unsigned long data[10];
    unsigned char *cp, key[8] = {0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef};
    unsigned char x[8] = {0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xe7};
    cp = x;
    des_key(&dc, key);
    des_enc(&dc, cp, 1);
    printf("Enc(0..7,0..7) = ");
    for(i=0; i<8; i++) printf("%02x ", ((unsigned int) cp[i])&0x00ff);
    printf("\n");
    des_dec(&dc, cp, 1);
    printf("Dec(above,0..7) = ");
    for(i=0; i<8; i++) printf("%02x ", ((unsigned int) cp[i])&0x00ff);
    printf("\n");
    cp = (unsigned char *) data;
    for(i=0; i<10; i++) data[i]=i;
    des_enc(&dc, cp, 5); /* Enc 5 blocks. */
    for(i=0; i<10; i+=2)
        printf("Block %01d = %08lx %08lx.\n", i/2, data[i], data[i+1]);
    des_dec(&dc, cp, 1);
    des_dec(&dc, cp+8, 4);
    for(i=0; i<10; i+=2)
        printf("Block %01d = %08lx %08lx.\n", i/2, data[i], data[i+1]);
}

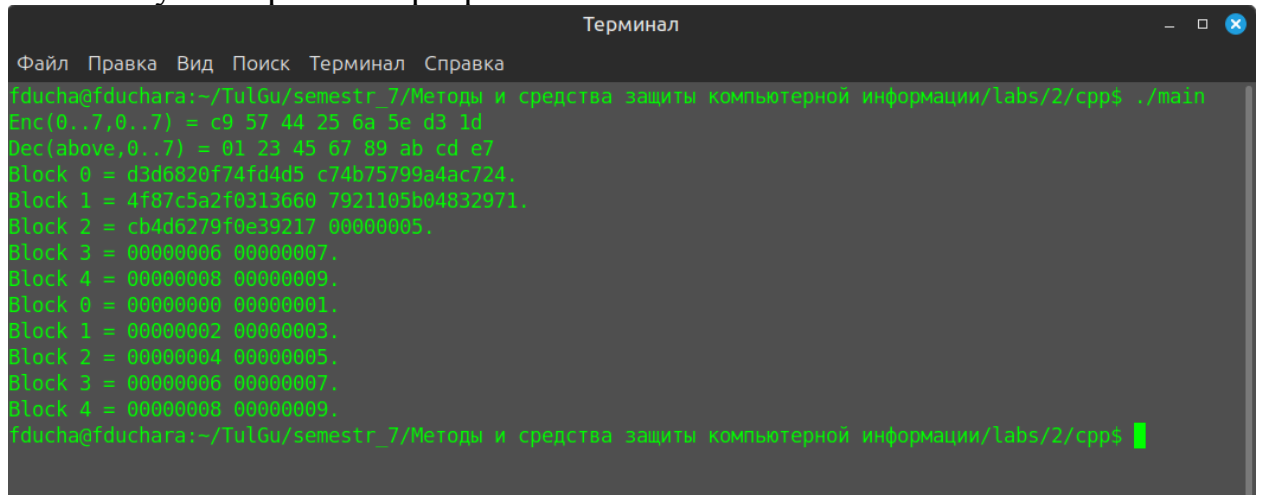
int main() {
    demo_DES_crypto();

    return 0;
}

```

В задание не входит объяснение работы программы, но для понимания, что мы здесь не шурум-бурум магией занимаемся, а криптографией, в 1-ой части демонстрации шифруется строка символов `unsigned char x[8] = {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef}` при помощи ключа `unsigned char key[8] = {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef}`, а во 2-ой части массив чисел от 0 до 9 при помощи того же ключа.

Результат работы программы:



```
Терминал
Файл Правка Вид Поиск Терминал Справка
fduchara@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$ ./main
Enc(0..7,0..7) = c9 57 44 25 6a 5e d3 1d
Dec(above,0..7) = 01 23 45 67 89 ab cd ef
Block 0 = d3d6820f74fd4d5 c74b75799a4ac724.
Block 1 = 4f87c5a2f0313660 7921105b04832971.
Block 2 = cb4d6279f0e39217 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
Block 0 = 00000000 00000001.
Block 1 = 00000002 00000003.
Block 2 = 00000004 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
fduchara@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$
```

Как видно, расшифрованные строка символов и массив соответствуют исходным.

2. При запуске данной программы использовать парольную аутентификацию. В качестве пароля для запуска программы использовать слово длиной до 16 символов. В файл паролей записать результат выполнения функции (в соответствии с вариантом) над исходным паролем. Включить в программу следующее условие: после трех неверных попыток введения пароля закрыть выполняемую программу и зафиксировать попытку входа в систему в журнале безопасности (текстовый файл, в котором фиксируется для неудачной попытки дата и время). При этом номер вариант соответствует последней цифре номера зачетки.

Номер зачетки — 220085: решаем вариант № 5.

Функция $f : q \cdot (X_1)^n + g \cdot (X_2)^n$, где:

где n — количество вводимых символов;

q, g — случайное число в интервале от 1 до 100. В нашем случае генератор выбрал $q = 98, g = 21$;

X_1 — левая половина пароля (сумма ASCII-кодов всех символов в левой части);

X_2 — правая половина пароля (сумма ASCII-кодов всех символов в правой части).

Значение заданной функции от правильного пароля будем хранить в файле `rwd`, и это будет единственное значение в этом файле.

При каждом запуске программа проверяет наличие файла `rwd` и при его отсутствии просит задать пароль. При успешной установке пароля создается файл `rwd`, содержащий значение нашей функции от нового пароля.

В случае существования файла паролей, программа просит ввести пароль. Из введенного пароля вычисляется значение функции и сравнивается со значением из файла. Если значения совпадают, то выполняется функция `demo_DES_crypto()` из 1-го задания. Если значения не совпадают, происходит запись в файл `my_logs.log` даты и времени неудачной попытки входа. Если количество неудачных попыток превышает 3, то программа завершает выполнение.

Расчет правильной работы заданной функции проведем на примере пароля «tulgu». Данный пароль содержит 5 ASCII — символов с номерами 116, 117, 108, 103, 117. В коде вычисления функции для определения длины левой и правой частей пароля используется операция $n / 2$, где n — длина пароля, поэтому при нечетной длине пароля левая часть будет содержать на 1 символ меньше, чем правая. Числа q и g заданы в коде, так как при генерации их как случайных чисел, значение функции от пароля каждый раз будет различным. При вычислении значения функции получаем:

$$98 \times (116 + 117)^5 + 21 \times (108 + 103 + 117)^5 = 98 \times 233^5 + 21 \times 328^5 = 98 \times 686719856393 + 21 \times 3796375994368 = 147022441808242$$

Для работы программы созданы следующие функции:

`string getPassword()` - функция возвращает значение введенного пароля из консоли, заменяя его отображение на * при вводе;

`unsigned long long my_hash_func(const string &p)` - функция вычисляет и возвращает значение заданной функции f от строки p ;

`unsigned long long get_my_hash_from_file()` - функция читает из файла `pwd` значение хеша и возвращает его. В случае отсутствия файла или ошибки доступа возвращает магическое число 777;

`void log_wrong_attempt()` - функция записывает в файл `my_logs.log` дату и время. Вызывается при вводе не верного пароля;

`bool set_password()` - функция записывает в файл `pwd` значение функции f от нового пароля, то есть устанавливает новый пароль. Возвращает `true` при удачной установке. Установка может пройти не удачно при не соответствии нового пароля и его подтверждения или ошибке доступа к файлу `pwd`.

Исходный код функций:

```
#include <stdio.h>
#include <iostream>
#include <cmath>
#include <termios.h>
#include <unistd.h>
#include <fstream>
#include <cstdlib>
#include <chrono>
#include <ctime>
#include "des.cpp"

using namespace std;
using std::ofstream;
using std::ifstream;
using std::exit;
string getPassword() {
    string password;
```

```

    termios oldt;
    tcgetattr(STDIN_FILENO, &oldt);
    termios newt = oldt;
    newt.c_lflag &= ~ECHO;
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);

    char ch;
    while ((ch = getchar()) != '\n') {
        if (ch == 127) {
            // Handle backspace
            if (!password.empty()) {
                std::cout << "\b \b";
                password.pop_back();
            }
        } else {
            password += ch;
            cout << '*';
        }
    }
    cout << '\n';

    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    return password;
}

unsigned long long my_hash_func(const string &p) {
    short n = p.size();
    if (n > 16) {
        n = 16;
        cout << "Only the first 16 characters will be used." << endl;
    }
    short q = 98;
    short g = 21;
    short m = n / 2;
    int lsum = 0;
    for (size_t i = 0; i < m; i++) {
        lsum += (int) p[i];
    }
    int rsum = 0;
    for (size_t i = m; i < n; i++) {
        rsum += (int) p[i];
    }

    /*
    example password - tulgu => ascii is 116, 117, 108, 103, 117
    left side - 2 chars, sum = 233
    right side - 3 chars, sum = 328
    233^5 = 686719856393
    328^5 = 3796375994368
    result = 98 * 686719856393 + 21 * 3796375994368 = 147022441808242
    */

    return q * powf64x(lsum, n) + g * powf64x(rsum, n);
}

unsigned long long get_my_hash_from_file() {
    ifstream in("pwd", ios::in);
    if (!in) return 777;
    unsigned long long hash;
    in >> hash;
    in.close();
    return hash;
}

void log_wrong_attempt() {
    ofstream out("my_logs.log", ios::app);

```

```

    if (!out) {
        cerr << "Log file is not opened" << endl;
        exit(1);
    }
    auto t = chrono::system_clock::now();
    std::time_t tt = chrono::system_clock::to_time_t(t);
    out << ctime(&tt);
    out.close();
}
bool set_password() {
    cout << "Enter new password: ";
    auto p1 = getPassword();
    cout << "Confirm password: ";
    auto p2 = getPassword();
    if (p1 != p2) {
        cout << "Passwords are different." << endl;
        return false;
    }
    ofstream out("pwd", ios::out);
    if (!out) {
        cerr << "Password file is not opened" << endl;
        return false;
    }
    out << my_hash_func(p1);
    out.close();
    return true;
}

```

Рассмотрим код функции main():

```

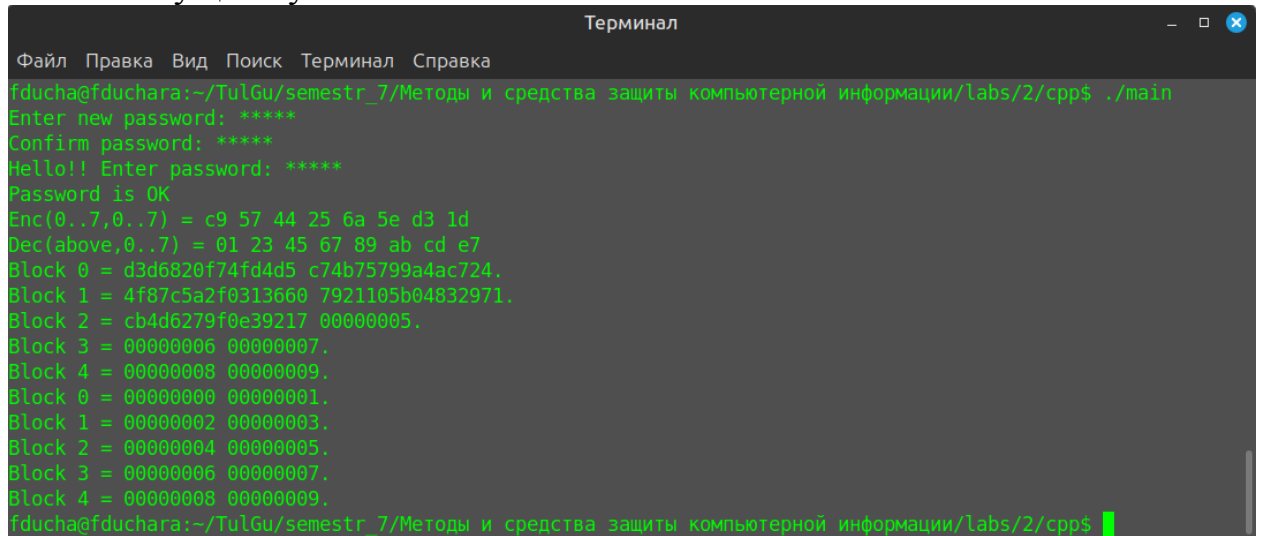
int main() {
    if (get_my_hash_from_file() == 777) {
        while (!set_password()) {}
    }
    short attempts = 3;
    cout << "Hello!! Enter password: ";
    while (attempts > 0) {
        string pswd = getPassword();
        auto right_hash = get_my_hash_from_file();
        if (my_hash_func(pswd) == right_hash) {
            cout << "Password is OK" << endl;
            demo_DES_crypto();
            return 0;
        } else {
            log_wrong_attempt();
            cout << "Password is wrong!! Enter password again: ";
            attempts--;
        }
    }
    if (attempts == 0) {
        cout << "App is terminating ... " << endl;
    }
    return 0;
}

```

В коде происходит вот что: читаем значение из файла пароля; если файл не существует, устанавливаем пароль до талого. Устанавливаем количество попыток ввода пароля в 3 и выводим приглашение на ввод пароля. Пока попытки не закончились, читаем введенный пароль из звездочек и сравниваем его с правильным значением. Если пароль верный выводим демонстрацию работы программы из 1-го задания и завершаем программу. Если не верный — пишем в лог дату-время и пробуем еще пару

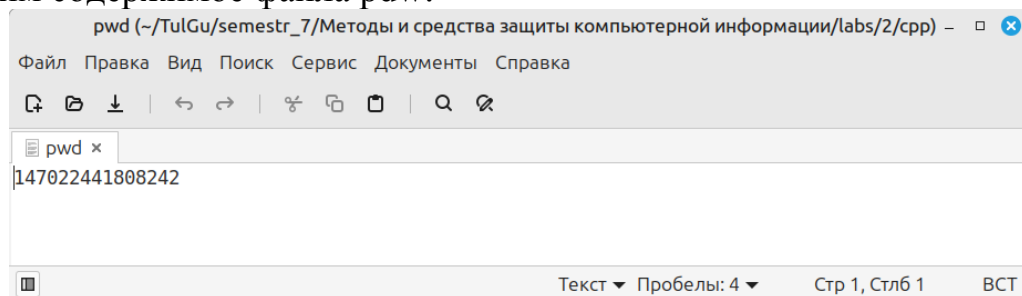
раз. Когда попытки закончатся, выводим «App is terminating ... » и завершаем программу.

Продemonстрируем работу программы: удаляем файлы лога и пароля, если они существуют.

A terminal window titled "Терминал" with a menu bar (Файл, Правка, Вид, Поиск, Терминал, Справка). The prompt is fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp\$. The user runs ./main. The program prompts for a new password (****), confirms it (****), and says "Hello!! Enter password: ****". The user enters "tulgu". The program outputs "Password is OK" and shows encryption/decryption results: Enc(0..7,0..7) = c9 57 44 25 6a 5e d3 1d, Dec(above,0..7) = 01 23 45 67 89 ab cd e7. It then lists five blocks of data, each with a hex string and a decimal value. The prompt returns to fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp\$.

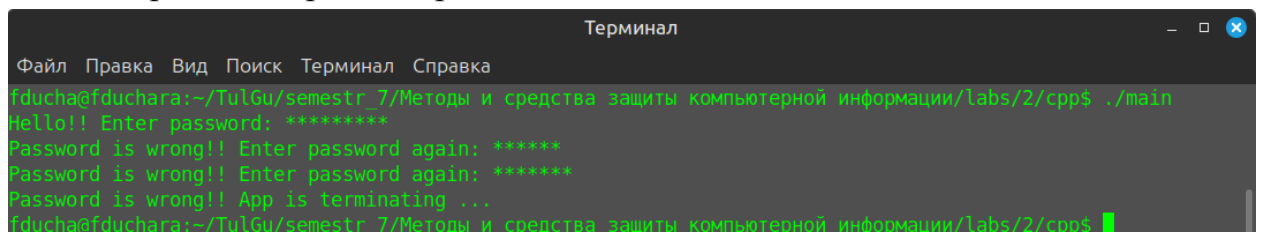
```
fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$ ./main
Enter new password: ****
Confirm password: ****
Hello!! Enter password: ****
Password is OK
Enc(0..7,0..7) = c9 57 44 25 6a 5e d3 1d
Dec(above,0..7) = 01 23 45 67 89 ab cd e7
Block 0 = d3d6820f74fd4d5 c74b75799a4ac724.
Block 1 = 4f87c5a2f0313660 7921105b04832971.
Block 2 = cb4d6279f0e39217 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
Block 0 = 00000000 00000001.
Block 1 = 00000002 00000003.
Block 2 = 00000004 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$
```

Вводим новый пароль «tulgu», подтверждаем, входим и готово! Смотрим содержимое файла pdw:

A file editor window titled "pwd (~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp) -". The menu bar includes "Файл, Правка, Вид, Поиск, Сервис, Документы, Справка". The file content is "147022441808242". The status bar at the bottom shows "Текст", "Пробелы: 4", "Стр 1, Стлб 1", and "ВСТ".

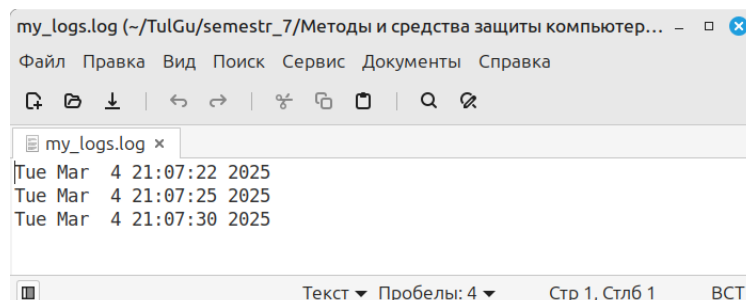
```
pwd (~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp) -
147022441808242
```

Видим, что значение в файле соответствует расчетному. Попробуем ввести 3 раза не верный пароль:

A terminal window titled "Терминал" with the same menu bar as before. The prompt is fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp\$. The user runs ./main. The program prompts for a password (*****). The user enters "tulgu". The program outputs "Password is wrong!! Enter password again: *****". The user enters "tulgu". The program outputs "Password is wrong!! Enter password again: *****". The user enters "tulgu". The program outputs "Password is wrong!! App is terminating ...". The prompt returns to fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp\$.

```
fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$ ./main
Hello!! Enter password: *****
Password is wrong!! Enter password again: *****
Password is wrong!! Enter password again: *****
Password is wrong!! App is terminating ...
fduchag@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$
```

Посмотрим содержимое лога в файле my_logs.log:

A file editor window titled "my_logs.log (~/TulGu/semestr_7/Методы и средства защиты компьютер... -". The menu bar includes "Файл, Правка, Вид, Поиск, Сервис, Документы, Справка". The file content shows three timestamps: "Tue Mar 4 21:07:22 2025", "Tue Mar 4 21:07:25 2025", and "Tue Mar 4 21:07:30 2025". The status bar at the bottom shows "Текст", "Пробелы: 4", "Стр 1, Стлб 1", and "ВСТ".

```
my_logs.log (~/TulGu/semestr_7/Методы и средства защиты компьютер... -
Tue Mar 4 21:07:22 2025
Tue Mar 4 21:07:25 2025
Tue Mar 4 21:07:30 2025
```

Видим 3 попытки с интервалом в несколько секунд. Попробуем ввести пару раз не верный пароль, а потом правильный:

```
Терминал
Файл Правка Вид Поиск Терминал Справка
fducha@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$ ./main
Hello!! Enter password: *****
Password is wrong!! Enter password again: *****
Password is wrong!! Enter password again: *****
Password is OK
Enc(0..7,0..7) = c9 57 44 25 6a 5e d3 1d
Dec(above,0..7) = 01 23 45 67 89 ab cd e7
Block 0 = d3d6820f74fd4d5 c74b75799a4ac724.
Block 1 = 4f87c5a2f0313660 7921105b04832971.
Block 2 = cb4d6279f0e39217 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
Block 0 = 00000000 00000001.
Block 1 = 00000002 00000003.
Block 2 = 00000004 00000005.
Block 3 = 00000006 00000007.
Block 4 = 00000008 00000009.
fducha@fduchara:~/TulGu/semestr_7/Методы и средства защиты компьютерной информации/labs/2/cpp$
```

Как видим, в лог добавились еще 2 записи:

```
my_logs.log (~/TulGu/semestr_7/Методы и средства защиты компьютерно...
Файл Правка Вид Поиск Сервис Документы Справка
🔍 📄 ⬇️ | ⬅️ ➡️ | 🗑️ 📄 | 🔍 🗑️
my_logs.log x
Tue Mar 4 21:07:22 2025
Tue Mar 4 21:07:25 2025
Tue Mar 4 21:07:30 2025
Tue Mar 4 21:10:59 2025
Tue Mar 4 21:11:01 2025
Текст ▾ Пробелы: 4 ▾ Стр 4, Стлб 25 ВСТ
```

Контрольные вопросы.

1. Что такое идентификация и аутентификация?

Идентификация – одна из функций подсистемы защиты. Эта функция выполняется в первую очередь, когда объект делает попытку войти в сеть. Если процедура идентификации завершается успешно, данный объект считается законным для данной сети.

Аутентификация объекта – проверка подлинности объекта. Эта процедура устанавливает, является ли данный объект именно таким, каким он себя объявляет.

2. Что можно использовать для подтверждения подлинности субъекта?

- predetermined information, находящуюся в распоряжении пользователя: пароль, персональный идентификационный номер, соглашение об использовании специальных закодированных фраз;

- аппаратные элементы обеспечения, находящиеся в распоряжении пользователя: ключи, магнитные карточки, микросхемы и т.п.;

- характерные личные особенности пользователя: отпечатки пальцев, рисунок сетчатки глаза, тембр голоса и т. п.;

- характерные приемы и черты поведения пользователя в режиме реального времени: особенности динамики и стиль работы на клавиатуре, приемы работы с манипулятором и т. п.;

- навыки и знания пользователя, обусловленные образованием, культурой, обучением, воспитанием, привычками и т. п.;

3. Какими свойствами должна обладать хэш-функция?

- хэш-функция должна быть чувствительна к всевозможным изменениям в тексте, таким как вставки, выбросы, перестановки и т.п.;

- хэш-функция должна обладать свойством необратимости, то есть задача подбора документа, который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;

- вероятность того, что значения хэш-функций двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала.

4. Нужно ли использовать алгоритмы шифрования для файла с паролями?

Да, использование алгоритмов шифрования для файла с паролями необходимо для обеспечения безопасности данных.

Задача шифрования — превратить данные, которые могут прочитать все, в данные, которые может прочитать только тот, у кого есть секретная часть (ключ безопасности, сертификат, пароль или расшифровочная матрица).

5. Можно ли, используя пароль, полностью обезопасить информационную систему от несанкционированного доступа?

Использование пароля является важным элементом защиты информационной системы, но одного только пароля недостаточно для полной защиты от несанкционированного доступа. Пароль — это лишь один

из слоев безопасности, и его эффективность зависит от множества факторов, таких как сложность пароля, его хранение и управление, а также наличие дополнительных мер защиты.

6. Каким должен быть пароль, чтобы возможность его взлома была как можно меньше?

Чтобы минимизировать вероятность взлома пароля, он должен быть сложным, уникальным и длинным. Длина пароля – минимум 12–15 символов. Сложность пароля: комбинация из заглавных букв (A–Z), строчных букв (a–z), цифр (0–9), специальных символов (!, @, #, \$, %, &, * и т.д.). Каждый пароль должен быть уникальным. Необходимо избегать очевидных комбинаций в пароле (password, qwerty, 123456). Необходимо менять пароль каждые 3 – 6 месяцев. Возможно использовать генератор паролей. Проверять надежность пароля на соответствующих сервисах.