Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Кафедра «Компьютерная безопасность»

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине

«Языки программирования»

Работу выполнила
студентка группы
СКБ-232 _____ Д.В. Иванова
подпись, дата

Работу проверил _____ С.А. Булгаков
подпись, дата

Москва 2023

Содержание

1 Постановка задачи

Необходимо было улучшить программу из Лабораторной работы №3 следующим образом:

- если входной файл содержит сигнатуру 0xEECC (либо 0xCCEE для случая big-endian), то параметр сдвига считывается из файла, а также выполняется проверка контрольной суммы;

- если при запуске программы ей передается параметр командной строки -, то следует выводить только результат, в противном случае вывод осуществляется в соответствии с определённым форматом(2 байта - сигнатура,1 байт - сдвиг, 4 байта - размер(size), size - данные, 4 байта - CRC-32);

- если при запуске программы ей передается параметр командной строки b, то следует использовать порядок байт big-endian.

Поскольку остальные улучшения программы были оставлены на усмотрение исполнителя, то я сделала следующие изменения:

1. Ввела переменные типа int: shift, ind_f, ind_d,ind_c, ind_k, ind_b, ind_t, ind_sign, ex, argccc для лучшего и упрощённого контроля над возможными вариантами ввода

2. Написала функцию rev, которая в качестве аргументов принимает и возвращает значения типа unsigned long.

3. Адаптировала для использования код из предыдущей работы, считающий crc32 для файла и оформила его в функцию crc32, которая принимает в качестве аргументов указатель на переменную типа unsigned char(начало массива из элементов типа unsigned char с данными для подсчёта контрольной суммы) и переменную типа unsigned long и возвращает значение unsigned long.

4. Написала функцию message_encryption_to_str_file, которая принимает в качестве аргумента значение сдвига и совершает шифрование сообщения и вывод результата в файл формата, описанного в задании. Тип функции - void.

5. Написала функцию enc_strange_file_to_output, которая принимает в качестве аргумента ссылку на начало файла, который имеет формат, описанный в задании и шифрует этот файл, записывая результат в обычный файл.

6. Написала функцию file_encryption_to_str_file, которая принимает в качестве аргументов указатель на начало файла с сообщением и сдвиг, после чего зашифровывает сообщение и записывает результат в файл с форматом, описанном в задании.

7. Написала функцию enc_strange_file_to_strange_file, которая принимает в качестве аргументов ссылку на начало файла с текстом для шифрования с форматом из задания, шифрует и записывает результат в файл, с форматом, описанном в задании.

8. Написала функцию decrypt_input_to_str_file, которая принимает в качестве аргументов сдвиг и расшифровывает пользовательский ввод с дефолтным ключом 5, записывая результат в файл с форматом, описанном в задании

9. Написала функцию decrypt_file_to_str_file, которая принимает в качестве аргумента сдвиг и ссылку на начало файла с сообщением в памяти, расшивровывает сообщение и выводит в виде файла с форматом, указанном в задании.

10. Написала функцию dec_strange_file_to_strange_file, которая принимает в качестве аргументов ссылку на начало файла с сообщением в описанном в задании формате, расшифровывает его и записывает в описанном задании формате в файл.

11. Написала функцию dec_strange_file_to_file, которая принимает в качестве аргументов ссылку на начало файла с сообщением, расшифровывает его и записывает результат в обычный файл.

12. Написала функцию troubleshooting_input, которая принимает в качестве аргументов ссылку на начало файла с сообщением и ссылку на начало файла, куда надо записать зашифрованное сообщение и проверяет, не NULL ли значения соответсвующие ссылкам на начала файлов

13. Написала функцию check_for_sign, которая принимает в качестве аргумента ссылку на начало в памяти файла с сообщением и проверяет его на наличие сигнатуры.

2 Описание функции main

2.1 Ввод и работа с аргументами

Назовём аргументы ‘c‘, ‘d‘, file, file_with_shift – "весомыми".

Если пользователь ввёл в качестве аргументов ‘-‘(аргумент ‘b‘ на на что не влияет):

- Если пользователь не вводил аргументов или вводил только ключ c, то после запуска программу пользователю предлагается ввести сообщение, которое он хочет закодировать и сдвиг, на который сдвинется алфавит вперёд.
- Если пользователь ввёл название файла и такой файл существует, то шифруемый текст считывается из файла, сдвиг пользователь вводит в ответ на запрос программы, после чего создаётся файл enc.txt в который записывается зашифрованное сообщение. В случае, когда пользователь ввёл название существующего файла и ключ c, действия программы аналогичны.
- Если пользователь ввёл только сдвиг в качестве аргумента, то шифруемое сообщение пользователю предложат ввести после слов ‘Input message:‘. Также в случае сдвига и ключа c в качестве аргументов
- Если пользователь в качестве аргументов ввёл и имя файла и число, на которое должен быть совершён сдвиг, при том не важно в каком порядке, то программа создаст файл enc.txt и запишет в него зашифрованное сообщение. Также пользователь, в дополнение к этим аргументам мог ввести c, действия программы бы не изменились.
- Если пользователь в качестве аргумента ввёл d, то запускается функция decrypt_input()
- Если пользователь в качестве аргументов ввёл d и название существующего файла, то запускается функция decrypt_input
- Если пользователь в качестве аргументов ввёл d, название существующего файла и сдвиг, то запускается функция decrypt_file_with_shift с передаваемым ей сдвигом shift
- Если пользователь ввёл в качестве аргумента файл с сигнатурой, то сообщение и сдвиг считываются из файла и результат пишется в текстовый файл encoded.txt.

Если пользователь не ввёл в качестве аргументов ‘-‘:

- Если пользователь не вводил аргументов или вводил только ключ c, то запускается функция message_encryption_to_str_file, которая просит вести сообщение и, если сдвиг не был введён, то используется сдвиг 5. Результат пишется в hex_file_encoded.txt
- Если пользователь ввёл название файла и такой файл существует, но в нём нет сигнатуры, то шифруемый текст считывается из файла, сдвиг пользователь вводит либо как аргумент, либо по дефолту используется сдвиг 5, после чего создаётся файл hex_file_encoded.txt в который записывается зашифрованное сообщение, по формату из задания. В случае, когда пользователь ввёл название существующего файла и ключ c, действия программы аналогичны. Если был ключ ‘b‘, то вывод будет адаптирован под систему с big endian.

- Если пользователь ввёл название файла и такой файл существует и в нём есть сигнатура, то шифруемый текст считывается из файла, как и сдвиг, после чего создаётся файл hex_file_encoded.txt в который записывается зашифрованное сообщение, по формату из задания. В случае, когда пользователь ввёл название существующего файла и ключ c, действия программы аналогичны.

- Если пользователь ввёл только сдвиг в качестве аргумента, то шифруемое сообщение пользователю предложат ввести после слов ‘Input message:‘. Результат выводится по формату в файл hex_file_encoded.txt. Также в случае сдвига и ключа c в качестве аргументов. Если был ключ ‘b‘, то вывод будет адаптирован под систему с big endian.

- Если пользователь ввёл только ‘d‘, то пользователю предложат ввести текст для расшифрования, сдвиг будет взят либо из аргументов, либо использован сдвиг 5. Если был ключ ‘b‘, то вывод будет адаптирован под систему с big endian.

2.2 Описание функции Caesar

Функция Caesar реализует агоритм шифрования Цезаря определённого элемента. Алгоритм шифрования Цезаря проверяет вводимый символ на то, заглавная это буква или строчная. После определения, ищется место этой буквы в алфавите, прибавляется сдвиг и проверяется, что не случилось выхода из алфавита, после чего в алфавите ищется буква с новым номером и запоминается как буква зашифрованного слова. Таким образом, шифруется каждая буква вводимого сообщения и получается зашифрованное сообщение.

2.3 Описание функции is_number

Функция получает на вход ссылку на элемент char* str. Дальше в цикле, пока не встречен символ перехода на следующую строчку или символ окончания строки, проверяется является ли данный символ цифрой. Если проверка не пройдена - функция возвращает 0, иначе, если за время работы цикла программа не выведет 0, то будет выведена 1.

2.4 Описание функции message_encryption

Функция принимает в качестве аргументов сдвиг int shift. Просит пользвателя ввести сообщение, которое надо зашифровать. После чего в цикле, выводит сначала в начале строку "Input message: после чего выводит зашифрованные символы с данным сдвигом

2.5 Описание функции message_encryption

Функция принимает в качестве аргументов сдвиг int shift. Просит пользвателя ввести сообщение, которое надо зашифровать. После чего в цикле, выводит сначала в начале строку "Input message: после чего выводит зашифрованные символы с данным сдвигом

2.6 Описание функции file_encryption

Функция принимает в качестве аргументов ссылку на начало файла с сообщением fm, на начало файла, куда надо записать зашифрованное сообщение fe и сдвиг int shift. Просит пользвателя ввести сообщение, которое надо зашифровать. После чего в цикле, шифруются символы файла с сообщением и пишутся в файл для зашифрованного текста.

2.7 Описание функции decrypt_input

Функция просит ввести сообщение, после чего в цикле перебираются все возможные сдвиги и выводятся пользователю

2.8 Описание функции decrypt_input_with_shift

Функция принимает в качестве аргумента сдвиг, после чего просит пользователя ввести сообщение и в цикле расшифровывает сообщение с данным сдвигом. Выводит пользователю результат

2.9 Описание функции decrypt_file

Функция получает в качестве аргументов ссылку на начало файла с сообщением fm, на начало файла, куда надо записать расшифрованное сообщение fe, после чего в цикле перебирает возможные сдвиги, записывая в файл для расшифрованного сообщение

2.10 Описание функции decrypt_file_with_shift

Функция получает в качестве аргументов ссылку на начало файла с сообщением fm, на начало файла, куда надо записать расшифрованное сообщение fe и сдвиг shift. После чего, расшифровывает файл с данным сдвигом и записывает в файл для расшифрованного сообщения

2.11 Описание функции rev

Функция получает в качестве аргумента unsigned long len, и совершает переворот байтов этого значения

2.12 Описание функции crc32

Функция получает в качестве аргумента ссылку в памяти на начало массива unsigned char *data и длину массива unsigned long data_len, после чего подсчитывает контрольную сумму данных в массиве и возвращает её.

2.13 Описание функции message_encryption_to_str_file

Функция получает в качестве аргумента сдвиг. После чего просит пользователя ввести текст

для шифрования, который записывается в массив посимвольно, подсчитывается длина ввода. После чего создаётся файл hex_file_encoded.txt, куда записывается сначала сигнатура, соответствующая endian, потом сдвиг, длина информации, сама информация и в конце crc32.

2.14 Описание функции enc_strange_file_to_output

Функция получает в качестве аргументов ссылку на начало файла с сообщением fm. После чего данные из это файла шифруются с данным в нём сдвигом, создаётся файл encoded.txt, куда пишется зашифрованный текст.

2.15 Описание функции file_encryption_to_str_file

Функция получает в качестве аргументов ссылку на начало файла с сообщением fm. После чего создаётся файл hex_file_encoded.txt, куда записывается сначала сигнатура, соответствующая endian, потом сдвиг, длина информации, потом сама закодированная информация из файла и в конце crc32.

2.16 Описание функции enc_strange_file_to_strange_file

Функция получает в качестве аргументов сдвиг и ссылку на начало файла с сообщением и сигнатурой. После чего данные из это файла шифруются с данным в нём сдвигом, после чего создаётся файл hex_file_encoded.txt, куда пишутся данные, согласно описанному в задании формату. Форма записи зависит от endian.

2.17 Описание функции decrypt_input_to_str_file

Функция получает в качестве аргумента сдвиг. После чего просит пользователя ввести текст для дешифрования, который записывается в массив посимвольно, подсчитывается длина ввода. После чего создаётся hex_file_decoded.txt, куда записываются данные, соотвествующие формату из задания и endian.

2.18 Описание функции decrypt_file_to_str_file

Функция получает в качестве аргументов сдвиг и ссылку на начало файла с сообщением. После чего создаётся hex_file_decoded.txt, куда записываются данные, согласно формату файла вывода из задания и endian.

2.19 Описание функции dec_strange_file_to_strange$_{file}$

Функция получает в качестве аргумента ссылку на начало файла с сообщением. После чего создаётся hex_file_decoded.txt, куда записываются дешифрованные данные, согласно формату из задания и endian.

2.20 Описание функции dec_strange_file_to_file

Функция получает в качестве аргумента ссылку на начало файла с сообщением. После чего создаётся decoded.txt, куда записываются расшифрованные данные.

2.21 Описание функции troubleshooting_input

Функция получает в качестве аргументов ссылки на начало файла с сообщением и файла для результата. После чего проверяет, не NULL ли значения соответсвующие ссылкам на начала файлов

2.22 Описание функции check_for_sign

Функция получает в качестве аргумента ссылку на начало файла с сообщением и проверяет, имеет ли файл сигнатуру.

3 Тестирование

3.1 Проверка работы с файлом с сигнатурой LE

Содержимое файла hex_file_LE.txt:

```
00000000   EE CC 05 22  00 00 00 53  6F 6D 65 20  69 6E 66 6F  ..."...Some info
00000010   72 6D 61 74  69 6F 6E 20  6F 72 20 74  65 78 74 20  rmation or text
00000020   74 6F 20 64  65 63 6F 64  65 0A                     to decode.
```

```
$ ./a.out hex_file_LE.txt
Result was written to hex_file_encoded.txt
```

Содержимое файла hex_file_encoded.txt:

```
00000000   EE CC 05 22  00 00 00 58  74 72 6A 20  6E 73 6B 74  ..."...Xtrj nskt
00000010   77 72 66 79  6E 74 73 20  74 77 20 79  6A 63 79 20  wrfynts tw yjcy
00000020   79 74 20 69  6A 68 74 69  6A 5B 09 BF  67           yt ijhtij[..g
```

Если будет ключ ‘b‘:

```
00000000   CC EE 05 00  00 00 22 58  74 72 6A 20  6E 73 6B 74  ......"Xtrj nskt
00000010   77 72 66 79  6E 74 73 20  74 77 20 79  6A 63 79 20  wrfynts tw yjcy
00000020   79 74 20 69  6A 68 74 69  6A 67 BF 09  5B           yt ijhtijg..[
```

3.2 Проверка работы с файлом с сигнатурой BE

Содержимое файла hex_file_BE.txt:

```
00000000   CC EE 05 00  00 00 22 53  6F 6D 65 20  69 6E 66 6F  ......"Some info
00000010   72 6D 61 74  69 6F 6E 20  6F 72 20 74  65 78 74 20  rmation or text
00000020   74 6F 20 64  65 63 6F 64  65 0A                     to decode.
```

```
$./a.out hex_file_BE.txt
Result was written to hex_file_encoded.txt
```

Содержимое файла hex_file_encoded.txt:

```
00000000   EE CC 05 22  00 00 00 58  74 72 6A 20  6E 73 6B 74  ..."...Xtrj nskt
00000010   77 72 66 79  6E 74 73 20  74 77 20 79  6A 63 79 20  wrfynts tw yjcy
00000020   79 74 20 69  6A 68 74 69  6A 5B 09 BF  67           yt ijhtij[..g
```

Если будет ключ ‘b‘:

```
00000000   CC EE 05 00  00 00 22 58  74 72 6A 20  6E 73 6B 74  ......"Xtrj nskt
00000010   77 72 66 79  6E 74 73 20  74 77 20 79  6A 63 79 20  wrfynts tw yjcy
00000020   79 74 20 69  6A 68 74 69  6A 67 BF 09  5B           yt ijhtijg..[
```

3.3 Проверка работы с файлом

```
$ cat msg.txt
Hello!!!

$ ./a.out msg.txt d
Result was written to hex_file_decoded.txt
```

Содержимое hex_file_decoded.txt:

```
00000000   EE CC 05 09  00 00 00 43  7A 67 67 6A  21 21 21 0A  .......Czggj!!!.
00000010   51 A8 F4 15                                        Q...
```

С ключом ‘b‘:

```
00000000   CC EE 05 00  00 00 09 43  7A 67 67 6A  21 21 21 0A  .......Czggj!!!.
00000010   15 F4 A8 51                                        ...Q
```

3.4 Проверка работы с пользовательским вводом

```
$ ./a.out b c
Input text:I hope it works
Result was written to  hex_file_encoded.txt
```

Содержимое hex_file_encoded.txt:

```
00000000   CC EE 05 0F  00 00 00 4E  20 6D 74 75  6A 20 6E 79  .......N mtuj ny
00000010   20 62 74 77  70 78 80 B7  CA C0                     btwpx....
```

3.5 Проверка работы с целым числом без файла

3.6 Проверка работы с целым числом и файлом

```
$ ./a.out 5
Input text:It’s nice
Result was written to  hex_file_encoded.txt
```

Содержимое hex_file_encoded.txt:

```
00000000   EE CC 05 09  00 00 00 4E  79 27 78 20  73 6E 68 6A  .......Ny’x snhj
00000010   1D 53 45 62                                        .SEb
```

3.7 Проверка вывода ошибки при вводе двух файлов: с сигнатурой и без

```
$ ./a.out hex_file_LE.txt msg.txt
something wrong with file
```

3.8 Проверка вывода ошибки при вводе c и d одновременно

```
$ ./a.out d c
usage: ./a.out [c/d] [k] [file] [-] [b]
```

3.9 Проверка вывода ошибки при вводе неверного ключа или неверного файла

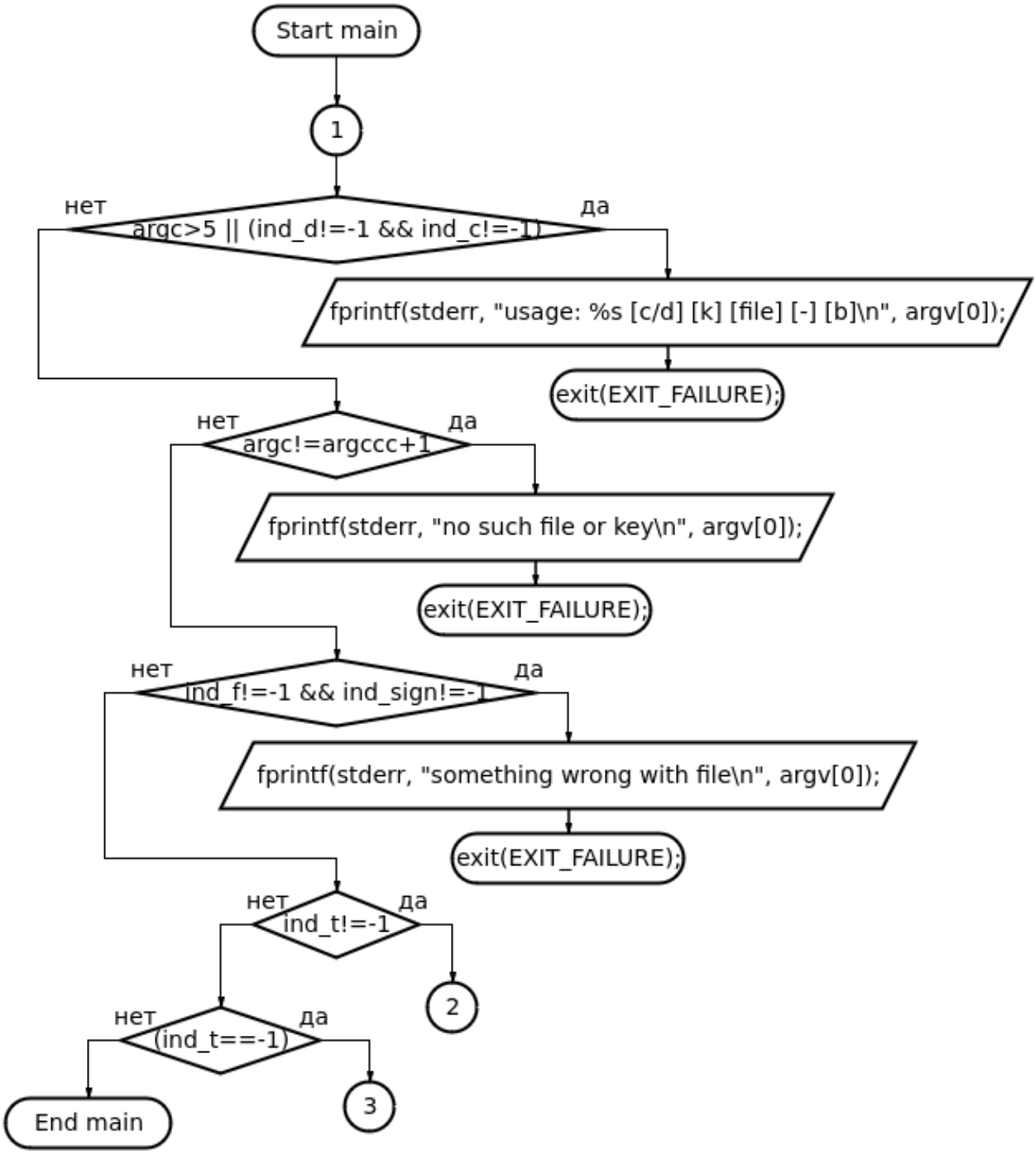
```
$ ./a.out k j
no such file or key
$ ./a.out somefile.txt
no such file or key
```

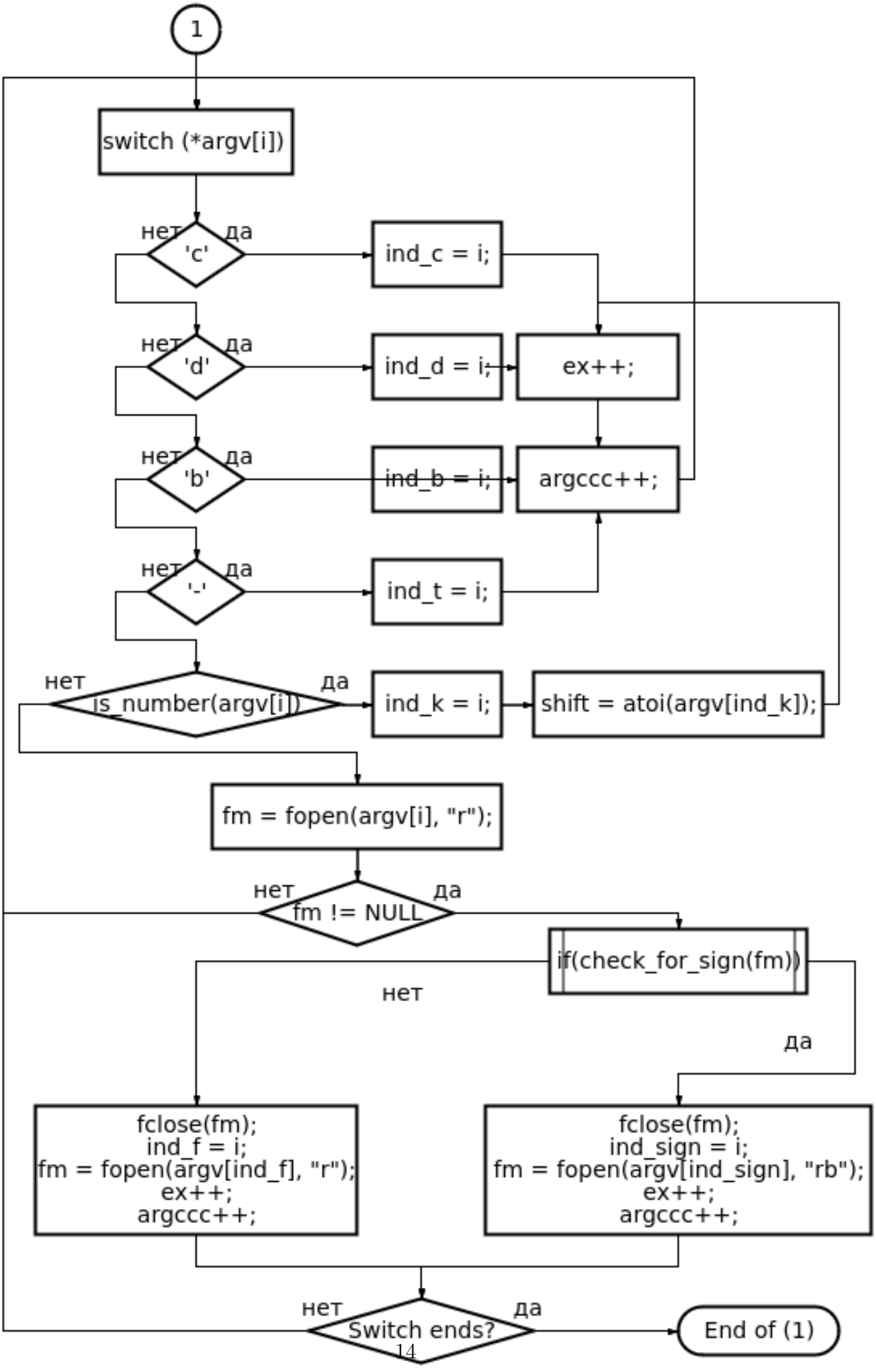
3.10 Проверка вывода ошибки при разных сдвигах в файле с сигнатурой и введённым пользователем

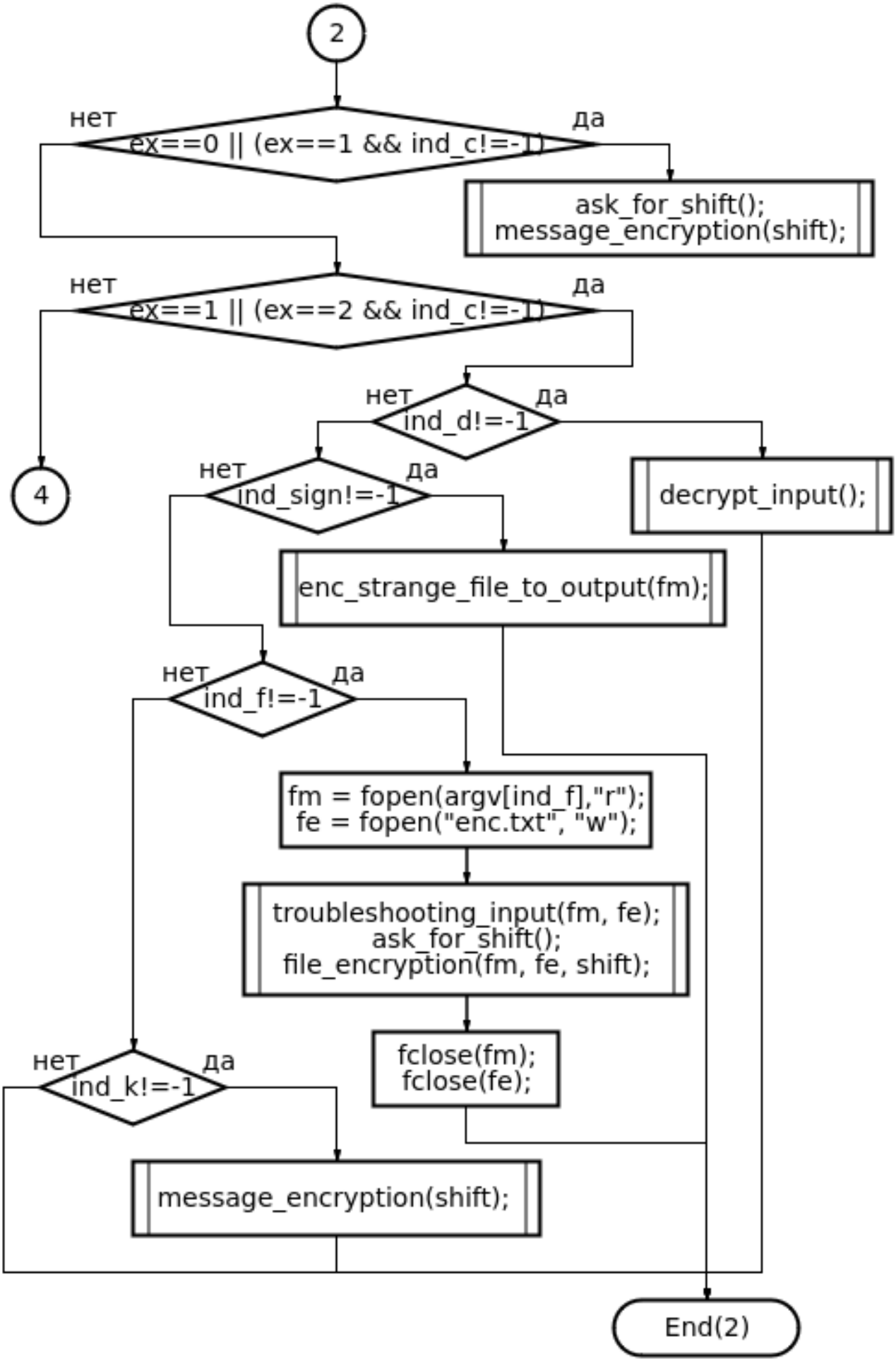
```
$ ./a.out 12 hex_file_LE.txt
Different shifts
```

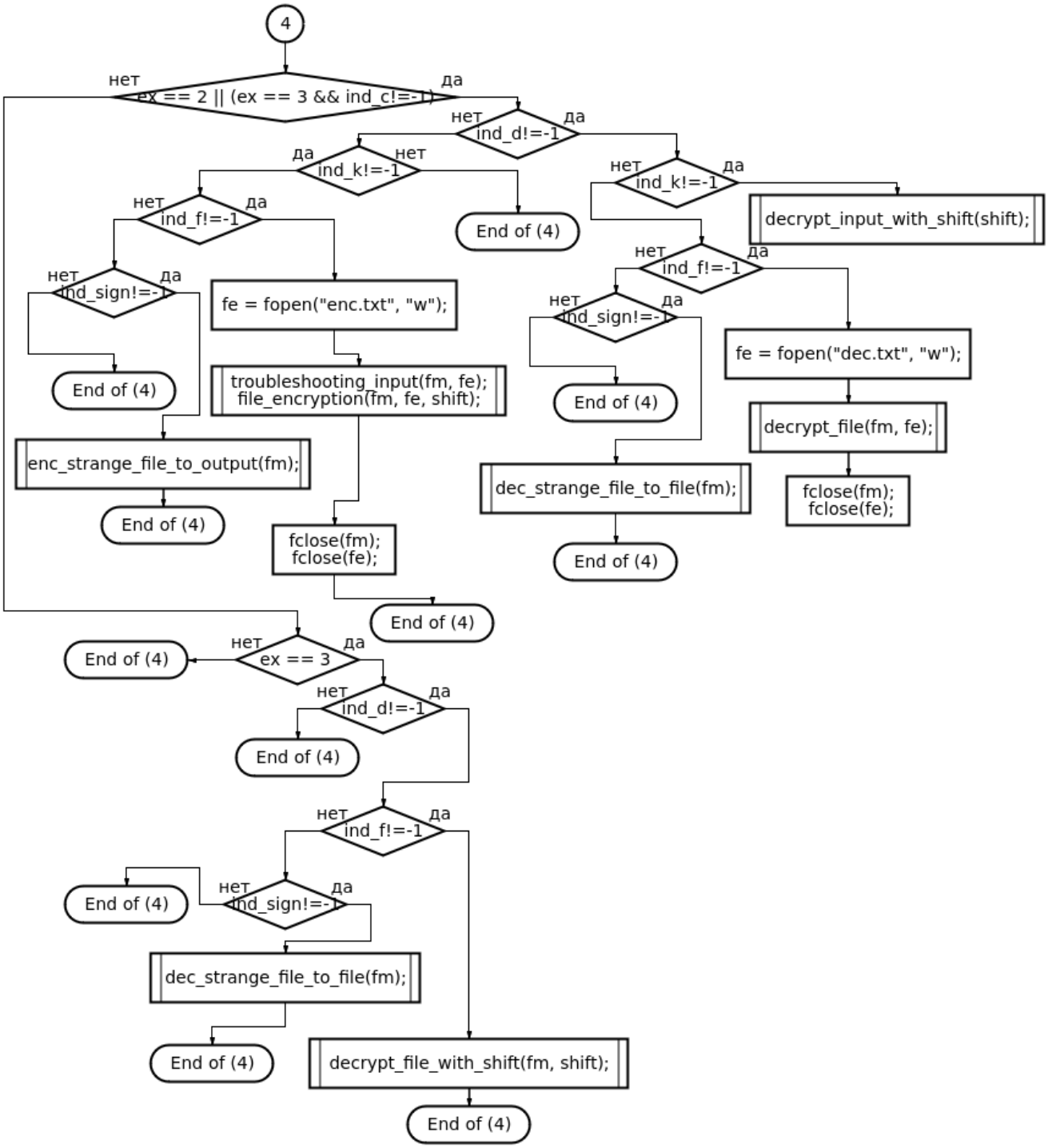
Приложение А

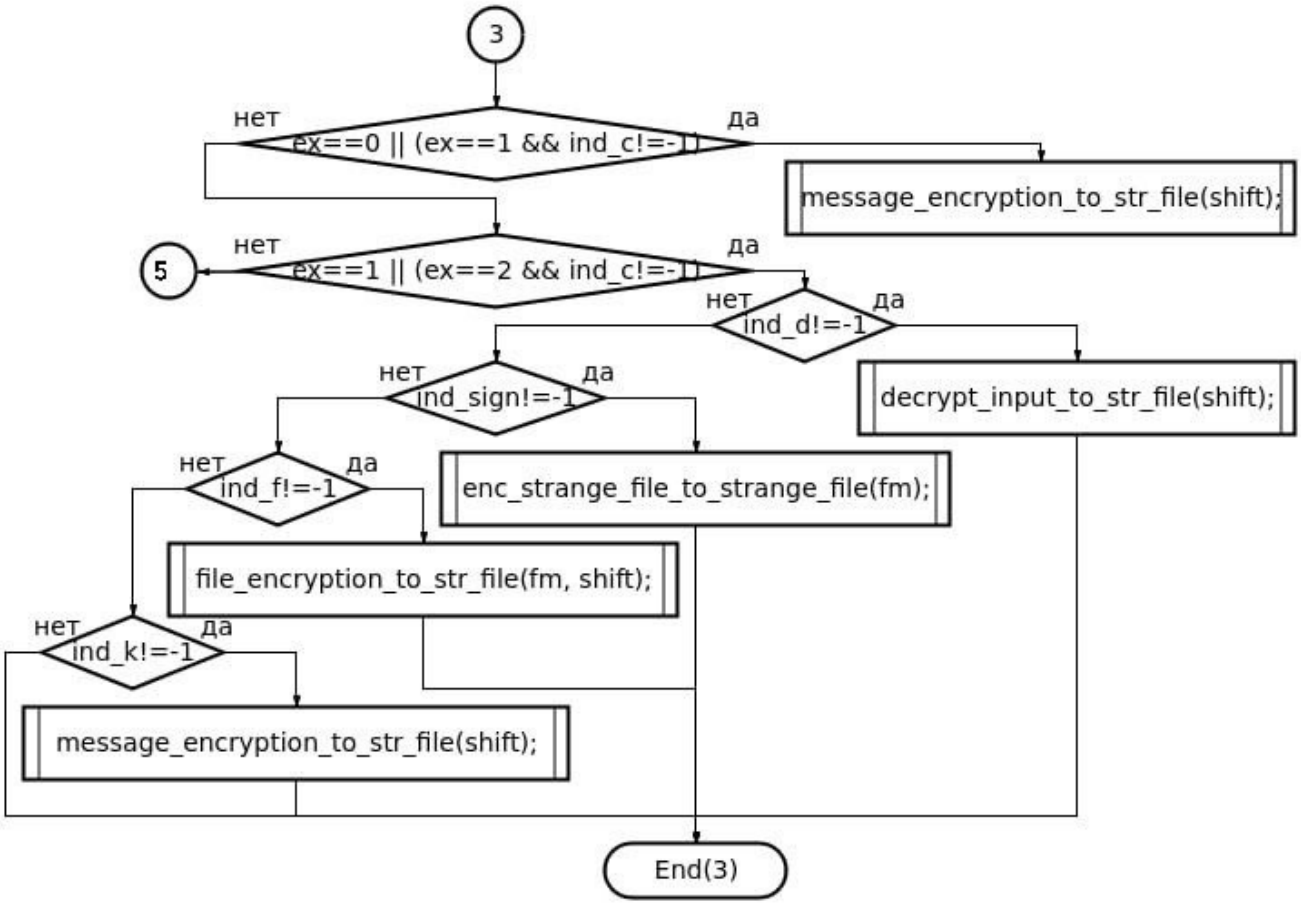
Блок-схемы *main*

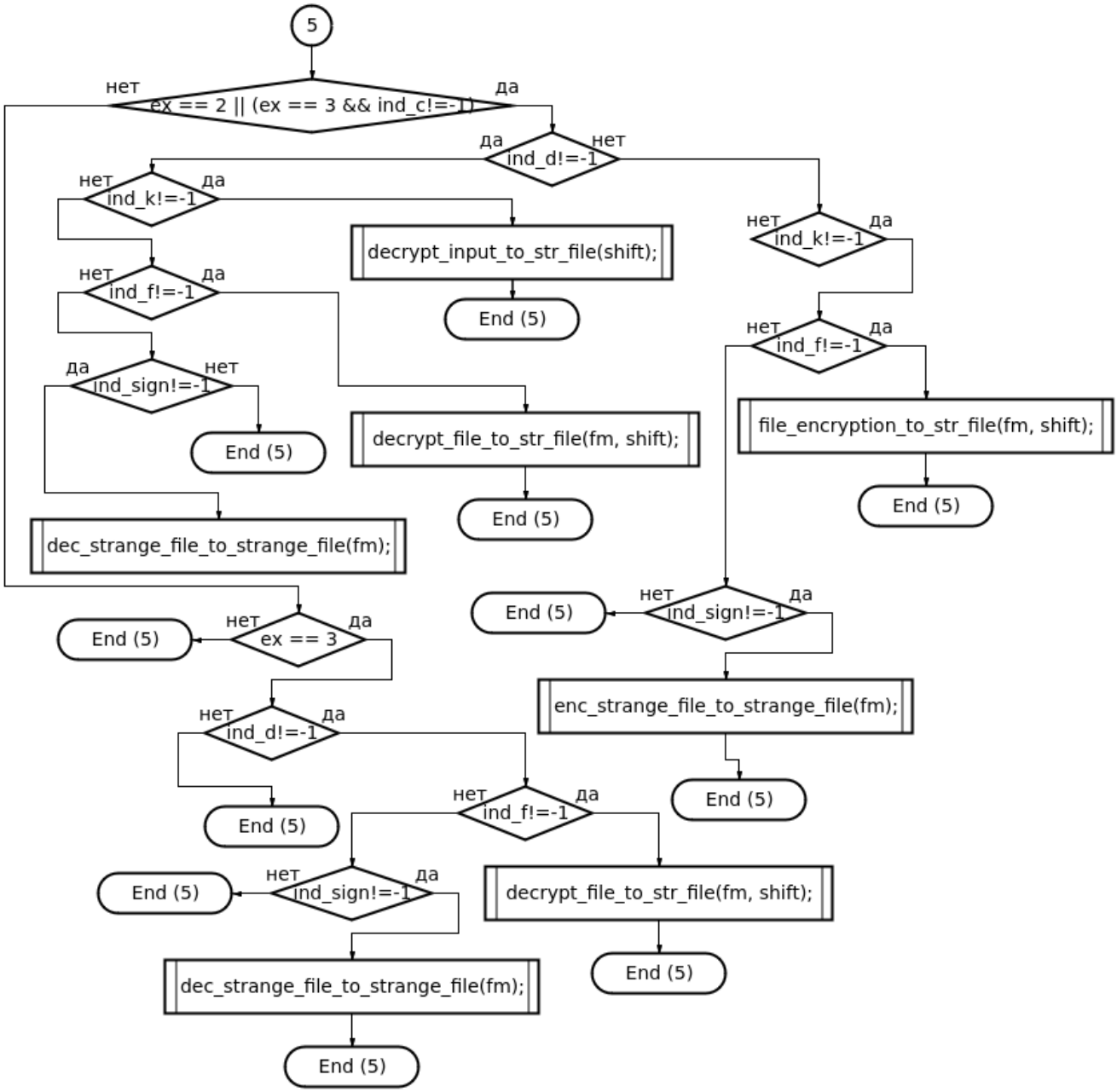



1
switch (*argv[i])
нет
'c'
да
ind_c = i;
нет
'd'
да
ind_d = i;
ex++;
нет
'b'
да
ind_b = i;
argccc++;
нет
'-'
да
ind_t = i;
нет
is_number(argv[i])
да
ind_k = i;
shift = atoi(argv[ind_k]);
fm = fopen(argv[i], "r");
нет
fm != NULL
да
if(check_for_sign(fm))
нет
да
fclose(fm);
ind_f = i;
fm = fopen(argv[ind_f], "r");
ex++;
argccc++;
fclose(fm);
ind_sign = i;
fm = fopen(argv[ind_sign], "rb");
ex++;
argccc++;
нет
Switch ends?
да
End of (1)

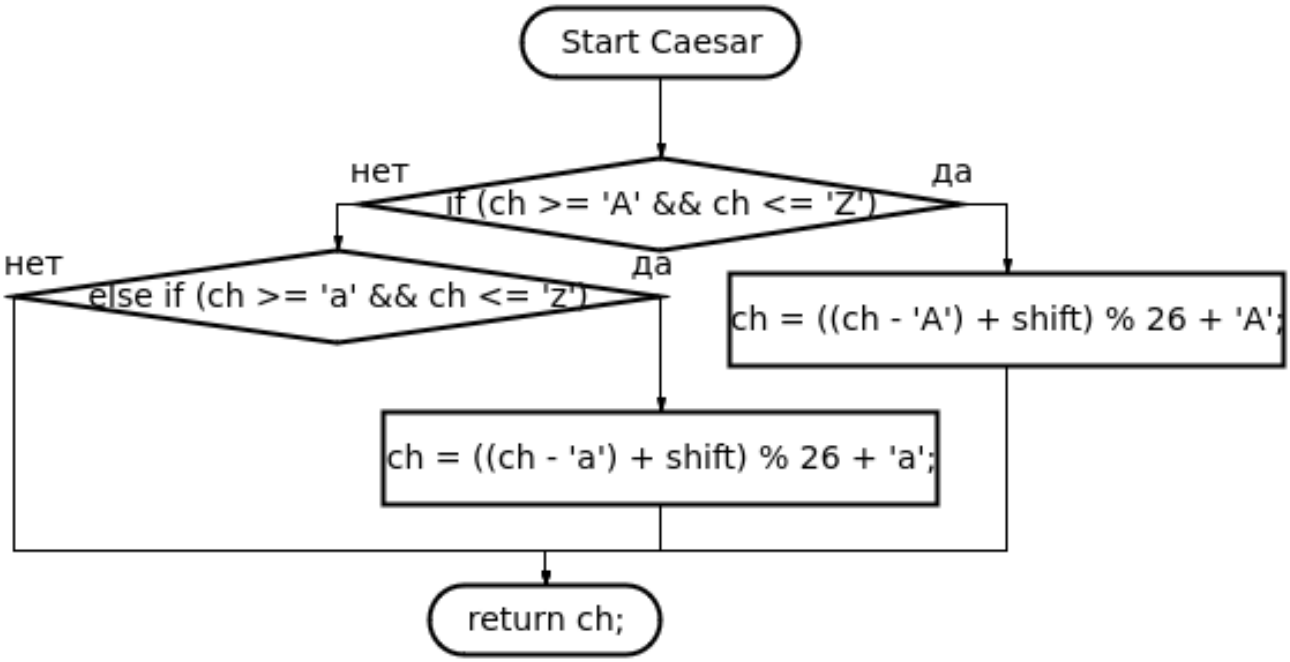

2
нет
ex==0 || (ex==1 && ind_c!=-1)
да
ask_for_shift();
message_encryption(shift);
нет
ex==1 || (ex==2 && ind_c!=-1)
да
нет
ind_d!=-1
да
4
нет
ind_sign!=-1
да
decrypt_input();
enc_strange_file_to_output(fm);
нет
ind_f!=-1
да
fm = fopen(argv[ind_f],"r");
fe = fopen("enc.txt", "w");
troubleshooting_input(fm, fe);
ask_for_shift();
file_encryption(fm, fe, shift);
fclose(fm);
fclose(fe);
нет
ind_k!=-1
да
message_encryption(shift);
End(2)


4
нет
ex == 2 || (ex == 3 && ind_c!=-1)
да
нет
ind_d!=-1
да
да
ind_k!=-1
нет
нет
ind_f!=-1
да
End of (4)
нет
ind_k!=-1
да
decrypt_input_with_shift(shift);
нет
ind_f!=-1
да
нет
ind_sign!=-1
да
fe = fopen("enc.txt", "w");
нет
ind_sign!=-1
да
fe = fopen("dec.txt", "w");
End of (4)
troubleshooting_input(fm, fe);
file_encryption(fm, fe, shift);
End of (4)
decrypt_file(fm, fe);
enc_strange_file_to_output(fm);
dec_strange_file_to_file(fm);
fclose(fm);
fclose(fe);
End of (4)
fclose(fm);
fclose(fe);
End of (4)
End of (4)
нет
ex == 3
да
End of (4)
нет
ind_d!=-1
да
End of (4)
нет
ind_f!=-1
да
End of (4)
нет
ind_sign!=-1
да
dec_strange_file_to_file(fm);
End of (4)
decrypt_file_with_shift(fm, shift);
End of (4)

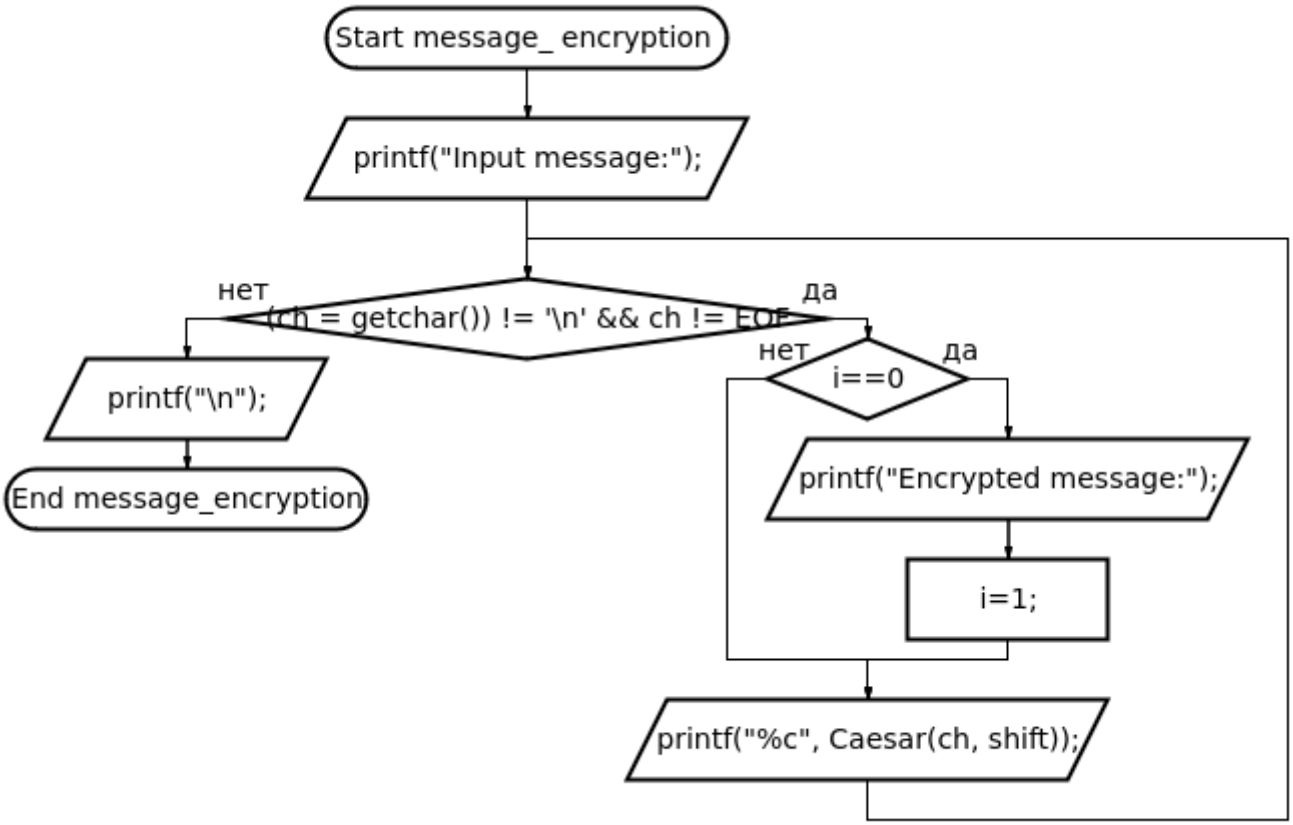

3
нет
ex==0 || (ex==1 && ind_c!=-1)
да
message_encryption_to_str_file(shift);
5
нет
ex==1 || (ex==2 && ind_c!=-1)
да
нет
ind_d!=-1
да
decrypt_input_to_str_file(shift);
нет
ind_sign!=-1
да
нет
ind_f!=-1
да
enc_strange_file_to_strange_file(fm);
file_encryption_to_str_file(fm, shift);
нет
ind_k!=-1
да
message_encryption_to_str_file(shift);
End(3)


5
нет
ex == 2 || (ex == 3 && ind_c!=-1)
да
да
ind_d!=-1
нет
нет
ind_k!=-1
да
decrypt_input_to_str_file(shift);
End (5)
нет
ind_f!=-1
да
да
ind_sign!=-1
нет
End (5)
decrypt_file_to_str_file(fm, shift);
End (5)
dec_strange_file_to_strange_file(fm);
нет
ind_k!=-1
да
нет
ind_f!=-1
да
file_encryption_to_str_file(fm, shift);
End (5)
End (5)
нет
ind_sign!=-1
да
enc_strange_file_to_strange_file(fm);
End (5)
End (5)
нет
ex == 3
да
нет
ind_d!=-1
да
End (5)
нет
ind_f!=-1
да
End (5)
нет
ind_sign!=-1
да
decrypt_file_to_str_file(fm, shift);
End (5)
End (5)
dec_strange_file_to_strange_file(fm);
End (5)

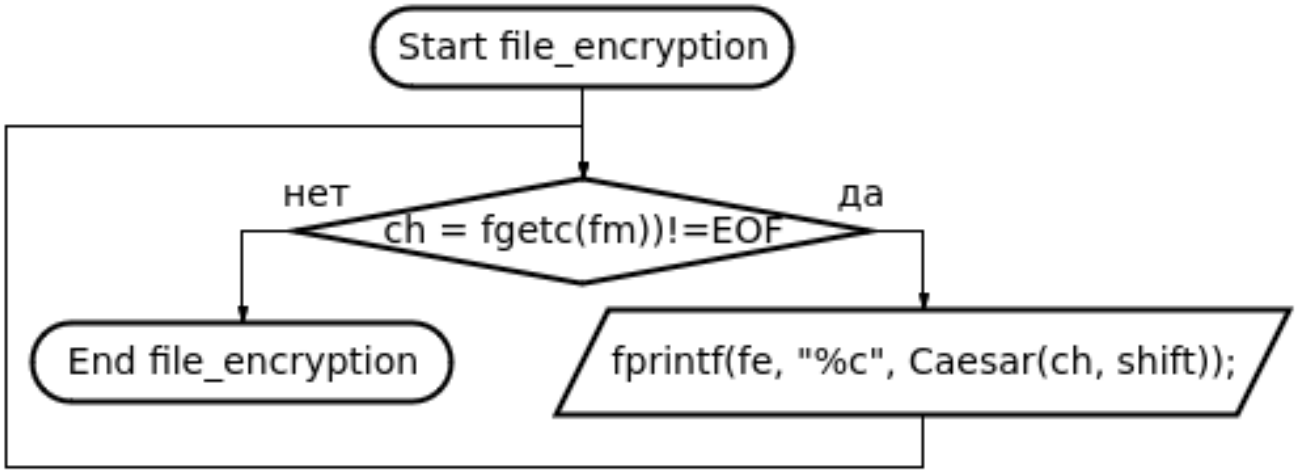
Блок-схема *Caesar*



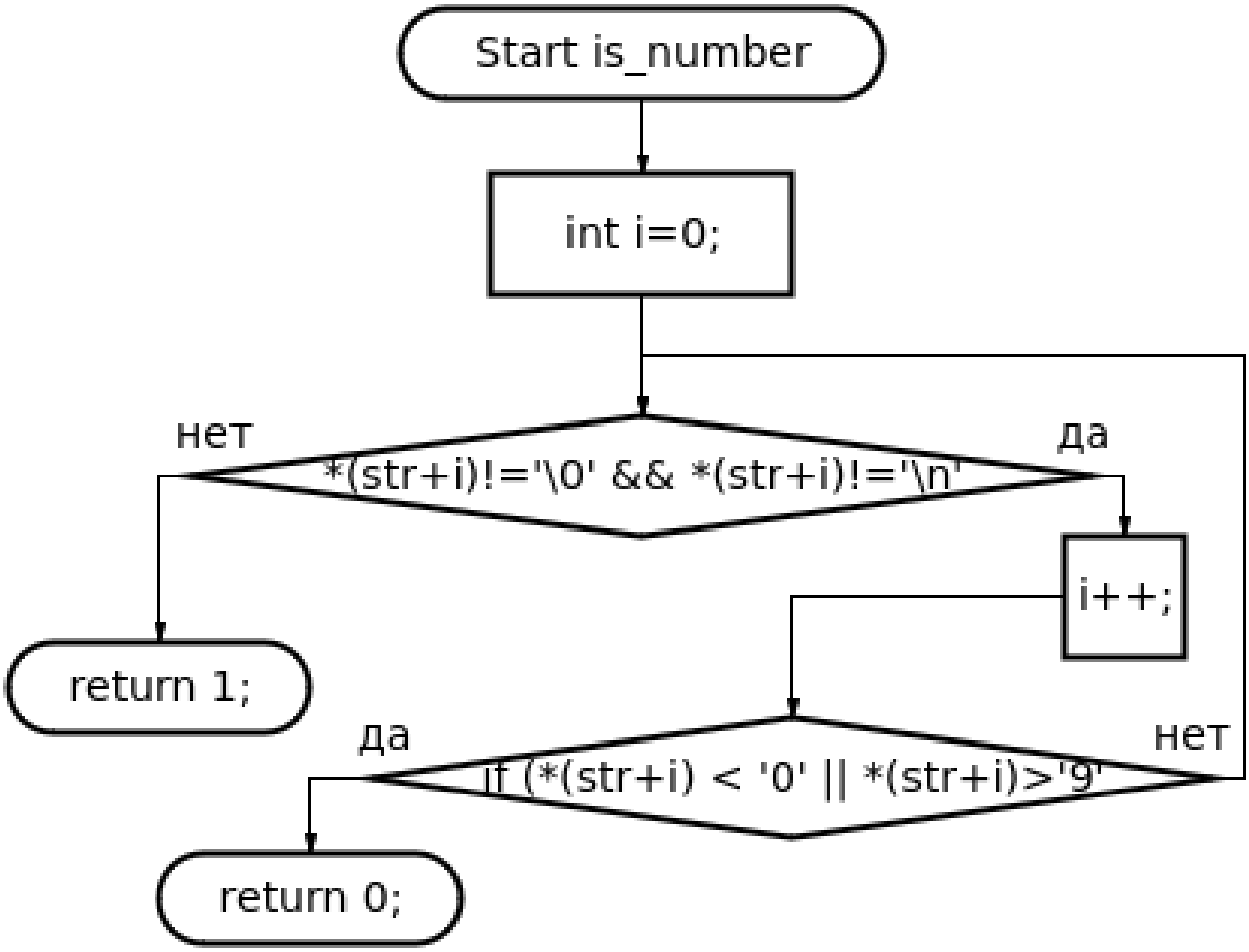
Блок-схема *message_ encryption*



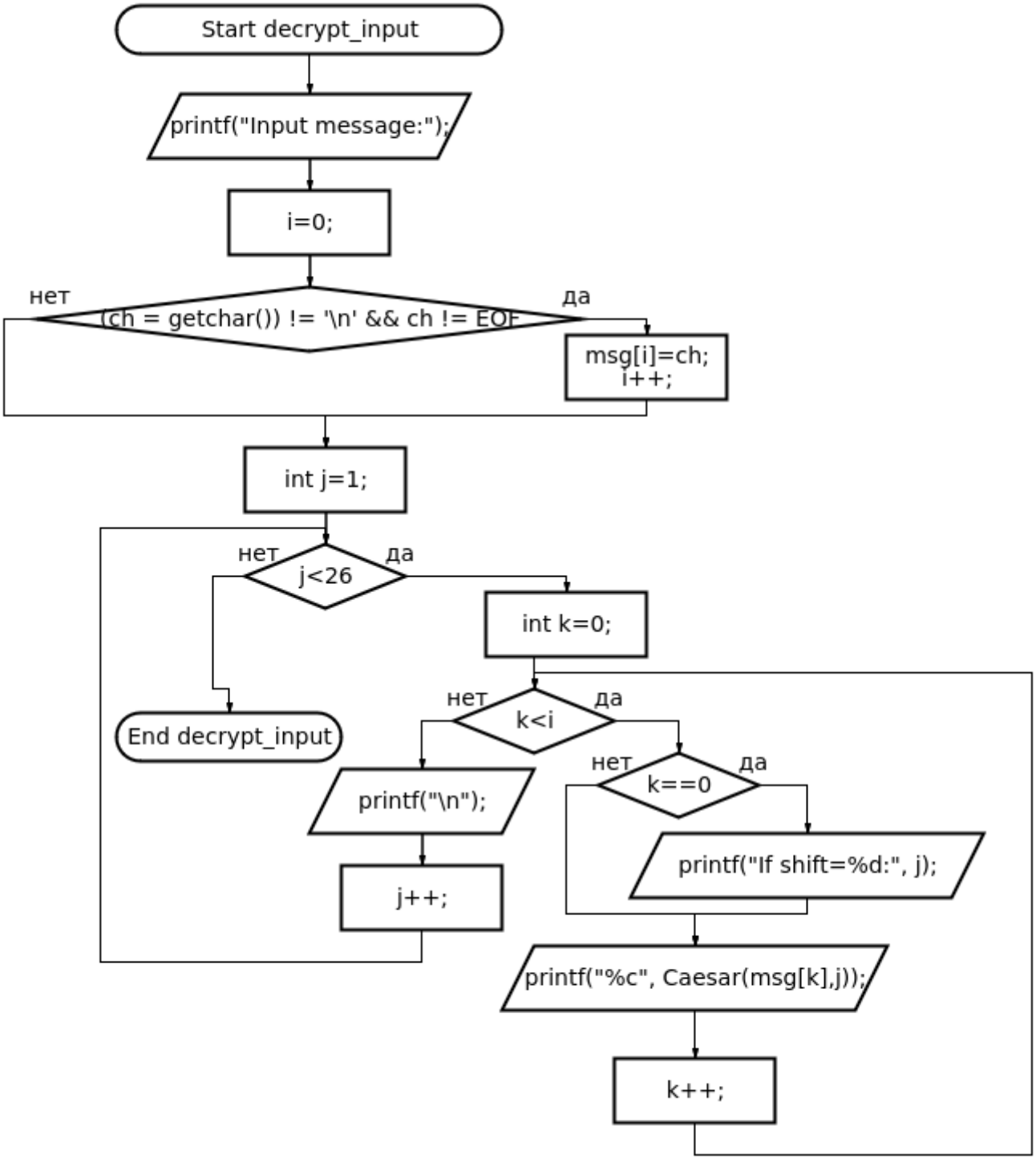
Блок-схема *file_ encryption*



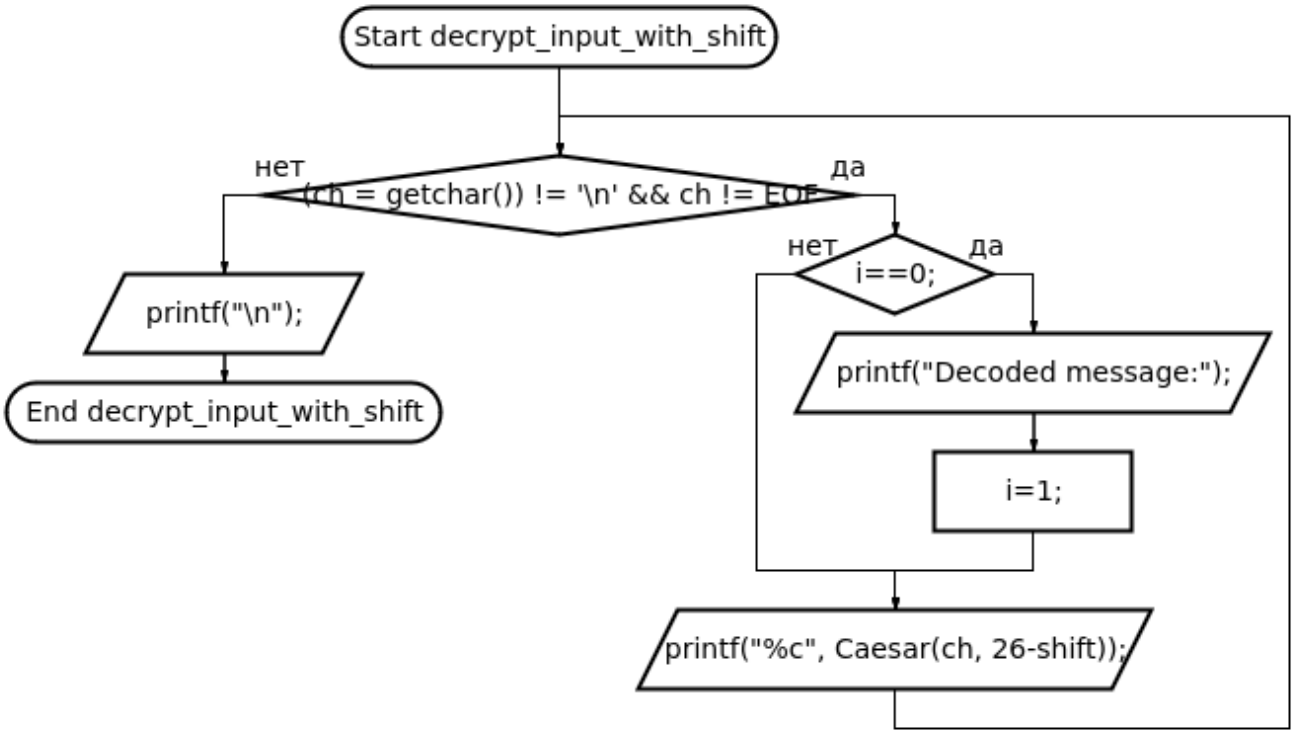
Блок-схема *is_ number*



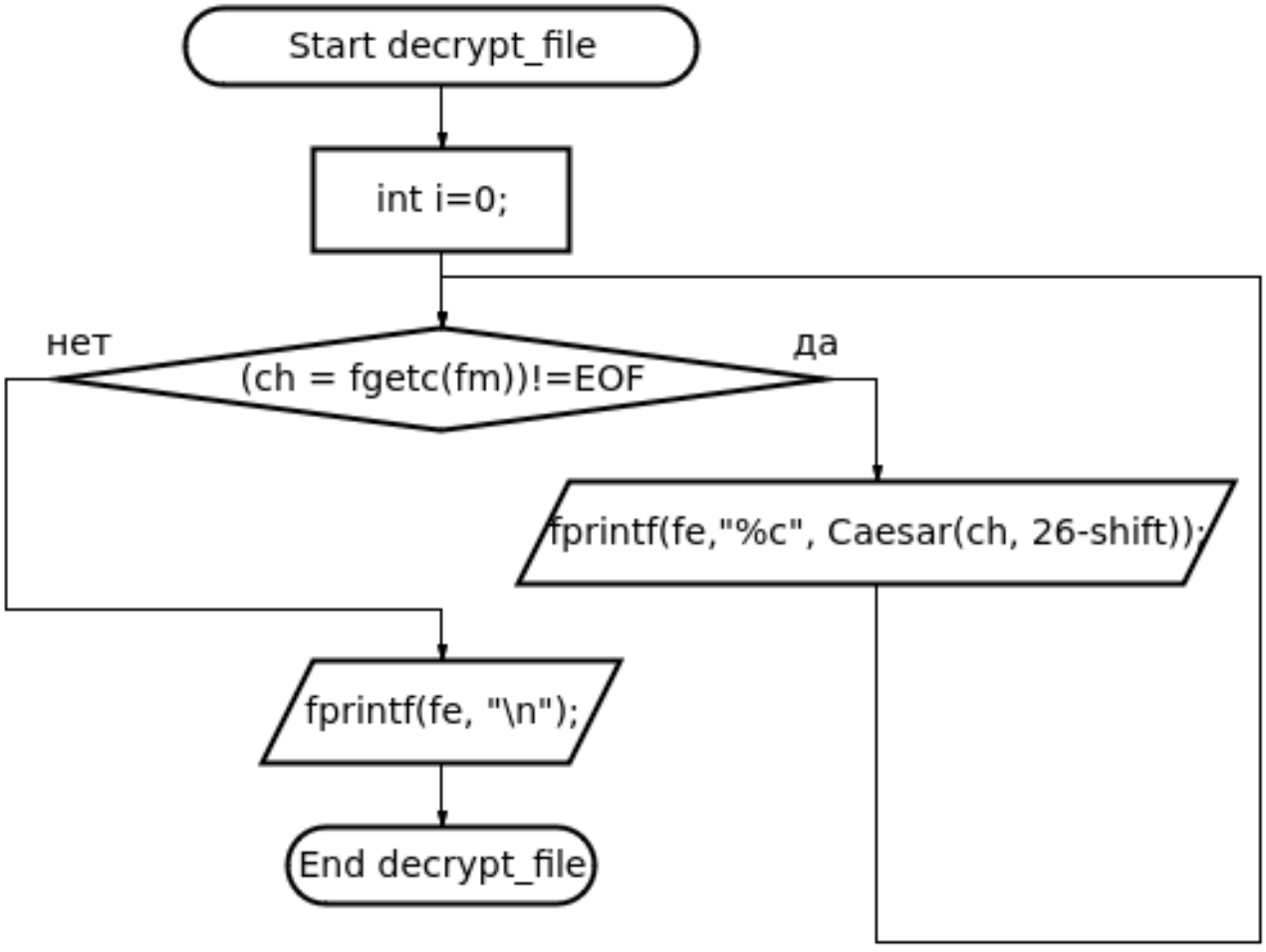
Блок-схема *decrypt_input*



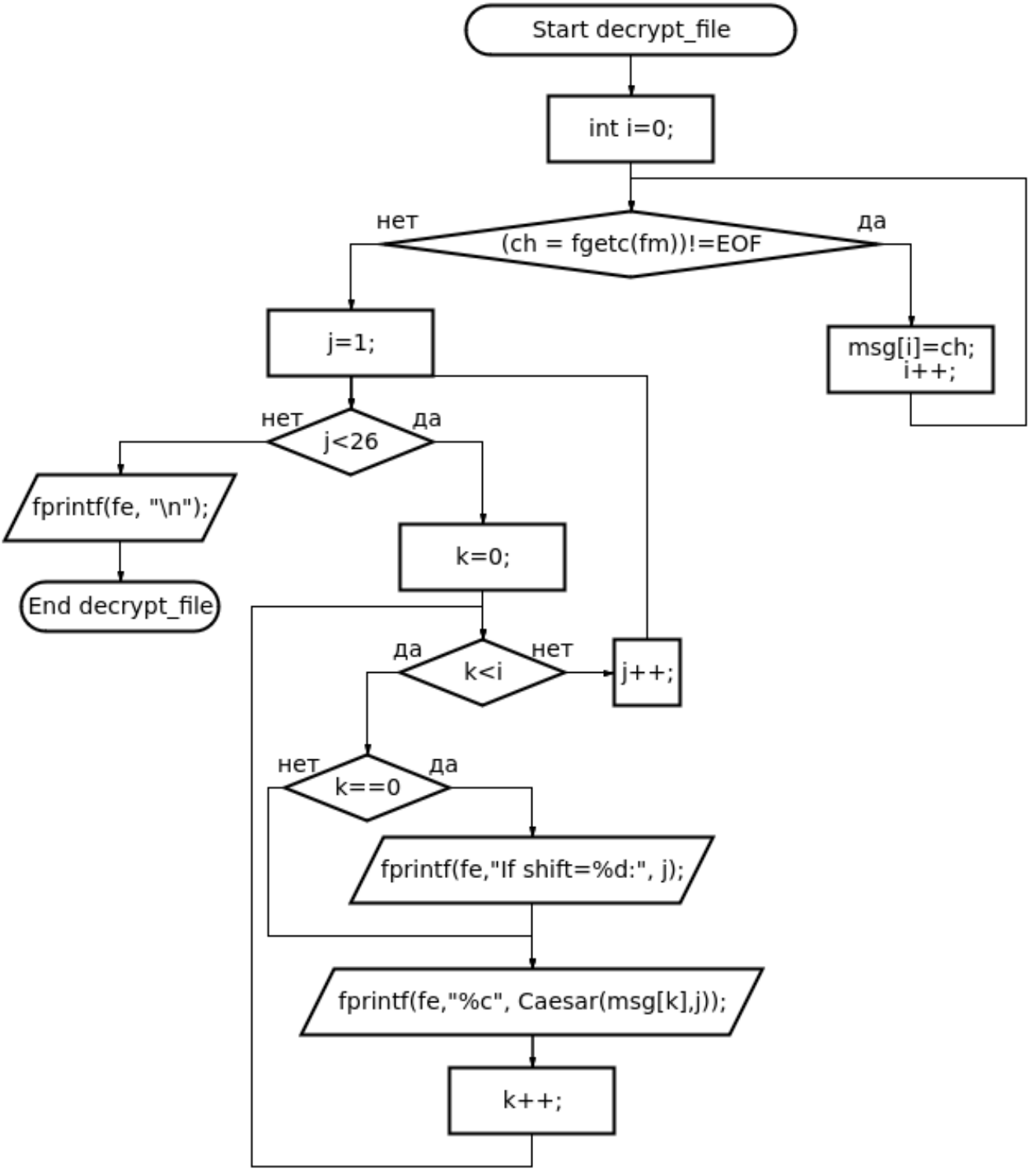
Блок-схема *decrypt_input_with_shift*



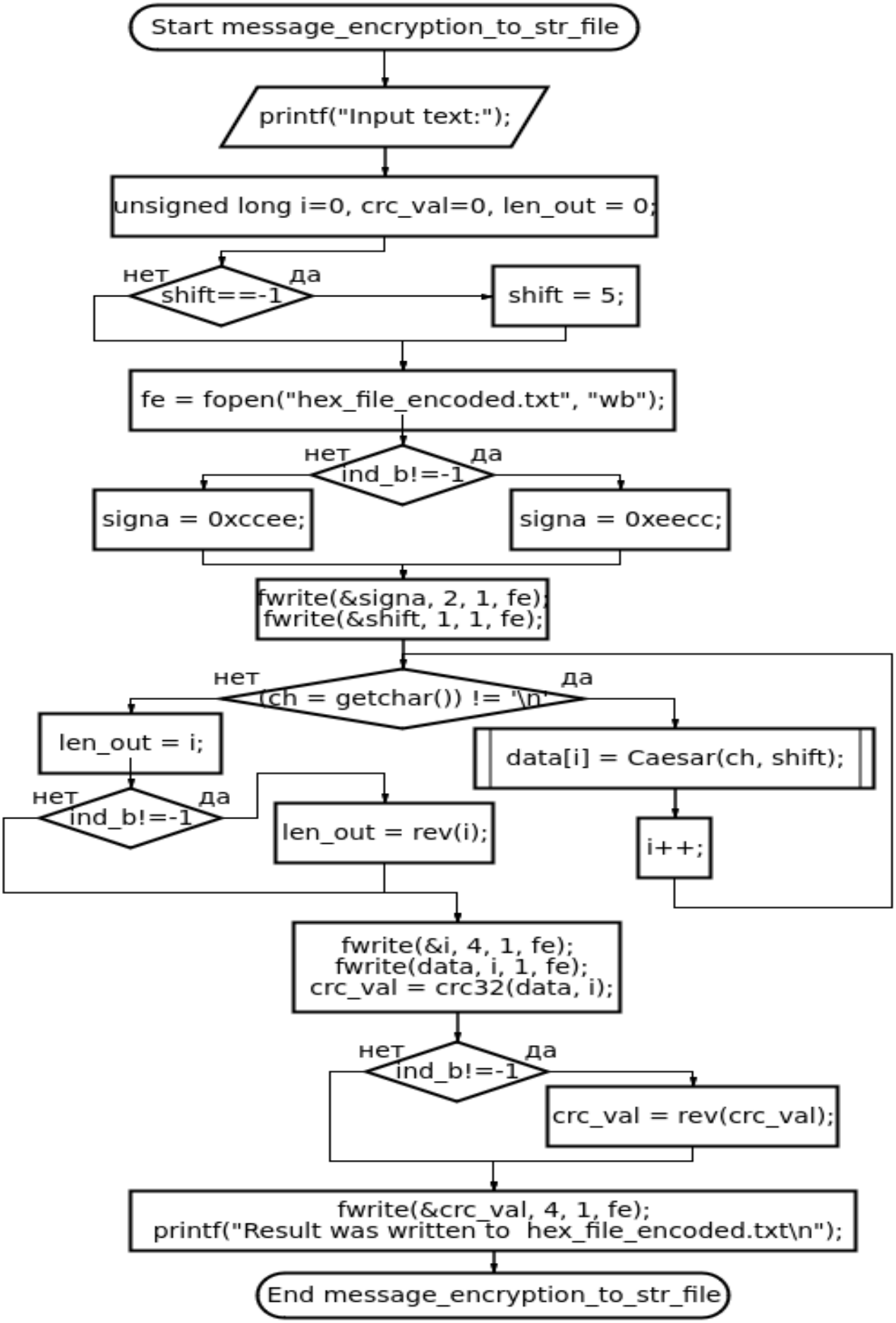
Блок-схема *decrypt_file_with_shift*



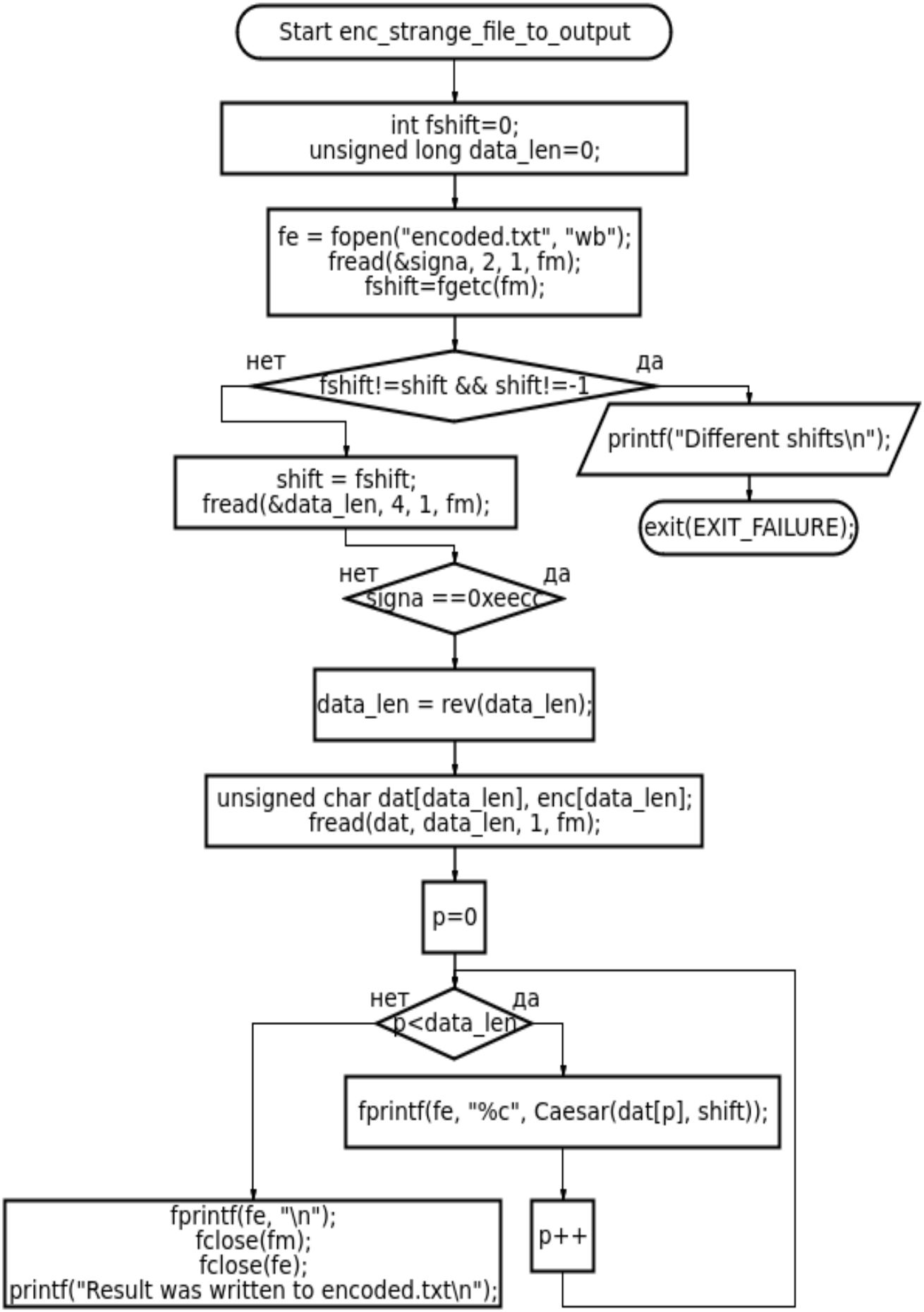
Блок-схема *decrypt_file*



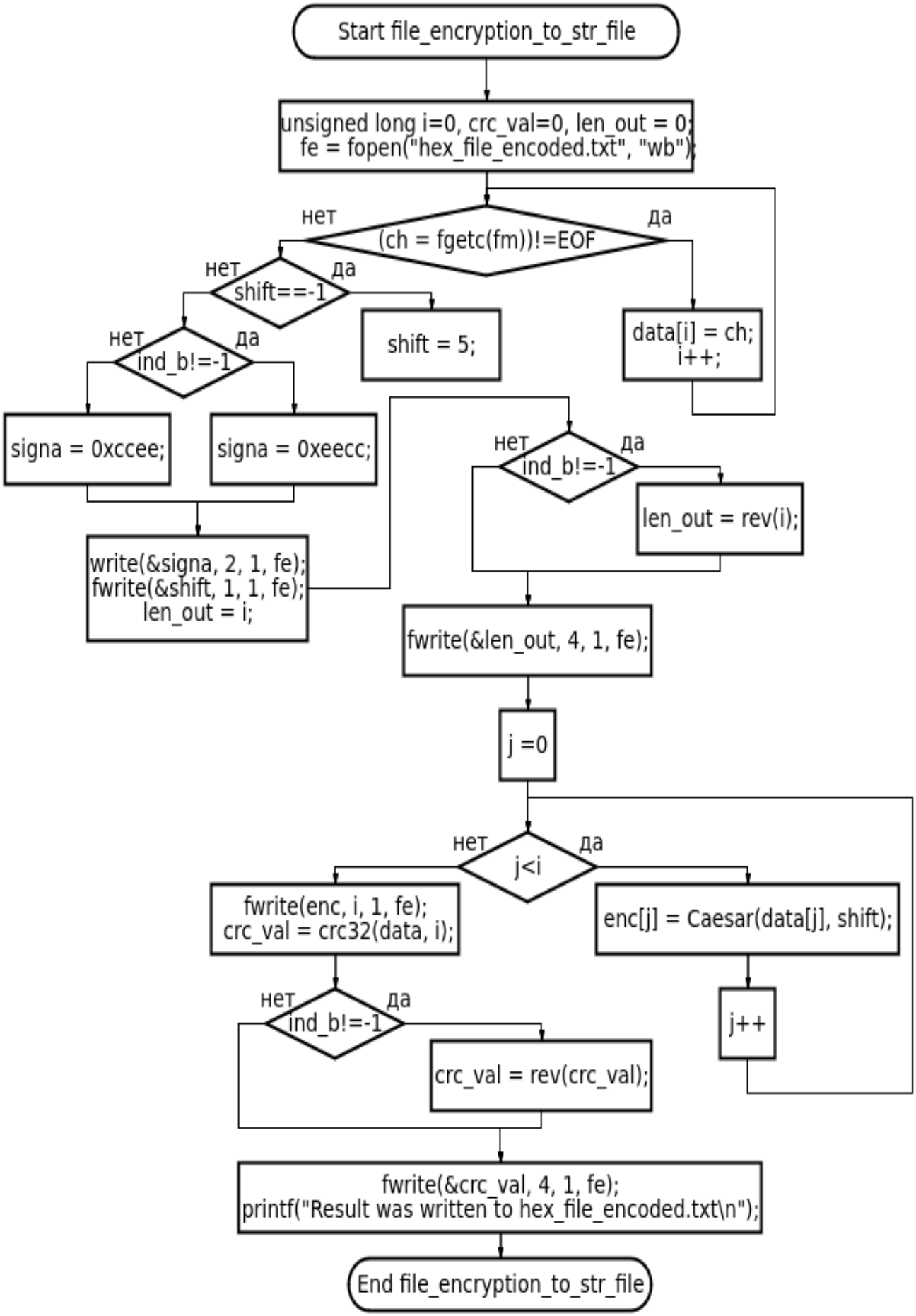
Блок-схема *message_encryption_to_str_file*



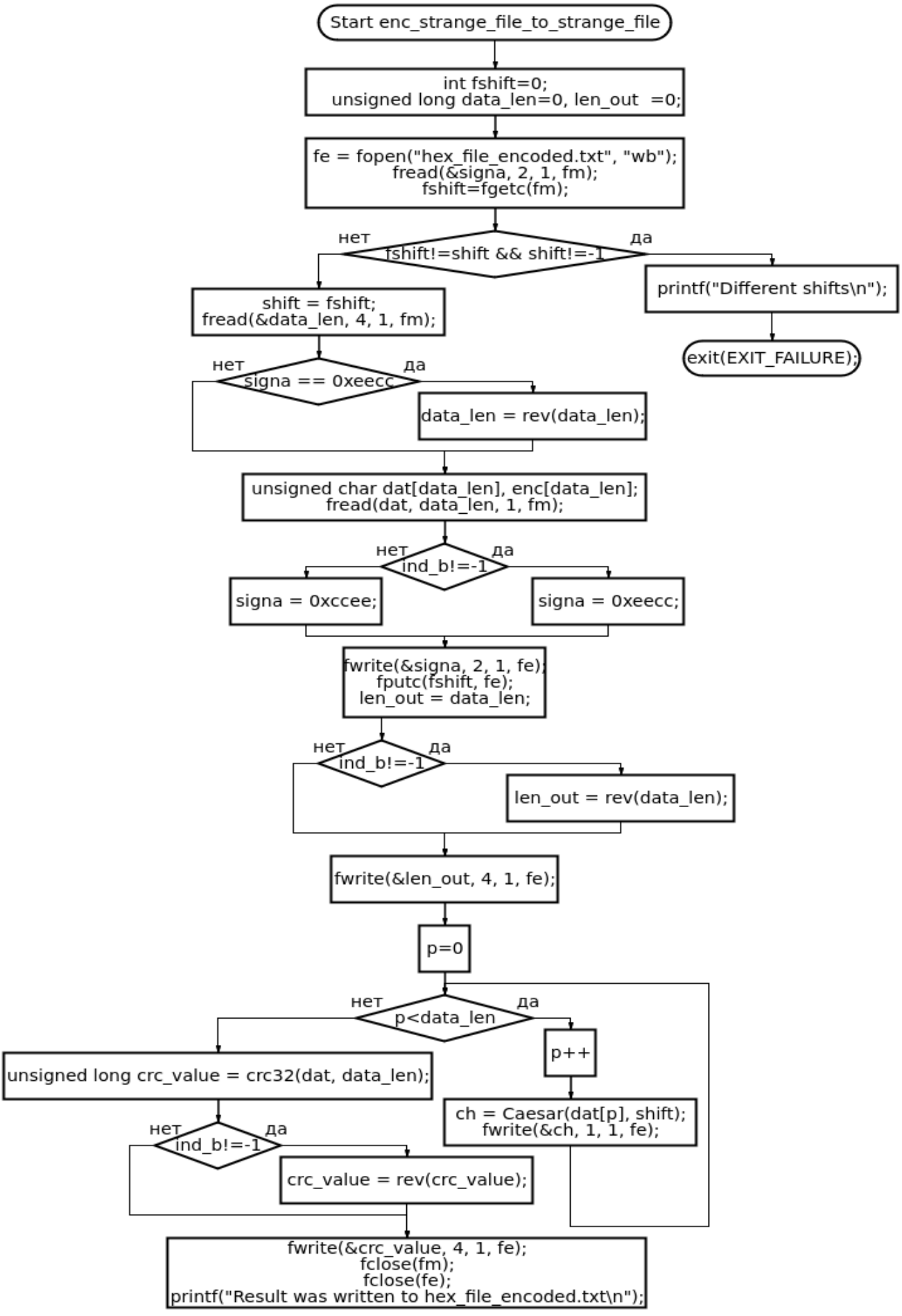
Блок-схема *enc_strange_file_to_output*



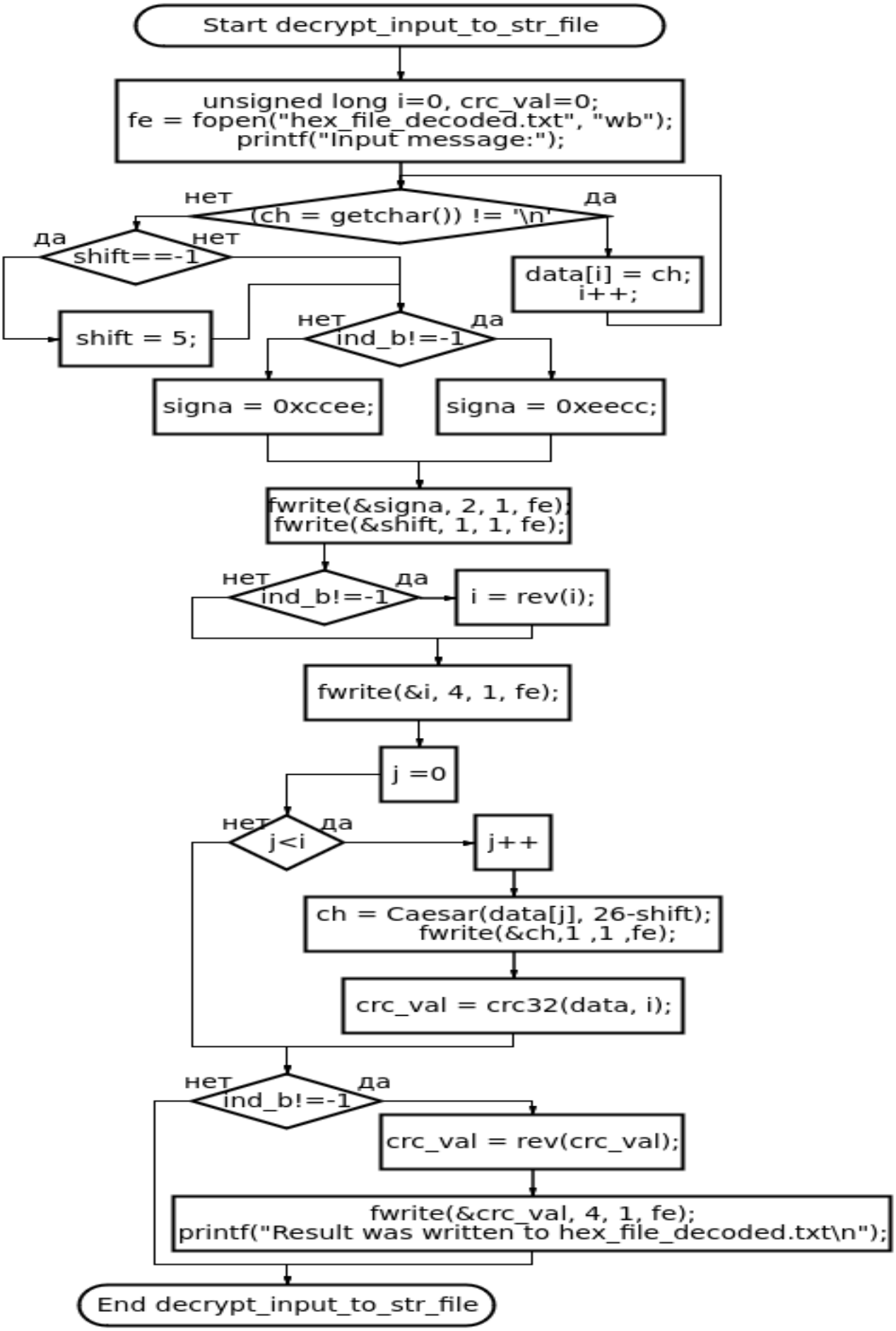
Блок-схема *file_ encryption_ to_ str_ file*



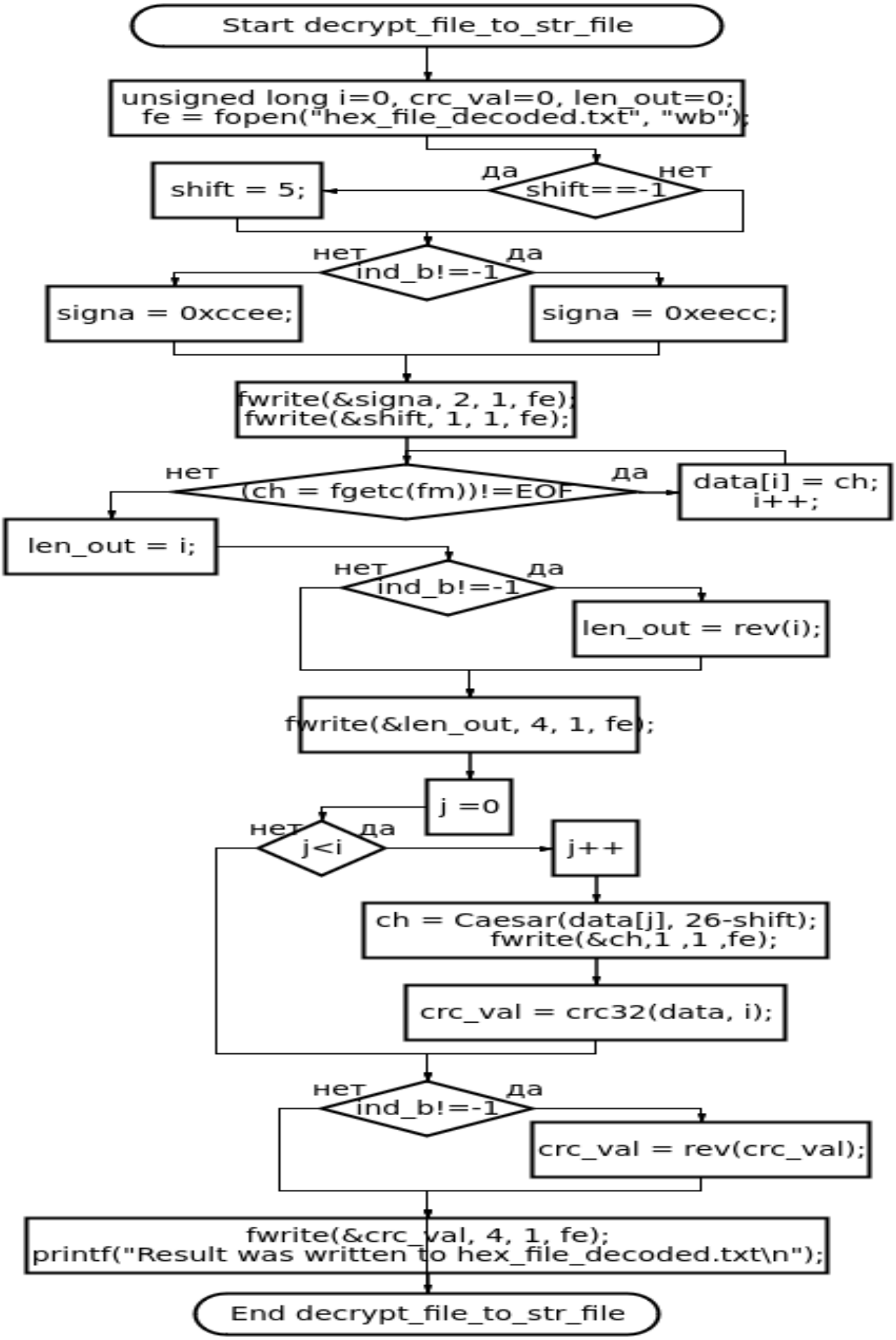
Блок-схема enc_strange_file_to_strange_file



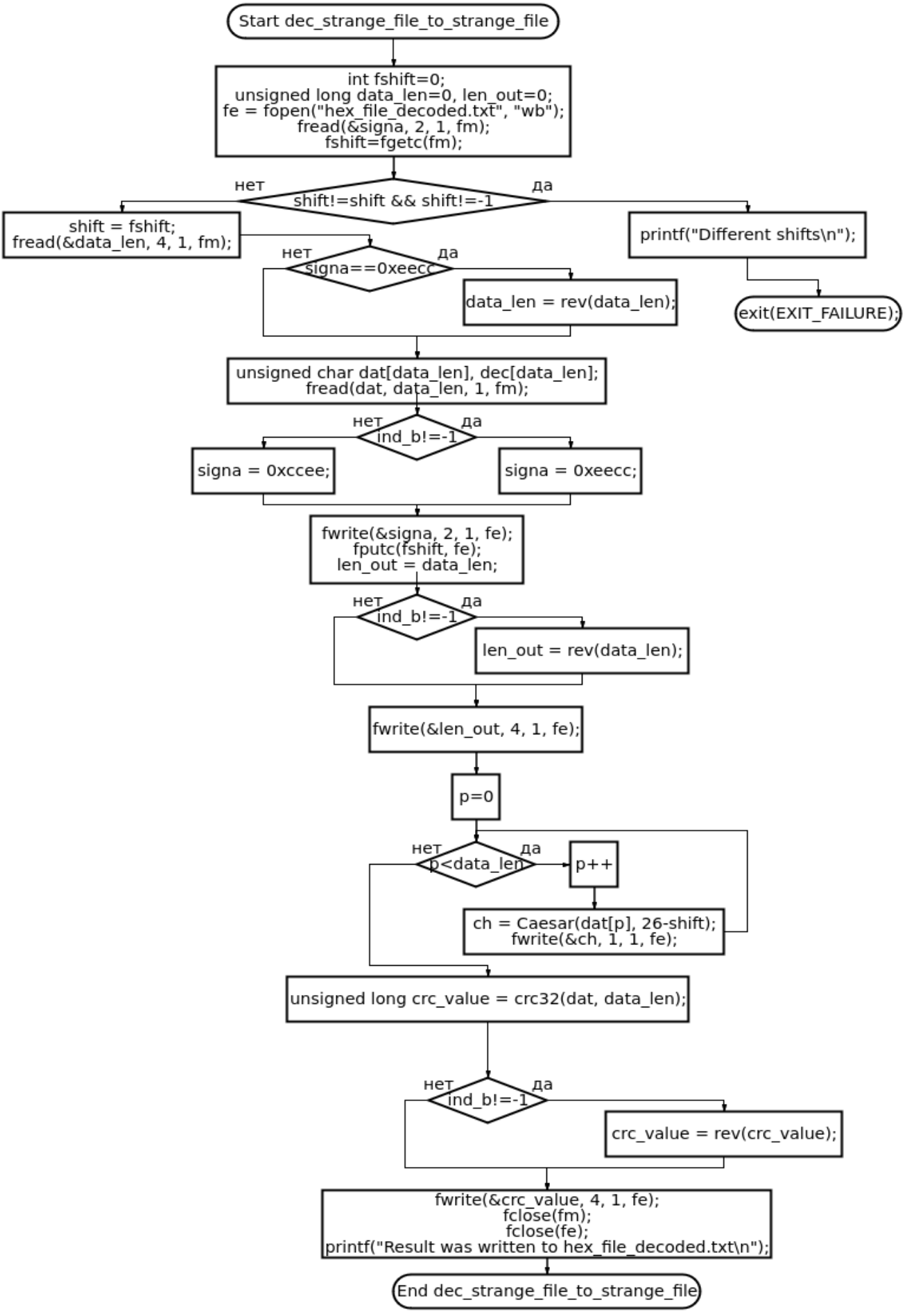
Блок-схема *decrypt_input_to_str_file*



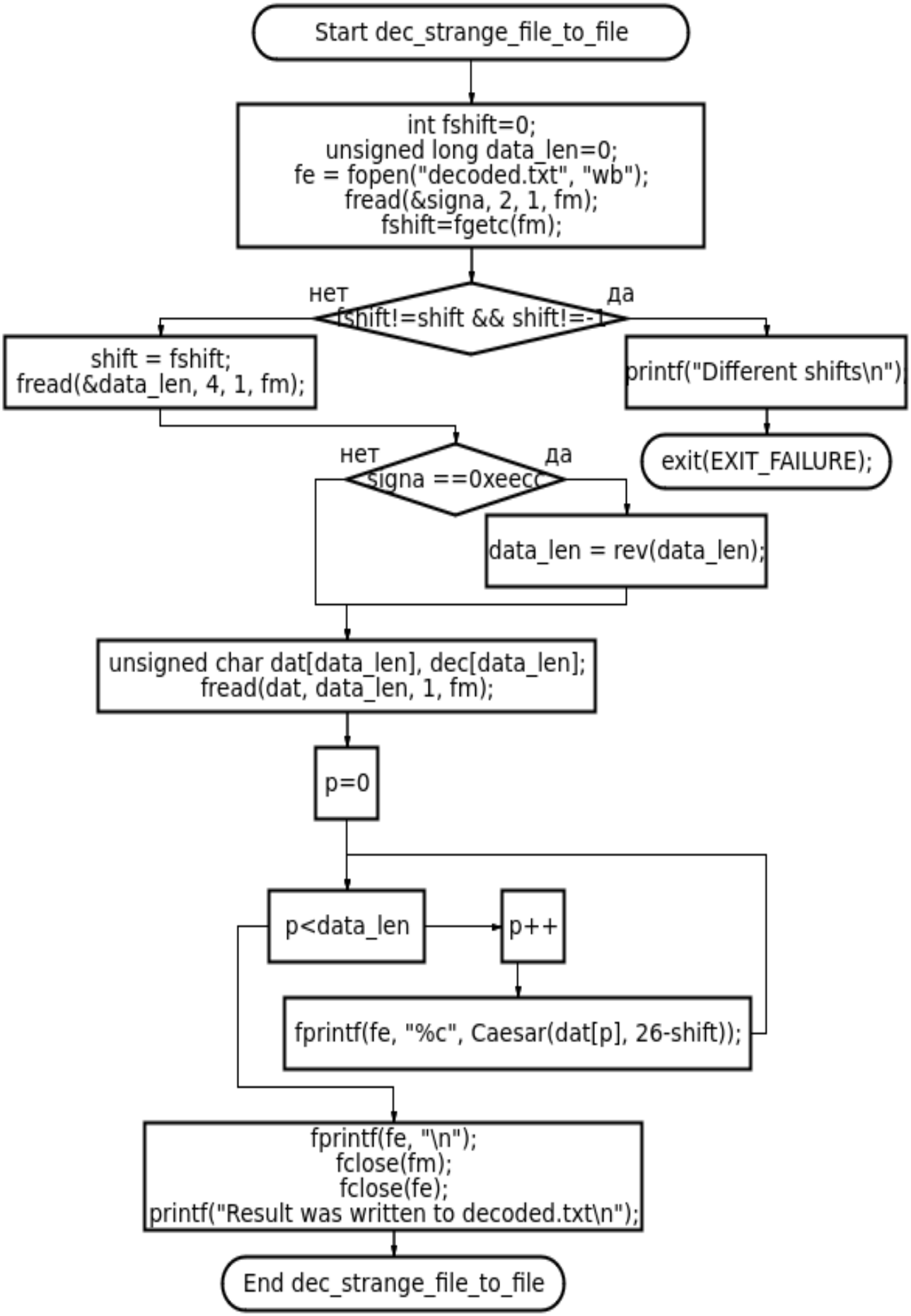
Блок-схема *decrypt_file_to_str_file*



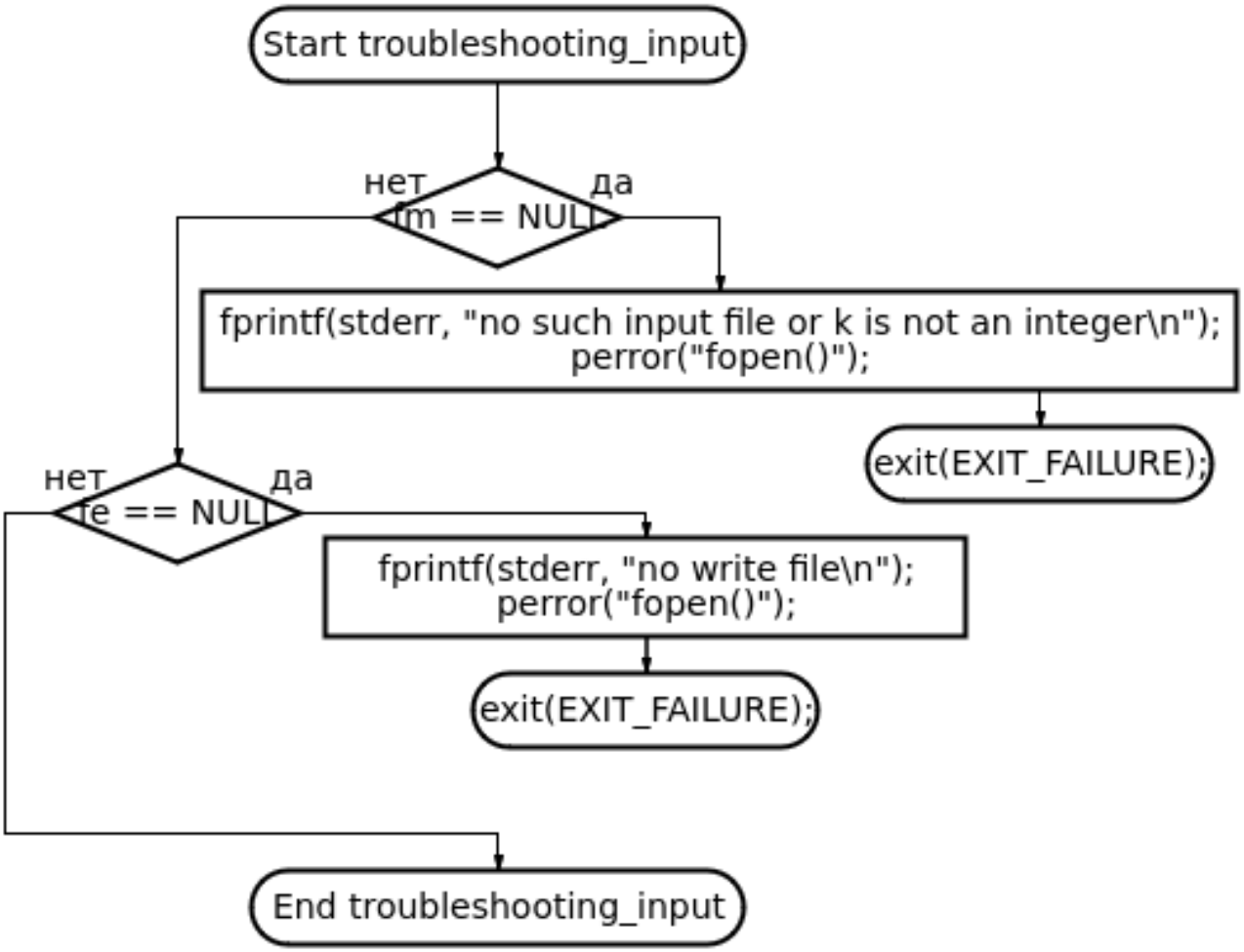
Блок-схема *dec_strange_file_to_strange_file*



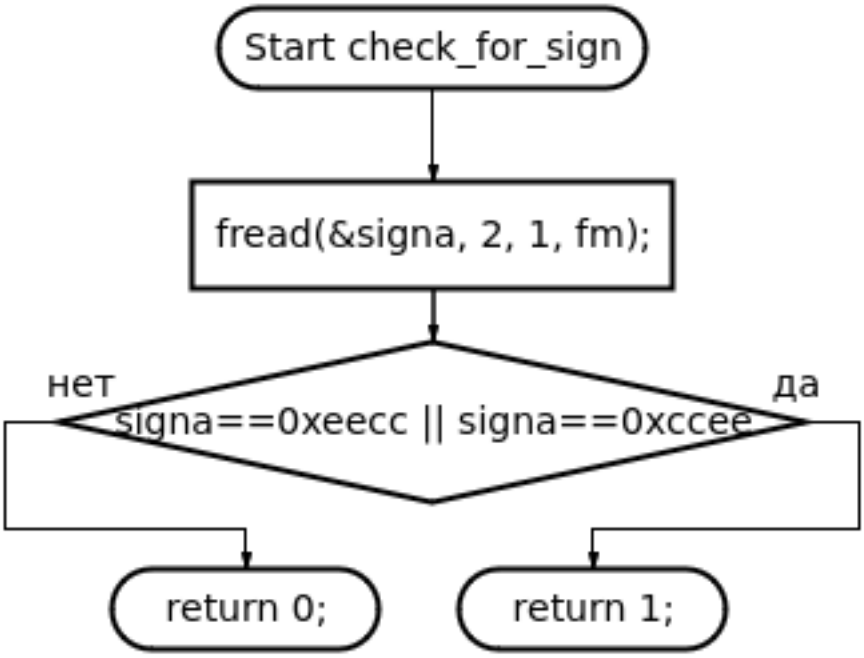
Блок-схема *dec_strange_file_to_file*



Блок-схема *troubleshooting_ input*



Блок-схема *check_for_sign*



Приложение Б

Код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

char ch;
int shift=-1, ind_f=-1, ind_d=-1,ind_c=-1, ind_k=-1, ind_b=-1, ind_t=-1, ind_sign=-1, ex=0, argccc=0;
unsigned long i=0;
char msg[256];
unsigned short signa;
unsigned char data[268435455], denc[268435455], enc[268435455]; /*4294967295*/
FILE *fe = NULL, *fm = NULL;

char Caesar(char ch, int shift){
    if (ch >= 'A' && ch <= 'Z')
        ch = ((ch - 'A') + shift) % 26 + 'A';
    else if (ch >= 'a' && ch <= 'z')
        ch = ((ch - 'a') + shift) % 26 + 'a';
    return ch;
}

void ask_for_shift(){
    printf("Input shift:");
scanf("%d", &shift);
getchar();
}

unsigned long rev(unsigned long len){
    return ((len & 0xFF) << 24) | (((len >> 8) & 0xFF) << 16) | (((len >> 16) & 0xFF) << 8) | ((len >> 24) & 0xFF);
}

unsigned long crc32(unsigned char *data, unsigned long data_len){
    unsigned int crc=0, i, j, size=0, crc_table[256];
    unsigned char c;
for (i=0; i<256; i++){
        crc = i<<24;
        for (j = 0; j<8; j++)
            crc = (crc & 0x80000000) ? (crc << 1)^ 0x04C11DB7 : crc << 1;
        crc_table[i] = crc;
    }
    crc =0UL;
    for (unsigned long i=0; i<data_len; i++){
    crc = crc_table[(crc >>24) ^data[i]] ^ (crc << 8);
size+=1;
}
while (size){
    crc = crc_table[(crc>>24) ^ (size & 0xFF)] ^ (crc << 8);
size >>=8;
}
return (unsigned long)(~crc);
}

void message_encryption(int shift){
    printf("Input message:");
while ((ch = getchar()) != '\n' && ch != EOF){
if (i++==0)
    printf("Encrypted message:");
printf("%c", Caesar(ch, shift));
}
printf("\n");
}

void message_encryption_to_str_file(int shift){
    printf("Input text:");
    unsigned long i=0, crc_val=0, len_out = 0;
    if (shift==-1)
        shift = 5;
    fe = fopen("hex_file_encoded.txt", "wb");
    if (ind_b!=-1)
    signa = 0xeecc;
else
    signa = 0xccee;
fwrite(&signa, 2, 1, fe);
fwrite(&shift, 1, 1, fe);
while ((ch = getchar()) != '\n'){
```

```
data[i] = Caesar(ch, shift);
i++;
}
len_out = i;
if (ind_b!=-1)
    len_out = rev(i);
fwrite(&i, 4, 1, fe);
fwrite(data, i, 1, fe);
crc_val = crc32(data, i);
if (ind_b!=-1)
    crc_val = rev(crc_val);
fwrite(&crc_val, 4, 1, fe);
printf("Result was written to  hex_file_encoded.txt\n");
}

void enc_strange_file_to_output(FILE* fm){
    int fshift=0;
    unsigned long data_len=0;
    fe = fopen("encoded.txt", "wb");
    fread(&signa, 2, 1, fm); /*0xccee || 0xeecc        2 bytes*/
    fshift=fgetc(fm);   /*shift                     1 byte*/
    if (fshift!=shift && shift!=-1){
        printf("Different shifts\n");
exit(EXIT_FAILURE);
    }
    shift = fshift;
    fread(&data_len, 4, 1, fm); /*data len          4 bytes*/
    if (signa ==0xeecc)
        data_len = rev(data_len);
    unsigned char dat[data_len], enc[data_len];
    fread(dat, data_len, 1, fm);
for (unsigned long p=0; p<data_len; p++)
        fprintf(fe, "%c", Caesar(dat[p], shift));
    fprintf(fe, "\n");
    fclose(fm);
    fclose(fe);
    printf("Result was written to encoded.txt\n");
}

void file_encryption(FILE* fm, FILE* fe, int shift){
    while ((ch = fgetc(fm))!=EOF)
fprintf(fe, "%c", Caesar(ch, shift));
}

void file_encryption_to_str_file(FILE* fm, int shift){
    unsigned long i=0, crc_val=0, len_out = 0;
    fe = fopen("hex_file_encoded.txt", "wb");
    while ((ch = fgetc(fm))!=EOF){
data[i] = ch;
i++;
}
if (shift==-1)
    shift = 5;
if (ind_b!=-1)
    signa = 0xeecc;
else
    signa = 0xccee;
fwrite(&signa, 2, 1, fe);
fwrite(&shift, 1, 1, fe);
len_out = i;
if (ind_b!=-1)
    len_out = rev(i);
fwrite(&len_out, 4, 1, fe);
for (int j =0; j<i; j++)
    enc[j] = Caesar(data[j], shift);
fwrite(enc, i, 1, fe);
crc_val = crc32(data, i);
if (ind_b!=-1)
    crc_val = rev(crc_val);
fwrite(&crc_val, 4, 1, fe);
printf("Result was written to hex_file_encoded.txt\n");
}

void enc_strange_file_to_strange_file(FILE* fm){
    int fshift=0;
    unsigned long data_len=0, len_out  =0;
    fe = fopen("hex_file_encoded.txt", "wb");
    fread(&signa, 2, 1, fm); /*0xccee || 0xeecc        2 bytes*/
    fshift=fgetc(fm);   /*shift                     1 byte*/
    if (fshift!=shift && shift!=-1){
```

```
        printf("Different shifts\n");
exit(EXIT_FAILURE);
    }
    shift = fshift;
    fread(&data_len, 4, 1, fm); /*data len          4 bytes*/
    if (signa == 0xeecc)
        data_len = rev(data_len);
    unsigned char dat[data_len], enc[data_len];
    fread(dat, data_len, 1, fm);
    if (ind_b!=-1)
        signa = 0xeecc;
    else
        signa = 0xccee;
    fwrite(&signa, 2, 1, fe);
fputc(fshift, fe);
len_out = data_len;
if (ind_b!=-1)
    len_out = rev(data_len);
fwrite(&len_out, 4, 1, fe);
for (unsigned long p=0; p<data_len; p++){
    ch = Caesar(dat[p], shift);
        fwrite(&ch, 1, 1, fe);
    }
    unsigned long crc_value = crc32(dat, data_len);
    if (ind_b!=-1)
        crc_value = rev(crc_value);
    fwrite(&crc_value, 4, 1, fe);
    fclose(fm);
    fclose(fe);
    printf("Result was written to hex_file_encoded.txt\n");
}

int is_number(char* str){
    for (int i=0; *(str+i)!='\0' && *(str+i)!='\n'; i++){
        if (*(str+i) < '0' || *(str+i)>'9')
            return 0;
    }
    return 1;
}

void decrypt_input(){
    printf("Input message:");
int i = 0;
while ((ch = getchar()) != '\n' && ch != EOF){
    msg[i]=ch;
    i++;
}
    for (int j =1; j<26; j++){
    for (int k=0; k<i; k++){
    if (k==0)
        printf("If shift=%d:", j);
    printf("%c", Caesar(msg[k],j));
}
printf("\n");
}
}

void decrypt_input_to_str_file(int shift){
    unsigned long i=0, crc_val=0, len_out = 0;
    fe = fopen("hex_file_decoded.txt", "wb");
    printf("Input message:");
while ((ch = getchar()) != '\n'){
    data[i] = ch;
    i++;
}
if (shift==-1)
    shift = 5;
if (ind_b!=-1)
    signa = 0xeecc;
else
    signa = 0xccee;
fwrite(&signa, 2, 1, fe);
fwrite(&shift, 1, 1, fe);
if (ind_b!=-1)
    len_out = rev(i);
fwrite(&len_out, 4, 1, fe);
    for (int j =0; j<i; j++){
    ch = Caesar(data[j], 26-shift);
        fwrite(&ch,1 ,1 ,fe);
    }
```

```
crc_val = crc32(data, i);
if (ind_b!=-1)
    crc_val = rev(crc_val);
fwrite(&crc_val, 4, 1, fe);
printf("Result was written to hex_file_decoded.txt\n");
}

void decrypt_input_with_shift(int shift){
    printf("Input message:");
while ((ch = getchar()) != '\n' && ch != EOF){
    if (i==0){
        printf("Decoded message:");
        i=1;
    }
printf("%c", Caesar(ch, 26-shift));
}
printf("\n");
}

void decrypt_file(FILE* fm, FILE* fe){
    int i = 0;
while ((ch = fgetc(fm))!=EOF){
msg[i]=ch;
    i++;
}
    for (int j =1; j<26; j++){
    for (int k=0; k<i; k++){
    if (k==0)
        fprintf(fe,"If shift=%d:", j);
    fprintf(fe,"%c", Caesar(msg[k],j));
}
fprintf(fe, "\n");
}
}

void decrypt_file_with_shift(FILE* fm, int shift){
    fe = fopen("decoded.txt", "w");
while ((ch = fgetc(fm))!=EOF)
fprintf(fe,"%c", Caesar(ch, 26-shift));
    fprintf(fe, "\n");
    printf("Result was written to decoded.txt\n");
}

void decrypt_file_to_str_file(FILE* fm, int shift){
    unsigned long i=0, crc_val=0, len_out=0;
    fe = fopen("hex_file_decoded.txt", "wb");
if (shift==-1)
    shift = 5;
if (ind_b!=-1)
    signa = 0xeecc;
else
    signa = 0xccee;
fwrite(&signa, 2, 1, fe);
fwrite(&shift, 1, 1, fe);
while ((ch = fgetc(fm))!=EOF ){
data[i] =  ch;
i++;
}
len_out = i;
if (ind_b!=-1)
    len_out = rev(i);
fwrite(&len_out, 4, 1, fe);
    for (int j =0; j<i; j++){
        ch = Caesar(data[j], 26-shift);
    fwrite(&ch, 1, 1, fe);
}
crc_val = crc32(data, i);
if (ind_b!=-1)
    crc_val = rev(crc_val);
fwrite(&crc_val, 4, 1, fe);
printf("Result was written to hex_file_decoded.txt\n");
}

void dec_strange_file_to_strange_file(FILE* fm){
    int fshift=0;
    unsigned long data_len=0, len_out=0;
    fe = fopen("hex_file_decoded.txt", "wb");
    fread(&signa, 2, 1, fm); /*0xccee || 0xeecc        2 bytes*/
    fshift=fgetc(fm);   /*shift                    1 byte*/
    if (fshift!=shift && shift!=-1){
```

```
        printf("Different shifts\n");
exit(EXIT_FAILURE);
    }
    shift = fshift;
    fread(&data_len, 4, 1, fm); /*data len          4 bytes*/
    if (signa==0xeecc)
    data_len = rev(data_len);
    unsigned char dat[data_len], dec[data_len];
    fread(dat, data_len, 1, fm);
    if (ind_b!=-1)
    signa = 0xeecc;
else
    signa = 0xccee;
    fwrite(&signa, 2, 1, fe);
fputc(fshift, fe);
len_out = data_len;
if (ind_b!=-1)
    len_out = rev(data_len);
fwrite(&len_out, 4, 1, fe);
for (unsigned long p=0; p<data_len; p++){
    ch = Caesar(dat[p], 26-shift);
        fwrite(&ch, 1, 1, fe);
    }
    unsigned long crc_value = crc32(dat, data_len);
    if (ind_b!=-1)
        crc_value = rev(crc_value);
    fwrite(&crc_value, 4, 1, fe);
    fclose(fm);
    fclose(fe);
    printf("Result was written to hex_file_decoded.txt\n");
}

void dec_strange_file_to_file(FILE* fm){
    int fshift=0;
    unsigned long data_len=0;
    fe = fopen("decoded.txt", "wb");
    fread(&signa, 2, 1, fm); /*0xccee || 0xeecc        2 bytes*/
    fshift=fgetc(fm);   /*shift                     1 byte*/
    if (fshift!=shift && shift!=-1){
        printf("Different shifts\n");
exit(EXIT_FAILURE);
    }
    shift = fshift;
    fread(&data_len, 4, 1, fm); /*data len          4 bytes*/
    if (signa ==0xeecc)
    data_len = rev(data_len);
    unsigned char dat[data_len], dec[data_len];
    fread(dat, data_len, 1, fm);
for (unsigned long p=0; p<data_len; p++)
        fprintf(fe, "%c", Caesar(dat[p], 26-shift));
    fprintf(fe, "\n");
    fclose(fm);
    fclose(fe);
    printf("Result was written to decoded.txt\n");
}

void troubleshooting_input(FILE* fm, FILE* fe){
    if (fm == NULL){
fprintf(stderr, "no such input file or k is not an integer\n");
        perror("fopen()");
exit(EXIT_FAILURE);
    }
    if (fe == NULL){
        fprintf(stderr, "no write file\n");
        perror("fopen()");
        exit(EXIT_FAILURE);
    }
}

int check_for_sign(FILE* fm){
    fread(&signa, 2, 1, fm);
    if (signa==0xeecc || signa==0xccee)
    return 1;
return 0;
}

int main(int argc, char **argv){
    for (int i = 1; i < argc; i++) {
        switch (*argv[i]) {
            case 'c':
```

```
                ind_c = i;
                ex++;
                argccc++;
                break;
            case 'd':
                ind_d = i;
                ex++;
                argccc++;
                break;
            case 'b':
                ind_b = i;
                argccc++;
                break;
            case '-':
                ind_t = i;
                argccc++;
                break;
            default:
                if (is_number(argv[i])) {
                    ind_k = i;
                    ex++;
                    shift = atoi(argv[ind_k]);
                    argccc++;
                } else {
                    fm = fopen(argv[i], "r");
                    if (fm != NULL) {
                        if (check_for_sign(fm)) {
                            fclose(fm);
                            ind_sign = i;
                            fm = fopen(argv[ind_sign], "rb");
                            ex++;
                            argccc++;

                        } else {
                            fclose(fm);
                            ind_f = i;
                            fm = fopen(argv[ind_f], "r");
                            ex++;
                            argccc++;
                        }
                    }
                }
        }
    }
    /*printf("%d %d %d %d %d %d %d\n", ind_f, ind_d,ind_c, ind_k, ind_b, ind_t, ind_sign);*/
if (argc>5 || (ind_d!=-1 && ind_c!=-1)){
fprintf(stderr, "usage: %s [c/d] [k] [file] [-] [b]\n", argv[0]);
exit(EXIT_FAILURE);
}
else if (argc!=argccc+1){
    fprintf(stderr, "no such file or key\n", argv[0]);
exit(EXIT_FAILURE);
}
else if (ind_f!=-1 && ind_sign!=-1){
    fprintf(stderr, "something wrong with file\n", argv[0]);
exit(EXIT_FAILURE);
}
else if (ind_t!=-1){
    if (ex==0 || (ex==1 && ind_c!=-1)){
    ask_for_shift();
    message_encryption(shift);
    } else if (ex==1 || (ex==2 && ind_c!=-1)){
            if (ind_d!=-1)
                decrypt_input();
            else if (ind_sign!=-1){
                enc_strange_file_to_output(fm);
            } else if (ind_f!=-1){
                fm = fopen(argv[ind_f],"r");
                fe = fopen("enc.txt", "w");
                troubleshooting_input(fm, fe);
                ask_for_shift();
        file_encryption(fm, fe, shift);
        fclose(fm);
        fclose(fe);
            } else if (ind_k!=-1)
        message_encryption(shift);
    }
    else if (ex == 2 || (ex == 3 && ind_c!=-1)) { /*ind_d/c, ind_f/sign, ind_k*/
        if (ind_d!=-1){
            if (ind_k!=-1)
```

```
        decrypt_input_with_shift(shift);
     else if (ind_f!=-1){
         fe = fopen("dec.txt", "w");
            decrypt_file(fm, fe);
            fclose(fm);
                fclose(fe);
     } else if (ind_sign!=-1)
         dec_strange_file_to_file(fm);
        }
        else if (ind_k!=-1){
            if (ind_f!=-1){
                fe = fopen("enc.txt", "w");
                troubleshooting_input(fm, fe);
         file_encryption(fm, fe, shift);
         fclose(fm);
         fclose(fe);
            } else if (ind_sign!=-1)
                enc_strange_file_to_output(fm);
        }
    }
    else if (ex == 3 ){
        if (ind_d!=-1){
            if (ind_f!=-1)
                decrypt_file_with_shift(fm, shift);
            else if (ind_sign!=-1)
                dec_strange_file_to_file(fm);
        }
    }
}
else if (ind_t==-1){
    if (ex==0 || (ex==1 && ind_c!=-1))
        message_encryption_to_str_file(shift);
    else if (ex==1 || (ex==2 && ind_c!=-1)){
            if (ind_d!=-1)
                decrypt_input_to_str_file(shift);
            else if (ind_sign!=-1)
                enc_strange_file_to_strange_file(fm);
            else if (ind_f!=-1)
                file_encryption_to_str_file(fm, shift);
            else if (ind_k!=-1)
                message_encryption_to_str_file(shift);
    }
    else if (ex == 2 || (ex == 3 && ind_c!=-1)) { /*ind_d/c, ind_f/sign, ind_k*/
        if (ind_d!=-1){
            if (ind_k!=-1)
                decrypt_input_to_str_file(shift);
     else if (ind_f!=-1)
         decrypt_file_to_str_file(fm, shift);
     else if (ind_sign!=-1)
            dec_strange_file_to_strange_file(fm);
        }
        else if (ind_k!=-1){
            if (ind_f!=-1)
                file_encryption_to_str_file(fm, shift);
            else if (ind_sign!=-1)
                enc_strange_file_to_strange_file(fm);
        }
    }
    else if (ex == 3 ){
        if (ind_d!=-1){
            if (ind_f!=-1)
                decrypt_file_to_str_file(fm, shift);
            else if (ind_sign!=-1)
                dec_strange_file_to_strange_file(fm);
        }
    }
}
return 0;
}
```