Правительство Российской Федерации

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

Кафедра «Компьютерная безопасность»

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №3

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 №2 следующим образом: если при запуске программы ей передается параметр командной строки ‘d‘, то выполняется дешифрование; если же параметр ‘d‘ отсутствует либо передается параметр ‘c‘, то выполняется шифрование.

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

1. Изменила тип возвращаемого значения функции ask_for_shit на void из-за глобальности переменной shift

2. Вынесла часть кода, отвечающую за шифрование сообщения в отдельную функцию void message_encryption, принимающую в качестве аргумента перемнную shift, отвечающую за сдвиг при шифровании

3. Вынесла часть кода, отвечающую за шифрование файла, в отдельную функцию void file_encryption, принимающую в качестве аргументов указатель на начало файла с сообщением fm, указатель на файл, куда записывается зашифрованное сообщение fe и сдвиг

4. Написала функцию void decrypt_input, которая расшифровывает сообщение, если ей не передано сдвига

5. Написала функцию void decrypt_input_with_shift, которая расшифровывает сообщение, принимающую в качестве аргумента сдвиг shift

6. Написала функцию void decrypt_file, которая расшифровывает файл, если ей не дано сдвига, принимающую на вход указатель на начало файла с сообщением fm и с указателем на файл, в который надо записать варианты расшифрованного сообщения fe

7. Написала функцию void decrypt_file_with_shift, которая расшифровывает файл по заданному сдвигу. Принимает в качестве аргументов указатель на начало файла с сообщением fm, указатель на начало файла, в который надо записать расшифрованное сообщение fe и сдвиг shift.

2 Описание функции main

2.1 Ввод и работа с аргументами

- 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

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 сдивг shift. После чего, расшифровывает файл с данным сдвигом и записывает в файл для расшифрованного сообщения

3 Тестирование

3.1 Проверка работы с файлом

Запишем в файл mes.txt "Hello проверим, что файла enc.txt изначально не было. Запустим программу с аргументом mes.txt, введём сдвиг 5. Появился файл enc.txt с зашифрованным по алгоритму шифра Цезаря со сдвигом 5 словом "Hello".

```
$ cat mes.txt
Hello
$ ls
a.out  Caesar.c  mes.txt  README.md
$ ./a.out mes.txt
Input shift:5
$ cat enc.txt
Mjqqt
```

3.2 Проверка работы с пользовтельским вводом

После запуска программы пользователся просят ввести сдвиг и сообщение, после чего выводится "Encrypted message:"и сообщение, зашифрованное сообщение с ведённым сдвигом по алгоритму шифрования Цезаря.

```
$ ./a.out
Input shift:5
Input message:Hello
Encrypted message:Mjqqt
```

3.3 Проверка работы с целым числом без файла

Если пользователь вводит в качестве параметра целое число, то программа просит его ввести сообщение для шифрования и выводит зашифрованное, со сдвигом на подаваемое число, сообщение.

```
$ ./a.out  5
Input message:Hello
Encrypted message:Mjqqt
```

3.4 Проверка работы с целым числом и файлом

Если пользователь вводит в качестве параметров целое число и файл, то программа создаёт файл enc.txt, если такового не было, и записывает в него зашифрованное сообщение.

```
$ ls
a.out  CoolCaesar.c  msg.txt  README.md
$ ./a.out  msg.txt 5
$ ls
a.out  CoolCaesar.c  enc.txt  msg.txt  README.md
$cat enc.txt
Mjqqt
```

3.5 Проверка работы с ключом d

Если пользователь вводит только ключ d, то программы перебирает все сдвиги

```
$./a.out d
Input message:Hello
If shift=1:Ifmmp
If shift=2:Jgnnq
If shift=3:Khoor
If shift=4:Lipps
If shift=5:Mjqqt
If shift=6:Nkrru
If shift=7:Olssv
If shift=8:Pmttw
If shift=9:Qnuux
If shift=10:Rovvy
If shift=11:Spwwz
If shift=12:Tqxxa
If shift=13:Uryyb
If shift=14:Vszzc
If shift=15:Wtaad
If shift=16:Xubbe
If shift=17:Yvccf
If shift=18:Zwddg
If shift=19:Axeeh
If shift=20:Byffi
If shift=21:Czggj
If shift=22:Dahhk
If shift=23:Ebiil
If shift=24:Fcjjm
If shift=25:Gdkkn
```

3.6 Проверка работы с ключом d и файлом

Если пользователь вводит d и имя файла, то программа перебирает все сдвиги и записывает их в enc.txt

```
$cat msg.txt
Xtrj yjcy yt jsh
```

```
$./a.out msg.txt d
$cat enc.txt
If shift=1:Yusk zkdz zu kti

If shift=2:Zvtl alea av luj

If shift=3:Awum bmfb bw mvk

If shift=4:Bxvn cngc cx nwl

If shift=5:Cywo dohd dy oxm

If shift=6:Dzxp epie ez pyn

If shift=7:Eayq fqjf fa qzo

If shift=8:Fbzr grkg gb rap

If shift=9:Gcas hslh hc sbq

If shift=10:Hdbt itmi id tcr

If shift=11:Iecu junj je uds

If shift=12:Jfdv kvok kf vet

If shift=13:Kgew lwpl lg wfu

If shift=14:Lhfx mxqm mh xgv

If shift=15:Migy nyrn ni yhw

If shift=16:Njhz ozso oj zix

If shift=17:Okia patp pk ajy

If shift=18:Pljb qbuq ql bkz

If shift=19:Qmkc rcvr rm cla

If shift=20:Rnld sdws sn dmb

If shift=21:Some text to enc

If shift=22:Tpnf ufyu up fod

If shift=23:Uqog vgzv vq gpe

If shift=24:Vrph whaw wr hqf
```

```
If shift=25:Wsqi xibx xs irg
```

3.7 Проверка работы с ключом d и числом

Если пользователь ввёл ключ d и сдвиг, то сообщение декодируется в соответствии с заданным сдвигом

```
$./a.out 5
Input message:Hello
Encrypted message:Mjqqt
$./a.out 5 d
Input message:Mjqqt
Decoded message:Hello
```

3.8 Проверка работы с ключом d, числом и файлом

Если пользователь ввёл ключ d, сдвиг и файл, то файл расшифровывается с соответстствии со сдвигом

```
$cat msg.txt
Xtrj yjcy yt jsh
$./a.out msg.txt 5 d
$cat enc.txt
Some text to enc
```

3.9 Проверка вывода сообщения об ошибке открытия файла

Если файла, который пользователь указал в аргументе к программе нет, то программа выводит ошибку с соответсвующим предупреждением.

```
$ ./a.out what.txt
no such file
fopen(): No such file or directory
```

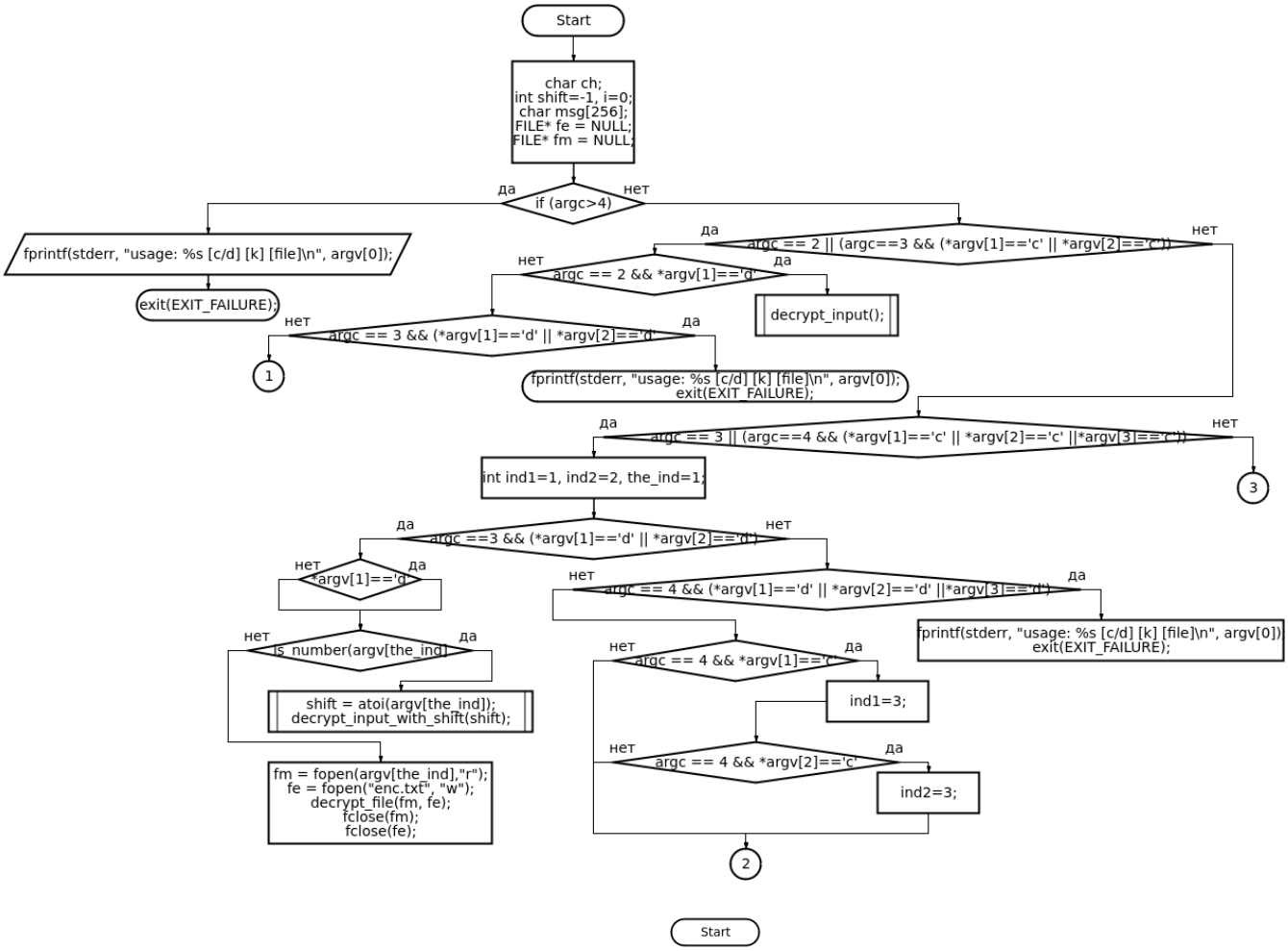
3.10 Проверка вывода ошибки о нецелом числе или отсутствии файла для чтения

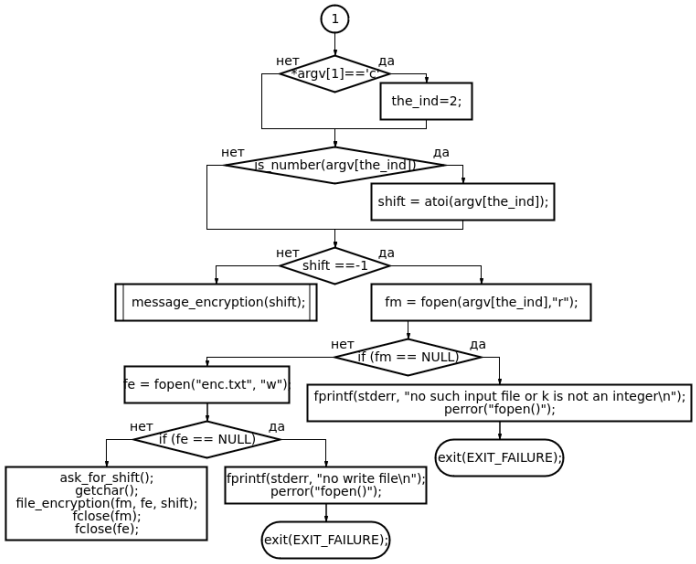
Если пользователь вводит в качестве параметра нецелое число, то программа должна завершить работу с ошибкой.

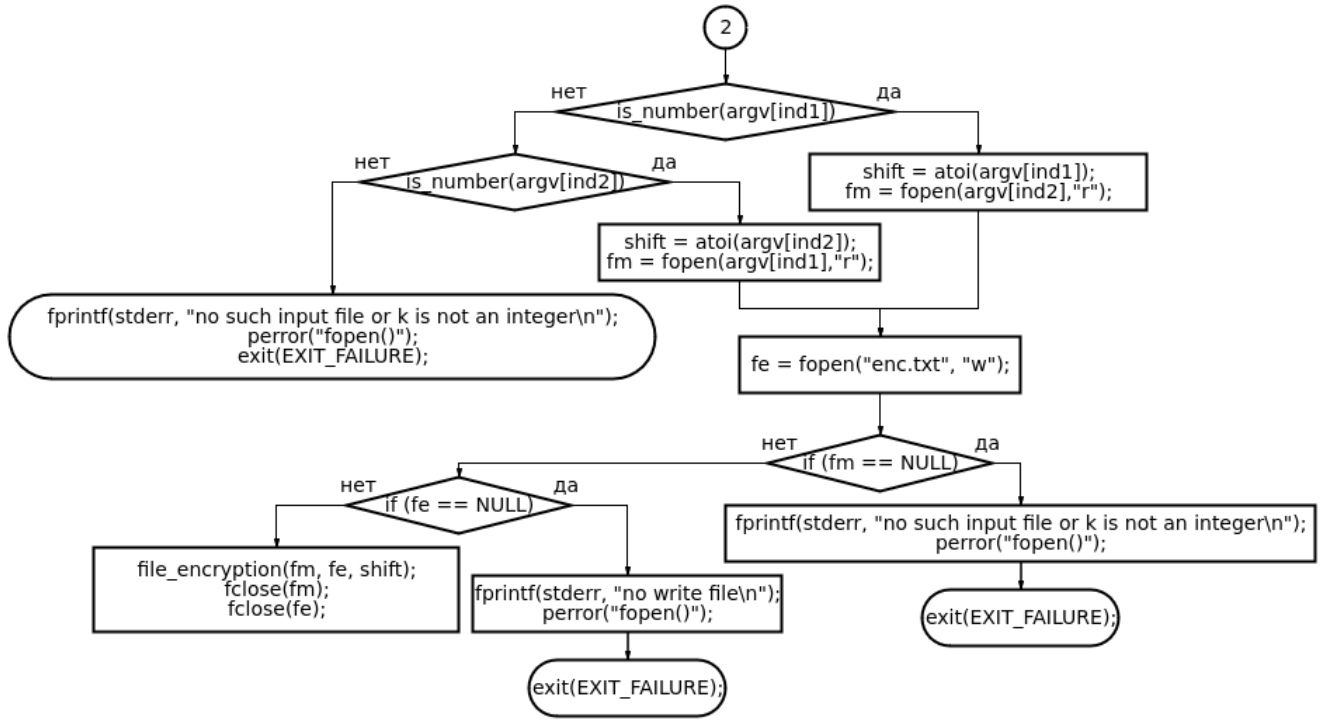
```
$ ./a.out  msg.txt 5.5
no such input file or k is not an integer
fopen(): Success

$ ./a.out  5.5
no such input file or k is not an integer
fopen(): No such file or directory
```

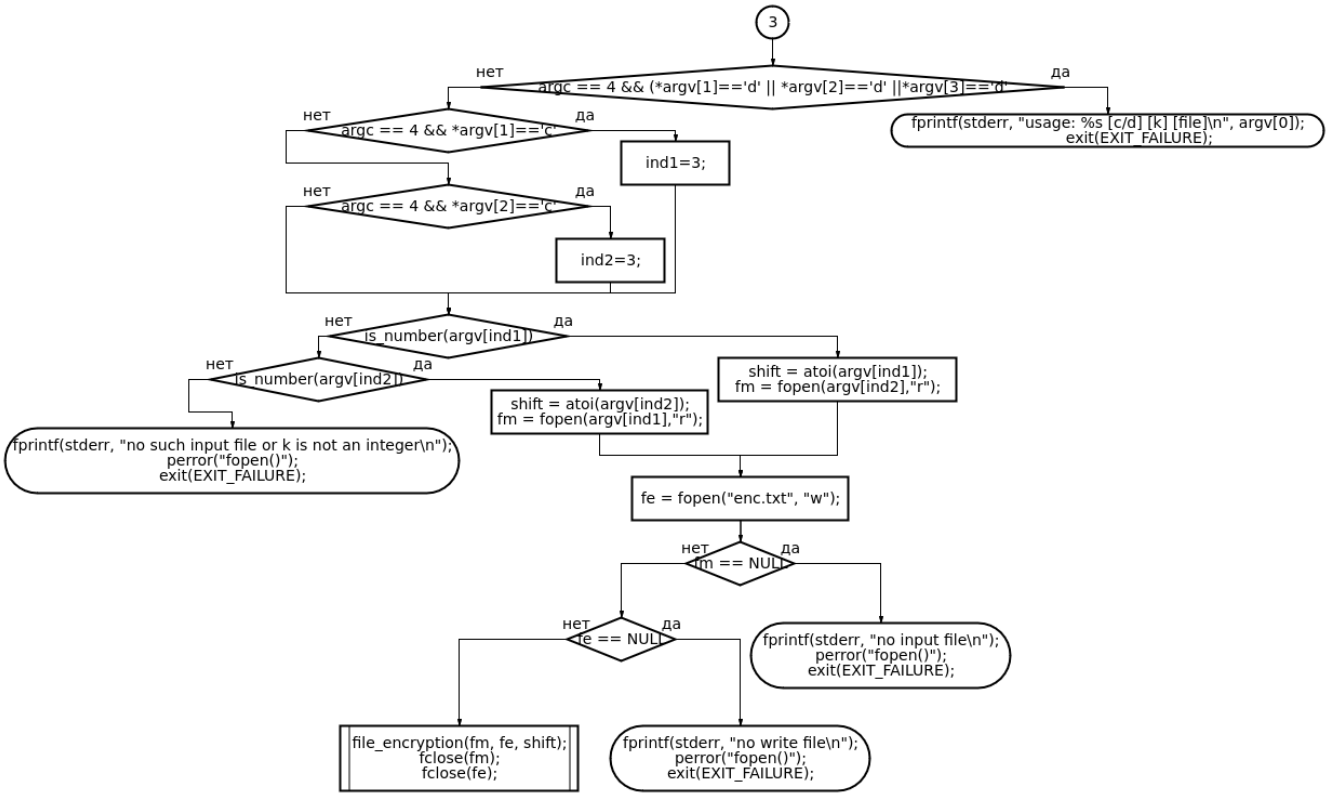
Приложение А

Блок-схемы *main*

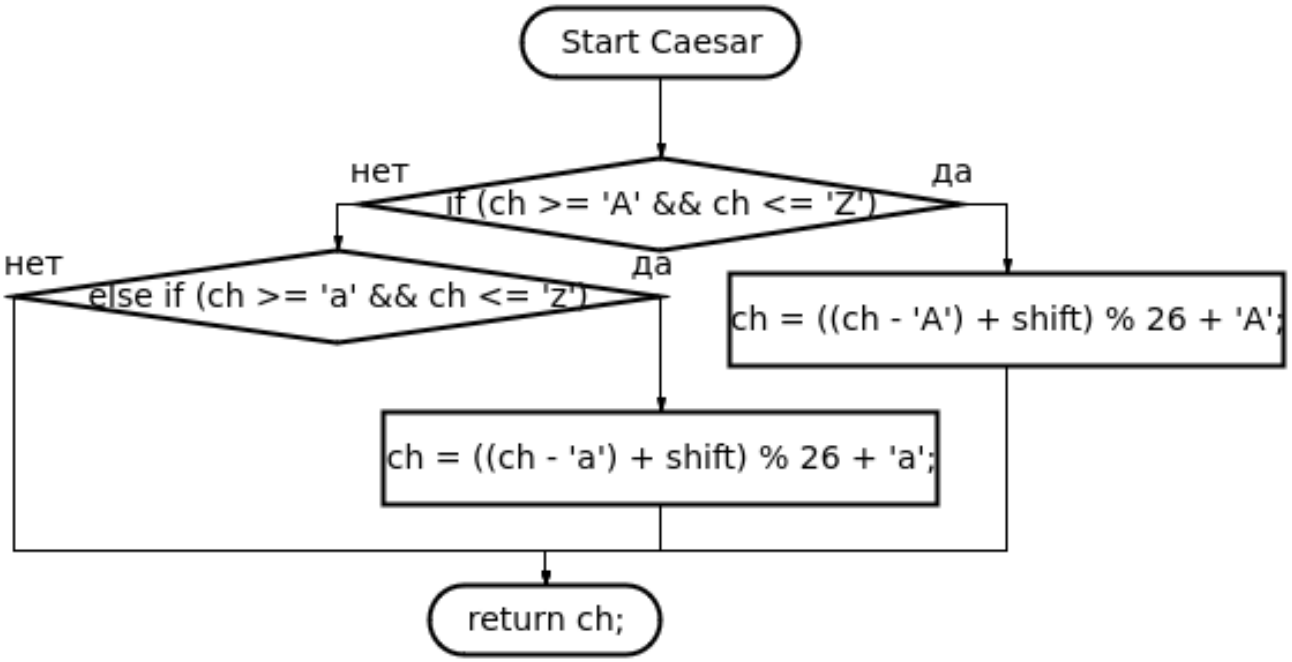




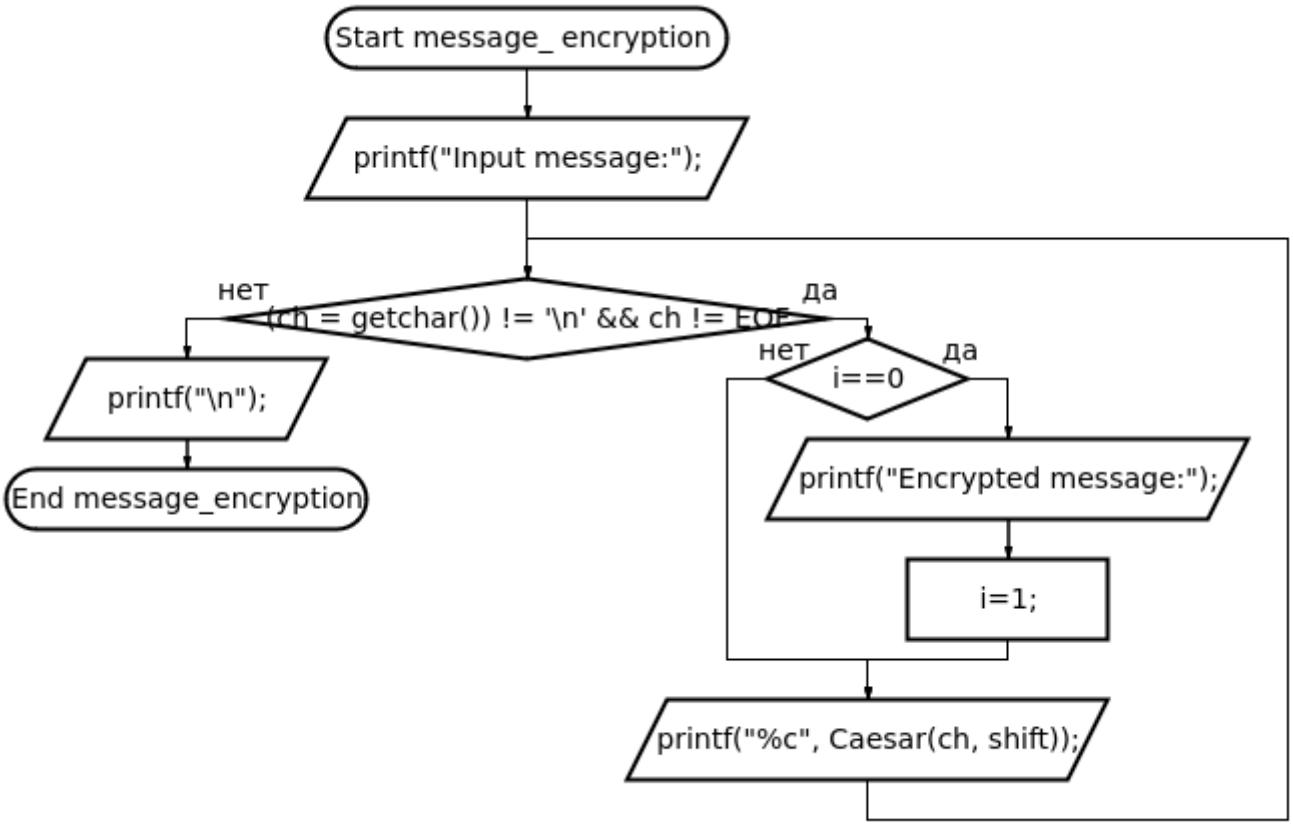

2
нет
is_number(argv[ind1])
да
нет
is_number(argv[ind2])
да
shift = atoi(argv[ind1]);
fm = fopen(argv[ind2],"r");
shift = atoi(argv[ind2]);
fm = fopen(argv[ind1],"r");
fprintf(stderr, "no such input file or k is not an integer\n");
perror("fopen()");
exit(EXIT_FAILURE);
fe = fopen("enc.txt", "w");
нет
if (fm == NULL)
да
нет
if (fe == NULL)
да
fprintf(stderr, "no such input file or k is not an integer\n");
perror("fopen()");
file_encryption(fm, fe, shift);
fclose(fm);
fclose(fe);
fprintf(stderr, "no write file\n");
perror("fopen()");
exit(EXIT_FAILURE);
exit(EXIT_FAILURE);


3
нет
argc == 4 && (*argv[1]=='d' || *argv[2]=='d' ||*argv[3]=='d'
да
fprintf(stderr, "usage: %s [c/d] [k] [file]\n", argv[0]);
exit(EXIT_FAILURE);
нет
argc == 4 && *argv[1]=='c'
да
ind1=3;
нет
argc == 4 && *argv[2]=='c'
да
ind2=3;
нет
is_number(argv[ind1])
да
нет
is_number(argv[ind2])
да
shift = atoi(argv[ind1]);
fm = fopen(argv[ind2],"r");
shift = atoi(argv[ind2]);
fm = fopen(argv[ind1],"r");
fprintf(stderr, "no such input file or k is not an integer\n");
perror("fopen()");
exit(EXIT_FAILURE);
fe = fopen("enc.txt", "w");
нет
fm == NULL
да
нет
fe == NULL
да
fprintf(stderr, "no input file\n");
perror("fopen()");
exit(EXIT_FAILURE);
file_encryption(fm, fe, shift);
fclose(fm);
fclose(fe);
fprintf(stderr, "no write file\n");
perror("fopen()");
exit(EXIT_FAILURE);

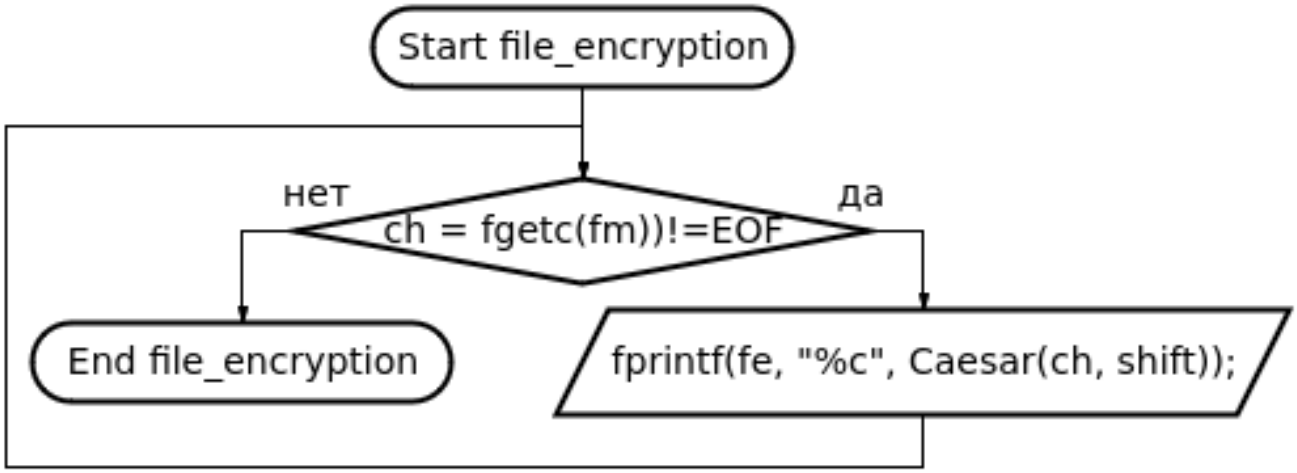
Блок-схема *Caesar*



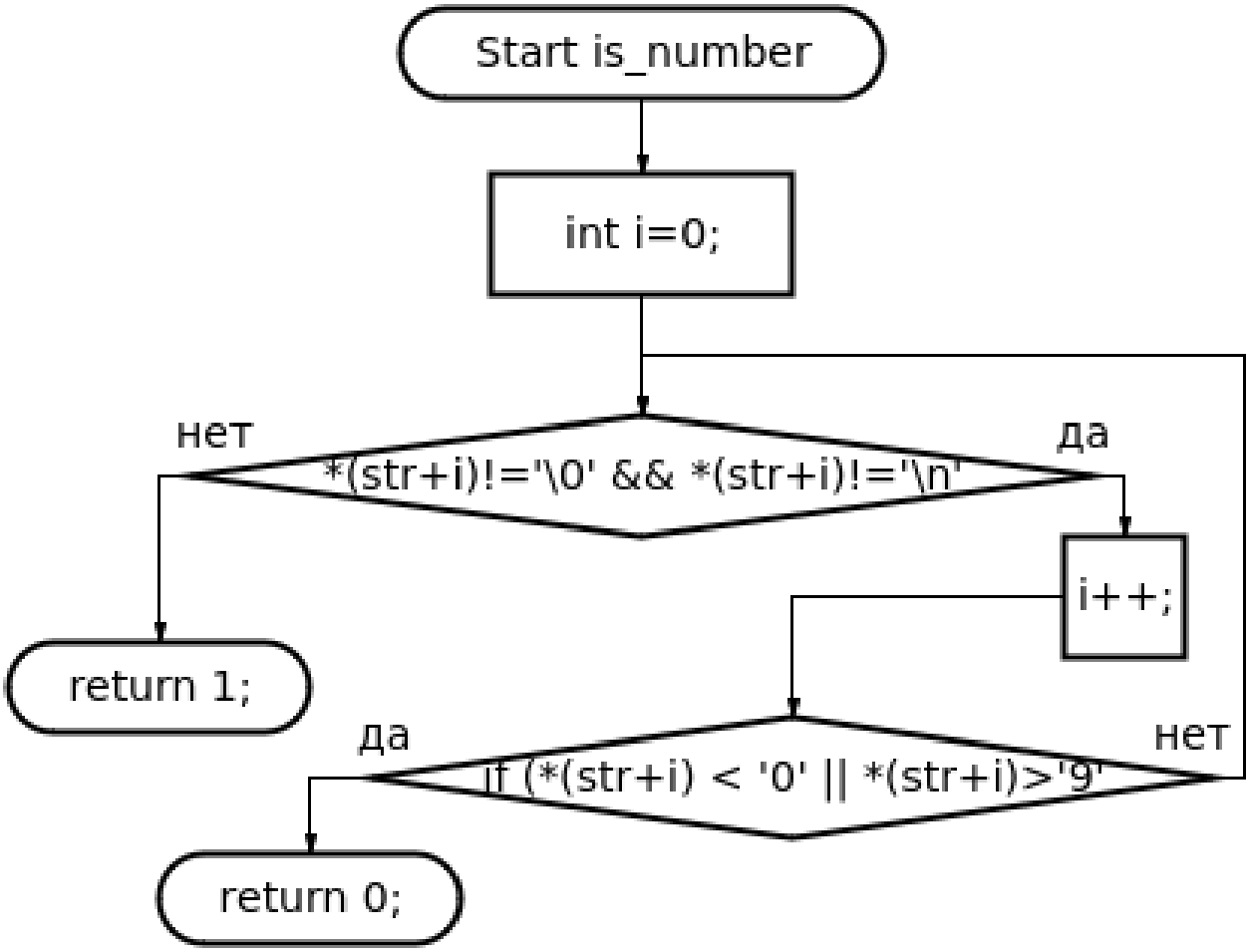
Блок-схема *message_ encryption*



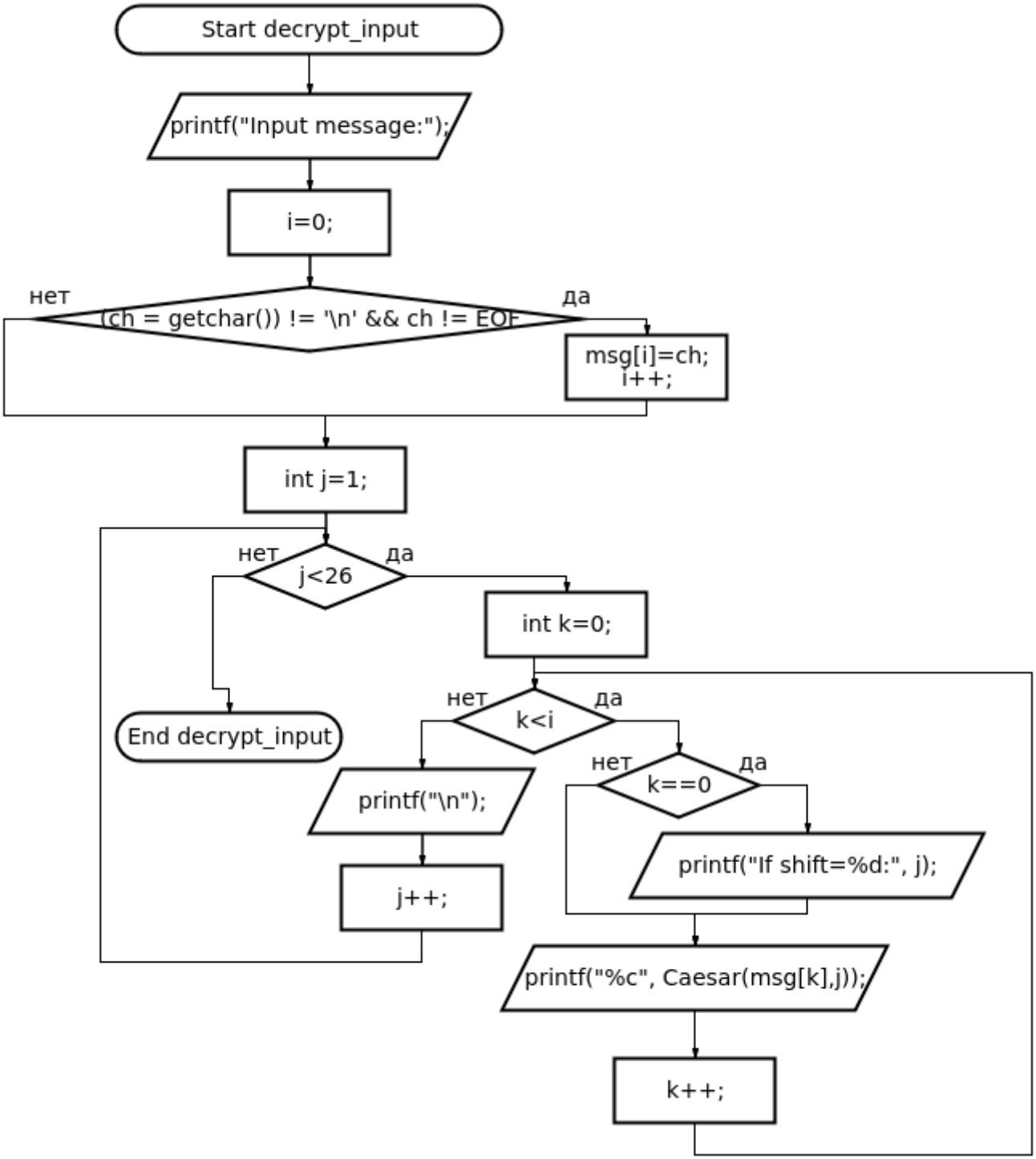
Блок-схема *file_ encryption*



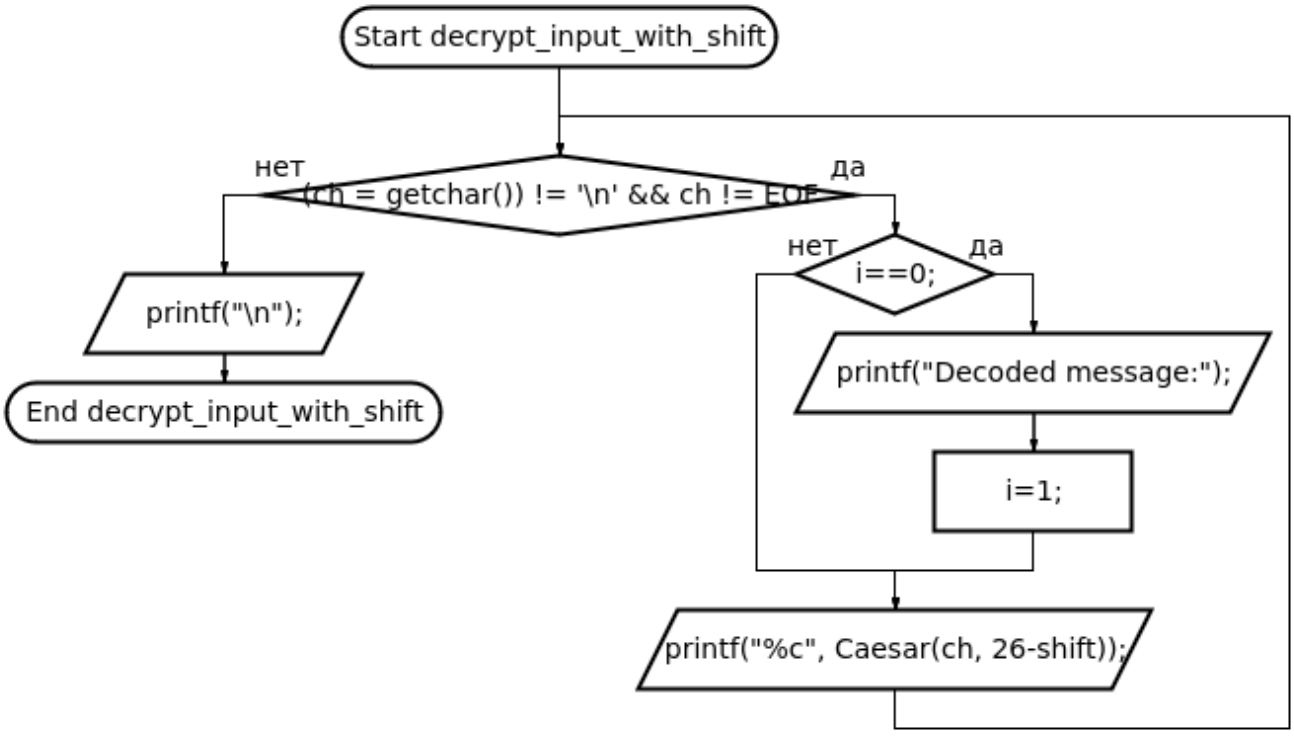
Блок-схема *is_ number*



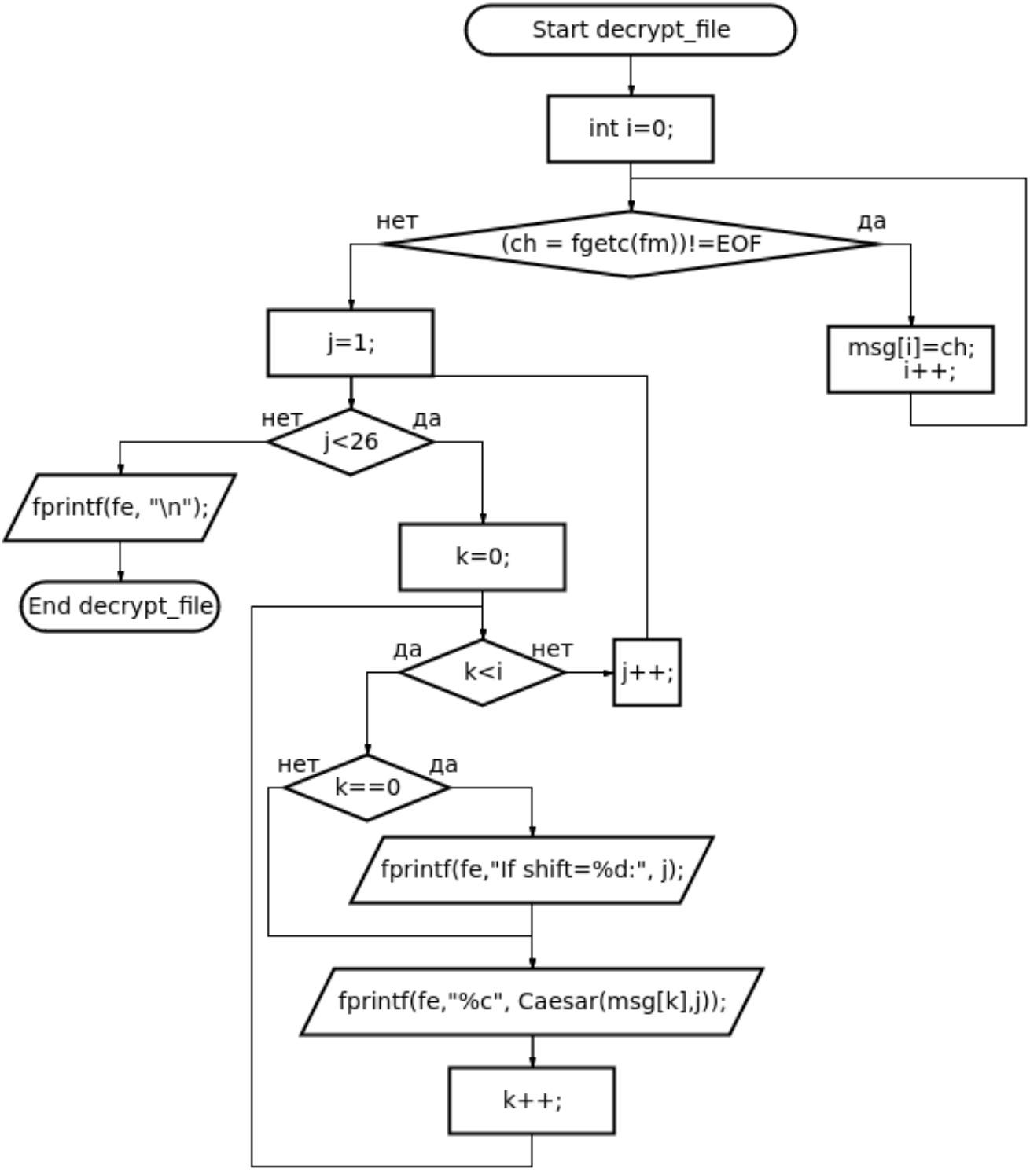
Блок-схема *decrypt_input*



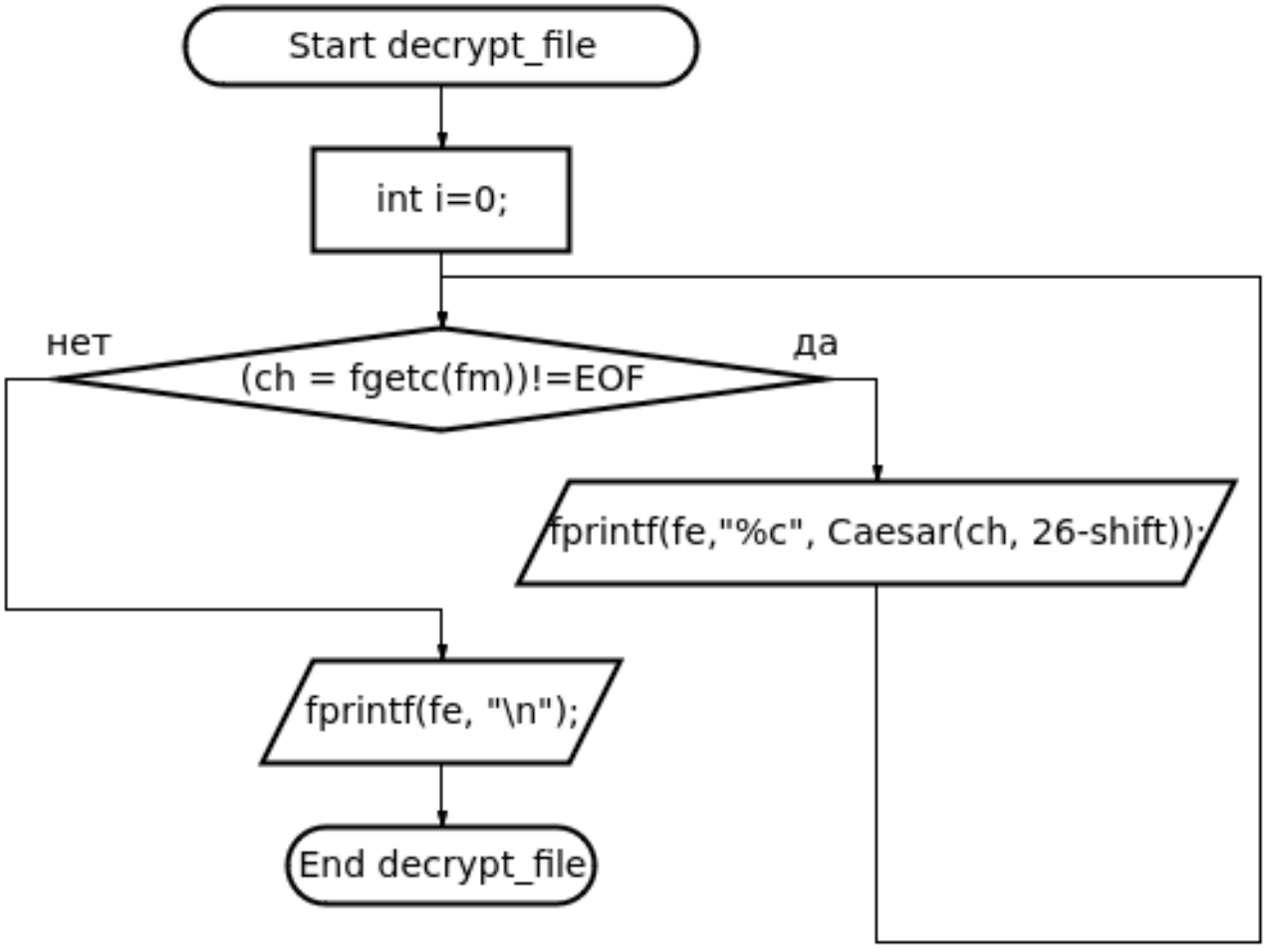
Блок-схема *decrypt_input_with_shift*



Блок-схема *decrypt_file*



Блок-схема *decrypt_file_with_shift*



Приложение Б

Код программы

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

char ch;
int shift=-1, i=0;
char msg[256];
FILE* fe = NULL;
FILE* 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);
}

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

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

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_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, FILE* fe, int shift){
while ((ch = fgetc(fm))!=EOF){
fprintf(fe,"%c", Caesar(ch, 26-shift));
}
    fprintf(fe, "\n");
}

int main(int argc, char **argv){

if (argc>4){
fprintf(stderr, "usage: %s [c/d] [k] [file]\n", argv[0]);
exit(EXIT_FAILURE);
}
else if (argc == 1 || (argc == 2 && *argv[1]=='c')){ /* ./a.out      ./a.out c */
ask_for_shift();
getchar();
message_encryption(shift);
}
else if (argc == 2 || (argc==3 && (*argv[1]=='c' || *argv[2]=='c'))){  /*./a.out d/file/k [c]*/
        if (argc == 2 && *argv[1]=='d') /*./a.out d */
            decrypt_input();
        else if (argc == 3 && (*argv[1]=='d' || *argv[2]=='d')) { /*./a.out [d] file/k [c]*/
            fprintf(stderr, "usage: %s [c/d] [k] [file]\n", argv[0]);
            exit(EXIT_FAILURE);
        }
        else {/*./a.out file/k [c]*/
            int the_ind=1;
            if (*argv[1]=='c')
                the_ind=2;
            /*-------------------------------------------------------------*/
    if (is_number(argv[the_ind]))
        shift = atoi(argv[the_ind]);
    if (shift ==-1){
            fm = fopen(argv[the_ind],"r");
            if (fm == NULL){
            fprintf(stderr, "no such input file or k is not an integer\n");
            perror("fopen()");
            exit(EXIT_FAILURE);
            }
            fe = fopen("enc.txt", "w");
            if (fe == NULL){
                            fprintf(stderr, "no write file\n");
                            perror("fopen()");
                            exit(EXIT_FAILURE);
                        }
            ask_for_shift();
            getchar();
            file_encryption(fm, fe, shift);
            fclose(fm);
            fclose(fe);
    }
    else
        message_encryption(shift);
}
    }
else if (argc == 3 || (argc==4 && (*argv[1]=='c' || *argv[2]=='c' ||*argv[3]=='c'))) { /*./a.out d/file/k d/file/k
    int ind1=1, ind2=2, the_ind=1;
    if (argc ==3 && (*argv[1]=='d' || *argv[2]=='d')){
        if (*argv[1]=='d')
            the_ind=2;
```

```
        if (is_number(argv[the_ind])){
    shift = atoi(argv[the_ind]);
    decrypt_input_with_shift(shift);
}
else{
    fm = fopen(argv[the_ind],"r");
    fe = fopen("enc.txt", "w");
    decrypt_file(fm, fe);
    fclose(fm);
        fclose(fe);
}
    }
    else{
        if(argc == 4 && (*argv[1]=='d' || *argv[2]=='d' ||*argv[3]=='d')){ /*./a.out [d] d/file/k [c]*/
            fprintf(stderr, "usage: %s [c/d] [k] [file]\n", argv[0]);
            exit(EXIT_FAILURE);
        }
        else if (argc == 4 && *argv[1]=='c')
            ind1=3;
        else if (argc == 4 && *argv[2]=='c')
            ind2=3;
        /*./a.out d/file/k d/file/k [c]*/

    if (is_number(argv[ind1])){
        shift = atoi(argv[ind1]);
    fm = fopen(argv[ind2],"r");
    }
    else if (is_number(argv[ind2])){
    shift = atoi(argv[ind2]);
    fm = fopen(argv[ind1],"r");
    }
    else {
    fprintf(stderr, "no such input file or k is not an integer\n");
        perror("fopen()");
    exit(EXIT_FAILURE);
    }
    fe = fopen("enc.txt", "w");
    if (fm == NULL){
    fprintf(stderr, "no input file\n");
    perror("fopen()");
    exit(EXIT_FAILURE);
    }
    if (fe == NULL){
                fprintf(stderr, "no write file\n");
                perror("fopen()");
                exit(EXIT_FAILURE);
            }
    file_encryption(fm, fe, shift);
        fclose(fm);
    fclose(fe);
    }
}
else{
    int ind1=1, ind2=2;
    if(argc == 4 && (*argv[1]=='c' || *argv[2]=='c' ||*argv[3]=='c')){
        fprintf(stderr, "usage: %s [c/d] [k] [file]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    else if (argc == 4 && *argv[1]=='d')
        ind1=3;
    else if (argc == 4 && *argv[2]=='d')
        ind2=3;
if (is_number(argv[ind1])){
    shift = atoi(argv[ind1]);
    fm = fopen(argv[ind2],"r");
}
else if (is_number(argv[ind2])){
    shift = atoi(argv[ind2]);
    fm = fopen(argv[ind1],"r");
}
else {
    fprintf(stderr, "no such input file or k is not an integer\n");
perror("fopen()");
exit(EXIT_FAILURE);
}
fe = fopen("enc.txt", "w");
if (fm == NULL){
fprintf(stderr, "no input file\n");
perror("fopen()");
exit(EXIT_FAILURE);
```

```
}
if (fe == NULL){
            fprintf(stderr, "no write file\n");
            perror("fopen()");
            exit(EXIT_FAILURE);
        }
        decrypt_file_with_shift(fm, fe, shift);
        fclose(fm);
fclose(fe);
}
return 0;
}
```