

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Бондаренко Кристина Антоновна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Самостоятельная работа	12
4	Выводы	19

Список иллюстраций

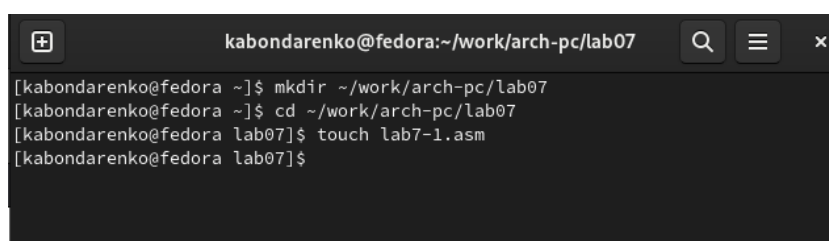
2.1	Создание директории	5
2.2	Создание копии файла для дальнейшей работы	5
2.3	исполняемый файл	6
2.4	редактирование файла	6
2.5	исполняемый файл	7
2.6	Изменяю текст	7
2.7	исполняемый файл	8
2.8	Создание новый файл	8
2.9	редактирование файла	9
2.10	запускаю исполняемый файл,	9
2.11	файл листинга	9
2.12	Редактирование файла	10
2.13	Файл листинга	11
2.14	Файл листинга	11
3.1	Создание файла	12
3.2	Редактирование файла	12
3.3	Запуск исполняемого файла	13
3.4	Редактирование файла	15
3.5	ввод программы в файл	16
3.6	запуск исполняемого файла	16

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга

2 Выполнение лабораторной работы

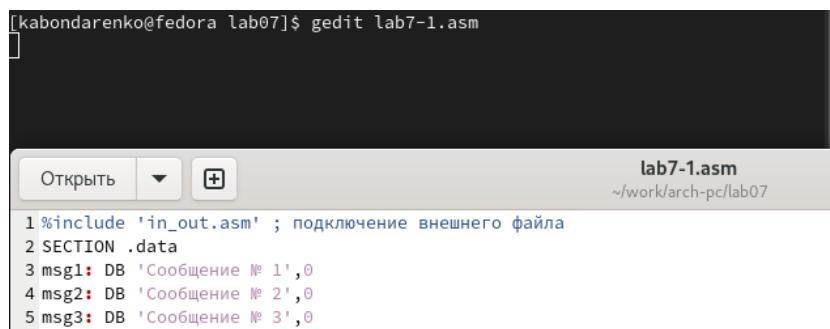
С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [2.1])



```
kabondarenko@fedora:~/work/arch-pc/lab07
[kabondarenko@fedora ~]$ mkdir ~/work/arch-pc/lab07
[kabondarenko@fedora ~]$ cd ~/work/arch-pc/lab07
[kabondarenko@fedora lab07]$ touch lab7-1.asm
[kabondarenko@fedora lab07]$
```

Рис. 2.1: Создание директории

Открываю созданный файл `lab7-1.asm`, вставляю в него программу реализации безусловных переходов (рис. [2.2]).



```
[kabondarenko@fedora lab07]$ gedit lab7-1.asm
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
-----
```

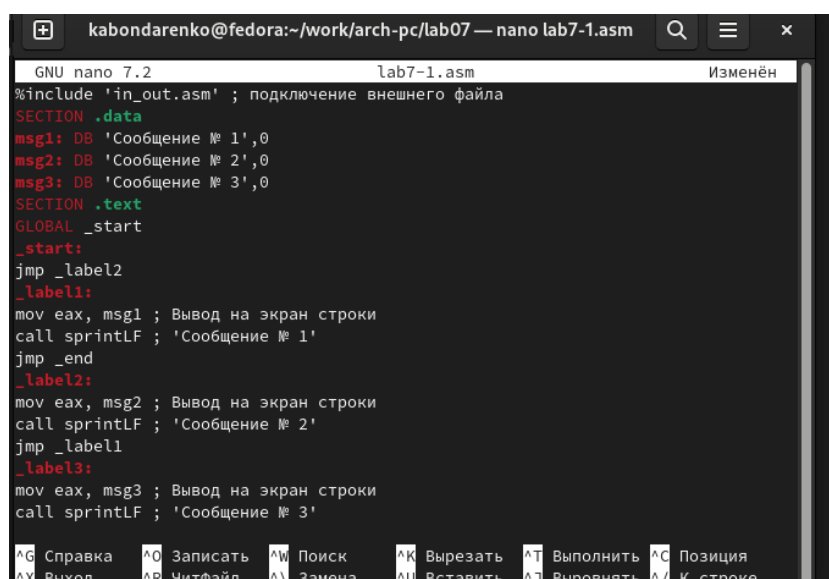
Рис. 2.2: Создание копии файла для дальнейшей работы

Создаю исполняемый файл программы и запускаю его. Инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`. (рис. [2.3]).

```
[kabondarenko@fedora lab07]$ nasm -f elf lab7-1.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[kabondarenko@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[kabondarenko@fedora lab07]$
```

Рис. 2.3: исполняемый файл

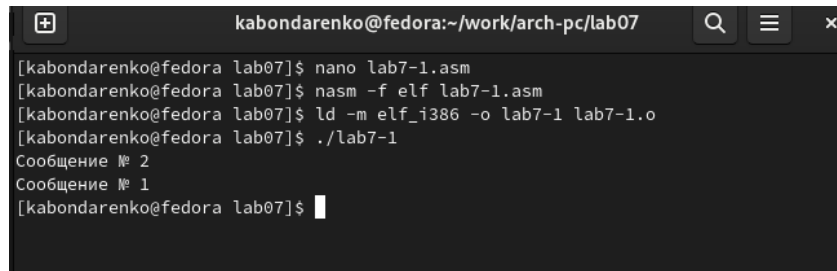
Изменяю текст программы так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. (рис. [2.4]).



```
GNU nano 7.2 lab7-1.asm Изменён
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
```

Рис. 2.4: редактирование файла

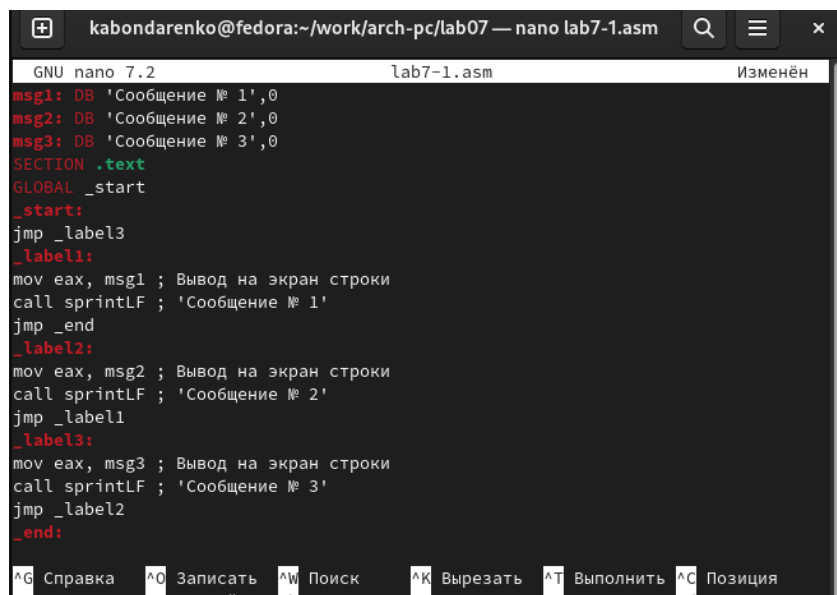
Создаю новый исполняемый файл программы и запускаю его. Убеждаюсь в том, программа работает верно.(рис. [2.5]).



```
kabondarenko@fedora:~/work/arch-pc/lab07
[kabondarenko@fedora lab07]$ nano lab7-1.asm
[kabondarenko@fedora lab07]$ nasm -f elf lab7-1.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[kabondarenko@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[kabondarenko@fedora lab07]$
```

Рис. 2.5: исполняемый файл

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [2.6]).



```
GNU nano 7.2 lab7-1.asm Изменён
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
```

Рис. 2.6: Изменяю текст

Создаю исполняемый файл и проверяю работу программы. Программа отработало верно.(рис. [2.7]).

```
[kabondarenko@fedora lab07]$ nano lab7-1.asm
[kabondarenko@fedora lab07]$ nasm -f elf lab7-1.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[kabondarenko@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[kabondarenko@fedora lab07]$
```

Рис. 2.7: исполняемый файл

Создаю новый файл lab7-2.asm для программы с условным оператором.(рис. [2.8]).

```
[kabondarenko@fedora lab07]$ touch lab7-2.asm
[kabondarenko@fedora lab07]$ nano lab7-2.asm
```

Рис. 2.8: Создание новый файл

Вставляю программу, которая определяет и выводит на экран наибольшее число (рис. [2.9]).


```
GNU nano 7.2 lab7-2.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B'
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
```

Рис. 2.9: редактирование файла

Создаю и запускаю новый исполняемый файл, проверяю работу программы для разных B при A=20 и C=50 (рис.[2.10]).

```
[kabondarenko@fedora lab07]$ nasm -f elf lab7-2.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[kabondarenko@fedora lab07]$ ./lab7-2
Введите B: 43
Наибольшее число: 50
[kabondarenko@fedora lab07]$ ./lab7-2
Введите B: 67
Наибольшее число: 67
[kabondarenko@fedora lab07]$
```

Рис. 2.10: запуская исполняемый файл,

Создаю файл листинга для программы в файле lab7-2.asm(рис. [2.11]).

```
[kabondarenko@fedora lab07]$ nasm -f elf -llab7-2.lst lab7-2.asm
[kabondarenko@fedora lab07]$ mcedit lab7-2.asm
[kabondarenko@fedora lab07]$ mcedit lab7-2.lst
```

Рис. 2.11: файл листинга

Открываю файл листинга с помощью редактора mcedit. Рассмотрим 9-11 стро-

ки:(рис. [2.12]).

```
1      <1> ;----- slen -----
2      <1> ; Функция вычисления длины сообщения
3      <1> slen:
4 00000000 53      <1> push    ebx
5 00000001 89C3     <1> mov     ebx, eax
6
7      <1> nextchar:
8 00000003 803800   <1> cmp     byte [eax], 0
9 00000006 7403     <1> jz      finished
10 00000008 40      <1> inc     eax
11 00000009 EBF8     <1> jmp     nextchar
12
13      <1> finished:
14 0000000B 2908     <1> sub     eax, ebx
15 0000000D 5B      <1> pop     ebx
16 0000000E C3      <1> ret
17
18
19      <1> ;----- sprint -----
20      <1> ; Функция печати сообщения
21      <1> ; входные данные: mov eax,<message>
22      <1> sprint:
23 0000000F 52      <1> push    edx
24 00000010 51      <1> push    ecx
25 00000011 53      <1> push    ebx
26 00000012 50      <1> push    eax
```

Рис. 2.12: Редактирование файла

9 строка:

- Первые цифры [9] - это номер строки файла листинга.
- Следующие цифры [00000006] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [7403] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латинского алфавита.
- следующее [jz finished] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями.

10 строка:

- Первые цифры [10] - это номер строки файла листинга.
- Следующие цифры [00000008] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [40] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтоу и появляются буквы латинского алфавита.

- следующее [inc eax] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

11 строка:

- Первые цифры [11] - это номер строки файла листинга.
- Следующие цифры [00000009] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [EBF8] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [jmp nextchar] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

Открываю файл lab7-2.asm с помощью редактора и Удаляю один операнд в инструкции cmp. (рис. [2.13]).

```

; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B'
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число

```

Рис. 2.13: Файл листинга

Открываю файл листинга с помощью редактора mscedit и замечаю, что в файле листинга появляется ошибка. (рис. [2.14]).

```

28      ; ----- Сравниваем 'A' и 'C' (как символы)
28      cmp ecx ; Сравниваем 'A' и 'C'
29 0000011C 7F0C      error: invalid combination of opcode and operands
30 0000011E 8B00[39000000] jg check_B ; если 'A>C', то переход на метку 'check_B'
31 00000124 8900[00000000] mov ecx,[C] ; иначе 'ecx = C'
32      mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число

```

Рис. 2.14: Файл листинга

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга

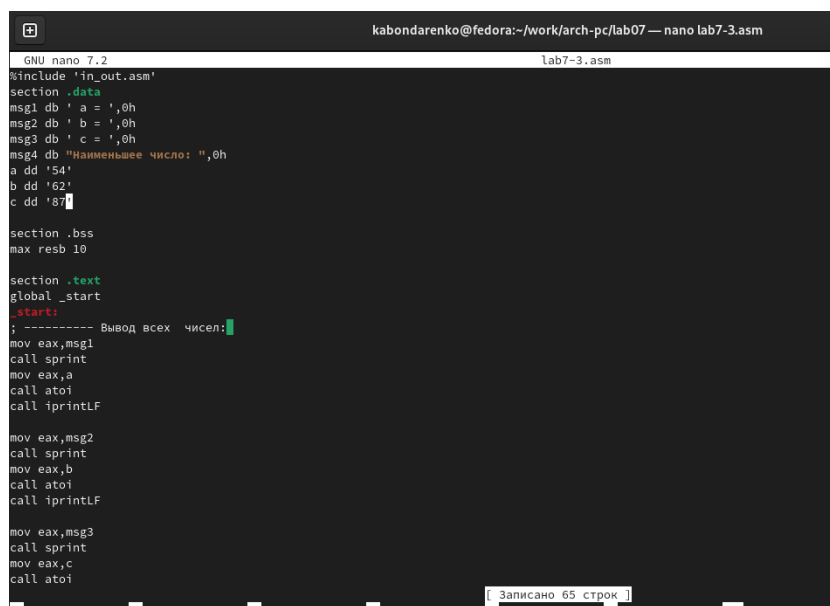
3 Самостоятельная работа

Создаю файл lab7-3.asm с помощью утилиты touch (рис. [3.1]).

```
[kabondarenko@fedora lab07]$ touch lab7-3.asm
[kabondarenko@fedora lab07]$ vim lab7-3.asm
```

Рис. 3.1: Создание файла

Ввожу в созданный файл текст программы для вычисления наименьшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 5 вариант (рис. [3.2]).



```
GNU nano 7.2 lab7-3.asm
#include "in_out.asm"
section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '54'
b dd '62'
c dd '87'

section .bss
max resb 10

section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF

mov eax,msg3
call sprint
mov eax,c
call atoi
```

Рис. 3.2: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [3.3]).

```

[kabondarenko@fedora lab07]$ nasm -f elf lab7-3.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[kabondarenko@fedora lab07]$ ./lab7-3
a = 54
b = 62
c = 87
Наименьшее число: 54
[kabondarenko@fedora lab07]$

```

Рис. 3.3: Запуск исполняемого файла

Текст программы

```

#include 'in_out.asm'

section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '54'
b dd '62'
c dd '87'

section .bss
max resb 10

section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF

```

```

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF

mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintLF

;-----сравнивание чисел
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov ecx,[a] ;
mov [max],ecx ;
;-----сравнивание чисел a c
cmp ecx,[c]; if a>c
jl check_b ; то перход на метку
mov ecx,[c] ;
mov [max],ecx ;
;-----метка check_b
check_b:
mov eax,max ;
call atoi
mov [max],eax ;

```

```

;-----
mov ecx,[max] ;
cmp ecx,[b] ;
jle check_c ;
mov ecx,[b] ;
mov [max],ecx ;
;-----

check_c:
mov eax,msg4 ;
call sprint ;
mov eax,[max];
call iprintLF ;
call quit

```

Создаю новый файл lab7-4 и ввожу в него программу, которая выводит значения функции. Функцию беру из таблицы в соответствии со своим 5 вариантом (Вариант рис. [3.5]).

$$5 \quad \begin{cases} 2(x-a), & x > a \\ 15, & x \leq a \end{cases} \quad (1;2) \quad (2;1)$$

Рис. 3.4: Редактирование файла

```

mov eax,msg2
call sprintf
mov ecx,a
mov edx,10
call sread
mov eax,a ;
call atoi
mov [a],eax ;
;-----
mov ecx,[a]
cmp ecx,[x] ;x>a
jl check_a ;
mov ebx,15
mov ecx,ebx
jmp _end
check_a:
mov ecx,[x]
sub ecx,[a]
mov eax,ecx
mov ebx,2
mul ebx
mov ecx,eax
;-----
_end:
mov eax,msg3 ;
call sprintf ;
mov eax,ecx ;
call iprintLF;

```

Рис. 3.5: ввод программы в файл

Создаю исполняемый файл и проверяю её выполнение при (x=1, a=2) при (x=2 и a=1) (рис. [3.6]). Программа отработала верно!

```

[kabondarenko@fedora lab07]$ nasm -f elf lab7-4.asm
[kabondarenko@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[kabondarenko@fedora lab07]$ ./lab7-4
Введите x: 1
Введите a: 2
f(x) = 15
[kabondarenko@fedora lab07]$ ./lab7-4
Введите x: 2
Введите a: 1
f(x) = 2
[kabondarenko@fedora lab07]$

```

Рис. 3.6: запуск исполняемого файла

Текст программы

```

#include 'in_out.asm'

section .data

msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h
msg3 db 'f(x) = ',0h

section .bss

```



```

x resb 10
a resb 10

section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
;-----
call atoi
mov [x],eax
;-----

mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a ;
call atoi
mov [a],eax ;
;-----
mov ecx,[a]
cmp ecx,[x] ;x>a
jl check_a ;

```

```

mov ebx,15
mov ecx,ebx
jmp _end
check_a:
mov ecx,[x]
sub ecx,[a]
mov eax,ecx
mov ebx,2
mul ebx
mov ecx,eax
;-----
_end:
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintLF;
call quit ;

```

4 Выводы

При выполнении данной лабораторной работы я освоила инструкции условного и безусловного вывода и ознакомилась с структурой файла листинга.