

# **Лабораторная работа №11**

**Программирование в командном процессоре ОС UNIX. Ветвления и  
циклы**

Хамдамова Айжана

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11
4	Контрольные вопросы	12
	Список литературы	16

# Список иллюстраций

fignoКод в файле . . . . .	6
fignoВызов команды . . . . .	6
fignoРезультат . . . . .	7
fignoФайл с++ . . . . .	7
fignoКод файла . . . . .	7
fignoРезультат . . . . .	8
fignoКод . . . . .	8
fignoРезультат . . . . .	9
fignoКод . . . . .	9
fignoВызов в терминале . . . . .	9
fignoРезультат 4 . . . . .	10

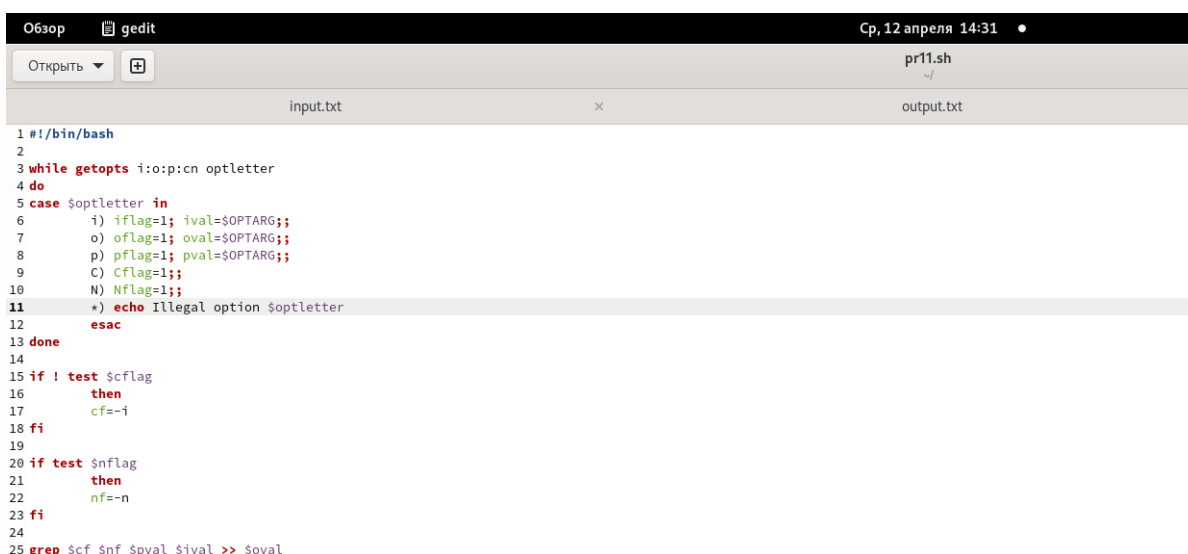
## **Список таблиц**

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

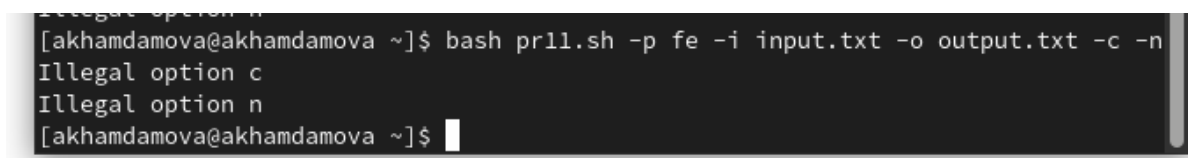
## 2 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-ршаблон` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.



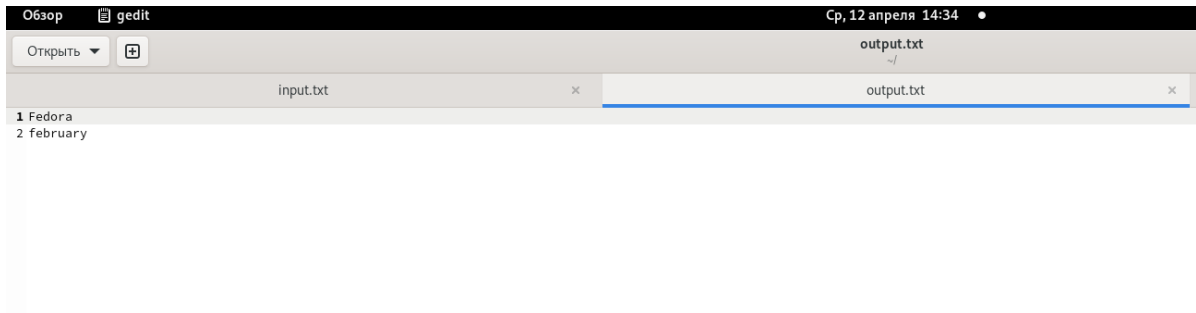
```
1 #!/bin/bash
2
3 while getopts i:o:p:cn optletter
4 do
5 case $optletter in
6   i) iflag=1; ival=$OPTARG;;
7   o) oflag=1; oval=$OPTARG;;
8   p) pflag=1; pval=$OPTARG;;
9   C) Cflag=1;;
10  N) Nflag=1;;
11  *) echo Illegal option $optletter
12     esac
13 done
14
15 if ! test $cflag
16 then
17   cf=-i
18 fi
19
20 if test $nflag
21 then
22   nf=-n
23 fi
24
25 grep $cf $nf $pval $ival >> $oval
```

Код в файле



```
Illegal option n
[akhamdamova@akhamdamova ~]$ bash pr11.sh -p fe -i input.txt -o output.txt -c -n
Illegal option c
Illegal option n
[akhamdamova@akhamdamova ~]$
```

Вызов команды

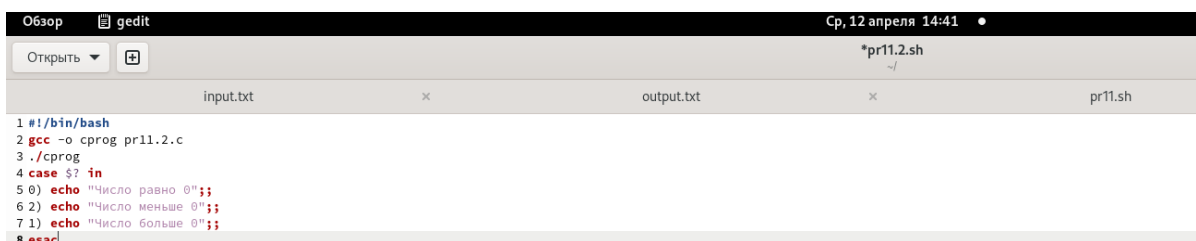


## Результат

2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.



## файл с++



## код файла

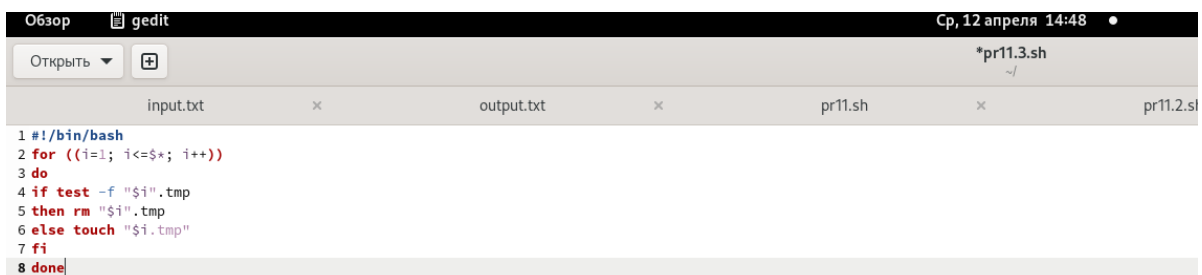
```

[akhamdamova@akhamdamova ~]$ gedit pr11.2.sh
[akhamdamova@akhamdamova ~]$ gedit pr11.2.c
[akhamdamova@akhamdamova ~]$ bash pr11.2.sh
Введите число : 7
Число больше 0
[akhamdamova@akhamdamova ~]$ bash pr11.2.sh
Введите число : 0
Число равно 0
[akhamdamova@akhamdamova ~]$

```

### Результат

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до  $\infty$  (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).



The screenshot shows a gedit editor window with the following content:

```

1 #!/bin/bash
2 for ((i=1; i<=$*; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i".tmp"
7 fi
8 done

```

The window title is "Обзор gedit" and the status bar shows "Ср, 12 апреля 14:48". The file name is "\*pr11.3.sh".

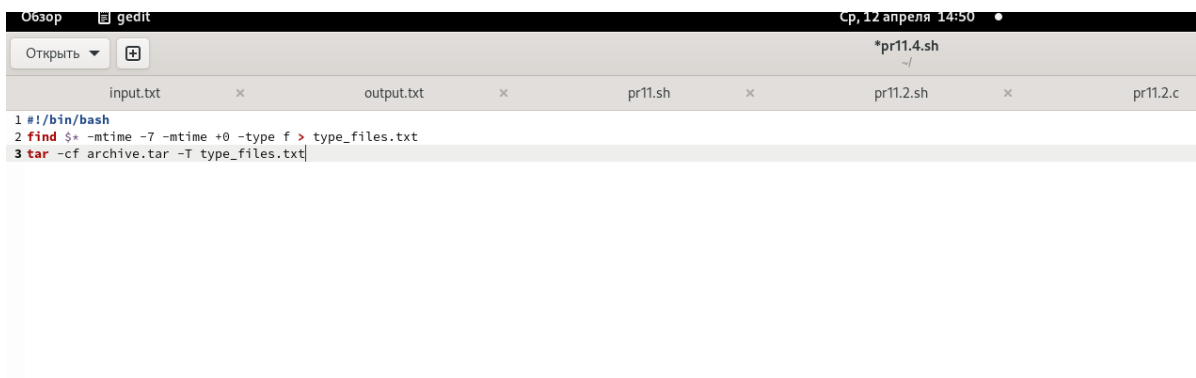
### Код



```
menlo paano 0
[akhamdamova@akhamdamova ~]$ gedit pr11.3.sh
[akhamdamova@akhamdamova ~]$ bash pr11.3.sh 2
[akhamdamova@akhamdamova ~]$ ls
1.tmp      feathers  input.txt  play       text.txt   Изображения
2.tmp      file1.sh  lab07.sh  pr11.2.c  tutorial   Музыка
backup     file2.sh  lab07.sh~ pr11.2.sh  work       Общедоступные
bin        file3.sh  lab09.txt pr11.3.sh  Видео      'Рабочий стол'
conf.txt   file4.sh  output.txt pr11.sh    Документы  Шаблоны
cprog     file.txt  p11.4.sh  ski.places Загрузки
[akhamdamova@akhamdamova ~]$
[akhamdamova@akhamdamova ~]$
```

## Результат

4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find)



```
Обзор  gedit  Ср, 12 апреля 14:50
Открыть  +  *pr11.4.sh
~/
input.txt  x  output.txt  x  pr11.sh  x  pr11.2.sh  x  pr11.2.c
1 #!/bin/bash
2 find -x -mtime -7 -mtime +0 -type f > type_files.txt
3 tar -cf archive.tar -T type_files.txt
```

## Код

```
[akhamdamova@akhamdamova ~]$ gedit pr11.4.sh
[akhamdamova@akhamdamova ~]$ bash pr11.4.sh /home/akhamdamova/work
tar: Удаляется начальный '/' из имен объектов
tar: Удаляются начальные '/' из целей жестких ссылок
[akhamdamova@akhamdamova ~]$
```

## Вызов в терминале

type_files.txt														
input.txt	output.txt	pr11.sh	pr11.2.sh	pr11.2.c	pr11.3.sh	pr11.4.sh	tupe_files.txt							
1 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/refs/heads/master													
2 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/refs/remotes/origin/master													
3 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/d8/3f97a201dfbf29d1cc20b357a7befd64780ff													
4 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/39/dbda1687541d403b04183af63deb195d7b2056													
5 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/86/fbee9343056b8ac1dccb2bffff8ff365178014													
6 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/d3/9bc189ec3e3ba56dd9ca50e694db159655c95d													
7 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/5b/cb288cddbdc516741298a70cbdebf6f1e884a													
8 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/35/ff4d4698d571695c4535633f1e714383c806e22													
9 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/3e/5701c2b84475d876a4e838d38b1231cd923e8d													
10 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/08/872202c9991b261edac79f66847a8c5161c511													
11 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/22/f866cb81ccf0d42a88f91bc0b11389b05fba7													
12 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/fd/a5d88674b7b38cc8b5673be8fa411cc84ba3													
13 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/c3/f3417bd7d7bce522705ae7843a7989f7d5c4a1													
14 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/96/e402b3324ae3f251893199c5042c2c31ffbad5													
15 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/64/115a9cd2302fabb42a2e986939c01258f5ee8f													
16 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/a8/a4fccab706af6ba77b7cd1805aebc1bc7c98e7f													
17 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/3a/eb81b37914e41ca9360ed22f5959fe44053af													
18 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/3a/433f8977c78fb36364c6e62918a45cc4dea75													
19 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/6f/d9374c30094485e3cbd94e0bbe248d200b5f2													
20 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/ae/997b034404d28d1933c263dca848412d3c641f													
21 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/91/d6d1302b573a88277eb4a3330d794dd722bb													
22 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/95/0861ae10ef82f39277e27a28a79db710852a9b													
23 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/b2/c48d44fd6e72fff067f2a6ee9f8af9e1841bd													
24 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/b3/02d18ad7585e56d4e435d0b74ee458148180b													
25 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/objects/b0/83a193ffa492e7dd20bdf903eb05e2fee3259ce													
26 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/logs/refs/remotes/origin/master													
27 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/logs/refs/heads/master													
28 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/logs/HEAD													
29 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/COMMIT_EDITMSG													
30 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/.git/index													
31 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/1.png													
32 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/2.png													
33 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/3.png													
34 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/4.png													
35 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/5.png													
36 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/6.png													
37 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/7.png													
38 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/8.png													
39 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image/9.png													
40 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/pandoc/filters/pandocxnos/__pycache__/_init__.cpython-311.pyc													
41 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/pandoc/filters/pandocxnos/__pycache__/_core.cpython-311.pyc													
42 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/pandoc/filters/pandocxnos/__pycache__/_pandocattributes.cpython-311.pyc													
43 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/pandoc/filters/pandocxnos/__pycache__/_main.cpython-311.pyc													
44 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/report.md													
45 /home/akhandamova/work/study/2022-2023/Операционные	системы/os-intro/labs/lab10/report/image_lab10.zip													

Текст ▾ Ширина таблицы: 8 ▾

## Результат 4

## **3 Выводы**

Я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 4 Контрольные вопросы

### 1. Каково предназначение команды `getopts`?

Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: 

```
while
getopts o:i:Ltr optletter do
case optletter in
o) oflag = 1; oval =OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option
$optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следова-

тельно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

## 2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы: – соответствует произвольной, в том числе и пустой строке; ? – соответствует любому одинарному символу; [с1-с2] – соответствует любому символу, лексикографически находящемуся между символами с1 и с2. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .с` – *выведет все файлы с последними двумя символами, совпадающими с .с*. `echo prog.?` – *выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog..* `[a-z]` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

## 3. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

#### 4. Какие операторы используются для прерывания цикла?

Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

#### 5. Для чего нужны команды `false` и `true`?

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т.е. ложь).

#### 6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

#### 7. Объясните различия между конструкциями `while` и `until`.

Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда

из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

## **Список литературы**