
Front matter

title: "Отчёт по лабораторной работе №11" subtitle: " Программирование в командном процессоре ОС UNIX. Ветвления и циклы " author: "Федорина Эрнест Васильевич НКНбд-01-21"

Generic otions

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables
fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt ## I18n polyglossia
polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true
polyglossia-otherlangs: name: english ## I18n babel babel-lang: russian babel-otherlangs:
english ## Fonts mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT
Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions:
Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase,Scale=0.9
Biblatex biblatex: true biblio-style: "gost-numeric" biblatexoptions: - parenttracker=true
- backend=biber - hyperref=auto - language=auto - autolang=other* - citestyle=gost-
numeric ## Pandoc-crossref LaTeX customization figureTitle: "Рис." tableTitle: "Таблица"
listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle: "Список таблиц" lolTitle:
"Листинги" ## Misc options indent: true header-includes: -

keep figures where there are in the text

– # keep figures where there are in the text

Цель работы

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

Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — вывести данные в указанный файл; `-r`шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.
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.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (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).
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`).

Теоретическое введение

При интерактивной работе с операционной системой пользователь постоянно сталкивается с необходимостью отдавать ей команды. В UNIX эту работу выполняет программа, которая называется командным процессором (shell). Иногда командный процессор называют шеллом или оболочкой, или интерпретатором команд (последнее неточно, потому что круг задач командного процессора шире, чем интерпретация команд).

Shell действует как посредник между вами и ядром операционной системы. Ядро — это часть операционной системы, которая всегда находится в памяти компьютера, это программа. Командный процессор преобразует ваши команды в инструкции для операционной системы, а операционная система превращает их в инструкции для аппаратных средств компьютера. По сути, именно оболочка придает определенную “персонализацию” системам UNIX.

Командный процессор выполняет в системе следующие задачи:

- интерпретация команд пользователя, в том числе разбор командной строки;
- запуск программ;
- организация перенаправлений потоков между процессами;
- интерпретация языка скриптов и их выполнение;
- управление заданиями;
- интерпретация шаблонов имен файлов;
- подстановка имен файлов в командную строку.

Кроме того, shell является мощным языком программирования.

Любая команда, являющаяся отдельной программой, т.е. не встроенной в интерпретатор, будет выполняться одинаково, независимо от shell'a. Например, если вы хотите что-то напечатать, команда печати всегда работает одинаково.

Некоторые команды встроены в shell, т.е. они являются частью программы оболочки и, как следствие, могут выполняться по-разному в зависимости от оболочки, в которой они запускаются. Есть три вида команд, встроенных в shell:

- общие команды запускаются несколько быстрее, так как они являются частью оболочки;
- команды адаптации позволяют адаптировать оболочку;
- команды программирования образуют язык программирования оболочки.

При смене shell'a вы не заметите никакой разницы между общими командами, которые встроены просто для повышения быстродействия. Однако команды адаптации и программирования изменяются.

Примером общих команд, встроенных в оболочку, служат команды `cd`, `echo`, `pwd`, `login`, `umask`.

Адаптация включает в себя создание новых команд или новых имен для старых команд и привлечение новых возможностей. Примером общепринятой адаптации служит изменение стимула (или приглашения системы).

Можно персонализировать свою работу, привлекая удобные возможности, встроенные в различные оболочки. Почти все такие возможности связаны с этапом преобразования команды. Большинство различий между оболочками кроется в том, как они адаптированы.

Примером команд адаптации могут служить следующие команды:

- `alias` – создает новое имя для команды;
- `export` – создает переменную среды;
- `set` – включает и выключает некоторые опции shell'a;
- `unalias` – удаляет псевдоним команды.

Выполнение лабораторной работы

Используя команды `getopts` `grep`, написал командный файл, который анализирует командную строку с ключами(рис.1,2)

```
lab111.sh [----] 0 L: [ 1+31 32/ 36] *(700 / 772b) 0032 0x020 [*] [X]
#!/bin/bash
while getopts i:o:p:Cn optletter
do case $optletter in
<----->i) iflag=1; ival=$OPTARG;;
<----->o) oflag=1; oval=$OPTARG;;
<----->p) pflag=1; pval=$OPTARG;;
<----->C) Cflag=1;;
<----->n) nflag=1;;
<----->*) echo Illegaloption $optletter
esac
done
if (((cflag==1)&&(nflag==1)))
then grep -e${pval} -i -n ${ival}
if ((oflag==1))
then grep -e${pval} -i -n ${ival} > ${oval}
fi
fi
if (((cflag==1)&&(nflag==0)))
then grep -e${pval} -i ${ival}
if ((oflag==1))
then grep -e${pval} -i ${ival} > ${oval}
fi
fi
if (((cflag==0)&&(nflag==1)))
then grep -e${pval} -n ${ival}
if ((oflag==1))
then grep -e${pval} -n ${ival} > ${oval}
fi
fi
if (((cflag==0)&&(nflag==0)))
then grep -e${pval} ${ival}
if ((oflag==1))
then grep -e${pval} ${ival} > ${oval}
fi
fi
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перезагрузить 7Поиск 8Удалить 9Меню 10Выход
```

{рис.1}

```
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p One
One
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p One -C
lab111.sh: недопустимый параметр - C
Illegaloption ?
One
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p One -c
Illegaloption c
One
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p g
GJGgjfkldgjlkd
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p g -n
5:GJGgjfkldgjlkd
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p Three -n
[evfedorina@evfedorina ~]$ bash lab111.sh -i lab11.txt -o lab11.txt -p three -n
3:three
[evfedorina@evfedorina ~]$
```

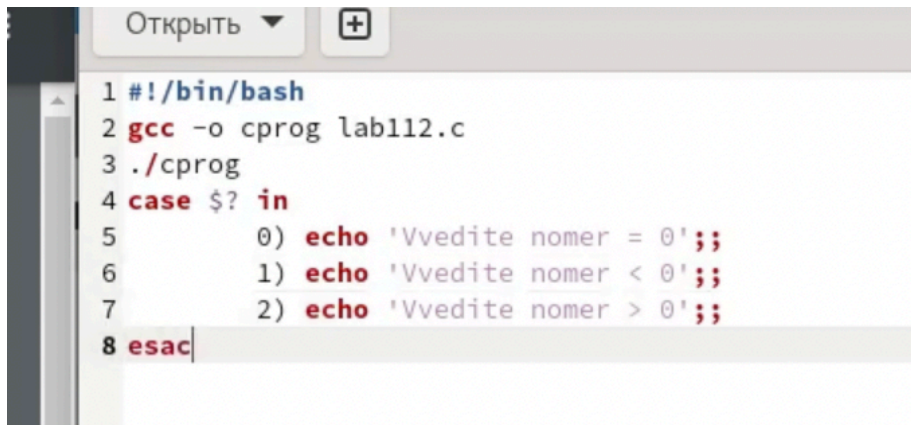
{рис.2}

Написал на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.(рис.3,4,5)

A screenshot of a code editor window titled 'lab112.c'. The editor shows a C program that reads an integer from the user and checks if it is zero, negative, or positive. The code is as follows:

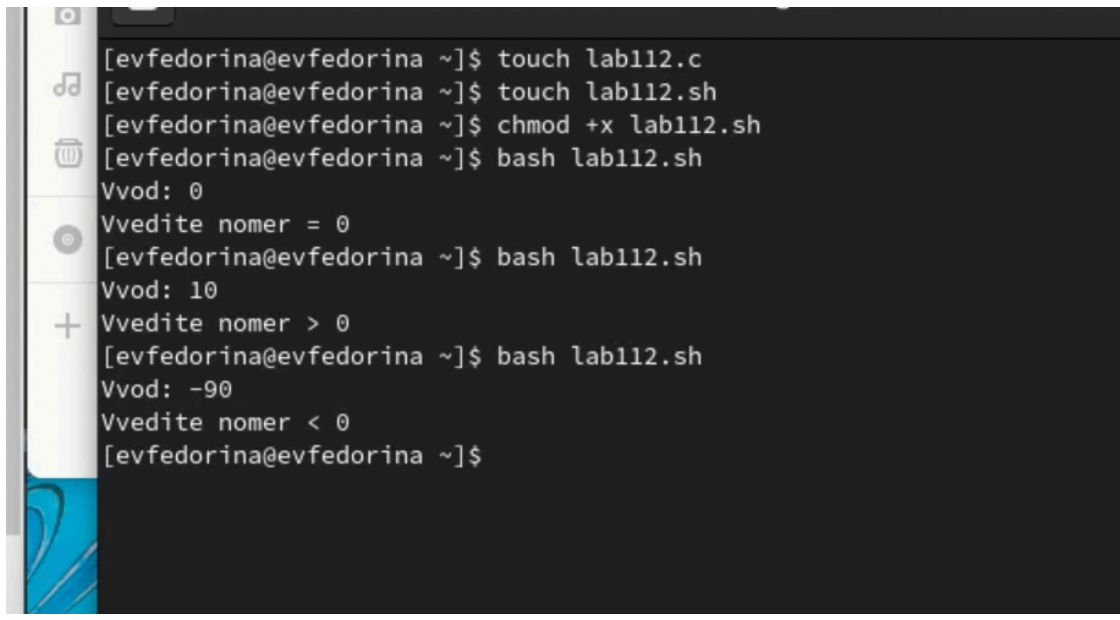
```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int a;
6     printf("Vvod: ");
7     scanf("%i", &a);
8     if (a==0)
9         exit(0);
10    else if (a<0)
11        exit(1);
12    else if (a>0)
13        exit(2);
14    return(3);
15 }
```

{рис.3}

A screenshot of a code editor window showing a shell script. The script uses 'gcc' to compile the C program from the previous image and then uses a 'case' statement to echo the exit status of the program. The code is as follows:

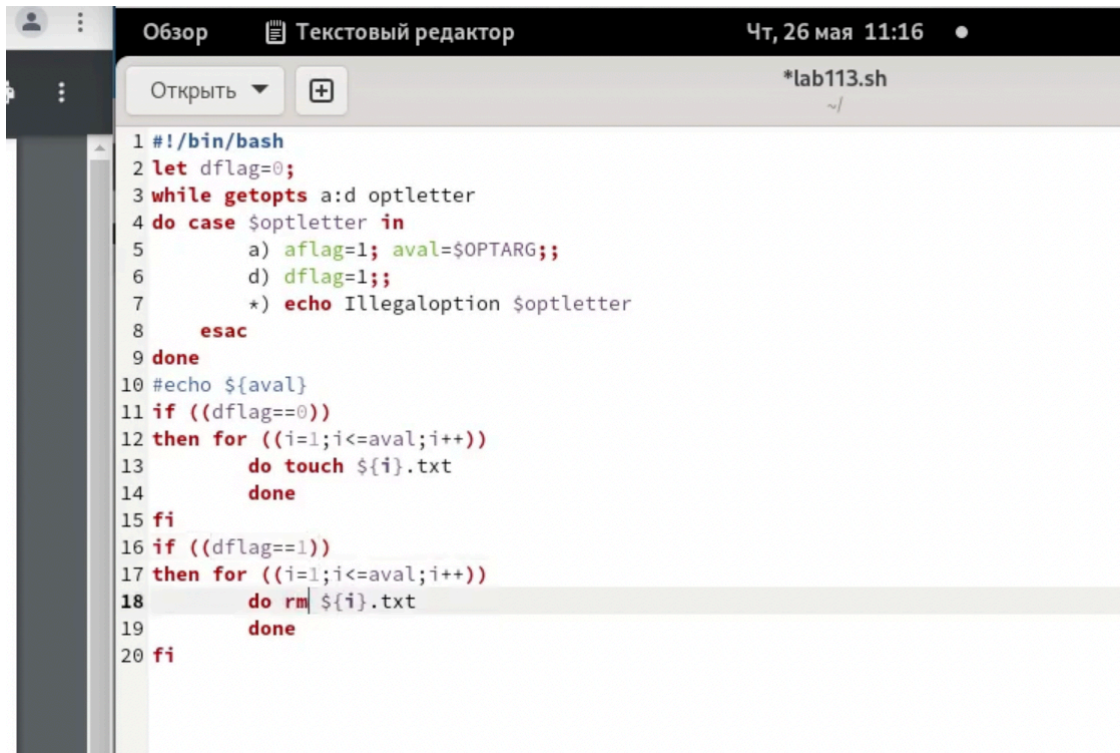
```
1 #!/bin/bash
2 gcc -o cprog lab112.c
3 ./cprog
4 case $? in
5     0) echo 'Vvedite nomer = 0';;
6     1) echo 'Vvedite nomer < 0';;
7     2) echo 'Vvedite nomer > 0';;
8 esac
```

{рис.4}

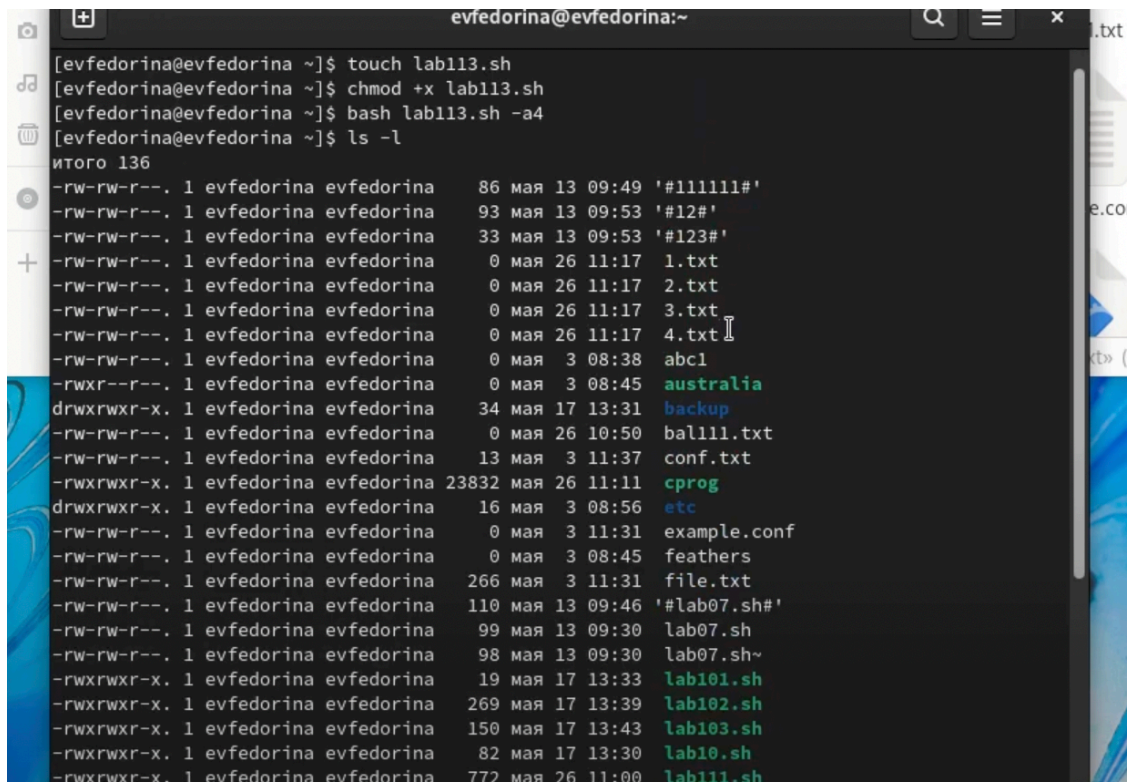


```
[evfedorina@evfedorina ~]$ touch lab112.c
[evfedorina@evfedorina ~]$ touch lab112.sh
[evfedorina@evfedorina ~]$ chmod +x lab112.sh
[evfedorina@evfedorina ~]$ bash lab112.sh
Vvod: 0
Vvedite nomer = 0
[evfedorina@evfedorina ~]$ bash lab112.sh
Vvod: 10
Vvedite nomer > 0
[evfedorina@evfedorina ~]$ bash lab112.sh
Vvod: -90
Vvedite nomer < 0
[evfedorina@evfedorina ~]$
```

Написал командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (рис.6,7)



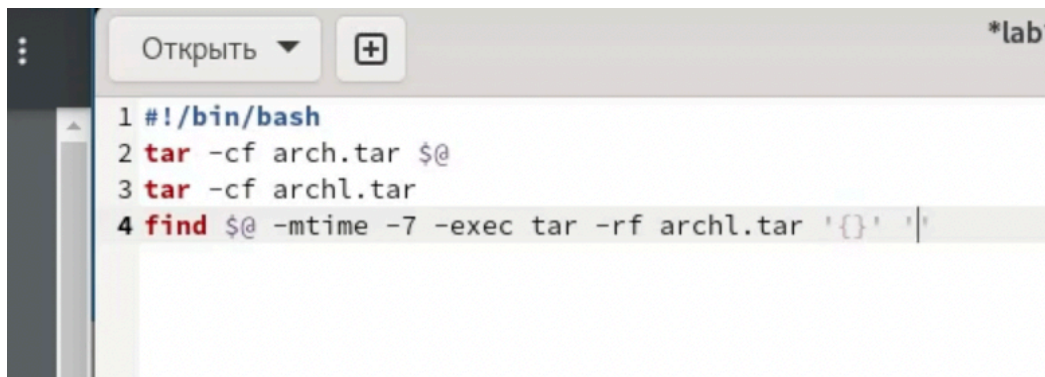
```
1 #!/bin/bash
2 let dflag=0;
3 while getopt a:d optletter
4 do case $optletter in
5     a) aflag=1; aval=$OPTARG;;
6     d) dflag=1;;
7     *) echo Illegaloption $optletter
8     esac
9 done
10 #echo ${aval}
11 if ((dflag==0))
12 then for ((i=1;i<=aval;i++))
13     do touch ${i}.txt
14     done
15 fi
16 if ((dflag==1))
17 then for ((i=1;i<=aval;i++))
18     do rm ${i}.txt
19     done
20 fi
```


A terminal window titled 'evfedorina@evfedorina:~' showing a series of commands and their outputs. The commands executed are: 'touch lab113.sh', 'chmod +x lab113.sh', 'bash lab113.sh -a4', and 'ls -l'. The output of 'ls -l' is a long list of files and directories with their permissions, owners, sizes, and timestamps. The files include '1.txt', '2.txt', '3.txt', '4.txt', 'abc1', 'australia', 'backup', 'ball111.txt', 'conf.txt', 'cprog', 'etc', 'example.conf', 'feathers', 'file.txt', 'lab07.sh', 'lab07.sh~', 'lab101.sh', 'lab102.sh', 'lab103.sh', 'lab10.sh', and 'lab111.sh'.

```
evfedorina@evfedorina:~$ touch lab113.sh
evfedorina@evfedorina:~$ chmod +x lab113.sh
evfedorina@evfedorina:~$ bash lab113.sh -a4
evfedorina@evfedorina:~$ ls -l
итого 136
-rw-rw-r--. 1 evfedorina evfedorina 86 мая 13 09:49 '#111111#'
-rw-rw-r--. 1 evfedorina evfedorina 93 мая 13 09:53 '#12#'
-rw-rw-r--. 1 evfedorina evfedorina 33 мая 13 09:53 '#123#'
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 26 11:17 1.txt
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 26 11:17 2.txt
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 26 11:17 3.txt
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 26 11:17 4.txt
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 3 08:38 abc1
-rwxr--r--. 1 evfedorina evfedorina 0 мая 3 08:45 australia
drwxrwxr-x. 1 evfedorina evfedorina 34 мая 17 13:31 backup
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 26 10:50 ball111.txt
-rw-rw-r--. 1 evfedorina evfedorina 13 мая 3 11:37 conf.txt
-rwxrwxr-x. 1 evfedorina evfedorina 23832 мая 26 11:11 cprog
drwxrwxr-x. 1 evfedorina evfedorina 16 мая 3 08:56 etc
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 3 11:31 example.conf
-rw-rw-r--. 1 evfedorina evfedorina 0 мая 3 08:45 feathers
-rw-rw-r--. 1 evfedorina evfedorina 266 мая 3 11:31 file.txt
-rw-rw-r--. 1 evfedorina evfedorina 110 мая 13 09:46 '#lab07.sh#'
-rw-rw-r--. 1 evfedorina evfedorina 99 мая 13 09:30 lab07.sh
-rw-rw-r--. 1 evfedorina evfedorina 98 мая 13 09:30 lab07.sh~
-rwxrwxr-x. 1 evfedorina evfedorina 19 мая 17 13:33 lab101.sh
-rwxrwxr-x. 1 evfedorina evfedorina 269 мая 17 13:39 lab102.sh
-rwxrwxr-x. 1 evfedorina evfedorina 150 мая 17 13:43 lab103.sh
-rwxrwxr-x. 1 evfedorina evfedorina 82 мая 17 13:30 lab10.sh
-rwxrwxr-x. 1 evfedorina evfedorina 772 мая 26 11:00 lab111.sh
```

{рис.7}

Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории.

A terminal window showing a script for creating tar archives. The script consists of four lines: a shebang line, and three lines using 'tar' and 'find' to create archives. The window title is '*lab1'.

```
*lab1
1 #!/bin/bash
2 tar -cf arch.tar $@
3 tar -cf archl.tar
4 find $@ -mtime -7 -exec tar -rf archl.tar '{}' \;
```

{рис.8}



{рис.9}

Выводы

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