

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Богату Ирина Владимировна

Содержание

Цель работы	5
Выполнение лабораторной работы	6
Выполнение задания для самостоятельной работы	16
Выводы	19

Список иллюстраций

1	Создание рабочей директории и файла lab8-1.asm	6
2	Запуск Midnight commander	6
3	Вставка кода из файла листинга 8.1	7
4	Копирование файла in_out.asm в рабочую директорию	8
5	Сборка программы из файла lab8-1.asm и её запуск	8
6	Изменение файла lab8-1.asm	9
7	Повторная сборка программы из файла lab8-1.asm и её запуск . .	9
8	Результат вывода	10
9	Результат вывода для чётного N	10
10	Редактирование файла lab8-1.asm	11
11	Повторная сборка программы из файла lab8-1.asm и её запуск . .	11
12	Создание второго файла: lab8-2.asm	12
13	Запись кода из листинга 8.2 в файл lab8-2.asm	12
14	Сборка программы из файла lab8-2.asm и её запуск	13
15	Создание третьего файла: lab8-3.asm	13
16	Запись кода из листинга 8.3 в файл lab8-3.asm	14
17	Сборка программы из файла lab8-2.asm и её запуск	14
18	Изменение файла lab8-3.asm	15
19	Повторная сборка программы из файла lab8-3.asm и её запуск . .	15
1	Создание файла самостоятельной работы	16
2	Код файла самостоятельной работы	17
3	Код файла самостоятельной работы (продолжение)	18
4	Сборка и запуск программы первого задания самостоятельной ра- боты, а также результат выполнения	18

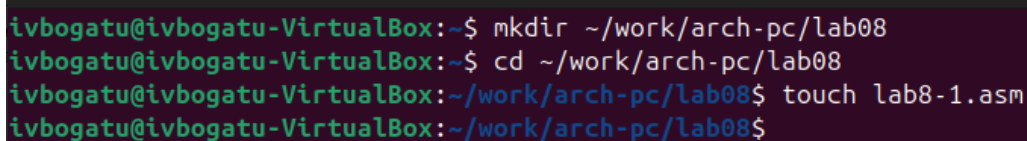
Список таблиц

Цель работы

Научиться работать с циклами на языке Ассемблера, а также научиться обрабатывать аргументы командной строки

Выполнение лабораторной работы

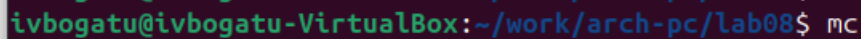
Для начала выполнения лабораторной работы создадим рабочую директорию и файл lab8-1.asm (рис. 2.1):



```
ivbogatu@ivbogatu-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
ivbogatu@ivbogatu-VirtualBox:~$ cd ~/work/arch-pc/lab08
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ touch lab8-1.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 1: Создание рабочей директории и файла lab8-1.asm

Далее, запустим Midnight commander (рис. 2.2):



```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ mc
```

Рис. 2: Запуск Midnight commander

Теперь, вставим в ранее созданный файл из листинга 8.1. Он должен запускать цикл и выводить каждую итерацию число, на единицу меньше предыдущего (начинается выводить с числа N) (рис. 2.3):

```

#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла

```

Рис. 3: Вставка кода из файла листинга 8.1

Чтобы собрать код, нужен файл in_out.asm. скопируем его из директории прошлой лабораторной работы (рис. 2.4):

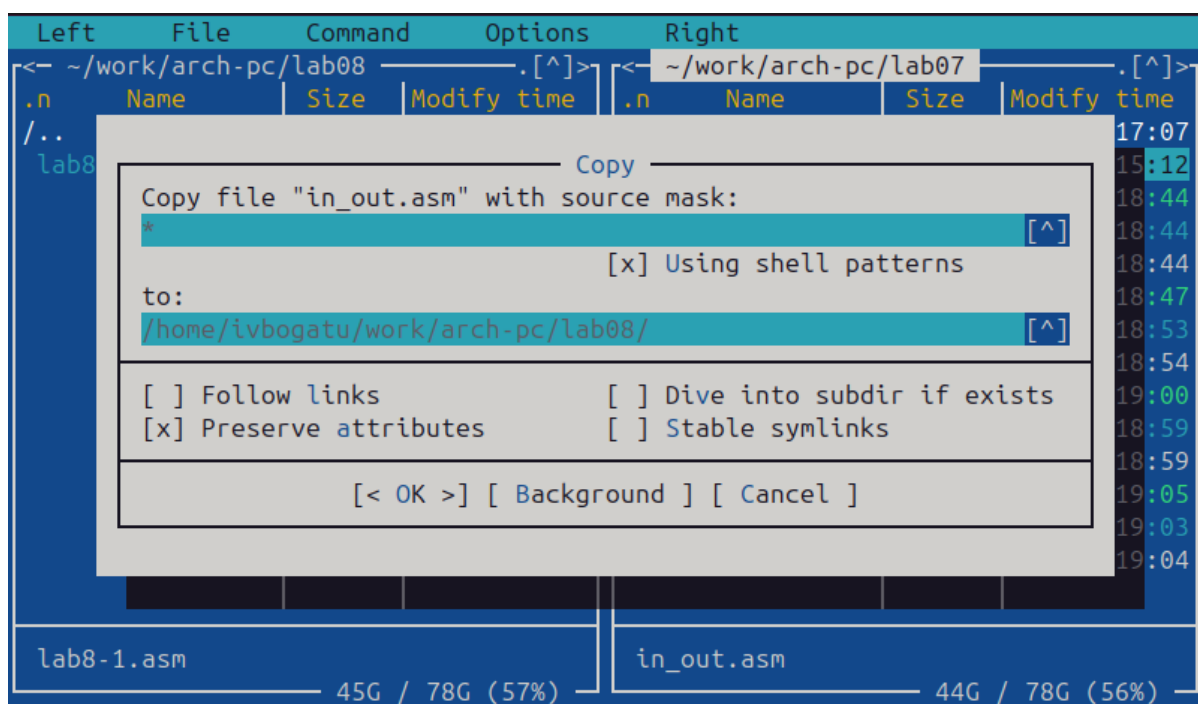


Рис. 4: Копирование файла in_out.asm в рабочую директорию

Теперь соберём программу и посмотрим на результат выполнения (рис. 2.5):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
5
4
3
2
1
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 5: Сборка программы из файла lab8-1.asm и её запуск

Как видим, она выводит числа от N до единицы включительно. Теперь попробуем изменить код, чтобы в цикле также отнималась единица у регистра ecx (рис. 2.6):


```

mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit

```

Рис. 6: Изменение файла lab8-1.asm

Попробуем собрать программу и запустить её (рис. 2.7):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1

```

Рис. 7: Повторная сборка программы из файла lab8-1.asm и её запуск

Введём в качестве N число 5 и посмотрим на результат выполнения (рис. 2.8):

```
4294271388
4294271386
4294271384
4294271382
4294271380
4294271378
4294271376
4294271374
4294271372
4294271370
4294271368
4294271366
4294271364
4294271362
4294271360
4294271358
4294271356
4294271354
4294271352
4294271350
4294271348
4294271346
429427134^C
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 8: Результат вывода

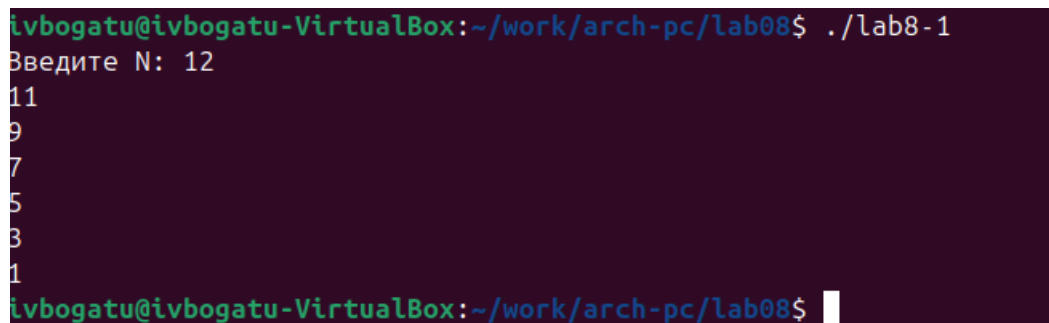
Как видим, цикл выполняется бесконечное количество раз. Это связано с тем, что цикл останавливается в тот момент, когда при проверке `esx` равен 0, но он каждое выполнение цикла уменьшается на 2, из-за чего, в случае нечётного числа, никогда не достигнет нуля. Регистр `esx` меняет своё значение дважды: стандартно -1 после каждой итерации и -1 в теле цикла из-за команды `sub`. Если на вход подать чётное число, цикл прогонится $N/2$ раз, выводя числа от $N-1$ до 1 (выводит через одно число) (рис. 2.9):

Результат вывода для чётного N

Рис. 9: Результат вывода для чётного N

Таким образом, количество итераций цикла не равно N ни при подаче на вход чётного числа, ни при подаче нечётного.

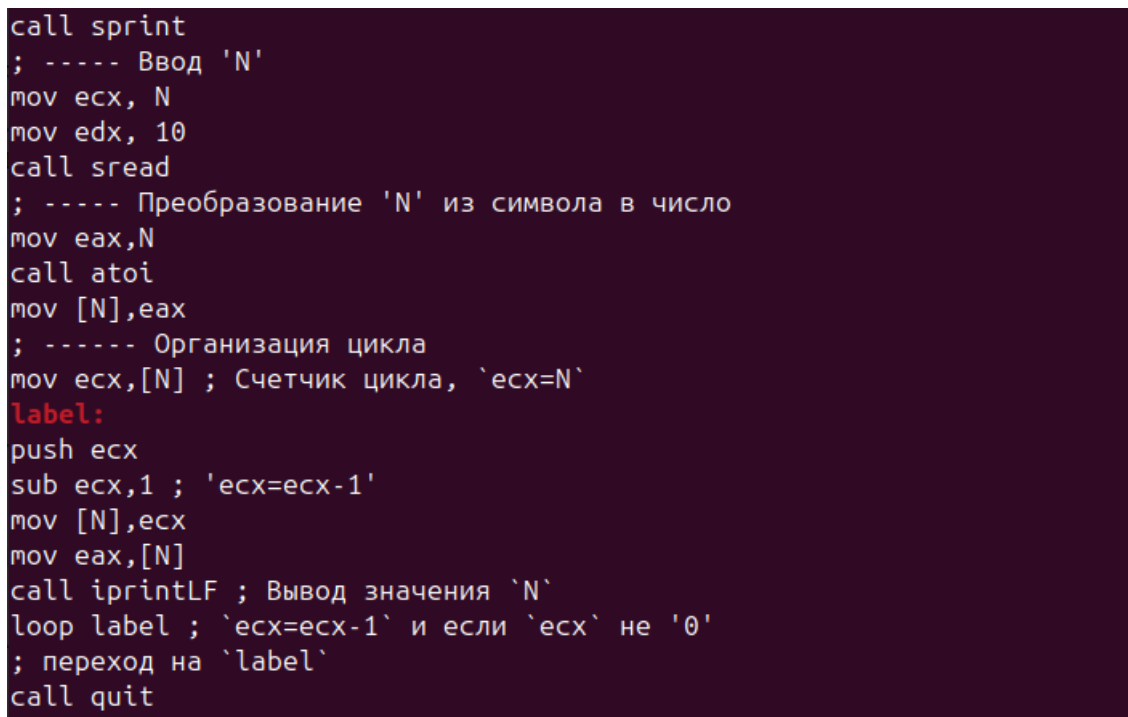
Теперь попробуем изменить программу так, чтобы она сохраняла значение регистра `ecx` в стек (рис. 2.10):



```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 12
11
9
7
5
3
1
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 10: Редактирование файла `lab8-1.asm`

Попробуем собрать и запустить программу (рис. 2.11):



```
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax, N
call atoi
mov [N], eax
; ----- Организация цикла
mov ecx, [N] ; Счетчик цикла, `ecx=N`
label:
push ecx
sub ecx, 1 ; `ecx=ecx-1`
mov [N], ecx
mov eax, [N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
call quit
```

Рис. 11: Повторная сборка программы из файла `lab8-1.asm` и её запуск

Теперь, программа выводит все числа от `N-1` до нуля. Таким образом, число прогонов цикла равно числу `N`. Создадим второй файл (рис. 2.12):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 5
4
3
2
1
0
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 12: Создание второго файла: lab8-2.asm

И вставим в него код из файла листинга 8.2 (рис. 2.13):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ touch lab8-2.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 13: Запись кода из листинга 8.2 в файл lab8-2.asm

Соберём и запустим его, указав некоторые аргументы. Посмотрим на результат (рис. 2.14):

```

GNU nano 7.2                                lab8-2.asm
%include 'in_out.asm'
SECTION .text
global _start
_start:
por ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
por edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
por eax ; иначе извлекаем аргумент из стека
call sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit

```

[Read 20 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

Рис. 14: Сборка программы из файла lab8-2.asm и её запуск

Как видим, он обработал 4 аргумента. Аргументы разделяются пробелом, либо, когда аргумент содержит в себе пробел, обрамляется в кавычки. Создадим третий файл (рис. 2.15):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ mc
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2
аргумент1
аргумент
2
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 15: Создание третьего файла: lab8-3.asm

И вставим в него код из листинга 8.3. Он будет находить сумму всех аргумен-

тов (рис. 2.16):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ touch lab8-3.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 16: Запись кода из листинга 8.3 в файл lab8-3.asm

Теперь соберём программу и запустим её (рис. 2.17):

```
GNU nano 7.2                                lab8-3.asm
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
[ Read 29 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 17: Сборка программы из файла lab8-2.asm и её запуск

Как видим, программа действительно выводит сумму всех аргументов. Изменим её так, чтобы она находила не сумму, а произведение всех аргументов (рис. 2.18):

```

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 18: Изменение файла lab8-3.asm

Соберём программу и запустим её (рис. 2.19):

```

GNU nano 7.2                                lab8-3.asm
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mul esi
mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:
mov ebx, eax
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, ebx ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы
[ Wrote 30 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line

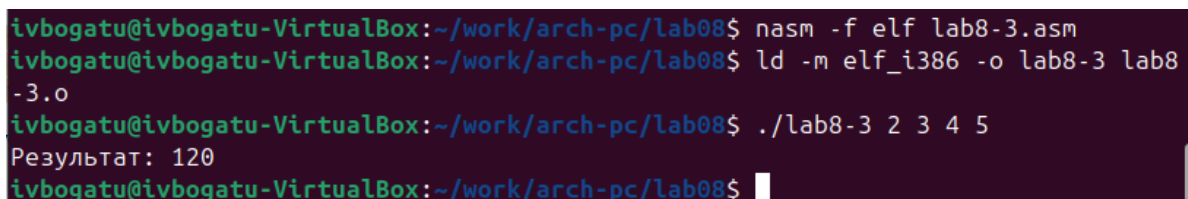
```

Рис. 19: Повторная сборка программы из файла lab8-3.asm и её запуск

Как видим, программа выводит правильный ответ

Выполнение задания для самостоятельной работы

Для выполнения самостоятельной работы создадим файл в формате .asm (рис. 3.1):



```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 2 3 4 5
Результат: 120
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 1: Создание файла самостоятельной работы

В рамках самостоятельной работы необходимо сделать задание под вариантом 9. Так, необходимо сложить результаты выполнения функции $f(x)=10x-4$ для всех введённых аргументов (рис. 3.2 и рис. 3.3):


```
GNU nano 7.2 task1v9.asm
cmp ecx,0h ; проверяем, есть ли ещё аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
mov ebx, 10
mul ebx
sub eax, 4
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент 'esi=esi+eax'
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg2
call sprintf
mov eax, msg ; вывод сообщения "Результат: "
call printf
mov eax, esi ; записываем сумму в регистр 'eax'
call printf ; печать результата
call quit ; завершаем программу
```

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

Рис. 3: Код файла самостоятельной работы (продолжение)

Соберём и запустим программу, вводя различные аргументы (рис. 3.4):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf task1v9.asm
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o task1v9 task1v9.o
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./task1v9 1 2 3 4
Функция: f(x)=10x-4
Результат: 44
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./task1v9 5 10 15 20
Функция: f(x)=10x-4
Результат: 284
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$ ./task1v9 5 3 6 4 54 6
Функция: f(x)=10x-4
Результат: 106
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4: Сборка и запуск программы первого задания самостоятельной работы, а также результат выполнения

Пересчитав результат вручную, убеждаемся, что программа работает верно

Выводы

В результате выполнения лабораторной работы были получены навыки работы с циклами и обработкой аргументов из командной строки. Были написаны программы, использующие все вышеописанные аспекты