

# **Лабораторная работа №10.**

**Понятие подпрограммы. Отладчик GDB**

Коршунова Полина Юрьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Задания для самостоятельной работы</b>	<b>18</b>
<b>4</b>	<b>Выводы</b>	<b>21</b>

# Список иллюстраций

2.1	Создание каталога и файла . . . . .	6
2.2	Работа программы . . . . .	6
2.3	Работа измененной программы . . . . .	6
2.4	Отладка второго файла . . . . .	7
2.5	Брежпоинт на метку _start . . . . .	7
2.6	Дисассимпированный код . . . . .	8
2.7	Intel'овское отображение . . . . .	9
2.8	Псевдографика . . . . .	10
2.9	Наличие меток . . . . .	10
2.10	Просмотр регистров . . . . .	11
2.11	Измененные регистры . . . . .	12
2.12	Просмотрим значения переменной . . . . .	12
2.13	Значение переменной msg2 . . . . .	12
2.14	Изменение значения переменной . . . . .	13
2.15	Изменение msg2 . . . . .	13
2.16	Значение регистров ехх и еах . . . . .	14
2.17	Значение регистров ебх . . . . .	15
2.18	Завершение работы с файлов . . . . .	15
2.19	Запуск файла в отладчике . . . . .	16
2.20	Запуск файла lab10-3 через метку . . . . .	16
2.21	Адрес вершины стека . . . . .	17
2.22	Все позиции стека . . . . .	17
3.1	Запуск программы . . . . .	18
3.2	Запуск программы . . . . .	18
3.3	Запуск программы в отладчике . . . . .	19
3.4	Анализ регистров . . . . .	19
3.5	Повторный запуск программы . . . . .	20

## Список таблиц

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Выполнение лабораторной работы

1. Создаю каталог для выполнения лабораторной работы No 10, перехожу в него и создаю файл lab10-1.asm (рис. 2.1)

```
pykorshunova@dk8n81 ~ $ mkdir ~/work/arch-pc/lab10
pykorshunova@dk8n81 ~ $ cd ~/work/arch-pc/lab10
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ touch lab10-1.asm
```

Рис. 2.1: Создание каталога и файла

2. Я ввела текст листинга в файл и запустила программу (рис. 2.2)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ./lab10-1
Введите x: 8
2x+7=23
```

Рис. 2.2: Работа программы

3. Я изменила текст программы, чтобы она решала выражение  $f(g(x))$  (рис. 2.3)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf lab10-1.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ./lab10-1
f(x)=2x+7
g(x)=3x-1
Введите x: 8
f(g(x))=0
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $
```

Рис. 2.3: Работа измененной программы

4. Я создала файл lab10-2.asm и вписала туда программу, запустила файл второй программы в отладчик gdb (рис. 2.4)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-2.lst lab10-2.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-2 lab10-2.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ gdb lab10-2
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab10-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/y/pykorshunova/work/arch-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 10893) exited normally]
(gdb) □
```

Рис. 2.4: Отладка второго файла

5. Я поставила брекпоинт на метку \_start и запустила программу (рис. 2.5)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/y/pykorshunova/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:9
]
```

Рис. 2.5: Брекпоинт на метку \_start

6. Я просмотрела дисассимплированный код программы начиная с метки (рис. 2.6)

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
    0x08049005 <+5>:      mov     $0x1,%ebx
    0x0804900a <+10>:     mov     $0x804a000,%ecx
    0x0804900f <+15>:     mov     $0x8,%edx
    0x08049014 <+20>:     int     $0x80
    0x08049016 <+22>:     mov     $0x4,%eax
    0x0804901b <+27>:     mov     $0x1,%ebx
    0x08049020 <+32>:     mov     $0x804a008,%ecx
    0x08049025 <+37>:     mov     $0x7,%edx
    0x0804902a <+42>:     int     $0x80
    0x0804902c <+44>:     mov     $0x1,%eax
    0x08049031 <+49>:     mov     $0x0,%ebx
    0x08049036 <+54>:     int     $0x80
End of assembler dump.
(gdb) 


```

Рис. 2.6: Дисассимплированный код

7. С помощью команды я переключилась на intel'овское отображение синтаксиса. Отличие заключается в командах, в дисассимилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть (рис. 2.7)



```

(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
    0x08049005 <+5>:      mov     ebx,0x1
    0x0804900a <+10>:     mov     ecx,0x804a000
    0x0804900f <+15>:     mov     edx,0x8
    0x08049014 <+20>:     int     0x80
    0x08049016 <+22>:     mov     eax,0x4
    0x0804901b <+27>:     mov     ebx,0x1
    0x08049020 <+32>:     mov     ecx,0x804a008
    0x08049025 <+37>:     mov     edx,0x7
    0x0804902a <+42>:     int     0x80
    0x0804902c <+44>:     mov     eax,0x1
    0x08049031 <+49>:     mov     ebx,0x0
    0x08049036 <+54>:     int     0x80
End of assembler dump.
(gdb) 

```

Рис. 2.7: Intel'овское отображение

8. Для удобства я включила режим псевдографики (рис. 2.8)

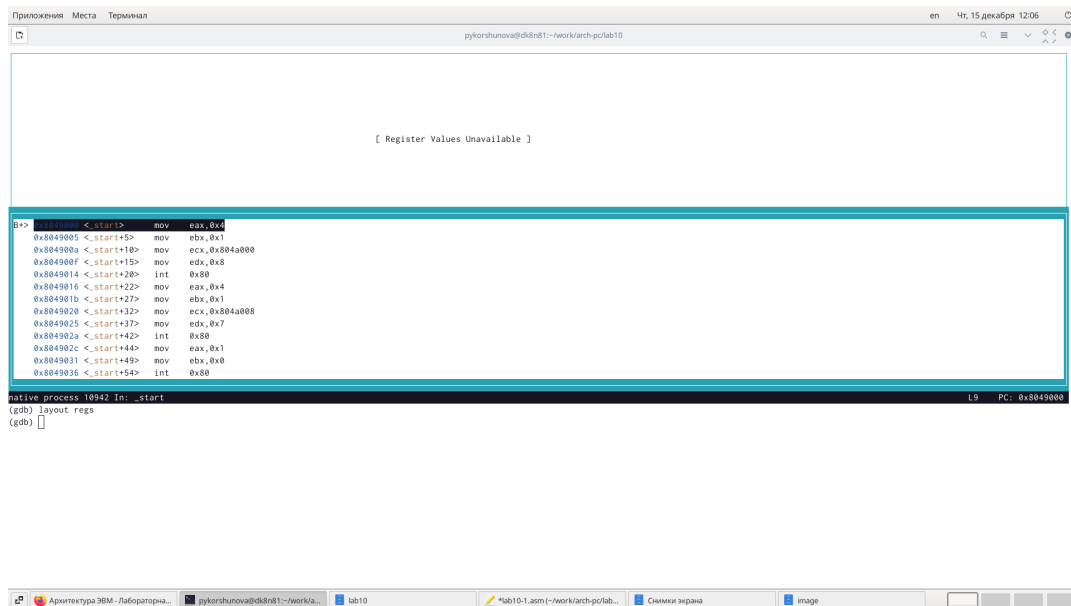


Рис. 2.8: Псевдографика

9. Я посмотрела наличие меток и добавила еще одну метку на предпоследнюю инструкцию (рис. 2.9)

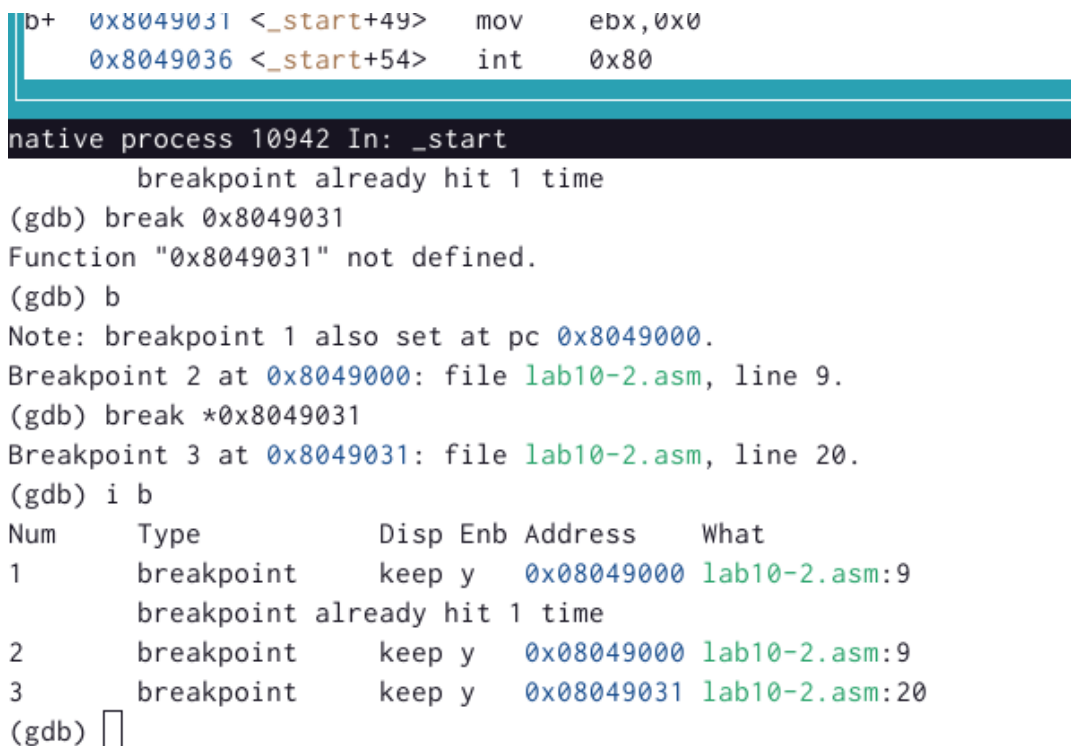


Рис. 2.9: Наличие меток

10. С помощью команды `si` я посмотрела регистры и изменила их (рис. 2.10) (рис. 2.11)

```
B+ 0x8049000 <_start>      mov     eax,0x4
> 0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>     mov     ecx,0x804a000
0x804900f <_start+15>     mov     edx,0x8
0x8049014 <_start+20>     int      0x80
0x8049016 <_start+22>     mov     eax,0x4
0x804901b <_start+27>     mov     ebx,0x1
0x8049020 <_start+32>     mov     ecx,0x804a008
0x8049025 <_start+37>     mov     edx,0x7
0x804902a <_start+42>     int      0x80
0x804902c <_start+44>     mov     eax,0x1
b+ 0x8049031 <_start+49>   mov     ebx,0x0
0x8049036 <_start+54>     int      0x80

native process 10942 In: _start
(gdb) break 0x8049031
Function "0x8049031" not defined.
(gdb) b
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab10-2.asm, line 9.
(gdb) break *0x8049031
Breakpoint 3 at 0x8049031: file lab10-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08049000 lab10-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y   0x08049000 lab10-2.asm:9
3        breakpoint      keep y   0x08049031 lab10-2.asm:20
(gdb) si
(gdb) ☐
```

Рис. 2.10: Просмотр регистров

ecx	0x0	0
esp	0xfffffc610	0xfffffc610
edi	0x0	0
cs	0x23	35
es	0x2b	43

Рис. 2.11: Измененные регистры

11. С помощью команды я посмотрела значение переменной msg1 (рис. 2.12)

```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) 
```

---

Рис. 2.12: Просмотрим значения переменной

12. Следом я посмотрела значение второй переменной msg2 (рис. 2.13)

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) 
```

---

Рис. 2.13: Значение переменной msg2

13. С помощью команды set я изменила значение переменной msg1 (рис. 2.14)

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:          "hhlllo, "
(gdb) 
```

---

Рис. 2.14: Изменение значения переменной

14. Я изменила переменную msg2 (рис. 2.15)

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:          "Lor d!\n\034"
(gdb) 
```

---

Рис. 2.15: Изменение msg2

15. Я вывела значение регистров esx и eax (рис. 2.16)

```
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb) 
```

---

Рис. 2.16: Значение регистров ecx и eax

16. Я изменила значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные (рис. 2.17)

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb) 
```

---

Рис. 2.17: Значение регистров ebx

17. Я завершила работу с файлов вышел (рис. 2.18)

```
0x0804902a <+42>:    int    0x80
0x0804902c <+44>:    mov     eax,0x1
0x08049031 <+49>:    mov     ebx,0x0
0x08049036 <+54>:    int    0x80
End of assembler dump.
(gdb) layout asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ 
```

---

Рис. 2.18: Завершение работы с файлов

18. Я скопировала файл lab9-2.asm и переименовал его. Запустила файл в отладчике и указала аргументы (рис. 2.19)

```

pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ gdb --args lab10-3 8 17 '24'
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab10-3...
(gdb) █

```

---

Рис. 2.19: Запуск файла в отладчике

19. Поставила метку на `_start` и запустила файл (рис. 2.20)

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/y/pykorshunova/work/arch-pc/lab10/lab10-3 8 17 24

Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) █

```

---

Рис. 2.20: Запуск файла lab10-3 через метку

20. Я проверила адрес вершины стека и убедилась что там хранится 5 элементов (рис. 2.21)



```

(gdb) x/x $esp
0xfffffc610:      0x00000004
(gdb) 

```

Рис. 2.21: Адрес вершины стека

21. Я посмотрел все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации (рис. 2.22)

```

(gdb) x/x $esp
0xfffffc610:      0x00000004
(gdb) x/s *(void**)( $esp + 4)
0xfffffc863:      "/afs/.dk.sci.pfu.edu.ru/home/p/y/pykorshunova/work/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffc8ac:      "8"
(gdb) x/s *(void**)( $esp + 12)
0xfffffc8ae:      "17"
(gdb) x/s *(void**)( $esp + 16)
0xfffffc8b1:      "24"
(gdb) x/s *(void**)( $esp + 20)
0x0:      <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)( $esp + 24)
0xfffffc8b4:      "SHELL=/bin/bash"
(gdb) 

```

Рис. 2.22: Все позиции стека

### 3 Задания для самостоятельной работы

1. Я преобразовала программу из лабораторной работы No9 и реализовала вычисления как подпрограмму (рис. 3.1)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-4.lst lab10-4.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-4 lab10-4.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ./lab10-4 5 9
f (x) =2x+15
Результат: 58
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ./lab10-4 5 3
f (x) =2x+15
Результат: 46
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ █
```

Рис. 3.1: Запуск программы

2. Я переписала программу и попробовала запустить ее чтобы увидеть ошибку. Ошибка была арифметическая, так как вместо 25, программа выводит 10. (рис. 3.2) После появления ошибки, я запустил программу в отладчике (рис. 3.3)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ touch lab10-5.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf lab10-5.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-5 lab10-5.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ./lab10-5
Результат: 10
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ █
```

Рис. 3.2: Запуск программы

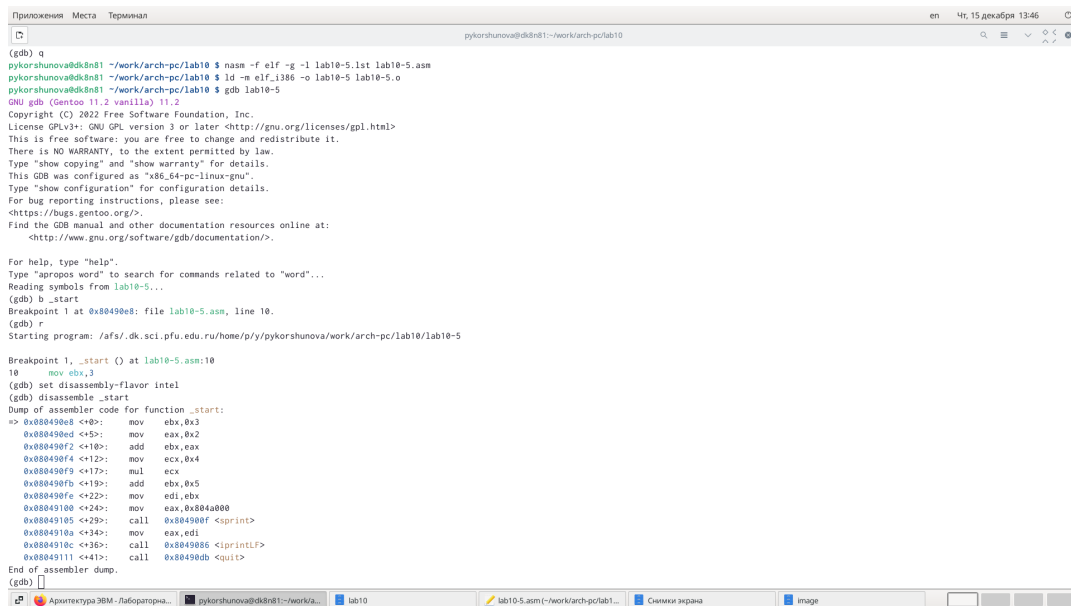


Рис. 3.3: Запуск программы в отладчике

3. Я открыла регистры и проанализировала их, поняла что некоторые регистры стоят не на своих местах и исправила это (рис. 3.4)



Рис. 3.4: Анализ регистров

4. Я изменила регистры и запустила программу, программа вывела ответ 25,

то есть все работает правильно (рис. 3.5)

```
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ nasm -f elf -g -l lab10-5.lst lab10-5.asm
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ ld -m elf_i386 -o lab10-5 lab10-5.o
pykorshunova@dk8n81 ~/work/arch-pc/lab10 $ gdb lab10-5
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab10-5...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/p/y/pykorshunova/work/arch-pc/lab10/lab10-5
Результат: 25
[Inferior 1 (process 19415) exited normally]
(gdb) □
```

Рис. 3.5: Повторный запуск программы

## 4 Выводы

Я приобрела навыки написания программ использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями