

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ЛАБОРАТОРНАЯ РАБОТА № 9

дисциплина: *Архитектура компьютера*

Студент: Мизинов М.Г.

Группа: НКАбд-04-25

№ ст. билета: 1032253549

МОСКВА

2025 г.

СОДЕРЖАНИЕ

Список иллюстраций	3
Основная часть	4
1. Цель работы.....	4
2. Теоретическое введение	4
3. Выполнение лабораторной работы	5
4. Задание для самостоятельной работы	13
Выводы	16
Список литературы	17

Список иллюстраций

Рисунок 1 – Создание каталога и lab9-1.asm	5
Рисунок 2 – Файл lab9-1.asm.....	5
Рисунок 3 – Выполнение lab9-1.asm	5
Рисунок 4 – Изменённый lab9-1.asm	6
Рисунок 5 – Запуск изменённого lab9-1.asm.....	6
Рисунок 6 – Файл lab9-2.asm.....	7
Рисунок 7 – Загрузка в отладчик gdb	7
Рисунок 8 – Проверка работы программы в gdb	8
Рисунок 9 – брейкпоинт	8
Рисунок 10 – Команда disassemble	8
Рисунок 11 – Intel синтаксис	9
Рисунок 12 – Режим псевдографики	9
Рисунок 13 – info breakpoints	10
Рисунок 14 – Адрес предпоследней инструкции	10
Рисунок 15 – Информация о установленных точках останова	10
Рисунок 16 – Содержимое регистров	11
Рисунок 17 – Значение переменной msg1	11
Рисунок 18 – Изменение символов.....	11
Рисунок 19 – Изменение значения регистра ebx	12
Рисунок 20 – Копия lab8-2.asm	12
Рисунок 21 – Создание lab9-3.asm и загрузка в откладчик	12
Рисунок 22 – Запуск lab9-3.asm и адрес вершины стека	13
Рисунок 23 – Позиция стека.....	13
Рисунок 24 – Преобразованная программа.....	14
Рисунок 25 – Работа lab9-4.asm	14
Рисунок 26 – Неверный результат.....	15
Рисунок 27 – Исправленная программа	15
Рисунок 28 – Верный результат.....	15

Основная часть

1. Цель работы

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

2. Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки;
- поиск её местонахождения;
- определение причины ошибки;
- исправление ошибки.

3. Задание

На основе методических указаний провести ознакомительную работу с методами отладки при помощи GDB.

4. Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

Создал каталог для выполнения лабораторной работы № 9 и файл lab9-1.asm (рис. 1):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ cd ~/Mizinov-study_2025-2026_arh-pc/labs/
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ mkdir lab9
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs$ cd lab9
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ touch lab9-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$
```

Рис. 1: Создание каталога и lab9-1.asm

Записал код из листинга в lab9-1.asm (рис. 2):

```
lab9-1.asm
-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax,result
call sprint
mov eax,[res]
call iprintfLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
```

Рис. 2: Файл lab9-1.asm

Запустил программу lab9-1.asm (рис. 3):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf lab9-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./lab9-1
Введите x: 4
2x+7=15
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$
```

Рис. 3: Выполнение lab9-1.asm

Изменил код программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul` (рис. 4):

```
lab9-1.asm x
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ', 0
result: DB '2(3x-1)+7=', 0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprint

    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi

    call _calcul

    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit

_calcul:
    push eax
    call _subcalcul

    mov ebx, 2
    mul ebx
    add eax, 7
```

Рис. 4: Изменённый lab9-1.asm

Запустил изменённый файл (рис.5):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf lab9-1.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./lab9-1
Введите x: 1
2x+7=9
```

Рис. 5: Запуск изменённого lab9-1.asm

4.2 Отладка программам с помощью GDB

Создал файл lab9-2.asm с текстом программы из Листинга 9.2 (рис.6):

```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2
SECTION .text
global _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, msg1Len
    int 0x80
    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, msg2Len
    int 0x80
    mov eax, 1
    mov ebx, 0
    int 0x80|
```

Рис. 6: Файл lab9-2.asm

Получил исполняемый файл. Добавил отладочную информацию, трансляцию программ проводил с ключом ‘-g’. Загрузил исполняемый файл в отладчик gdb (рис.7):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/Lab9$ nasm -f elf lab9-2.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-2 lab9-2.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./lab9-2
Hello, world!
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
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 lab9-2...
(No debugging symbols found in lab9-2)
(gdb)
```

Рис. 7: Загрузка в отладчик gdb.

Проверил работу программы (рис.8):

```
(gdb) run
Starting program: /home/mgmizinov/Mizinov-study_2025-2026_arh-pc/labs/lab9/lab9-2
Hello, world!
[Inferior 1 (process 5031) exited normally]
(gdb) █
```

Рис. 8: Проверка работы программы в gdb.

Установил брейкпоинт на метку _start (рис.9):

```
[Inferior 1 (process 5031) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/mgmizinov/Mizinov-study_2025-2026_arh-pc/labs/lab9/lab9-2

Breakpoint 1, 0x08049000 in _start ()
(gdb) █
```

Рис. 9: брейкпоинт

Посмотрел дисассимилированный код программы с помощью команды disassemble начиная с метки _start (рис. 10):

```
(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.
```

Рис. 10: Команда disassemble

Переключился на отображение команд с Intel синтаксисом (рис.11):

```
(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. 11: Intel синтаксис

Различия между синтаксисом ATT и Intel заключаются в порядке операндов, их размере, именах регистров

Включил режим псевдографики (рис.12):

```
Register group: general
eax      0x0          0
ecx      0x0          0
edx      0x0          0
ebx      0x0          0
esp      0xfffffd1d0  0xfffffd1d0
ebp      0x0          0x0
esi      0x0          0
edi      0x0          0

B+>0x08049000 <_start>    mov    eax,0x4
    0x08049005 <_start+5>  mov    ebx,0x1
    0x0804900a <_start+10>  mov    ecx,0x804a000
    0x0804900f <_start+15>  mov    edx,0x8
    0x08049014 <_start+20>  int    0x80
    0x08049016 <_start+22>  mov    eax,0x4
    0x0804901b <_start+27>  mov    ebx,0x1
    0x08049020 <_start+32>  mov    ecx,0x804a008

native process 1264 (asm) In: _start          L9    PC: 0x08049000
(gdb) layout regs
(gdb) |
```

Рис. 12: режим псевдографики

4.3 Добавление точек останова

На предыдущих шагах была установлена точка останова по имени метки.

Проверил это с помощью команды info breakpoints(рис.13):

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xfffffd1d0 0xfffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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,0x804a000

native process 1264 (asm) In: _start          L9    PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num  Type      Disp Enb Address  What
1    breakpoint  keep y  0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
(gdb) |
```

Рис. 13: info breakpoints

Определил адрес предпоследней инструкции (mov ebx,0x0) и установил точку останова (рис.14):

```
(gdb) break *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab9-2.asm, line 9.
(gdb) |
```

Рис. 14: Адрес предпоследней инструкции

Посмотрел информацию о всех установленных точках останова (рис.15):

```
(gdb) i b
Num  Type      Disp Enb Address  What
1    breakpoint  keep y  0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
2    breakpoint  keep y  0x08049000 lab9-2.asm:9
(gdb) |
```

Рис. 15: Информация о установленных точках останова

4.4 Работа с данными программы в GDB

Посмотрел содержимое регистров (рис.16):

The screenshot shows the GDB interface. At the top, it displays the general register group:

eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xfffffd1d0	0xfffffd1d0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

Below this, the assembly code for the program's entry point (`_start`) is shown:

```
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,0x804a000
```

At the bottom, the current state is summarized:

native process 1264 (asm) In: _start L9 PC: 0x8049000
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xfffffd1d0 0xfffffd1d0
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
--type <RET> for more, q to quit, c to continue without paging--

Рис. 16: Содержимое регистров

Посмотрел значение переменной msg1 по имени (рис. 17):

The screenshot shows the GDB command `x/1sb &msg1` being run, which displays the memory location of the variable `msg1`:

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

Рис. 17: Значение переменной msg1

Изменил первый символ переменной msg1 и заменил случайный символ во второй переменной msg2 (рис. 18):

The screenshot shows a sequence of GDB commands used to modify the values of variables `msg1` and `msg2`:

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "xworld!\n\034"
(gdb)
```

Рис. 18: Изменение символов

С помощью команды set изменил значение регистра ebx (рис. 18):

The screenshot shows the GDB command `set $ebx=2` being run, followed by `p/s $ebx` to print the new value:

```
(gdb) set $ebx=2
(gdb) p/s $ebx
$2 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$3 = 2
(gdb) |
```

Рис. 19: Изменение значения регистра ebx

Разница вывода команд `p/s $ebx` в том, что в `set $ebx=2'`

это, а не число, а символ '2' имеет код 50. Команда set \$ebx='2' записывает в регистр код символа '2', то есть число 50

В set \$ebx=2 2 - это число. Команда записывает в регистр EBX число 2

4.5 Обработка аргументов командной строки в GDB

Скопировал файл lab8-2.asm в файл с именем lab9-3.asm (рис. 20):

```
mORIZINOV@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ cp ~/Mizinov-study_2025-2026_arh-pc/labs/lab8/lab8-2.asm ~/Mizinov-study_2025-2026_arh-pc/labs/lab9-3.asm
```

Рис. 20: Копия lab8-2.asm

Создал исполняемый файл и загрузила его в отладчик (рис. 21):

```
mORIZINOV@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
mORIZINOV@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-3 lab9-3.o
mORIZINOV@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ gdb --args lab9-3 argument1 argument2 'argument 3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
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 lab9-3...
(gdb) 
```

Рис. 21: Создание lab9-3.asm и загрузка в отладчик

Установил точку останова перед первой инструкцией в программе и запустил ее (рис. 22):

```
Type apropos word to search for commands related to word ...
Reading symbols from lab9-3...
(gdb) b start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/mORIZINOV/Mizinov-study_2025-2026_arh-pc/labs/lab9/lab9-3 argument1 argument2 argument\ 3

Breakpoint 1, start () at lab9-3.asm:5
5      pop    ecx ; Извлекаем из стека в `ecx` количество
(gdb) 
```

Рис. 22: Запуск lab9-3.asm и адрес вершины стека

Посмотрел остальные позиции стека (рис. 23):

```
5      pop    ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/s *(void**)$esp +4)
0xfffffd1f0:    "/home/mORIZINOV/Mizinov-study_2025-2026_arh-pc/labs/lab9/lab9-3"
(gdb) x/s *(void**)$esp +8)
0xfffffd230:    "argument1"
(gdb) x/s *(void**)$esp +12)
0xfffffd23a:    "argument"
(gdb) x/s *(void**)$esp +16)
0xfffffd243:    "2"
(gdb) x/s *(void**)$esp +20)
0xfffffd245:    "argument 3"
(gdb) x/s *(void**)$esp +24)
0x0:   <error: Cannot access memory at address 0x0>
(gdb) 
```

Рис. 23: Позиции стека

Шаг равен 4, потому что в 32-битной архитектуре размер каждого элемента в стеке составляет 4 байта (32 бита = 4 байта).

5. Задание для самостоятельной работы

1) Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции как подпрограмму (рис. 24):

```
%include 'in_out.asm'

SECTION .data
msg_func db "@Функция: f(x) = 2x + 15", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprintLF
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0
    jz _end
    pop eax      ; берем очередное число
    call atoi     ; преобразуем строку в число, результат в eax
    push eax     ; сохраним х для передачи в func
    call func     ; вызов функции f(x) = 2x + 15
    add esi, eax ; суммируем результат
    loop next

_end:
    mov eax, msg_result
    call sprint
    mov eax, esi
    call iprintLF
    call quit

; Подпрограмма func:
; Вход: входное значение x в eax
; Выход: результат f(x) = 2*x + 15 в eax
func:
    push ebx      ; сохраняем используемые регистры
    mov ebx, 2
    mul ebx      ; умножаем eax на 2: результат в edx:eax, edx = старшая часть
    add eax, 15   ; eax = 2*x + 15
    pop ebx
    ret|
```

Рис. 24: Преобразованная программа

Запуск преобразованной программы (рис. 25):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf uniq9.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o uniq9 uniq9.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./uniq9
Функция: f(x) = 2x + 15
Результат: 0
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./uniq9 5
Функция: f(x) = 2x + 15
Результат: 25
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$
```

Рис. 25: Работа lab9-4.asm

В листинге 9.3 приведена программа вычисления выражения

При запуске данная программа дает неверный результат (рис. 26):

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf lab9-4.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-4 lab9-4.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./lab9-4
Результат: 10
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$
```

Рис. 26: Неверный результат

Исправим программу (рис. 27):

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start

_start:
; ----- Вычисление выражения (3+2)*4+5

mov ebx,3
mov eax,2
add ebx, eax
mov eax, ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ----- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 27: Исправленная программа

```
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ nasm -f elf lab9-4.asm
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ld -m elf_i386 -o lab9-4 lab9-4.o
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$ ./lab9-4
Результат: 25
mgmizinov@mint:~/Mizinov-study_2025-2026_arh-pc/labs/lab9$
```

Рис. 28: Верный результат

Ссылка на github: https://github.com/MihailMizinov/Mizinov-study_2025-2026_arh-pc

Вывод

При выполнении данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм. Познакомился с методами отладки при помощи GDB и его основными возможностями

Список литературы

1) Лабораторная работа №9.

[https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%
%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%
0%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%
B0%20%E2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%
B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%
B0%D0%BD%D0%B8%D0%BD%D0%
B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%
B0.%20%D0%9E%D0%B1%D1%
80%D0%B0%D0%
B1%D0%BE%D1%82%D0%BA%D0%
B0%20%D0%
B0%D1%80%D0%
B3%D1%83%D0%BC%D0%
B5%D0%BD%D1%82%D0%BE%D0%
B2%20%D0%
BA%D0%BE%D0%BC%D0%
B0%D0%BD%D0%
B4%D0%BD%D0%BE%D0%
B9%20%D1%81%D1%82%D1%80%D0%
BE%D0%BA%D0%
B8..pdf](https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%
%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%
0%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%
B0%20%E2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%
B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%
B0%D0%BD%D0%B8%D0%BD%D0%
B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%
B0.%20%D0%9E%D0%B1%D1%
80%D0%B0%D0%
B1%D0%BE%D1%82%D0%BA%D0%
B0%20%D0%
B0%D1%80%D0%
B3%D1%83%D0%BC%D0%
B5%D0%BD%D1%82%D0%BE%D0%
B2%20%D0%
BA%D0%BE%D0%BC%D0%
B0%D0%BD%D0%
B4%D0%BD%D0%BE%D0%
B9%20%D1%81%D1%82%D1%80%D0%
BE%D0%BA%D0%
B8..pdf)

2) Википедия. <https://en.wikipedia.org/wiki/GitHub>