

Лабораторная работа №9

Архитектура компьютера

Косолапов Матвей Эдуардович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание каталога lab09 и файла lab9-1.asm	8
4.2	Текст программы №1	9
4.3	Создание исполняемого файла, проверка его работы	9
4.4	Изменение текста программы	10
4.5	Результат выполнения измененной программы	10
4.6	Новое изменение программы	10
4.7	Результат работы программы с новым изменением	11
4.8	Программа №2	11
4.9	Создание исполняемого файла, проверка его работы	12
4.10	Программа №3	12
4.11	Проверяем работу файла	13
4.12	Программа №4	13
4.13	Создание исполняемого файла, проверка его работы	13
4.14	Код программы	14
4.15	Создание исполняемого файла, проверка его работы	14

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

3 Теоретическое введение

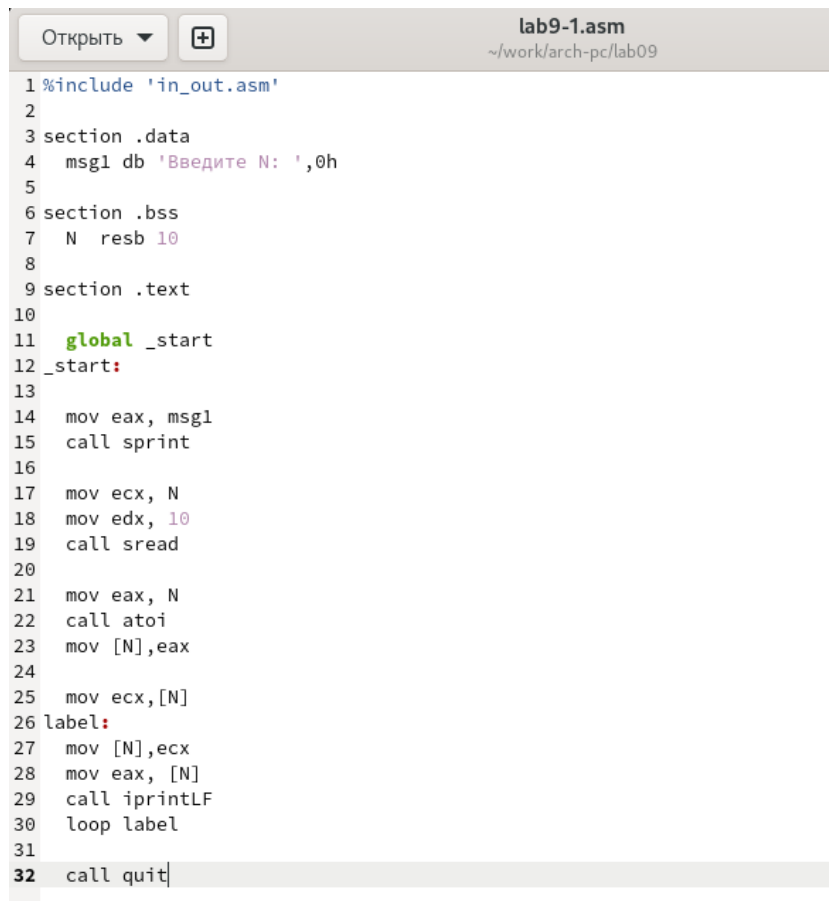
4 Выполнение лабораторной работы

1. Создаём каталог lab09, в нём создаём файл lab9-1.asm(рис. 4.1):

```
[mekosolapov@fedora arch-pc]$ mkdir ~/work/arch-pc/lab09
[mekosolapov@fedora arch-pc]$ cd lab09
[mekosolapov@fedora lab09]$ ls
[mekosolapov@fedora lab09]$ touch lab9-1.asm
[mekosolapov@fedora lab09]$ ls
lab9-1.asm
[mekosolapov@fedora lab09]$
```

Рис. 4.1: Создание каталога lab09 и файла lab9-1.asm

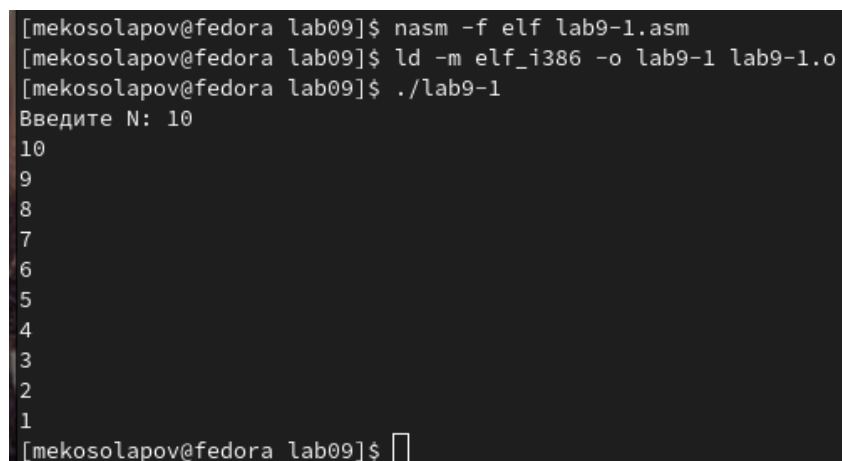
2. Переносим в файл программу из листинга №1(рис. 4.2):



```
1 %include 'in_out.asm'
2
3 section .data
4     msg1 db 'Введите N: ',0h
5
6 section .bss
7     N resb 10
8
9 section .text
10
11     global _start
12     _start:
13
14     mov eax, msg1
15     call sprint
16
17     mov ecx, N
18     mov edx, 10
19     call sread
20
21     mov eax, N
22     call atoi
23     mov [N],eax
24
25     mov ecx,[N]
26 label:
27     mov [N],ecx
28     mov eax, [N]
29     call iprintLF
30     loop label
31
32     call quit
```

Рис. 4.2: Текст программы №1

3. Создаём исполняемый файл, проверяем работу. Выводится 10 цифр [от 10 до 0], как и было введено(рис. 4.3):



```
[mekosolapov@fedora lab09]$ nasm -f elf lab9-1.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[mekosolapov@fedora lab09]$ ./lab9-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
[mekosolapov@fedora lab09]$
```

Рис. 4.3: Создание исполняемого файла, проверка его работы

4. Меняем программу, добавляя команду **sub ecx, [N]** (рис. 4.4):

```
20  
21 mov eax, N  
22 call atoi  
23 mov [N],eax  
24  
25 mov ecx,[N]  
26 label:  
27 sub ecx,1  
28 mov [N],ecx  
29 mov eax, [N]  
30 call iprintLF  
31 loop label  
32  
33 call quit
```

Рис. 4.4: Изменение текста программы

5. Проверяем работу программы. Теперь нам выводятся нечётные числа от 0 до 10(5 чисел)(рис. 4.5):

```
Введите N: 10  
9  
7  
5  
3  
1  
[mekosolapov@fedora lab09]$
```

Рис. 4.5: Результат выполнения измененной программы

6. Снова меняем программу, добавляя команду **push ecx** (рис. 4.6):

```
26 label:  
27 push ecx  
28 sub ecx,1  
29 mov [N],ecx  
30 mov eax, [N]  
31 call iprintLF  
32 pop ecx  
33 loop label  
34  
35 call quit
```

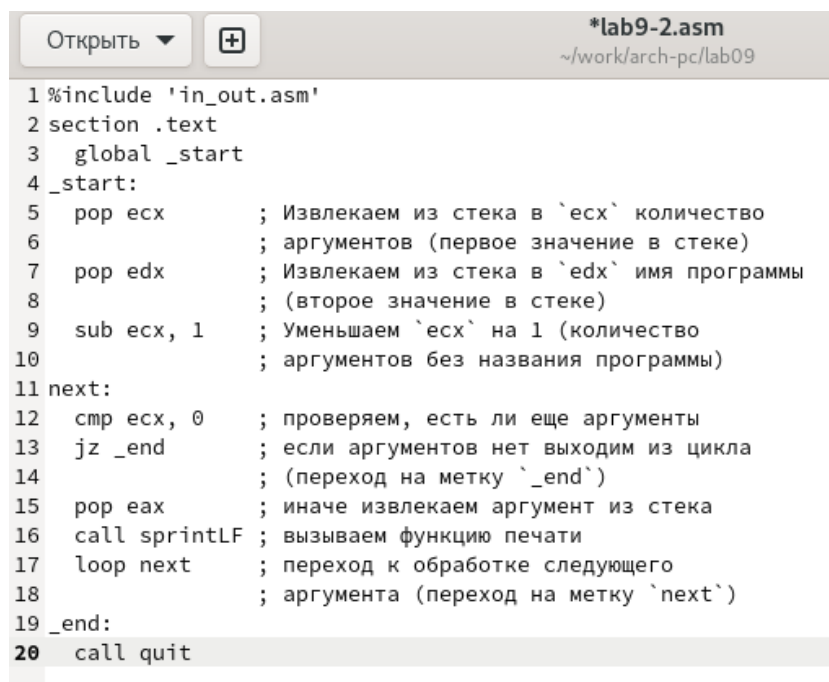
Рис. 4.6: Новое изменение программы

7. Проверяем работу программы. Теперь нам выводятся числа от 9 до 0 (рис. 4.7)

```
[mekosolapov@fedora lab09]$ nasm -f elf lab9-1.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[mekosolapov@fedora lab09]$ ./lab9-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 4.7: Результат работы программы с новым изменением

8. Создаём файл lab9-2.asm и переносим предложенную программу из листинга №2 (рис. 4.8)



```
*lab9-2.asm
~/work/arch-pc/lab09

1 %include 'in_out.asm'
2 section .text
3 global _start
4 _start:
5     pop ecx        ; Извлекаем из стека в `ecx` количество
6                   ; аргументов (первое значение в стеке)
7     pop edx        ; Извлекаем из стека в `edx` имя программы
8                   ; (второе значение в стеке)
9     sub ecx, 1     ; Уменьшаем `ecx` на 1 (количество
10                  ; аргументов без названия программы)
11 next:
12     cmp ecx, 0     ; проверяем, есть ли еще аргументы
13     jz _end        ; если аргументов нет выходим из цикла
14                   ; (переход на метку `_end`)
15     pop eax        ; иначе извлекаем аргумент из стека
16     call sprintf   ; вызываем функцию печати
17     loop next      ; переход к обработке следующего
18                   ; аргумента (переход на метку `next`)
19 _end:
20 call quit
```

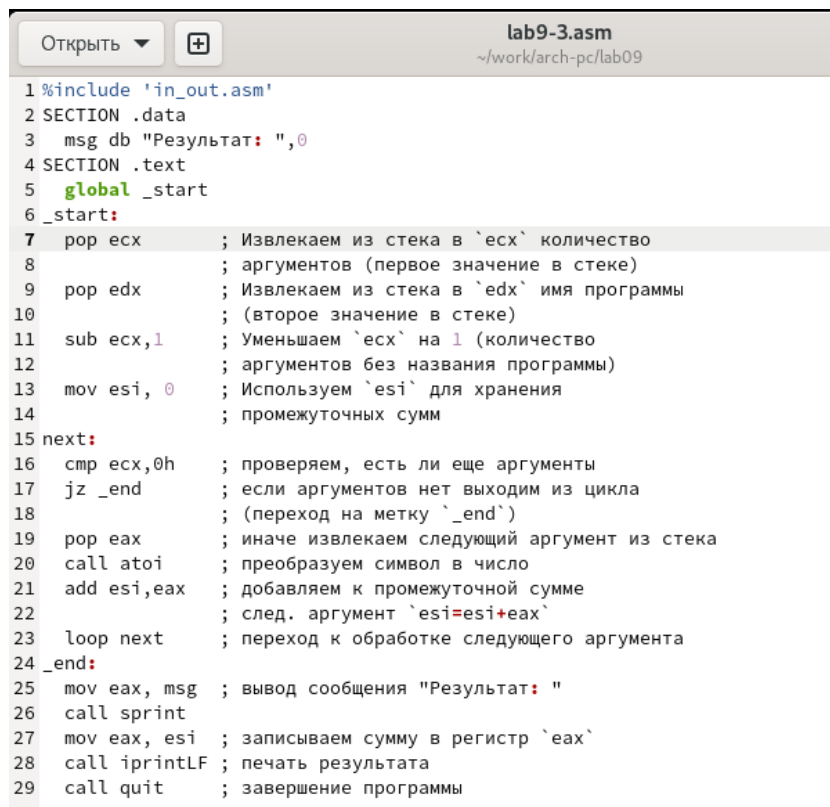
Рис. 4.8: Программа №2

9. Создаём исполняемый файл и проверяем работу. Выводятся все введённые аргументы (рис. 4.9)

```
[mekosolapov@fedora lab09]$ nasm -f elf lab9-2.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[mekosolapov@fedora lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[mekosolapov@fedora lab09]$
```

Рис. 4.9: Создание исполняемого файла, проверка его работы

10. Создаём файл lab9-3.asm и переносим предложенную программу из листинга №3, которая суммирует все аргументы(рис. 4.10):



```
lab9-3.asm
~/work/arch-pc/lab09

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx      ; Извлекаем из стека в `ecx` количество
8              ; аргументов (первое значение в стеке)
9 pop edx      ; Извлекаем из стека в `edx` имя программы
10             ; (второе значение в стеке)
11 sub ecx,1    ; Уменьшаем `ecx` на 1 (количество
12             ; аргументов без названия программы)
13 mov esi, 0   ; Используем `esi` для хранения
14             ; промежуточных сумм
15 next:
16 cmp ecx,0h   ; проверяем, есть ли еще аргументы
17 jz _end      ; если аргументов нет выходим из цикла
18             ; (переход на метку `_end`)
19 pop eax      ; иначе извлекаем следующий аргумент из стека
20 call atoi    ; преобразуем символ в число
21 add esi,eax   ; добавляем к промежуточной сумме
22             ; след. аргумент `esi=esi+eax`
23 loop next    ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit    ; завершение программы
```

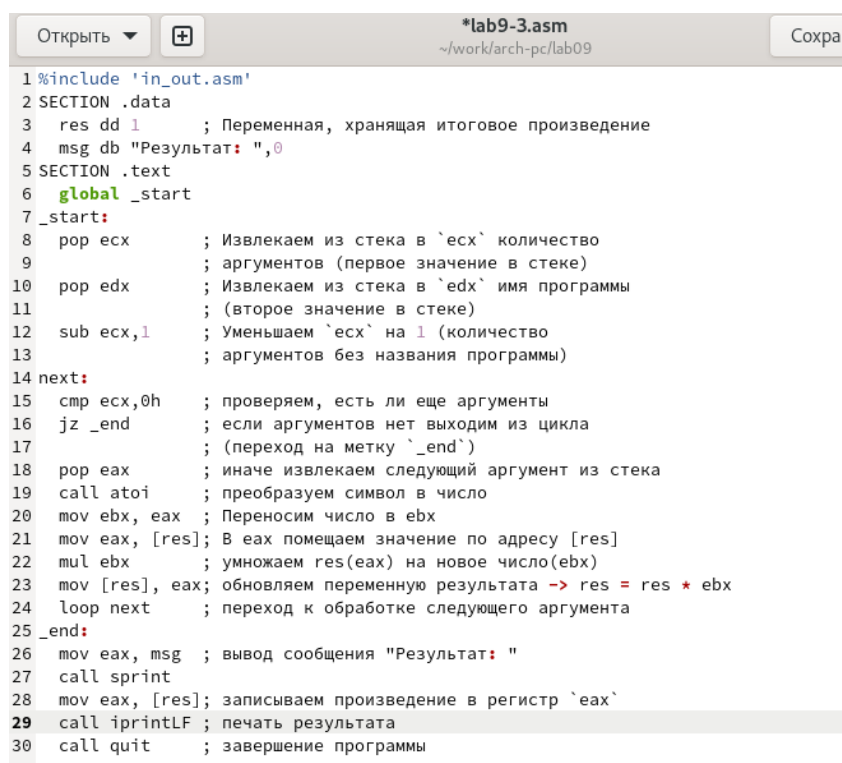
Рис. 4.10: Программа №3

11. Создаём исполняемый файл и проверяем его работу (рис. 4.11):

```
[mekosolapov@fedora lab09]$ nasm -f elf lab9-3.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[mekosolapov@fedora lab09]$ ./lab9-3 12 13 7 10 5
Результат: 47
[mekosolapov@fedora lab09]$
```

Рис. 4.11: Проверяем работу файла

12. Меняем программу так, чтобы она выводил произведение аргументов(рис. 4.12):



```
*lab9-3.asm
~/work/arch-pc/lab09
Сохранить

1 %include 'in_out.asm'
2 SECTION .data
3     res dd 1 ; Переменная, хранящая итоговое произведение
4     msg db "Результат: ",0
5 SECTION .text
6     global _start
7 _start:
8     pop ecx ; Извлекаем из стека в `ecx` количество
9             ; аргументов (первое значение в стеке)
10    pop edx ; Извлекаем из стека в `edx` имя программы
11           ; (второе значение в стеке)
12    sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
13           ; аргументов без названия программы)
14 next:
15    cmp ecx,0h ; проверяем, есть ли еще аргументы
16    jz _end ; если аргументов нет выходим из цикла
17           ; (переход на метку `_end`)
18    pop eax ; иначе извлекаем следующий аргумент из стека
19    call atoi ; преобразуем символ в число
20    mov ebx, eax ; Переносим число в ebx
21    mov eax, [res]; В eax помещаем значение по адресу [res]
22    mul ebx ; умножаем res(eax) на новое число(ebx)
23    mov [res], eax; обновляем переменную результата -> res = res * ebx
24    loop next ; переход к обработке следующего аргумента
25 _end:
26    mov eax, msg ; вывод сообщения "Результат: "
27    call sprint
28    mov eax, [res]; записываем произведение в регистр `eax`
29    call iprintLF ; печать результата
30    call quit ; завершение программы
```

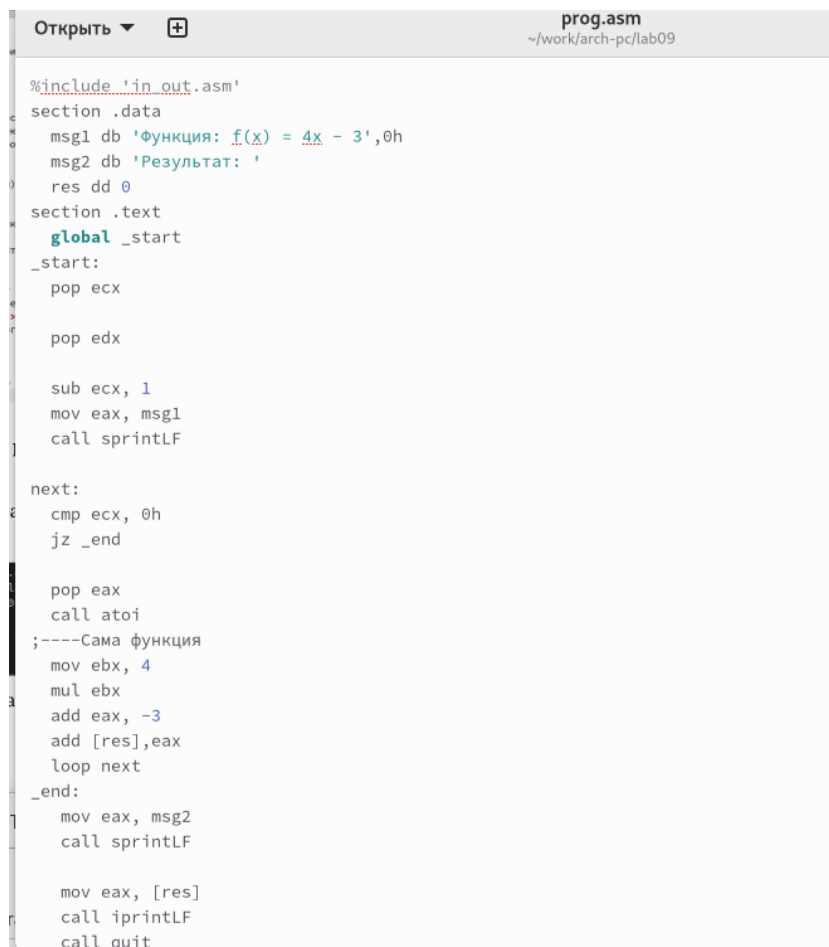
Рис. 4.12: Программа №4

13. Создаём исполняемый файл и проверяем работу программы (рис. 4.13):

```
[mekosolapov@fedora lab09]$ nasm -f elf lab9-3.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[mekosolapov@fedora lab09]$ ./lab9-3 12 13 7 10 5
Результат: 54600
[mekosolapov@fedora lab09]$ ./lab9-3 5 6 4 3 2 1
Результат: 720
[mekosolapov@fedora lab09]$
```

Рис. 4.13: Создание исполняемого файла, проверка его работы

Задания для самостоятельной работ 14. Программа, суммирующая $f(x)$ для множества x -ов (рис. 4.14):



```
Открыть ▾ + prog.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
section .data
    msg1 db 'Функция: f(x) = 4x - 3',0h
    msg2 db 'Результат: '
    res dd 0
section .text
    global _start
_start:
    pop ecx

    pop edx

    sub ecx, 1
    mov eax, msg1
    call sprintLF

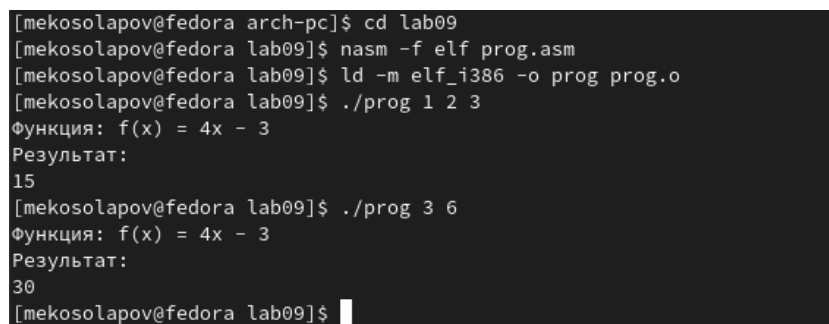
next:
    cmp ecx, 0h
    jz _end

    pop eax
    call atoi
;----Сама функция
    mov ebx, 4
    mul ebx
    add eax, -3
    add [res],eax
    loop next
_end:
    mov eax, msg2
    call sprintLF

    mov eax, [res]
    call iprintLF
    call quit
```

Рис. 4.14: Код программы

15. Создаём исполняемый файл и проверяем его работу (рис. 4.15):



```
[mekosolapov@fedora arch-pc]$ cd lab09
[mekosolapov@fedora lab09]$ nasm -f elf prog.asm
[mekosolapov@fedora lab09]$ ld -m elf_i386 -o prog prog.o
[mekosolapov@fedora lab09]$ ./prog 1 2 3
Функция: f(x) = 4x - 3
Результат:
15
[mekosolapov@fedora lab09]$ ./prog 3 6
Функция: f(x) = 4x - 3
Результат:
30
[mekosolapov@fedora lab09]$
```

Рис. 4.15: Создание исполняемого файла, проверка его работы

5 Выводы

В ходе данной лабораторной работы я научился работать с циклами и использовать их для вычисления различных функций. Мне понравилось, так как я уже могу написать какую-то простую программу для вычисления чего-либо. Например, факториала.

Список литературы