



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

**ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

**КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

**НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01**

**Отчет  
по лабораторной работе № 5**

**Название: Программирование с использованием разноязыковых модулей**

**Дисциплина: Машинно-зависимые языки и основы компиляции**

Студент

ИУ6-41Б

(Группа)

\_\_\_\_\_  
(Подпись, дата)

\_\_\_\_\_  
(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

\_\_\_\_\_  
(И.О. Фамилия)

Москва, 2025

## СОДЕРЖАНИЕ

Цель работы .....	3
Ход работы .....	5
Разработка программы № 1 .....	5
Тестирование программы №1 .....	7
Разработка программы № 2 .....	9
Тестирование программы №2 .....	14
Вывод.....	18
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	19

### **Цель работы**

Изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.

### **Задание**

1. Разработать ассемблерную вставку в программу на C++. Даны два целых числа типа `int`. Выполнить над ними операцию `TEST`. Проверить значение флага `SF` после выполнения операции. Результат вывести на экран.

2. Разработать программу из трех частей: основная функция на C++, подпрограмма на ассемблере, подпрограмма на C++, вызываемая из подпрограммы на ассемблере. Даны две строки по длине не превышающие 255 символов. Определить какая строка содержит больше гласных букв.

## Ход работы

### Разработка программы № 1

Разработаем схему алгоритма для выполнения задачи. Полученная схема алгоритма представлена на рисунке 1.

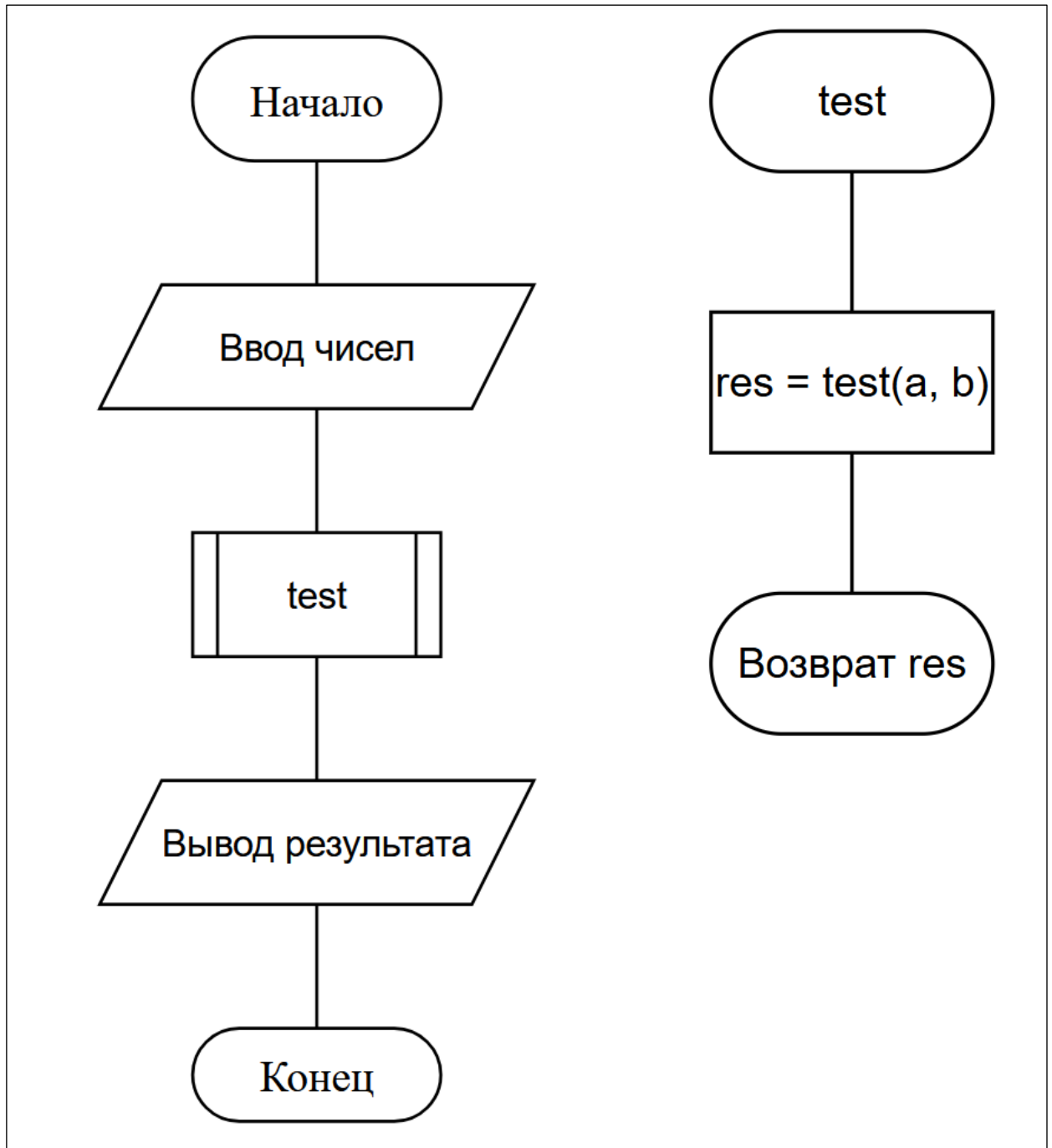


Рисунок 1 – Схема алгоритма программы

Разработаем программу в соответствии со схемой алгоритма выше. Код программы показан в листинге 1.

#### Листинг 1 – Код программы

```
#include <iostream>
using namespace std;
int main() {
    int a;
    int b;
    cin >> a;
    cin >> b;
    unsigned char sf = 0;
    asm(
        "test %[val1], %[val2]\n\t"
        "sets %[sf_flag]"
        : [sf_flag] "=r" (sf)
        : [val1] "r" (a), [val2] "r" (b)
        : "cc"
    );

    cout << "a = " << a << ", b = " << b << std::endl;
    cout << "Sign Flag (SF) after TEST: " <<
static_cast<int>(sf) << endl;

    if (sf)
        cout << "SF = 1 (результат отрицательный)"
<< endl;
    else
```

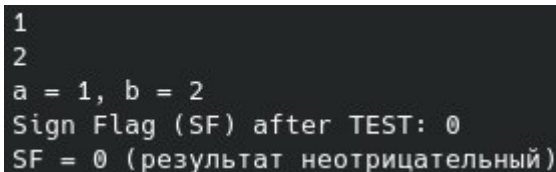
#### Продолжение листинга 1

```
        cout << "SF = 0 (результат неотрицательный)" <<  
endl;  
  
    return 0;  
}
```

### Тестирование программы №1

Выполним тестирование:

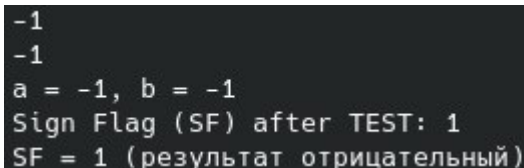
1) Входные данные: 1 и 2. Ожидаемый результат: 0. Представленный результат представлен на рисунке 2.



```
1  
2  
a = 1, b = 2  
Sign Flag (SF) after TEST: 0  
SF = 0 (результат неотрицательный)
```

Рисунок 2 – Результат тестирования 1

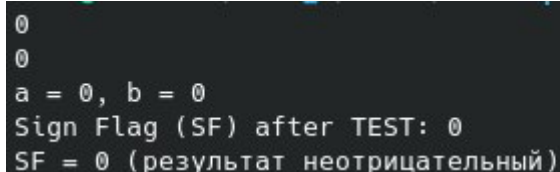
2) Входные данные: -1 и -1. Ожидаемый результат: 1. Представленный результат представлен на рисунке 3.



```
-1  
-1  
a = -1, b = -1  
Sign Flag (SF) after TEST: 1  
SF = 1 (результат отрицательный)
```

Рисунок 3 – Результат тестирования 2

3) Входные данные: 0 и 0. Ожидаемый результат: 0. Представленный результат представлен на рисунке 4.



```
0  
0  
a = 0, b = 0  
Sign Flag (SF) after TEST: 0  
SF = 0 (результат неотрицательный)
```

Рисунок 4 – Результат тестирования 3

4) Входные данные: 1 и -2. Ожидаемый результат: 0. Представленный результат представлен на рисунке 5.

```
1
-2
a = 1, b = -2
Sign Flag (SF) after TEST: 0
SF = 0 (результат неотрицательный)
```

Рисунок 5 – Результат тестирования 4

5) Входные данные: -3 и -3. Ожидаемый результат: 1. Представленный результат представлен на рисунке 6.

```
-3
-3
a = -3, b = -3
Sign Flag (SF) after TEST: 1
SF = 1 (результат отрицательный)
```

Рисунок 6 – Результат тестирования 5

Результаты тестирования представлены в таблице 1.

Таблица 1 – Тесты программы №1

Данные	Ожидаемое значение	Полученное значение
1, 2	0	0
-1, -1	1	1
0, 0	0	0
1, -2	0	0
-3, -3	1	1



## Разработка программы № 2

Разработаем схему алгоритма для выполнения задачи. Полученная схема алгоритма представлена на рис. 7

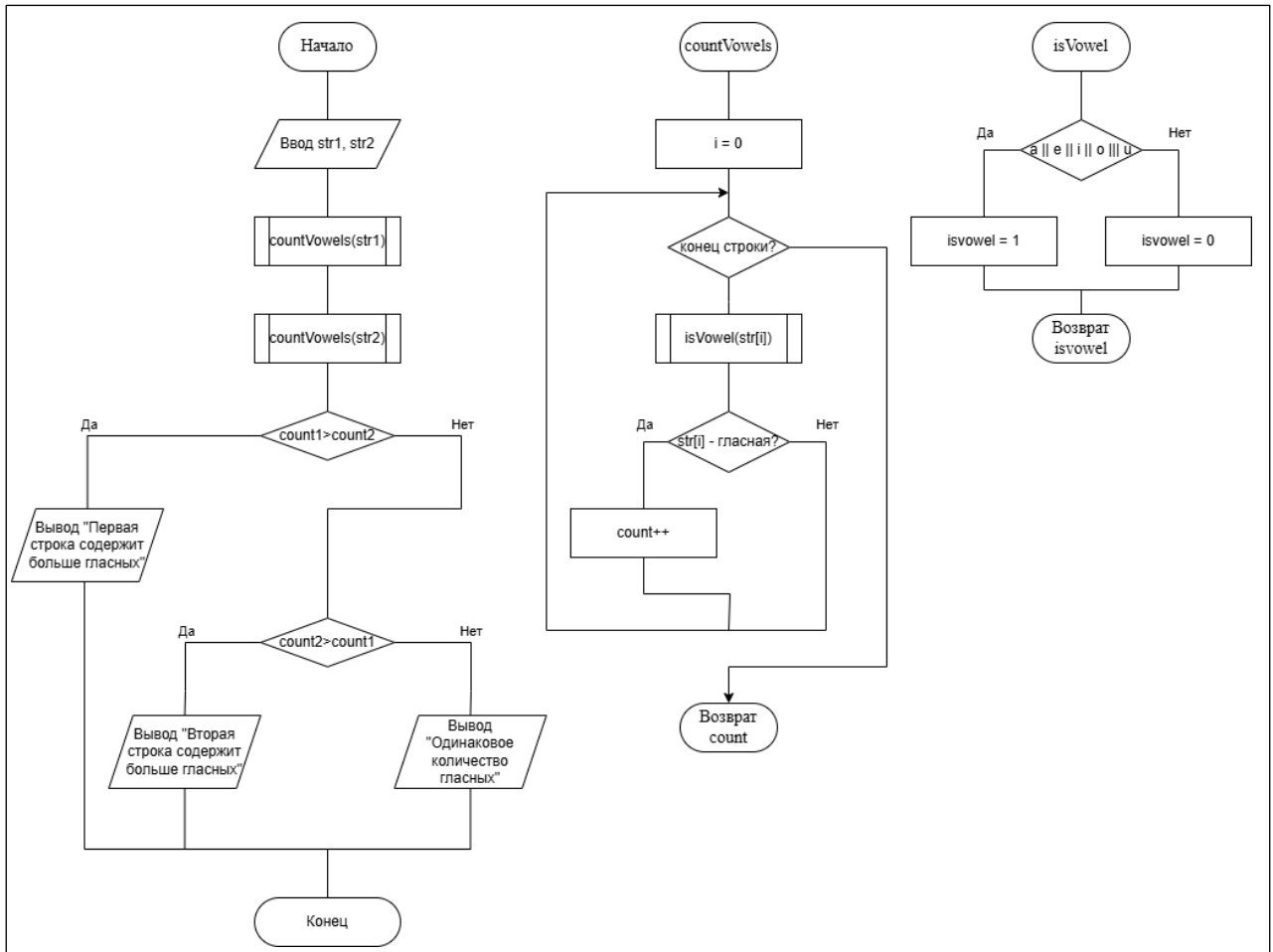


Рисунок 7 - Схема алгоритма программы ч.2

Разработаем программу в соответствии со схемой алгоритма выше. Код программы main показан в листинге 2, код подпрограммы на языке ассемблера показан в листинге 3.

### Листинг 2 - Код программы “main.cpp”

```
#include <iostream>

#include <cstring>

using namespace std;
```

## Продолжение листинга 2

```
// Объявление внешней ассемблерной функции
extern "C" int countVowels(const char* str, int
(*isVowelFunc)(int));

// Функция проверки на гласную
extern "C" int isVowel(int ch) {
    ch |= 0x20; // Перевод в нижний регистр
    return ch == 'a' || ch == 'e' || ch == 'i' || ch ==
'o' || ch == 'u';
}

int main() {
    const int MAX_LEN = 256;
    char str1[MAX_LEN], str2[MAX_LEN];

    cout << "Enter first string (max 255 chars): ";
    cin.getline(str1, MAX_LEN);

    cout << "Enter second string (max 255 chars): ";
    cin.getline(str2, MAX_LEN);
```

## Продолжение листинга 2

```
// Вызываем ассемблерную функцию, передаём
указатель на isVowel

int count1 = countVowels(str1, isVowel);
int count2 = countVowels(str2, isVowel);

// Сравниваем количество гласных
if (count1 > count2) {
    cout << "First string has more vowels (" <<
count1 << " vs " << count2 << ")" << endl;
} else if (count2 > count1) {
    cout << "Second string has more vowels (" <<
count2 << " vs " << count1 << ")" << endl;
} else {
    cout << "Both strings have the same number of
vowels (" << count1 << ")" << endl;
}

return 0;
}
```

Ниже представлено схематическое изображение содержимого стека в момент передачи управления. Полученная схема представлена на рисунке 10.

### Листинг 3 – Код подпрограммы на языке ассемблера

```
section .text
global countVowels

countVowels:
    push rbp
    mov rbp, rsp
    push rbx
    push r12
    push r13

    ; rdi = const char* str
    ; rsi = int (*isVowelFunc)(int)

    mov r12, rdi    ; указатель на строку
    mov r13, rsi    ; указатель на isVowel функцию
    xor ebx, ebx    ; счётчик гласных

.check_char:
    movzx eax, byte [r12] ; al = текущий символ
    test al, al          ; конец строки?
    jz .end

    mov edi, eax          ; передаём символ как int
    call r13              ; вызываем isVowel(int ch)
    test eax, eax         ; если вернуло не ноль –
гласная
    jz .next_char
    inc ebx
```

### Продолжение листинга 3

```
.next_char:
    inc r12
    jmp .check_char

.end:
    mov eax, ebx           ; возвращаемое значение
    pop r13
    pop r12
    pop rbx
    mov rsp, rbp
    pop rbp
    ret
```

RDI	str
RSI	isVowel

Рисунок 2 – Содержание регистров 64-х разрядной программы в момент передачи управления подпрограмме

Команды для сборки программы представлены на листинге 4.

Листинг 4 – команды для сборки программы.

```
g++ -c main.cpp -o main.o

nasm -f elf64 vowels.asm -o vowels.o

g++ main.o vowels.o -o lab5
```

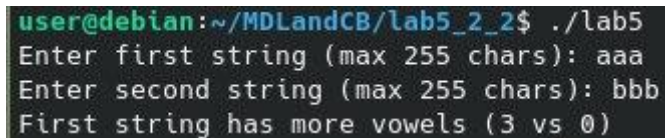
## Тестирование программы №2

Выполним тестирование:

1) Входные данные:

- Первая строка: aaa;
- Вторая строка: bbb.

Представленный результат представлен на рисунке 9.



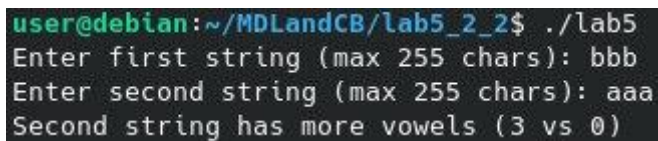
```
user@debian:~/MDLandCB/lab5_2_2$ ./lab5
Enter first string (max 255 chars): aaa
Enter second string (max 255 chars): bbb
First string has more vowels (3 vs 0)
```

Рисунок 9 – Результат тестирования 1

2) Входные данные:

- Первая строка: bbb;
- Вторая строка: aaa.

Представленный результат представлен на рисунке 10.



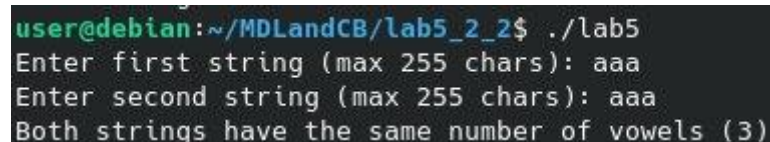
```
user@debian:~/MDLandCB/lab5_2_2$ ./lab5
Enter first string (max 255 chars): bbb
Enter second string (max 255 chars): aaa
Second string has more vowels (3 vs 0)
```

Рисунок 3 – Результат тестирования 2

3) Входные данные:

- Первая строка: aaa;
- Вторая строка: aaa.

Представленный результат представлен на рисунке 11.



```
user@debian:~/MDLandCB/lab5_2_2$ ./lab5
Enter first string (max 255 chars): aaa
Enter second string (max 255 chars): aaa
Both strings have the same number of vowels (3)
```

Рисунок 4 – Результат тестирования 3

4) Входные данные:

- Первая строка: bbb;
- Вторая строка: bbb.

Представленный результат представлен на рисунке 11.

```
user@debian:~/MDLandCB/lab5_2_2$ ./lab5
Enter first string (max 255 chars): bbb
Enter second string (max 255 chars): bbb
Both strings do not contain vowels
```

Рисунок 11 – Результат тестирования 4

Результаты тестирования программы №2 представлены в таблице 2.

Таблица 2 – Тесты программы №2

Данные	Ожидаемое значение	Полученное значение
aaa, bbb	First string has more vowels (3 vs 0)	First string has more vowels (3 vs 0)
bbb, aaa	Second string has more vowels (3 vs 0)	Second string has more vowels (3 vs 0)
aaa, aaa	Both strings have the same number of vowels (3)	Both strings have the same number of vowels (3)
bbb, bbb	Both lines do not contain vowels.	Both lines do not contain vowels.

## **Контрольные вопросы**

### **1. Что такое «конвенции о связи»? В чем заключается конвенция register?**

Конвенции о связи — это набор правил, определяющих:

- как передаются аргументы в функции (через регистры или стек)
- какие регистры сохраняются вызываемой функцией
- как возвращаются результаты

Конвенция Register — это частный случай, при котором:

Аргументы передаются через регистры (например, в x64: RDI, RSI, RDX, RCX).

### **2. Что такое «пролог» и «эпилог»? Где располагается область локальных данных?**

Пролог — код в начале функции, который:

- Сохраняет предыдущий RBP (push rbp).
- Устанавливает новый RBP (mov rbp, rsp).

Эпилог — код в конце функции, который:

- Восстанавливает RSP и RBP.
- Возвращает управление (ret).

### **3. Как связана структура данных стека в момент передачи управления и текст программы и подпрограмм?**

Структура стека в момент передачи управления отражает логику выполнения программы и взаимодействие между вызывающей и вызываемой подпрограммами. При вызове подпрограммы в стек помещается адрес возврата, что позволяет после завершения работы подпрограммы продолжить выполнение основной программы с нужного места.

### **4. С какой целью применяют разноязыкорвые модули в одном проекте?**

Использование разных языков в одном проекте необходимо для:



- Оптимизации производительности;
- Интеграции готовых решений.

Использование библиотек, доступных только для определённого языка.

## **Вывод**

В ходе работы изучены принципы взаимодействия C++ и ассемблера через вызовы функций с соблюдением соглашений о вызовах. Практически освоена передача параметров. Полученные навыки позволяют эффективно интегрировать ассемблерные оптимизации в C++

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Иванова Г.С., Ничушкина Т.Н., лабораторный практикум по программированию на ассемблере в операционной системе LINUX. М.: МГТУ имени Н.Э. Баумана, 2022. 77 с.