

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

**Программа для подсчёта количества букв и цифр в
ASCII - строке**

Вариант 25

Пояснительная записка

Выполнила: Резунник Людмила
студент гр. БПИ198

Москва
2020

Содержание

1. Текст задания	2
2. Постановка задачи.....	2
3. Описание алгоритма.....	2
3.1. Обработка пользовательского ввода	2
3.2. Подсчёт букв и цифр в строке.....	2
3.3. Вывод результата и завершение работы	4
4. Описание входных и выходных данных	4
5. Тестирование программы	4
ПРИЛОЖЕНИЕ 1. Список использованных источников	6
ПРИЛОЖЕНИЕ 2. Код программы.....	7

1. Текст задания

Вариант 25

Разработать программу, которая вычисляет количество цифр и букв в заданной ASCII-строке.

2. Постановка задачи.

Разработать программу, которая подсчитывает количество цифр и латинских букв в строке, введённой пользователем в консоли. Программа должна запрашивать пользовательский ввод, выводить строку введённую пользователем для проверки и возвращать результат работы.

3. Описание алгоритма

Работу алгоритма можно разделить на 3 этапа. Первый этап – считывание строки, введённой пользователем. Второй этап – подсчёт отдельно цифр и букв в цикле. Третий этап – вывод результата в консоль и завершение работы программы. Далее подробно расписывается процесс работы алгоритма на каждом из этих этапов.

3.1. Обработка пользовательского ввода

Программа выводит на экран консоли сообщение о запросе пользовательского ввода. Далее происходит считывание строки, введённой пользователем, в переменную `inputString` с помощью функции `scanf`. Строка считывается до первого пробела, пробел в этом случае является обозначением конца строки. После этого с помощью `printf` эта же строка выводится на экран (для проверки).

3.2. Подсчёт букв и цифр в строке.

Как написано выше, обработка строки осуществляется в цикле. Запуск цикла осуществляется с помещения адреса введённой строки в регистр `esi`. Так будет получен доступ к индексу элемента строки.

Далее, начинается работа самого цикла. Сначала, значение элемента строки с нужным индексом, заносится в регистр `dl`. После этого происходит проверка на

конец строки, в случае если достигнут конец строки, программа переходит на метку end.

Далее сравниваем значение элемента в регистре с кодом символа '0', если код элемента строки меньше, это значит, что проверяемый символ не является ни буквой, ни цифрой (так как ASCII – код цифр меньше кодов букв), программа переходит на метку notLetter, где происходит увеличение индекса элемента строки и продолжается работа цикла.

После этого происходит сравнение кодов символа '9' и элемента строки. В случае если он больше, происходит переход на метку .notdigit, которая осуществляет проверку на то, что это заглавная буква. Далее, программа в зависимости от результата сравнения элементов может перейти на метку .notuppercase, где уже происходит проверка, на то, что элемент строки – строчная буква.

Схема работы цикла представлена ниже (рис. 1)

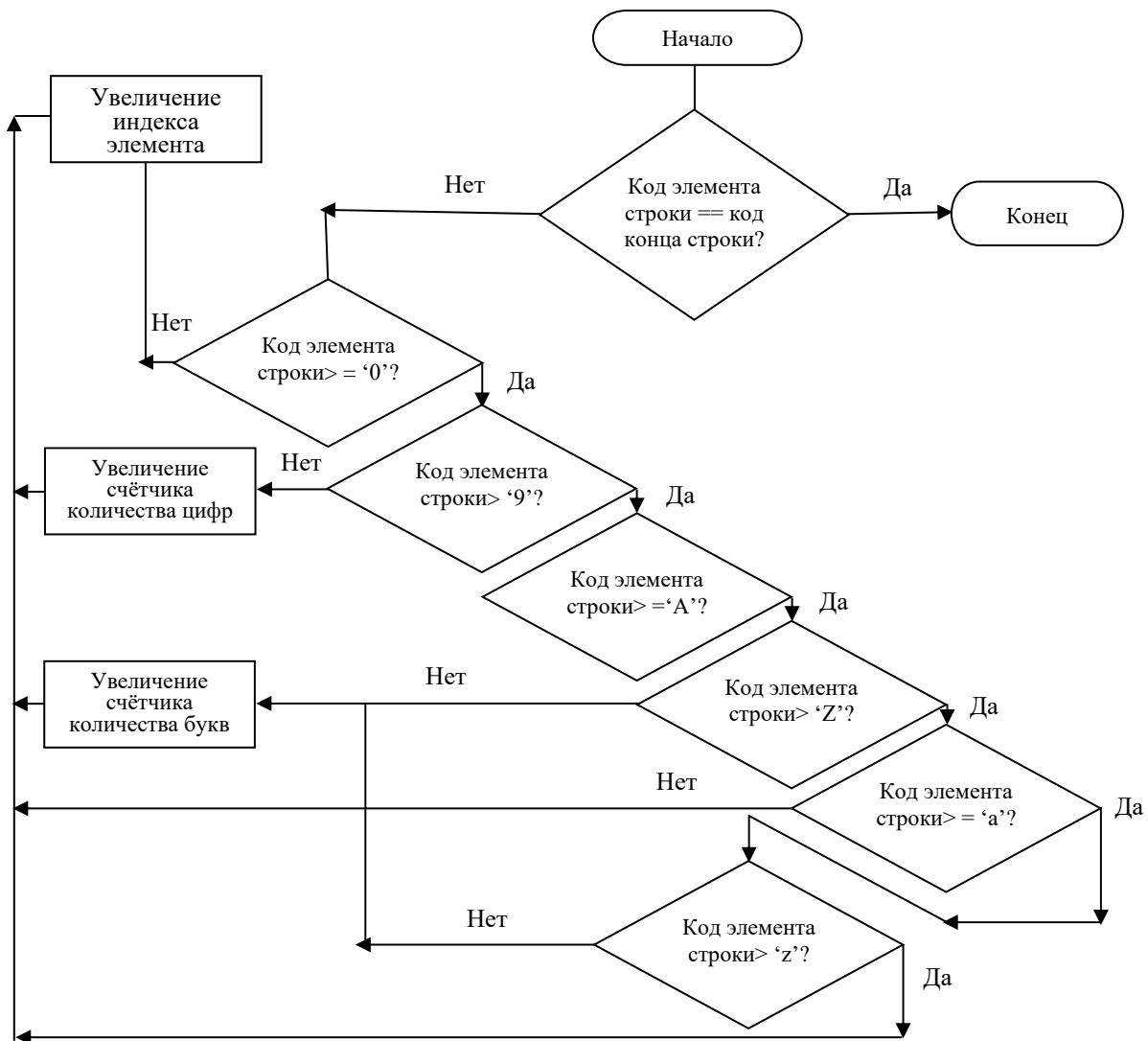


Рисунок 1 – Схема работы цикла

3.3. Вывод результата и завершение работы

Для вывода результата осуществляется переход на метку `printRes`. В стек заносятся значения регистров `ebx` и `ecx`, в которых хранятся значения счётчиков количества цифр и букв. Результат выводится на консоль с помощью функции `printf`. Далее осуществляется переход на метку `finish`, где осуществляется завершение работы программы в случае, если пользователь нажмёт на какую-либо клавишу.

4. Описание входных и выходных данных

Входные данные должны представлять собой строку, состоящую из любой последовательности латинских букв и цифр. Пробелы служат символом окончания строки, то есть всё, что следует в строке за пробелом отбрасывается. Перенос строки также является пробельным символом.

Выходные данные представляют собой строку изначально введённую пользователем, а так же результат подсчёта цифр и букв в строке в следующем формате: «Numbers: X Letters: X».

5. Тестирование программы

Далее будут представлены скриншоты работы программы на различных тестах.

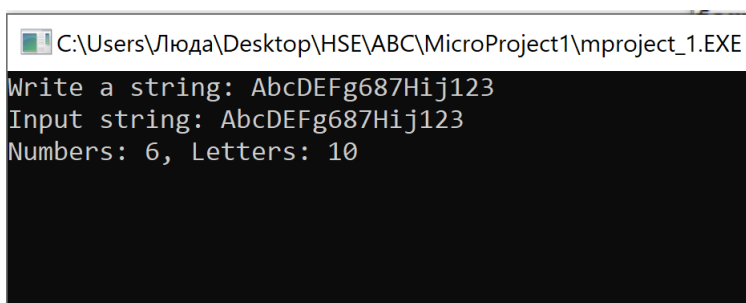
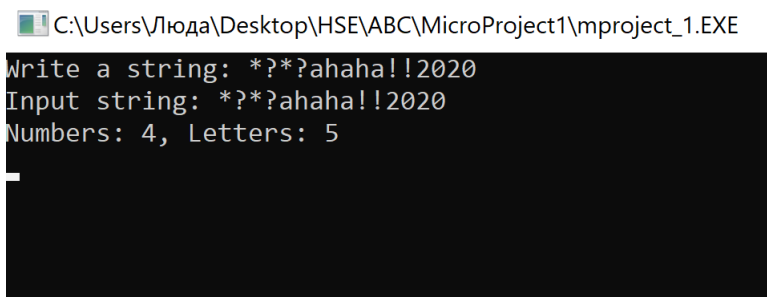


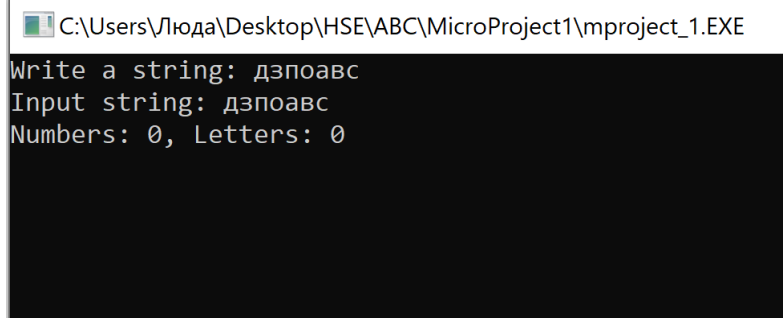
Рисунок 2 – Строка длины 16 с числами и латинскими буквами (заглавными и строчными)



```
C:\Users\Людa\Desktop\HSE\ABC\MicroProject1\mproject_1.EXE
```

```
Write a string: *?*?ahaha!!2020
Input string: *?*?ahaha!!2020
Numbers: 4, Letters: 5
```

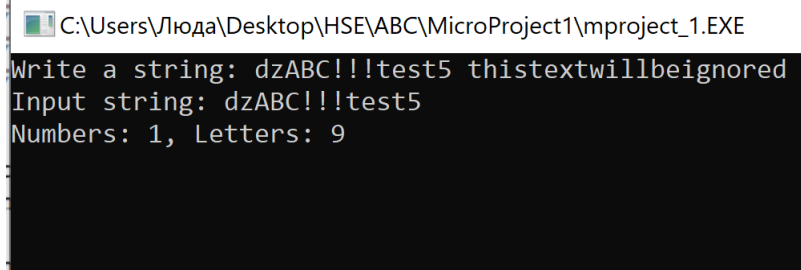
Рисунок 3 – Вводим также другие символы, помимо чисел и букв



```
C:\Users\Людa\Desktop\HSE\ABC\MicroProject1\mproject_1.EXE
```

```
Write a string: дэпоавс
Input string: дэпоавс
Numbers: 0, Letters: 0
```

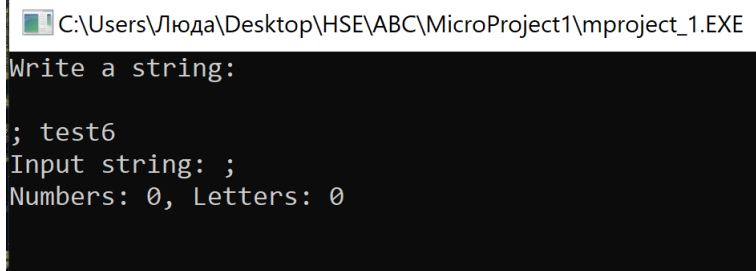
Рисунок 4 – Русские буквы не подсчитываются



```
C:\Users\Людa\Desktop\HSE\ABC\MicroProject1\mproject_1.EXE
```

```
Write a string: dzABC!!!test5 this text will be ignored
Input string: dzABC!!!test5
Numbers: 1, Letters: 9
```

Рисунок 5 – Весь текст после пробела игнорируется



```
C:\Users\Людa\Desktop\HSE\ABC\MicroProject1\mproject_1.EXE
```

```
Write a string:
; test6
Input string: ;
Numbers: 0, Letters: 0
```

Рисунок 6 – Пример с переносом строки

ПРИЛОЖЕНИЕ 1**Список использованных источников**

1. Программирование на языке ассемблера. [Электронный ресурс] // URL: <http://natalia.appmat.ru/c&c++/assembler.html> (дата обращения: 30.11.2020)
2. Примеры на ассемблере. [Электронный ресурс] // URL: <http://av-assembler.ru/source/> (дата обращения: 30.11.2020)

Код программы

```

; ВАРИАНТ 25
; Разработать программу, которая вычисляет количество цифр и букв в заданной
ASCII-строке
;
format PE console
entry start

include 'win32a.inc'

;-----
section '.data' readable writable

    letterCount    dd 0
    numberCount    dd 0
    inputString    rb 4096 ; Переменная для хранения строки, введённой
пользователем.

    scanString     db "%s", 0
    startString    db 'Write a string: ', 0
    strShowInp     db 'Input string: %s', 10, 0
    resultString   db 'Numbers: %d, Letters: %d', 10, 0

;-----

section '.code' readable executable

start:

    call input ; Получение строки от пользователя.
    stdcall processString, inputString ; Подсчёт букв и чисел в строке.
    call printRes ; Вывод результата и завершение работы программы.

finish:

    call[getch] ; Ожидание пока пользователь не введёт символ для завершения
программы.
    push 0
    call [ExitProcess]

```


;-----

; Здесь происходит считывание пользовательского ввода.

input:

```
    push startString
    call [printf] ; Просим пользователя ввести строку.
    add esp, 4 ; Очищаем стек.
```

```
    push inputString
    push scanString
    call[scanf] ; Считываем строку с консоли.
    add esp, 8 ; Очищаем стек.
```

```
    push inputString ; Помещаем переменную, содержащую строку, в стек.
    push strShowInp ; Помещаем в стек строку для вывода.
    call[printf] ; Вывод строки, введённой пользователем.
    add esp, 8
```

```
    ret ; Возвращаемся в start.
```

; В этом методе происходит подсчёт цифр и букв в строке.

processString:

```
    mov esi, [esp+4] ; Помещаем адрес строки в esi.
    xor ebx, ebx ; Зануляем регистр ebx.
    xor ecx, ecx ; Зануляем регистр ecx.
```

; Цикл производит проверку элементов строки и подсчёт цифр и букв в ней.

.loop:

```
    mov dl, [esi] ; Заносим значение элемента строки в dl.
    cmp dl, 0
    jz .end ; Если элемент равен 0, значит дошли до конца строки.
```

```
    cmp dl, '0' ; Сравниваем код символа 0 с элементом строки.
    jb .notLetter ; Если код элемента меньше, это не цифра. Проверяем дальше.
    cmp dl, '9' ; Сравниваем код символа 9 с элементом строки.
    ja .notdigit ; Если код элемента больше, это не цифра. Проверяем дальше.
    inc ebx ; Если прошло все предыдущие условие, значит это цифра.
    jmp .notLetter ; Не буква.
```

; Переход сюда осуществляется, если не была пройдена проверка на то, что элемент

строки - это цифра.

.notdigit:

cmp dl, 'A' ; Сравниваем код символа A с элементом строки.

jb .notLetter ; Если код элемента меньше, это не заглавная буква. Завершаем проверку этого элемента.

cmp dl, 'Z' ; Сравниваем код символа Z с элементом строки.

ja .notuppercase ; Если код элемента больше, это не заглавная буква. Завершаем проверку этого элемента.

inc ecx ; Если пройдены все проверки - это заглавная буква. Увеличиваем счётчик.

; Переход сюда осуществляется, если не была пройдена проверка на заглавную букву.

.notuppercase:

cmp dl, 'a' ; Сравниваем код символа a с элементом строки.

jb .notLetter ; Если код символа строки меньше - это не буква. Завершаем проверку данного символа.

cmp dl, 'z' ; Сравниваем код символа z с элементом строки.

ja .notLetter ; Если код символа строки больше - это не буква. Завершаем проверку данного символа.

inc ecx ; Если пройдены все проверки - это буква. Увеличиваем счётчик.

; Этот код выполняется, если символ строки не буква и не цифра.

.notLetter:

inc esi ; Увеличиваем индекс элемента строки.

jmp .loop ; Продолжаем работу цикла.

; Завершение проверки строки.

.end:

ret ; Переходим к выводу результата.

; Здесь осуществляется вывод результата.

printRes:

push ecx ; Заносим в стек счётчик букв.

push ebx ; Заносим в стек счётчик цифр.

push resultString ; Заносим в стек строку для вывода.

call [printf] ; Вывод результата.

jmp finish ; Завершаем работу программы.

;-----third act - including HeapApi-----

section '.idata' import data readable

```
library kernel, 'kernel32.dll',\  
      msvcrt, 'msvcrt.dll',\  
      user32, 'USER32.DLL'
```

```
include 'api\user32.inc'  
include 'api\kernel32.inc'  
import kernel,\  
      ExitProcess, 'ExitProcess',\  
      HeapCreate, 'HeapCreate',\  
      HeapAlloc, 'HeapAlloc'  
include 'api\kernel32.inc'  
import msvcrt,\  
      printf, 'printf',\  
      scanf, 'scanf',\  
      getch, '_getch'
```