

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

**ПРОГРАММА НА FASM ДЛЯ ПОИСКА В ЗАДАННОЙ ASCII-СТРОКЕ ПЕРВУЮ
СПРАВА НАЛЕВО ПОСЛЕДОВАТЕЛЬНОСТЬ 5 СИМВОЛОВ, ГДЕ КАЖДЫЙ БОЛЬШЕ
ПРЕДЫДУЩЕГО**

Пояснительная записка

Исполнитель:

студент группы БПИ191 (2 подгруппа),

Кирюхин Андрей Алексеевич

Москва 2020

Москва 2020**СОДЕРЖАНИЕ**

1. ВВЕДЕНИЕ	2
1.1. Текст задания	2
1.2. Алгоритм программы	2
1.3. Ограничение входных данных.....	2
2. ТЕСТИРОВАНИЕ ПРОГРАММЫ	3
2.1. Строка, полностью состоящая из элементов, где выполняется условие подстроки	3
2.2. Строка, содержащая только одну удовлетворяющую условию подстроку.....	3
2.3. Строка, не имеющая необходимой подстроки	4
ПРИЛОЖЕНИЕ 1	5
ТЕКСТ ПРОГРАММЫ	6

1. ВВЕДЕНИЕ

1.1. Текст задания

Разработать программу, находящую в заданной ASCII-строке первую справа налево последовательность N символов, каждый элемент которой определяется по условию "больше предшествующего" (N=5)

1.2. Алгоритм программы

Полный код программы представлен в ПРИЛОЖЕНИИ 1. В этом разделе содержится краткая информация об алгоритме программы. Весь код можно разделить на следующие этапы:

1. Считывание начальной строки
2. Вычисление и проверка длины строки

При длине строки меньше 5, то пользователю выводится сообщение о некорректной длине и запрашивается повторить ввод.

3. Запуск цикла итерации по строке

Предварительно заводится специальная переменная (**pointer**), которая хранит адрес текущего элемента, до начала цикла она указывает на последний элемент строки.

В самом цикле происходит последовательное попарное сравнение элементов строки с конца, если нашлось подряд 5 идущих элементов, удовлетворяющих условию, то программа переходит к следующему этапу работы.

Если в цикле не находится подходящей подстроки, то программа переходит к выполнению 5 этапа.

4. Восстановление и вывод ответа

Программа идет с последнего корректного элемента из начальной строки, поочередно добавляя по одному элементу в новую строку. Также, в конце к строке дописывается байт, означающий конец строки.

5. Сообщение о невозможности найти подстроку и завершение программы

1.3. Ограничение входных данных

На вход программе подается строка, но длины меньше не больше 249 символов. Также, если пользователь введет строку длиной меньше 5, то программа попросит повторить ввод, так как невозможно в такой строке найти подстроку длиной 5.

2. ТЕСТИРОВАНИЕ ПРОГРАММЫ

2.1. Строка, полностью состоящая из элементов, где выполняется условие подстроки

Для данных тестов был взят русский и английский алфавит.

```
Enter a string longer than 5:
abcdefghijklmnopqrstuvwxyz
The required substring was found successfully:
vwxyz
Press Enter to end the program
```

```
Enter a string longer than 5:
абвгдеёжзийклмнопрстуфхцчшщъыьэюя
The required substring was found successfully:
ыьэюя
Press Enter to end the program
```

Как можно заметить из скриншотов, программа корректно работает вне зависимости от языка и выводит первую удовлетворяющую условию последовательность элементов.

Также, проверим программу на строку, состоящую из цифр от 0 до 9.

```
Enter a string longer than 5:
0123456789
The required substring was found successfully:
56789
Press Enter to end the program
```

На данном тесте программа также корректно работает.

2.2. Строка, содержащая только одну удовлетворяющую условию подстроку

Для начала возьмём удовлетворяющую подстроку “abcde” и окружим её символами “a”.

```
Enter a string longer than 5:
abcdeaaaaaaaaaaaaaaaaaaaaaaaaa
The required substring was found successfully:
abcde
Press Enter to end the program
```

```
Enter a string longer than 5:
aaaaaaaaaaaaabcdeaaaaaaaaaaaaa
The required substring was found successfully:
abcde
Press Enter to end the program
```

```
Enter a string longer than 5:
aaaaaaaaaaaaaaaaaaaaabcde
The required substring was found successfully:
abcde
Press Enter to end the program
```

Как можно заметить, программа корректно работает.

2.3. Строка, не имеющая необходимой подстроки

[illegible]

```
Enter a string longer than 5:  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
aaaaaaaaaa  
Could not find the required substring  
  
Press Enter to end the program
```

[illegible]

На данных тестах программа так же корректно обрабатывает строки.

3. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) StackOverflow [Электронный ресурс] // URL: <https://stackoverflow.com/> (Режим доступа: свободный).
- 2) FASM forum [Электронный ресурс] // URL: <https://board.flatassembler.net/> (Режим доступа: свободный).
- 3) FASM documentation [Электронный ресурс] // URL: <https://flatassembler.net/> (Режим доступа: свободный).

ТЕКСТ ПРОГРАММЫ

; Кирюхин
 Андрей
 Алексеевич,
 БПИ191

; Вариант - 18

; Разработать программу, находящую в
 ; заданной ASCII-строке первую справа
 ; налево последовательность 5 символов,
 ; каждый элемент которой определяется по
 ; условию "больше предшествующего"

format PE console

entry start

include 'win32a.inc'

;-----

section '.data' data readable writable

startStr db 'Enter a string longer than 5:', 10,13,0
 wrongSizeStr db 'Wrong length! Repeat with correct length of a string',
 10,13,0
 noAnswerStr db 'Could not find the required substring', 10,13,0
 foundAnswerStr db 'The required substring was found successfully:', 10,13,0
 pressEnterStr db 'Press Enter to end the program', 10,13,0
 formatS db '%s', 0 ; Текстовый формат считывания/вывода
 formatN db '%d', 0 ; Числовой формат считывания/вывода
 newLine db 10, 13, 0 ; Перевод каретки на новую строку
 pointer dd ? ; Указатель на адрес конкретного элемента в строке
 numCorrect dd ? ; Количество подряд идущих символов,
 удовлетворяющих условию
 strLen dd ? ; Длина начальной строки
 addrDiff dd ? ; Разница в адресах между старой и новой строкой
 text db 256 dup(0) ; Изначальный строка
 resultString db 256 dup(0) ; Полученная строка

;-----

section '.code' code readable executable

start:

; Считывание изначальной строки

cinvoke printf, startStr

cinvoke scanf, formatS, text

; 1) Вычисление и проверка длины строки

```
ccall strlen, text
mov [strLen], ecx
mov [pointer], ecx
cmp [pointer], 5
jl wrongLen
```

; Присваивание переменной pointer значения адреса конца строки

```
mov eax, text
mov ecx, text
add ecx, [pointer]
mov [pointer], ecx
dec [pointer]
```

; 2) Запуск процедуры итерации по строке

```
xor ecx, ecx
mov [numCorrect], ecx
call IterateString
```

; 3) Получение ответа

```
dec [pointer]
call GetAnswer
```

; Вывод найденной подстроки

```
cinvoke printf, foundAnswerStr
cinvoke printf, resultString
```

; Конец работы программы

```
cinvoke printf, newLine
cinvoke printf, newLine
cinvoke printf, pressEnterStr
call [getch]
push 0
call [ExitProcess]
```

; Оповещение о неправильной длине строки и повтор программы

wrongLen:

```
cinvoke printf, wrongSizeStr
jmp start
```

; Процедура поиска подстроки

IterateString:

```
mov eax, [numCorrect]
cmp eax, 5
je endStringLoop
```

; если нашли 5 подходящих подряд элементов -

заканчиваем цикл

```
xor ebx, ebx
inc ebx
```

mov [numCorrect], ebx ; присваиваем значение numCorrect = 1 (считая что текущий элемент подходит)

```
mov eax, [pointer]
```



```

    cmp eax, text
    jle noAnswer          ; если pointer указывает на начало строки, то значит
ответа не было найдено, переходим в noAnswer
iterate_loop:
    mov eax, [numCorrect]
    cmp eax, 5
    je endStringLoop      ; если нашли 5 подходящих подряд элементов - заканчиваем
цикл

    mov eax, [pointer]     ; сохраняем в eax pointer (текущий элемент)
    dec [pointer]
    mov ebx, [pointer]     ; сохраняем в ebx pointer (текущий элемент на предыдущей
позиции)

    mov edx, [pointer]
    cmp edx, text
    jl noAnswer           ; если pointer указывает на адрес до начала строки, то
значит ответа не было найдено, переходим в noAnswer

    mov dl, [eax]
    mov dh, [ebx]
    cmp dl, dh            ; сравниваем два элемента строки, текущего и стоящего
перед ним
    jle IterateString     ; если элементы не удовлетворяют условию (сравнение
элемента и предшествующего тоже идет слева-направо), то переходим обратно в
IterateString (сбрасывается numCorrect и обновляется счетчик цикла)

    inc [numCorrect]      ; если удовлетворяет условию, то увеличиваем количество
найденных удовлетворяющих символов
    jmp iterate_loop
endStringLoop:
    ret

; Процедура формирования ответа
GetAnswer:
    mov eax, resultString
    sub eax, [pointer]
    mov [addrDiff], eax
    dec [addrDiff]        ; считаем значение для разницы в адресах между нужным
элементом в text и началом строки resultString

    mov ecx, [numCorrect] ; numCorrect = 5, значит делаем 5 итераций для
формирования ответа
answer_loop:
    inc [pointer]         ; переходим к рассмотрению следующего элемента в
начальной строке
    mov ebx, [pointer]    ; ebx указывает на элемент в начальной строке text
    mov eax, [pointer]

```

```

    add eax, [addrDiff]    ; eax указывает на элемент в строке resultString
    mov dl, [ebx]          ; берем элемент из начальной строки
    mov [eax], dl          ; и записываем его в новую
    loop answer_loop
endAnswerLoop:
    mov eax, resultString
    add eax, 5              ; получаем адрес последнего элемента строки resultString
    mov byte [eax], 0       ; записываем в него байт конца строки
    ret

; Оповещение о невозможности найти подстроку и завершение программы
noAnswer:
    cinvoke printf, noAnswerStr
    cinvoke printf, newLine
    cinvoke printf, pressEnterStr
    call [getch]
    push 0
    call [ExitProcess]

; Функция получения длины строки
proc strlen, strInput
    mov ecx, -1
    mov al, 0              ; заносим в регистр al байт конца строки
    mov edi, [strInput]
    cld                    ; выбор направления обхода строки
    repne scasb            ; сравнивает каждый байт строки со значением в регистре al,
    ; результат записывается в ecx
    not ecx                ; так как идет уменьшение с каждой итерацией поиска, убираем знак
    ; минус из ecx
    dec ecx
    ret
endp

;-----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',\  
gets, 'gets'
```