```
format PE console
entry start

include 'win32a.inc'

section '.data' data readable writeable

    message    db  'Привет мир', 0
    s          db  'П', 0
    output     db  'Index = %d', 0
    wrong      db  'Substring isnt in string', 0
    i          dd  ?



section '.code' code readable executable

start:
    mov esi, message       ;Что искать
    mov edi, s             ;Где искать
    call pos               ;Вызываем функцию поиска
                           ;Она вернет в EAX номер символа, с которого начинается совпадение строки
    invoke ExitProcess,0   ;Это вызов корректного выхода из программы для Винды
    ret

; Тело функции поиска первого входящего
proc pos
    xor eax,eax   ;Очищаем регистры, с которыми будем работать при поиске - для возврата позиции
    xor ecx,ecx   ;Для просчета кол-ва пройденных символов в подстроке - длина подстроки по факту
    xor ebx,ebx   ;А это временный, куда будет поступать сравниваемый символ

    mov edx,edi   ;EDX нам понадобится для перескока в начало искомого, при несовпадении символов в поиске
```

```
    cmp byte [esi],0       ;Если конец исходной строки
    je exit                ;Выходим
    cmp byte [edi],0       ;Если конец искомой строки
    je exit                ;Тоже выходим

;Иначе начнем цикл поиска
for:
    inc eax                ;Увеличим счетчик позиции, на которой сравниваем символ
    mov bl,byte [esi]      ;Получим очередной сравниваемый символ
    cmp bl,[edi]           ;И сравним его с символом в искомой строке
    jne no                 ;Если символы не равны перескакиваем на следующий в исходной строке и заканчиваем сравнение блока
    inc ecx                ;Иначе увеличим счетчик длины искомого
    inc edi                ;Перескочим на следующий символ в искомой строке
    cmp byte [edi],0       ;И узнаем не конец ли искомого
    je exitproc            ;Если конец - строка найдена. Выходим.
    jmp nextiter           ;Иначе пойдем на следующую итерацию цикла поиска
no:
    sub esi,ecx            ;Поскольку используется один цикл для прохода
    ;После поиска подстроки нам приходится возвращаться на тот символ, с которого мы начали.
    ;Достигается это минусованием прошедших символов в искомом и текущей позиции в источнике
    xor ecx,ecx            ;Потом не забудим сбросить в ноль кол-во совпавших символов, раз они не найдены
    mov edi,edx            ;И переместиться в начало искомой строки для поиска с следующего символа
    cmp byte [esi],0       ;Если конец исходной строки, значит подстрока не входит
    je exit                ;Выходим
nextiter:
    inc esi                ;При каждой итерации будем перескакивать с символа на символ в исходной строке
    jmp for                ;И прыгать на следующий шаг цикла
exitproc:
    sub eax,ecx            ;Поскольку у нас EAX будет указывать на позицию последнего символа искомого,
    ;Придется отнять от него длину искомого, чтоб получить позицию начинающуюся с нуля
```

```
        mov [i], eax          ;Кладем полученный индекс в переменную
        push [i]              ;И выводим ответ
        push output
        call [printf]
        call [getch]
        ret
exit:
        push wrong            ;Кладем сообщение с ошибкой
        call [printf]
        call [getch]
        ret
endp;

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'
```