

# Основы языка ассемблер

# Использование массивов

The background features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to dark navy blue. The shapes are primarily triangles and polygons, creating a modern, layered effect. The text is centered horizontally and positioned in the lower half of the frame.

# Массивы

**Массивом** называется последовательный набор однотипных данных, именованный одним идентификатором.

Примеры инициализации

M1 DD 0,1,2,3,4,5,6,7,8,9

M2 DD 0,1,2,3

Для инициализации всех элементов массива одинаковыми значениями используется оператор DUP:

**Идентификатор Тип Размер DUP (Значение)**

**Идентификатор** - имя массива;

**Тип** - определяет количество байт, занимаемое одним элементом;

**Размер** - константа, характеризующая количество элементов в массиве

**Значение** - начальное значение элементов.

a DD 20 DUP (0) - описывает массив a из 20 элементов, начальные значения которых равны 0.

Если необходимо выделить память, но не инициализировать ее, в качестве поля **Значение** используется знак ?. Например,

b DD 20 DUP(?)

# Задача

Заданы массивы  $A[N]$  и  $B[N]$  из элементов типа Byte (8-разрядные целые без знака). Составить программу, формирующую массив  $C[N]$  из произведения элементов массивов  $A$  и  $B$ :  $C[i]=A[i]*B[i]$ . Размерность элементов массива  $C[N]$  должна обеспечивать корректное умножение (если результат не помещается в 8 разрядов).

# Программа

```
org 100h
.model tiny
.data
N dw 10 ; Кол-во элементов в массиве.
A db 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
B db 3, 4, 15, 6, 1, 0, 0, 2, 2, 18
C dw 10 dup(0)
.code
Start:
mov si, 0 ; индекс массивов А и В.
mov di, 0 ; индекс массива С.
M1: mov ah, 0
mov al, A[si]
mul B[si] ; Умножение AX = AL*B[si].
mov C[di], ax ; Запись результата.
inc si ; Завершение
add di, 2 ; тела цикла.
cmpsi, N
jb M1
end Start
ret
```

# Просмотр результата

The screenshot shows the main emulator window titled "emulator: summa.com\_". The "view" menu is open, listing various debugging options. The "variables" option is highlighted. On the left, the registers window shows the following values:

Register	H	L
AX	00	00
BX	00	00
CX	00	3A
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The status bar at the bottom shows "0711A: 00 000 NULL".

The "variables" window shows the following configuration and data:

- size: word
- elements: 1
- show as: hex

N	000Ah
A	0FFFFh, 0FFFBh, 0003h, 0005h, 001Ch, 0FFF5h, 0007h, 0008h, 0020h, 0FFA6h
SUM	0000h
ABS	0000h

The "variables" window shows the updated state of variables:

- size: word
- elements: 1
- show as: hex

N	000Ah
A	0FFFFh, 0FFFBh, 0003h, 0005h, 001Ch, 0FFF5h, 0007h, 0008h, 0020h, 0FFA6h
SUM	0FFE8h
ABS	00BEh

# Задача

Задан массив  $A[N]$  из элементов типа целое 16-разрядное со знаком.

Составить программу суммирования элементов массива и абсолютных значений элементов массива.

Обычное суммирование провести в переменной  $Sum$ , суммирование по модулю - в переменной  $Abs$ .

# Программа

```
org 100h
.model tiny
.data
N dw 10 ; Количество элементов в массиве A.
A dw -1, -5, 3, 5, 28, -11, 7, 8, 32, -90
Sum dw 0 ; Результат обычного суммирования.
Abs dw 0 ; Результат суммирования по модулю.
.code
Start:
mov si, 0
mov cx, N
M1: mov ax, A[si] ; Обычное суммирование.
add Sum, ax ; Проверка перед суммированием по модулю.
or ax, ax ; Если число положит. - сразу прибавить к Abs.
jns M2 ; Если число отрицательное - взять по модулю.
neg ax
M2: add Abs, ax
add si, 2 ; индекс на следующий элемент.
loop M1 ; Повторять тело цикла N раз.
end Start
ret
```



# Задание

Задан массив  $A[N]$  из элементов типа целое 8-разрядное со знаком. Составить программу нахождения максимального и минимального элемента. Разместить индексы максимального и минимального элемента в отдельных ячейках памяти.

Индекс максимального элемента разместить в ячейке IndMax, а индекс минимального элемента разместим в ячейке IndMin.

# Программа

```
.model tiny
.data
N    dw 10          ; Размерность массива.
A    db -13, 5, 2, 6, 11, 5, -4, 127, -9, 10
IndMax dw ?        ; Номер максимального
                        ;элемента в A.
IndMin dw ?        ; Номер минимального
                        ;элемента в A.

.code
org 100h
Start:
mov si, 0           ; инициализируем счётчик цикла.
mov ch, -128        ; инициализируем максимальное
                        ; значение
mov cl, 127         ; инициализируем минимальное
                        ; значение

M1: mov al, A[si]
    cmp al, ch      ; Поиск максимума
    jle M2
    mov ch, al      ; Текущий элемент больше
                        ; максимума
    mov IndMax, si ; Запоминаем его номер
    inc IndMax     ; Корректировка , т.к. si=i-1.
```

```
M2: cmp al, cl
    jge M3
    mov cl, al

    mov IndMin, si
    inc IndMin
    si=i-1.

M3: inc si

    cmpsi, N
    jb M1
end Start
ret
```

```
; Поиск минимума.
; Текущий элемент
; меньше минимума.
; Запоминаем его номер.
; Корректировка номера, т.к.

; Команды завершения тела
; цикла.
```

# Задача

Задан массив  $A[N]$  из элементов типа Word (целое 16-разрядное без знака).

Составить программу сортировки массива по убыванию.

Алгоритм

1. Просматриваем массив целиком, сравнивая каждый раз парные элементы:  $A[i-1]$  и  $A[i]$ .
2. Если возникла ситуация  $A[i-1] < A[i]$ , то меняем элементы местами.
3. Затем повтор просмотра массива сначала.
4. Прекращение сортировки тогда, когда в ходе текущего просмотра массива не произойдёт ни одного обмена ( $ChFlag=0$ ).

# Программа

```
org 100h
.model tiny
.data
N      dw  10          ; Количество элементов в массиве A.
A      dw  0, 4, 1, 2, 15, 10, 20, 11, 3, 5
ChFlag db  0          ; Если ChFlag>0, то в ходе просмотра был обмен.
.code
Start:  mov ChFlag, 0   ; Обнуляем ChFlag перед текущим просмотром.
        mov cx, N      ; Устанавливаем счётчик внутреннего цикла.
        dec cx         ; Пар в массиве на 1 меньше, чем элементов.
        mov si, 2      ; Устанавливаем индекс массива.
M1:     mov ax, A[si-2] ; считываем A[i-1]-й элемент.
        mov bx, A[si]  ; считываем A[i]-й элемент.
        cmp ax, bx     ; сравниваем их.
        jnb M2        ; A[i-1]<A[i] - нужен обмен.
        mov A[si], ax ; A[i-1]<A[i] - нужен обмен.
        mov A[si-2], bx
        inc ChFlag     ; произошёл очередной обмен.
M2:     add si, 2      ; завершение тела внутреннего цикла.
        loop M1
        cmp ChFlag, 0 ; завершение тела внешнего цикла.
        jne Start     ; были обмены - нужна ещё итерация.
        end Start

ret
```