Add array elements:

INCLUDE Irvine32.inc

.data

arr DWORD 10, 20, 30, 40, 50 ; Array of 5 elements

len DWORD LENGTHOF arr ; Number of elements

sum DWORD 0 ; Variable to store sum

msg BYTE "Sum: ", 0 ; Message to display

.code

main PROC

mov ecx, len ; Set loop counter (5 elements)

mov esi, OFFSET arr ; Load address of array

mov eax, 0 ; Initialize sum to 0

sum\_loop:

add eax, [esi] ; Add array element to sum

add esi, 4 ; Move to next element (DWORD = 4 bytes)

loop sum\_loop ; Repeat until ecx = 0

mov sum, eax ; Store final sum

; Display "Sum: "

mov edx, OFFSET msg

call WriteString

; Display sum

mov eax, sum

call WriteInt

call Crlf ; Print newline

exit ; Exit program

main ENDP

END main

Multiply elements of array:

INCLUDE Irvine32.inc

.data

arr DWORD 2, 3, 4, 5, 6 ; Array of 5 elements

len DWORD LENGTHOF arr ; Number of elements

product DWORD 1 ; Variable to store product (initialize to 1)

msg BYTE "Product: ", 0 ; Message to display

.code

main PROC

mov ecx, len ; Set loop counter (5 elements)

mov esi, OFFSET arr ; Load address of array

mov eax, 1 ; Initialize product to 1

mul\_loop:

imul eax, [esi] ; Multiply eax by current array element

add esi, 4 ; Move to next element (DWORD = 4 bytes)

loop mul\_loop ; Repeat until ecx = 0

mov product, eax ; Store final product

; Display "Product: "

mov edx, OFFSET msg

call WriteString

; Display product

mov eax, product

call WriteInt

call Crlf ; Print newline

exit ; Exit program

main ENDP

END main

Elements of an array using loop:

INCLUDE Irvine32.inc

.data

arr DWORD 10, 20, 30, 40, 50 ; Array of 5 elements

len DWORD LENGTHOF arr ; Number of elements

msg BYTE "Array Elements: ", 0 ; Message to display

newline BYTE 10, 0 ; Newline character

.code

main PROC

; Display "Array Elements: "

mov edx, OFFSET msg

call WriteString

; Call recursive function to print elements

mov ecx, len ; Load array length

mov esi, OFFSET arr ; Load array address

call PrintArray

call Crlf ; Print newline after elements

exit ; Exit program

main ENDP

; Recursive procedure to print array elements

PrintArray PROC

push ecx ; Save ecx (counter)

push esi ; Save esi (array pointer)

cmp ecx, 0 ; Base case: If ecx == 0, return

je done

mov eax, [esi] ; Load current element

call WriteInt ; Print element

call Crlf ; Print newline

add esi, 4 ; Move to next element (DWORD = 4 bytes)

dec ecx ; Decrement counter

call PrintArray ; Recursive call

done:

pop esi ; Restore esi

pop ecx ; Restore ecx

ret

PrintArray ENDP

END main

First 10 elements of fibonacci sequence:

INCLUDE Irvine32.inc

.code

main PROC

mov edx, 0

mov ebx, 1

mov ecx, 10

L1:

mov eax, edx

call WriteDec

call Crlf

add eax, ebx

mov edx, ebx

mov ebx, eax

loop L1

invoke ExitProcess, 0

main ENDP

End main

Pattern printing ascending:

INCLUDE Irvine32.inc

.code

main PROC

mov ecx, 4

mov ebx, 1

outer:

mov edx, ecx

mov ecx, ebx

inc ebx

inner:

mov eax, 1

call writedec

loop inner

call crlf

mov ecx, edx

loop outer

exit

main ENDP

end main

Pattern printing descending:

INCLUDE Irvine32.inc

.code

main PROC

mov ecx, 4

mov eax, 1

mov ebx, 4

outer:

mov edx, ecx

mov ecx, ebx

dec ebx

inner:

call writedec

loop inner

call crlf

mov ecx, edx

loop outer

exit

main ENDP

end main

Reversal of an array:

INCLUDE Irvine32.inc

.data

arr DWORD 10, 20, 30, 40, 50 ; Array of 5 elements

len DWORD LENGTHOF arr ; Number of elements

msg1 BYTE "Original Array: ", 0

msg2 BYTE "Reversed Array: ", 0

newline BYTE 10, 0

.code

main PROC

; Display "Original Array: "

mov edx, OFFSET msg1

call WriteString

call PrintArray

call Crlf

; Reverse the array using two-pointer approach

mov esi, OFFSET arr ; Start pointer (left)

mov edi, OFFSET arr ; End pointer (right)

add edi, (len-1) \* 4 ; Move to last element

mov ecx, len ; Loop counter = len / 2

shr ecx, 1 ; Divide by 2 (since swapping pairs)

reverse\_loop:

mov eax, [esi] ; Load left element

mov ebx, [edi] ; Load right element

mov [esi], ebx ; Swap left with right

mov [edi], eax ; Swap right with left

add esi, 4 ; Move left pointer forward

sub edi, 4 ; Move right pointer backward

loop reverse\_loop ; Repeat for half the array

; Display "Reversed Array: "

mov edx, OFFSET msg2

call WriteString

call PrintArray

call Crlf

exit ; Exit program

main ENDP

; Procedure to print array elements

PrintArray PROC

mov ecx, len ; Set loop counter (5 elements)

mov esi, OFFSET arr ; Load array address

print\_loop:

mov eax, [esi] ; Load current element

call WriteInt ; Print element

call Crlf ; Print newline

add esi, 4 ; Move to next element (DWORD = 4 bytes)

loop print\_loop ; Repeat until ecx = 0

ret

PrintArray ENDP

END main

In the array, take cube of each element and display:

INCLUDE Irvine32.inc

.data

arr DWORD 2, 3, 4, 5, 6 ; Array of 5 elements

len DWORD LENGTHOF arr ; Number of elements

msg BYTE "Cubes of Array Elements:", 0

.code

main PROC

; Display message

mov edx, OFFSET msg

call WriteString

call Crlf

mov ecx, len ; Set loop counter (5 elements)

mov esi, OFFSET arr ; Load array address

cube\_loop:

mov eax, [esi] ; Load current element

imul eax, eax ; Square the number (n²)

imul eax, [esi] ; Multiply by original number (n³)

call WriteInt ; Print cube of element

call Crlf ; Print newline

add esi, 4 ; Move to next element (DWORD = 4 bytes)

loop cube\_loop ; Repeat until ecx = 0

exit ; Exit program

main ENDP

END main

**5 elements as input** from the user and stores them in an array:

INCLUDE Irvine32.inc

.data

arr DWORD 5 DUP(?) ; Array to store 5 elements

len DWORD 5 ; Number of elements

msg BYTE "Enter 5 numbers:", 0

prompt BYTE "Enter a number: ", 0

newline BYTE 10, 0

.code

main PROC

; Display "Enter 5 numbers:"

mov edx, OFFSET msg

call WriteString

call Crlf

; Read 5 numbers from user

mov ecx, len ; Loop counter (5 times)

mov esi, OFFSET arr ; Load array address

input\_loop:

mov edx, OFFSET prompt

call WriteString ; Print "Enter a number: "

call ReadInt ; Read user input

mov [esi], eax ; Store input in array

add esi, 4 ; Move to next element

loop input\_loop ; Repeat for 5 numbers

exit ; Exit program

main ENDP

END main

Add elements of two consecutive arrays:

INCLUDE Irvine32.inc

.data

arr1 DWORD 1, 2, 3, 4, 5 ; First array (hardcoded)

arr2 DWORD 6, 7, 8, 9, 10 ; Second array (hardcoded)

arr3 DWORD 5 DUP(?) ; Result array (arr1[i] + arr2[i])

len DWORD 5 ; Number of elements

msg BYTE "Sum of corresponding elements:", 0

.code

main PROC

; Display message

mov edx, OFFSET msg

call WriteString

call Crlf

; Compute sum of corresponding elements

mov ecx, len ; Loop counter (5 times)

mov esi, OFFSET arr1 ; Load address of arr1

mov edi, OFFSET arr2 ; Load address of arr2

mov ebx, OFFSET arr3 ; Load address of arr3

sum\_loop:

mov eax, [esi] ; Load arr1[i]

add eax, [edi] ; Add arr2[i]

mov [ebx], eax ; Store sum in arr3[i]

add esi, 4 ; Move to next element in arr1

add edi, 4 ; Move to next element in arr2

add ebx, 4 ; Move to next element in arr3

loop sum\_loop ; Repeat for all elements

; Print arr3 elements

mov ecx, len ; Loop counter (5 times)

mov esi, OFFSET arr3 ; Load address of arr3

print\_loop:

mov eax, [esi] ; Load current sum element

call WriteInt ; Print the sum

call Crlf ; Print newline

add esi, 4 ; Move to next element

loop print\_loop ; Repeat for 5 numbers

exit ; Exit program

main ENDP

END main