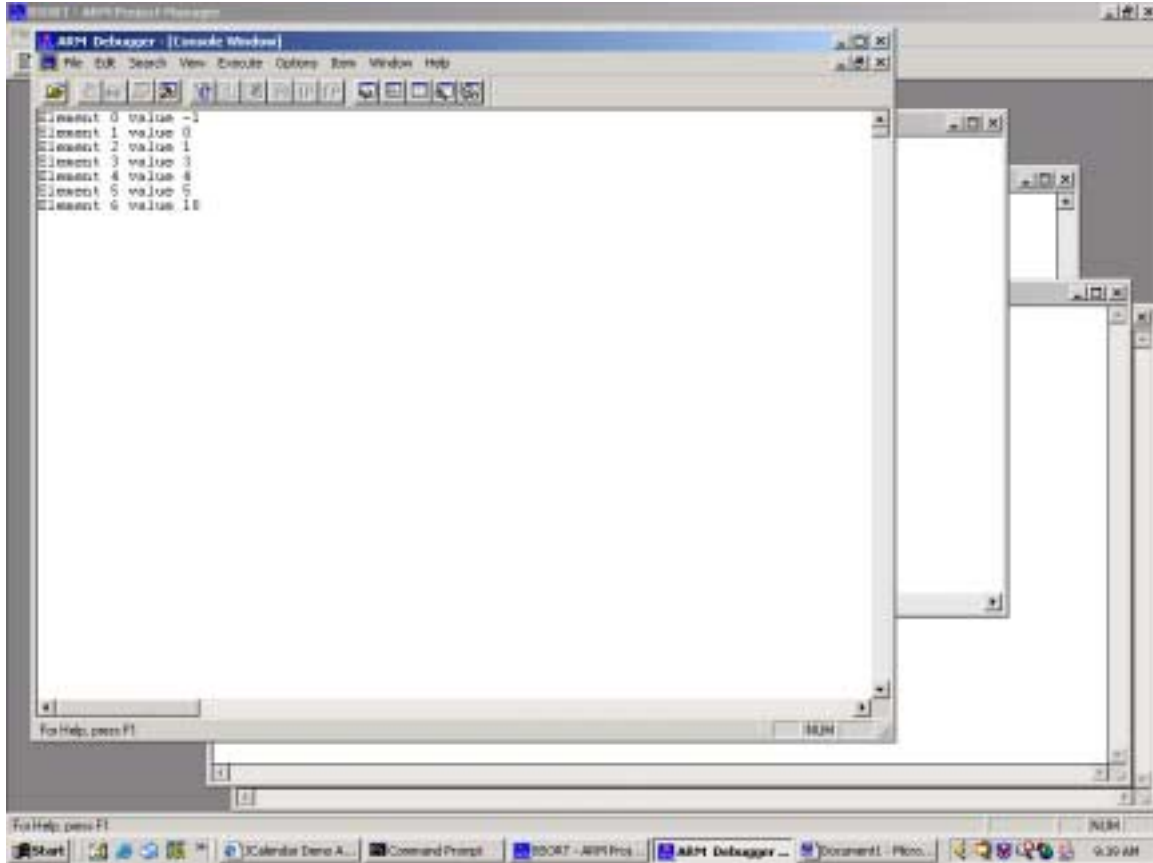


## Example Write-up for Sort Routine

Description of solution: The sorting of the array of integers was done using the bubble sort. Two loops were used to sweep through the array. The inner loop interchanges array elements that are out of order and sets a flag when this occurs. The outer loop checks for interchanged elements. If there are any it repeats the inner loop, otherwise the loop is finished. The array is output using the C language `_printf` routine.

Below is a screen capture of the final output of the sorted array.



The screenshot shows a window titled 'ARM Debugger [Console Window]' with a menu bar (File, Edit, Search, View, Execute, Options, Run, Window, Help) and a toolbar. The console output displays the following text:

```
Element 0 value -1  
Element 1 value 0  
Element 2 value 1  
Element 3 value 3  
Element 4 value 8  
Element 5 value 9  
Element 6 value 18
```

The window also features a 'RUN' button at the bottom right and a status bar at the bottom with the text 'For Help, press F1'. The taskbar at the bottom of the screen shows several open applications, including 'Calendar Demo A...', 'Command Prompt', 'SDORT - ARM Proj...', 'ARM Debugger...', and 'Document1 - Ploco...'. The system clock in the bottom right corner indicates the time is 9:39 AM.

The information included below would normally be attached as listings from the ARM development environment using the 'File' menu item 'Print' (not a screen capture).

```
#include <stdlib.h>

extern int asm_bsort( int array[], int size );

int main( int argc, char * argv[] )
{
    int arraytosort[] = { 10, 3, -1, 0, 5, 4, 1 };
    int result ;
    int amt_to_sort = sizeof(arraytosort)/sizeof(int) ;

    result = asm_bsort( arraytosort, amt_to_sort ) ;

    exit( 0 ) ;
}
```

```
; asm_bsort( int arraytosort, int sizeof(arraytosort) )
```

```
    AREA |asm_bsort$code|, CODE, READONLY  
    EXPORT asm_bsort  
    IMPORT _printf
```

```
; r0 = a1, pointer from C  
; r1 = a2, number of elements in array  
; r2 = a3  
; r3 = a4  
; r4 = v1, copy of pointer from C  
; r5 = v2, count of elements-1  
; r6 = v3, element i of array  
; r7 = v4, element i+1 of array  
; r8 = v5, flag if a switch  
; r9 = v6, temporary location for element switch
```

```
asm_bsort
```

```
    stmfd sp!, {v1-v6,lr} ; save registers, standard entry
```

```
outloop
```

```
    mov v1, a1 ; copy pointer for use below  
    mov v2, a2 ; copy count of elements  
    sub v2, v2, #1 ; one less comparison than elements  
    mov v5, #0 ; set swap flag to zero
```

```
inrlop
```

```
    ldmbia v1, {v3, v4} ; get two elements to compare  
    cmp v3, v4 ; set condition flags  
    movgt v6, v3 ; interchange them if gt is true  
    movgt v3, v4  
    movgt v4, v6  
    stmgia v1, {v3, v4} ; put them back if order is changed  
    movgt v5, #1 ; set the swapped element flag  
    add v1, v1, #4 ; move to next integer  
    subs v2, v2, #1 ; all comparisons done?  
    bne inrlop ; not yet  
    cmp v5, #1 ; did we do a swap?  
    beq outloop ; yes, sweep thru again
```

```
; v1 = pointer from C  
; v2 = number of elements  
; v3 = element number for printing
```

```
    mov v1, a1  
    mov v2, a2  
    mov v3, #0
```

```
ploop
```

```
    adr    a1, stext      ; printf format string
    mov    a2, v3         ; element number
    ldr    a3, [v1], #4   ; array element, increment pointer
    bl     _printf
    add    v3, v3, #1     ; next array element number
    cmp    v2, v3        ; done all elements?
    bne    ploop

    mov    r0, #1        ; result value for C (ignored)
    ldmfd  sp!, {v1-v6, pc} ; restore the registers and pc

AREA |asm_bsort$data|, DATA
stext DCB  "Element %d value %d\n",0

END
```