Lab 9: Advanced Procedures:

#Task1:

INCLUDE Irvine32.inc

.data

prompt BYTE "Enter 3 integers: ", 0

resultMsg BYTE "Product of 3 numbers: ", 0

.code

main PROC

mov edx, OFFSET prompt

call WriteString

call Crlf

call ReadInt

push eax

call ReadInt

push eax

call ReadInt

push eax

call ThreeProd

call Crlf

exit

main ENDP

ThreeProd PROC

push ebp

mov ebp, esp

mov eax, [ebp + 8]

imul eax, [ebp + 12]

imul eax, [ebp + 16]

mov edx, OFFSET resultMsg

call WriteString

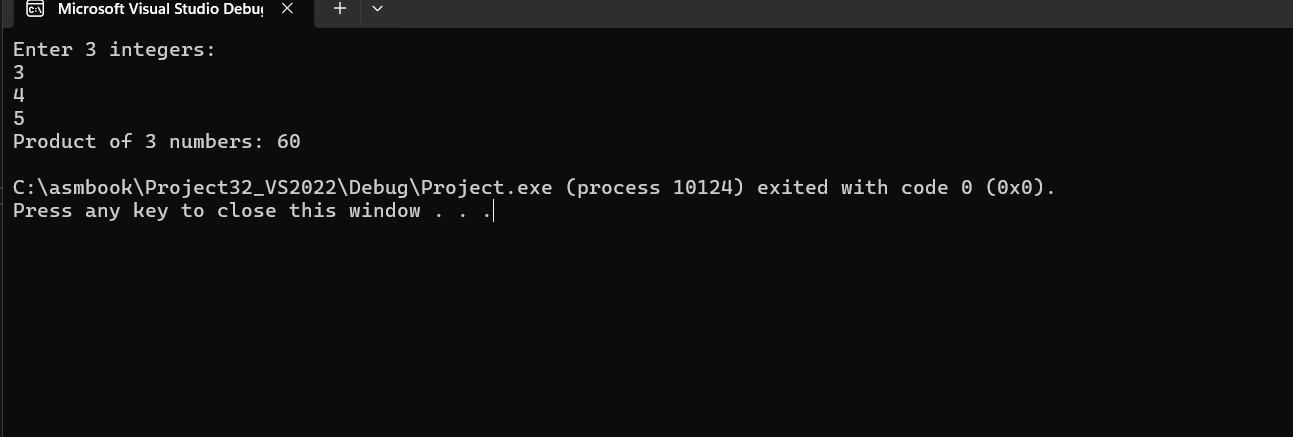
call WriteDec

pop ebp

ret 12

ThreeProd ENDP

END main



#Task2:

INCLUDE Irvine32.inc

MinMaxArray PROTO, pArray:PTR DWORD

.data

array DWORD 45, 12, 89, 4, 73, 8, 91, 25, 67, 34, 21, 3, 17, 56, 78, 99, 2, 100, 33, 65

msgMin BYTE "Minimum value: ", 0

msgMax BYTE "Maximum value: ", 0

.code

main PROC

INVOKE MinMaxArray, ADDR array

call Crlf

exit

main ENDP

MinMaxArray PROC,

pArray: PTR DWORD

push esi

push edi

push ecx

mov esi, pArray

mov eax, [esi]

mov ebx, [esi]

mov ecx, 19

add esi, 4

L1:

mov edx, [esi]

cmp edx, eax

jl SetMin

cmp edx, ebx

jg SetMax

jmp Skip

SetMin:

mov eax, edx

jmp Skip

SetMax:

mov ebx, edx

Skip:

add esi, 4

loop L1

mov edx, OFFSET msgMin

call WriteString

mov eax, eax

call WriteDec

call Crlf

mov edx, OFFSET msgMax

call WriteString

mov eax, ebx

call WriteDec

call Crlf

pop ecx

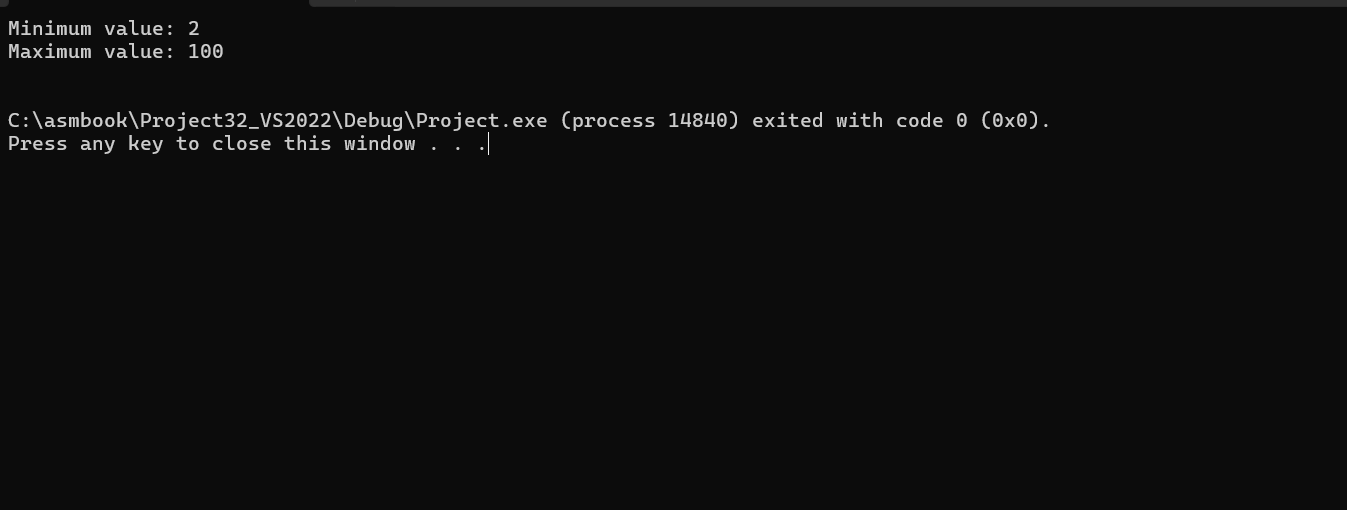
pop edi

pop esi

ret

MinMaxArray ENDP

END main



#Task3:

INCLUDE Irvine32.inc

.data

prompt BYTE "Enter a number: ", 0

msg BYTE "Square of the number: ", 0

.code

main PROC

call LocalSquare

call Crlf

exit

main ENDP

LocalSquare PROC

enter 4, 0

mov edx, OFFSET prompt

call WriteString

call ReadInt

mov [ebp - 4], eax

mov eax, [ebp - 4]

imul eax, eax

mov edx, OFFSET msg

call WriteString

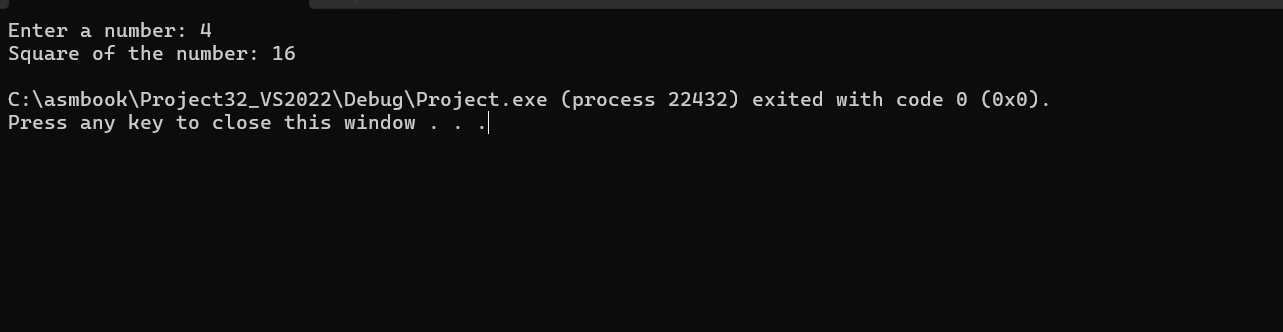
call WriteDec

leave

ret

LocalSquare ENDP

END main



#Task4:

INCLUDE Irvine32.inc

CheckPrime PROTO, num: DWORD

LargestPrime PROTO, pArr: PTR DWORD

.data

prompt BYTE "Enter 4 integers: ", 0

notAllPrime BYTE "Not all numbers are prime.", 0

largestMsg BYTE "Largest prime is: ", 0

nums DWORD 4 DUP(?)

.code

main PROC

mov edx, OFFSET prompt

call WriteString

call Crlf

mov ecx, 4

lea esi, nums

read\_loop:

call ReadInt

mov [esi], eax

add esi, 4

loop read\_loop

mov ecx, 4

lea esi, nums

mov ebx, 0

check\_loop:

mov eax, [esi]

INVOKE CheckPrime, eax

cmp eax, 0

je not\_all\_prime

add esi, 4

loop check\_loop

INVOKE LargestPrime, ADDR nums

jmp done

not\_all\_prime:

mov edx, OFFSET notAllPrime

call WriteString

call Crlf

done:

call Crlf

exit

main ENDP

CheckPrime PROC uses eax ecx edx,

num: DWORD

mov eax, num

cmp eax, 2

jl not\_prime

cmp eax, 2

je is\_prime

mov ecx, 2

mov edx, 0

check\_loop:

mov ebx, num

mov edx, 0

div ecx

cmp edx, 0

je not\_prime

inc ecx

mov edx, 0

mov eax, num

cmp ecx, eax

jl check\_loop

is\_prime:

mov eax, 1

ret

not\_prime:

mov eax, 0

ret

CheckPrime ENDP

LargestPrime PROC uses esi ecx eax,

pArr: PTR DWORD

mov esi, pArr

mov eax, [esi]

mov ecx, 3

add esi, 4

check\_max:

cmp [esi], eax

jle skip

mov eax, [esi]

skip:

add esi, 4

loop check\_max

mov edx, OFFSET largestMsg

call WriteString

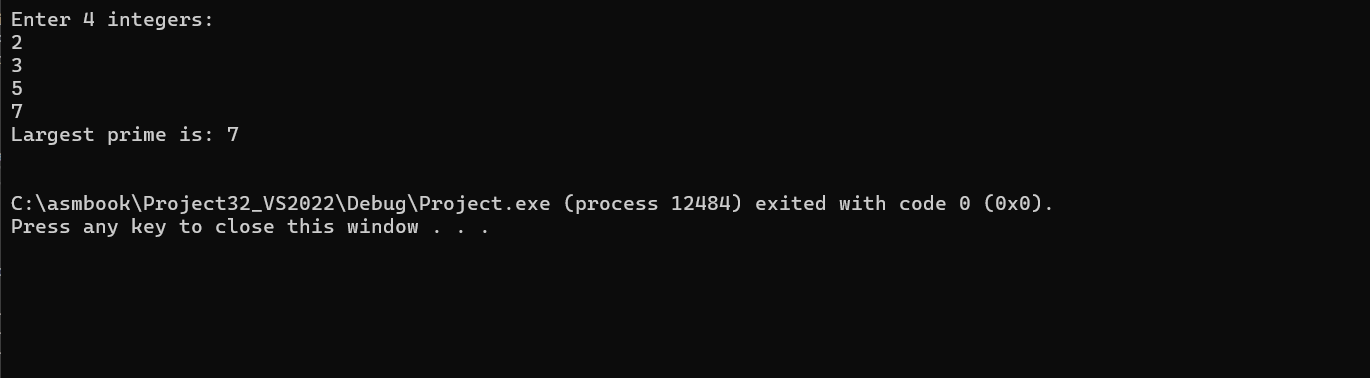
call WriteDec

call Crlf

ret

LargestPrime ENDP

END main



#Task5:

INCLUDE Irvine32.inc

.data

array DWORD 72, 12, 29, 18, 50, 31, 82

before BYTE "Original array: ", 0

after BYTE "Sorted array: ", 0

space BYTE " ", 0

.code

main PROC

mov edx, OFFSET before

call writeString

call crlf

mov ecx, LENGTHOF array

mov esi, OFFSET array

original:

mov eax, [esi]

call writedec

mov edx, offset space

call writestring

add esi, TYPE array

LOOP original

call crlf

PUSH OFFSET array

PUSH LENGTHOF array

call BubbleSort

mov edx, OFFSET after

call writeString

call crlf

mov ecx, LENGTHOF array

mov esi, OFFSET array

L1:

mov eax, [esi]

call writedec

mov edx, offset space

call writestring

add esi, TYPE array

LOOP L1

call crlf

exit

main ENDP

BubbleSort PROC

PUSH ebp

mov ebp, esp

mov ecx, [ebp + 8]

dec ecx

mov esi, [ebp + 12]

outer:

mov edx, ecx

mov edi, esi

inner:

mov eax, [edi]

mov ebx, [edi + TYPE array]

cmp eax, ebx

jle skip

mov [edi], ebx

mov [edi + TYPE array], eax

skip:

add edi, TYPE array

dec edx

jnz inner

dec ecx

jnz outer

pop ebp

ret

BubbleSort ENDP

END main

