CSE101 Assignment4

Homework Solution

```
/*
 * CSE101 Assignment 4
 * Author: Dixing Xu (1509014)
 * Date: 2016/12/05
 * Used both array handling method shown on PDF
 * Used FPU to handle median
 * Used bubble sort to arrange the array
 * maximum loop : 999 times
 * Future Improvement: non digit handling
  */



#include "stdafx.h"


int _tmain(int argc, _TCHAR* argv[])
{
    int myarr[999];
    int bigone;
    int bigtwo;
    int smallone;
    int smalltwo;
    int checkpointstop;
    int upper;
    int lower;

char ask[] = "\nEnter the current reading(use 99 to stop): ";
char format_int[] = "%d";

char output[] = "The number is: %d\n";


float a; //midpoint + (midpoint+1)
float b=2.0; //2
```

```
char wronginputms[] = "\nOut of boundary. Please try again!";
char outms1[] = "\nThe smallest number is: %d";
char outms2[] = "\nThe second smallest number is: %d";
char outms3_i[] = "\nThe median number is: %d";
char outms3_f[] = "\nThe median number is: %.1f"; //for float display
char outms4[] = "\nThe second largest number is: %d";
char outms5[] = "\nThe largest number is: %d\n";


int count;
int ctr;
_asm {
    mov count, 0
    mov checkpointstop, 99
    mov ctr, 999
    mov ebx, 0
    mov eax, 0
    mov esi, 0

    mov upper, 15 //set upper bound to 15
    mov lower, -15 //set lower bound to -15

    read:
        lea eax, ask
        push eax
        call printf_s   //print the prompt
        add esp, 4

        lea eax, myarr[ebx]
        push eax
        lea eax, format_int
        push eax
        call scanf_s  //read input from user and store to myarr[ebx]
        add esp, 8

        mov eax, myarr[ebx]
        cmp eax, checkpointstop
        je reset //use 99 to stop
        cmp eax, upper
        jg wronginput
        cmp eax, lower
        jl wronginput

        add ebx, 4
        dec ctr
```

```
        inc count
        cmp ctr, 0  //max loop 999 times
        jg read

        wronginput:
        lea eax, wronginputms
        push eax
        call printf_s
        add esp, 4
        jmp read


        reset:
    mov eax, 0
    mov ebx, 0

    mov ecx, count; //OuterLoop counter


    OuterLoop:
       cmp ecx, 1; //check length
       jle resetfinal
       lea edi, myarr; //move in list
       push ecx;

       mov ecx, count;
       dec ecx; //Decrement inner loop
    InnerLoop:
       mov eax, [edi]; //First Value in list
       cmp eax, [edi+4]; //next int is 4 bits away
       jg  Swap;
       jmp Continue;
    Swap:
       mov edx, [edi+4]; //Move to temp register
       mov [edi+4], eax; //Move stored value there
       mov [edi], edx; //Return temp value to list
    Continue:
       add edi, 4; //Increment to move to next int
       loop InnerLoop;
       pop ecx; //reset counter
       loop OuterLoop;


        resetfinal:
        mov edi, 0 // reset edi to 0
```

```
        mov eax, [myarr + edi] //read an element of the array at the addres
s stored in ebx
        add edi, 4
        push eax
        lea eax, outms1
        push eax
        call printf_s
        add esp, 8

        mov eax, [myarr + edi]
        add edi, 4
        push eax
        lea eax, outms2
        push eax
        call printf_s
        add esp, 8

        mov ecx, count
        shr ecx, 1; // ecx >>= 1 , LSB£¨Least Significant Bit£© moved to CF

        jc odd_number // use CF to judge the number is odd or even

        //use float to display
        mov ecx, count //reset
        sar ecx, 1
        imul ecx, 4
        mov edi, ecx
        mov esi, edi //esi stores mid_point+1
        sub edi, 4 //edi stores mid_point
        mov eax, [myarr + edi]
        add eax, [myarr + esi]
        mov a, eax
        fild dword ptr [a] //load float number at the address of a to st(0)
        fdiv b
        fstp qword ptr [esp] // IMPORTANT: convert to double and store, bec
ause printf expects a double not a float
        lea  eax, outms3_f
        push eax
        call printf_s
        add  esp, 4
        jmp lasttwo
```

```
        odd_number:
        mov ecx, count //reset
        sar ecx, 1 //arraylength divided by 2
        imul ecx, 4 //int size
        mov edi, ecx
        mov eax, [myarr + edi]
        push eax
        lea eax, outms3_i
        push eax
        call printf_s
        add esp, 8

        lasttwo:
        mov ecx, count
        imul ecx, 4
        mov edi, ecx
        sub edi, 8
        mov eax, [myarr + edi]
        push eax
        lea eax, outms4
        push eax
        call printf_s
        add esp, 8

        add edi, 4
        mov eax, [myarr + edi]
        push eax
        lea eax, outms5
        push eax
        call printf_s
        add esp, 8




}
return 0;
}
```