ALT SEVİYE PROGRAMLAMA

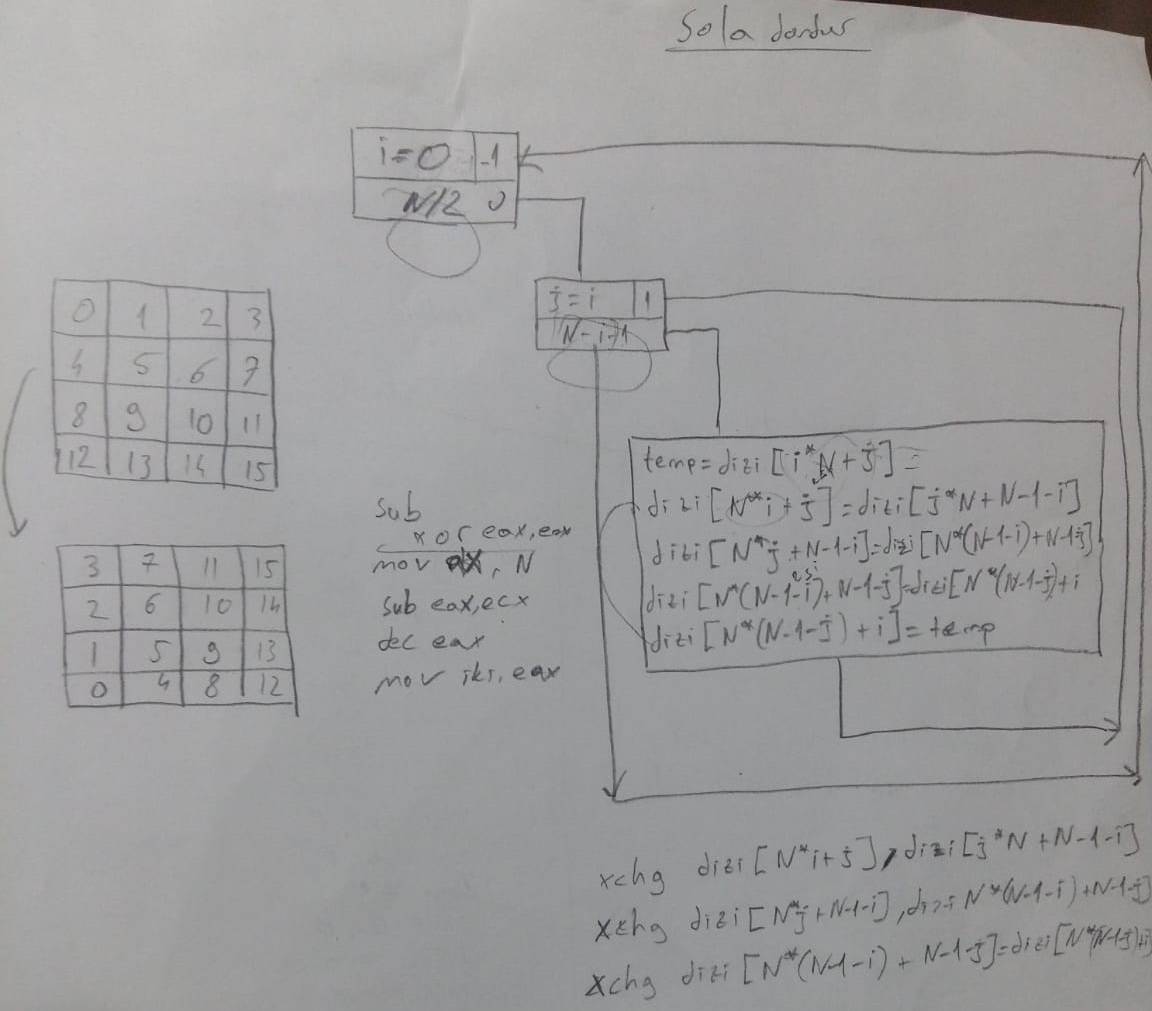
1.ÖDEV

ÖMER LÜTFÜ TORTUMLU

16011110

1.SORU

🡪Sola döndürme algoritması



🡪Resmi sağa ve sola döndürme kodu

void sagaDondur(short n, int resim) {

//KODUNUZU BURADAN BASLAYARAK YAZINIZ

unsigned int bir, iki, i; //word tipinde değişkenler atıyorum

unsigned short temp; //byte tipinde swap için değişken atiyorum

\_\_asm {

xor ecx, ecx //yapicagım işlemler 16 biti aşicağı için dd de işlem yapıyorum

xor eax, eax //yapicagım işlemler 16 biti aşicağı için dd de işlem yapıyorum

mov ax, n //n degerini ax e atiyorum

shr eax, 1 //n i 2 ye bölüyorum yarısını

mov bir, eax //eax teki degeri ilk lopun dongu sayısına atiyorum

l1 :

xor eax, eax //eax i boşaltıyorum

mov ax, n //n degerini eax e atiyorum

sub eax, ecx //eax ten ecx i cıkartiyorum

dec eax //eax teki degeri 1 azaltiyorum

mov iki, eax //ikinci lopun sinirini belirliyorum

mov i, ecx //ikinci lopun başlangıç değerini atiyorum

push ecx //ecx i değişmemesi için stack e atıyorum

l2 :

xor eax, eax //eax i boşaltıyorum

mov ax, n //n degerini eax e atiyorum

dec eax //n-1 değerini elde ediyorum

sub eax, ecx //eax ten ecx i çıkartiyorum n-1-j ecx te j değeri tutuluyor

xor ebx, ebx //ebx i boşaltıyorum

mov ebx, eax //N - 1 - j değeri ebx e atandı

;birinci exchange temp<-esi

xor eax, eax

mov ax, n

mul i //kodda dx i her mulda değişeceği için kullanmadım

add eax, ecx

shl eax, 1

mov esi, resim //resim dizisinin ilk indisine erişiliyor

add esi, eax //işlem yapmak istediğimiz indise ulaşıyoruz

mov ax, word ptr[esi] //swap için resimden değer alıyoruz

mov temp, ax //aldığımız indis değerini tempte saklıyoruz

;ikinci Exchange // esi<-edi

xor eax, eax

mov ax, n

mul ebx

add eax, i

shl eax, 1

mov edi, resim

add edi, eax

mov ax, word ptr[edi]

mov word ptr[esi], ax

;ucuncu exchange //edi<-esi

xor eax, eax

mov ax, n

mul iki

add eax, ebx // N\*(N - 1 - i) + N - 1 - j değeri eax e atandı

shl eax, 1

mov esi, resim

add esi, eax

mov ax, word ptr[esi]

mov word ptr[edi], ax

;dorduncu exchange //esi<-edi

xor eax, eax

mov ax, n

mul ecx

add eax, iki

shl eax, 1

mov edi, resim

add edi, eax

mov ax, word ptr[edi]

mov word ptr[esi], ax

;besinci exchange //edi<-temp

mov ax, temp

mov word ptr[edi], ax

inc ecx

cmp ecx, iki

jl l2

pop ecx

inc ecx

cmp ecx, bir

jl l1

}

//KODUNUZU YAZMAYI BURADA BITIRINIZ

}

//////////

Sola dondur işleminde ikinci Exchange ile ile dorduncu exchange yer degistirip işlemleri aynı şekilde yapıyorum

//////////

void solaDondur(short n, int resim) {

//KODUNUZU BURADAN BASLAYARAK YAZINIZ

unsigned int bir, iki, i;

unsigned short temp;

\_\_asm {

xor ecx,ecx

xor eax,eax

mov ax,n

shr eax,1

mov bir,eax

l1:

xor eax,eax

mov ax,n

sub eax,ecx

dec eax

mov iki,eax

mov i,ecx

push ecx

l2:

;birinci Exchange temp<-esi

xor eax,eax

mov ax,n

mul i

add eax,ecx

shl eax,1

mov esi,resim

add esi,eax

mov ax, word ptr[esi]

mov temp,ax

;ikinci exchange esi<-edi

xor eax,eax

mov ax,n

mul ecx

add eax,iki

shl eax,1

mov edi,resim

add edi,eax

mov ax,word ptr[edi]

mov word ptr[esi],ax

;ucuncu exchange edi<-esi

xor eax,eax

mov ax,n

dec eax

sub eax,ecx

xor ebx,ebx

mov ebx,eax ;N-1-j değeri ebx e atandı

xor eax,eax

mov ax,n

mul iki

add eax,ebx ;N\*(N-1-i)+N-1-j değeri eax e atandı

shl eax,1

mov esi,resim

add esi,eax

mov ax,word ptr[esi]

mov word ptr[edi],ax

;dorduncu exchange esi<-edi

xor eax,eax

mov ax,n

mul ebx

add eax,i

shl eax,1

mov edi,resim

add edi,eax

mov ax,word ptr[edi]

mov word ptr[esi],ax

;besinci exchange edi<-temp

mov ax,temp

mov word ptr[edi],ax

inc ecx

cmp ecx,iki

jl l2

pop ecx

inc ecx

cmp ecx,bir

jl l1

}

}

2.Ödev

Bu soruda sadece kullanıcıdan verileri alıp diziye atadım.

myss SEGMENT PARA STACK 'STACK'

stk db 100 dup(0)

myss ENDS

myds SEGMENT PARA 'DATA'

CR EQU 13

LF EQU 10

MSG1 DB '.nci sayi :',0

Msg2 DB 'Dizinin boyutunu giriniz:',0

buyukhata db cr,lf, 'Dikkat!!! Deger 0-100 arasinda olsun' , 0

hata db cr,lf, 'Dikkat!!!' , 0

bosmsg db cr,lf, '' , 0

dizi db 100 dup(?)

pivot db ?

n dw ?

son db ?

myds ends

mycs segment para 'code'

assume cs:mycs ,ss:myss ,ds:myds

ana proc far

push ds

xor ax,ax

push ax

mov ax,myds

mov ds,ax

albak:

mov ax,offset msg2

call put\_str

call getn

cmp ax,100

jg error1

cmp ax,0

jle error1

mov n,ax

jmp atla1

error1:

mov ax,offset buyukhata

call put\_str

jmp albak

atla1:

xor cx,cx

mov cx,n

xor si,si

input:

mov ax,offset bosmsg

call put\_str

push ax

mov ax,si

inc ax

call putn

pop ax

mov ax,offset msg1

call put\_str

call getn

mov dizi[si],al

inc si

dec cx

cmp cx,0

jne input

xor ax,ax

mov ax,n

mov son,al

call quick

xor cx,cx

mov cx,n

xor si,si

yazdir:

mov ax,offset bosmsg

call put\_str

push ax

mov ax,si

inc ax

call putn

pop ax

mov ax,offset msg1

call put\_str

mov al,dizi[si]

call putn

inc si

dec cx

cmp cx,0

jne yazdir

;call getn kullanicidan deger aliniyor

;mov ax,offset cumle cumlenin ilk adresi ax e ataniyor

;call put\_str cumle yazdiriliyor

;call putn ax te ne varsa yazdırıyor

;putc al de hangi simge varsa o yazdiriliyor

retf

ana endp

quick proc NEAR

ret

quick endp

partition proc NEAR

ret

partition endp

getc proc near

mov ah,1h

int 21h

ret

getc endp

putc proc near

push ax

push dx

mov dl,al

mov ah,2

int 21h

pop dx

pop ax

ret

putc endp

getn proc near

push bx

push cx

push dx

getn\_start:

mov dx,1

xor bx,bx

xor cx,cx

new:

call getc

cmp al,cr

je fin\_read

cmp al,'-'

jne ctrl\_num

negatif:

mov dx,-1

jmp new

ctrl\_num:

cmp al,'0'

jb error

cmp al,'9'

ja error

sub al,'0'

mov bl,al

mov ax,10

push dx

mul cx

pop dx

mov cx,ax

add cx,bx

jmp new

error:

mov ax,offset hata

call put\_str

jmp getn\_start

fin\_read:

mov ax,cx

cmp dx,1

je fin\_getn

neg ax

fin\_getn:

pop dx

pop cx

pop dx

ret

getn endp

putn proc near

push cx

push dx

xor dx,dx

push dx

mov cl,10

cmp al,0

jge hesapla

neg al

push ax

mov al,'-'

call putc

pop ax

hesapla:

div cx

add dl,'0'

push dx

xor dx,dx

cmp al,0

jne hesapla

goruntu:

pop ax

cmp al,0

je end\_goruntu

call putc

jmp goruntu

end\_goruntu:

pop dx

pop cx

ret

putn endp

put\_str proc near

push bx

mov bx,ax

mov al, byte ptr[bx]

put\_loop:

cmp al,0

je put\_fin

call putc

inc bx

mov al,byte ptr[bx]

jmp put\_loop

put\_fin:

pop bx

ret

put\_str endp

mycs ends

end ana