Извештај о пројектном задатку из предмета

РАЧУНАРСКА ЕЛЕКТРОНИКА

Студенти:

Богдан Ђорђевић 2015/063

Тодосијевић Ана 2015/332

Предметни професор: Милан Прокин

Предметни асистент : Александра Лекић

Пројектни задатак број 2

Текст задатка

Написати програм за скалирање слике применом методе најближег суседа. Фактор s, којим се дрши скалирање, уноси се са стандардног улаза и увек је већи или једнак 1. Поред тога, са стандардног улаза на одређени начин дефинише се да ли се врши повећање или децимација слике s пута.

Објашњење кода

Процедуре

Процедура intenzitet служи да се врати интензитет пиксела са одређеним координатама. Координате су x0 и y0. Користимо регистре edx, eax, ecx. Интензитет пиксела се смешта у numBuffer.

Процедура intToString служи за конверзију децималног броја у стринг, користимо; koristi registre edx, ecx i edi и смешта нови стринг у бафер са почетном адресом у edi. Враћа број уписаних бајтова у регистру eax .

Декларисање променљивих

BUFFER_SIZE = 256*256*30 - Процењена величина слике

bufferIn BYTE BUFFER_SIZE DUP(?) – улазни бафер дужине BUFFER_SIZE бајтова, служи за дохватање дела фајла

bufferOut BYTE 10 DUP(?) – излазни бафер који се користи приликом исписа у излазни фајл

inputImageName BYTE 80 DUP(0)- име слике на којој вршимо обраду

outputImageName BYTE 80 DUP(0) – име обрађене слике

inputImageHandle HANDLE ? - handle за улазни фајл

outputImageHandle HANDLE ? - handle за излазни фајл

spom BYTE sbuff DUP(?)- помоћна променљива при уписивању фактора

s DWORD ? – фактор скалирања

koef BYTE ? – 1 за повећање, 0 за смањивање, унос је са стандардног улаза

bytesWritten DWORD ? – бројач уписаних бајтова

numBuffer BYTE 6 DUP(?) – помоћни бафер при испису у излазни фајл, дужина бафера је 6 бајтова јер је за четвороцифрен број потребно додатна 2 бајта за размак и нов ред

buffPtr DWORD 0 - показивач на улазни бафер

pixHeight DWORD 0 –почетна висина слике

pixWidth DWORD 0 – почетна ширина слике

pixWidthScaled DWORD 0 – скалирана ширина

pixHeightScaled DWORD 0 – скалирана висина

pixCount BYTE 0 – бројач који означава да треба да се упише нов ред

pixCount1 DWORD 0 - бројач за врсте у процедури intenzitet

pixCount2 DWORD 0 - бројач за колоне у процедури intenzitet

x0 DWORD ? - x координата пиксела који се преписује

y0 DWORD ? - укоордината пиксела који се преписује

xr DWORD ? –остатак при дељењу x са s

yr DWORD ? –остатак при дељењу y са s

xCount DWORD 0 –помоћни бројач за x

yCount DWORD 0 –помоћни бројач за y

pomPtr DWORD 0 - помоћни бројач за улазну слику

valueBuffer BYTE 4 DUP (?) – бафер у који се смешта вредност пиксела са координате x0 y0

флегови:

prepisuj BYTE 1 – иникатор који указује на то да ли је тренутни пиксел преписан

EOF_indicator BYTE 0 – индикатор за крај фајла

widthIndicator BYTE 1 - индикатор који указује на то да ли је завршено преписивање димензија

pixValueIndicator BYTE 1 - индикатор да ли је уписана максимимална вредност интензитета пиксела

Ток главног програма

Прво се учитавају имена улазног и излазног фајла и проверава се њихова валидност, ако нису валидна имена избацује се одговарајућа порука и завршава се програм. После тога се учитавају коефицијент скалирања и коефицијент који одређује да ли се слика смањује или повећава.

Позива се функција OpenInputFile и CreateOutputFile које учитавају улазну слику и праве излазни фајл.

Редом се преписују "магични број" P2, скалиране димензије слике и максимална вредност пиксела. Приликом скалирања димензија је узето у обзир да ли се повећава број цифара. Коментари се не преписују.

Приликом обраде слике бројачи pixCount1 и pixCount2 представљају координате у излазној слици. Према одређеним формулама се одређују координате у улазној слици x0 i y0.

За децимацију x0=pixCount1*s, y0=pixCount2*s.

За повећање x0=pixCount1/s, y0= pixCount2/s, xr=pixCount1-s*x0, yr=pixCount2-s*y0. Ако је xr<s-xr онда је x0=x0+1, yr<s-yr онда је y0=y0+1,

Позива се процедура intenzitet која враћа интензит пиксела са координатама x0 i y0, који се преписује у излазни фајл. После 20 преписаних пиксела се уписује нов ред.

Из Irvine.inc библиотеке се користе рутине

- ReadFromFile
- WriteToFile
- ParseDecimal32
- ReadString
- OpenInputFile
- CreateOutputFile
- CloseFile

```
.code

;procedura za vracanje intenziteta slike sa koordinatama x0,y0
;intenzitet se smesta u numBuffer
intenzitet proc
    push eax
    mov ecx,0
    mov eax, 0
    mov xCount, eax
    mov eax, 0
    mov yCount, eax
    mov eax, buffPtr    ; pocetak ulazne slike
    mov pomPtr, eax
pocetak_brojanja:
    mov edx, pomPtr
    mov al, [edx]
```

```
    cmp al, 20h
    je povecaj_xCount
    cmp al, 0ah
    je nov_red
    jmp nastavak_brojanja
nov_red:
    inc pomPtr
    jmp pocetak_brojanja
povecaj_xCount:
    inc xCount
    inc pomPtr
    mov eax, xCount     ; provera da li xCount ide u nov red
    cmp eax, pixWidth
    je resetuj_xCount
    jmp pocetak_brojanja
resetuj_xCount:         ; nov red , xCount=0 yCount+1
    mov eax, 0
    mov xCount, eax
    inc yCount
    mov eax, yCount     ; provera da li je yCount kraj
    cmp eax, pixHeight
    je kraj
    jmp pocetak_brojanja
nastavak_brojanja:
    mov eax, xCount
    cmp eax, x0
    je dobar_X
    jmp promasaj
dobar_X:
    mov eax, yCount
    cmp eax, y0
    je dobar_XiY
    jmp promasaj
promasaj:
    inc pomPtr
    jmp pocetak_brojanja
dobar_XiY:
    mov edx, pomPtr
    mov al, [edx]
    call IsDigit
    jnz kraj
    mov numBuffer[ecx], al
    inc ecx
    inc pomPtr
    jmp dobar_XiY
kraj:
    pop eax
    ret
intenzitet endp


; rutina za konverziju decimalnog broja u string
; koristi registre edx, ecx i edi i smesta novi string
; u bafer sa pocetnom adresom u edi
; vraca broj upisanih bajtova u registru eax
intToString proc
    push  edx
```

```
    push  ecx
    push  edi
    push  ebp
    mov   ebp, esp
    mov   ecx, 10
 pushDigits:
    xor   edx, edx        ; zero-extend eax
    div   ecx             ; divide by 10; now edx = next digit
    add   edx, 30h        ; decimal value + 30h => ascii digit
    push  edx             ; push the whole dword
    test  eax, eax        ; leading zeros
    jnz   pushDigits
 popDigits:
    pop   eax
    stosb                 ; don't write the whole dword, just the low byte,
eax->edi
    cmp   esp, ebp        ; if esp==ebp, we've popped all the digits
    jne   popDigits
    xor   eax, eax        ; add trailing nul
    stosb
    mov   eax, edi
    pop   ebp
    pop   edi
    pop   ecx
    pop   edx
    sub   eax, edi        ; return number of bytes written
    ret
intToString endp



main proc
    mWrite "Unesite ime ulazne slike: "
    mov edx, OFFSET inputImageName
    mov ecx, SIZEOF inputImageName
    call ReadString
    mWrite "Unesite ime izlazne slike: "
    mov edx, OFFSET outputImageName
    mov ecx, SIZEOF outputImageName
    call ReadString
; s = ?
    mWrite "s= "
    mov edx, OFFSET spom
    mov ecx, SIZEOF spom
    call ReadString
    mov ecx, LENGTHOF spom
    mov edx, OFFSET spom
    call ParseDecimal32
    mov s, eax
; smer, 1 za povecanje 0 za decimaciju
    mWrite "1 za povecanje 0 za decimaciju:"
    call ReadChar
    mov  koef, al
    mov edx, OFFSET inputImageName
    call OpenInputFile
    mov inputImageHandle, eax
    cmp eax, INVALID_HANDLE_VALUE
```

```
    jne create_output
    mWrite<"Neispravno ime ulaznog fajla", 0dh, 0ah>
create_output :
    mov  edx, OFFSET outputImageName
    call CreateOutputFile
    mov  outputImageHandle, eax
    cmp eax, INVALID_HANDLE_VALUE
    jne files_ok
    mWrite<"Neispravno ime izlaznog fajla", 0dh, 0ah>
files_ok :
    mov eax, inputImageHandle
    mov edx, OFFSET bufferIn
    mov ecx, BUFFER_SIZE
    call ReadFromFile
    jnc P2
    mWrite "Error u citanju fajla"
    call WriteWindowsMsg
    jmp close_files
; prepisivanje  P2
P2 :
    cld
    mov ecx, 3
    mov esi, OFFSET bufferIn
    mov edi, OFFSET bufferOut
    rep movsb
;   mov bufferOut[3], 0ah
    mov  eax, outputImageHandle
    mov  edx, OFFSET bufferOut
    mov  ecx, 3                 ;ako se upisuje jos jedan 0ah onda se
odkomentarise ovo
    call WriteToFile            ; u eax se upisuje 4 jer je 4 bajta upisano u
izl fajl
    jc   error_writing
    add  bytesWritten, eax      ;doda se 4 na 0
    mov buffPtr, esi
    mov widthIndicator, 1
    ;read_char ucitava jedan znak i proverava se da li se
    ;ucitava sirina,duzina,intenzitet max, ili obrada
read_char:
    cmp EOF_indicator, 1
    je close_files
    mov edx, buffPtr
    mov al, [edx]
    cmp al, "#"
    je comment_sign
    cmp widthIndicator, 1
    je width_scaling
    cmp pixValueIndicator, 1
    je pix_value_prepisi
    cmp koef, 31h
    je obradaPovecanja
    jmp obradaSmanjivanja
width_scaling:
    mov ecx,0
load_width:
    mov edx, buffPtr
    mov al, [edx]
```

```
    call IsDigit
    jnz width_not_digit
    mov numBuffer[ecx], al
    inc ecx
    inc buffPtr
    jmp load_width
width_not_digit:
    mov edx, OFFSET numBuffer
    call ParseDecimal32
    mov pixWidth, eax
    mov ebx, s
    mov edx,0                    ; mora da se resetuje edx zato sto se deli
EDX:EAX / EBX,tj EDX je high
    cmp koef, 31h
    je povecanjedimenzijaW
    div ebx
    jmp nastavakW
povecanjedimenzijaW:
    mul ebx
nastavakW:
    mov pixWidthScaled , eax
    cmp eax, 10
    jb new_width_under10
    cmp eax, 100
    jb new_width_under100
    cmp eax, 1000
    jb new_width_under1000
    ja new_width_over1000
    inc ecx
    jmp widthnastavak
new_width_under10:
    mov ecx, 1
    jmp widthnastavak
new_width_under100:
    mov ecx, 2
    jmp widthnastavak
new_width_under1000:
    mov ecx, 3
    jmp widthnastavak
new_width_over1000:
    mov ecx, 4
    jmp widthnastavak
widthnastavak:
    mov edi, OFFSET numBuffer
    call intToString
    mov  numBuffer[ecx], 20h
    inc ecx
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    mov widthIndicator, 0
    inc buffPtr
height_scaling:
    mov ecx, 0
load_height:
```

```
    mov edx, buffPtr
    mov al, [edx]
    call IsDigit
    jnz height_not_digit
    mov numBuffer[ecx], al
    inc ecx
    inc buffPtr
    jmp load_height
height_not_digit:
    mov edx, OFFSET numBuffer
    mov numBuffer[ecx], 00h
    call ParseDecimal32
    mov pixHeight, eax
    mov ebx,s
    mov edx,0
    cmp koef, 31h
    je povecanjedimenzijah
    div ebx
    jmp nastavakH
povecanjedimenzijaH:
    mul ebx
nastavakH:
    mov pixHeightScaled, eax
    cmp eax, 10
    jb new_height_under10
    cmp eax, 100
    jb  new_height_under100
    cmp eax, 1000
    jb new_height_under1000
    ja new_height_over1000
    jmp cont_height
new_height_under10:
    mov ecx, 1
    jmp cont_height
new_height_under100:
    mov ecx, 2
    jmp cont_height
new_height_under1000:
    mov ecx, 3
    jmp cont_height
new_height_over1000:
    mov ecx, 4
cont_height:
    mov edi, OFFSET numBuffer
    call intToString
    mov numBuffer[ecx], 0ah
    inc ecx
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    inc buffPtr
    jmp read_char
;kraj dela za prepisivanje i korekciju informacije o visini i sirini slike
comment_sign:
    inc bytesWritten
```

```
    inc buffPtr
    mov edx, buffPtr
    mov al, [edx]
    cmp al, 0ah
    jne comment_sign
    inc buffPtr
    jmp read_char
; kraj dela za obradu komentara
pix_value_prepisi:
    mov ecx, 3
    mov esi, buffPtr
    mov edi, OFFSET numBuffer
    rep movsb
    mov buffPtr, esi
    inc buffPtr
    mov numBuffer[3], 0ah
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 4
    call WriteToFile
    jc   error_writing
    add  bytesWritten,eax
    mov pixValueIndicator, 0
    jmp read_char
; kraj dela za prepisivanje informacije o maksimalnoj vrednosti piksela
;   POCETAK OBRADE

obradaPovecanja:
    mov eax,0
    mov pixCount1, eax
    mov pixCount2, eax
pocetak_obradePOV:
    xor edx, edx
    mov ebx,s
    mov eax, pixCount1
    div ebx                 ; eax=x'/s
    mov x0, eax             ; x0=eax
    mov xr, edx             ; xr=x'-x0*s ostatak pri deljenju
    xor edx, edx
    mov eax, pixCount2
    div ebx
    mov y0,eax
    mov yr,edx
    mov eax, xr
    shl eax, 1              ; mnozenje xr sa 2
    cmp eax, s
    ja x0_povecava_za1
    jmp nastavakx
x0_povecava_za1:
    inc x0
    mov eax, x0
    cmp eax, pixWidth
    je x0_vece_od_W
    jmp nastavakx
x0_vece_od_W:
    dec x0
    jmp nastavakx
```

```
nastavakx:
    mov eax, yr
    shl eax, 1              ; mnozenje yr sa 2
    cmp eax, s
    ja y0_povecava_za1
    jmp nastavaky
y0_povecava_za1:
    inc y0
    mov eax, y0
    cmp eax, pixHeight
    je y0_vece_od_H
    jmp nastavaky
y0_vece_od_H:
    dec y0
    jmp nastavaky
nastavaky:
    call intenzitet         ;intenzitet x0,y0 u numbufferu
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    mov numBuffer[0], 20h   ;razmak posle svakog broja
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    inc pixCount
    mov al, pixCount
    cmp al, 20
    je upisi_nov_redPOV
    jmp ne_novi_redPOV
upisi_nov_redPOV:
    mov numBuffer[0], 0ah
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    mov al, 0
    mov pixCount, al
ne_novi_redPOV:
    inc pixCount1
    mov eax, pixCount1
    cmp eax, pixWidthScaled
    je reset_pixCount1
    jmp pocetak_obradePOV
reset_pixCount1:
    mov eax, 0
    mov pixCount1, eax
    inc pixCount2
    mov eax, pixCount2
    cmp eax, pixHeightScaled
    je close_files
    jmp pocetak_obradePOV
```

```
obradaSmanjivanja:
    mov eax,0
    mov pixCount1, eax
    mov pixCount2, eax
pocetak_obradeDEC:
    xor edx, edx
    mov ebx,s
    mov eax, pixCount1
    mul ebx
    mov x0,eax          ;koordinata x0 je koordinata u originalnoj slici
    xor edx, edx
    mov ebx,s
    mov eax, pixCount2
    mul ebx
    mov y0, eax
    call intenzitet     ;intenzitet x0,y0 u numbufferu
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    call WriteToFile
    jc   error_writing
    mov numBuffer[0], 20h   ;razmak posle svakog broja
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    inc pixCount        ;poveca se pixCount
    mov al, pixCount
    cmp al, 20
    je upisi_nov_redDEC
    jmp ne_novi_redDEC
upisi_nov_redDEC:
    mov numBuffer[0], 0ah
    mov  eax, outputImageHandle
    mov  edx, OFFSET numBuffer
    mov  ecx, 1
    call WriteToFile
    jc   error_writing
    mov al, 0
    mov pixCount, al
ne_novi_redDEC:
    inc pixCount1
    mov eax, pixCount1
    cmp eax, pixWidthScaled
    je reset_pixCount1DEC
    jmp pocetak_obradeDEC
reset_pixCount1DEC:
    mov eax, 0
    mov pixCount1, eax
    inc pixCount2
    mov eax, pixCount2
    cmp eax, pixHeightScaled
    je close_files
    jmp pocetak_obradeDEC
error_writing:
    mWrite <"Error prilikom upisivanja ", 0dh, 0ah>
; zatvaranje fajlova i izlazak
```

```
close_files :
    mWrite < 0ah ,"Kraj obrade ", 0dh, 0ah>
    mov eax, outputImageHandle
    call CloseFile
close_input_file :
    mov eax, inputImageHandle
    call CloseFile
quit:
exit
main endp
end main
```