Računarska elektronika
Projekat 12

Problem:

Zadatak je bio da se napravi program koji, nakon što se zada pozicija kursora, crta horizontalne (pritiskom strelice na gore/dole) i vertikalne linije (pritiskom strelice ulevo/udesno) pomerene za +20/-20. Program završava sa radom pritiskom tastera ENTER.

Rešenje:

Nakon pokretanja se ispisuje poruka koja objašnjava korisniku kako da upravlja programom. Pritiskom tastera F1 moguće je izabrati vrednost za koju će linije biti pomerene prilikom crtanja (moveSize). Podrazumevana vrednost je 20. Pritiskom tastera ENTER prelazi se na zadavanje pozicije kursora. Pozicija kursora se bira pritiskom strelica gore/dole/levo/desno. Nakon biranja pozicije (X_start, Y_start), pritiskom tastera INSERT program prelazi u režim crtanja. Pritiskom strelice na gore crta se horizontalna linija čija je jednačina Y = Y_start + moveSize. Analogno, rezultat je isti, samo pomeren za –moveSize, pritiskom strelice na dole. Pritiskom strelice ulevo, dobija se vertikalna linija čija je jednačina X = X_start – moveSize. Analogno, pritiskom strelice udesno, crta se vertikalna linija pomerena za +moveSize.
Nakon crtanja, moguće je vratiti se na biranje položaja kursora pritiskom tastera INSERT.
Program zavrsava sa radom pritiskom tastera ENTER nezavisno od toga da li se nalazi u režimu pozicioniranja kursora ili crtanja linija.

Neke od situacija koje nisu pomenute u specifikaciji i definicije ponašanja programa u istim:

- Veličina prozora je unapred definisana i smatra se da neće menjati tokom izvršavanja. Kao posledica ovoga javlja se situacija u kojoj dolazi do izobličenja slike ukolko se prozor smanji/poveća.
- Korisnik za poziciju kursora može da izabere samo one koordinate koje su u opsegu koji je ograničen veličinom prozora.
- Ukoliko se izabere crtanje linije van opsega, linija ce biti nacrtana na onom mestu na kojoj se trenutno nalazi kursor. Tako, na primer, ukoliko je kursor na poziciji (0,0) i korisnik pritisne strelicu ulevo, dobiće se vertikalna linija čija je jednačina Y = 0.
- Kada je program u režimu crtanja ili pozicioniranja pritisak bilo kog tastera različitog od ENTER, INSERT, ← , ↑ , → , ↓ , se ignoriše.
- Nakon crtanja i vraćanja u režim biranja pozicije kursora, kursor se nalazi na onoj poziciji na kojoj je bio kada je program počeo sa crtanjem.
- Promenljiva moveSize ne može biti veća od 30h.

Struktura projekta

Folder Project sadrži : Definitions.inc, DrawProcedures.asm, HelloProcedures.asm, SetCursorPositionProcedures.asm, Main.asm

HelloProcedures.asm

```
SetMoveSize PROTO moveSize : PTR WORD

Welcome PROTO moveSize : PTR WORD


---


endl EQU <0dh, 0ah>
BufSize = 80

messageHello LABEL BYTE
BYTE "Use arrows to move cursor or draw, INSERT to toggle between ", endl
BYTE "setting cursor position and drawing. Press ENTER to end the program. ", endl
BYTE "Default move value is 20. To change press F1. ", endl
BYTE "Press ENTER to start the program...", endl
messageSizeHello DWORD($ - messageHello)

messageMoveSize LABEL BYTE
BYTE "Move value?", endl
messageMoveSizeSize DWORD($ - messageMoveSize)

consoleHandle HANDLE 0
bytesWritten  DWORD ?
buffer BYTE BufSize DUP(? ), 0, 0
stdInHandle HANDLE ?
bytesRead DWORD ?
```

DrawProcedures.asm

```
Draw PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

Up PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Down PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Left PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Right PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

DrawVerticalLine PROTO ScreenHeight : WORD
DrawHorizontalLine PROTO ScreenLength : WORD

---


outHandle DWORD ?
xyPos COORD <0, 0>
consoleInfo CONSOLE_SCREEN_BUFFER_INFO <>
```

SetCursorPositionProcedures.asm

```
SetCursorPos PROTO ScreenLength : WORD, ScreenHeight : WORD

GoUp PROTO ScreenHeight : WORD
GoDown PROTO ScreenHeight : WORD
GoRight PROTO ScreenLength : WORD
GoLeft PROTO ScreenLength : WORD

-----------------------------------------------------------------------

outHandle DWORD ?
xyPos COORD <0, 0>
```

Main.asm

```
ProgramLoop PROTO

-----------------------------------------------------------------------

moveSize WORD 20
ScreenLength  WORD  60
ScreenHeight  WORD  60
outHandle DWORD ?
windowRect SMALL_RECT <0, 0, 60, 60>
```

Main.asm

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
INCLUDE Definitions.inc

.data
moveSize WORD 20
ScreenLength  WORD  60
ScreenHeight  WORD  30
outHandle DWORD ?
windowRect SMALL_RECT <0, 0, 60, 60>

.code

ProgramLoop PROC

SetPos : INVOKE SetCursorPos, ScreenLength, ScreenHeight
         INVOKE Draw, moveSize, ScreenLength, ScreenHeight
                jmp SetPos
ret
ProgramLoop ENDP



main PROC

INVOKE Welcome, OFFSET moveSize
call Clrscr

INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov outHandle, eax
INVOKE SetConsoleWindowInfo, outHandle, TRUE, ADDR windowRect

INVOKE ProgramLoop
INVOKE ExitProcess, 0

main ENDP
END main
```

HelloProcedures.asm

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
INCLUDE Definitions.inc

.data
endl EQU <0dh, 0ah>
BufSize = 80

messageHello LABEL BYTE
BYTE "Use arrows to move cursor or draw, INSERT to toggle between ", endl
BYTE "setting cursor position and drawing. Press ENTER to end the program. ", endl
BYTE "Default move value is 20. To change press F1. ", endl
BYTE "Press ENTER to start the program...", endl
messageSizeHello DWORD($ - messageHello)

messageMoveSize LABEL BYTE
BYTE "Move value?", endl
messageMoveSizeSize DWORD($ - messageMoveSize)

consoleHandle HANDLE 0
bytesWritten  DWORD ?
buffer BYTE BufSize DUP(? ), 0, 0
stdInHandle HANDLE ?
bytesRead DWORD ?

.code

;Writes welcome message and invokes SetMoveSize if F1 is pressed

Welcome PROC,
       moveSize : PTR WORD

INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov consoleHandle, eax

INVOKE WriteConsole, consoleHandle, ADDR messageHello, messageSizeHello, ADDR
bytesWritten, 0

Hello : mov   eax, 10
        call Delay
        call ReadKey
        cmp ax, 1C0Dh
        je Finish
        cmp ax, 3B00h
        jne Hello
        INVOKE SetMoveSize, moveSize

Finish : ret
Welcome ENDP
```

```
SetMoveSize PROC,

	moveSize : PTR WORD


call Clrscr

INVOKE WriteConsole, consoleHandle, ADDR messageMoveSize, messageMoveSizeSize, ADDR
bytesWritten, 0

INVOKE GetStdHandle, STD_INPUT_HANDLE

mov stdInHandle, eax

INVOKE ReadConsole, stdInHandle, ADDR buffer, BufSize, ADDR bytesRead, 0

mov edx, OFFSET buffer

mov ecx, bufSize

call ParseInteger32

cmp ax, 30h

ja Finish

mov ebx, moveSize

mov WORD PTR [ebx], ax


Finish : ret

SetMoveSize ENDP
```

SetCursorPositionProcedures.asm

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
INCLUDE Definitions.inc

.data
outHandle DWORD ?
xyPos COORD <0,0>

.code

SetCursorPos PROC,
	ScreenLength : WORD, ScreenHeight : WORD

INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov outHandle, eax

SetPos : INVOKE SetConsoleCursorPosition, outHandle, xyPos
	  mov eax, 10
	  call Delay
	  call ReadKey
	  jz SetPos
	  cmp ax, 4B00h
	  jz   LeftKey
	  cmp ax, 4D00h
	  jz   RightKey
	  cmp ax, 4800h
	  jz   UpKey
	  cmp ax, 5000h
	  jz   DownKey
	  cmp ax, 5200h
	  jz Finish
	  cmp ax, 1C0Dh
	  jz endProgram
	  jmp SetPos

	LeftKey: INVOKE GoLeft, ScreenLength
	jmp SetPos

	RightKey: INVOKE GoRight, ScreenLength
	jmp SetPos

	UpKey: INVOKE GoUp, ScreenHeight
	jmp SetPos

	DownKey: INVOKE GoDown, ScreenHeight
	jmp SetPos

	endProgram : INVOKE ExitProcess, 0

Finish: ret
SetCursorPos ENDP
```

```
GoLeft PROC,
       ScreenLength : WORD

sub WORD PTR xyPos[0], 1
mov bx, screenLength
cmp WORD PTR xyPos[0], bx
ja LeftException
jmp Finish
LeftException : add WORD PTR xyPos[0], 1
jmp Finish

Finish: ret
GoLeft ENDP

GoRight PROC,
       ScreenLength : WORD

add WORD PTR xyPos[0], 1
mov bx, screenLength
cmp WORD PTR xyPos[0], bx
ja RightException
jmp Finish
RightException : sub WORD PTR xyPos[0], 1
jmp Finish

Finish: ret
GoRight ENDP

GoUp PROC,
       ScreenHeight : WORD

sub WORD PTR xyPos[2], 1
mov bx, screenHeight
cmp WORD PTR xyPos[2], bx
ja UpException
jmp Finish
UpException : add WORD PTR xyPos[2], 1
jmp Finish

Finish: ret
GoUp ENDP

GoDown PROC,
       ScreenHeight : WORD

add WORD PTR xyPos[2], 1
mov bx, screenHeight
cmp WORD PTR xyPos[2], bx
ja DownException
jmp Finish
DownException : sub WORD PTR xyPos[2], 1
jmp Finish

Finish: ret
GoDown ENDP

END
```

DrawProcedures.asm

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc
INCLUDE Definitions.inc

.data
outHandle DWORD ?
xyPos COORD <0, 0>
consoleInfo CONSOLE_SCREEN_BUFFER_INFO <>

.code

DrawVerticalLine PROC,
       ScreenHeight : WORD

mov WORD PTR xyPos[2], 0
mov bx, screenHeight

    L : INVOKE SetConsoleCursorPosition, outHandle, xyPos
        mov  al, 0DBh
        call WriteChar
        dec bx
        cmp bx, 0
        jz Finish
        add WORD PTR xyPos[2], 1
        jmp L

Finish : ret
DrawVerticalLine ENDP

DrawHorizontalLine PROC,
       ScreenLength : WORD

mov WORD PTR xyPos[0], 0
mov bx, screenLength

    L : INVOKE SetConsoleCursorPosition, outHandle, xyPos
       mov  al, 0DBh
       call WriteChar
       dec bx
       cmp bx, 0
       jz Finish
       add WORD PTR xyPos[0], 1
       jmp    L

Finish : ret
DrawHorizontalLine ENDP
```

```
Draw PROC,
       moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov outHandle, eax

INVOKE GetConsoleScreenBufferInfo, outHandle, ADDR consoleInfo
mov ax, WORD PTR consoleInfo[4]
mov WORD PTR xyPos[0], ax
mov ax, WORD PTR consoleInfo[6]
mov WORD PTR xyPos[2], ax

   L : mov eax, 10
       call Delay
       call   ReadKey
       jz     L
       cmp ax, 4B00h
       jz     LeftKey
       cmp ax, 4D00h
       jz RightKey
       cmp ax, 4800h
       jz UpKey
       cmp ax, 5000h
       jz DownKey
       cmp ax, 5200h
       jz Finish
       cmp ax, 1C0Dh
       jz endProgram
       jmp L

LeftKey : INVOKE Left, moveSize, ScreenLength, ScreenHeight
          jmp L

RightKey : INVOKE Right, moveSize, ScreenLength, ScreenHeight

           jmp L

UpKey : INVOKE Up, moveSize, ScreenLength, ScreenHeight

        jmp L

DownKey : INVOKE Down, moveSize, ScreenLength, ScreenHeight

          jmp L

endProgram : INVOKE ExitProcess, 0


Finish: ret

Draw ENDP
```

```
Left PROC,
	moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

mov dx, moveSize
sub WORD PTR xyPos[0], dx
mov bx, screenLength
cmp WORD PTR xyPos[0], bx
ja LeftException
jmp DrawVertical
LeftException : add WORD PTR xyPos[0], dx
jmp DrawVertical

DrawVertical : INVOKE DrawVerticalLine, ScreenHeight

ret
Left ENDP


Right PROC,
	moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

mov dx, moveSize
add WORD PTR xyPos[0], dx
mov bx, screenLength
cmp WORD PTR xyPos[0], bx
ja RightException
jmp DrawVertical
RightException : sub WORD PTR xyPos[0], dx
jmp DrawVertical

DrawVertical : INVOKE DrawVerticalLine, ScreenHeight

ret
Right ENDP
```

```
Up PROC,
	moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

mov dx, moveSize
sub WORD PTR xyPos[2], dx
mov bx, screenHeight
cmp WORD PTR xyPos[2], bx
ja UpException
jmp DrawHorizontal
UpException : add WORD PTR xyPos[2], dx
jmp DrawHorizontal

DrawHorizontal : INVOKE DrawHorizontalLine, ScreenLength

ret
Up ENDP



Down PROC,
	moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

mov dx, moveSize
add WORD PTR xyPos[2], dx
mov bx, screenHeight
cmp WORD PTR xyPos[2], bx
ja DownException
jmp DrawHorizontal
DownException : sub WORD PTR xyPos[2], dx
jmp DrawHorizontal

DrawHorizontal : INVOKE DrawHorizontalLine, ScreenLength

ret
Down ENDP

END
```

Definitions.inc

```
SetMoveSize PROTO moveSize : PTR WORD
Welcome PROTO moveSize : PTR WORD

SetCursorPos PROTO ScreenLength : WORD, ScreenHeight : WORD

GoUp PROTO ScreenHeight : WORD
GoDown PROTO ScreenHeight : WORD
GoRight PROTO ScreenLength : WORD
GoLeft PROTO ScreenLength : WORD

Draw PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

Up PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Down PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Left PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD
Right PROTO moveSize : WORD, ScreenLength : WORD, ScreenHeight : WORD

DrawVerticalLine PROTO ScreenHeight : WORD
DrawHorizontalLine PROTO ScreenLength : WORD

ProgramLoop PROTO
```

Diskusija priloženog rešenja

Cilj je bio doći do rešenja koje je što manje prostorne i vremenske kompleksnosti sa akcentom na tome da se dobije kôd koji je jasan i pregledan i koji lako može da se modifikuje i nadogradi. Obzirom da se radi o programu koji ne zauzima mnogo mesta u memoriji, tamo gde je trebalo izabrati između manje prostorne kompleksnosti i preglednosti, izabrana je druga opcija.

Tako, na primer, xyPos promenljiva postoji i u fajlu DrawProcedures.asm i SetCursorPosition.asm. U prvoj verziji je ta promenljiva bila u fajlu Main.asm i ovim procedurama bila prosleđivana preko pokazivača. Pa je zato, nakon crtanja, kursor bio na prvoj sledećoj slobodnoj poziciji nakon poslednje popunjene ćelije. Mnogo prirodnije rešenje je da se kursor vrati na početno mesto. Ovo je dovelo do toga da svakako treba izdvojiti mesto u memoriji za čuvanje te vrednosti. Nakon toga je bilo jasno da je bolje rešenje da se postavi xyPos promenljiva u oba fajla, ozbirom da će se ionako zauzeti mesto u memoriji, a ovaj način rezultuje daleko preglednijim kodom.

Takođe, treba primetiti da ne postoje lokalne promenljive. Naime, one bi, ukoliko postoje bile definisane u sklopu glavnih procedura .asm fajlova (Draw, SetCursorPostion), a kako se ostale procedure koriste u isključivo u okviru glavnih, nije bilo razlike između definisanja promenljivih u okviru fajla ili kao lokalnih u sklopu glavnih procedura. Opet, zbog preglednosti je izabrano da budu u .data segmentu.

Iz istog razloga se parametri procedura se prosleđuju memorijski ili preko pokazivača (ukoliko se menja vrednost), umesto preko steka ili registara. Nijedna funkcija ne vraća povratnu vrednost preko registara, jer tako postoji mnogo veća opasnost da ponašanje programa ne bude uvek definisano, a da se to ne primeti prilikom testiranja.

Jedna od stvari koja može da se poboljša jeste da se doda obaveštenje korisniku da je izabrao moveSize izvan opsega i procedura pozove ponovo. Naravno, u tom slučaju bi trebalo voditi računa da program poziva proceduru najviše određen broj puta (da se ne bi vrteo u beskonačnoj petlji ukoliko korisnik uporno upisuje pogrešnu vrednost).

Drugi problem jeste što procedure računaju na to da im se prosledi screenLength, screenHeight, što ostavlja prostora da se pozovu sa pogrešnim vrednostima, a da ne postoji način da se to detektuje. Obzirom da se veličina zadaje u main proceduri, a da ona komunicira sa ovim funkcijama ova provera nije bila neophodna. Međutim, ukoliko bi one bile deo neke biblioteke bezbedije bi bilo izbaciti ove argumente i iskoristiti GetConsoleScreenBufferInfo za određivanje tih vrednosti. Smatram da bi kontekstu ovog programa, to rešenje bilo bespotrebno komplikovanje.

Savić Jovana
2013/0243