UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku
Računarska elektronika

Predmetni professor: prof. dr Milan Prokin
Asistent na predmetu: as. dr Aleksandra Lekić

Projekat 20, 2017/18
Grupa 7

Nikola Pejić 2014/0656
Vuk Popović 2014/0277

Datum:
27.06.2018

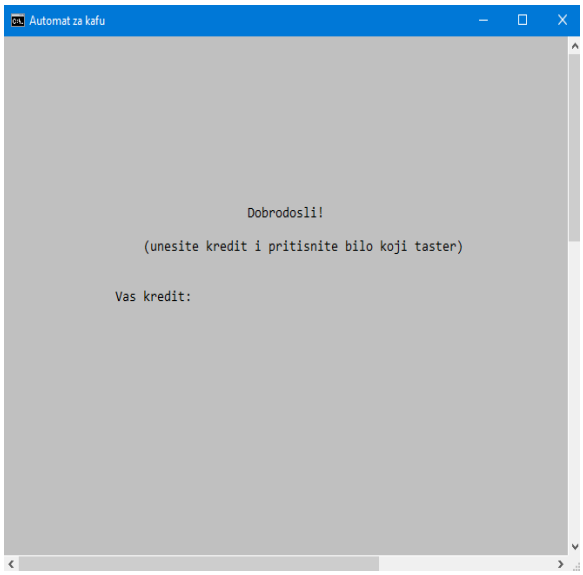
Tekst projekta

Projekat 20 - Automat za kafu

Ideja je da se isprojektuje automat za kafu kakav se viđa na hodnicima fakulteta. Automat ima sledeće funkcije: prikaz raspoloživih toplih napitaka, nivo šećera, trenutni iznos kredita i, ukoliko postoji, kusur. Pomoću broja se bira željena vrsta napitka a ukoliko se unesu broj i znak plus ili minus(u kombinaciji 1+, 1-), naručuje se više tj manje napitaka iste vrste. Može se naručiti više napitaka od jednom. Priprema svakog napitka traje 2s. Šećer se podešava unosom znaka + ili -. Voditi računa da se ni jedan brojač ne može prevrteti. Ukoliko nema dovoljno kredita, ispisati poruku, i stornirati porudzbinu. Na odeljku kusur ispisivati ogovarajuću cifru nakon isteka vremena za pripremu napitaka.

Realizacija projekta

Realizacija projekta je osmišljena tako da se na početku ispiše poruka dobrodošlice i da se pita korisnik da unese količinu kredita koju poseduje. Broj koji korisnik unese se prihvata kao string i prevodi se u integer posebnim delom koda koji ima taj zadatak. Taj podatak se smešta u promenljivu ***credits***. Zatim se na ekranu posebnim procedurama iscrtavaju kvadrati ili odeljci koji odvajaju ponudu proizvoda od prostora na ekranu na kome se ispisuje količina preostalih kredita, kao i od dela ekrana koji je rezervisan za ispisivanje kusura i poručenih proizvoda i računa. Kusur će se ispisati na samom kraju porudžbine, kada korisnik nakon poručenih proizvoda, pritisne ***ENTER***. Korisnik u svakom momentu ima uvid u količinu preostalih kredita, kao i u to koje je proizvode i u kojoj količini poručio. Tasterima ***+*** i ***-***, korisnik podešava količinu šećera koju želi, dok brojevima poručuje napitak pod brojem koji je pritisnuo. U kombinaciji ***broj*** i ***+/-*** korisnik dodaje, odnosno smanjuje količinu poručenih napitaka.



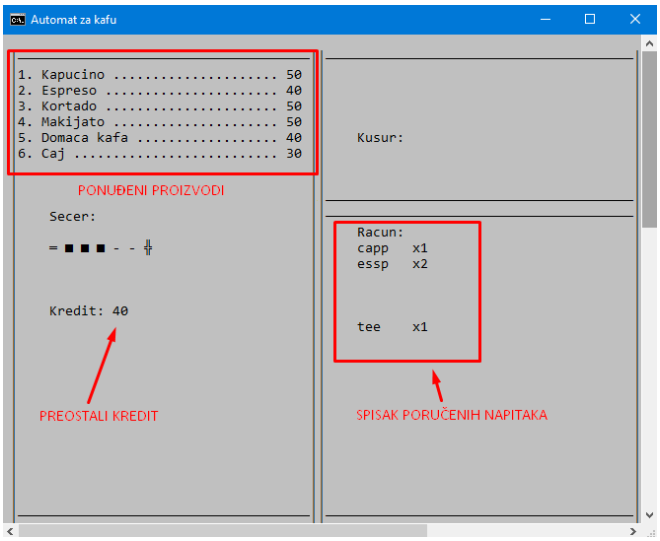
Slika 1. Poruka dobrodošlice

Pritiskom na ***ENTER*** na ekranu se ispisuje ukupan račun i kusur i time se završava rad. Pritiskom na ***ESC*** na ekranu se ispisuje odjavna poruka.

Na samom početku koda uključujemo biblioteke ***Irvine.inc*** i ***Macros.inc***, posle kojih se navode konstante u programu koje se koriste kao koordinate prilikom iscrtavanja naslova kao I iscrtavanja pravougaonika koji odvajaju odeljke na ekranu:

```
endl EQU <0dh,0ah>
;----------‘’’’’--------------------CONSTANTS----------------------------------------
windX = 80
windY = 30
sugarDiam = 4 ;black diamond
sugarMin = 45 ;minus
buffSize = 80
;----------------------------RECTANGLES & TITLES POSITIONS----------------------------
	productListX = 1
	productListY = 1
		sugarX = productListX+5
		sugarY = productListY+10
			sugarLevelX = sugarX
			sugarLevelY = sugarY+2
		creditsTitleX = sugarX
		creditsTitleY = sugarY+6
	billInfoX = 40
	billInfoY = 11
		billTitleX = billInfoX+5
		billTitleY = billInfoY+1
		billX = billTitleX
		billY = billTitleY+1
			cappBillX = billX
			esspBillX = billX
			cortBillX = billX
			machBillX = billX
			coffBillX = billX
			teeBillX = billX
	changeInfoX = 40
	changeInfoY = 1
		changeTitleX = changeInfoX+5
		changeTitleY = changeInfoY+5
```

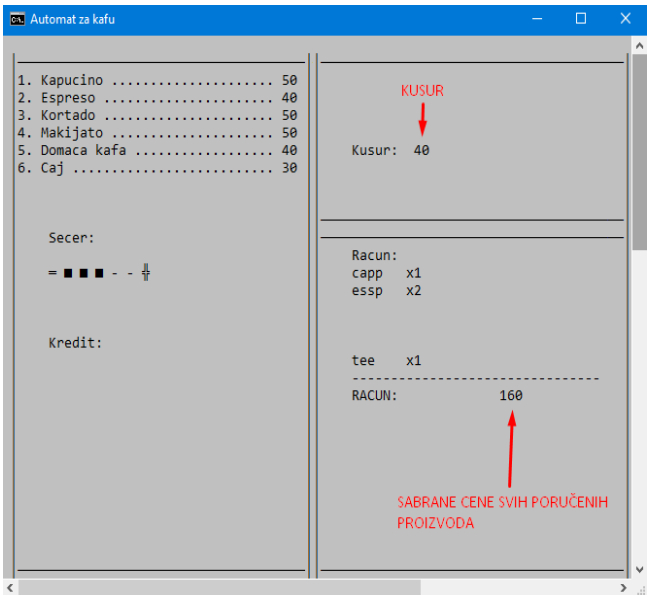
Nakon toga su u okviru **.data** segmenta navedene poruke koje će se ispisivati na ekranu, kao I poruke obaveštenja I slično. Nakon njih su navedene promenljive koje će se koristiti kao cene proizvoda, količina poručenih proizvoda, račun, kusur, količina šećera I druge. I na kraju **.data** segmenta se nalaze promeljive koje nam služe za ispisivanje I čitanje konzole.



Slika 2. Poručivanje napitaka

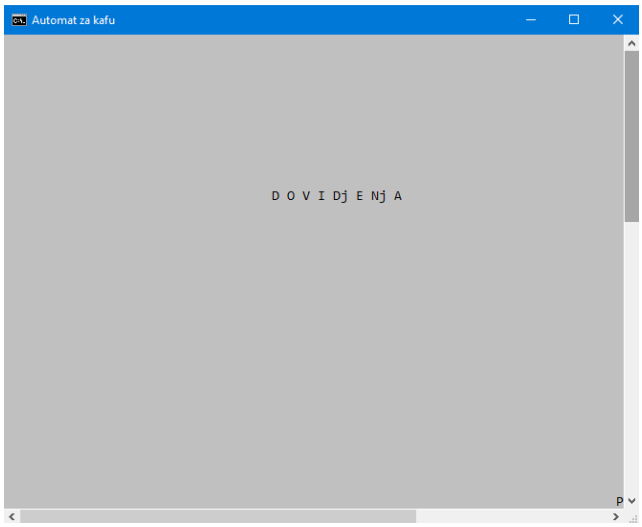
U **.code** segmentu se nalazi glavna procedura, koja se zbog velikog broja ponavljanja istog koda poziva na druge, pomoćne procedure:

- **drawRect** procedura prihvata početne I krajnje coordinate pomoću kojih iscrtava pravougaonike na ekranu.
- **print** procedura koja prihvata mesto na ekranu na kom treba da se ispiše odgovarajuća poruka u vidu početnih i krajnjih koordinata, kao I samu poruku koju ispisuje.



Slika 3. Pritiskom na ENTER ispisuje se racun i kusur

- **detect** procedura koja je zadužena da detektuje koji je taster pritisnut I u zavisnosti od toga skače na određeni deo procedure. Deo procedure je I kod koji završava porudžbinu ukoliko je pritisnut ENTER, kao I deo koji je zadužen za prekidanje programa I ispisivanja odjavne poruke ukoliko je pritisnut **ESC.** Isto tako, ova procedura poseduje kod koji reguliše nivo šećera, ukoliko korisnik pritisne **+/-**. U okviru ove procedure se pozivaju dodatne procedure, kao što su:
  - **correction** procedura koja u zavisnosti od korisnikovog izbora koriguje listu naručenih napitaka(**printBill**), njihovu količinu ukoliko su već poručeni(**updateAmount**), kao I preostali kredit, kusur I račun(**updateCredits**). Procedura ispisuje I obaveštenje ukoliko korisnik nema dovoljno kredita za proizvod koji je poručio.



Slike 3. Odjavna poruka

U **main** proceduri su, pored delova koda koji se pozivaju na već pomenute procedure, deo koda koji prevodi *string* koji korisnik unese kao kredit u *integer* tako sto cifru najmanje težine množi jedinicom, sledeću cifru desetica množi brojem deset, narednu cifru stotina množi brojem sto I tako dalje nastavlja do cifre najveće težine. Sve ove rezultate sabira I tako dobija iznos kredita.

U delu koda koji obavlja interakciju sa korisnikom, pomoću funkcije **Delay** i uslovnog skoka ukoliko nije pritisnuto nije dugme na tastaturi, korisnik se čeka da odabere napitak, nakon čega se poziva funkcija **detect** koja je već opisana.

Kod projekta (izosavljeni su komentari koji se nalaze u .asm fajlu radi bolje čitljivosti)

```
INCLUDE Irvine32.inc
INCLUDE Macros.inc


endl EQU <0dh,0ah>
; --------------------------------CONSTANTS---------------------------------------------
windX = 80
windY = 30
sugarDiam = 4
sugarMin = 45
buffSize = 80
;--------------------------RECTANGLES & TITLES POSITIONS-------------------------------
	productListX = 1
	productListY = 1
		sugarX = productListX+5
		sugarY = productListY+10
			sugarLevelX = sugarX
			sugarLevelY = sugarY+2
		creditsTitleX = sugarX
		creditsTitleY = sugarY+6
	billInfoX = 40
	billInfoY = 11
		billTitleX = billInfoX+5
		billTitleY = billInfoY+1
		billX = billTitleX
		billY = billTitleY+1
			cappBillX = billX
			esspBillX = billX
			cortBillX = billX
			machBillX = billX
			coffBillX = billX
			teeBillX = billX
	changeInfoX = 40
	changeInfoY = 1
		changeTitleX = changeInfoX+5
		changeTitleY = changeInfoY+5

.data
;-----------------------------------TITLES-------------------------------------------
	titlemsg BYTE "Automat za kafu", endl, 0
	prodTitle     BYTE "1. Kapucino ..................... 50", endl
			BYTE "  2. Espreso ..................... 40", endl
			BYTE "  3. Kortado ..................... 50", endl
			BYTE "  4. Makijato .................... 50", endl
			BYTE "  5. Domaca kafa ................. 40", endl
			BYTE "  6. Caj ......................... 30", endl, 0
	sugarTitle    BYTE "Secer:", endl, 0
	billTitle     BYTE "Racun:", endl, 0
	creditsTitle BYTE "Kredit:", endl, 0
	changeTitle BYTE "Kusur:", endl, 0
	welcomeTitle  BYTE "Dobrodosli!", endl, endl
		BYTE "            (unesite kredit i pritisnite bilo koji taster)", endl, 0
	goodbyeMsg    BYTE "D O V I Dj E Nj A", endl, 0
	infoCaption BYTE "INFORMATION", endl, 0
	infoMsg       BYTE "Nemate dovoljno kredita.", endl, 0
```

```
	;-------------------------------------------------------------------------------

	welcomePos COORD <35,10>
	exitPos COORD <80,30>
	sugarLevel BYTE 3
	credits DWORD 0
	change DWORD ?
	bill DWORD 0
	cappAmount BYTE 0
	esspAmount BYTE 0
	cortAmount BYTE 0
	machAmount BYTE 0
	coffAmount BYTE 0
	teeAmount BYTE 0
	cappPrice DWORD 50 ; =50
	esspPrice DWORD 40 ; =40
	cortPrice DWORD 50 ; =50
	machPrice DWORD 50 ; =50
	coffPrice DWORD 40 ; =40
	teePrice DWORD 30    ; =30


;-----------------------------------CONSOLE CONTROLS-------------------------------------
	outHandle HANDLE ?
	consoleHandle HANDLE 0
	scrSize COORD <120,80>
	windowRect SMALL_RECT <0,0,windX,windY> ; <left,right,top,bottom>
	consoleInfo CONSOLE_SCREEN_BUFFER_INFO <>
	cursorInfo CONSOLE_CURSOR_INFO <>

	buffer BYTE buffSize DUP(?)
	inHandle HANDLE ?
	bytesRead DWORD ?

;================================================================================================
;================================================================================================

.code
drawRect PROC c ;--------------DRAWING RECTANGLE PROCEDURE-----------------------------
		LOCAL startX:WORD, startY:WORD, endX:WORD, endY:WORD
		mov ax, [ebp+8]
		mov endY, ax
		mov ax, [ebp+10]
		mov endX, ax
		mov ax, [ebp+12]
		mov startY, ax
		mov ax, [ebp+14]
		mov startX, ax
		;---------------------DRAWING TOP LINE-----------------------------------
		mov dl, BYTE PTR startX
		mov dh, BYTE PTR startY
		mov al, 196

		.REPEAT
			call Gotoxy
			call Writechar
			inc dl
		.UNTIL dl > BYTE PTR (endX)
```

```
		;-----------------------DRAWING BOTTOM LINE-----------------------------
		mov dl, BYTE PTR startX
		mov dh, BYTE PTR endY
		mov al, 196

		.REPEAT
			call Gotoxy
			call Writechar
			inc dl
		.UNTIL dl > BYTE PTR endX
		;-----------------------DRAWING SIDE LINES-----------------------------
		mov dl, BYTE PTR startX
		mov dh, BYTE PTR startY
		mov al, 179

		.REPEAT
			call Gotoxy
			call Writechar
			inc dh
		.UNTIL dh > BYTE PTR endY

		mov dl, BYTE PTR endX
		mov dh, BYTE PTR startY
		mov al, 179

		.REPEAT
			call Gotoxy
			call Writechar
			inc dh
		.UNTIL dh > BYTE PTR endY

		ret
drawRect ENDP

print PROC c------------------PRINTING TEXT PROCEDURE---------------------------------
	LOCAL startX:BYTE, startY:BYTE, msg:DWORD
	mov al, [ebp+8]
	mov startY, al
	mov al, [ebp+12]
	mov startX, al
	mov eax, [ebp+16]
	mov msg, eax

	mov dl, startX
	mov dh, startY
	call Gotoxy
	mov edx, msg
	call WriteString

	ret

print ENDP

detect PROC c;------------------KEY DETECTION PROCEDURE---------------------------------
	LOCAL key:BYTE
	mov dl, [ebp+8]
	mov key, dl
	cmp key, 31h
```

```
        je pressed1
        cmp key, 32h
        je pressed2
        cmp key, 33h
        je pressed3
        cmp key, 34h
        je pressed4
        cmp key, 35h
        je pressed5
        cmp key, 36h
        je pressed6
        cmp key, 2bh
        je pressedPlus
        cmp key, 2dh
        je pressedMinus
        cmp key, 0dh
        je pressedEnter
        jmp finish

pressedEnter:
        mov dl, finalX
        mov dh, finalY
        call Gotoxy
        mWrite <"--------------------------------">
        inc dh
        call Gotoxy
        mWrite <"RACUN:             ">
        mov eax, bill
        call WriteDec

        mov dl, changeTitleX+8
        mov dh, changeTitleY
        call Gotoxy
        mWrite <"          ">
        call Gotoxy
        mov eax, change
        call WriteDec
        mov dl, creditsTitleX+8
        mov dh, creditsTitleY
        call Gotoxy
        mWrite <"          ">
escape:
        mov eax, 10
        call Delay
        call ReadKey
        jz escape

        cmp al, 1bh
        jne escape

        call Clrscr
        INVOKE SetConsoleCursorPosition, outHandle, welcomePos
        mov edx, OFFSET goodbyeMsg
        call WriteString
        mov eax, 300
        call Delay
        INVOKE SetConsoleCursorPosition, outHandle, exitPos
```

```
        exit


pressedPlus:
        cmp sugarLevel, 5
        je finish

        inc sugarLevel

        mov dl, sugarLevelX
        mov dh, sugarLevelY
        mov al, sugarLevel
        shl al, 1
        add dl, al
        call Gotoxy
        mWrite <" ">
        call Gotoxy
        xor eax, eax
        mov al, 254
        call WriteChar
        jmp finish
pressedMinus:
        cmp sugarLevel, 0
        je finish

        mov dl, sugarLevelX
        mov dh, sugarLevelY
        mov al, sugarLevel
        shl al, 1
        add dl, al
        call Gotoxy
        mWrite <" ">
        call Gotoxy
        xor eax, eax
        mov al, 45
        call WriteChar

        dec sugarLevel
        jmp finish

pressed1:
        xor eax, eax
        mov al, cappAmount
        mov dl, key
        push edx
        push cappPrice
        push eax
        call correction
        add esp, 4
        mov cappAmount, al
        jmp finish
pressed2:
        xor eax, eax
        mov al, esspAmount
        mov dl, key
        push edx
        push esspPrice
        push eax
```

```
        call correction
        add esp, 4
        mov esspAmount, al
        jmp finish
pressed3:
        xor eax, eax
        mov al, cortAmount
        mov dl, key
        push edx
        push cortPrice
        push eax
        call correction
        add esp, 4
        mov cortAmount, al
        jmp finish
pressed4:
        xor eax, eax
        mov al, machAmount
        mov dl, key
        push edx
        push machPrice
        push eax
        call correction
        add esp, 4
        mov machAmount, al
        jmp finish
pressed5:
        xor eax, eax
        mov al, coffAmount
        mov dl, key
        push edx
        push coffPrice
        push eax
        call correction
        add esp, 4
        mov coffAmount, al
        jmp finish
pressed6:
        xor eax, eax
        mov al, teeAmount
        mov dl, key
        push edx
        push teePrice
        push eax
        call correction
        add esp, 4
        mov teeAmount, al

finish:
        ret

detect ENDP

correction PROC;--------------------CORRECTION PROCEDURE--------------------------------
        LOCAL amount:BYTE, price: DWORD, key:BYTE
        mov al, [ebp+8]
        mov amount, al
        mov eax, [ebp+12]
```

```
        mov price, eax
        mov al, [ebp+16]
        mov key, al

        mov eax, 200
        call Delay
        call ReadKey
        cmp al, 2bh
        je incAmount
        cmp al, 2dh
        je decAmount
        cmp amount,0
        jne finish
pressedPlusOnBeg:
        mov eax, credits
        mov edx, price
        cmp eax, edx
        jb warning
        sub eax, edx
        add bill, edx
        mov credits, eax
        mov change, eax
        push eax
        call updateCredits
        add esp, 4

        inc amount

        xor eax, eax
        xor edx, edx
        mov al, amount
        mov dl, key
        push eax
        push edx
        call printBill
        add esp, 8
        jmp finish
incAmount:
        cmp amount, 0
        je pressedPlusOnBeg
        mov eax, credits
        mov edx, price
        cmp eax, edx
        jb warning
        sub eax, edx
        add bill, edx
        mov credits, eax
        mov change, eax
        push eax
        call updateCredits
        add esp, 4

        inc amount

        xor eax, eax
        mov al, amount
        mov dl, key
        push eax
```

```
        push edx
        call updateAmount
        add esp, 8
        jmp finish
decAmount:
        cmp amount, 0
        je finish
        mov eax, credits
        mov edx, price
        add eax, edx

        sub bill, edx
        mov credits, eax
        mov change, eax
        push eax
        call updateCredits
        add esp, 4

        dec amount

        xor eax, eax
        xor edx, edx
        mov al, amount
        mov dl, key
        push eax
        push edx
        call updateAmount
        add esp, 8
        jmp finish


warning:
        INVOKE MessageBox, NULL, ADDR infoMsg, ADDR infoCaption, MB_OK

finish:
        mov eax, DWORD PTR amount
        ret
correction ENDP

updateAmount PROC c, key:BYTE, amount:BYTE ;----REFRESHING SCREEN PROCEDURE-------------
        cmp key, 31h
        je update1
        cmp key, 32h
        je update2
        cmp key, 33h
        je update3
        cmp key, 34h
        je update4
        cmp key, 35h
        je update5
        cmp key, 36h
        je update6
        jmp finish

update1:
        mov dl, cappBillX+8
        mov dh, cappBillY
        call Gotoxy
```

```
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish
update2:
        mov dl, esspBillX+8
        mov dh, esspBillY
        call Gotoxy
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish
update3:
        mov dl, cortBillX+8
        mov dh, cortBillY
        call Gotoxy
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish
update4:
        mov dl, machBillX+8
        mov dh, machBillY
        call Gotoxy
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish
update5:
        mov dl, coffBillX+8
        mov dh, coffBillY
        call Gotoxy
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish
update6:
        mov dl, teeBillX+8
        mov dh, teeBillY
        call Gotoxy
        mWrite <"   ">
        call Gotoxy
        xor eax, eax
        mov al, amount
        call WriteDec
        jmp finish

finish:
```

```
        ret
updateAmount ENDP

printBill PROC c, key:BYTE, amount:BYTE ;--------PRINTING BILL ON THE SCREEN------------
        cmp key, 31h
        je print1
        cmp key, 32h
        je print2
        cmp key, 33h
        je print3
        cmp key, 34h
        je print4
        cmp key, 35h
        je print5
        cmp key, 36h
        je print6
        jmp finish

print1:
        cappBillY = billY
        mov dl, cappBillX
        mov dh, cappBillY
        xor eax, eax
        mov al, amount
        call Gotoxy
        mWrite <"capp   x">
        call WriteDec
        billY = billY+1
        jmp finish
print2:
        esspBillY = billY
        mov dl, esspBillX
        mov dh, esspBillY
        xor eax, eax
        mov al, amount
        call Gotoxy
        mWrite <"essp   x">
        call WriteDec
        billY = billY+1
        jmp finish
print3:
        cortBillY = billY
        mov dl, cortBillX
        mov dh, cortBillY
        xor eax, eax
        mov al, amount
        call Gotoxy
        mWrite <"cort   x">
        call WriteDec
        billY = billY+1
        jmp finish
print4:
        machBillY = billY
        mov dl, machBillX
        mov dh, machBillY
        xor eax, eax
        mov al, amount
        call Gotoxy
```

```
	mWrite <"mach   x">
	call WriteDec
	billY = billY+1
	jmp finish
print5:
	coffBillY = billY
	mov dl, coffBillX
	mov dh, coffBillY
	xor eax, eax
	mov al, amount
	call Gotoxy
	mWrite <"coff   x">
	call WriteDec
	billY = billY+1
	jmp finish
print6:
	teeBillY = billY
	mov dl, teeBillX
	mov dh, teeBillY
	xor eax, eax
	mov al, amount
	call Gotoxy
	mWrite <"tee    x">
	call WriteDec
	billY = billY+1
	jmp finish

	finalX = billX
	finalY = billY

finish:
	ret

printBill ENDP

updateCredits PROC c, newCredit:DWORD ;--------REFRESHING CREDITS PROCEDURE-------------
	mov dl, creditsTitleX+8
	mov dh, creditsTitleY
	mov eax, newCredit
	call Gotoxy
	mWrite <"         ">
	call Gotoxy
	call WriteDec

	ret

updateCredits ENDP

;=================================================================================================
;                                  MAIN PROCEDURE
;=================================================================================================
main PROC;
	INVOKE  GetStdHandle, STD_OUTPUT_HANDLE
	mov outHandle, eax
	INVOKE GetStdHandle, STD_INPUT_HANDLE
	mov inHandle, eax

	INVOKE GetConsoleCursorInfo, outHandle, ADDR cursorInfo
```

```
	mov cursorInfo.bVisible,0
	INVOKE SetConsoleCursorInfo, outHandle, ADDR cursorInfo
	INVOKE SetConsoleScreenBufferSize, outHandle, scrSize
	INVOKE SetConsoleWindowInfo, outHandle, TRUE, ADDR windowRect
	INVOKE SetConsoleTitle, ADDR titlemsg
	INVOKE GetConsoleScreenBufferInfo, outHandle, ADDR consoleInfo

	mov eax, black + (lightGray*16)
	call SetTextColor
	;------------------------------DRAWING TEXT-------------------------------------
	;ispisivanje poruke dobrodoslice na ekranu:
	call Clrscr
	INVOKE SetConsoleCursorPosition, outHandle, welcomePos
	mov edx, OFFSET welcomeTitle
	call WriteString
	mWrite <endl, endl, "              Vas kredit: ">

	;------------------------INPUT TO INTEGER CONVERSION-----------------------------
	INVOKE ReadConsole, inHandle, ADDR buffer, buffSize, ADDR bytesRead, 0
	xor edx, edx
	xor eax, eax
	xor esi, esi
	xor edi, edi
	xor ebx, ebx
	mov ecx, bytesRead
	sub ecx, 2
	mov esi, OFFSET buffer
convertstr2int:
	mov ebx, ecx
	xor eax, eax
	mov al, [esi]
	sub eax, 30h
	sub ebx, 1
	jz addition
	inc esi
calc:
	mov edi, 10
	mul edi
	dec ebx
	cmp ebx, 0
	jnz calc
addition:
	add credits, eax
	loop convertstr2int
	;-------------------------------------------------------------------------------
	call ClrScr

	;ispisivanje liste proizvoda:
	push OFFSET prodTitle
	push productListX+1
	push productListY+1
	call print
	add esp, 12

	;ispivanje racuna:
	push OFFSET billTitle
	push billTitleX
	push billTitleY
```

```
        call print
        add esp, 12

        ;ispisivanje secera:
        push OFFSET sugarTitle
        push sugarX
        push sugarY
        call print
        add esp, 12

        ;ispisivanje skale za secer:
        mov dl, sugarLevelX
        mov dh, sugarLevelY
        call Gotoxy
        mov al, 205
        call WriteChar
        inc dl
        inc dl
        xor ecx, ecx
        mov cl, sugarLevel
printSq:
        call Gotoxy
        mov al, 254
        call WriteChar
        inc dl
        inc dl
        loop printSq

        xor eax, eax
        mov eax, 5
        sub al, sugarLevel
        xor ecx, ecx
        mov cl, al
printMin:
        call Gotoxy
        mov al, 45
        call WriteChar
        inc dl
        inc dl
        loop printMin
        mov al, 206
        call Gotoxy
        call WriteChar


        ;ispisivanje kredita:
        push OFFSET creditsTitle
        push creditsTitleX
        push creditsTitleY
        call print
        add esp, 12
        mov eax, credits
        mov dl, creditsTitleX+8
        mov dh, creditsTitleY
        call Gotoxy
        call WriteDec
```

```
	;ispisivanje kusura:
	push OFFSET changeTitle
	push changeTitleX
	push changeTitleY
	call print
	add esp, 12
	mov eax, credits
	mov change, eax
	;------------------------------DRAWING RECTANGLES--------------------------------
	;prosledjivanje koordinata za iscrtavanje okvira za proizvode:
	push WORD PTR productListX
	push WORD PTR productListY

	push WORD PTR billInfoX-1
	push WORD PTR windY
	call drawRect
	add esp, 8

	;prosledjivanje koordinata za iscrtavanje okvira za informacije o racunu;
	push WORD PTR billInfoX
	push WORD PTR billInfoY
	push WORD PTR windX
	push WORD PTR 10
	call drawRect
	add esp, 8

	;prosledjivanje koordinata za iscrtavanje okvira za informacije o kusuru:
	push WORD PTR changeInfoX
	push WORD PTR changeInfoY
	push WORD PTR windX
	push WORD PTR windY
	call drawRect
	add esp, 8

	;----------------------INTERACTION WITH USER-----------------------------------
xor eax, eax
waiting:                                        ; cekamo na unos broja proizvoda
	mov eax, 10
	call Delay
	call ReadKey
	jz waiting

	push eax
	call detect
	add esp, 4
	jmp waiting


main ENDP
END main
```