Programowanie niskopoziomowe

# Uniwersytet Przyrodniczo-humanistyczny w Siedlcach

## Wydział Nauk Ścisłych

Instytut Informatyki

PROJEKT nr 6:

Program obliczający wyznacznik macierzy 3x3, argumenty pobierane z konsoli lub plików tekstowych (zaimplementować obie metody), wynik w konsoli.

# Dokumentacja

Spis treści

[Temat projektu](#_Toc439774622) 1

[Opis dostępnych funkcji w programie 2](#_Toc439774623)

[Przykładowe dane i wyniki 3](#_Toc439774624)

[Kod programu](#_Toc439774626) 5

OPIS DOSTĘPNYCH FUNKCJI W PROGRAMIE

Projekt ma na celu obliczanie wyznacznika macierzy 3x3. Dla ułatwienia osoby korzystającej z programu jest możliwość wyboru, czy dane będą bezpośrednio pobierane z konsoli, czy też zostaną wczytane z wcześniej utworzonego pliku (należy w nim umieścić dane do zadania w zakresie do 9999 dla jednej liczby – jest to spowodowane ilością wczytywanych danych).

Jeśli użytkownik nie dokona żadnego wyboru z dostępnych opcji, program zostanie zamknięty (inaczej, jeśli użytkownik poda inną liczbę niż z możliwego wyboru, który proponuje program). Program zakończy się dopiero, kiedy użytkownik poda inną liczbę niż wskazywane przez program, nie zrobi tego sam, czyli po uzyskaniu wyniku obliczeń możemy liczyć od nowa lub wczytać liczby z pliku i tak w pętli.

Podając kolejno liczby uwzględniania jest ich kolejność w macierzy (program sam informuje w którym miejscu w macierzy znajdzie się podana liczba).

Użyta została także procedura do wprowadzania danych ScanInt.

Utworzona macierz oczywiście zostaje wyświetlona w oknie konsoli w czytelny sposób.

Program działa na zasadzie skoków do odpowiednich etykiet zależnych od wyboru użytkownika, w jaki sposób chce wykonywać obliczenia. Pod etykietami zawierają się szeregi instrukcji obliczających/bądź wprowadzających dane, również kończących program.

Czytanie danych z pliku odbywa się za pomocą edycji utworzonego już wcześniej pliku tekstowego, bądź poprzez podanie ścieżki do pliku bezpośrednio w kodzie programu.

Do projektu dołączam także bibliotekę masm32.lib, z której to funkcje ułatwiły pisanie programu (wypisanie tekstu na ekranie).

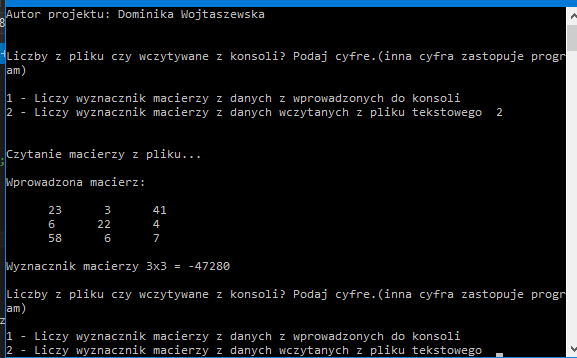
Program został napisany w środowisku Microsoft Visual Studio.

PRZYKŁADOWE DANE I WYNIKI

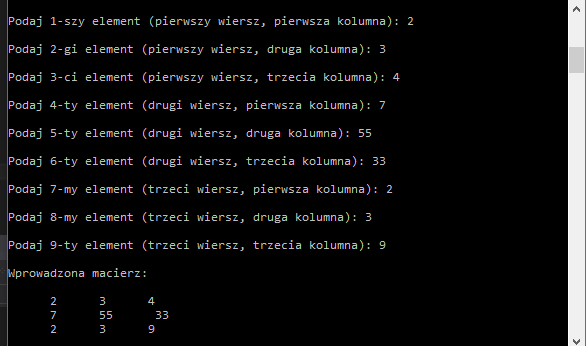
Przykład wczytania z pliku, po odpowiednim wyborze metody.

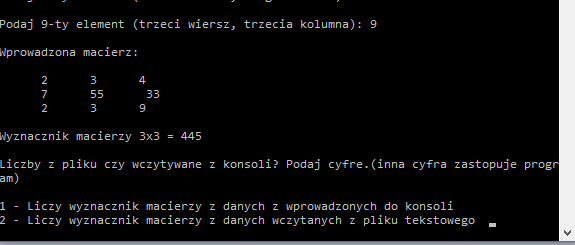
Jak widać po policzeniu wyznacznika możemy dalej liczyć wyznaczniki.

Do wczytania z pliku trzeba było wpisać cyfrę 2 przy możliwości wyboru.



Przy wyborze nr 1 elementy macierzy wpisujemy do konsoli, po kolei. Program informuje w którym miejscu w macierzy znajdzie się podana liczba.





I oczywiście wynik. Jeśli w programie użytkownik poda inną cyfrę/liczbę niż 1, bądź 2 program zostanie zamknięty. Nie stanie się to zaś dopóki użytkownik będzie podawał cyfry 1 i 2.

KOD PROGRAMU

|  |
| --- |
| .386  .MODEL flat, STDCALL  ;--- stale ---  STD\_INPUT\_HANDLE equ -10  STD\_OUTPUT\_HANDLE equ -11  GENERIC\_READ equ 80000000h  GENERIC\_WRITE equ 40000000h  OPEN\_EXISTING equ 3  ;--- funkcje API Win32 ---  ;--- z pliku user32.inc ---  CharToOemA PROTO :DWORD,:DWORD  ;--- z pliku kernel32.inc ---  GetStdHandle PROTO :DWORD  ReadConsoleA PROTO :DWORD, :DWORD, :DWORD, :DWORD, :DWORD  WriteConsoleA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD  ExitProcess PROTO :DWORD  lstrlenA PROTO :DWORD  StdOut PROTO :DWORD  StdIn PROTO :DWORD, :DWORD  wsprintfA PROTO C :VARARG  CreateFileA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD  CloseHandle PROTO :DWORD  ReadFile PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD  dwtoa PROTO dwValue:DWORD, lpBuffer:DWORD  \_DATA SEGMENT  bufor DB 128 dup(0)  rozmBufor DD 128  rout DD 0 ;faktyczna liczba wyprowadzonych znakĂłw  rinp DD 0 ;faktyczna liczba wprowadzonych znakĂłw  text byte "C:\Users\MaxSylverWolf\Documents\Visual Studio 2015\Projects\ProjektZAsma\ProjektZAsma\macierz.txt",0  hFile DD ?  numberofbytesread DD ?  liczbaDoOdczytu DD 4  hout DD ?  hinp DD ?  zm1 DD 4  zm2 DD 4  zm3 DD 4  zm4 DD 4  zm5 DD 4  zm6 DD 4  zm7 DD 4  zm8 DD 4  zm9 DD 4  witaj DB 0Dh,0Ah,"Autor projektu: Dominika Wojtaszewska",0  naglowek DB 0Dh,0Ah,"Liczby z pliku czy wczytywane z konsoli? Podaj cyfre.(inna cyfra zastopuje program)",0  naglowek1 DB 0Dh,0Ah,"1 - Liczy wyznacznik macierzy z danych z wprowadzonych do konsoli",0  naglowek2 DB 0Dh,0Ah,"2 - Liczy wyznacznik macierzy z danych wczytanych z pliku tekstowego ",0  zaproszenie DB 0Dh,0Ah,"Program liczy wyznacznik macierzy 3x3. Podaj elementy: ",0  podaj1 DB 0Dh,0Ah,"Podaj 1-szy element (pierwszy wiersz, pierwsza kolumna): ",0  podaj2 DB 0Dh,0Ah,"Podaj 2-gi element (pierwszy wiersz, druga kolumna): ",0  podaj3 DB 0Dh,0Ah,"Podaj 3-ci element (pierwszy wiersz, trzecia kolumna): ",0  podaj4 DB 0Dh,0Ah,"Podaj 4-ty element (drugi wiersz, pierwsza kolumna): ",0  podaj5 DB 0Dh,0Ah,"Podaj 5-ty element (drugi wiersz, druga kolumna): ",0  podaj6 DB 0Dh,0Ah,"Podaj 6-ty element (drugi wiersz, trzecia kolumna): ",0  podaj7 DB 0Dh,0Ah,"Podaj 7-my element (trzeci wiersz, pierwsza kolumna): ",0  podaj8 DB 0Dh,0Ah,"Podaj 8-my element (trzeci wiersz, druga kolumna): ",0  podaj9 DB 0Dh,0Ah,"Podaj 9-ty element (trzeci wiersz, trzecia kolumna): ",0  wynikProg DB 0Dh,0Ah,"Wyznacznik macierzy 3x3 = %ld",0  ALIGN 4  rozmWynikProg DD $ - wynikProg  macierz DB 0Dh,0Ah,"Wprowadzona macierz:",0  spacja DB 0Dh,0Ah," %ld",0  ALIGN 4  rozmSpacja DD $ - spacja  spacja3 DB 0Dh,0Ah," ",0  spacjaBEZ DB " %ld",0  ALIGN 4  rozmSpacjaBEZ DD $ - spacjaBEZ  zaproszeniePlik DB 0Dh,0Ah,"Czytanie macierzy z pliku...",0  \_DATA ENDS  \_TEXT SEGMENT  main proc  invoke GetStdHandle, STD\_OUTPUT\_HANDLE  mov hout, **EAX**  invoke GetStdHandle, STD\_INPUT\_HANDLE  mov hinp, **EAX**  invoke StdOut,offset witaj  invoke StdOut,offset spacja3  starto:  invoke StdOut,offset spacja3  invoke StdOut,offset naglowek  invoke StdOut,offset spacja3  invoke StdOut,offset naglowek1  invoke StdOut,offset naglowek2  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie liczby  push OFFSET bufor  call ScanInt  cmp **EAX**,1  je zKonsoli  cmp **EAX**,2  je zPliku  jmp koniec  zKonsoli:  invoke StdOut,offset zaproszenie  invoke StdOut,offset spacja3  ;--- 1 liczba w macierzy  invoke StdOut,offset podaj1  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 1  push OFFSET bufor  call ScanInt  mov zm1, **EAX**  ;--- 2 liczba w macierzy  invoke StdOut,offset podaj2  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 2  push OFFSET bufor  call ScanInt  mov zm2, **EAX**  ;--- 3 liczba w macierzy  invoke StdOut,offset podaj3  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 3  push OFFSET bufor  call ScanInt  mov zm3, **EAX**  ;--- 4 liczba w macierzy  invoke StdOut,offset podaj4  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 4  push OFFSET bufor  call ScanInt  mov zm4, **EAX**  ;--- 5 liczba w macierzy  invoke StdOut,offset podaj5  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 5  push OFFSET bufor  call ScanInt  mov zm5, **EAX**  ;--- 6 liczba w macierzy  invoke StdOut,offset podaj6  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 6  push OFFSET bufor  call ScanInt  mov zm6, **EAX**  ;--- 7 liczba w macierzy  invoke StdOut,offset podaj7  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 7  push OFFSET bufor  call ScanInt  mov zm7, **EAX**  ;--- 8 liczba w macierzy  invoke StdOut,offset podaj8  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 8  push OFFSET bufor  call ScanInt  mov zm8, **EAX**  ;--- 9 liczba w macierzy  invoke StdOut,offset podaj9  push 0  push OFFSET rinp  push rozmBufor  push OFFSET bufor  push hinp  call ReadConsoleA  lea **EBX**,bufor  mov **EDI**,rinp  mov BYTE PTR [**EBX**+**EDI**-2],0  ;--- przeksztalcenie zmiennej 9  push OFFSET bufor  call ScanInt  mov zm9, **EAX**  ;--- macierz wyswietlenie  wyswietl:  invoke StdOut,offset macierz  invoke StdOut,offset spacja3  mov **EAX**,zm1  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;-----------------------------------------  mov **EAX**,zm2  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm3  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm4  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm5  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm6  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm7  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;-------------------------------------  mov **EAX**,zm8  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm9  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  ;--- policzenie wyznacznika podanej macierzy  mov **EAX**,zm3  mov **EBX**,zm5  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm7  mul **EBX**  xor **EDX**,**EDX**  push **EAX**  mov **EAX**,zm1  mov **EBX**,zm6  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm8  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  push **EAX**  mov **EAX**,zm2  mov **EBX**,zm4  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm9  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**    push **EAX**  mov **EAX**,zm1  mov **EBX**,zm5  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm9  mul **EBX**  xor **EDX**,**EDX**  push **EAX**  mov **EAX**,zm2  mov **EBX**,zm6  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm7  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  push **EAX**  mov **EAX**,zm3  mov **EBX**,zm4  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm8  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  pop **EBX**  sub **EAX**,**EBX**  push **EAX**  ;--- wyświetlenie wyznacznika macierzy  invoke StdOut,offset spacja3  pop **EAX**  push **EAX**  push OFFSET wynikProg  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  jmp starto  zPliku:  ;--- macierz z pliku  invoke StdOut,offset spacja3  invoke StdOut,offset zaproszeniePlik  invoke StdOut,offset spacja3  invoke CreateFileA, offset text, GENERIC\_READ, 0, 0, OPEN\_EXISTING, 0, 0  mov hFile, **eax**  xor **eax**,**eax**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm1, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm2, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm3, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm4, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm5, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm6, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm7, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm8, **EAX**  invoke ReadFile,hFile,OFFSET liczbaDoOdczytu,4,OFFSET numberofbytesread,0  xor **EAX**,**EAX**  push OFFSET liczbaDoOdczytu  call ScanInt  mov zm9, **EAX**    invoke CloseHandle,hFile  ;--- wyświetlenie macierzy  invoke StdOut,offset macierz  invoke StdOut,offset spacja3  mov **EAX**,zm1  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;-----------------------------------------  mov **EAX**,zm2  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm3  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm4  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm5  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm6  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm7  push **EAX**  push OFFSET spacja  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;-------------------------------------  mov **EAX**,zm8  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  mov **EAX**,zm9  push **EAX**  push OFFSET spacjaBEZ  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  ;--------------------------------------  ;--- policzenie wyznacznika podanej macierzy  mov **EAX**,zm3  mov **EBX**,zm5  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm7  mul **EBX**  xor **EDX**,**EDX**  push **EAX**  mov **EAX**,zm1  mov **EBX**,zm6  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm8  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  push **EAX**  mov **EAX**,zm2  mov **EBX**,zm4  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm9  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**    push **EAX**  mov **EAX**,zm1  mov **EBX**,zm5  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm9  mul **EBX**  xor **EDX**,**EDX**  push **EAX**  mov **EAX**,zm2  mov **EBX**,zm6  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm7  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  push **EAX**  mov **EAX**,zm3  mov **EBX**,zm4  mul **EBX**  xor **EDX**,**EDX**  mov **EBX**,zm8  mul **EBX**  xor **EDX**,**EDX**  pop **EBX**  add **EAX**,**EBX**  pop **EBX**  sub **EAX**,**EBX**  push **EAX**  ;--- wyświetlenie wyznacznika macierzy  invoke StdOut,offset spacja3  pop **EAX**  push **EAX**  push OFFSET wynikProg  push OFFSET bufor  call wsprintfA  add **ESP**,12  mov rinp,**EAX**  push 0  push OFFSET rout  push rinp  push OFFSET bufor  push hout  call WriteConsoleA  jmp starto  koniec:  invoke ExitProcess,0  main endp  ScanInt PROC  ;; funkcja ScanInt przekształca ciąg cyfr do liczby, którą jest zwracana przez EAX  ;; argument - zakończony zerem wiersz z cyframi  ;; rejestry: EBX - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w wierszu, EDI - tymczasowy  ;--- początek funkcji  push **EBP**  mov **EBP**, **ESP** ; wskaźnik stosu ESP przypisujemy do EBP  ;--- odkładanie na stos  push **EBX**  push **ECX**  push **EDX**  push **ESI**  push **EDI**  ;--- przygotowywanie cyklu  mov **EBX**, [**EBP**+8]  push **EBX**  call lstrlenA  mov **EDI**, **EAX** ;liczba znaków  mov **ECX**, **EAX** ;liczba powtórzeń = liczba znaków  xor **ESI**, **ESI** ; wyzerowanie ESI  xor **EDX**, **EDX** ; wyzerowanie EDX  xor **EAX**, **EAX** ; wyzerowanie EAX  mov **EBX**, [**EBP**+8] ; adres tekstu  ;--- cykl --------------------------  pocz:  cmp BYTE PTR [**EBX**+**ESI**], 0h ;porównanie z kodem \0  jne **@F**  jmp et4  @@:  cmp BYTE PTR [**EBX**+**ESI**], 0Dh ;porównanie z kodem CR  jne **@F**  jmp et4  @@:  cmp BYTE PTR [**EBX**+**ESI**], 0Ah ;porównanie z kodem LF  jne **@F**  jmp et4  @@:  cmp BYTE PTR [**EBX**+**ESI**], 02Dh ;porównanie z kodem -  jne **@F**  mov **EDX**, 1  jmp nast  @@:  cmp BYTE PTR [**EBX**+**ESI**], 030h ;porównanie z kodem 0  jae **@F**  jmp nast  @@:  cmp BYTE PTR [**EBX**+**ESI**], 039h ;porównanie z kodem 9  jbe **@F**  jmp nast  ;----  @@:  push **EDX** ; do EDX procesor może zapisać wynik mnożenia  mov **EDI**, 10  mul **EDI** ;mnożenie EAX \* EDI  mov **EDI**, **EAX** ; tymczasowo z EAX do EDI  xor **EAX**, **EAX** ;zerowani EAX  mov **AL**, BYTE PTR [**EBX**+**ESI**]  sub **AL**, 030h ; korekta: cyfra = kod znaku - kod 0  add **EAX**, **EDI** ; dodanie cyfry  pop **EDX**  nast:  inc **ESI**  loop pocz  ;--- wynik  or **EDX**, **EDX** ;analiza znacznika EDX  jz **@F**  neg **EAX**  @@:  et4:;--- zdejmowanie ze stosu  pop **EDI**  pop **ESI**  pop **EDX**  pop **ECX**  pop **EBX**  ;--- powrót  mov **ESP**, **EBP** ; przywracamy wskaźnik stosu ESP  pop **EBP**  ret 4  ScanInt ENDP  \_TEXT ENDS  END |