/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* INTERACTIVE MATRIX LANGUAGE \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

Jezyk macierzowy, podstawowa jednostka jest macierz;

Uwaga: tylko macierze dwuwymiarowe;

The fundamental data element with SAS/IML software is the matrix,

a two-dimensional (row × column) array of numeric or character values.;

\*/

\* W OnlineDocu: SAS Products/SAS IML;

\* Dwa tryby pracy: -tryb "wsadowy" ...;

**proc** **iml**;

a=**2**;

print a;

**quit**;



\*...oraz interaktywny:;

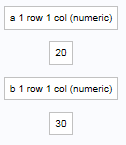
\*(do sesji interaktywnej wygodnie spowodowac wyswietlanie wszystkiego do Logu);

**proc** **iml**;

reset log print;

a=**20**;

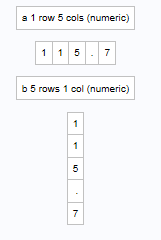
b=**30**;



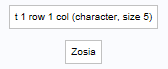
/\* definiowanie wektorów i macierzy \*/

a={**1** **1** **5** **.** **7**};

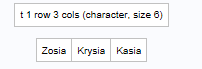
b={**1**,**1**,**5**,**.**,**7**};



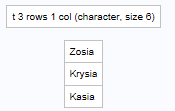
t='Zosia'; /\* gdyby nie pisac w ciapkach, szukalby macierzy Zosia \*/



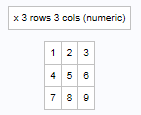
t={'Zosia' 'Krysia' 'Kasia'};



t={'Zosia', 'Krysia', 'Kasia'};

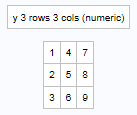


x={**1** **2** **3**, **4** **5** **6**, **7** **8** **9**};



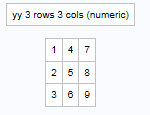
\*np. transpozycja;

y=x`;



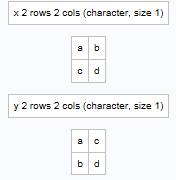
\*inaczej;

yy=t(x);



x={'a' 'b', 'c' 'd'};

y=x`;



\*cala masa bardzo uzytecznych funkcji macierzowych !!!;

\*Patrz Online Doc: SAS IML/Language Reference/Statements, Functions, and Subroutines;

/\* funkcje i operatory ulatwiajace tworzenie wektorów i macierzy \*/

\*Patrz Online Doc: SAS IML/Language Reference/Operators;

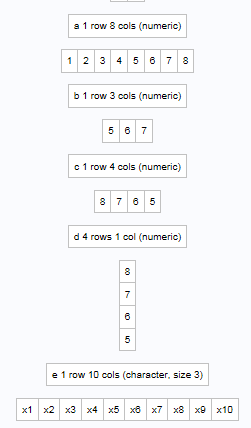
a=**1**:**8**;

b=**5**:**7**;

c=**8**:**5**;

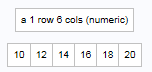
d=(**8**:**5**)`;

e='x1':'x10';



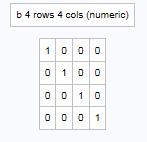
\*ciagi arytmetyczne;

a=do(**10**,**20**,**2**);



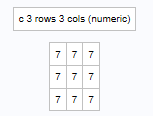
\*macierz jednostkowa;

b=i(**4**);



\*macierz stala;

c=j(**3**,**3**,**7**);

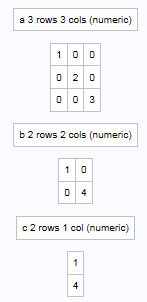


\*macierz diagonalna;

a=diag({**1** **2** **3**});

b=diag({**1** **2**,**3** **4**});

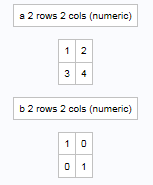
c=vecdiag({**1** **2**,**3** **4**});



/\* proste operacje na macierzach \*/

a={**1** **2**,**3** **4**};

b=i(**2**);

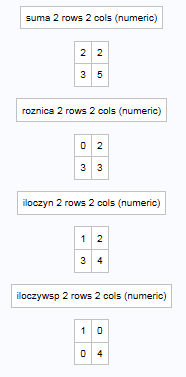


suma=a+b;

roznica=a-b;

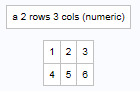
iloczyn=a\*b;

iloczywsp=a#b;



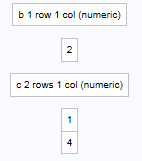
/\* indeksowanie macierzy \*/

a={**1** **2** **3**,**4** **5** **6**};

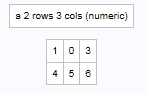


b=a[**1**,**2**];

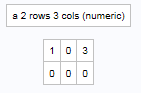
c=a[,**1**];



a[**1**,**2**]=**0**;

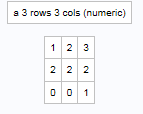


a[**2**,]=**0**; \*zmienia drugi wiersz na zera;



\*ciekawostki;

a={**1** **2** **3**,**2** **2** **2**,**0** **0** **1**};

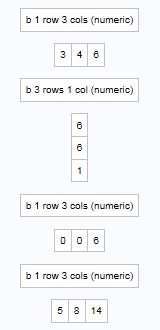


b=a[+,]; \*suma kolumn;

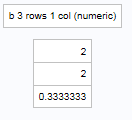
b=a[,+]; \*suma wierszy;

b=a[#,]; \*iloczyny kolumn;

b=a[##,]; \*kwadraty kolumn;

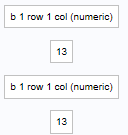


b=a[,:]; \*srednie wierszy;



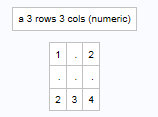
b=a[+,+];

b=sum(a); \*to jest szybsze;

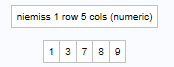


/\* Funkcja LOC \*/

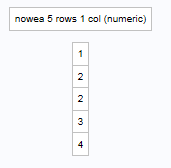
a={**1** **.** **2**,**.** **.** **.**,**2** **3** **4**};



niemiss=loc(a[,]^=**.**);



nowea=a[niemiss]; \* to daje wektor;



\*The LOC function creates a 1 ×n row vector, where n is the number of

nonzero elements in the argument. Missing values are treated as zeros.

The values in the resulting row vector are the locations of the nonzero

elements in the argument (in row-major order, like subscripting).;

\*uwaga - ostroznie z brakami danych:

SAS IML/Working with Matrices/More on Missing Values;

/\* Wspólpraca z DATA Stepami \*/

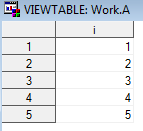
**data** a;

do i=**1** to **5**;

output;

end;

**run**;



**proc** **iml**;

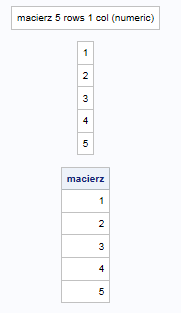
reset log print;

use a;

read all into macierz;

print macierz;

**quit**;



**proc** **iml**;

a={**1** **2**,**3** **4**};

b={**3** **4**,**5** **6**};

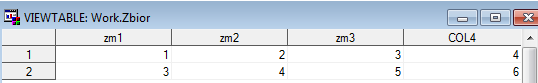
c=a||b;

kol={'zm1' 'zm2' 'zm3'};

create zbior from c[colname=kol]; \*tworzy strukture;

append from c; \*fizycznie dolacza;

**quit**; /\* UWAGA: otworzyc ten zbiór mozna dopiero po zamknieciu IML \*/



\* po kolei:;

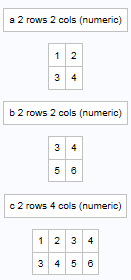
**proc** **iml**;

reset log print;

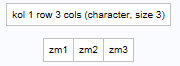
a={**1** **2**,**3** **4**};

b={**3** **4**,**5** **6**};

c=a||b;



kol={'zm1' 'zm2' 'zm3'};



/\* Elementy programowania\*/

\* petle, instrukcje warunkowe;

\* OnlineDoc:SAS IML/Programming Statements;

/\* MODULY \*/

\*-"procedury";

**proc** **iml**;

start F(x,a,b,y);

y=a\*x+b;

finish;

\*NOTE: Module F defined.;

**run** F(**0**,**1**,**1**,y1);

print y1;



\*-"funkcje";

start dlugosc\_kolumn(x);

sumakwad=x[##,];

return (sqrt(sumakwad));

finish;

x={**1**,**2**,**3**};

d=dlugosc\_kolumn(x);

print d;

**quit**;



\* Tworzenie makrozmiennych w IML;

**proc** **iml**;

x={**1** **2** **3**};

call symput('mz',x);

/\* konwersja niejawna NIE DZIALA w IMLu !!! \*/

/\* ERROR: (execution) Numeric argument should be character. \*/

%put \_user\_; /\* pusto...\*/

/\* jak przeprowadzic konwersje jawna? \*/

x\_znak=char(x);

/\* w druga strone dziala NUM \*/

print x;



call symput('mz',x\_znak);

%put \_user\_;



x={**10**,**20**,**30**};

x\_znak=char(x);

call symput('mz',x\_znak);

%put \_user\_;



x={**1** **2**,**3** **4**,**5** **6**};

x\_znak=char(x);

call symput('mz',x\_znak);

%put \_user\_;



/\* zatem "ostatni" element jest zapisywane na makrozmienna \*/

/\*jak zapisac elementy wektora ?\*/

call symput ('mz1',compress(char(x[**2**,**1**])));

%put \_user\_;



x={**1** **2** **3**};

x\_znak=char(x);

y=rowcat(x\_znak); \*concatenates rows

without using blank compression ;

print y;



call symput('mz',compbl(rowcat(x\_znak)));

\*Funkcja datastepowa - Removes multiple blanks from a character string ;

%put \_user\_;



**quit**;

\* Wysylanie emaili;

**data** \_null\_;

filename outbox email emailsys=MAPI;

file outbox to=("abc@mini.pw.edu.pl")

/\*cc=("cde@mini.pw.edu.pl")\*/

subject="wyniki"

/\*attach="d:\plik"\*/;

put 'Witam,';

put 'tralalala';

put 'Pozdrowienia';

**run**;

/\* sa pewne wymagania techniczne, aby to zadzialalo - do przeczytania w Online Docu \*/