Self-Amusement In MyLab

whyiami@163.com

1 Introduction

This program is a simple implementation for mathematics and matrix calculation. I had ever spent a period of time in Oracle Corporation, this program is my tribute to Oracle whose greatness goes far beyond its products.

2 General information

MyLab is written in C++ using a few C++ 11 features and it needs a compiler supporting these features, I use gcc 10.2.1 on GNU/Linux 5.15.77-amd64-desktop to compile it. Also it needs flex and bison to generate a scanner and a parser.

MyLab mostly contains a shared library along with a API and an executable program. The former implements the basic functions for vector and matrix. The latter is an interactive command line tool based on the former to run some maths and matrix calculation.

The test folder include a sample program to illustrate how to use the API, covering most of the interfaces.

3 Interactive tool

The interactive tool *mylab* generated in the bin folder depends on the previously mentioned so file. Before starting it I always run the env.sh to set the $LD_LIBRARY_PATH:

```
wxy@wxy-PC:~/01.program/MyLab$ . env.sh
add /home/wxy/01.program/MyLab/bin to $LD_LIBRARY_PATH
wxy@wxy-PC:~/01.program/MyLab$ bin/mylab
>
```

3.1 Native variables

Name	Meaning
ans	Store the result of the last operation
pi	circumference ratio
euler	Euler number e

```
>53 + 234
ans = 287
>pi
3.141593
>euler
2.718281
```

3.2 Assign statement

```
>abc=pi*euler-355
>abc
-346.460269
```

3.3 Mathematics Functions

Name	Function
abs	Absolute value
sqrt	Square root
pow	Power function
exp	value of e raised to the power of x
ln	natural logarithms
lg	The logarithm of base 10
log	The logarithm of base x
sin	Sine function
cos	Cosine function
tan	Tangent function

3.3.1 abs

```
>abs(99-3454)
ans = 3355
>abs(-363.5/567)
ans = 0.641093
```

3.3.2 sqrt

```
>sqrt(100); sqrt(-100); sqrt(3)
ans = 10.000000
ans = -nan
ans = 1.732051
```

3.3.3 pow

```
>pow(pi,2); pow(sqrt(97),2); pow(-5.3,5)
ans = 9.869604
ans = 97.000000
ans = -4181.954930
```

3.3.4 exp

```
>exp(3); euler*euler*euler; pow(euler,3)
ans = 20.085537
ans = 20.085519
ans = 20.085519
```

3.3.5 ln

```
>ln(euler); ln(-122); ln(122); ln(pow(euler,5.392))
ans = 1.000000
ans = nan
ans = 4.804021
ans = 5.391998
```

3.3.6 lg

```
>lg(10); lg(-23.4); lg(849.66)
ans = 1.000000
ans = nan
ans = 2.929245
```

3.3.7 log

```
>a=100.0983
>log(euler,a); ln(a); log(5,sqrt(5))
ans = 4.606154
ans = 4.606153
ans = 0.500000
```

3.3.8 sin

```
>sin(0.0);sin(pi/6);sin(pi/2);sin(pi)
ans = 0.000000
ans = 0.500000
ans = 1.000000
ans = 0.000000
>sin(pi/4); sqrt(2)/2
ans = 0.707107
ans = 0.707107
>sin(pi/3); sqrt(3)/2
ans = 0.866025
ans = 0.866025
```

3.3.9 cos

```
>cos(0);cos(100);cos(-100)
ans = 1.000000
ans = 0.862319
ans = 0.862319
>a=934.245
>pow(sin(a),2)+pow(cos(a),2)
ans = 1.000000
```

3.3.10 tan

```
>tan(0);tan(pi/4);tan(pi/2);tan(-8342)
ans = 0.000000
ans = 1.000000
ans = 37320539.634355
ans = -1.833574
```

3.4 Vector and Matrix computation

Name	Function
zeros	Create all zero matrix
ones	Create all one matrix
eye	Create identity matrix
rand	Create random matrix
diag	Create a diagonal matrix or get diagonal elements of a matrix
blkdiag	Create chunked diagonal matrix
cat	Concatenate some matrices
transpose	Transpose a matrix

inv	Matrix inversion
det	Matrix determinants
magic	Magic square matrix
find	Find the index and value of a non-zero element
length	The length of the maximum array dimension
linespace	Generate a linear spacing vector
logspace	Generate a logarithmic spacing vector
max	The largest element of the array
min	The smallest element of the array
prod	The product of array elements
size	Array size
numel	Number of array elements
reshape	Refactor the array
sort	Sort array elements
sum	Sum of array elements
dot	Dot product

3.4.1 vector scalar computation

```
>a=<345 -5465454543 32344 3345>
>a
 345  -5465454543  32344  3345
>a=<1 2 3 -6 2.7E5>
>a
 1.000000  2.000000  3.000000  -6.000000  270000.000000
>b=a*2; c=a/2
>b
 2.000000  4.000000  6.000000  -12.000000  540000.000000
>c
 0.500000  1.000000  1.500000  -3.000000  135000.000000
```

3.4.2 vector plus/minus

```
>a=<1,2,3,-6,2.7E5>
>b=<7,99, -100, -5456.35, 735>
>c=a + b;  d = a-b;
>c
 8.000000  101.000000  -97.000000  -5462.350000  270735.000000
>d
 -6.000000  -97.000000  103.000000  5450.350000  269265.000000
```

3.4.3 arithmetic progression

```
>a=12:-2:-17
>a
 12  10  8  6  4  2  0  -2  -4  -6  -8  -10  -12  -14  -16
```

3.4.4 matrix scalar computation

```
>a=rand(3,'int32')
>a
3×3
 1448212786  2117751120   299806422
 1194762178   615448417  1588529825
 1344254818  1783864547  1410438195
>a+5; a-39584523; a*1.56; a/577
ans = 3×3
 1448212791  2117751125   299806427
 1194762183   615448422  1588529830
 1344254823  1783864552  1410438200
ans = 3×3
 1408628263  2078166597   260221899
 1155177655   575863894  1548945302
 1304670295  1744280024  1370853672
ans = 3×3
 1448212786  2117751120   299806422
 1194762178   615448417  1588529825
 1344254818  1783864547  1410438195
ans = 3×3
 2509900  3670279   519595
 2070645  1066635  2753084
 2329731  3091619  2444433
```

3.4.5 matrix plus/minus/multiply/left-divide/right-divide

```
>a=[1 2 3;-4 -5 -6;7 8 9]; b=[-9 -8 -7;3 2 1;-6 -5 -4]
>a+b
ans = 3×3
 -8  -6  -4
 -1  -3  -5
  1   3   5
>a-b
ans = 3×3
 10  10  10
 -7  -7  -7
 13  13  13
>a*b
ans = 3×3
 -21  -19  -17
  57   52   47
 -93  -85  -77
>a/b
ans = 3×3
  0.000000  -2.500000  -1.000000
 -1.000000   2.000000   0.000000
  0.000000  -3.000000  -2.000000
>a\b
ans = 3×3
  -9.000000   -9.000000   -9.000000
  15.000000   14.000000   13.000000
 -13.000000  -12.000000  -11.000000
```

3.4.6 matrix zeros/ones/eye/rand

```
>zeros(2,4)
```

```
2×4
 0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000
>a=zeros(3, 'int64') ; a
3×3
 0  0  0
 0  0  0
 0  0  0
>b=ones(3,2,'single'); b
3×2
 1.000000  1.000000
 1.000000  1.000000
 1.000000  1.000000
>c=eye(3, 'int16'); c
3×3
 1  0  0
 0  1  0
 0  0  1
>d=rand(3,5)
>d
3×5
 0.968504  0.977058  0.256706  0.106121  0.017222
 0.251612  0.639529  0.468474  0.352592  0.614893
 0.784169  0.179397  0.355693  0.198715  0.942123
```

The last parameter is an optional string, it may be single|double|int8|int16|int32|int64|uint8|uint16|uint32|uint64. The default value is “double”.

3.4.7 matrix diag

```
>a=<10 -20 30 -40 50>
>b=diag(a); c=diag(a,2); d=diag(c,2)
>a
 10  -20  30  -40  50
>b
5×5
 10     0    0     0    0
  0   -20    0     0    0
  0     0   30     0    0
  0     0    0   -40    0
  0     0    0     0   50
>c
7×7
 0  0  10     0   0     0   0
 0  0   0   -20   0     0   0
 0  0   0     0  30     0   0
 0  0   0     0   0   -40   0
 0  0   0     0   0     0  50
 0  0   0     0   0     0   0
 0  0   0     0   0     0   0
```

```
>d
 10  -20  30  -40  50
```

3.4.8 matrix blkdiag

```
>a=[1 2 3;-4 -5 -6]; b=rand(3,4); c=eye(2,'int64')
>b
3×4
 0.478803  0.144940  0.261703  0.567272
 0.678177  0.011318  0.069300  0.559039
 0.256169  0.096485  0.445205  0.440989
>d=blkdiag(c,b*1000,a,'int32')
>d
7×9
 1  0    0    0    0    0   0   0   0
 0  1    0    0    0    0   0   0   0
 0  0  478  144  261  567   0   0   0
 0  0  678   11   69  559   0   0   0
 0  0  256   96  445  440   0   0   0
 0  0    0    0    0    0   1   2   3
 0  0    0    0    0    0  -4  -5  -6
```

3.4.9 cat

```
>a=rand(2,4,'int8'); b=ones(2,4,'int8')
>c=cat(1,a,b)
>c
4×4
 -68.000000  -81.000000  46.000000   29.000000
 -65.000000   66.000000  21.000000  -24.000000
   1.000000    1.000000   1.000000    1.000000
   1.000000    1.000000   1.000000    1.000000
>a=rand(3,2,'int8'); b=eye(3,'int8')
>d=cat(2,a,b,'int32')
>a
3×2
 -15   79
 -82    8
  -4  -44
>b
3×3
 1  0  0
 0  1  0
 0  0  1
>d
3×5
 -15   79  1  0  0
 -82    8  0  1  0
  -4  -44  0  0  1
```

The first parameter of cat is an integer, 1 means 1st-dimension, 2 means 2nd-dimension.

3.4.10 transpose

```
>a=rand(3,'int8')
>a
3×3
  95  -48   15
  20   45  -65
 -34   39   10
>transpose(a)
ans = 3×3
  95   20  -34
 -48   45   39
  15  -65   10
```

3.4.11 inv

```
>a=rand(3)
>a
3×3
 0.994614  0.382558  0.175796
 0.921419  0.674296  0.166885
 0.433566  0.578383  0.205632
>b=inv(a)
>b
3×3
  1.069506    0.584102  -1.388365
 -2.972873    3.256860  -0.101647
  6.106803  -10.392128   8.076250
>b*a
ans = 3×3
 1.000000   0.000000  0.000000
 0.000000   1.000000  0.000000
 0.000000  -0.000000  1.000000
>a*b
ans = 3×3
 1.000000  0.000000  -0.000000
 0.000000  1.000000  -0.000000
 0.000000  0.000000   1.000000
```

3.4.12 det

```
>a=rand(5,'int8')
>a
5×5
   19  -118  -45   -29    84
   95   -11  -22   -20   -77
 -104   -76  -33   -15    67
   66   113    7  -125   102
  112     9   89   -52  -127
>det(a)
ans = -24266198059.000000
```

This function use a recursion algorithm which takes a O(n!) complexity, tremendous sub-matrix are generated and destroyed during it’s execution. The max scale I ever tried is 13, it seems never return when the size goes up to 14.

3.4.13 magic

```
>magic(7)
ans = 7×7
 39  24  43   3   7  11  25
 28  33  42  16  40  41   5
 31  48  17  36   2  13  37
 15  34  22  14  26  32  35
 38  18   1   8  23  30  10
  9  19  27  44  29  12  45
 46  21  47  49   4   6  20
```

3.4.14 find

```
>X = [1 0 2; 0 1 1; 0 0 4]
>find(X)
ans =   1  5  7  8  9
>find(~X)
ans =   2  3  4  6
>find(X==1)
ans =   1  5  8
```

3.4.15 length

```
>a=<2 3 4 6 6 >
>length(a)
ans = 5
>b=ones(3,7)
>length(b)
ans = 7
```

3.4.16 linespace

```
>linspace(-5,5)
generate a row vector of one hundred equally spaced points
>linspace(-5,5,7)
ans =   -5.000000  -3.333333  -1.666667  0.000000  1.666667
3.333333  5.000000
```

3.4.17 logspace

```
>logspace(1,5)
Generate a row vector of one hundred log-spaced points
>logspace(1,5,7)
ans =   10.000000  46.415888  215.443469  1000.000000  4641.588834
21544.346900  100000.000000
```

3.4.18 max

```
>a=<345 5677 8 4 -24 56>
>max(a)
ans = 5677
>b=rand(5,'int16')
>b
5×5
 30499   10740  16217    9573  -32499
 19070   22736  23499  -18432    5180
 28434   26466  -5619   12079  -23287
 22226    2251   1361    9600   32575
 26576  -11730  31315  -10933  -27037
>max(b)
ans =   30499  26466  31315  12079  32575
```

3.4.19 min

```
>a=<75 766 -3662 53 889>
>min(a)
ans = -3662
>b=rand(5,'int16')
>b
5×5
  22924    5604  -15703   27456   -4747
 -30811    1346    1471   10894   20156
 -11924   -9659  -11814  -31722  -14940
  -3761  -11436  -21580  -11884    4501
  11135      84  -27175   13078  -15266
>min(b)
ans =   -30811  -11436  -27175  -31722  -15266
```

3.4.20 prod

```
>a=<4 6 78 88 -45>
>prod(a)
ans = -7413120
>b=rand(3)
>b
3×3
 0.392816  0.308717  0.831351
 0.418629  0.710342  0.125121
 0.889152  0.850775  0.581261
>prod(b)
ans =   0.146216  0.186570  0.060462
```

3.4.21 size

```
>a=<56 6 7  89 4 7>
>size(a)
ans = 6
>b=ones(3,5)
>b
```

```
3×5
 1.000000  1.000000  1.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  1.000000  1.000000
>size(b)
ans =   3  5
```

3.4.22 numel

```
>a=<3 4 4 56 7>
>numel(a)
ans = 5
>b=zeros(3,5)
>numel(b)
ans = 15
```

3.4.23 reshape

```
>a=rand(4,'int8')
>a
4×4
  13   17   68  -70
  29  -17  -15   59
 -74  111  121  -79
 -48    2   96  -58
>reshape(a,<2 8>)
>a
2×8
  13   17   68  -70   29  -17  -15   59
 -74  111  121  -79  -48    2   96  -58
```

3.4.24 sort

```
>a=<13   17   68  -70   29  -17  -15   59>
>sort(a)
>a
 -70  -17  -15  13  17  29  59  68
>sort(a, ‘ascend’)
>a
 -70  -17  -15  13  17  29  59  68
>sort(a,'descend')
>a
 68  59  29  17  13  -15  -17  -70
>a=rand(4,3,'int8')
>sort(a)
>a
4×3
 -69  -118  -19
 -12  -106   64
  27   -69   79
 116   -10  123
>a=rand(4,3,'int8')
>a
```

```
4×3
  12  100    69
   0  -23   -32
 -76   88    84
  20    4  -123
>sort(a)
>a
4×3
 -76  -23  -123
   0    4   -32
  12   88    69
  20  100    84
>sort(a,'ascend')
>a
4×3
 -76  -23  -123
   0    4   -32
  12   88    69
  20  100    84
>sort(a,'descend')
>a
4×3
  20  100    84
  12   88    69
   0    4   -32
 -76  -23  -123
```

3.4.25 sum

```
>a=1.2:1.1:10
>a
 1.200000  2.300000  3.400000  4.500000  5.600000  6.700000
7.800000  8.900000  10.000000
>sum(a)
ans = 50.400000
>a=rand(4,'int8')
>a
4×4
 -64    9   -3  -75
  80  -96   -2   82
 -57   57  -13  -91
 -59  122  103  110
>sum(a)
ans =   -100  92  85  26
```

3.5 MyLab command

Name	Function
who	List variables
whos	List variables in detail
clc	Clean screen
clear	Clear variables

exit	Exit MyLab

3.5.1 who

```
>who
Your variables are:
a  ans  b  euler  pi
```

3.5.2 whos

```
>whos
Name   Class     Type      Size
a      vector    double    9
ans    scalar    integer
b      matrix    double    73×18
euler  scalar    double
pi     scalar    double
```

3.5.3 clear

```
>who
Your variables are:
a  ans  b  c  d  euler  pi
>clear a b
>who
Your variables are:
ans  c  d  euler  pi
>clear
>who
Your variables are:
ans  euler  pi
```

3.5.4 exit

Both *Ctrl-D* and *exit* will terminate the program.