

学生：王雅滢

批改：李娟

得分：36/40

PS6_王雅滢

陈鹏翰、龚国庆、蒋莲洁向我解释了此次作业，跟他们学习了很多，非常感谢他们。

1. Matrix multiplication

1.1 Write a program Main.f90 to read fortran_demo1/M.dat as the matrix M, and fortran_demo1/N.dat as the matrix N.

Main.f90:

Program Main

implicit none

integer :: x1, x2, mc, mr, nc, nr, i, j

real(8),dimension(:,:),allocatable :: M, N

x1=50

x2=51

mc=3

mr=4

nc=4

nr=3

open(unit=x1,file='M.dat',status='old')

open(unit=x2,file='N.dat',status='old')

allocate(M(mr,mc))

allocate(N(nr,nc))

do i=1,mr

 read(x1,*) M(i,:)

enddo

do i=1,nr

 read(x2,*) N(i,:)

enddo

do i=1,mr

 write(*,*) 'Line',i,':',M(i,:)

enddo

do i=1,nr

 write(*,*) 'Line',i,':',N(i,:)

enddo

deallocate(M)

deallocate(N)

End Program Main

输出:

```
[ese-wangyy@login02 fortran_demo1]$ gfortran Main.f90 -o Main.x
[ese-wangyy@login02 fortran_demo1]$ ./Main.x
Line      1 :  19.480000000000000      15.789999999999999      19.280000000000001
Line      2 :  19.280000000000001      12.920000000000000      15.859999999999999
Line      3 :  15.859999999999999      11.289999999999999      14.039999999999999
Line      4 :  11.930000000000000      18.600000000000001      18.230000000000000
Line      1 :  7.719999999999999      4.110000000000000      1.439999999999999
4.799999999999999
Line      2 :  5.549999999999999      4.799999999999999      4.040000000000000
0.589999999999999
Line      3 :  0.589999999999999      8.580000000000001      2.259999999999999
7.719999999999999
```

1.2 Write a subroutine Matrix_multip.f90 to do matrix multiplication.

subroutine Matrix_multip(M,N,MN)

脚本:

subroutine Matrix_multip(M,N,MN)

implicit none

real(8),dimension(4,3),intent(in) :: M

real(8),dimension(4,3),intent(in) :: N

real(8),dimension(4,4),intent(out) :: MN

integer :: i,j,k

real(8) :: t

do i=1,4

do j=1,4

t=0

do k=1,3

t=t+M(i,k)*N(k,j)

enddo

MN(i,j)=t

enddo

enddo

End subroutine Matrix_multip

1.3 Call the subroutine Matrix_multip() from Main.f90 to compute $M \cdot N$; write the output to a new file MN.dat, values are in formats of f9.2.

脚本:

Program MainM

implicit none

```
integer                :: x1, x2, mc, mr, nc, nr, i, j
real(8), dimension(:, :), allocatable :: M, N
real(8), dimension(4,4) :: MN
```

x1=50

x2=51

mc=3

mr=4

nc=4

nr=3

open(unit=x1,file='M.dat',status='old')

open(unit=x2,file='N.dat',status='old')

allocate(M(mr,mc))

allocate(N(nr,nc))

do i=1,mr

 read(x1,*) M(i,:)

enddo

do i=1,nr

 read(x2,*) N(i,:)

enddo

close(x1)

close(x2)

do i=1,mr

 write(*,*) "Line ",i,":",M(i,:)

enddo

do i=1,nr

 write(*,*) "Line ",i,":",N(i,:)

enddo

call Matrix_multip(M,N,MN)

```

do i=1,4
    write(*,*) "Line ",i,".",MN(i,:)
enddo

open(unit=u1,file='new1.dat',status='replace')

do i=1,4
    write(u1,'(f9.2)') MN(i,:)
enddo

close(x1)

deallocate(M)
deallocate(N)

End Program MainM

```

输出： 请以F9.2格式输出，-1

```

[ese-wangyy@login02 fortran_demo1]$ gfortran MainM.f90 Matrix_multip.f90 -o a.x
[ese-wangyy@login02 fortran_demo1]$ ./a.x
Line 1 : 19.480000000000000 15.789999999999999 19.280000000000001
Line 2 : 19.280000000000001 12.920000000000000 15.859999999999999
Line 3 : 15.859999999999999 11.289999999999999 14.039999999999999
Line 4 : 11.930000000000000 18.600000000000001 18.230000000000000
Line 1 : 7.719999999999999 4.110000000000003 1.439999999999999
4.799999999999999 4.799999999999999 4.040000000000000
Line 2 : 5.549999999999999 4.799999999999999 4.040000000000000
0.589999999999999 8.580000000000001 2.259999999999999
Line 3 : 0.589999999999999 8.580000000000001 2.259999999999999
7.719999999999999 256.7453999999999 131.1919999999999
Line 1 : 249.3953000000000 226.2360000000000 114.9461999999999
8.025896322107514E-314 229.9049999999999 193.2137999999999
Line 2 : 229.9049999999999 193.2137999999999 98.3191999999999
6.6022120305699865E-314 5.8445735962430244E-314
Line 3 : 193.2137999999999 98.3191999999999 243.1032000000000
5.8445735962430244E-314 206.0852999999999 126.9975000000000
Line 4 : 206.0852999999999 126.9975000000000 7.5889892810028306E-314

```

2. Calculate the Solar Elevation Angle

The solar elevation angle (SEA) is the angle between the imaginary horizontal plane on which you are standing and the sun in the sky. SEA is very important in deciding the inclination of solar panels, in both photovoltaics (PV) and thermal. The value of the SEA depends on the location on the Earth and the local date and time.

Please read this Solar Elevation Angle – Calculating Altitude of Sun and links therein for how to calculate SEA.

2.1 Write a module Declination_angle that calculates the declination angle on a given date.
[Hint: using the “Better formula” from Solar Declination Angle & How to Calculate it]

创建: vi Declination_angle.f90

脚本:

```
module Declination_angle
```

```
implicit none
```

```
! 考虑一个月 30 天。
```

the result was wrong,-1

```
real, parameter :: pi=3.1415926536
```

```
contains
```

```
subroutine cal_angle(m,d,da)
```

```
implicit none
```

```
integer,intent(in) :: m, d
```

```
real(8),intent(out) :: da
```

```
integer ::doy
```

```
doy=(m-1)*30+d
```

```
da=asin(sin(-23.44/180*pi)*cos(((360/365.24)*(doy+10)+360/pi*0.0167*sin(360/365.24*(doy-2)))/180*pi))
```

```
da=da/pi*180
```

```
End subroutine cal_angle
```

```
End module Declination_angle
```

经过测试，可以运行：

```
[ese-wangyy@login02 fortran_demo1]$ gfortran Main.f90 Declination_angle.f90 -o a.x
[ese-wangyy@login02 fortran_demo1]$ ./a.x
Line      1 :   19.480000000000000      15.789999999999999      19.280000000000001
Line      2 :   19.280000000000001      12.920000000000000      15.859999999999999
Line      3 :   15.859999999999999      11.289999999999999      14.039999999999999
Line      4 :   11.930000000000000      18.600000000000001      18.230000000000000
Line      1 :    7.719999999999999      4.110000000000003      1.439999999999999
   4.799999999999999
Line      2 :    5.549999999999998      4.799999999999998      4.040000000000000
   0.589999999999997
Line      3 :    0.589999999999997      8.580000000000001      2.259999999999998
   7.719999999999998
```

2.2 Write a module Solar_hour_angle that calculates the solar hour angle in a given location for a given date and time.

[Hint: using the formulas from Solar Hour Angle & How to Calculate it]

创建：vi Solar_hour_angle

脚本：

```
module Solar_angle_hour
```

```
implicit none
```

```
real, parameter :: pi=3.1415926536
```

```
contains
```

```
subroutine cal_sla(lon,m,d,t,sah)
```

```
implicit none
```

```
integer,intent(in) :: m,d
```

```
real(8),intent(in) :: lon, t
```

```
real(8),intent(out) :: sah
```

```
integer :: doy
```

```
real(8) :: offset, eot, gam
```

```
doy=(m-1)*30+d
```

```
gam=2*pi/365*(doy-1+(t-12)/24)
```

```
eot=229.18*(0.000075+0.001868*cos(gam)-0.032077*sin(gam)-0.014615*cos(2*gam)-0.040849
*sin(2*gam))
```

```
offset=eot+MOD(lon,15.0)
```

```
sah=15*(t-12)+offset/60
```

```
End subroutine cal_sla
```

经过测试，可以运行。

test 结果不对？-1

```
[ese-wangyy@login02 fortran_demo1]$ vi Test2.f90
[ese-wangyy@login02 fortran_demo1]$ gfortran Solar_hour_angle.f90 Test2.f90 -o b.x
[ese-wangyy@login02 fortran_demo1]$ ./b.x
52.513306131446733
```

2.3 Write a main program (Solar_elevation_angle.f90) that uses module Declination_angle and Solar_hour_angle to calculate and print the SEA in a given location for a given date and time.

创建： vi Solar_elevation_angle.f90

脚本：

Program Solar_elevation_angle

use Declination_angle

use Solar_angle_hour

implicit none

real, parameter :: pii=3.1415926536

real(8) :: lat,lon,t,sah,da

integer :: m,d

real(8) :: aes

lat=32.22

lon=1.0

t=10.0

m=3

d=3

call cal_angle(m,d,da)

call cal_sla(lon,m,d,t,sah)

aes=asin(sin(lat/180*pii)*sin(da/180*pii)+cos(lat/180*pii)*cos(da/180*pii)*cos(sah/180*pii))

aes=aes/pii*180.0

write(*,*) aes

End program Solar_elevation_angle

输出：

```
[ese-wangyy@login02 fortran_demo1]$ gfortran Solar_elevation_angle.f90 Declination_angle.f90 Solar_hour_angle.f90 -o c.x
[ese-wangyy@login02 fortran_demo1]$ ./c.x
41.045703954998608
```

2.4 Create a library (libsea.a) that contains Declination_angle.o and Solar_hour_angle.o. Compile Solar_elevation_angle.f90 using libsea.a. Print the SEA for Shenzhen (22.542883N, 114.062996E) at 10:32 (Beijing time; UTC+8) on 2021-12-31.

创建 libsea.a:

```
[ese-wangyy@login02 fortran_demo1]$ gfortran -c Declination_angle.f90
[ese-wangyy@login02 fortran_demo1]$ gfortran -c Solar_hour_angle.f90
[ese-wangyy@login02 fortran_demo1]$ ar rcvf libsea.a Declination_angle.o Solar_hour_angle.o
a - Declination_angle.o
a - Solar_hour_angle.o
```

深圳的 SEA 脚本:

Program SZ_SEA

use Declination_angle

use Solar_angle_hour

implicit none

real, parameter :: pii=3.1415926536

real(8) :: lat,lon,t,sah,da

integer :: m,d

real(8) :: aes

lat=22.542883

lon=114.062996

t=10.0+32/60

m=12

d=31

call cal_angle(m,d,da)

call cal_sla(lon,m,d,t,sah)

aes=asin(sin(lat/180*pii)*sin(da/180*pii)+cos(lat/180*pii)*cos(da/180*pii)*cos(sah/180*pii))

aes=aes/pii*180.0

write(*,*) aes

End program SZ_SEA

输出:

```
[ese-wangyy@login02 fortran_demo1]$ gfortran SZ_SEA.f90 -o d.x -L. -lsea
[ese-wangyy@login02 fortran_demo1]$ ./d.x
35.790305803209272
```

参考结果 : SEA=36.61 ° , -1