

学生：毛梅雪

批改：李娟

得分：28/40

1. Matrix multiplication

1.1 [5 points] Write a program `Main.f90` to read `fortran_demo1/M.dat` as the matrix `M`, and `fortran_demo1/N.dat` as the matrix `N`.

(用服务器跑代码报错：ld returned 1 exit status, 本地 VScode 运行代码和结果如下)

```
D: > Main.f90
1  program Main
2      implicit none
3      integer , parameter :: na = 3 , ma = 5 , nb = 5 , mb = 3
4      Real(8) :: M(3,5), N(5,3) , P (3 ,3)
5
6      open(unit=20, file='../fortran_demo1/M.dat', status='old')
7
8      read(20,*) M
9
10     close(20)
11
12     open(unit=20, file='../fortran_demo1/N.dat', status='old')
13
14     read(20,*) N
15
16     close(20)
```

```
≡ M.dat
1  19.48 15.79 19.28
2  19.28 12.92 15.86
3  15.86 11.29 14.04
4  11.93 18.60 18.23
5  19.28 12.92 15.86
6
```

```
≡ N.dat
1  7.72 4.11 1.44 4.80 5.55
2  5.55 4.80 4.04 0.59 8.58
3  0.59 8.58 2.26 7.72 4.11
4
```

1.2 [5 points] Write a subroutine `Matrix_multip.f90` to do matrix multiplication.

```
D: > Matrix_multip.f90 > ...
1  subroutine Matrix_multip (C ,D , P )
2      implicit none
3      integer i , j , k
4      integer , parameter :: nc = 3 , mc = 5 , md = 3
5      real , intent ( in ) :: C ( nc , mc ) , D ( mc , md )
6      real , intent ( out ) :: P ( nc , md )
7      real :: sum
8
9      do i =1 , nc
10         do j =1 , md
11             sum = 0.
12             do k =1 , mc
13                 sum = sum + C ( i , k ) * D ( k , j )
14             enddo
15             P ( i , j ) = sum
16         enddo
17     enddo
18
19 end subroutine Matrix_multip
```

1.3 [5 points] Call the subroutine `Matrix_multip()` from `Main.f90` to compute $M*N$; write the output to a new file `MN.dat`, values are in formats of `f9.2`.

```
D: > ≡ Main.f90
1  program Main
2      implicit none
3      integer , parameter :: na = 3 , ma = 5 , nb = 5 , mb = 3
4      Real(8)  :: M(3,5), N(5,3) , P (3 ,3)
5
6      open(unit=20, file='~/fortran_demo1/M.dat', status='old')
7
8      read(20,*) M
9
10     close(20)
11
12     open(unit=20, file='~/fortran_demo1/N.dat', status='old')
13
14     read(20,*) N
15
16     close(20)
17
18     call Matrix_multip( M , na , ma ,N , mb , P )
19
20     open(unit=20, file='~/fortran_demo1/MN.dat', status='replace')
21
22     write(20,'(3/(3f9.2))') P
23     ! 3/ : 3 rows; 3f9.2 : f9.2 for 3 times.
24     close(20)
25
26     end program Main
```

Output:

```
≡ MN.dat ×
≡ MN.dat
1
2
3
4  416.73  352.24  409.77
5  437.19  317.09  386.69
6  384.10  342.38  385.10
7
```

2.1 [5 points] 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](#)]

```
module Declination_angle
    implicit none
    real(8), parameter      :: pi=3.1415926536
contains
    subroutine CalDeclinationAngle(day,sigma)
        integer, intent(in)    :: day
        real(8), intent(out)   :: sigma
        !SIND=SIN(angle*pi/180)  COSD=COS(angle*pi/180)
        sigma = ASIND(SIND(-23.44)*COSD(360*(day+10)/365.24+360/pi*0.0167*SIN(360*(day-2)/365.24)))
    end subroutine CalDeclinationAngle
end module Declination_angle
```

代码公式错误，转弧度，-2

2.2 [10 points] 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](#)]

```
module Solar_hour_angle
    implicit none
    real, parameter      :: pi = 3.1415926536
contains
    subroutine CalSolarHourAngle(day,time,gma,longitude,timezone,h)
        integer, intent(in)    :: day
        real(8), intent(in)    :: time, longitude, timezone
        real(8)                :: gma, EOT, OFFSET, LST
        real(8), intent(out)   :: h
        gma = 2*pi/365*(day-1+(time-12)/24)
        EOT = 229.18*(0.000075+0.001868*COS(gma)-0.032077*SIN(gma)-0.014615*COS(2*gma)-0.040849*SIN(2*gma))
        OFFSET = EOT+4*(longitude-15*timezone)
        LST = time+OFFSET/60
        h = 15*(LST-12)
    end subroutine CalSolarHourAngle
end module Solar_hour_angle
```

2.3与2.4未作答，-10

