```fortran
! CGV/solution/parts/cgvp01.F90

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                                               !
! This file has been written as a sample solution to an         !
! exercise in a course given at the High Performance            !
! Computing Centre Stuttgart (HLRS).                            !
! It is made freely available with the understanding that       !
! every copy of this file must include this header and that     !
! HLRS takes no responsibility for the use of the enclosed      !
! teaching material.                                            !
!                                                               !
! Authors: Dr. Rolf Rabenseifner                                !
!          Stephan Scheiderer                                   !
!                                                               !
! Contact: rabenseifner@hlrs.de                                 !
!                                                               !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                                             !EOH
! >>>>>>>>>>>>>>>>>>>>>> begin of task 01 <<<<<<<<<<<<<<<<<<<<<<

! ********************* vector routines ***********************

!-------------------------------------------
! add_vec_vec
!  IN  REAL(0:) v1      : vector 1
!  IN  REAL     lambda  : factor lambda
!  IN  REAL(0:) v2      : vector 2
!  OUT REAL(0:) v3      : vector 3
!
! RETURNS:
!   --
!
! TASK:
!   calculate:   v3 := v1 + lambda * v2
!
! GLOBAL DATA:
!   IN     INTEGER rowsize : number of data entries in a row of the local chunk
!   IN     INTEGER colsize : number of data entries in a column of the local chunk
!   IN/OUT REAL    flops   : amount of FLOPS
!
! USED FUNCTIONS:
!   --
SUBROUTINE add_vec_vec(v1, lambda, v2, v3)
    REAL, DIMENSION(0:), INTENT(IN)  ::  v1
    REAL, INTENT(IN)                 ::  lambda
    REAL, DIMENSION(0:), INTENT(IN)  ::  v2
    REAL, DIMENSION(0:), INTENT(OUT) ::  v3
    INTEGER                          ::  i

    ! addition of two vectors, second vector is multiplied by a scalar
    DO i = (rowsize + 2), ((rowsize + 2) * (colsize + 1)) - 1
        v3(i) = v1(i) + lambda * v2(i);
    END DO

    ! for the MFLOPS calculation
    flops = flops + (rowsize + 2) * colsize * 2
END SUBROUTINE add_vec_vec

!------------------------------
! dot_product_vec_vec
!  IN  REAL(0:) v1  : vector 1
!  IN  REAL(0:) v2  : vector 2
!
!  RETURNS:
!    REAL: dot product
!
! TASK:
!   calculate dot product for vector 1 and vector 2
!
! GLOBAL DATA:
!   IN     INTEGER rowsize : number of data entries in a row of the local chunk
!   IN     INTEGER colsize : number of data entries in a column of the local chunk
!   IN/OUT REAL    flops   : amount of FLOPS
!
! USED FUNCTIONS:
!   APPL:
!     --
!   MPI :
!     MPI_Allreduce
REAL FUNCTION dot_product_vec_vec(v1, v2)
    REAL, DIMENSION(0:), INTENT(IN) :: v1
    REAL, DIMENSION(0:), INTENT(IN) :: v2

    INTEGER :: i
    REAL    :: sum, global_sum

    ! sum = dot product of vectors
    sum = 0
    DO i = (rowsize + 2),((rowsize + 2) * (colsize + 1)) - 1
        sum = sum + v1(i) * v2(i)
    END DO

! global sum
#ifdef serial
    global_sum = sum
#endif
```

```
422  #ifndef serial
423      commtime_start = MPI_WTIME()
424      CALL MPI_ALLREDUCE(sum, global_sum, 1, MPI_REAL, MPI_SUM, comm_cart, ierror)
425      commtime_end = MPI_WTIME()
426      commtime_sum = commtime_sum + commtime_end - commtime_start
427      flops = flops + 1
428  #endif
429
430      ! for the MFLOPS calculation
431      flops = flops + (rowsize + 2) * colsize * 2
432
433      dot_product_vec_vec = global_sum
434  END FUNCTION dot_product_vec_vec
435
```

```
436 !------------------------------
437 ! max_norm_vec
438 !  IN  REAL(0:) v1  : vector 1
439 !
440 !  RETURNS:
441 !    REAL: maximum norm
442 !
443 ! TASK:
444 !   calculate (vector maximum norm) for vector 1
445 !
446 ! GLOBAL DATA:
447 !   IN INTEGER rowsize : number of data entries in a row of the local chunk
448 !   IN INTEGER colsize : number of data entries in a column of the local chunk
449 !
450 ! USED FUNCTIONS:
451 !   APPL:
452 !     --
453 !   MPI :
454 !     MPI_ALLREDUCE
455 REAL FUNCTION max_norm_vec(v1)
456     REAL, DIMENSION(0:), INTENT(IN) :: v1
457
458     INTEGER :: i
459     REAL    :: global_max = 0
460     ! maxabs = maximum norm of the local vector
461     REAL    :: maxabs = 0
462     DO i = (rowsize + 2), ((rowsize + 2) * (colsize + 1)) - 1
463        IF ( v1(i) > maxabs) THEN
464             maxabs =  v1(i)
465         END IF
466         IF (-v1(i) > maxabs) THEN
467             maxabs = -v1(i)
468         END IF
469     END DO
470
471     ! global maximum
472 #ifdef serial
473     global_max = maxabs
474 #endif
475 #ifndef serial
476     commtime_start = MPI_WTIME()
477     CALL MPI_ALLREDUCE(maxabs, global_max, 1, MPI_REAL, MPI_MAX, comm_cart, ierror)
478     commtime_end = MPI_WTIME()
479     commtime_sum = commtime_sum + commtime_end - commtime_start
480 #endif
481     ! no flops added, because we have only assignments and no operations
482     max_norm_vec = global_max
483 END FUNCTION max_norm_vec
484
485 !------------------------------
486 ! sqr_norm_vec
487 !  IN  REAL(0:) v1  : vector 1
488 !
489 !  RETURNS:
490 !    REAL: square of norm
491 !
492 ! TASK:
493 !   calculate global (vector norm Euklid)^2 for vector 1
494 !
495 ! GLOBAL DATA:
496 !   IN     INTEGER rowsize : number of data entries in a row of the local chunk
497 !   IN     INTEGER colsize : number of data entries in a column of the local chunk
498 !   IN/OUT REAL    flops   : amount of FLOPS
499 !
500 ! USED FUNCTIONS:
501 !   APPL:
502 !     --
503 !   MPI :
504 !     MPI_ALLREDUCE
505 REAL FUNCTION sqr_norm_vec(v1)
506     REAL, DIMENSION(0:), INTENT(IN) :: v1
507
508     INTEGER :: i
509     REAL    :: sum, global_sum
510
511     ! sum = dot product of vector
512     sum = 0
513     DO i = (rowsize + 2), ((rowsize + 2) * (colsize + 1)) - 1
514         sum = sum + v1(i) * v1(i)
515     END DO
516
517     ! global sum
518 #ifdef serial
519     global_sum = sum;
520 #endif
521 #ifndef serial
522     commtime_start = MPI_WTIME()
523     CALL MPI_Allreduce(sum, global_sum, 1, MPI_REAL, MPI_SUM, comm_cart, ierror)
524     commtime_end = MPI_WTIME()
525     commtime_sum = commtime_sum + commtime_end - commtime_start
526     flops = flops + 1
527 #endif
528     ! for the MFLOPS calculation
529     flops = flops + (rowsize + 2) * colsize * 2
530     sqr_norm_vec = global_sum
531 END FUNCTION sqr_norm_vec
532 !>>>>>>>>>>>>>>>>>>>>>> -end- of task 01 <<<<<<<<<<<<<<<<<<<<<<
```