Programowanie współbieżne

Laboratorium - 7a: Fortran 95

Andrzej Baran
baran@kft.umcs.lublin.pl

Zadania7.

Fortran 95

1. Przeczytaj pracę W.R. Hamiltona “On quaternions” (http://www.maths.tcd.ie/pub/HistMath/People/Hamilton/Quatern2/Quatern2.html). Napisz moduł `Quaternions` (Fortran95) do działań na kwaternionach (`+, -, /, *, conjg, norm`). Zastosuj intarface i overloading (przeciążania operatorów). Pamiętaj o mnożeniu przez liczbę (z lewej strony, z prawej strony, real, integer)
2. Napisz interfejs przypisania (=) i drukowania (`q_print`) kwaternionów.
3. Napisz prosty program używający modułu `Quaternions` i sprawdź proste działania w ciele kwaternionów.
4. Jak wykorzystać kwaterniony w opisie obrotów? Napisz program realizujący obrót wektora (napisz odpowiedni moduł “wektory”) wykorzystujący moduł `Quaternions`,

Mnożenie w ciele kwaternionów ...?

Quaternions I

```
module Quaternions

  type, public :: quaternion
     real :: a, b, c, d
  end type quaternion

  intrinsic :: conjg

  private quat_mul_real, real_mul_quat, quat_mul_int, &
       int_mul_quat, quat_mul, quat_sub, quat_div, &
       quat_div_real, quat_div_int, quat_conjg

  interface operator (+)
     module procedure quat_add
  end interface

  interface operator (*)
     module procedure quat_mul_real
     module procedure real_mul_quat
     module procedure quat_mul_int
     module procedure int_mul_quat
     module procedure quat_mul
  end interface

  interface operator (-)
     module procedure quat_sub
  end interface
```

Quaternions II

```
  interface operator (/)
     module procedure quat_div
!     module procedure quat_div_real
!     module procedure quat_div_int
  end interface

  interface conjg
     module procedure quat_conjg
  end interface

contains

  function quat_add(x,y) result (res)
    type(quaternion), intent(in) :: x, y
    type(quaternion) :: res
    res % a = x % a + y % a
    res % b = x % b + y % b
    res % c = x % c + y % c
    res % d = x % d + y % d
  end function quat_add

  function quat_sub(x,y) result (res)
    type(quaternion), intent(in) :: x, y
    type(quaternion) :: res
    res % a = x % a - y % a
    res % b = x % b - y % b
    res % c = x % c - y % c
    res % d = x % d - y % d
  end function quat_sub

  function quat_conjg(x) result (res)
```

Quaternions III

```
  type(quaternion), intent(in) :: x
  type(quaternion) :: res
  res % a = x % a
  res % b = -(x % b)
  res % c = -(x % c)
  res % d = -(x % d)
end function quat_conjg

function quat_mul_real(x,r) result (res)
  ! quat * real
  type(quaternion), intent(in) :: x
  real, intent(in) :: r
  type(quaternion) :: res
  res % a = x % a *r
  res % b = x % b *r
  res % c = x % c *r
  res % d = x % d *r
end function quat_mul_real

function real_mul_quat(r,x) result (res)
  ! real * quat
  type(quaternion), intent(in) :: x
  real, intent(in) :: r
  type(quaternion) :: res
  res = quat_mul_real(x,r)
end function real_mul_quat

function int_mul_quat(i,x) result (res)
  ! integer * quat
  type(quaternion), intent(in) :: x
  integer, intent(in) :: i
```

Quaternions IV

```
  type(quaternion) :: res
  res = quat_mul_real(x,real(i))
end function int_mul_quat

function quat_mul_int(x,i) result (res)
  ! quat * integer
  type(quaternion), intent(in) :: x
  integer, intent(in) :: i
  type(quaternion) :: res
  res = int_mul_quat(i,x)
end function quat_mul_int

function quat_norm(q) result(res)
  type(quaternion), intent(in) :: q
  real res
  real :: ap, bp, cp, dp
  ap = q % a
  bp = q % b
  cp = q % c
  dp = q % d
  res = sqrt(ap*ap+bp*bp+cp*cp+dp*dp)
end function quat_norm

function quat_div(x,y) result (res)
  type(quaternion), intent(in) :: x, y
  type(quaternion) :: res
  real r
  r = 1.0/quat_norm(y)**2
  res = quat_mul_real(x,r)
end function quat_div
```

Quaternions V

```
  ! MULTIPLICATION
  function quat_mul(x,y) result (res)
    type(quaternion), intent(in) :: x, y
    type(quaternion) :: res
    real r
    res%a=0.; res%b=0.; res%c=0.; res%d=0.
  end function quat_mul

  subroutine quaternion_print(q)
    type(quaternion), intent(in) :: q
    real :: ap, bp, cp, dp
    ap = q % a
    bp = q % b
    cp = q % c
    dp = q % d
    print "(a1,4f12.6,a1)", "(",ap,bp,cp,dp,")"
  end subroutine quaternion_print

end module Quaternions
!
!-----------------------------------------------
! TEST
Program Quater
  use Quaternions
  type(quaternion) :: u, v, w
  u=quaternion(1,2,2,1)
  v=quaternion(4,3,2,1)
  w=u+v
  call quaternion_print(w)
  call quaternion_print(conjg(w))
  call quaternion_print(w/w)
```

Quaternions VI

```
  call quaternion_print(2.*w)
  call quaternion_print(w*2.)
  u = quaternion(1,0,0,0)
  print *, quat_norm(u)
end program Quater
```

Uzupełnij moduł `Quaternions` o brakujące procedury, funkcje, operacje.