Source Code: heateq_simple.f90

```fortran
program example
  ! Example 10.3 from NMUM/Mathews.
  ! Integrate the 1D heat equation forward in time given simple initial
  ! conditions.  This is a simple example of how to use Fortran90.  There
  ! are better ways to code this problem...

  ! Begin the declaration section.
  ! ALWAYS start with implicit none.
  implicit none

  ! Declare some variables.  Set intital values on some.
  ! Note that xLim is a 2-element vector.
  real    :: dt=0.02, dx=0.2, tLimit=0.2, xLim(2)=(/0,1/)
  real    :: r
  integer :: nX, nT, i, j
  logical :: DoTest = .true.

  ! Create character variables.  We must declare their size.
  character(len=23) :: fmt1
  character(len=17) :: fmt2

  ! Domain array.  We don't know the size yet because we need to
  ! calculate that.  So let's make them "allocatable."
  real, allocatable :: Domain_II(:,:), xGrid(:)

  ! Constants.  Make them "parameters" that cannot be changed.
  integer, parameter :: iUnitFile=10
  real,    parameter :: cDiffusion = 1.0

  ! Now, begin execution section.
  !---------------------------------------------------------------------
  ! Write a message to Standard Out with no defined format:
  write(*,*) "Setting up simulation..."

  ! Calculate the number of points in X and in time.
  ! Ceiling rounds up and returns an integer, which matches the data
  ! type of "ceiling".  Without ceiling, our value would be a "real" type,
  ! which may be rounded up, down, or truncated (compiler dependent).
  nX = ceiling( (xLim(2)-xLim(1))/dx ) + 1
  nT = ceiling( tLimit/dt ) + 1

  ! If logical DoTest is .true., produce extra information to screen.
  if(DoTest)write(*,*) '     Grid size (nX, nT) = ', nX, nT

  ! Allocate arrays now that we know their size.
  ! Remember: if we do not de-allocate, it's possible to create a mem leak.
  allocate(Domain_II(nX, nT))
  allocate(xGrid(nX))

  !It's usually a good idea to fill matrices with zeros.
  Domain_II = 0
  xGrid     = 0

  ! Set the grid values and initial conditions:
  do i=1, nX
     xGrid(i)       = (i-1) * dx
     Domain_II(i,1) = 4.0*xGrid(i) - 4.0*xGrid(i)**2.0
  end do

  ! Check stability as described in class.
  ! "if () then" means >1 line after if statement.
  if ( dt > (dx**2.0 / (2.0*cDiffusion**2.0)) ) then
     write(*,*) 'ERROR!  WE ARE NOT STABLE!'
     stop  ! Remember, fortran's stop is not good for parallel programming.
  end if

  ! integrate.  See notes from class on the meaning below.
  write(*,*) 'Integrating...'
  r = cDiffusion**2.0 * dt / dx**2.0
  ! Loop from time 0 (j=1) to time t_final-deltaT.
  do j=1, nT-1
     Domain_II(2:nX-1, j+1) = (1.0 - 2.0*r) * Domain_II(2:nX-1, j) + &
          r*(Domain_II(1:nx-2,j) + Domain_II(3:nx, j))
  end do

  ! Now we want to write our results to file.
  write(*,*) 'Saving results to file.'

  ! Start by opening file in replace mode (over write existing file).
  ! Assign it to a file unit, iUnitFile.
  open(iUnitFile, file='results.txt', status='replace')

  ! Write a header line.  Our write statement now writes to our
  ! file unit and not "*" for standard out.  We also use format codes
  ! in place of our 2nd "*".
  write(iUnitFile, '(a)') 'Example 10.3 Results.'

  ! Write information about domain.  Note the format code that fills in
  ! brackets, commas, etc.
  write(iUnitFile, "(a,'[',f3.0,',',f4.0,'] ',a,f5.1,a)") &
       'Domain: x=',xLim, 't=[0.0,',tLimit,']'
  write(iUnitFile, "(a, i5.5, 'x',i5.5)") 'Domain size (x, Time) = ', nX, nT

  ! Our next format code depends on the size of our domain, which
  ! we won't know until run time.  So, we'll write the format code
  ! to a character variable of the right size:
  write(fmt1, "(a, i6.6, a)") '(a13,', nX, '(1x, E12.6))'
  if(doTest) write(*,*) 'fmt1 = ', fmt1
  ! Write grid to file:
  write(iUnitFile, fmt1) 'Grid:', xGrid

  ! Create format code for time and result lines:
  write(fmt2, "(a, i4.4,a)") '(', nX+1, '(1x, E12.6))'
  if(doTest) write(*,*) 'fmt2 = ', fmt2
  ! Loop over results and write to file.
  do j=1, nT
     write(iUnitFile, fmt2) (j-1)*dt, Domain_II(:,j)
  end do

  ! Close our file:
  close(iUnitFile)

  !Deallocate arrays.  ALWAYS DO THIS FOR ALLOCATABLE ARRAYS!
  deallocate(Domain_II)
  deallocate(xGrid)

  ! And that's it.
end program example
```