Programming Techniques 2024-2025

Lecture 3: More about data types

Hannu Parviainen

Universidad de la Laguna

October 2, 2024

Data type precision

- Fortran is used in very different computer architectures
  - Can use different number of bytes
- Many different ways to specify the precision of a data type
  - C: float, double, int, longint, etc...
  - Fortran: : integer(kind=x), real(kind=x)

  where x is an integer stating the precision... but the the meaning of x depends on the compiler...

Precision in GNU Fortran

- In GNU Fortran, ‘x‘ stands for the number of bytes used for the data type.
  - x = 1, 2, 4, 8, 16
- Defaults to:
  - 4 for logical, integer, real, and complex.
  - 8 for double precision.
  - 1 for character.
- But other compilers don’t necessiraly use the same logic!
- Don’t use the kind specification directly if you want your program to be portable!

In Gnu Fortran

```
real(kind=4) ! 32-bit single-precision float
real(kind=8) ! 64-bit double-precision float
integer(kind=4) ! 32-bit signed int
integer(kind=8) ! 64-bit signed int
```

Portability with SELECTED_KIND

- For portability, use:
  - SELECTED_INT_KIND(R) returns the kind value of the smallest integer type that can represent all values ranging from -10^R to 10^R
  - SELECTED_REAL_KIND(P, R) returns the kind value of a real data type with decimal precision of at least P digits and exponent range of at least R

```
program ex3a
  implicit none
  integer, parameter :: si = selected_int_kind(5)
  integer, parameter :: li = selected_int_kind(15)
  integer(kind=si) :: o
  integer(kind=li) :: p
  print *, huge(o), huge(p)
end program ex3a
```

Using ISO_FORTRAN_ENV

- The Fortran 2003 standard includes an intrinsinc `ISO_FORTRAN_ENV` module that allows you to specify the number of bits directly.
- Does not necessarily guarantee the desired precision, but provides control over the number of bits.
- Common types:
  - `int32`, `int64`
  - `real32`, `real64`

```fortran
program ex3b
  use iso_fortran_env
  implicit none
  integer(int32) :: i
  integer(int64) :: j
  real(real32) :: x
  real(real64) :: y
  print *, huge(i), huge(j)
  print *, tiny(x), huge(x)
  print *, tiny(y), huge(y)
end program ex3b
```

Using ISO_C_BIND

- The Fortran 2003 standard includes an intrinsinc `ISO_C_BIND` module that enables interoperability with C.
- This allows direct relation to C data types.
- Common types:
  - `c_float`
  - `c_double`

```
program ex3c
  use iso_c_bind
  implicit none
  real(c_float) :: x
  real(c_double) :: y
  print *, tiny(x), huge(x)
end program ex3c
```

Exercises

Exercise 1:

- Write a program that computes and prints the matrix multiplication of two real arrays.

$$A = \begin{pmatrix} 3 & 2 & 4 & 1 \\ 2 & 4 & 2 & 2 \\ 1 & 2 & 3 & 7 \end{pmatrix} \quad B = \begin{pmatrix} 3 & 2 & 4 \\ 2 & 1 & 2 \\ 3 & 0 & 2 \end{pmatrix}$$

Exercise 2:

- Write a program that reads two real arrays of length n and prints the sum of these arrays.

Exercise 3:

- Modify the matrix multiplication program to use a subroutine for the multiplication.