# Project 1.

Our first project serves as an introduction to Fortran 95, the role of modules, and initializing data through user-defined data structures.

An excellent example is the `atoms\_par` module in the `dbsr\_zcom.f90` library for ([DBSR\_HF](https://github.com/compas/DBSR\_HF)), a compas repository. The first few lines of this module are shown here.

!======================================================================

Module atoms\_par

!======================================================================

! atomic parameters according periodic table

!----------------------------------------------------------------------

Implicit none

Integer, parameter :: n\_atoms = 104

Type atomic

INTEGER :: an

CHARACTER(2) :: symbol

CHARACTER(4) :: core

CHARACTER(40) :: conf

REAL(8) :: weight

End type atomic

Type(atomic), dimension(n\_atoms), parameter, public :: atoms = (/ &

atomic( 1, 'H ', ' ', '1s(1)', 1.0079), &

atomic( 2, 'He', ' ', '1s(2)', 4.0026), &

atomic( 3, 'Li', '[He]', '2s(1)', 6.9410), &

atomic( 4, 'Be', '[He]', '2s(2)', 9.0122), &

atomic( 5, 'B ', '[Be]', '2p-(1)', 10.8110), &

atomic( 6, 'C ', '[Be]', '2p-(1)2p(1)', 12.0107), &

atomic( 7, 'N ', '[Be]', '2p-(1)2p(2)', 14.0067), &

atomic( 8, 'O ', '[Be]', '2p-(1)2p(3)', 15.9994), &

etc.

So `atoms` is an array whose elements are of type `atomic` and the module initializes this array.

In `GRASP` the finite nucleus is modelled in terms of parameters, namely the atomic number Z, and a mass number, A, although a user may override the values. An important parameter in nuclear physics is the "root-mean-square" (RRMS) of the radius. For Z < 91, W.R. Johnson, G.Soff, Atomic Data and Nuclear Data Tables {\bf 33}, p.405 (1985) showed that a good approximation was `RRMS = 0.836 A^(1/3) + 0.570` and this is the default in GRASP, but for many combinations of (Z,A) more accurate values have been determined. They have been tabulated by I. Angeli, K. P. Marinova, Atomic Data and Nuclear Data Tables {\bf 99}, 69-95 (2013). The program `RNUCLEUS` initializes a 2-dimensional array RRMSVEC(Z,A) and enters the non-zero values through assignment statements. The dimensions are RRMSVEC(120,300). Hence the program is exceedingly long. At the same time, the maximum number of mass values for a given Z is less than 30. A possible data type for each Z might be  
 Type A\_rrms

Integer :: a\_min

Real(8), Dimension(1:30) :: Z\_rrms

End Type A\_rrms

Where a\_min is the minimum value of A for the list of A-values. Then the list for the n\_atoms elements of the periodic table could be:

Type(atomic\_rrms), dimension(n\_atoms), parameter, public :: atoms\_rrms(/ &

A\_rrms( 1, Z\_rrms(1:3)=(/ 0.8783\_dp 2.1421\_dp, 1.6591\_dp/) ), &

A\_rrms( 3, Z\_rrms(1 :8)=(/1.9661\_dp, 1.6755\_dp, 0.0000\_dp, 2.0660\_dp, 0.0000\_dp, &

1.9239\_dp/) ), &

Etc.

Here we assume that Kind= ‘dp’ has been declared to be a Real double precision (64-bit) value.

The goal of this project is to derive a data type for each atomic number Z and develop a module to initialize an `atoms\_rrms` array, and assign values through a data statements for the data type. This program could then replace the `geniso.f90` program in GRASP. The easiest way might be to write a program that transforms the numbers in the assignment statements to data statements for the appropriate data type.