Symmetry Interfaces

Implemented simple symmetry class named FSymmetry
and matrix transformation functions with FSymmetry object
(pull from “symmetry” branch on github)

In “PhfSymmetry.h”

```
struct FSymmetry
{
    FORTINT
        PointGroup;
    FORTINT
        nRep;
    FORTINT
        nSO[8];
    TArray<FORTINT>
        SymUniqueList;
    TArray<FORTINT>
        TSymIndex;
};
```

In “PhfSymInterface.h”

```
void SymTransX(
    TArray<doulbe>& DataAO,
    TArray<double>& DataSO,
    const FSolidModel& Solid,
    const FSymmetry& sym);
```

Usage: Controlled by X

1: AO (Unit) -> SO (Unit)

2: AO (Unit) -> SO (Super)

3: AO (Super) -> SO (Super)

4: SO (Super) -> AO (Super)

Symmetry Interfaces

Also, PhfMain.cpp creates symmetry object from lattice information
So, people who wants to transform its matrix by lattice symmetry, just call SymTransX function with created symmetry object from main routine

Data structure of symmetry-adapted AOs is supposed to be,

```
TArray<double>::iterator
    itDataSO = DataSO.begin();

for ( uint iRep = 0; iRep < nRep; ++ iRep ) {
    for ( uint iSO = 0; iSO < nSO[iRep]; ++ iSO ) {
        for ( uint jSO = 0; jSO < nSO[iRep]; ++ jSO ) {
            MATRIX[iRep][iSO][jSO] = itDataSO ++;
        }
    }
}
```

I have implemented translational symmetry
but, haven’t yet implemented (crystal) point group symmetry

Next Hackathon (?)

Symmetry part of Periodic HF code:

- Inside structure of point group symmetry
- Space group symmetry of crystal to reduce sum over super-cell

Difference / Improvement for next time:

- Everyone should have the same part, or be divided into few parts (?)
  e.g. divided into 3 groups and discussed and coded the same theme but independently for each group

Preparation before next time:

- Tutorials for cross-language coding (?)
  e.g. how to use C++ classes/functions from FORTRAN or, how to use common block in FORTRAN from C/C++