Practical 2: GRIB API indexing

```
> cd $SCRATCH/practicals/exercise2
> cd F90 # or 'cd C' or 'cd Python'
> ls
Makefile  eps grib_api_index.f90    solution
```

- The file 'eps' contains EPS fields for all (50) (perturbed) ensemble members for some (4) parameters, e.g. see output of 'grib_ls eps'.
- The objective of this exercise is to write a Fortran, C code or a Python script using the GRIB API and the indexed access method to compute the ensemble mean for one parameter.
- The file grib_api_index.f90 (.c or .py) contains a skeleton of the code or script. Please complete it. If you are confident, you can start from 'scratch'.

Practical 2: Main program

```
program index
  use grib_api
  implicit none

  integer             :: iret
  integer,dimension(:),allocatable :: paramId,number
  integer                          :: paramIdSize,numberSize
  integer             :: i,j
  integer             :: idx,igrib,count,numberOfValues
  real (KIND=8),dimension(:), allocatable   :: values, result
```

2 index tables will be needed

two tables are defined for the data values.

- Compile/link with:

  Fortran: gfortran –o grib_api_index grib_api_index.f90 $GRIB_API_INCLUDE $GRIB_API_LIB

  C: gcc -o grib_api_index grib_api_index.c $GRIB_API_INCLUDE $GRIB_API_LIB -lm

  or use make.

Practical 2: GRIB API indexing

- Run the resulting code with:

```
> ./grib_api_index  # for Fortran or C
> python grib_api_index.py  # for Python
```

- Now change the link for the input file ‘eps’ to the grib2 file (also available from ~trx/grib_api/data) and run the program again.

```
> ln –fs ~trx/grib_api/data/eps.grib2 eps
```

Practical 3: GRIB API timings

```
> cd $SCRATCH
> cd practicals/exercise3
> ls

Makefile    ensmean_api_indexed.f90 ensmean_api_indexed_read.f90 run
dirs ensmean_api_reduced_grib_get.f90 ensmean_api.f90 input
> make
> ./run
>
```

- **The 4 fortran codes do the same thing. They all compute ensemble means and standard deviations with the GRIB API for 12 fields (4 parameters – 3 levels).**

Practical 3: GRIB API timings

- **The code in** ensmean_api.f90 **reads the complete grib file for each computation of a mean and std. It also decodes the data values even if a field is not used.**
- **The code in** ensmean_api_reduced_grib_get.f90 **is like the first code, but the data values for a field are decoded only when they are needed.**
- **The code in** ensmean_api_indexed.f90 **builds an index based on the keys parameter, ensemble number and level. The grib messages are then accessed through this index. The index is then saved into a file.**
- **The code in** ensmean_api_indexed_read.f90 **is exactly the same as the previous except that the index is read from a file, not built.**
- **Note the different run times! Beware of best access method for different access pattern:**
  - **Sequential i/o (grib_new_from_file) suitable for sequential access.**
  - **Indexed i/o (grib_new_from_index) suitable for random access.**

Practical 4: GRIB API encoding

```
> cd $SCRATCH
> cd practicals/exercise4
> cd F90 # or C or Python
> ls
Makefile eps grib_api_create.f90 solution
```

- The objective of this exercise is to extend the code used in practical 2 to create a new grib message containing the ensemble mean, using GRIB API.
- Different options are available to create a grib message:
  - Clone the new field to produce from one of the input grib fields.
  - Use a sample (or template) from the default samples directory. See 'grib_info'.
  - Use a sample from a private samples directory.

Practical 4: GRIB API encoding

- The first option is the easiest to implement. For simplicity, we suggest you to use this option.
- The file grib_api_create.f90 (or grib_api_create.c or grib_api_create.py) contains a skeleton of the code to create a grib message. Please can you try to add the code needed to create a grib message. Use ‘make’, ‘gfortran’ or ‘gcc’ to compile the codes, then run the program or run the Python script.
- Now change the link for the input file ‘eps’ to the grib2 file (also available from ~trx/grib_api/data) and run the program again.

```
> ln –s ~trx/grib_api/data/eps.grib2 eps
```

Practical 5: GRIB API grid packing

```
> cd $SCRATCH
> cd practicals/exercise5
> ls
eps.grib1  eps.grib2  pack_data  pack_data.cmd  pack_data.f90

> ./pack_data.cmd
```

- The objective of this exercise is to see the impact of different types of packing.
- For simplicity, we only look at some packing types for grid point data.
- Note that the timings may vary.
- Note that the grib_api may not do the packing requested. Check the packingType of output files with grib_ls.

Practical 5: GRIB API grid packing

- Which packing is the fastest, the slowest?
- Which packing does achieve the best compression'?
- Which packing types are not available for GRIB1?