GammaLib - Feature #610

Rethink FITS image classes

12/03/2012 09:42 PM - Knödlseder Jürgen

Status:	New	**Start date:**	12/03/2012
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Following Bug #609, FITS images are not really dealt with correctly. We may want to redesign the interface.

When opening a FITS image, the data will be converted automatically using the FITS interface. We should therefore distinguish the physical storage type on disk from the storage type in memory.

Ideally, we should design a class GFitsImage that can handle different storage types in memory and on disk. The latter seems trivial, the former is probably not trivial. An easy solution would be that the internal storage type is always double precision, and if not otherwise stated, the image is saved as such. However, if another physical disk storage type is selected, the class should perform the conversion as needed.

cfitsio supports the following image types:

```
#define BYTE_IMG    8 /* 8-bit unsigned
#define SHORT_IMG 16 /* 16-bit signed
#define LONG_IMG    32 /* 32-bit signed
#define LONGLONG_IMG 64 /* 64-bit signed
#define FLOAT_IMG -32 /* 32-bit single precision floating point */
#define DOUBLE_IMG -64 /* 64-bit double precision floating point */
```

The following 4 data type codes are also supported by CFITSIO:

```
#define SBYTE_IMG 10 /* 8-bit signed integers, equivalent to BITPIX = 8, BSCALE = 1, BZERO = -128 */
#define USHORT_IMG 20 /* 16-bit unsigned integers, equivalent to BITPIX = 16, BSCALE = 1, BZERO = 32768 */
#define ULONG_IMG 40 /* 32-bit unsigned integers, equivalent to BITPIX = 32, BSCALE = 1, BZERO = 2147483648 */
```

An alternative would be to have various pointers in GFitsImage for data storage, e.g.

```
long*   m_long;
double* m_double;
...
```

and that only one of these pointers is indeed used for memory storage. Depending on the memory storage used, corresponding code could be called. But maybe this is not really needed, as a double seems to be able to store all data types. Maybe some roundoff errors may occur for conversion to long long, but this needs to be checked.

In any case, one could always start with implementing a class that only holds data in double precision format, and extend the code later if really long long is needed.

History

#1 - 12/21/2012 12:40 AM - Knödlseder Jürgen

- Target version set to 00-08-00

#2 - 11/18/2013 08:50 PM - Knödlseder Jürgen

- Target version deleted (00-08-00)