compareChars(), compareFloats(), compareInts(), com-pareLongs(), compareShorts(), compareStrings()

SUMMARY Compare two primitive variables

Declared In: Foundation/FCCompare.h

SYNOPSIS

FCCompareType **compareFloats**(float *first*, float *second*)
FCCompareType **compareChars**(char *first*, char *second*)
FCCompareType **compareStrings**(const char **first*, const char **second*)
FCCompareType **compareShorts**(short *first*, short *second*)
FCCompareType **compareInts**(int *first*, int *second*)
FCCompareType **compareLongs**(long *first*, long *second*)

DESCRIPTION These functions compare two primitive type variables and return FC_COMPARE_LESS_THAN, FC_COMPARE_EQUAL_TO, or FC_COMPARE_GREATER_THAN

depending on whether first is less than, equal to, or greater than second, respectively.

You will most often use these functions when writing your own comparison method for the sortable FCCollection subclasses. For example, if you had a class called Employees and you wanted to be able to sort employees by the number of years they've been with the firm, you might write the following method:

```
- (FCCompareType) compareSeniority:comparisonObject
{
    return compareInts([self yearsWithFirm],
           [comparisonObject yearsWithFirm]);
}
```

If your collection is a subclass of FCSortedCollection, you could then call the `[employees setSortSelector: @selector(compareSeniority)]` and your collection would stay constantly sorted by seniority. Or, if the collection were a subclass of FCOrderedCollection, you could call `[employees sortByCompare: @selector(compareSeniority)]` and your collection would be sorted by seniority until you added another employee to it or otherwise changed the ordering.

See also: **-€ setSortSelector:** (FCSortedCollection), **-€ sortByCompare:** (FCOrderedCollection)