```
#ifndef PAIRS_H
#define PAIRS_H

class pairs_t {
        public:
        // constructors (overloaded)
        pairs_t(float ix=0.0, char ic='a') : \
                        x(ix), c(ic)                    { };

        // copy constructor
        pairs_t(const pairs_t& rhs);

        // destructors (default ok)
        //~pairs_t()                                    { };

        // friends -- note the extra <> telling the compiler to instantiate
        // a templated version of the operator<< -- <T> is also legal, i.e.,
        // friend std::ostream& operator<< <T>(std::ostream& s, const pairs_t&);
friend  std::ostream& operator<<(std::ostream& s, const pairs_t& rhs);
friend  std::ostream& operator<<(std::ostream& s, pairs_t *rhs)
                                                { return(s << (*rhs)); }

        // assignment operator
        const pairs_t&  operator=(const pairs_t&);

        // operators
        bool            operator<(const pairs_t& rhs);
        bool            operator>(const pairs_t& rhs);

        // member functions
        char            getc()                  { return c; }
        float           getx()                  { return x; }

        // private: only available to this class
        private:
        float           x;
        char            c;
};

#endif
```