Course C^{++}, Exercise Number 8

Deadline: 06.05.2014

The Fifteen Puzzle

The fifteen puzzle `http://en.wikipedia.org/wiki/15_puzzle`) was invented by Noyes Palmer Chapman in 1875. In the beginning of 1880, the puzzle became a craze, that lasted approximately half a year. (In 1981, the same effect was obtained by Rubik's cube.)

We will solve the 15-puzzle by the search algorithm that is described in the slides.

We implement the function F by an `unordered_map`, and the set U by a `priority_queue`.

1. Download the files in directory `fifteen` from the course homepage.

   Complete members of class `class fifteen`, and `operator == ( )` and `std::ostream& operator << ( std::ostream &, const fifteen& );`

   You have to make these operators friends.

   The method `unsigned int distance( ) const` is very important, because it will decide which element from U will be selected in the search algorithm. Wikipedia has some suggestions.

2. We want to implement f using an `unordered_map`. In order to do this, we need a hash object and a compare object. Just write in file **solve.cpp**:

```
namespace
{
   struct fifteenhash
   {
      inline unsigned int operator( ) ( const fifteen& f ) const
      { return f. hashvalue( ); }
   };

   struct fifteenequals
   {
      inline
      bool operator( ) ( const fifteen& f1, const fifteen& f2 ) const
```

```
      {
         return f1 == f2;
      }
   };
}
```

The functions are just placeholders which call the corresponding methods of **class fifteen**. The keyword **namespace** creates an anonymous namespace. This is the standard way in C^{++} to avoid that names in a **.cpp** file are visible from the rest of the program. Don't use the keyword **static** for this purpose!

When everything went well, you can now declare

```
std::unordered_map< fifteen, unsigned int, fifteenhash, fifteenequals > f;
```

Test that you can find and insert states.

3. Next we can turn our attention to the priority queue. We need a heuristic that 'decides' when a fifteen state is better than another state. The best way is to use the distance function:

```
namespace
{
   struct fifteenpriority
   {
      inline bool operator( ) ( const fifteen& f1, const fifteen& f2 ) const
      {
         return f1. distance( ) > f2. distance( );
            // Nearer to solution is more preferred.
      }
   };
}
```

4. Now it should be unproblematic to implement the main search algorithm. Actually, it is already written, but I didn't test it, so I give no guarantees. Test it, and correct it if necessary.

5. Finally, write a function

```
void printpath( const std::unordered_map< fifteen, unsigned int,
                fifteenhash, fifteenequals > & f,
                const fifteen& goal, unsigned int level,
                std::ostream& out )
```

which prints the path to goal. Use it to print a solution, in case a solution was found. You can also modify the function so that it returns a vector or list of moves, and print the result in the main function, if you think that that is better.

P.S. Note that half of the states of the fifteen puzzle has no solution.