VGP260 Assignment #3

W. Dobson 5-14-2018

Modify your character class implemented in assignment #1 to include a Finite State Machine (FSM) to track the character's behavior state. Your state machine can be of your own design but must have at least 5 states. Submit a drawing of your <u>FSM Bubble Diagram</u> and a <u>State Transition Table</u> with your source code. If needed these can be hand written on paper submitted in class.

- Your character class FSM should use a C++ enumerated class for the states you define.
  Example:
  `enum class Cstate {IDLE, ALERT, DEFEND, ATTACK, FLEE};`
  Note: the enum items will be mapped to integers: 0, 1, 2, 3, 4 in the order of the list.

- Your character class should include functions for managing and updating the character state.

1. **.state() -** returns the present state of your character
2. **.state_reset()** - resets back to the initial state (which may be "idle").
3. **.state_update(**var1, var2, var3**)** - function to update to the next state based on input variables you define to control your state machine.
4. **.state_set(** stvar **)** - sets the state to a user specified state (stvar).

- Modify your original read and write file functions to include the the character state using the type keyword "fsmstate" for parsing while storing and loading character files (think of saving the game state).

Use the item types when parsing (sorting) the character name, strength, and FSM state along with item data to the different sets in your character class. The item type keywords for processing:

item type	action
cname	initialize character name (*cname*) and *strength* at the top level
fsmstate	set the state of the character's FSM using the associated value
weapon, protect, healing, other	store in new item object and inserted into appropriate set

File format example: character.txt: Note: that by padding the fsmstate with an extra string "xxx" you can use your original *Item* class for reading the state information from the file (without extra coding).

```
cname      Boromir    560
fsmstate   xxx        0
weapon     sword      80
protect    helm       25
protect    chain-mail 48
weapon     dagger     18
healing    salve      10
other      horn       12
protect    shield     50
healing    elixir     18
```

Your `main()` program body should:

1. Include at least one instance of your character object.
2. Read the contents of a character text file and add settings and items to your character object.
3. Have code to exercise all of your character state control functions.
4. Have code to test all state transitions for your character's FSM printing state transitions to the screen.
5. Write the character object data (including the present state) to an output file.

Example of data Item class:

```
class Item{
      public:
      string      type,  // item type
                  name;  // item name
      int         value; // parameter for scoring
}
// Boolean operator overload of '<' needed for use with sets to sort set items.
// These are needed for any set using objects for elements.  Here it sorts by
the .name field.
bool operator <(Item const& a, Item const& b)
{
      return a.name < b.name;
}
```