```
/***GOOSE ESCAPE ACTORS HEADER FILE BEGINNING!!***/

#ifndef GOOSE_ESCAPE_ACTORS
#define GOOSE_ESCAPE_ACTORS
#include <cmath>
#include "BearLibTerminal_0.15.7\Include\C\BearLibTerminal.h"
#include "gooseEscapeUtil.hpp"

/*
    Modify this class to contain more characteristics of the "actor".  Add
    functions that will be useful for playing the game that are specific to
    the Actor.

    Feel free to add additional Classes to your program.
*/

/*
    Going further:  Learn the other syntax for implementing a class that is
    more appropriate for working with multiple files, and improve the class code.
*/

class Actor
{
  private:
    int actorChar;
    int location_x, location_y;
    // Move buffer to allow for control over actors speed
    int move_buffer;
    int max_buffer;

  public:
    Actor()
    {
        actorChar = int('A');
        location_x = MIN_SCREEN_X;
        location_y = MIN_SCREEN_Y;
        set_buffer(0);
        set_max(0);
        put_actor();
    }

    Actor(char initPlayerChar, int x0, int y0, int buffer_state, int speed)
    {
        change_char(initPlayerChar);
        location_x = MIN_SCREEN_X;
        location_y = MIN_SCREEN_Y;
        set_buffer(buffer_state);
        set_max(speed);
        update_location(x0,y0);
    }

    int get_x() const
    {
        return location_x;
    }

    int get_y() const
    {
        return location_y;
    }

    // Returns buffer
    int get_buffer() const
    {
        return move_buffer;
    }

    string get_location_string() const
    {
        char buffer[80];
        itoa(location_x,buffer,10);
        string formatted_location = "(" + string(buffer) + ",";
        itoa(location_y,buffer,10);
        formatted_location += string(buffer) + ")";
        return formatted_location;
    }

    void change_char(char new_actor_char)
    {
        actorChar = min(int('~'),max(int(new_actor_char),int(' ')));
    }

    void set_buffer(int buffer_state)
    {
        if(buffer_state < 0)
        {
            buffer_state = 0;
        }
        move_buffer = buffer_state;
    }

    void set_max(int max)
    {
        if(max < 0)
        {
            max = 0;
        }
        max_buffer = max;
    }

    // Increments buffer like a timer and resets at designated "speed"
    void increment_buffer()
    {
        move_buffer++;
        if(move_buffer > max_buffer)
        {
            set_buffer(0);
        }
    }

    bool can_move(int delta_x, int delta_y) const
    {
        int new_x = location_x + delta_x;
        int new_y = location_y + delta_y;

        return new_x >= MIN_BOARD_X && new_x <= MAX_BOARD_X
          && new_y >= MIN_BOARD_Y && new_y <= MAX_BOARD_Y;
    }

    void update_location(int delta_x, int delta_y)
    {
        if (can_move(delta_x, delta_y))
        {
            terminal_clear_area(location_x, location_y, 1, 1);
            location_x += delta_x;
            location_y += delta_y;
            put_actor();

        }
    }

    void put_actor() const
    {
        terminal_put(location_x, location_y, actorChar);
        terminal_refresh();
    }

    void setLocation(int x, int y)
    {
        this->location_x = x;
        this->location_y = y;
    }

};
#endif
```