cisc 1110 fall 2010

exercises for class V.2

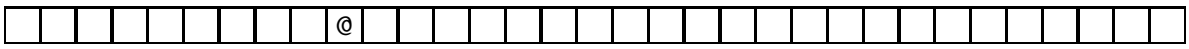
I. Modify the highlighted step from the first exercise given last class, which was:

- Write a program in which you initialize the random number generator and select a value between 1 and 100.
- Tell the user "I'm thinking of a number between 1 and 100. Can you guess what it is?"
- Then ask the user for a guess and read in the user's number.
- Compare the number the user entered with the random number that the user picked, and display a message telling the user either (a) "guess again, go higher!" or (b) "guess again, go lower!" or (c) "you got it!"
- **Put all this inside a loop, giving the user 5 chances to get the right answer**
- Compile and run your program to make sure it works.

If you completed this in the last class, you probably used a `for` loop and were unsure how to write your code so that if the user gets the right answer in less than 5 chances, the loop quits early—i.e., as soon as the user gets the right answer. Today we introduced **while** loops, and with a `while` loop, you can make one of the conditions for looping to be that the user has not guessed the right answer. Then, once the user does get the right answer, the loop will terminate.

So, modify your code from last class to use a `while` loop for the highlighted step (above).

II. The program on the next page simulates a sprite moving around in an imaginary 1-dimensional game space. The space might be envisioned something like this, where the sprite is represented with `@`:



Read the code on the next page, and then complete the steps outlined below.

1. What if the space that the sprite is wandering around in is not limitless? This means that the x values which indicates the sprite's location has limits, i.e, boundaries.

   Assume that the minimum possible value for x is -20 and the maximum possible value is $+20$.

   Create a program using the code on the next page, and modify it to make sure that the x value does go out of bounds. Be sure to give the user helpful information if she enters a number that would cause the sprite to go out of bounds.

2. Unlike physical spaces, in the virtual world, it is not uncommon for a sprite to be able to "wrap around". This means that if the sprite wanders to one end of its space (say position 20), and if it keeps going in the same direction, it will re-appear again at the other end (say position -20).

   Create another version of the program that implements the sprite's world as a wrap-around space.

3. Suppose that you wanted to include this as part of a bigger game, and you wanted the user to be able to enter something other than numbers—maybe you want the sprite to pick up something.

   Create another version of the program that lets the user enter any of the following character inputs (instead of the numeric input that you used in the previous two steps):

   F or f to go forward (increase x)

   B or b to go backward (decrease x)

   Q or q to quit the program (stop playing)

```
/**
   sprite.cpp
   14-oct-2010/sklar
*/

#include <iostream>
using namespace std;

int main() {
  //--- declare variables
  int x; // sprite's position
  int n; // user's input
  bool more; // does user want to play more?
  //--- initialize variables
  x = 0;
  more = true;
  //--- loop until user enters 0 to stop playing
  while ( more == true ) {
    cout << "the sprite is in position " << x << endl;
    cout << "how many spaces should the sprite move? (enter 0 to stop playing) ";
    cin >> n;
    cout << "you entered: " << n << "\n";
    if ( n == 0 ) {
      more = false;
    }
    else {
      x += n;
    }
  } // end while more
} // end of main()
```