RND(n) and RANDOMIZE

-
Alyce

Random Number Generator | Need for Random Numbers | Pseudorandom Number Generator | RND(n) | Seeding | RANDOMIZE | BIAS

Random Number Generator

An article on Wikipedia defines Random Number Generator as, "A random number generator (often abbreviated as RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random."

Need for Random Numbers

Some uses for random numbers are given in the Wikipedia article: "Some simple examples might be presenting a user with a "Random Quote of the Day", or determining which way a computer-controlled adversary might move in a computer game. Weaker forms of randomness are also closely associated with hash algorithms and in creating amortized searching and sorting algorithms."

The tutorial that ships with Liberty BASIC builds a "hi-lo" guessing game that relies on random number generation.

```
  'Here is an interactive HI-LO program
[start]
  guessMe = int(rnd(1)*100) + 1
  'Clear the screen and print the title and instructions
  cls
  print "HI-LO"
  print
  print "I have decided on a number between one"
  print "and a hundred, and I want you to guess"
  print "what it is.  I will tell you to guess"
  print "higher or lower, and we'll count up"
  print "the number of guesses you use."
  print

[ask]
  'Ask the user to guess the number and tally the guess
  input "OK.  What is your guess?"; guess
  'Now add one to the count variable to count the guesses
```

```
  let count = count + 1
  'check to see if the guess is right
  if guess = guessMe then goto [win]
  'check to see if the guess is too low
  if guess < guessMe then print "Guess higher."
  'check to see if the guess is too high
  if guess > guessMe then print "Guess lower."
  'go back and ask again
  goto [ask]

[win]
  'beep once and tell how many guesses it took to win
  beep
  print "You win!  It took"; count; "guesses."
  'reset the count variable to zero for the next game
  let count = 0
  'ask to play again
  input "Play again (Y/N)"; play$
  if instr("YESyes", play$) > 0 then goto [start]
  print "Press ALT-F4 to close this window."
  end
```

Pseudorandom Number Generator

Liberty BASIC's random number generator produces a pseudorandom number. According to Wikipedia, "A pseudorandom process is a process that appears to be random but is not."

Wikipedia describes a pseudorandom number generator, "A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG),[1] is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state, which may include a truly random seed. "

RND(n)

Description:
This function returns a random number between 0 and 1. The number parameter is usually set to 1, but the value is unimportant because it is not actually used by the function. The function will always return an arbitrary number between 0 and 1.

You can obtain a random number within a desired range by mathmatically altering the value returned by RND(). The following example from the helpfile creates a random number between one and ten by multiplying the result by 10, which gives answers in the range of 0-9. It then adds 1 to make the range

1-10.

Usage:

```
'print ten numbers between one and ten
for a = 1 to 10
  print int(rnd(1)*10) + 1
next a
```

Seeding

Some languages require a seed (starting number) for random number generation. In QBasic, one used the following method to seed the random number generator. The seed was the CPU's timer value for the number or seconds since midnight.

```
RANDOMIZE TIMER
```

Liberty BASIC does not require a seed.

RANDOMIZE

Some languages require a random seed to produce a random number. That is NOT true of Liberty BASIC. The RANDOMIZE statement in Liberty BASIC is used to seed the random number generator so that it produces the SAME sequence of numbers each time it is run. This is helpful for testing and debugging.

The RANDOMIZE statement is optional.

Syntax:

```
RANDOMIZE n
```

Description:

This function seeds the random number generator in a predictable way. The seed numbers must be greater than 0 and less than 1. Numbers such as 0.01 and 0.95 are used with RANDOMIZE.

Usage:

```
'this will always produce the same 10 numbers
randomize 0.5
for x = 1 to 10
  print int(rnd(1)*100)
next x
```

BIAS

When run many times, any pseudorandom number generator will display a bias. The bias displayed by Liberty BASIC's RND() is slight, but in rare instances it might be significant. If that is the case, methods can be used to further randomize results.

One improvement is the pseudorandom number generator called the Mersenne Twister

Chris Iverson - - ChrisIverson - has created a DLL that uses the Mersenne Twister. There are two versions of the DLL by Chris.

The first version of his DLL depends on the Microsoft Visual C++ 2008 SP1 Redistributable Package (x86), which is available for Windows XP and higher. It might have been installed already on your system by another program.
Mersenne Twister DLL

MersTwistLB.zip

- Details
- Download
- 2 MB

The GCC version of his DLL depends on the MSVCRT.DLL and might not work on systems prior to Windows XP.
GCC version of Mersenne Twister DLL

MersTwistLB-GCC.zip

- Details
- Download
- 10 KB

A version compiled with Pelles C by - StPendl does not depend on any additional DLL and should work on any Windows version.
Pelles C version of Mersenne Twister DLL