

## Re: random number problem

**Source:** <http://coding.derkeiler.com/Archive/General/comp.programming/2004-07/0944.html>

---

**From:** Paul Hsieh (*qed\_at\_pobox.com*)

**Date:** 07/12/04

Date: 12 Jul 2004 03:17:15 -0700

"cpx" <invalid@invalid.com> wrote:  
> <Jens.Toerring@physik.fu-berlin.de> schrieb:  
> > cpx <invalid@invalid.com> wrote:  
> > [snip]  
> > > *random\_int = (rand() \* max) + min*  
> >  
> > > *And I think this additional step destroys the uniformity..*  
> >  
> > > *Why should it do that? It's a simple linear transformation.*  
> > > *If you have uniformly distributed points drawn on an elastic*  
> > > *band and then stretch it they stay uniformly distributed.*  
> > > *And a simple multiplication does nothing else (and adding an*  
> > > *offset doesn't change anything).*  
> >  
> > > *Really? Well, I guess you're right because I really don't*  
> > > *know sh\*t about mathematics (I'm just a hobby coder)*  
> > > *and you sound like you know what you're talking about.*

First of all, to have this nice linear property you really do need a floating point random number generator. The C language doesn't have one of those. If you do the integer transformation as suggested (whether or not you use the upper bits) your distribution will be wrong if it does not divide evenly into (RAND\_MAX + 1).

On my C compiler RAND\_MAX is defined to the very tiny number: 32767. Since you say you shy away from math, lets just prove it with a program:

```
=====
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#ifdef USE_MERSENNE_TWISTER
#include <limits.h>
```

comp.programming: Re: random number problem

```
#include "mt.h"

void srand_TEST (unsigned long s) {
    init_genrand (s);
}

unsigned long rand_TEST (void) {
    return genrand_int32 ();
}

#define RAND_MAX_TEST ULONG_MAX

#else

#define srand_TEST srand
#define rand_TEST rand
#define RAND_MAX_TEST RAND_MAX

#endif

#define AVERAGE_SAMPLES_PER_BUCKET (1000)

int main () {
    int i, d = 0, n, minb, maxb;
    static unsigned long * bucket;
    double scale, a2;

    scanf ("%d", &d);
    if (d <= 0) {
        fprintf (stderr, "Enter a positive integer\n");
        exit (-1);
    }

    scale = d / (RAND_MAX_TEST + 1.0);

    bucket = (unsigned long *) malloc (sizeof (unsigned long) * d);
    for (i=0; i < d; i++) {
        bucket[i] = 0;
    }

    srand_TEST (time (NULL) + clock () + d);

    n = d * AVERAGE_SAMPLES_PER_BUCKET;

    for (i=0; i < n; i++) {
        int r = rand_TEST () * scale;
        bucket[r]++;
    }

    /* Some statistical tests */
}
```

comp.programming: Re: random number problem

```
minb = n;
maxb = 0;

a2 = 0;
for (i=0; i < d; i++) {
    if (minb > bucket[i]) minb = bucket[i];
    if (maxb < bucket[i]) maxb = bucket[i];
    a2 += ((double)bucket[i]) * bucket[i];
}

a2 = a2 / d - (AVERAGE_SAMPLES_PER_BUCKET*AVERAGE_SAMPLES_PER_BUCKET);

printf ("min bucket: %d\n", minb);
printf ("max bucket: %d\n", maxb);
printf ("last bucket: %d\n", bucket[d-1]);
printf ("variance: %g (supposed to be about %d)\n", a2,
        AVERAGE_SAMPLES_PER_BUCKET);

free (bucket);

return 0;
}
```

=====

For most small numbers (less than 1000) you enter, when you run the program, the important number that it outputs, the variance, is around 1000, which is what theory says it should be for a uniform distribution. But as soon as you start trying out numbers in excess of 1000 you will see some disturbing results in the variance (actually it depends on how close the number is to one substantially divisible into 32768 — compare the result of 4096 with 4000). It just increases and increases, until you approach the value of RAND\_MAX itself, when it finally converges back to about 1000. Numbers beyond RAND\_MAX are clearly useless.

If you want a \*serious\* random number generator, without thinking too hard about it, use the "Mersenne Twister" which can be located here:

<http://www.math.sci.hiroshima-u.ac.jp/%7Em-mat/MT/emt.html>

It passes the test above for much larger integer values, as well as coming with a real number random generator. If you want to \*study\* random number generators, I suggest you do a search in google groups for "George Marsaglia".

--  
Paul Hsieh  
<http://www.pobox.com/~qed/>  
<http://bstring.sf.net/>