Subject: random functions proposal
Posted by ratah on Mon, 16 Jan 2012 16:06:00 GMT
View Forum Message <> Reply to Message

Hello everybody,

I looked for a random function between an interval (a,b) and did not find (maybe I am so pressed). So, I decide to write these 2 functions c_random_a_b and upp_random_a_b for you.

```
#include <Core/Core.h>

using namespace Upp;

int putBetween(const int x1, const int x2, const int x3)
{
 int copie_x1 = x1;
 int copie_x2 = x2;



 // Put x3 between x1 et x2

 int d1 = x3 - copie_x1;
 int d2 = x3 - copie_x2;

 if(copie_x2 < copie_x1)
 {
  int tmp = copie_x2;

  copie_x2 = copie_x1;
  copie_x1 = tmp;
 }

 if(x3 >= copie_x1 && x3 <= copie_x2) return x3;

 if(d1 == 0) d1 = (copie_x2-copie_x1)/2;

 double rapport =  100*cos(d1+d2)*(copie_x2-copie_x1);
 if(rapport < 0) rapport *= -1;

 double dnewx3 = (100*copie_x1 + rapport)/100;
 int newx3 = (int) round(dnewx3);

 return newx3;
}

int c_rand_a_b(int x1, int x2)
```

```
{

 // Using standard C function rand()

 if(x1 == x2)
  return x1;

 int x3 = rand();

 int newx3 = putBetween(x1, x2, x3);

 return newx3;
}


int upp_rand_a_b(int x1, int x2)
{

 // Using U++ function Random()

 if(x1 == x2)
  return x1;

 String s3;

 dword dw3 = Random();
 s3 << dw3;

 int x3 = StrInt(s3);

 int newx3 = putBetween(x1, x2, x3);

 return newx3;
}


CONSOLE_APP_MAIN
{
 Cout() << "Random function proposal"; Cout().PutEol();

 int a, b;
 String sa, sb;

 do
 {
  Cout().PutEol();
  Cout() << "Enter a number named \"a\":";
```

```
  sa = ReadStdIn();

  a = StrInt(sa);
 }
 while(a<0);

 do
 {
  Cout().PutEol();
  Cout() << "Enter another number different of " << a << " named \"b\":";

  sb = ReadStdIn();

  b = StrInt(sb);
 }
 while(b<0 || b==a);

 Cout().PutEol();
 Cout() << "--------------------------";

/// Proposal 1: c_rand_a_b(a,b)

   // To use c_rand_a_b correctly, you might reset srand each time you reexecute the program
   // Otherwise you get the same sequential data

   Time t = GetSysTime();
   srand((unsigned int) t.second*1000);

 Cout().PutEol();
 Cout() << "Using standard C rand() function : c_rand_a_b(" << a << ", " << b << ") ";
 Cout().PutEol();
 for(int i=0;  i<10; i++)
 {
  Cout() << c_rand_a_b(a, b) << " ";
 }

/// Proposal 2: upp_rand_a_b(a,b)

 Cout().PutEol();
 Cout() << "Using U++ Random() function : upp_rand_a_b(" << a << ", " << b << ")";
 Cout().PutEol();
 for(int i=0;  i<10; i++)
 {
  Cout() << upp_rand_a_b(a, b) << " ";
 }

 Cout().PutEol();
}
```

If similar function does not exist yet, could you add this to UPP Core.

Thank you

Subject: Re: random functions proposal
Posted by dolik.rce on Mon, 16 Jan 2012 16:58:31 GMT
View Forum Message <> Reply to Message

Hi Ratah,

I have few comments

First: I don't like the implementation of putBetween function. Cos() is too slow function to use for scaling. Also I admit I haven't really got the idea behind the scaling, as I was too lazy to think about it too much.

Second: Converting dword -> int using String is not a good idea It will silence the compiler warnings, but still result into errors, because dword is unsigned 32 bit integer while int is signed 32bit integer (on most platforms), so the conversion will run into trouble when the dword is bigger then INT_MAX. Also the conversion through string is not efficient. In most cases you can probably just use simple cast: int(my_dword) and possibly check if the result is not negative.

Anyway short and simple solution to generate random number from given range is this:int min=10;
int max=1000;
int rand = int(Random(max-min)+min); // you don't have to care about the range here as long as max < INT_MAX

I hope I haven't made any stupid mistake, as this all comes from the top of my head

Best regards,
Honza

Subject: Re: random functions proposal
Posted by Sender Ghost on Mon, 16 Jan 2012 17:31:15 GMT
View Forum Message <> Reply to Message

Hello, ratah, dolik.rce.

ratah wrote on Mon, 16 January 2012 17:06
I looked for a random function between an interval (a,b) and did not find.

dolik.rce wrote on Mon, 16 January 2012 17:58

you don't have to care about the range here as long as max < INT_MAX

There is another implementation:

```
#include <Core/Core.h>

using namespace Upp;

// Returns Random from a to b, inclusively.
dword Random(dword a, dword b)
{
 if (a == b)
  return a;
 if (a < b)
  return a + Random(b - a + 1);
 return b + Random(a - b + 1);
}

CONSOLE_APP_MAIN
{
 const int n = 100;
 for (int i = 1, j = n; i <= n; ++i, --j)
  Cout() << Format("%3d, %3d: %s\n", i, j, AsString(Random(i, j)));
}
```

Subject: Re: random functions proposal
Posted by ratah on Tue, 17 Jan 2012 08:56:39 GMT
View Forum Message <> Reply to Message

Thank you for your reply,

Dolik.rce, I aggree with you for the use of cos and dword conversion. The idea of scaling is because I do not know it is possible to pass a parameter to Random(). So I could not limit the max value I get with Random().

Dolik.rce, Sender Ghost, all of your solutions are OK.
Why not to introduce that usefull function into Core?

Best regards,

Ratah

Subject: Re: random functions proposal
Posted by mirek on Tue, 17 Jan 2012 10:13:57 GMT

ratah wrote on Tue, 17 January 2012 03:56Thank you for your reply,

Dolik.rce, I aggree with you for the use of cos and dword conversion. The idea of scaling is because I do not know it is possible to pass a parameter to Random(). So I could not limit the max value I get with Random().

Dolik.rce, Sender Ghost, all of your solutions are OK.
Why not to introduce that usefull function into Core?

Best regards,

Ratah

I think that with Random(n), it is really not much useful to add second parameter.

BTW, what IS actually missing is double Randomf(); (returning 0 < x < 1) - but with all bells and whistles... which might be a little bit harder than it seems to be done correctly.

Subject: Re: random functions proposal
Posted by Sender Ghost on Tue, 17 Jan 2012 11:58:23 GMT

Hello, Mirek.
mirek wrote on Tue, 17 January 2012 11:13
BTW, what IS actually missing is double Randomf(); (returning 0 < x < 1)

Something like this?

```
// Returns random value from 0 to 1 - 10^-9 with 10^-9 precision.
double Randomf()
{
 static const dword n = 1000000000;
 return double(Random(n)) / n;
}
```

Subject: Re: random functions proposal
Posted by koldo on Tue, 17 Jan 2012 12:10:41 GMT

Hello all

I think it would be good to have a double Randomf() function.

However random numbers have to be treated very seriously. It would be necessary to have a reliable Randomf() so IMHO an implementation based in dword Random() would not be good.

Subject: Re: random functions proposal
Posted by mirek on Tue, 17 Jan 2012 12:59:41 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Tue, 17 January 2012 06:58Hello, Mirek.
mirek wrote on Tue, 17 January 2012 11:13
BTW, what IS actually missing is double Randomf(); (returning 0 < x < 1)

Something like this?

```
// Returns random value from 0 to 1 - 10^-9 with 10^-9 precision.
double Randomf()
{
 static const dword n = 1000000000;
 return double(Random(n)) / n;
}
```

Yes, but that 10^-9 is perahps not enough. We have 56 bits to fill...

Sure, it is enough for most applications, but so far I believe not good enough as addition to library. Especially if it is such one-liner.

Subject: Re: random functions proposal
Posted by mirek on Tue, 17 Jan 2012 13:24:17 GMT
View Forum Message <> Reply to Message

...but it looks like int64 -> double conversion is CPU opcode, so perhaps Random64(int n) as prerequisite, than Randomf() is a good path...

Subject: Re: random functions proposal
Posted by dolik.rce on Tue, 17 Jan 2012 17:45:43 GMT
View Forum Message <> Reply to Message

Looking for the details of floating point RNGs, I found this: http://allendowney.com/research/rand/. This pseudo scientific paper seems to propose reasonably simple algorithm that overcomes the obvious problems of the implementation Sender Ghost proposed. Only problem I see that the code linked on this page is GPL3 licensed... But even if we don't use it, the explanation about why using Random()/RAND_MAX is bad is worth reading

Subject: Re: random functions proposal
Posted by Sender Ghost on Tue, 17 Jan 2012 19:08:14 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 17 January 2012 18:45This pseudo scientific paper seems to propose reasonably simple algorithm that overcomes the obvious problems of the implementation Sender Ghost proposed.

Well, what I "proposed" (and I didn't propose, but just showed a simple solution) is useful for float numbers, not double, hence this Randomf, instead of Randomd, I think.
With qword Random64(qword n) it will be possible to do the same, of course.

mirek wrote on Tue, 17 January 2012 14:24...but it looks like int64 -> double conversion is CPU opcode, so perhaps Random64(int n) as prerequisite, than Randomf() is a good path...
According to "Random number generators discussion" it is possible to combine two random dword values (MAKEQWORD macro might be useful here) to get random qword value.

```
qword Random64()
{
 return MAKEQWORD(Random(), Random());
}

qword Random64(qword n)
{
 qword mask = n, r;
 mask |= mask >> 1; mask |= mask >> 2;
 mask |= mask >> 4; mask |= mask >> 8;
 mask |= mask >> 16; mask |= mask >> 32;

 do
  r = Random64() & mask;
 while(r >= n);
 return r;
}

qword Random64(qword a, qword b)
{
 if (a == b)
  return a;
 if (a < b)
  return a + Random64(b - a + 1);
 return b + Random64(a - b + 1);
}
```

Edit: Added possible Random64 implementation(s).

Subject: Re: random functions proposal
Posted by dolik.rce on Tue, 17 Jan 2012 19:25:32 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Tue, 17 January 2012 20:08
Well, what I "proposed" (and I didn't propose, but just showed a simple solution) is useful for float numbers, not double, hence this Randomf, instead of Randomd, I think.
With qword Random64(qword n) it will be possible to do the same, of course.

mirek wrote on Tue, 17 January 2012 14:24...but it looks like int64 -> double conversion is CPU opcode, so perhaps Random64(int n) as prerequisite, than Randomf() is a good path...
According to "Random number generators discussion" it is possible to combine two random dword values (MAKEQWORD macro might be useful here) to get random qword value.
The type doesn't really matter, I referred to "float" just as a sloppy abbreviation for "floating point type". The problem is that even with qword the simple division will not generate uniform distribution because floating point numbers are not equally spaced (as is explained in the document I linked to).

Honza

Subject: Re: random functions proposal
Posted by Sender Ghost on Tue, 17 Jan 2012 19:37:46 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 17 January 2012 20:25The type doesn't really matter, I referred to "float" just as a sloppy abbreviation for "floating point type". The problem is that even with qword the simple division will not generate uniform distribution because floating point numbers are not equally spaced (as is explained in the document I linked to).
The article shows, that there is more range for float numbers, than just using sufficient range. I agree, there is more range. Therefore, the function has limitations, which described in the comment for it.

Subject: Re: random functions proposal
Posted by cbpporter on Wed, 18 Jan 2012 14:44:52 GMT
View Forum Message <> Reply to Message

What Random needs is the ability to give it a seed. So you can produce the same sequence of numbers twice. Or is there already support for this?

Subject: Re: random functions proposal
Posted by dolik.rce on Wed, 18 Jan 2012 16:45:45 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 18 January 2012 15:44What Random needs is the ability to give it a seed. So you can produce the same sequence of numbers twice. Or is there already support for this?
These functions should be able to do that:void SeedRandom(dword *seed,int len){
 if(!sRng) {
  sRng = new(sRb) MTrand;
 }
 sRng->init_by_array(seed,len);
}

void SeedRandom(dword Seed){
 if(!sRng) {
  sRng = new(sRb) MTrand;
 }
 sRng->init_genrand(Seed);
}
But I didn't have the time to test it

Honza

Subject: Re: random functions proposal
Posted by mirek on Wed, 18 Jan 2012 18:24:13 GMT
View Forum Message <> Reply to Message

Random64 and SeedRandom now in Core.

I have also added "trivial" implementation of Randomf.

Mirek