randomWH API

Generated by Doxygen 1.8.4

Wed Feb 26 2014 16:14:05

Contents

Chapter 1

A library for generating uniformly distributed random variates

This implements the algorithm described by Wichmann and Hill in

- B.A. Wichmann and I.D. Hill, "Generating good pseudo-random numbers", *Computational Statistics & Data Analysis*, vol. 51, no. 3., pp. 1614-1622, Dec. 2006.

Two implementations are provided. One is for compilers that use 32 bits to store variables of the `long int` type. A faster algorithm is also provided for use with compilers that feature a 64-bit `long int` type. The algorithm to be used is determined automatically at run time.

The 64-bit version performs the following calculations:

$$
\begin{aligned}
x &\leftarrow (11600 \times x) \bmod 2147483579 \\
y &\leftarrow (47003 \times y) \bmod 2147483543 \\
z &\leftarrow (23000 \times z) \bmod 2147483423 \\
t &\leftarrow (33000 \times t) \bmod 2147483123 \\
W &\leftarrow \frac{x}{2147483579} + \frac{y}{2147483543} + \frac{z}{2147483423} + \frac{t}{2147483123} \\
W &\leftarrow W - \lfloor W \rfloor \\
&\text{return } W
\end{aligned}
$$

The 32-bit version calculates the same sequence, but as follows:

$$x \leftarrow 11600 \times (x \bmod 185127) - 10379 \times \left\lfloor \frac{x}{185127} \right\rfloor$$
$$y \leftarrow 47003 \times (y \bmod 45688) - 10479 \times \left\lfloor \frac{y}{45688} \right\rfloor$$
$$z \leftarrow 23000 \times (z \bmod 93368) - 19423 \times \left\lfloor \frac{z}{93368} \right\rfloor$$
$$t \leftarrow 33000 \times (t \bmod 65075) - 8123 \times \left\lfloor \frac{t}{65075} \right\rfloor$$
$$x \leftarrow \begin{cases} x & \text{if } x \geq 0 \\ x + 2147483579 & \text{if } x < 0 \end{cases}$$
$$y \leftarrow \begin{cases} y & \text{if } y \geq 0 \\ y + 2147483579 & \text{if } y < 0 \end{cases}$$
$$z \leftarrow \begin{cases} z & \text{if } z \geq 0 \\ z + 2147483579 & \text{if } z < 0 \end{cases}$$
$$t \leftarrow \begin{cases} t & \text{if } t \geq 0 \\ t + 2147483579 & \text{if } t < 0 \end{cases}$$
$$W \leftarrow \frac{x}{2147483579} + \frac{y}{2147483543} + \frac{z}{2147483423} + \frac{t}{2147483123}$$
$$\text{return } W - \lfloor W \rfloor$$

Remarks

- v 0.2 alpha - first public release
- Changes for v 0.3 alpha
  - renamed `congruence` to Congruence
  - Changed `Congruence` struct initialisation to avoid "sorry, unimplemented: non-trivial designated initializers not supported" error in g++ (C++, unlike C, does not support designated initializers - see `here`).
  - Changed randomWH_error() `message` argument from `char*` to `const char*`

  Code will now compile without error or warnings in g++ as well as gcc (v2.8.1)

Copyright

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 congruent Struct Reference

An instance of the `Congruence struct`.

`#include <randomWH.h>`

4.1.1 Detailed Description

An instance of the `Congruence struct`.

The four seeds used by the generator are stored in this `struct`. Because they must have global scope, they are contained within a structure to minimise the chances of a name conflict with other variables of global scope. The functions to process these values are accessed via function pointers contained within the struct. This simulates the member function concept of C++ using only C code. Most applications will require only a single generator instance - this one has safe values assigned to the member function pointers.

Usage

```
#include <stdio.h>
#include "randomWH.h"
#define SIM_DURATION 40000000L
.
.
.
 double mean=0, moment2=0, unif, variance;
    long int count;
 congruent.init(&congruent,1L,2L,3L,4L);
    for(count =0;count<SIM_DURATION;count++){
        unif = congruent.randomWH(&congruent);
        mean += unif;
        moment2 += (unif*unif);
    }
    mean /= SIM_DURATION;
    moment2 /= (SIM_DURATION-1.0); // using Bessel's correction
    variance = moment2-mean*mean*(SIM_DURATION)/(SIM_DURATION-1.0); // Bessel's correction
       continued
    printf("Mean after %ld iterations is %lf; sample variance is %lf\n", SIM_DURATION, mean, variance);
```

See Also

Congruence

Note

You may create other generators by declaring other structs of type `Congruence` but you **must** initialise them correctly.

The documentation for this struct was generated from the following file:

- randomWH.h

4.2 generator Struct Reference

`Typedef` of `struct` to store and process the four generator seeds.

`#include <randomWH.h>`

Data Fields

Variables

- long int x
- long int y
- long int z
- long int t
- enum BOOLEAN initialised

Member Functions

- void(∗ init)(struct generator ∗self, long int x, long int y, long int z, long int t)
  Initialise the generator with four values of type `long int`.
- double(∗ randomWH)(struct generator ∗self)
  Generate a (pseudo-)random number drawn from the U(0,1) distribution.
- long int ∗(∗ read)(struct generator ∗self)
  Read the current state of the pseudo-random number generator.
- enum BOOLEAN(∗ bernoulli)(struct generator ∗self, double prob)
  Generate a binomially distributed random variate.
- long int(∗ geometric)(struct generator ∗self, double prob)
  Generate a geometrically distributed random variate.
- long int(∗ binomial)(struct generator ∗self, long int trials, double prob)
  Generate a binomially distributed random variate.
- long int(∗ fairdie)(struct generator ∗self, long int lower_inclusive, long int upper_inclusive)
  Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

4.2.1 Detailed Description

`Typedef` of `struct` to store and process the four generator seeds.

In ordinary usage, you will not use this typedef directly. Instead, you will use the instance of it called `Congruence` which is pre-initialised in randomWH.c. An exception would be if your code requires multiple instances of the random number generator. A second instance can be generated using the following code (failure to initialise this correctly will almost always cause fatal errors at run time).

```
#include "randomWH.h"


Congruence generator2 = {
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
```

This generator can be accessed using `generator2.init()`, `generator2.read()`, etc.

If multiple generators are instantiated, the number sequences generated may overlap; this is *certain* to occur if two or more are seeded with the same values.

See Also

congruent

Definition at line 147 of file randomWH.h.

4.2.2 Field Documentation

4.2.2.1 enum BOOLEAN(∗ bernoulli)(struct generator ∗self, double prob)

Generate a binomially distributed random variate.

Usage

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
        if (generatorX.bernoulli(&generatorX,0.5)==istrue)
         printf("Heads!\n");
     else
         printf("Tails!\n");
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success

Returns

a BOOLEAN variable indicating an outcome of success or failure

Definition at line 324 of file randomWH.h.

4.2.2.2 long int(∗ binomial)(struct generator ∗self, long int trials, double prob)

Generate a binomially distributed random variate.

Usage

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
```

```
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
long int wheelspin;
.
.
.
        wheelspin = generatorX.binomial(&generatorX,100,1.0/38.0)
     printf("My (un)lucky number came up %ld times in 100 spins of a roulette wheel\n", wheelspin);
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
trials	The number of trials
prob	The probability of success in each trial

Definition at line 395 of file randomWH.h.

4.2.2.3 long int(∗ fairdie)(struct generator ∗self, long int lower_inclusive, long int upper_inclusive)

Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

Usage

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
long int lotto;
.
.
.
     lotto = generatorX.fairdie(&generatorX,1L,45L)
     printf("The first Lotto number drawn was %ld\n", lotto);
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
lower_inclusive	the lowest value of variate that can be returned
upper_inclusive	the highest value of variate that can be returned

Definition at line 432 of file randomWH.h.

4.2.2.4 long int(∗ geometric)(struct generator ∗self, double prob)

Generate a geometrically distributed random variate.

Usage

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
```

```
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
long int patience;
.
.
.

           patience = generatorX.geometric(&generatorX,1.0/52.0)
           printf("It took %ld draws before my card appeared\n", patience);
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success in each trial

Definition at line 360 of file randomWH.h.

4.2.2.5 void(∗ init)(struct generator ∗self, long int x, long int y, long int z, long int t)

Initialise the generator with four values of type `long int`.

We need to populate the `congruent` structure with seed values before calling `randomWH()`. This function isolates the initialisation from the implementation.

Usage

```
 #include "randomWH.h"

Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
     generatorX.init(&generatorX, 1L,2L,3L,4L);
```

Parameters

x	The seed for the first generator
y	The seed for the second generator
z	The seed for the third generator
t	The seed for the fourth generator
self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Note

Initialising the generator with the same seeds from program run to program run will produce *exactly the same sequence* . Usually in simulation programs (particularly when testing and debugging them), this is **exactly what you want**.

Definition at line 198 of file randomWH.h.

4.2.2.6 enum BOOLEAN initialised

initialised to false; set to true when function pointers updated

Definition at line 155 of file randomWH.h.

4.2.2.7 double(* randomWH)(struct generator *self)

Generate a (pseudo-)random number drawn from the U(0,1) distribution.

Usage

```
#include <stdio.h>
#include "randomWH.h"
#define SIM_DURATION 40000000L
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
        long int count;
        double mean=0, moment2=0, unif, variance;
        generatorX.init(&generatorX,1L,2L,3L,4L);
        for(count =0;count<SIM_DURATION;count++){
            unif = generatorX.randomWH(&generatorX);
            mean += unif;
            moment2 += (unif*unif);
        }
        mean /= SIM_DURATION;
        moment2 /= (SIM_DURATION-1.0); // using Bessel's correction
        variance = moment2-mean*mean*(SIM_DURATION)/(SIM_DURATION-1.0); // Bessel's correction
       continued
        printf("Mean after %ld iterations is %lf; sample variance is %lf\n", SIM_DURATION, mean, variance);
```

Returns

a `double` value drawn (to a good approximation) from a uniform distribution in the range [0,1]

See Also

Congruence_init()
Congruence_read()
http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y

Warning

Never use a random number generator that you have not tested extensively first. Never use a generator if you do not know and understand the algorithm it implements.

Note

Initialising the generator with the same seeds from program run to program run will produce *exactly the same sequence* . Usually in simulation programs (particularly when testing and debugging them), this is **exactly what you want**.

Definition at line 248 of file randomWH.h.

4.2.2.8 long int*(* read)(struct generator *self)

Read the current state of the pseudo-random number generator.

Usage

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
        long int* read_con;
        int count;
        read_con=generatorX.read(&generatorX);
        for(count=0;count<4;count++)
            printf("%ld\n", read_con[count]);
```

Returns

A pointer to an array of **four** `long int` values .

See Also

init()

Note

This function can be used immediately prior to program shutdown to read the current seed values prior to saving them in persistent storage. Loading these values from persistent storage during program initialisation and using them as the arguments in a call to the relevant .init() will allow execution to resume exactly as if the previous program run had continued.

Warning

Only **four** values are returned. Do **not** use an index greater than 3 to access the array.

Definition at line 289 of file randomWH.h.

4.2.2.9 long int t

The seed for the fourth generator

Definition at line 154 of file randomWH.h.

4.2.2.10 long int x

The seed for the first generator

Definition at line 151 of file randomWH.h.

4.2.2.11 long int y

The seed for the second generator

Definition at line 152 of file randomWH.h.

4.2.2.12 long int z

The seed for the third generator

Definition at line 153 of file randomWH.h.

The documentation for this struct was generated from the following file:

- randomWH.h

Chapter 5

File Documentation

5.1 randomWH.c File Reference

A pseudorandom number generator with good statistical properties.

`#include "randomWH.h"`

Functions

- void Congruence_init (Congruence *self, long int x, long int y, long int z, long int t)

  Initialise the generator with four values of type `long int`.
- void randomWH_error (const char *message)

  A very basic error handling function.
- long int * Congruence_read (Congruence *self)

  Read the current state of the pseudo-random number generator.
- double randomWH_null (Congruence *self)

  Generate an error message and triggers a program exit if the calling struct has not been initialised properly.
- double randomWH_32 (Congruence *self)

  Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 32-bit W-H algorithm .
- double randomWH_64 (Congruence *self)

  Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 64-bit W-H algorithm .
- long int randomWH_geometric (Congruence *self, double prob)

  Generate a geometrically distributed random variate.
- enum BOOLEAN randomWH_bernoulli (Congruence *self, double prob)

  Generate a Bernoulli random variate.
- long int randomWH_binomial (Congruence *self, long int trials, double prob)

  Generate a binomially distributed random variate.
- long int randomWH_fairdie (Congruence *self, long int lower_inclusive, long int upper_inclusive)

  Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

Variables

- Congruence congruent

5.1.1 Detailed Description

A pseudorandom number generator with good statistical properties.

Author

Martin Collier

Date

11 February 2014

See Also

http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y
http://www.viva64.com/en/a/0004/

Version

0.3 Alpha

Details

This implements the algorithm described by Wichmann and Hill in

- B.A. Wichmann and I.D. Hill, "Generating good pseudo-random numbers", *Computational Statistics & Data Analysis*, vol. 51, no. 3., pp. 1614-1622, Dec. 2006.

Two implementations are provided. One is for compilers that use 32 bits to store variables of the `long int` type. A faster algorithm is also provided for use with compilers that feature a 64-bit `long int` type. The algorithm to be used is determined automatically at run time.

Definition in file randomWH.c.

5.1.2 Function Documentation

5.1.2.1 void Congruence_init (Congruence * *self*, long int *x*, long int *y*, long int *z*, long int *t*)

Initialise the generator with four values of type `long int`.

We need to populate the `congruent` structure with seed values before calling `randomWH()`. This function isolates the initialisation from the implementation.

Usage

see Congruence.init()

```
#include "randomWH.h"
.
.
.
    Congruence_init(&congruent,1L,2L,3L,4L);
```

Parameters

x	The seed for the first generator

y	The seed for the second generator
z	The seed for the third generator
t	The seed for the fourth generator
self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Note

Initialising the generator with the same seeds from program run to program run will produce *exactly the same sequence* . Usually in simulation programs (particularly when testing and debugging them), this is **exactly what you want**.

Warning

Private function. Do not call it directly.

Definition at line 52 of file randomWH.c.

5.1.2.2 long int* Congruence_read (Congruence * *self*)

Read the current state of the pseudo-random number generator.

Usage

see Congruence.read()

```
#include <stdio.h>
#include "randomWH.h"
.
.
.
        long int* read_con;
        int count;
        read_con=Congruence_read(&congruent);
        for(count=0;count<4;count++)
            printf("%ld\n", read_con[count]);
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

A pointer to an array of **four** `long int` values .

See Also

Congruence_init()

Note

This function can be used immediately prior to program shutdown to read the current seed values prior to saving them in persistent storage. Loading these values from persistent storage during program initialisation and using them as the arguments in a call to Congruence_init() will allow execution to resume exactly as if the previous program run had continued.

Warning

Only **four** values are returned. Do **not** use an index greater than 3 to access the array.
Private function. Do not call it directly.

Definition at line 125 of file randomWH.c.

5.1.2.3 double randomWH_32 (Congruence * *self*)

Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 32-bit W-H algorithm .

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

a `double` value drawn (to a good approximation) from a uniform distribution in the range [0,1]

See Also

Congruence_init()
Congruence_read()
http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y

Warning

Never call this function directly as it requires struct initialisation before use. No test is made for correct initialisation here in the interest of performance. Bad things will happen if you bypass initialisation.

Definition at line 161 of file randomWH.c.

5.1.2.4 double randomWH_64 (Congruence * *self*)

Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 64-bit W-H algorithm .

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

a `double` value drawn (to a good approximation) from a uniform distribution in the range [0,1]

See Also

Congruence_init()
Congruence_read()
http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y

Warning

Never call this function directly as it requires struct initialisation before use. No test is made for correct initialisation here in the interest of performance. Bad things will happen if you bypass initialisation.

Definition at line 198 of file randomWH.c.

5.1.2.5 enum BOOLEAN randomWH_bernoulli (Congruence * *self*, double *prob*)

Generate a Bernoulli random variate.

Usage

see Congruence.bernoulli()

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
  .x=0,
  .y=0,
```

```
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
    if (generatorX.bernoulli(&generatorX,0.5)
        printf("Heads!\n");
     else
        printf("Tails!\n");
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success

Returns

a BOOLEAN variable indicating an outcome of success or failure

Definition at line 268 of file randomWH.c.

5.1.2.6 long int randomWH_binomial (Congruence * *self*, long int *trials*, double *prob*)

Generate a binomially distributed random variate.

Usage

see Congruence.binomial()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success in each trial
trials	The number of trials

Definition at line 288 of file randomWH.c.

5.1.2.7 void randomWH_error (const char * *message*)

A very basic error handling function.

This just prints an error message to `stderr` and then exits.

Usage

```
 #include "randomWH.h"
.
.
.
        randomWH_error("A pertinent error message");
```

Parameters

message	A byte string containing a message about the error.

Returns

The function never returns. The program is exited with status EXIT_FAILURE.

Definition at line 90 of file randomWH.c.

5.1.2.8 long int randomWH_fairdie (Congruence * *self*, long int *lower_inclusive*, long int *upper_inclusive*)

Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

Usage

see Congruence.fairdie()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
lower_inclusive	the lowest value of variate that can be returned
upper_inclusive	the highest value of variate that can be returned

Definition at line 307 of file randomWH.c.

5.1.2.9 long int randomWH_geometric (Congruence * *self*, double *prob*)

Generate a geometrically distributed random variate.

Usage

see Congruence.geometric()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success in each trial

Definition at line 223 of file randomWH.c.

5.1.2.10 double randomWH_null (Congruence * *self*)

Generate an error message and triggers a program exit if the calling struct has not been initialised properly.

This simulates exception handling for the exception "you forgot to read the documentation".

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

nothing - the program will exit, returning EXIT_FAILURE.

See Also

Congruence_init()
Congruence_read()

Definition at line 141 of file randomWH.c.

5.1.3 Variable Documentation

5.1.3.1 Congruence congruent

Initial value:

```
={
    .x=0,
    .y=0,
    .z=0,
```

```
    .t=0,
    .initialised=isfalse,
    .init=Congruence_init,
    .randomWH=randomWH_null,
    .read=Congruence_read,
    .bernoulli=randomWH_bernoulli,
    .geometric=randomWH_geometric,
    .binomial=randomWH_binomial,
    .fairdie=randomWH_fairdie,
}
```

The initialised congruent struct -

See Also

congruent

Definition at line 12 of file randomWH.c.

5.2 randomWH.h File Reference

A pseudorandom number generator with good statistical properties.

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

Data Structures

- struct generator

  Typedef of struct to store and process the four generator seeds.

Macros

- #define _RANDOMWH_H

Typedefs

- typedef struct generator Congruence

  Typedef of struct to store and process the four generator seeds.

Enumerations

- enum BOOLEAN { isfalse, istrue }

Functions

- void Congruence_init (Congruence *self, long int x, long int y, long int z, long int t)

  Initialise the generator with four values of type long int.
- long int * Congruence_read (Congruence *self)

  Read the current state of the pseudo-random number generator.
- double randomWH_32 (Congruence *self)

  Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 32-bit W-H algorithm .
- double randomWH_64 (Congruence *self)

Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 64-bit W-H algorithm .

- double randomWH_null (Congruence *self)

  Generate an error message and triggers a program exit if the calling struct has not been initialised properly.

- void randomWH_error (const char *message)

  A very basic error handling function.

- long int randomWH_geometric (Congruence *self, double prob)

  Generate a geometrically distributed random variate.

- enum BOOLEAN randomWH_bernoulli (Congruence *self, double prob)

  Generate a Bernoulli random variate.

- long int randomWH_binomial (Congruence *self, long int trials, double prob)

  Generate a binomially distributed random variate.

- long int randomWH_fairdie (Congruence *self, long int lower_inclusive, long int upper_inclusive)

  Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

Variables

- Congruence congruent

5.2.1 Detailed Description

A pseudorandom number generator with good statistical properties.

Author

Martin Collier

Date

11 February 2014

See Also

http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y
http://www.viva64.com/en/a/0004/

Version

0.3 Alpha

Details

This implements the algorithm described by Wichmann and Hill in

- B.A. Wichmann and I.D. Hill, "Generating good pseudo-random numbers", *Computational Statistics & Data Analysis*, vol. 51, no. 3., pp. 1614-1622, Dec. 2006.

Two implementations are provided. One is for compilers that use 32 bits to store variables of the `long int` type. A faster algorithm is also provided for use with compilers that feature a 64-bit `long int` type. The algorithm to be used is determined automatically at run time.

Definition in file randomWH.h.

5.2.2 Macro Definition Documentation

5.2.2.1 #define _RANDOMWH_H

The usual do-not-load-me-twice macro

Definition at line 96 of file randomWH.h.

5.2.3 Typedef Documentation

5.2.3.1 typedef struct generator Congruence

`Typedef` of `struct` to store and process the four generator seeds.

In ordinary usage, you will not use this typedef directly. Instead, you will use the instance of it called `Congruence` which is pre-initialised in randomWH.c. An exception would be if your code requires multiple instances of the random number generator. A second instance can be generated using the following code (failure to initialise this correctly will almost always cause fatal errors at run time).

```
#include "randomWH.h"

Congruence generator2 = {
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
```

This generator can be accessed using `generator2.init()`, `generator2.read()`, etc.

If multiple generators are instantiated, the number sequences generated may overlap; this is *certain* to occur if two or more are seeded with the same values.

See Also

congruent

5.2.4 Enumeration Type Documentation

5.2.4.1 enum BOOLEAN

A boolean type

< Declares an enumeration data type called BOOLEAN

Enumerator

isfalse false = 0, true = 1

istrue

Definition at line 107 of file randomWH.h.

5.2.5 Function Documentation

5.2.5.1 void Congruence_init (Congruence * *self*, long int *x*, long int *y*, long int *z*, long int *t*)

Initialise the generator with four values of type `long int`.

We need to populate the `congruent` structure with seed values before calling `randomWH()`. This function isolates the initialisation from the implementation.

Usage

see Congruence.init()

```
#include "randomWH.h"
.
.
.
    Congruence_init(&congruent,1L,2L,3L,4L);
```

Parameters

x	The seed for the first generator
y	The seed for the second generator
z	The seed for the third generator
t	The seed for the fourth generator
self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Note

Initialising the generator with the same seeds from program run to program run will produce *exactly the same sequence* . Usually in simulation programs (particularly when testing and debugging them), this is **exactly what you want**.

Warning

Private function. Do not call it directly.

Definition at line 52 of file randomWH.c.

5.2.5.2 long int* Congruence_read (Congruence * *self*)

Read the current state of the pseudo-random number generator.

Usage

see Congruence.read()

```
#include <stdio.h>
#include "randomWH.h"
.
.
.
        long int* read_con;
        int count;
        read_con=Congruence_read(&congruent);
        for(count=0;count<4;count++)
            printf("%ld\n", read_con[count]);
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

A pointer to an array of **four** `long int` values .

See Also

Congruence_init()

Note

This function can be used immediately prior to program shutdown to read the current seed values prior to saving them in persistent storage. Loading these values from persistent storage during program initialisation and using them as the arguments in a call to Congruence_init() will allow execution to resume exactly as if the previous program run had continued.

Warning

Only **four** values are returned. Do **not** use an index greater than 3 to access the array.
Private function. Do not call it directly.

Definition at line 125 of file randomWH.c.

5.2.5.3 double randomWH_32 (Congruence * *self*)

Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 32-bit W-H algorithm .

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

a `double` value drawn (to a good approximation) from a uniform distribution in the range [0,1]

See Also

Congruence_init()
Congruence_read()
http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y

Warning

Never call this function directly as it requires struct initialisation before use. No test is made for correct initialisation here in the interest of performance. Bad things will happen if you bypass initialisation.

Definition at line 161 of file randomWH.c.

5.2.5.4 double randomWH_64 (Congruence * *self*)

Generate a (pseudo-)random number drawn from the U(0,1) distribution using the 64-bit W-H algorithm .

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

a `double` value drawn (to a good approximation) from a uniform distribution in the range [0,1]

See Also

Congruence_init()
Congruence_read()
http://www.sciencedirect.com/science/article/pii/S0167947306001836?np=y

Warning

Never call this function directly as it requires struct initialisation before use. No test is made for correct initialisation here in the interest of performance. Bad things will happen if you bypass initialisation.

Definition at line 198 of file randomWH.c.

5.2.5.5 enum BOOLEAN randomWH_bernoulli (Congruence * *self*, double *prob*)

Generate a Bernoulli random variate.

Usage

see Congruence.bernoulli()

```
#include <stdio.h>
#include "randomWH.h"
Congruence generatorX={
   .x=0,
   .y=0,
   .z=0,
   .t=0,
   .initialised=isfalse,
   .init=Congruence_init,
   .randomWH=randomWH_null,
   .read=Congruence_read,
   .bernoulli=randomWH_bernoulli,
   .geometric=randomWH_geometric,
   .binomial=randomWH_binomial,
   .fairdie=randomWH_fairdie,
};
.
.
.
    if (generatorX.bernoulli(&generatorX,0.5)
        printf("Heads!\n");
     else
        printf("Tails!\n");
```

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success

Returns

a BOOLEAN variable indicating an outcome of success or failure

Definition at line 268 of file randomWH.c.

5.2.5.6 long int randomWH_binomial (Congruence * *self*, long int *trials*, double *prob*)

Generate a binomially distributed random variate.

Usage

see Congruence.binomial()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
prob	The probability of success in each trial
trials	The number of trials

Definition at line 288 of file randomWH.c.

5.2.5.7 void randomWH_error (const char * *message*)

A very basic error handling function.

This just prints an error message to `stderr` and then exits.

Usage

```
#include "randomWH.h"
.
.
.
        randomWH_error("A pertinent error message");
```

Parameters

message	A byte string containing a message about the error.

Returns

The function never returns. The program is exited with status EXIT_FAILURE.

Definition at line 90 of file randomWH.c.

5.2.5.8 long int randomWH_fairdie (Congruence * *self*, long int *lower_inclusive*, long int *upper_inclusive*)

Generate a discrete-valued uniformly distributed random variate drawn from a specified range.

Usage

see Congruence.fairdie()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).
lower_inclusive	the lowest value of variate that can be returned
upper_inclusive	the highest value of variate that can be returned

Definition at line 307 of file randomWH.c.

5.2.5.9 long int randomWH_geometric (Congruence * *self*, double *prob*)

Generate a geometrically distributed random variate.

Usage

see Congruence.geometric()

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

prob	The probability of success in each trial

Definition at line 223 of file randomWH.c.

5.2.5.10 double randomWH_null (Congruence * *self*)

Generate an error message and triggers a program exit if the calling struct has not been initialised properly.

This simulates exception handling for the exception "you forgot to read the documentation".

Parameters

self	A pointer to the calling `struct` (required since C does not support the **this** keyword).

Returns

nothing - the program will exit, returning EXIT_FAILURE.

See Also

Congruence_init()
Congruence_read()

Definition at line 141 of file randomWH.c.

5.2.6 Variable Documentation

5.2.6.1 Congruence congruent

The initialised congruent struct -

See Also

congruent

Definition at line 12 of file randomWH.c.

5.3 randomWH_usage.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "randomWH.h"
```

Macros

- #define SIM_DURATION 40000000L

Functions

- void Congruence_dump (Congruence *generatorZ)
- int main ()

Variables

- Congruence congruent2

5.3.1 Macro Definition Documentation

5.3.1.1 #define SIM_DURATION 40000000L

Definition at line 39 of file randomWH_usage.c.

5.3.2 Function Documentation

5.3.2.1 void Congruence_dump (Congruence * *generatorZ*)

Definition at line 62 of file randomWH_usage.c.

5.3.2.2 int main ( )

Definition at line 72 of file randomWH_usage.c.

5.3.3 Variable Documentation

5.3.3.1 Congruence congruent2

Initial value:

```
={
    .x=0,
    .y=0,
    .z=0,
    .t=0,
    .initialised=isfalse,
    .init=Congruence_init,
    .randomWH=randomWH_null,
    .read=Congruence_read,
    .bernoulli=randomWH_bernoulli,
    .geometric=randomWH_geometric,
    .binomial=randomWH_binomial,
    .fairdie=randomWH_fairdie,
}
```

Definition at line 45 of file randomWH_usage.c.

Index