

cRabbitPopulationSimulation

Généré par Doxygen 1.8.16

1 cRabbitPopulation	1
2 Index des structures de données	3
2.1 Structures de données	3
3 Index des fichiers	5
3.1 Liste des fichiers	5
4 Documentation des structures de données	7
4.1 Référence de la structure configSimu_t	7
4.1.1 Description détaillée	7
4.1.2 Documentation des champs	7
4.1.2.1 ageDisminish	7
4.1.2.2 initPredator	8
4.1.2.3 littleRabbitSurvRate	8
4.1.2.4 maxBabyPerLitter	8
4.1.2.5 maxLitterPerYear	8
4.1.2.6 maxRabbitYear	8
4.1.2.7 maxYear	8
4.1.2.8 minBabyPerLitter	8
4.1.2.9 minLitterPerYear	9
4.1.2.10 probaLittersBabys	9
4.1.2.11 probaLittersYear	9
4.1.2.12 rabbitSurvRate	9
4.1.2.13 rateDisminish	9
4.2 Référence de la structure link	9
4.2.1 Documentation des champs	10
4.2.1.1 next	10
4.2.1.2 rabbit	10
4.3 Référence de la structure link_t	10
4.3.1 Description détaillée	10
4.4 Référence de la structure list_rabbit_t	10
4.4.1 Description détaillée	10
4.5 Référence de la structure rabbit_t	11
4.5.1 Description détaillée	11
4.5.2 Documentation des champs	11
4.5.2.1 alive	11
4.5.2.2 nb_years	11
4.5.2.3 sexe	11
5 Documentation des fichiers	13
5.1 Référence du fichier fibo.c	13
5.1.1 Description détaillée	13
5.1.2 Documentation des fonctions	13

5.1.2.1 fibo_explicite()	13
5.1.2.2 fibo_recurrence()	14
5.1.2.3 fibo_recuratif()	14
5.2 Référence du fichier fibo.h	14
5.2.1 Description détaillée	15
5.2.2 Documentation des fonctions	15
5.2.2.1 fibo_explicite()	15
5.2.2.2 fibo_recurrence()	15
5.2.2.3 fibo_recuratif()	16
5.3 Référence du fichier main.c	16
5.3.1 Description détaillée	17
5.3.2 Documentation des fonctions	17
5.3.2.1 calcul_moyenne_long()	17
5.3.2.2 compute_S()	17
5.3.2.3 intervalle_de_confiance()	18
5.3.2.4 main()	18
5.3.2.5 question1()	18
5.3.2.6 question2_list()	19
5.3.2.7 question2_TS()	19
5.4 Référence du fichier rabbit.c	19
5.4.1 Description détaillée	20
5.4.2 Documentation des fonctions	20
5.4.2.1 give_birth()	20
5.4.2.2 give_birth_all()	20
5.4.2.3 is_alive()	21
5.4.2.4 is_female()	21
5.4.2.5 is_male()	21
5.4.2.6 new_rabbit()	22
5.4.2.7 new_rabbit2()	22
5.4.2.8 realistic_simulation()	22
5.4.2.9 survive()	23
5.4.2.10 survive_all()	23
5.5 Référence du fichier rabbit.h	23
5.5.1 Description détaillée	24
5.5.2 Documentation des fonctions	24
5.5.2.1 give_birth()	24
5.5.2.2 is_alive()	24
5.5.2.3 is_female()	25
5.5.2.4 is_male()	25
5.5.2.5 new_rabbit()	25
5.5.2.6 new_rabbit2()	26
5.5.2.7 realistic_simulation()	26

5.5.2.8 survive()	26
5.5.2.9 survive_all()	27
5.6 Référence du fichier rabbit_list.c	27
5.6.1 Description détaillée	27
5.6.2 Documentation des fonctions	28
5.6.2.1 add_head()	28
5.6.2.2 delete_dead()	28
5.6.2.3 delete_head()	28
5.6.2.4 display_death()	29
5.6.2.5 display_rabbit()	29
5.6.2.6 display_rabbit_all()	29
5.6.2.7 empty()	30
5.6.2.8 head()	30
5.6.2.9 new_list()	30
5.6.2.10 size_list()	31
5.7 Référence du fichier rabbit_list.h	32
5.7.1 Description détaillée	33
5.7.2 Documentation des fonctions	33
5.7.2.1 add_head()	33
5.7.2.2 delete_dead()	33
5.7.2.3 delete_head()	34
5.7.2.4 display_death()	34
5.7.2.5 display_rabbit()	34
5.7.2.6 display_rabbit_all()	35
5.7.2.7 empty()	35
5.7.2.8 head()	35
5.7.2.9 new_list()	36
5.7.2.10 size_list()	36
5.8 Référence du fichier rabbitTreeStatus.c	36
5.8.1 Description détaillée	37
5.8.2 Documentation des fonctions	37
5.8.2.1 addBabys()	37
5.8.2.2 CalcSumYears()	37
5.8.2.3 checkMallocNull()	38
5.8.2.4 copyRabbitTab()	38
5.8.2.5 createRabbitTabZero()	39
5.8.2.6 free_double_tab()	39
5.8.2.7 makeBabys()	39
5.8.2.8 rabbit_simu_iteration()	40
5.8.2.9 realistic_simulation_TS()	40
5.8.2.10 sum_array()	41
5.8.2.11 surviveRabbitYear()	41

5.9 Référence du fichier rabbitTreeStatus.h	42
5.9.1 Description détaillée	42
5.9.2 Documentation des fonctions	42
5.9.2.1 realistic_simulation_TS()	42
5.10 Référence du fichier reverseDistLow.c	43
5.10.1 Description détaillée	43
5.10.2 Documentation des fonctions	43
5.10.2.1 calcCumulProba()	43
5.10.2.2 discDispLoop()	44
5.10.2.3 genRand_discDist()	44
5.10.2.4 probaCalc()	45
5.11 Référence du fichier reverseDistLow.h	45
5.11.1 Description détaillée	45
5.11.2 Documentation des fonctions	46
5.11.2.1 calcCumulProba()	46
5.11.2.2 discDispLoop()	46
5.11.2.3 genRand_discDist()	46
5.11.2.4 probaCalc()	48
Index	49

Chapitre 1

cRabbitPopulation

Simule une population de lapins en prenant en compte leur probabilité de survivre ainsi que leur probabilité de donner naissance à l'aide de nombres pseudo-aléatoires.

Projet réalisé dans le cadre de mes études en L2 informatique

Générateur de nombres pseudo-aléatoires utilisé : Mersenne Twister de 2002 par Makoto Matsumoto link ↔
: <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html>

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

configSimu_t	Configuration d'une simulation	7
link	9
link_t	Structure qui définit une cellule de liste chaînée de lapins	10
list_rabbit_t	Structure qui définit une liste chaînée de lapins	10
rabbit_t	Structure qui définit un lapin	11

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

fibonacci.c	Programmes pour calculer les termes de la suite de Fibonacci	13
fibonacci.h	Programmes pour calculer les termes de la suite de Fibonacci	14
main.c	Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin .	16
mt19937ar.h		??
rabbit.c	Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps	19
rabbit.h	Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps	23
rabbit_list.c	Programmes pour stocker des lapins dans une liste	27
rabbit_list.h	Programmes pour stocker des lapins dans une liste	32
rabbitTreeStatus.c	Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états	36
rabbitTreeStatus.h	Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états	42
reverseDistLow.c	Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné . .	43
reverseDistLow.h	Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné . .	45

Chapitre 4

Documentation des structures de données

4.1 Référence de la structure configSimu_t

configuration d'une simulation

```
#include <rabbitTreeStatus.h>
```

Champs de données

- int `maxBabyPerLitter`
- int `minBabyPerLitter`
- double `littleRabbitSurvRate`
- double `rabbitSurvRate`
- double `rateDisminish`
- int `ageDisminish`
- int `minLitterPerYear`
- int `maxLitterPerYear`
- double * `probaLittersYear`
- double * `probaLittersBabys`
- int `maxRabbitYear`
- int `maxYear`
- int `initPredator`

4.1.1 Description détaillée

configuration d'une simulation

4.1.2 Documentation des champs

4.1.2.1 `ageDisminish`

```
int ageDisminish
```

age où les lapin perde du taux de survie

4.1.2.2 initPredator

```
int initPredator
```

mois a partir duquel les bébé peuvent mourir

4.1.2.3 littleRabbitSurvRate

```
double littleRabbitSurvRate
```

taux de survie d'un bébé lapin

4.1.2.4 maxBabyPerLitter

```
int maxBabyPerLitter
```

nombre Max de bébé par portées

4.1.2.5 maxLitterPerYear

```
int maxLitterPerYear
```

nombre max de portées par ans

4.1.2.6 maxRabbitYear

```
int maxRabbitYear
```

age mas des lapins

4.1.2.7 maxYear

```
int maxYear
```

nombre d'années de simulation

4.1.2.8 minBabyPerLitter

```
int minBabyPerLitter
```

nombre min de bébé par portées

4.1.2.9 minLitterPerYear

```
int minLitterPerYear
```

nombre min de portées par ans

4.1.2.10 probaLittersBabys

```
double* probaLittersBabys
```

probabilités de bébé

4.1.2.11 probaLittersYear

```
double* probaLittersYear
```

probabilités de portées

4.1.2.12 rabbitSurvRate

```
double rabbitSurvRate
```

taux de survie d'un lapin adulte

4.1.2.13 rateDisminish

```
double rateDisminish
```

taux de survie perdu après ageDisminish

La documentation de cette structure a été générée à partir du fichier suivant :

— [rabbitTreeStatus.h](#)

4.2 Référence de la structure link

Champs de données

— [rabbit_t](#) rabbit
— struct [link](#) * next

4.2.1 Documentation des champs

4.2.1.1 next

```
struct link* next
```

pointeur sur la cellule suivante

4.2.1.2 rabbit

```
rabbit_t rabbit
```

valeur de la cellule, ici un lapin

La documentation de cette structure a été générée à partir du fichier suivant :

— [rabbit_list.h](#)

4.3 Référence de la structure link_t

structure qui définit une cellule de liste chaînée de lapins

```
#include <rabbit_list.h>
```

4.3.1 Description détaillée

structure qui définit une cellule de liste chaînée de lapins

La documentation de cette structure a été générée à partir du fichier suivant :

— [rabbit_list.h](#)

4.4 Référence de la structure list_rabbit_t

structure qui définit une liste chaînée de lapins

```
#include <rabbit_list.h>
```

4.4.1 Description détaillée

structure qui définit une liste chaînée de lapins

La documentation de cette structure a été générée à partir du fichier suivant :

— [rabbit_list.h](#)

4.5 Référence de la structure rabbit_t

structure qui définit un lapin

```
#include <rabbit_list.h>
```

Champs de données

- `sexe_t` `sexe`
- `int` `nb_years`
- `Boolean` `alive`

4.5.1 Description détaillée

structure qui définit un lapin

4.5.2 Documentation des champs

4.5.2.1 alive

`Boolean` `alive`

variable qui définit si le lapin est vivant

4.5.2.2 nb_years

`int` `nb_years`

age du lapin

4.5.2.3 sexe

`sexe_t` `sexe`

sexe du lapin

La documentation de cette structure a été générée à partir du fichier suivant :

- `rabbit_list.h`

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier fibo.c

Programmes pour calculer les termes de la suite de Fibonacci.

```
#include "fibo.h"
```

Fonctions

- unsigned long [fibo_recuratif](#) (int n)
- unsigned long [fibo_recurrence](#) (int n)
- unsigned long [fibo_explicite](#) (int n)

5.1.1 Description détaillée

Programmes pour calculer les termes de la suite de Fibonacci.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.1.2 Documentation des fonctions

5.1.2.1 fibo_explicite()

```
unsigned long fibo_explicite (  
    int n )
```

fonction pour obtenir le n-ième terme de la suite de Fibonacci implémentée à l'aide de la formule explicite

Paramètres

n	indice du terme voulu
-----	-----------------------

Renvoie

le terme à la n -ième place dans la suite

5.1.2.2 fibo_recurrence()

```
unsigned long fibo_recurrence (  
    int  $n$  )
```

fonction pour obtenir le n -ième terme de la suite de Fibonacci implémentée à l'aide de la formule de récurrence

Paramètres

n	indice du terme voulu
-----	-----------------------

Renvoie

le terme à la n -ième place dans la suite

5.1.2.3 fibo_recuratif()

```
unsigned long fibo_recuratif (  
    int  $n$  )
```

fonction pour obtenir le n -ième terme de la suite de Fibonacci implémentée récursivement

Paramètres

n	indice du terme voulu
-----	-----------------------

Renvoie

le terme à la n -ième place dans la suite

5.2 Référence du fichier fibo.h

Programmes pour calculer les termes de la suite de Fibonacci.

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

Fonctions

- unsigned long [fibo_recuratif](#) (int n)
- unsigned long [fibo_recurrence](#) (int n)
- unsigned long [fibo_explicite](#) (int n)

5.2.1 Description détaillée

Programmes pour calculer les termes de la suite de Fibonacci.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.2.2 Documentation des fonctions

5.2.2.1 [fibo_explicite\(\)](#)

```
unsigned long fibo_explicite (  
    int n )
```

fonction pour obtenir le n-ième terme de la suite de Fibonacci implémentée à l'aide de la formule explicite

Paramètres

<i>n</i>	indice du terme voulu
----------	-----------------------

Renvoie

le terme à la n-ième place dans la suite

5.2.2.2 [fibo_recurrence\(\)](#)

```
unsigned long fibo_recurrence (  
    int n )
```

fonction pour obtenir le n-ième terme de la suite de Fibonacci implémentée à l'aide de la formule de récurrence

Paramètres

n	indice du terme voulu
-----	-----------------------

Renvoie

le terme à la n -ième place dans la suite

5.2.2.3 fibo_recuratif()

```
unsigned long fibo_recuratif (  
    int  $n$  )
```

fonction pour obtenir le n -ième terme de la suite de Fibonacci implémentée récursivement

Paramètres

n	indice du terme voulu
-----	-----------------------

Renvoie

le terme à la n -ième place dans la suite

5.3 Référence du fichier main.c

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin.

```
#include "fibo.h"  
#include "rabbit.h"  
#include "rabbitTreeStatus.h"  
#include <stdio.h>
```

Fonctions

- void [question1](#) ()
- double [compute_S](#) (unsigned long tab[], int n , double moyenne)
- double [calcul_moyenne_long](#) (unsigned long tab[], int n)
- void [intervalle_de_confiance](#) (unsigned long tab[], int nbExp)
- void [question2_list](#) ()
- void [question2_TS](#) ()
- int [main](#) ()

5.3.1 Description détaillée

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.3.2 Documentation des fonctions

5.3.2.1 calcul_moyenne_long()

```
double calcul_moyenne_long (
    unsigned long tab[],
    int n )
```

fonction pour calculer la moyenne d'un tableau de unsigned long

Paramètres

<i>tab</i>	tableau
<i>n</i>	taille du tableau

Renvoie

moyenne du tableau

5.3.2.2 compute_S()

```
double compute_S (
    unsigned long tab[],
    int n,
    double moyenne )
```

calcul l'estimateur sans biais de la variance

Paramètres

<i>tab</i>	tableau
<i>n</i>	taille du tableau
<i>moyenne</i>	moyenne du tableau

Renvoie

S

5.3.2.3 intervalle_de_confiance()

```
void intervalle_de_confiance (
    unsigned long tab[],
    int nbExp )
```

fonction pour calculer et afficher l'intervalle de confiance d'un tableau de unsigned long

Paramètres

<i>tab</i>	tableau
<i>nbExp</i>	taille du tableau / nombre d'espériences

Renvoie

void

5.3.2.4 main()

```
int main ( )
```

point d'entré du programme, ici elle appelle question2_TS et initialise le générateur de nombre pseudo-aléatoire

Renvoie

void

5.3.2.5 question1()

```
void question1 ( )
```

calcul le 11 eme élément de la suite de fibonacci avec trois méthodes différentes

Renvoie

void

5.3.2.6 question2_list()

```
void question2_list ( )
```

fonction pour réaliser une simulation d'évolution de population de lapin avec une structure list

Renvoie

void

5.3.2.7 question2_TS()

```
void question2_TS ( )
```

fonction pour réaliser une simulation d'évolution de population de lapin avec une structure d'arbre d'états

Renvoie

void

5.4 Référence du fichier rabbit.c

Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps.

```
#include "rabbit.h"
```

Fonctions

- `rabbit_t new_rabbit` (int years)
- `rabbit_t new_rabbit2` (int years, `sexe_t` sexe)
- `sexe_t is_male` (`rabbit_t` rabbit)
- `sexe_t is_female` (`rabbit_t` rabbit)
- `Boolean give_birth` (`rabbit_t` rabbit)
- `Boolean survive` (`rabbit_t` rabbit)
- `Boolean is_alive` (`rabbit_t` rabbit)
- `list_rabbit_t give_birth_all` (`list_rabbit_t` list_rabbit)
- `list_rabbit_t survive_all` (`list_rabbit_t` list_rabbit)
- `void realistic_simulation` ()

Variables

- `int nb_loop` = 1

5.4.1 Description détaillée

Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.4.2 Documentation des fonctions

5.4.2.1 give_birth()

```
Boolean give_birth (
    rabbit_t rabbit )
```

permet de savoir si le lapin donne naissance à une portée

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin donne naissance

5.4.2.2 give_birth_all()

```
list_rabbit_t give_birth_all (
    list_rabbit_t list_rabbit )
```

ajoute tout les nouveaux nés dans la liste de lapins

Paramètres

<i>list_rabbit</i>	une liste de lapin
--------------------	--------------------

Renvoie

la nouvelle liste de lapins après insertion des nouveaux nés

5.4.2.3 is_alive()

```
Boolean is_alive (  
    rabbit_t rabbit )
```

retourne la variable booléenne alive du lapin

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est en vie

5.4.2.4 is_female()

```
sexe_t is_female (  
    rabbit_t rabbit )
```

permet de savoir si le lapin est une femelle

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est une femelle

5.4.2.5 is_male()

```
sexe_t is_male (  
    rabbit_t rabbit )
```

permet de savoir si le lapin est un mâle

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est un mâle

5.4.2.6 new_rabbit()

```
rabbit_t new_rabbit (
    int years )
```

créer un lapin de sexe aléatoire

Paramètres

<i>years</i>	age du lapin
--------------	--------------

Renvoie

un lapin

5.4.2.7 new_rabbit2()

```
rabbit_t new_rabbit2 (
    int years,
    sexe_t sexe )
```

créer un lapin de sexe défini

Paramètres

<i>years</i>	age du lapin
<i>sexe</i>	sexe du lapin

Renvoie

un lapin

5.4.2.8 realistic_simulation()

```
void realistic_simulation ( )
```

fonction pour réaliser une simulation d'évolution de population de lapin sur plusieurs années

5.4.2.9 survive()

```
Boolean survive (
    rabbit_t rabbit )
```

permet de savoir si le lapin survit jusqu'à l'année suivante

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin survit

5.4.2.10 survive_all()

```
list_rabbit_t survive_all (
    list_rabbit_t list_rabbit )
```

change la valeur du booléen alive pour tous les lapins de la liste

Paramètres

<i>list_rabbit</i>	une liste de lapin
--------------------	--------------------

Renvoie

la nouvelle liste de lapins après modification du booléen

5.5 Référence du fichier rabbit.h

Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps.

```
#include "fibonacci.h"
#include "mt19937ar.h"
#include "rabbit_list.h"
#include <stdio.h>
#include <stdlib.h>
```

Fonctions

- `rabbit_t new_rabbit` (int years)
- `rabbit_t new_rabbit2` (int years, `sexe_t` sexe)
- `sexe_t is_male` (rabbit_t rabbit)
- `sexe_t is_female` (rabbit_t rabbit)
- `Boolean give_birth` (rabbit_t rabbit)
- `Boolean survive` (rabbit_t rabbit)
- `Boolean is_alive` (rabbit_t rabbit)
- `list_rabbit_t survive_all` (list_rabbit_t list_rabbit)
- `void realistic_simulation` ()

5.5.1 Description détaillée

Programmes pour gérer les lapins ainsi que la simulation de l'évolution de la population de lapin dans le temps.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.5.2 Documentation des fonctions

5.5.2.1 give_birth()

```
Boolean give_birth (
    rabbit_t rabbit )
```

permet de savoir si le lapin donne naissance à une portée

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin donne naissance

5.5.2.2 is_alive()

```
Boolean is_alive (
    rabbit_t rabbit )
```

retourne la variable booléenne alive du lapin

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est en vie

5.5.2.3 is_female()

```
sexe_t is_female (
    rabbit_t rabbit )
```

permet de savoir si le lapin est une femelle

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est une femelle

5.5.2.4 is_male()

```
sexe_t is_male (
    rabbit_t rabbit )
```

permet de savoir si le lapin est un mâle

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin est un mâle

5.5.2.5 new_rabbit()

```
rabbit_t new_rabbit (
    int years )
```

créer un lapin de sexe aléatoire

Paramètres

<i>years</i>	age du lapin
--------------	--------------

Renvoie

un lapin

5.5.2.6 new_rabbit2()

```
rabbit_t new_rabbit2 (
    int years,
    sexe_t sexe )
```

créer un lapin de sexe défini

Paramètres

<i>years</i>	age du lapin
<i>sexe</i>	sexe du lapin

Renvoie

un lapin

5.5.2.7 realistic_simulation()

```
void realistic_simulation ( )
```

fonction pour réaliser une simulation d'évolution de population de lapin sur plusieurs années

5.5.2.8 survive()

```
Boolean survive (
    rabbit_t rabbit )
```

permet de savoir si le lapin survit jusqu'à l'année suivante

Paramètres

<i>rabbit</i>	un lapin
---------------	----------

Renvoie

true si le lapin survit

5.5.2.9 survive_all()

```
list_rabbit_t survive_all (
    list_rabbit_t list_rabbit )
```

change la valeur du booléen alive pour tous les lapins de la liste

Paramètres

<i>list_rabbit</i>	une liste de lapin
--------------------	--------------------

Renvoie

la nouvelle liste de lapins après modification du booléen

5.6 Référence du fichier rabbit_list.c

Programmes pour stocker des lapins dans une liste.

```
#include "rabbit_list.h"
```

Fonctions

- `list_rabbit_t new_list` (void)
- `rabbit_t head` (list_rabbit_t list_rabbit)
- `Boolean empty` (list_rabbit_t list_rabbit)
- `list_rabbit_t add_head` (list_rabbit_t list_rabbit, rabbit_t rabbit)
- `list_rabbit_t delete_head` (list_rabbit_t list_rabbit)
- `list_rabbit_t delete_dead` (list_rabbit_t list_rabbit)
- `void display_rabbit` (rabbit_t rabbit)
- `void display_rabbit_all` (list_rabbit_t list_rabbit)
- `void display_death` (list_rabbit_t list_rabbit)
- `int size_list` (list_rabbit_t list_rabbit)

5.6.1 Description détaillée

Programmes pour stocker des lapins dans une liste.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.6.2 Documentation des fonctions

5.6.2.1 add_head()

```
list_rabbit_t add_head (
    list_rabbit_t list_rabbit,
    rabbit_t rabbit )
```

ajoute un lapin dans la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
<i>rabbit</i>	lapin à ajouter à la liste

Renvoie

la liste avec le lapin ajouté

5.6.2.2 delete_dead()

```
list_rabbit_t delete_dead (
    list_rabbit_t list_rabbit )
```

supprime tout les lapins morts de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

la liste avec les lapins morts supprimés

5.6.2.3 delete_head()

```
list_rabbit_t delete_head (
    list_rabbit_t list_rabbit )
```

supprime le premier lapin de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

la liste avec le lapin supprimé

5.6.2.4 display_death()

```
void display_death (
    list_rabbit_t list_rabbit )
```

affiche la valeur de la variable booléenne alive (pour debug)

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

5.6.2.5 display_rabbit()

```
void display_rabbit (
    rabbit_t rabbit )
```

affiche les informations d'un lapin

Paramètres

<i>rabbit</i>	un lapins
---------------	-----------

5.6.2.6 display_rabbit_all()

```
void display_rabbit_all (
    list_rabbit_t list_rabbit )
```

affiche les informations de tous les lapins de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

5.6.2.7 empty()

```
Boolean empty (
    list_rabbit_t list_rabbit )
```

permet de savoir si la liste est vide

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

true si la liste est vide, false sinon

5.6.2.8 head()

```
rabbit_t head (
    list_rabbit_t list_rabbit )
```

permet d'obtenir le premier élément de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

le premier lapin de la liste

5.6.2.9 new_list()

```
list_rabbit_t new_list (
    void )
```

créer une liste vide

Renvoie

NULL

5.6.2.10 size_list()

```
int size_list (
    list_rabbit_t list_rabbit )
```

compte le nombre d'éléments de la liste

Paramètres

<code>list_rabbit</code>	liste de lapins
--------------------------	-----------------

Renvoie

le nombre d'éléments de la liste

5.7 Référence du fichier `rabbit_list.h`

Programmes pour stocker des lapins dans une liste.

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

Structures de données

- struct `rabbit_t`
structure qui définit un lapin
- struct `link`

Définitions de type

- typedef struct `link` `link_t`
- typedef struct `link` * `list_rabbit_t`

Énumérations

- enum `sexe_t` { `male`, `female` }
structure pour définir le sexe d'un lapin
- enum `Boolean` { `false`, `true` }
structure pour définir un type boolean

Fonctions

- `list_rabbit_t` `new_list` (void)
- `rabbit_t` `head` (`list_rabbit_t` `list_rabbit`)
- `Boolean` `empty` (`list_rabbit_t` `list_rabbit`)
- `list_rabbit_t` `add_head` (`list_rabbit_t` `list_rabbit`, `rabbit_t` `rabbit`)
- `list_rabbit_t` `delete_head` (`list_rabbit_t` `list_rabbit`)
- `list_rabbit_t` `delete_dead` (`list_rabbit_t` `list_rabbit`)
- void `display_rabbit` (`rabbit_t` `rabbit`)
- void `display_rabbit_all` (`list_rabbit_t` `list_rabbit`)
- void `display_death` (`list_rabbit_t` `list_rabbit`)
- int `size_list` (`list_rabbit_t` `list_rabbit`)

5.7.1 Description détaillée

Programmes pour stocker des lapins dans une liste.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.7.2 Documentation des fonctions

5.7.2.1 add_head()

```
list_rabbit_t add_head (  
    list_rabbit_t list_rabbit,  
    rabbit_t rabbit )
```

ajoute un lapin dans la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
<i>rabbit</i>	lapin à ajouter à la liste

Renvoie

la liste avec le lapin ajouté

5.7.2.2 delete_dead()

```
list_rabbit_t delete_dead (  
    list_rabbit_t list_rabbit )
```

supprime tout les lapins morts de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

la liste avec les lapins morts supprimés

5.7.2.3 delete_head()

```
list_rabbit_t delete_head (
    list_rabbit_t list_rabbit )
```

supprime le premier lapin de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

la liste avec le lapin supprimé

5.7.2.4 display_death()

```
void display_death (
    list_rabbit_t list_rabbit )
```

affiche la valeur de la variable booléenne alive (pour debug)

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

5.7.2.5 display_rabbit()

```
void display_rabbit (
    rabbit_t rabbit )
```

affiche les informations d'un lapin

Paramètres

<i>rabbit</i>	un lapins
---------------	-----------

5.7.2.6 display_rabbit_all()

```
void display_rabbit_all (
    list_rabbit_t list_rabbit )
```

affiche les informations de tous les lapins de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

5.7.2.7 empty()

```
Boolean empty (
    list_rabbit_t list_rabbit )
```

permet de savoir si la liste est vide

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

true si la liste est vide, false sinon

5.7.2.8 head()

```
rabbit_t head (
    list_rabbit_t list_rabbit )
```

permet d'obtenir le premier élément de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

le premier lapin de la liste

5.7.2.9 new_list()

```
list_rabbit_t new_list (
    void )
```

créer une liste vide

Renvoie

NULL

5.7.2.10 size_list()

```
int size_list (
    list_rabbit_t list_rabbit )
```

compte le nombre d'éléments de la liste

Paramètres

<i>list_rabbit</i>	liste de lapins
--------------------	-----------------

Renvoie

le nombre d'éléments de la liste

5.8 Référence du fichier rabbitTreeStatus.c

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états.

```
#include "rabbitTreeStatus.h"
```

Fonctions

- void [checkMallocNull](#) (void *tab)
- unsigned long ** [createRabbitTabZero](#) (int maxRabbitYear)
- void [copyRabbitTab](#) (unsigned long **src, unsigned long **dest, int maxRabbitYear)
- int [surviveRabbitYear](#) (int old, [configSimu_t](#) config)
- unsigned long [makeBabys](#) (int old, double cumulProbaLitter[], double cumulProbaBabys[], int lenCumulProbaLitter, int lenCumulProbaBabys, int minLitter, int minBaby)
- unsigned long [CalcSumYears](#) (unsigned long **tab, int maxOld)
- void [addBabys](#) (unsigned long babys, unsigned long **rabbit)
- void [free_double_tab](#) (unsigned long **tab, int n)
- unsigned long [sum_array](#) (unsigned long array[], int n)
- unsigned long [rabbit_simu_iteration](#) ([configSimu_t](#) config, unsigned long **rabbits, unsigned long **rabbitsTmp, int year, double *cumulProbaLitter, double *cumulProbaBabys, int lenCumulProbaLitter, int lenCumulProbaBabys)
- unsigned long [realistic_simulation_TS](#) ([configSimu_t](#) config)

5.8.1 Description détaillée

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.8.2 Documentation des fonctions

5.8.2.1 addBabys()

```
void addBabys (
    unsigned long babys,
    unsigned long ** rabbit )
```

fonction pour ajouter les lapins a la liste des lapins

Paramètres

<i>babys</i>	nombre de bébés a ajouter
<i>rabbit</i>	tableaux d'états de lapins

Renvoie

void

5.8.2.2 CalcSumYears()

```
unsigned long CalcSumYears (
    unsigned long ** tab,
    int maxOld )
```

fonction pour calculer la somme des lapins

Paramètres

<i>tab</i>	tableau
<i>maxOld</i>	age max des lapins

Renvoie

la somme des lapins

5.8.2.3 checkMallocNull()

```
void checkMallocNull (
    void * tab )
```

vérifi si un malloc c'est bien passé

Paramètres

<i>tab</i>	tableau
------------	---------

Renvoie

void

5.8.2.4 copyRabitTab()

```
void copyRabitTab (
    unsigned long ** src,
    unsigned long ** dest,
    int maxRabbitYear )
```

fonction pour copier des tableaux d'états de lapin

Paramètres

<i>src</i>	tableau source
<i>dest</i>	tableau de destination
<i>maxRabbitYear</i>	l'age maximum d'un lapin

Renvoie

void

5.8.2.5 createRabbitTabZero()

```
unsigned long** createRabbitTabZero (
    int maxRabbitYear )
```

crée un tableau pour stocker des états de lapin, initialisé avec des zero

Paramètres

<i>maxRabbitYear</i>	l'age maximum d'un lapin
----------------------	--------------------------

Renvoie

un tableau de tableaux de unsigned long

5.8.2.6 free_double_tab()

```
void free_double_tab (
    unsigned long ** tab,
    int n )
```

fonction pour free un tableau de tableaux de unsigned long

Paramètres

<i>tab</i>	tableau
<i>n</i>	nombre d'éléments dans le tableau

Renvoie

void

5.8.2.7 makeBabys()

```
unsigned long makeBabys (
    int old,
    double cumulProbaLitter[],
    double cumulProbaBabys[],
    int lenCumulProbaLitter,
    int lenCumulProbaBabys,
    int minLitter,
    int minBaby )
```

fonction pour savoir combien de lapins va faire un lapin donné pendant une année

Paramètres

<i>old</i>	age du lapin
<i>cumulProbaLitter</i>	tableau de probabilité cumulé pour le nombre de portée
<i>cumulProbaBabys</i>	tableau de probabilité cumulé pour le nombre de bébés par portée
<i>lenCumulProbaLitter</i>	longueur du tableau de proba de portée
<i>lenCumulProbaBabys</i>	longueur du tableau de proba de bébés
<i>minLitter</i>	nombre minimum de portées
<i>minBaby</i>	nombre minimum de bébés

Renvoie

1 ou 0

5.8.2.8 rabbit_simu_iteration()

```
unsigned long rabbit_simu_iteration (
    configSimu_t config,
    unsigned long ** rabbits,
    unsigned long ** rabbitsTmp,
    int year,
    double * cumulProbaLitter,
    double * cumulProbaBabys,
    int lenCumulProbaLitter,
    int lenCumulProbaBabys )
```

fonction qui réalise une itération de la simulation d'évolution de population de lapin

Paramètres

<i>config</i>	configuration de la simulation
<i>rabbits</i>	tableau d'états de lapins
<i>rabbitsTmp</i>	tableau d'états de lapins temporaire
<i>year</i>	année de l'iteration
<i>cumulProbaLitter</i>	tableau de probabilité cumulé pour le nombre de portée
<i>cumulProbaBabys</i>	tableau de probabilité cumulé pour le nombre de bébés par portée
<i>lenCumulProbaLitter</i>	longueur du tableau de proba de portée
<i>lenCumulProbaBabys</i>	longueur du tableau de proba de bébés

Renvoie

nombre de bébés à ajouter

5.8.2.9 realistic_simulation_TS()

```
unsigned long realistic_simulation_TS (
    configSimu_t config )
```

fonction pour réaliser une simulation d'évolution de population de lapin sur plusieurs années avec les configurations données

Paramètres

<i>config</i>	configuration de la simulation
---------------	--------------------------------

Renvoie

nombre de lapin final

5.8.2.10 sum_array()

```
unsigned long sum_array (  
    unsigned long array[],  
    int n )
```

fonction pour faire la somme d'un tableaux de unsigned long

Paramètres

<i>array</i>	tableau
<i>n</i>	nombre d'éléments dans le tableau

Renvoie

somme

5.8.2.11 surviveRabbitYear()

```
int surviveRabbitYear (  
    int old,  
    configSimu_t config )
```

fonction qui dit si un lapin survie ou non (renvoie 1 ou 0)

Paramètres

<i>old</i>	age du lapin
<i>config</i>	configuration de la simulation

Renvoie

1 ou 0

5.9 Référence du fichier rabbitTreeStatus.h

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états.

```
#include "mt19937ar.h"
#include "reverseDistLow.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

Structures de données

- struct `configSimu_t`
configuration d'une simulation

Fonctions

- unsigned long `realistic_simulation_TS` (`configSimu_t` config)

5.9.1 Description détaillée

Programmes pour réaliser une simulation stochastique de l'évolution d'une population de lapin avec des arbres d'états.

Auteur

Aurelien DOUARD et Anthony BERTRAND

Version

0.1

Date

16 mars 2020

5.9.2 Documentation des fonctions

5.9.2.1 `realistic_simulation_TS()`

```
unsigned long realistic_simulation_TS (
    configSimu_t config )
```

fonction pour réaliser une simulation d'évolution de population de lapin sur plusieurs années avec les configurations données

Paramètres

<code>config</code>	configuration de la simulation
---------------------	--------------------------------

Renvoie

nombre de lapin final

5.10 Référence du fichier reverseDistLow.c

Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné

```
#include "reverseDistLow.h"
```

Fonctions

- `double * probaCalc` (int tab[], int len)
Fonction qui calcule la probabilité de chaque élément du tableau.
- `void discDist` (double rand, int tab[], double cumulProba[], int nbClass)
- `int genRand_discDist` (double rand, double cumulProba[], int nbClass)
Fonction qui génère un nombre qui suit une probabilité donné à partir d'un nombre pseudo-aléatoire.
- `int * discDispLoop` (double cumulProba[], int nbClass, int nbExp)
Fonction qui générer des nombre qui suivent une probabilité pour x tirage.
- `double * calcCumulProba` (double tabProb[], int len)
Fonction qui calcule la probabilité cumulé de chaque élément du tableau.

5.10.1 Description détaillée

Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné

Auteur

Aurelien DOUARD

Version

0.1

Date

16 février 2020

5.10.2 Documentation des fonctions

5.10.2.1 calcCumulProba()

```
double * calcCumulProba (  
    double tabProb[],  
    int len )
```

Fonction qui calcule la probabilité cumulé de chaque élément du tableau.

Paramètres

<i>tabProb</i>	tableau rempli avec les probabilité
<i>len</i>	la taille du tableau

Renvoie

un tableau avec la probabilité cumulé de chaque élément

5.10.2.2 discDispLoop()

```
int * discDispLoop (
    double cumulProba[],
    int nbClass,
    int nbExp )
```

Fonction qui générer des nombre qui suivent une probabilité pour x tirage.

Paramètres

<i>nbExp</i>	nombre de tirages
<i>nbClass</i>	nombre de classes différentes du dans le tableau
<i>cumulProba</i>	tableau rempli avec la probabilité cumulé pour chaque classes

Renvoie

tableau rempli avec fréquences

5.10.2.3 genRand_discDist()

```
int genRand_discDist (
    double rand,
    double cumulProba[],
    int nbClass )
```

Fonction qui génère un nombre qui suit une probabilité donné à partir d'un nombre pseudo-aléatoire.

Paramètres

<i>nbClass</i>	nombre de classes différentes du dans le tableau
<i>cumulProba</i>	tableau rempli avec la probabilité cumulé pour chaque classes
<i>rand</i>	nombre pseudo-aléatoire

5.10.2.4 probaCalc()

```
double * probaCalc (
    int tab[],
    int len )
```

Fonction qui calcule la probabilité de chaque élément du tableau.

Paramètres

<i>tab</i>	tableau rempli avec les evenements
<i>len</i>	la taile du tableau

Renvoie

un tableau avec la probabilité de chaque élément

5.11 Référence du fichier reverseDistLow.h

Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné

```
#include "mt19937ar.h"
#include <stdio.h>
#include <stdlib.h>
```

Fonctions

- double * [probaCalc](#) (int tab[], int len)
Fonction qui calcule la probabilité de chaque élément du tableau.
- void **discDist** (double rand, int tab[], double cumulProba[], int nbClass)
- int * [discDispLoop](#) (double cumulProba[], int nbClass, int nbExp)
Fonction qui générer des nombre qui suivent une probabilité pour x tirage.
- double * [calcCumulProba](#) (double tabProb[], int len)
Fonction qui calcule la probabilité cumulé de chaque élément du tableau.
- int [genRand_discDist](#) (double rand, double cumulProba[], int nbClass)
Fonction qui génère un nombre qui suit une probabilité donné à partir d'un nombre pseudo-aléatoire.

5.11.1 Description détaillée

Programmes pour générer des nombre pseudo-aléatoire qui suivent une probabilité donné

Auteur

Aurelien DOUARD

Version

0.1

Date

16 février 2020

5.11.2 Documentation des fonctions

5.11.2.1 calcCumulProba()

```
double* calcCumulProba (
    double tabProb[],
    int len )
```

Fonction qui calcule la probabilité cumulé de chaque élément du tableau.

Paramètres

<i>tabProb</i>	tableau rempli avec les probabilité
<i>len</i>	la taile du tableau

Renvoie

un tableau avec la probabilité cumulé de chaque élément

5.11.2.2 discDispLoop()

```
int* discDispLoop (
    double cumulProba[],
    int nbClass,
    int nbExp )
```

Fonction qui générer des nombre qui suivent une probabilité pour x tirage.

Paramètres

<i>nbExp</i>	nombre de tirages
<i>nbClass</i>	nombre de classes différentes du dans le tableau
<i>cumulProba</i>	tableau rempli avec la probabilité cumulé pour chaque classes

Renvoie

tableau rempli avec fréquences

5.11.2.3 genRand_discDist()

```
int genRand_discDist (
    double rand,
```

```
double cumulProba[],  
int nbClass )
```

Fonction qui génère un nombre qui suit une probabilité donné à partir d'un nombre pseudo-aléatoire.

Paramètres

<i>nbClass</i>	nombre de classes différentes du dans le tableau
<i>cumulProba</i>	tableau rempli avec la probabilité cumulé pour chaque classes
<i>rand</i>	nombre pseudo-aléatoire

5.11.2.4 probaCalc()

```
double* probaCalc (
    int tab[],
    int len )
```

Fonction qui calcule la probabilité de chaque élément du tableau.

Paramètres

<i>tab</i>	tableau rempli avec les evenements
<i>len</i>	la taile du tableau

Renvoie

un tableau avec la probabilité de chaque élément

Index

- add_head
 - rabbit_list.c, [28](#)
 - rabbit_list.h, [33](#)
- addBabys
 - rabbitTreeStatus.c, [37](#)
- ageDisminish
 - configSimu_t, [7](#)
- alive
 - rabbit_t, [11](#)
- calcCumulProba
 - reverseDistLow.c, [43](#)
 - reverseDistLow.h, [46](#)
- CalcSumYears
 - rabbitTreeStatus.c, [37](#)
- calcul_moyenne_long
 - main.c, [17](#)
- checkMallocNull
 - rabbitTreeStatus.c, [38](#)
- compute_S
 - main.c, [17](#)
- configSimu_t, [7](#)
 - ageDisminish, [7](#)
 - initPredator, [7](#)
 - littleRabbitSurvRate, [8](#)
 - maxBabyPerLitter, [8](#)
 - maxLitterPerYear, [8](#)
 - maxRabbitYear, [8](#)
 - maxYear, [8](#)
 - minBabyPerLitter, [8](#)
 - minLitterPerYear, [8](#)
 - probaLittersBabys, [9](#)
 - probaLittersYear, [9](#)
 - rabbitSurvRate, [9](#)
 - rateDisminish, [9](#)
- copyRabitTab
 - rabbitTreeStatus.c, [38](#)
- createRabbitTabZero
 - rabbitTreeStatus.c, [38](#)
- delete_dead
 - rabbit_list.c, [28](#)
 - rabbit_list.h, [33](#)
- delete_head
 - rabbit_list.c, [28](#)
 - rabbit_list.h, [34](#)
- discDispLoop
 - reverseDistLow.c, [44](#)
 - reverseDistLow.h, [46](#)
- display_death
 - rabbit_list.c, [29](#)
 - rabbit_list.h, [34](#)
- display_rabbit
 - rabbit_list.c, [29](#)
 - rabbit_list.h, [34](#)
- display_rabbit_all
 - rabbit_list.c, [29](#)
 - rabbit_list.h, [34](#)
- empty
 - rabbit_list.c, [30](#)
 - rabbit_list.h, [35](#)
- fibonacci, [13](#)
 - fibonacci_explicite, [13](#)
 - fibonacci_recurrence, [14](#)
 - fibonacci_recuratif, [14](#)
- fibonacci.h, [14](#)
 - fibonacci_explicite, [15](#)
 - fibonacci_recurrence, [15](#)
 - fibonacci_recuratif, [16](#)
- fibonacci_explicite
 - fibonacci.c, [13](#)
 - fibonacci.h, [15](#)
- fibonacci_recurrence
 - fibonacci.c, [14](#)
 - fibonacci.h, [15](#)
- fibonacci_recuratif
 - fibonacci.c, [14](#)
 - fibonacci.h, [16](#)
- free_double_tab
 - rabbitTreeStatus.c, [39](#)
- genRand_discDist
 - reverseDistLow.c, [44](#)
 - reverseDistLow.h, [46](#)
- give_birth
 - rabbit.c, [20](#)
 - rabbit.h, [24](#)
- give_birth_all
 - rabbit.c, [20](#)
- head
 - rabbit_list.c, [30](#)
 - rabbit_list.h, [35](#)
- initPredator
 - configSimu_t, [7](#)
- intervalle_de_confiance
 - main.c, [18](#)
- is_alive

- rabbit.c, 21
 - rabbit.h, 24
- is_female
 - rabbit.c, 21
 - rabbit.h, 25
- is_male
 - rabbit.c, 21
 - rabbit.h, 25
- link, 9
 - next, 10
 - rabbit, 10
- link_t, 10
- list_rabbit_t, 10
- littleRabbitSurvRate
 - configSimu_t, 8
- main
 - main.c, 18
- main.c, 16
 - calcul_moyenne_long, 17
 - compute_S, 17
 - intervalle_de_confiance, 18
 - main, 18
 - question1, 18
 - question2_list, 18
 - question2_TS, 19
- makeBabys
 - rabbitTreeStatus.c, 39
- maxBabyPerLitter
 - configSimu_t, 8
- maxLitterPerYear
 - configSimu_t, 8
- maxRabbitYear
 - configSimu_t, 8
- maxYear
 - configSimu_t, 8
- minBabyPerLitter
 - configSimu_t, 8
- minLitterPerYear
 - configSimu_t, 8
- nb_years
 - rabbit_t, 11
- new_list
 - rabbit_list.c, 30
 - rabbit_list.h, 35
- new_rabbit
 - rabbit.c, 22
 - rabbit.h, 25
- new_rabbit2
 - rabbit.c, 22
 - rabbit.h, 26
- next
 - link, 10
- probaCalc
 - reverseDistLow.c, 44
 - reverseDistLow.h, 48
- probaLittersBabys
 - configSimu_t, 9
- probaLittersYear
 - configSimu_t, 9
- question1
 - main.c, 18
- question2_list
 - main.c, 18
- question2_TS
 - main.c, 19
- rabbit
 - link, 10
- rabbit.c, 19
 - give_birth, 20
 - give_birth_all, 20
 - is_alive, 21
 - is_female, 21
 - is_male, 21
 - new_rabbit, 22
 - new_rabbit2, 22
 - realistic_simulation, 22
 - survive, 22
 - survive_all, 23
- rabbit.h, 23
 - give_birth, 24
 - is_alive, 24
 - is_female, 25
 - is_male, 25
 - new_rabbit, 25
 - new_rabbit2, 26
 - realistic_simulation, 26
 - survive, 26
 - survive_all, 26
- rabbit_list.c, 27
 - add_head, 28
 - delete_dead, 28
 - delete_head, 28
 - display_death, 29
 - display_rabbit, 29
 - display_rabbit_all, 29
 - empty, 30
 - head, 30
 - new_list, 30
 - size_list, 30
- rabbit_list.h, 32
 - add_head, 33
 - delete_dead, 33
 - delete_head, 34
 - display_death, 34
 - display_rabbit, 34
 - display_rabbit_all, 34
 - empty, 35
 - head, 35
 - new_list, 35
 - size_list, 36
- rabbit_simu_iteration
 - rabbitTreeStatus.c, 40

- rabbit_t, [11](#)
 - alive, [11](#)
 - nb_years, [11](#)
 - sexe, [11](#)
- rabbitSurvRate
 - configSimu_t, [9](#)
- rabbitTreeStatus.c, [36](#)
 - addBabys, [37](#)
 - CalcSumYears, [37](#)
 - checkMallocNull, [38](#)
 - copyRabitTab, [38](#)
 - createRabbitTabZero, [38](#)
 - free_double_tab, [39](#)
 - makeBabys, [39](#)
 - rabbit_simu_iteration, [40](#)
 - realistic_simulation_TS, [40](#)
 - sum_array, [41](#)
 - surviveRabbitYear, [41](#)
- rabbitTreeStatus.h, [42](#)
 - realistic_simulation_TS, [42](#)
- rateDisminish
 - configSimu_t, [9](#)
- realistic_simulation
 - rabbit.c, [22](#)
 - rabbit.h, [26](#)
- realistic_simulation_TS
 - rabbitTreeStatus.c, [40](#)
 - rabbitTreeStatus.h, [42](#)
- reverseDistLow.c, [43](#)
 - calcCumulProba, [43](#)
 - discDispLoop, [44](#)
 - genRand_discDist, [44](#)
 - probaCalc, [44](#)
- reverseDistLow.h, [45](#)
 - calcCumulProba, [46](#)
 - discDispLoop, [46](#)
 - genRand_discDist, [46](#)
 - probaCalc, [48](#)
- sexe
 - rabbit_t, [11](#)
- size_list
 - rabbit_list.c, [30](#)
 - rabbit_list.h, [36](#)
- sum_array
 - rabbitTreeStatus.c, [41](#)
- survive
 - rabbit.c, [22](#)
 - rabbit.h, [26](#)
- survive_all
 - rabbit.c, [23](#)
 - rabbit.h, [26](#)
- surviveRabbitYear
 - rabbitTreeStatus.c, [41](#)