Projet 0 - Correction

Structures de données et algorithmes

Code de base

```
#include <stdio.h>
#include <stdlib.h>
#define uint unsigned int
#define N 101

int f(int m, int M, uint* s){

    *s ^= (uint)(*s << 13);
    s[0] ^= (uint)(*s >> 17);
    *s ^= (uint)(s[0] << 5);
    return m + (*s%(M-m));
}

void f2(int* a,int* b, int c)
{
        if(c==0 )
                return;
        int t = *b;
        *b = *a;
        *a = t;
}
```

```
void f1(int* a)
{
    int t, i, j;
    for(i=0; i<N; i++)
        for(j=N; j>i; j--)
            f2(&a[j], a+j-1, a[j] < a[j-1]);
}

int median()
{
    uint s;
    int *b=0, i;
    int* a = (int*) malloc(4 * N);
    for(i=0;i<N;i++)
       a[i] = f(-10, 11, &s);
    f1(a);
    *b = a[N/2];
    return *b;
}
```

Noms explicites et respect des conventions

```c
#include <stdio.h>
#include <stdlib.h>
#define uint unsigned int
#define LENGTH 101

int drawInt(int min, int max,
            uint* seed){

    *seed^= (uint)(*seed << 13);
    seed[0] ^= (uint)(*seed >> 17);
    *seed ^= (uint)(seed[0] << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a,int* b, int swap)
{
        if(swap==0 )
                return;
        int tmp = *b;
        *b = *a;
        *a = tmp;
}
```

```c
void bubbleSort(int* array)
{
    int tmp, i, j
    for(i = 0; i<LENGTH; i++)
        for(j=LENGTH; j>i; --j)
            condSwap(&array[j],
                     array+j-1,
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int *med=0, i;
    int* array = (int*) malloc(4 * LENGTH);
    for(i=0;i<LENGTH;i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

Spécification, documentation et commentaires

```
/* ------------------------------------------------------------------ *
 * Draw uniformely an integer in the range [min, max[
 *
 * ARGUMENTS
 * min              The minimum value (inclusive)
 * max              The maximum value (exclusive)
 * seed             A pointer to the seed (will be modified in place)
 *
 * RETURN
 * rand             An integer in the range [min, max[
 * ------------------------------------------------------------------*/
int drawInt(int min, int max, uint* seed)

/* ------------------------------------------------------------------ *
 * Swap the value pointed at if the condition is fulfilled
 *
 * ARGUMENTS
 * a                Pointer to the first value
 * b                Pointer to the second value
 * swap             Whether to swap the values
 * ------------------------------------------------------------------*/
void condSwap(int* a, int* b, int swap)
```

Choix et respect d'un style

```
#include <stdio.h>
#include <stdlib.h>
#define uint unsigned int
#define LENGTH 101

int drawInt(int min, int max,
            uint* seed){
{
    *seed ^= (uint)(*seed << 13);
    *seed[0] ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed[0] << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, int swap)
{
    ____if(swap==0 )
            ____return;
    ____int tmp = *b;
    ____*b = *a;
    ____*a = tmp;
}
```

```
void bubbleSort(int* array)
{
    int tmp, i, j;
    for(i = 0; i<LENGTH; i++)
        for(j=LENGTH; j>i; --j)
            condSwap(&array[j],
                     array+j-1,
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int *med=0, i;
    int* array = (int*) malloc(4 * LENGTH);
    for(i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

Pas de « define »

```
#include <stdio.h>
#include <stdlib.h>
#define uint unsigned int
typedef unsigned int uint;
#define LENGTH 101
const int LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}
void condSwap(int* a, int* b, int swap)
{
    if(swap==0)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
void bubbleSort(int* array)
{
    int tmp, i, j;
    for(i=0; i<LENGTH; i++)
        for(j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int i;
    int* med = 0;
    int* array = (int*) malloc(4 * LENGTH);
    for(i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

« include » inutile

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

typedef unsigned int uint;
const int LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, int swap)
{
    if(swap==0)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
void bubbleSort(int* array)
{
    int tmp, i, j;
    for(i=0; i<LENGTH; i++)
        for(j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int i;
    int* med = 0;
    int* array = (int*) malloc(4 * LENGTH);
    for(i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

Utilisation des types adéquats

```c
#include <stdlib.h>
#include <stdbool.h>

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```c
void bubbleSort(int* array)
{
    int tmp;
    size_t i, j;
    for(i=0; i<LENGTH; i++)
        for(j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    size_t i;
    int* med = NULL;
    int* array = (int*) malloc(4 * LENGTH);
    for(i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return med;
}
```

Style C99

```c
#include <stdlib.h>
#include <stdbool.h>

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```c
void bubbleSort(int* array)
{
    int tmp;
    size_t i, j;
    for(size_t i=0; i<LENGTH; i++)
        for(size_t j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    size_t i;
    int* med = NULL;
    int* array = (int*) malloc(4 * LENGTH);
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med= array[LENGTH/2];
    return *med;
}
```

Portabilité

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

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
void bubbleSort(int* array)
{
    int tmp;
    for(size_t i=0; i<LENGTH; i++)
        for(size_t j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med= NULL;
    int* array = (int*) malloc(4 * LENGTH);
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med= array[LENGTH/2];
    return *med;
}
```

Vérification du `malloc`

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

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
void bubbleSort(int* array)
{
    int tmp;
    for(size_t i=0; i<LENGTH; i++)
        for(size_t j=LENGTH; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med= NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med= array[LENGTH/2];
    return *med;
}
```

Modularité

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

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b, bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
void bubbleSort(int* array, size_t length)
{
    int tmp;
    for(size_t i=0; i<length; i++)
        for(size_t j=length; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med = NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array, LENGTH);
    *med = array[LENGTH/2];
    return *med;
}
```

Compilation

```
gcc main.c Median.c --std=c99 -o prog
```

Le nom du compilateur
Le(s) fichier(s) à compiler
Le flag c99
Le nom du programme de sortie

```
gcc main.c Medianc.  --std=c99 --pedantic -Wall –Wextra -Wmissing-prototypes –o p
```

Des *flags* additionnels pour générer plus de *warnings*

```
main.c:8:5: attention : no previous prototype for ‘drawInt’ [-Wmissing-prototypes]

main.c:17:6: attention : no previous prototype for ‘condSwap’ [-Wmissing-prototypes]

main.c:26:6: attention : no previous prototype for ‘bubbleSort’ [-Wmissing-prototypes]

main.c: In function ‘bubbleSort’:
main.c:28:9: attention : unused variable ‘tmp’ [-Wunused-variable]

Median.c: Hors de toute fonction :
Median.c:31:5: attention : no previous prototype for ‘median’ [-Wmissing-prototypes]
```

Variable inutile

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

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int, int, uint*);
void condSwap(int*, int* bool);
void bubbleSort(int*, size_t);

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b,
              bool swap)
{
   …
}
```

```
void bubbleSort(int* array,
                        size_t length)
{
    int tmp;
    for(size_t i=0; i<length; i++)
        for(size_t j=length; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med = NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

Oubli d'un header

```
#include <stdlib.h>
#include <stdbool.h>
#include "Median.h"

typedef unsigned int uint;
const size_t LENGTH = 101;

int drawInt(int, int, uint*);
void condSwap(int*, int* bool);
void bubbleSort(int*, size_t);

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b,
              bool swap)
{
   …
}
```

```
void bubbleSort(int* array,
                        size_t length)
{
    for(size_t i=0; i<length; i++)
        for(size_t j=length; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med = NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med = array[LENGTH/2];
    return *med;
}
```

Prototypes

```
#include <stdlib.h>
#include <stdbool.h>
#include "Median.h"

typedef unsigned int uint;
static const size_t LENGTH = 101;

int drawInt(int, int, uint*);
void condSwap(int*, int* bool);
void bubbleSort(int*, size_t);

int drawInt(int min, int max,
            uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

void condSwap(int* a, int* b,
              bool swap)
{
   …
}
```

```
void bubbleSort(int* array,
                        size_t length)
{
    for(size_t i=0; i<length; i++)
        for(size_t j=length; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med= NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med= array[LENGTH/2];
    return *med;
}
```

Encapsulation

```
#include <stdlib.h>
#include <stdbool.h>
#include "Median.h"

typedef unsigned int uint;
static const size_t LENGTH = 101;

static int drawInt(int min, int max,
                   uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed%(max-min));
}

static void condSwap(int* a, int* b,
                     bool swap)
{
    if(swap==0)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
static void bubbleSort(int* array,
                       size_t length)
{
    for(size_t i=0; i<length; i++)
        for(size_t j=length; j>i; j--)
            condSwap(&array[j],
                     &array[j-1],
                     array[j]<array[j-1]);
}

int median()
{
    uint seed;
    int* med= NULL;
    int* array = \
      (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array);
    *med= array[LENGTH/2];
    return *med;
}
```

Valgrind

Compilation: `gcc main.c Median.c --std=c99 -g -o prog`

Lancement: `valgrind ./prog`

```
==16678== Memcheck, a memory error detector
==16678== Copyright (C) 2002-2010, and GNU GPL'd, by Julian S
==16678== Using Valgrind-3.6.1 and LibVEX; rerun with -h for
==16678== Command: ./prog
==16678==
==16678== Invalid read of size 4
==16678==    at 0x80484AC: bubbleSort (main.c:32)
==16678==    by 0x8048595: main (main.c:45)
==16678==  Address 0x403f1bc is 0 bytes after a block of size
==16678==    at 0x4006D69: malloc (vg_replace_malloc.c:236)
==16678==    by 0x8048526: main (main.c:39)
==16678==
==16678== Conditional jump or move depends on uninitialised v
==16678==    at 0x804846B: condSwap (main.c:19)
==16678==    by 0x80484EA: bubbleSort (main.c:30)
==16678==    by 0x8048595: main (main.c:45)
==16678==
==16678== Invalid read of size 4
==16678==    at 0x8048478: condSwap (main.c:22)
==16678==    by 0x80484EA: bubbleSort (main.c:30)
==16678==    by 0x8048595: main (main.c:45)
==16678==  Address 0x403f1bc is 0 bytes after a block of size
==16678==    at 0x4006D69: malloc (vg_replace_malloc.c:236)
==16678==    by 0x8048526: main (main.c:39)
```

```
==16678== Invalid write of size 4
==16678==    at 0x8048485: condSwap (main.c:23)
==16678==    by 0x80484EA: bubbleSort (main.c:30)
==16678==    by 0x8048595: main (main.c:45)
==16678==  Address 0x403f1bc is 0 bytes after a block of size 404 alloc'd
==16678==    at 0x4006D69: malloc (vg_replace_malloc.c:236)
==16678==    by 0x8048526: main (main.c:39)
==16678==
==16678== Invalid write of size 4
==16678==    at 0x80485AA: main (main.c:46)
==16678==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==16678==
==16678==
==16678== Process terminating with default action of signal 11 (SIGSEGV)
==16678==  Access not within mapped region at address 0x0
==16678==    at 0x80485AA: main (main.c:46)
==16678==  If you believe this happened as a result of a stack
==16678==  overflow in your program's main thread (unlikely but
==16678==  possible), you can try to increase the size of the
==16678==  main thread stack using the --main-stacksize= flag.
==16678==  The main thread stack size used in this run was 8388608.
==16678==
==16678== HEAP SUMMARY:
==16678==     in use at exit: 404 bytes in 1 blocks
==16678==   total heap usage: 1 allocs, 0 frees, 404 bytes allocated
==16678==
==16678== LEAK SUMMARY:
==16678==    definitely lost: 0 bytes in 0 blocks
==16678==    indirectly lost: 0 bytes in 0 blocks
==16678==      possibly lost: 0 bytes in 0 blocks
==16678==    still reachable: 404 bytes in 1 blocks
==16678==         suppressed: 0 bytes in 0 blocks
==16678== Rerun with --leak-check=full to see details of leaked memory
==16678==
==16678== For counts of detected and suppressed errors, rerun with: -v
==16678== Use --track-origins=yes to see where uninitialised values come from
==16678== ERROR SUMMARY: 5255 errors from 5 contexts (suppressed: 12 from 8)
```

Valgrind - segfault

```
==16678== Invalid write of size 4
==16678==    at 0x80485AA: main (Median:46)
==16678==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==16678==
==16678==
==16678== Process terminating with default action of signal 11 (SIGSEGV)
==16678==  Access not within mapped region at address 0x0
==16678==    at 0x80485AA: main (Median:46)
```

```
*med = array[LENGTH/2];(:46)

int* med = NULL;        (:38)
```

```
int median = array[LENGTH/2]; (:46)
return median; (:47)
```

Valgrind – Conditional jump or move depends on uninitialised value(s)

```
==16678== Conditional jump or move depends on uninitialised value(s)
==16678==    at 0x804846B: condSwap (Median.c:19)
==16678==    by 0x80484EA: bubbleSort (Median.c:30)
```

```
condSwap(&array[j], &array[j-1], array[j]<array[j-1]); (:30)
if(swap==0)                                            (:19)

array[i] = drawInt(-10, 11, &seed) ;                   (:44)

uint seed;                                             (:37)
```

```
#include <time.h>
…
uint seed = (uint) time(NULL);                  (:37)
```

Valgrind – Invalid read

```
==16678== Invalid read of size 4
==16678==    at 0x80484AC: bubbleSort (Median.c:32)
==16678==  Address 0x403f1bc is 0 bytes after a block of size 404 alloc'd
==16678==    at 0x4006D69: malloc (vg_replace_malloc.c:236)
```

```
for(size_t i=0; i<length; i++)
    for(size_t j=length; j>i; j--)
        condSwap(&array[j], &array[j-1], array[j]<array[j-1]);
```

```
for(size_t i=0; i<length; i++)
    for(size_t j=length-1; j>i; j--)
        condSwap(&array[j], &array[j-1], array[j]<array[j-1]);
```

Valgrind – fuite mémoire

```
==16846== HEAP SUMMARY:
==16846==     in use at exit: 404 bytes in 1 blocks
==16846==   total heap usage: 1 allocs, 0 frees, 404 bytes allocated
==16846==
==16846== LEAK SUMMARY:
==16846==    definitely lost: 404 bytes in 1 blocks
```

```
int median()
{
    uint seed = (uint) time(NULL);
    int* array = (int*) malloc(sizeof(int) * LENGTH);
    if (!array)
        return -1;
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, 11, &seed);
    bubbleSort(array, LENGTH);
    int med = array[LENGTH/2];
    free(array);
    return med;
}
```

Solution

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "Median.h"

typedef unsigned int uint;
static const size_t LENGTH = 101;

static size_t drawInt(int min, int max,
                      uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed % (max-min));
}

static void condSwap(int* a, int* b,
                     bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
static void bubbleSort(int* array,
                       size_t length)
{
    for(size_t i=0; i<length; i++)
        for(size_t j=length-1; j>i; j--)
            condSwap(array+j,
                     array+j-1,
                     array[j]<array[j-1]);
}


int median()
{
    uint seed = (uint)time(NULL);

    int* array = \
       (int*) malloc(sizeof(int) * LENGTH);
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, -5, &seed);

    bubbleSort(array, LENGTH);
    int med = array[LENGTH/2];
    free(array);
    return med;
}
```

Solution – C99

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "Median.h"

typedef unsigned int uint;
static const size_t LENGTH = 101;

static size_t drawInt(int min, int max,
                      uint* seed)
{
    *seed ^= (uint)(*seed << 13);
    *seed ^= (uint)(*seed >> 17);
    *seed ^= (uint)(*seed << 5);
    return min + (*seed % (max-min));
}

static void condSwap(int* a, int* b,
                     bool swap)
{
    if(!swap)
        return;
    int tmp = *b;
    *b = *a;
    *a = tmp;
}
```

```
static void bubbleSort(int* array,
                       size_t length)
{
    for(size_t i=0; i<length; i++)
        for(size_t j=length-1; j>i; j--)
            condSwap(array+j,
                     array+j-1,
                     array[j]<array[j-1]);
}
```

Attention aux sorties de tableaux !!

```
int median()
{
    uint seed = (uint)time(NULL);

    int array[LENGTH];
    for(size_t i=0; i<LENGTH; i++)
       array[i] = drawInt(-10, -5, &seed);

    bubbleSort(array, LENGTH);
    free(array);
    return array[LENGTH/2];
}
```

Modularité globale

```
/* ------------------------------------------------------------
 * Return the median value of a odd-size array generated randomly
 *
 * ARGUMENTS
 * length            The length of the random array
 * min               The minimum value which can be drawn (included)
 * max               The maximum value which can be drawn (excluded)
 *
 * RETURN
 * median            The median of the random array
 * ------------------------------------------------------------
 */
int median(size_t length, int min, int max);
```

```
static const size_t LENGTH = 101;
static const int MIN = -10;
static const int MAX = 11;
…
median(LENGTH, MIN, MAX);
```

Dans le `main.c`

Remarques diverses

- Ne pas modifier le header!
- Attention aux *Warnings*
  - `Static`/prototype
- Style
  - Tabulation constante (4 espaces)
  - Taille de ligne (80-100 caractères max.)
- Commentaires
  - **Anglais** ou Français mais il faut choisir
  - Trop de commentaires: nuit la lecture
- Noms de fonctions
  - Clairs et conventionnels
  - Verbes d'action/Nom d'algorithme