InfoEmb TP2

Jérôme Skoda

October 19, 2017

Abstract

Colimaçon:
001 002 003 004 005 006 007 008 009 010
022 023 024 025 026 027 028 029 030 011
021 020 019 018 017 016 015 014 013 012

1 Comment compiler le projet

- make all: Compile tout les fichiers
- make test: Lancement de la série de tests automatiques
- make lib: Génération de la bibliothéque static et dynamique
- make doc: Génération de la documentation (doxygen)
- make rapport: Génération du rapport (latex)
- make clean: Nettoyage du projet (supression des objets et binaires)
- make demo-1 ligne=X col=X: Lancer la démo colimacon
- make demo-2: Lancer la démo horizontal
- make demo-3: Lancer la démo vertical
- make demo-4: Lancer la démo carré
- make demo-5 ligne=X col=X: Lancer la démo colimacon sans print (utile pour le bench)
- make demo-6 ligne=X col=X: Lancer la démo colimacon avec sortie temps (utile pour le bench)
- make bench echantillon=X fichier=X : Generation des bench de temps pour le rapport
- make tgz: Creation de l'achive de rendu

2 Arborescence

- bin : Binaire exécutable
  - demo : Exécutable de démonstration
  - test : Exécutable de test
- lib : Bibliothéque
- doc : Documentation doxygen sous differents formats
- rapport : Source du rapport
- res : Ressources necessaire au projet (fichier de bdd)
- script : Script utilisé pour les test
- src : Source du projet
- bench: Fichier de bench pour le rapport
  - colimacon : Source de colimacon
  - demo : Sources des differentes démonstrations d'utilisation
  - test : Sources des dufferents tests
- sujet.pdf : Sujet du projet
- README.md : Le readme du projet
- rappot.pdf : C'est moi

3 Test disponible

Les test fonctione de la manière suivante: un script bash exécute chacun de binaire de test un par un en enregistrant le sortie standard dans un fichier .out Ensuite il compare avec la commande diff chacun des fichier .out avec la valeur attendu dont la valeur est stoqué dans un fichier .expected

Les fichier .out et .expected sont dans le répertoire src/test/ Le script se situe dans le repertoire script

Les test disponible sont:

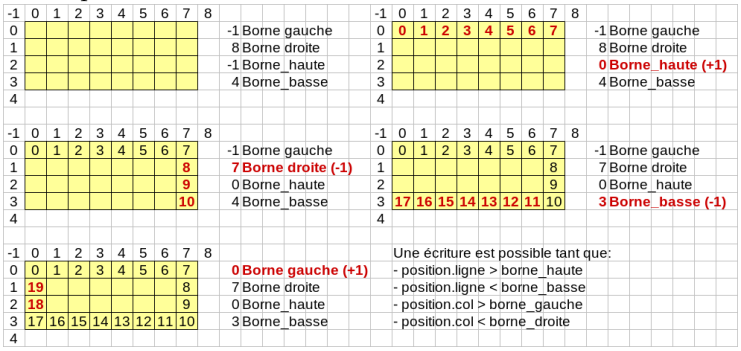
- colimacon 1x1
- colimacon 1x10
- colimacon 10x1
- colimacon 10x2
- colimacon 2x10
- colimacon 10x10
- colimacon 1000x1000 (sans print)

4 Fonctionnement

Le remplissage s'effectue avec un systeme de borne (voir struct borne) et de séquence de direction (voir direction_t). Chacune des écriture sont faite dans un ordre de direction: DROITE puis BAS puis GAUCHE puis HAUT jusqu'à que le tableau soit remplis. L'écriture dans une direction fonctionne de cette manière suivante:

- On calcule la position inital du curseur en fonction des borne et de la direction. (_get_position_initiale)
- On regarde si l'on peux ecrire dans la case calculé (_canWrite)
- S'il est possible d'écrire, on le fait puis on réitére l'operation en avançant le curseur d'une case à chaque fois (_iter_curseur)
- Enfin quand l'écriture n'est plus possible, on avance la borne en fonction de la direction d'écriture (_iter_borne) et l'on passe à la direction suivante

Voici une petite illustration:



5 Estimation du temps

J'ai estimé la réalisation en une journée (environs 7h-12h). Je me suis un peu surestimé, il m'a fallu une demi-journée en plus.

Concernant la partie codage (sans compter la doc), d'après l'historique de mon dépôt, j'estime le temps réellement passé sur le 1h sur colimacon.c et une 30taine de minutes sur les demos et tests.

Tout le reste est essentiellement passé dans toutes les fonctionnalité: makefile, doc, bench, rapport plein de fautes, schéma dégeux sur excel, latex qui dégénère et les sympathiques ifndef de makefile qui ne doivent pas être indenté dans les rules.

6 Choix et problémes

- Bien nommer les choses, ce n'est parfois pas évidant. Exemple: avec un tableau 3x3: les valeurs lors de l'initilisation des bornes sont: borne haute= -1 et borne basse=3 ça peux preter confusion car haute<basse.

- Le script de test, j'en suis particuliérmeent fière, je le resort à chaque fois en projet de c. C'est un simple foreach sur chaque executable de test avec la sortie standard redirigé vers un .out et procéde à un diff sur le .expected.
- J'ai eu des problémes à comprendre comment utiliser perf.
- Le travaille de relecture... Parler à des gens quelle horreur!

7 Prototype

Mon prototype:

```
int ** colimacon(int line, int col);

void delete_colimacon(int** table, int line, int col);
```

L'avantage est que la fonction d'occupe de l'allocation memoire, il n'y a donc pas de risque de l'utilisateur passe un tableau avec le nombre de ligne ou colonne qui ne correspondent pas. Le defaut est que du coup, pour le delete, il y a besoin du nombre de ligne et de colonnes.

```
int colimacon (int ** tab, int line, int col)
```

Ce propotype est interessant car il permet de retourné une erreur par exemple.

8 Paralelisation

Le programme n'est pas dutout parralelisable, cependant avec quelques modification il est possible de le rendre sur chacune des directions.

Il faudra cependant modifier quelques functions:

```
static int _ecriture_direction(int** tab, struct borne* bor, int* val,
  direction_t dir)
```

Devras travailler avec une bornes et un valeur en copie ce qui donnera:

```
static int _ecriture_direction(int** tab, struct borne bor, int val,
  direction_t dir)
```

Il faudra ajouter un fonction qui calcule l'incrementation des valeurs après chaque direction. Par exemple:

```
int _iter_val(struct borne bor, direction_t dir)
```

Enfin dans la boucle de colimacon:

```
  while(1) {

    a lancer sur le thread 1:
    _ecriture_direction(table, borne, valeur, DROITE)

    valeur= _iter_val(bor, DROITE)
```

```
    _iter_borne(bor, DROITE)

    a lancer sur le thread 2:
    _ecriture_direction(table, borne, valeur, BAS)

    valeur= _iter_val(bor, BAS)
    _iter_borne(bor, BAS)

    [...]
  }
```

Le crois qu'il faudra aussi faire de l'allocation en mémoire partagée. Globalement l'idée de l'algorithme est compatible avec une paralelisation.

Cependant, il ne sera pas possible de remplir chaqu'une des cellules en parallèle, car il faudrait trouver un autre mécanisme que celui du curseur qui se déplace vers une direction.

Pour conclure: Ecrire plusieurs rangée complete du tableau c'est possible en gardant la logique de l'algorithme Ecrire chacune des cellule du tableau ce n'est pas possible avec la logique de l'algorithme.

Remarque: Je ne vois pas trop l'intrêt de paralleliser toutes les cases car les mécanismes d'enregistrement et lancement de thread risque d'avoir une performance médiocre.

9 Temps d'execution

La mesure s'est effectué avec la commande make bench [nombre-d'échantillon] [fichier de sortie]

Cette commande lance le script/bench.sh qui consite à faire la moyenne du temps d'execution de bin/demo/6-colimacon-no-print sur n échantillons et ce sur differente taille.

Voici le resultat sur 1 000 échantillons:

Taille	Laptop	Desktop
10x10	0.035371 ms	0.021329 ms
100x100	0.209286 ms	0.112920 ms
1000x1000	15.301931 ms	7.218361ms
10000x10000	1.513250620s	.989898073s

- Laptop:
  - CPU: i5-2430M CPU @ 2.40GHz 2 cores 4 threads
  - RAM: 16Go @ 1 600MHz
- Desktop:
  - CPU: i7 4790K @ 4.3GHz 4 cores 8 threads
  - RAM: 32Go @ 1 600MHz

10 Accès mémoire

```
int ** colimacon(int line, int col);
```

Cette fonction accède 0 fois au tableau en lecture et N fois en écriture.

```
void print_colimacon(int** table, int line, int col);
```

Cette fonction accède N fois au tableau en lecture et 0 fois en écriture.

11 Occupation mémoire

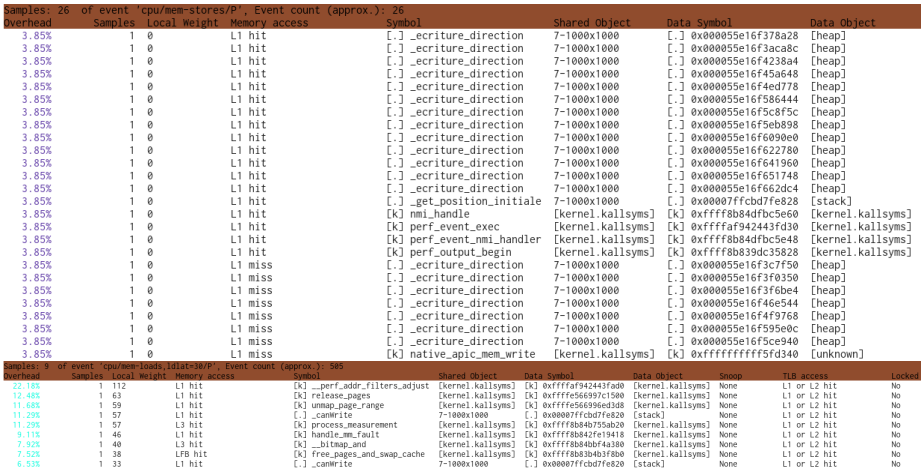
Sur le profilage de bin/test/7-1000x1000, on remarque que la plupart des écritures memoire s'effectue dans le cache L1 et qu'il n'y pas enormément de latance (3.85

Concenrnat la lecture, la fonction _can_write a le plus de latance (11.29% et 6.53%) et utilise la pile.

_get_position_initiale utilise la pile _ecritrue_direction utilise le tas.

Petit memo:

```
sudo perf mem record bin/test/7-1000x1000
sudo perf mem report
```



```
Samples: 26  of event 'cpu/mem-stores/P', Event count (approx.): 26
Overhead       Samples  Local Weight  Memory access               Symbol                       Shared Object       Data Symbol             Data Object
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f378a28  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f3aca8c  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f4238a4  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f45a648  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f4ed778  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f586444  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f5c8f5c  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f5eb898  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f6090e0  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f622780  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f641960  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f651748  [heap]
   3.85%             1  0             L1 hit                      [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f662dc4  [heap]
   3.85%             1  0             L1 hit                      [.] _get_position_initiale   7-1000x1000         [.] 0x00007ffcbd7fe828  [stack]
   3.85%             1  0             L1 hit                      [k] nmi_handle               [kernel.kallsyms]   [k] 0xffff8b84dfbc5e60  [kernel.kallsyms]
   3.85%             1  0             L1 hit                      [k] perf_event_exec          [kernel.kallsyms]   [k] 0xffffaf942443fd30  [kernel.kallsyms]
   3.85%             1  0             L1 hit                      [k] perf_event_nmi_handler   [kernel.kallsyms]   [k] 0xffff8b84dfbc5e48  [kernel.kallsyms]
   3.85%             1  0             L1 hit                      [k] perf_output_begin        [kernel.kallsyms]   [k] 0xffff8b839dc35828  [kernel.kallsyms]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f3c7f50  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f3f0350  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f3f6be4  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f46e544  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f4f9768  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f595e0c  [heap]
   3.85%             1  0             L1 miss                     [.] _ecriture_direction      7-1000x1000         [.] 0x000055e16f5ce940  [heap]
   3.85%             1  0             L1 miss                     [k] native_apic_mem_write    [kernel.kallsyms]   [k] 0xffffffffff5fd340  [unknown]
Samples: 9  of event 'cpu/mem-loads,ldlat=30/P', Event count (approx.): 505
Overhead       Samples  Local Weight  Memory access               Symbol                       Shared Object       Data Symbol             Data Object         Snoop       TLB access              Locked
  22.18%             1  112           L1 hit                      [k] __perf_addr_filters_adjust  [kernel.kallsyms]   [k] 0xffffaf942443fad0  [kernel.kallsyms]   None        L1 or L2 hit            No
  12.48%             1  63            L1 hit                      [k] release_pages            [kernel.kallsyms]   [k] 0xffffe566997c1500  [kernel.kallsyms]   None        L1 or L2 hit            No
  11.68%             1  59            L1 hit                      [k] unmap_page_range         [kernel.kallsyms]   [k] 0xffffe566996ed3d8  [kernel.kallsyms]   None        L1 or L2 hit            No
  11.29%             1  57            L1 hit                      [.] _canWrite                7-1000x1000         [.] 0x00007ffcbd7fe820  [stack]             None        L1 or L2 hit            No
  11.29%             1  57            L3 hit                      [k] process_measurement      [kernel.kallsyms]   [k] 0xffff8b84b755ab20  [kernel.kallsyms]   None        L1 or L2 hit            No
   9.11%             1  46            L1 hit                      [k] handle_mm_fault          [kernel.kallsyms]   [k] 0xffff8b842fe19418  [kernel.kallsyms]   None        L1 or L2 hit            No
   7.92%             1  40            L3 hit                      [k] __bitmap_and             [kernel.kallsyms]   [k] 0xffff8b84bbf4a380  [kernel.kallsyms]   None        L1 or L2 hit            No
   7.52%             1  38            LFB hit                     [k] free_pages_and_swap_cache  [kernel.kallsyms]   [k] 0xffff8b83b4b3f8b0  [kernel.kallsyms]   None        L1 or L2 hit            No
   6.53%             1  33            L1 hit                      [.] _canWrite                7-1000x1000         [.] 0x00007ffcbd7fe820  [stack]             None        L1 or L2 hit            No
```

12 Relecture

Liang m'a dit qu'ils n'ont rien compris au code. Du coups, j'ai augmenté la documentation. Li Xiang m'a dit j'ai fait la même chose que lui sauf que mes fonctions auxilaire rendais le tout plus structuré.

InfoEmb TP2

Généré par Doxygen 1.8.13

Contents

1 Commandes

- **make all** : Compile tout les fichiers
- **make test** : Lancement de la série de tests automatiques
- **make lib** : Génération de la bibliothéque static et dynamique
- **make doc** : Génération de la documentation (doxygen)
- **make rapport** : Génération du rapport (latex)
- **make clean** : Nettoyage du projet (supression des objets et binaires)
- **make demo-1 ligne=X col=X** : Lancer la démo colimacon
- **make demo-2**: Lancer la démo horizontal
- **make demo-3**: Lancer la démo vertical
- **make demo-4**: Lancer la démo carr
- **make demo-5 ligne=X col=X** : Lancer la démo colimacon sans print (utile pour le bench)
- **make demo-6 ligne=X col=X** : Lancer la démo colimacon avec sortie temps (utile pour le bench)

Arborescence

- **bin** : Binaire exécutable
  - **demo** : Exécutable de démonstration
  - **test** : Exécutable de test
- **lib** : Bibliothéque
- **doc** : Documentation doxygen sous differents formats
- **rapport** : Source du rapport
- **res** : Ressources necessaire au projet (fichier de bdd)
- **script** : Script utilisé pour les test
- **src** : Source du projet
  - **bdd** : Source de la bibliothéque
  - **demo** : Sources des differentes démonstrations d'utilisation
  - **test** : Sources des dufferents tests
- *sujet.pdf* : Sujet du projet
- *README.md* : C'est moi
- *rappot.pdf* : Le rapport du projet

Tests disponibles

- colimacon 1x1
- colimacon 1x10
- colimacon 10x1
- colimacon 10x2
- colimacon 2x10
- colimacon 10x10
- colimacon 1000x1000 (sans print)

2 Navigation

- **colimacon** : Source de colimacon
- **demo** : Exemple d'utilisation de la BDD
- **test** : Test unitaire de la BDD

3 Index des classes

3.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

4 Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

5 Documentation des classes

5.1 Référence de la structure borne

Borne de remplissage.

Attributs publics

- int haute

  Borne haute.
- int droite

  Borne droite.
- int basse

  Borne basse.
- int gauche

  Borne gauche.

5.1.1 Description détaillée

Borne de remplissage.

5.1.2 Documentation des données membres

5.1.2.1 basse

```
int borne::basse
```

Borne basse.

5.1.2.2 droite

```
int borne::droite
```

Borne droite.

5.1.2.3 gauche

```
int borne::gauche
```

Borne gauche.

5.1.2.4 haute

```
int borne::haute
```

Borne haute.

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

- src/colimacon/colimacon.c

5.2 Référence de la structure curseur

Cursuer de postion de remplissage.

Attributs publics

- int col
  Position dans les collones.
- int ligne
  Position dans les lignes.

5.2.1 Description détaillée

Cursuer de postion de remplissage.

5.2.2 Documentation des données membres

5.2.2.1 col

```
int curseur::col
```

Position dans les collones.

5.2.2.2 ligne

```
int curseur::ligne
```

Position dans les lignes.

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

- src/colimacon/colimacon.c

6 Documentation des fichiers

6.1 Référence du fichier README.md

6.2 Référence du fichier src/README.md

6.3 Référence du fichier src/colimacon/colimacon.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "colimacon.h"
```

Graphe des dépendances par inclusion de colimacon.c:

6.4 Référence du fichier src/colimacon/colimacon.h

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Fonctions

- int ** colimacon (int line, int col)

  Allocation d'un tableau et remplissage en colimacon.
- void print_colimacon (int **table, int line, int col)

  Affiche un tableau en colimaçon.
- void delete_colimacon (int **table, int line, int col)

  Supression d'un tableau.

6.4.1 Documentation des fonctions

6.4.1.1 colimacon()

```
int** colimacon (
            int line,
            int col )
```

Allocation d'un tableau et remplissage en colimacon.

Paramètres

in	*line*	nombre de ligne
in	*col*	nombre de colonne

Renvoie

retourne un pointer vers le tableau

6.4.1.2 delete_colimacon()

```
void delete_colimacon (
            int ** table,
            int line,
            int col )
```

Supression d'un tableau.

Paramètres

in	*table*	table à supprimer
in	*line*	nombre de ligne de la table
in	*col*	nombre de colonnes de la table

6.4.1.3 print_colimacon()

```
void print_colimacon (
            int ** table,
            int line,
            int col )
```

Affiche un tableau en colimaçon.

Paramètres

in	*table*	table à imprimmer
in	*line*	nombre de ligne de la table
in	*col*	nombre de colonnes de la table

6.5 Référence du fichier src/demo/1-colimacon.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 1-colimacon.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration Affichage en fonction des parammetres Utilisation: colimacon [ligne] [col].

6.5.1 Documentation des fonctions

6.5.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration Affichage en fonction des parammetres Utilisation: colimacon [ligne] [col].

6.6 Référence du fichier src/demo/2-horizontal.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 2-horizontal.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration Taille 2x10.

6.6.1 Documentation des fonctions

6.6.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration Taille 2x10.

6.7 Référence du fichier src/demo/3-vertical.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 3-vertical.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration Taille 10x2.

6.7.1 Documentation des fonctions

6.7.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration Taille 10x2.

6.8 Référence du fichier src/demo/4-carre.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 4-carre.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration Taille 10x10.

6.8.1 Documentation des fonctions

6.8.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration Taille 10x10.

6.9 Référence du fichier src/demo/5-colimacon-no-print.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 5-colimacon-no-print.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration sans print Utilisation: colimacon [ligne] [col].

6.9.1 Documentation des fonctions

6.9.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration sans print Utilisation: colimacon [ligne] [col].

6.10 Référence du fichier src/demo/6-colimacon-time.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Graphe des dépendances par inclusion de 6-colimacon-time.c:

Fonctions

- int main (int argc, char **argv)

  Programme de démonstration affichant le temps de génération Utilisation: colimacon [ligne] [col].

6.10.1 Documentation des fonctions

6.10.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de démonstration affichant le temps de génération Utilisation: colimacon [ligne] [col].

6.11 Référence du fichier src/test/1-1x1.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 1-1x1.c:

Fonctions

- int main (int argc, char **argv)

  Programme de test Taille 1x1.

6.11.1 Documentation des fonctions

6.11.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 1x1.

6.12 Référence du fichier src/test/2-1x10.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 2-1x10.c:

Fonctions

- int main (int argc, char **argv)
  Programme de test Taille 1x10.

6.12.1 Documentation des fonctions

6.12.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 1x10.

6.13 Référence du fichier src/test/3-10x1.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 3-10x1.c:

Fonctions

- int main (int argc, char **argv)
  Programme de test Taille 10x1.

6.13.1 Documentation des fonctions

6.13.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 10x1.

6.14 Référence du fichier src/test/4-10x2.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 4-10x2.c:

Fonctions

- int main (int argc, char **argv)

  Programme de test Taille 10x2.

6.14.1 Documentation des fonctions

6.14.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 10x2.

6.15 Référence du fichier src/test/5-2x10.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 5-2x10.c:

Fonctions

- int main (int argc, char **argv)

  Programme de test Taille 2x10.

6.15.1 Documentation des fonctions

6.15.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 2x10.

6.16 Référence du fichier src/test/6-10x10.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 6-10x10.c:

Fonctions

- int main (int argc, char **argv)

  Programme de test Taille 10x10.

6.16.1 Documentation des fonctions

6.16.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test Taille 10x10.

6.17 Référence du fichier src/test/7-1000x1000.c

```
#include "../colimacon/colimacon.h"
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de 7-1000x1000.c:

Fonctions

- int main (int argc, char **argv)

  Programme de test (sans print) Taille 1000x1000.

6.17.1 Documentation des fonctions

6.17.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Programme de test (sans print) Taille 1000x1000.