**Université de Strasbourg**
**UFR de mathématique et d'informatique**

*L2 informatique*
*automne 2016*

# Structures de données et algorithmes

TP1 : Plus grande sous-suite strictement croissante

## Avant de commencer

Pour que toute la promotion ait la même structure de répertoire, je vous recommande de suivre le canevas suivant. Dans votre répertoire personnel, créez un sous-répertoire de nom `SDA1`, puis dans ce répertoire un sous-répertoire de nom `TP1`. Vous ajouterez des sous-répertoires au fur et à mesure des TP. Dans le répertoire `SDA1`, faites également deux sous-répertoires de noms respectifs `include` et `lib`, dans lesquels vous rangerez respectivement les fichiers header communs à tous vos projets et les fichiers binaires (.o) ou les bibliothèques (.a) correspondantes.

## Plus grande sous-suite strictement croissante : version mutable

Nous avons vu en cours que pour résoudre le problème de la plus grande sous-suite croissante d'une liste, nous avons introduit deux types abstraits `suite` et `sous_suite` que nous avons (incomplètement) spécifiés par une signature se traduisant par les deux fichiers header `suites.h` et `ssuites.h` que vous pouvez récupérer sur moodle.

Dans cette implantation, on considère des structures de données *non-mutables*, c'est-à-dire que quand un "objet" est créé, on ne le modifie pas, on produit de nouveaux objets qui sont des versions modifiées de l'original. Cela implique qu'il y a recopie des arguments et des valeurs retournées. Comme les structures considérées contiennent des tableaux, ce temps de recopie est coûteux et pour améliorer l'efficacité, on vous demande d'en faire une version mutable en considérant un point sur une structure plutôt que la structure elle même.

**1.** Cela conduit à de nouvelles versions de fichiers `suites.h`, `suites.c`, `ssuites.h` et `ssuites.c` qu'on vous demande d'écrire et qui serviront de base pour la suite. Bien sûr, il est possible de reprendre le code source des fichiers initiaux qui vous sont donnés sur moodle et de les transformer. Pour vous aider, voici le début de chaque fichier transformé :

Pour le fichier **suites.h**

```
 /*------------------------------------------------------------------------*
 *         fichier suites.h         version MUTABLE                       *
 *------------------------------------------------------------------------ */
#include <stdbool.h>

#ifndef __SUITES__
#define __SUITES__
#define LG_MAX 128

typedef struct s_suite {
                          int val[LG_MAX];   // valeur des termes
                          int lg;            // longueur de la suite
                         } *suite;               // changé

                                          // effets de bord
suite s_vide();
suite s_del(suite u);                        // l'espace est désalloué
suite s_const(int lg, int valeur);
suite s_t2s(int tab[], int nb);
suite s_adj(int t, suite u);              // la suite u est modifiée
// ....
// à compléter
```

Pour le fichier **suites.c**

```
 /*-----------------------------------------*
 *         fichier suites.c                 *
 *          version MUTABLE                 *
 *-----------------------------------------*/
#include "include/suites.h"
#include <stdio.h>
#include <stdlib.h>
/*
typedef struct s_suite {
                         int val[LG_MAX];   // valeur des termes
                         int lg;            // longueur de la suite
                        } *suite;
*/

suite s_vide()
{
    suite u = (suite)malloc(sizeof(struct s_suite));
    u->lg = 0;
    return u;
}

void s_del(suite u)
{
    free(u);
}
// ....
// à compléter
```

Pour le fichier **ssuites.h**

```
 /*-----------------------------------------*
 *         fichier ssuites.h                *
 *         version MUTABLE                  *
 *-----------------------------------------*/
#include "suites.h"

#ifndef __S_SUITES__
#define __S_SUITES__

typedef struct s_ssuite {
                     suite ms;                // changé
                     int prem;
                     int ch[LG_MAX];
                } *sous_suite;                // changé

sous_suite ss_vide(suite u);
int ss_longueur(sous_suite su);
int ss_ieme(sous_suite su, int i);     // prec : 0 <= i < ss_longueur(su)

/*---------fonction ss_modifie_suiv----------------*/
* prec : 0<= i < n_suiv < ss_longeur(su)           *
* effet de bord : la sous-suite su est modifiée    *
*-------------------------------------------------*/
sous_suite ss_modifie_suiv(sous_suite su, int i, int n_suiv);
// ...
// à compléter
```

Pour le fichier **ssuites.c**

```
 /*-----------------------------------------*
 *         fichier ssuites.c               *
 *         version MUTABLE                 *
 *-----------------------------------------*/
#include "suites.h"
#include "ssuites.h"
#include <stdlib.h>
#include <stdio.h>

/*
typedef struct  s_ssuite  {
                    suite *ms;
                    int prem;
                    int ch[LG_MAX];
                } sous_suite;
*/

sous_suite ss_vide(suite u)
{
    sous_suite su = (sous_suite)malloc(sizeof(struct s_ssuite));

    su->ms = u;
    su->prem = 0;
    for(int i=0; i<LG_MAX; i++) su->ch[i] = -1;

    return su;
}

void ss_del(sous_suite su)
{
    free(su);
}

// ... à compléter ...
```

Compilez vos fichiers avec l'option -c pour vérifier que vous n'avez pas d'erreur !

```
~$gcc -c suites.c
~$gcc -c ssuites.c
```

**2.** Le fichier `test3.c` est également disponible sur la page moodle dédiée à ce cours. Utilisez ce fichier pour tester votre nouvelle version des suites et des sous-suites. Avez-vous dû corriger des erreurs pour que le code compile ? pour que l'exécutable fonctionne correctement ?

## De nouvelles sous-suites

Dans cette section, on examine quelques fonctions pour produire ou modifier une sous-suite comme nous l'avons vu en TD.

**1.** On vous demande notamment de programmer les fonctions correspondant aux prototypes suivants :

```
/*---------------------------------------*
*       fichier ssuites_extended.h      *
*       version MUTABLE                 *
*---------------------------------------*/
#include <stdbool.h>
#include "suites.h"
#include "ssuites.h"

sous_suite ss_supprime(sous_suite su, int ind_of_u);
      // supprime le terme u_(ind_of_u) de la sous-suite su
      // si ce terme n'est pas dans su, celle-ci reste inchangée
      // prec : à compléter
      // effet de bord : su est modifiée

suite ss_ss2s(sous_suite su);
     // produit une suite dont les termes sont ceux de su
     // prec : aucune (on doit bien avoir une sous-suite)
     // effet de bord aucun

sous_suite ss_id(suite u);
     // produit la sous-suite dont les termes sont ceux de u
     // effet de bord : aucun

sous_suite ss_n_prem(suite u, int n);
     // produit la sous-suite des n premiers termes de u
     // prec : 0 < n <= s_longueur(u)
     // effet de bord aucun

sous_suite ss_1_sur_2(suite u);
     // produit la sous_suite des termes de u pris un sur deux à partir de u_0
     // prec: aucune
     // effet de bord aucun

sous_suite ss_croissante_de_u0(suite u);
     // produit la plus grande sous-suite strictement croissante de termes
     // consécutifs
     // depuis u0
     // prec : aucune
     // effet de bord aucun

sous_suite ss_croissante(suite u);
     // produit la plus grande sous-suite strictement croissante de termes
     // consécutifs
     // prec : aucune
     // effet de bord aucun

bool s_strict_croissante(suite u);
     // retourne vrai ssi la suite u est strictement croissante

sous_suite ss_extract(suite u, suite mod);
     // produit la sous_suite des termes u_(mod_i)
     // prec : la suite mod est strictement croissante et
     //        0< s_longueur(mod) <= s_longueur(u) et
     //        0 <= mod_i  <= s_longueur(u)
```

**2.** Programmez des fonctions permettant d'engendrer des suites pour construire par composition les premières sous-suites demandées à la question précédente.