TD1 — Programmes et modules C

1. *Lecture commentée d'un programme C*

Distribuer le polycopié et le mémento C.

Un programme C est constitué d'unités. Certaines de ces unités sont des unités de bibliothèques fournies par l'implémentation.

D'autres sont écrites par le programmeur. Elles ont toutefois les mêmes caractéristiques.

Les principales unités sont les sous-programmes (fonctions et procédures), et les paquetages. Un paquetage (ou module) est un regroupement d'éléments. Le programme principal est une procédure.

Un paquetage `A` *est constitué d'un fichier spécification* `a.h` *(l'interface), et éventuellement d'un fichier corps* `a.c` *(l'implémentation).*

Structure :

- *le fichier* `a.h` *contient les* déclarations *des éléments du paquetage (types, fonctions, variables globales)*
- *le fichier* `a.c` *contient la* définition/implémentation *des éléments déclarés + des définitions locales (fonctions/procédures locales intermédiaires)*

Le programme principal *est une fonction nommée* main.

Ici, le paquetage `type_tableau` *est uniquement constitué d'une interface (la définition du type* `tableau_entiers`*).*

Les deux autres ont une partie implémentation : chaque fichier .h contient les profils des sous-programmes et le fichier .c contient les corps de ces sous-programmes.

Un fichier implémentation peut aussi contenir d'autres sous-programmes qui lui sont propres et non déclarés dans le fichier .h.

Utilisation d'éléments définis dans d'autres paquetages : clause `#include` *(inclusion d'un fichier externe).*

Commenter les éléments de syntaxe si nécessaire.

2. *Exercices*

Exercice 1. Lire le programme C fourni en annexe.

1. Dessiner un schéma de la structure de ce programme (modules et programme principal), en indiquant les dépendances entre modules.
2. Quel est le résultat de l'exécution du programme ?
3. Quelles sont les contraintes sur les valeurs en entrée ?

Comportement du programme : lit un tableau dans un fichier nommé `data1.txt`*. Ce fichier contient la taille du tableau, suivi des valeurs du tableau. Le programme inverse les valeurs du tableau (la première valeur se retrouve en dernier, etc.), et écrit ce tableau inversé dans un fichier nommé* `data1_renverse.txt`*.*

Ce programme est constitué des paquetages :

- `type_tableau`,
- `operations_tableau`,

— `es_tableau`

et du programme principal `td1` *(dans lequel se trouve la fonction* `main`*).*

Les paquetages `operations_tableau` *et* `es_tableau` *dépendent du paquetage* `type_tableau`*. Le programme principal dépend directement des trois paquetages.*

Contraintes sur les entrées : la taille du tableau doit être comprise entre 0 et 10000. La première valeur dans le fichier `data1.txt` *doit donc être un entier entre 0 et 10000. Cette valeur n doit être suivie d'au moins n entiers.*

NB : une spécification peut très bien être «sur-contrainte», on peut par exemple considérer qu'un «tableau» contient au moins 1 valeur.

Exercice 2. Modifier le programme pour qu'il affiche la valeur maximum du tableau en entrée. Les contraintes sur les valeurs d'entrées changent-elles avec cette nouvelle spécification ?

Par exemple :

```c
#include "es_tableau.h"
#include "operations_tableau.h"
#include "type_tableau.h"
#include <stdio.h>

int main() {
  tableau_entiers tableau;
  int i, max;
  /* Lire le tableau dans le fichier "data1.txt" */
  lire_tableau("data1.txt", &tableau);
  /* Calcul du maximum */
  max = tableau.tab[0];
  for (i = 1; i < tableau.taille; i++) {
    if (tableau.tab[i] > max) {
      max = tableau.tab[i];
    }
  }
  /* Affichage du maximum */
  printf("Maximum : %d\n", max);
}
```

Concernant les contraintes sur les valeurs d'entrées : si on affiche «la» valeur maximum, c'est qu'elle existe, donc qu'il y a au moins 1 élément dans le tableau.

A. Programme principal

Programme principal `td1` composé d'un seul fichier `td1.c` :

```c
#include "es_tableau.h"
#include "operations_tableau.h"
#include "type_tableau.h"

int main() {
  tableau_entiers tableau;

  /* Lire le tableau dans le fichier "data1.txt" */
  lire_tableau("data1.txt", &tableau);
  /* Renverser le tableau */
  renverser(&tableau);
  /* Ecrire le tableau renversé dans le fichier "data1_renverse.txt" */
  ecrire_tableau("data1_renverse.txt", &tableau);
}
```

B. Paquetage `type_tableau`

Paquetage `type_tableau` composé du seul fichier « spécification » ou « interface » `type_tableau.h` :

```c
#ifndef _TYPE_TABLEAU_H_
#define _TYPE_TABLEAU_H_

#define TAILLE_MAX 10000

/* Définition du type vecteur_entiers : tableau d'entiers de taille TAILLE_MAX */
typedef int vecteur_entiers[TAILLE_MAX];

/* Structure contenant un tableau (de taille TAILLE_MAX) et un entier
   taille : le nombre d'entiers du tableau */
typedef struct {
  int taille;
  vecteur_entiers tab;
} tableau_entiers;

#endif /* _TYPE_TABLEAU_H_ */
```

C. Paquetage es_tableau

Paquetage es_tableau composé :

— d'un fichier « spécification » ou « interface » es_tableau.h :

```c
#ifndef _ES_TABLEAU_H_
#define _ES_TABLEAU_H_

#include "type_tableau.h"

void lire_tableau(char *nomFichier, tableau_entiers *t);

void ecrire_tableau(char *nomFichier, tableau_entiers *t);

#endif /* _ES_TABLEAU_H_ */
```

— d'un fichier « corps » ou « implémentation » es_tableau.c :

```c
#include "es_tableau.h"
#include <stdio.h>

void lire_tableau(char *nomFichier, tableau_entiers *t) {
  /* Descripteur du fichier d'entrée */
  FILE *fichier;
  int i;

  /* Ouverture du fichier en lecture */
  fichier = fopen(nomFichier, "r");

  /* Lecture de la taille du tableau */
  fscanf(fichier, "%d", &(t->taille));

  /* Lecture des valeurs du tableau */
  for (i = 0; i < t->taille; i++) {
    fscanf(fichier, "%d", &(t->tab[i]));
  }
  fclose(fichier);
}

void ecrire_tableau(char *nomFichier, tableau_entiers *t) {
  /* Descripteur du fichier de sortie */
  FILE *fichier;
  int i;

  /* Créer et ouvrir le fichier en écriture */
  fichier = fopen(nomFichier, "w");

  /* Écrire la taille du tableau dans le fichier */
  fprintf(fichier, "%d\n", t->taille);

  /* Écrire les valeurs du tableau */
  for (i = 0; i < t->taille; i++) {
    fprintf(fichier, "%d\n", t->tab[i]);
  }
  /* Fermeture du fichier */
  fclose(fichier);
}
```

D. Paquetage operations_tableau

Paquetage operations_tableau composé :

— d'un fichier « spécification » ou « interface » operations_tableau.h :

```c
#ifndef _OPERATIONS_TABLEAU_H_
#define _OPERATIONS_TABLEAU_H_

#include "type_tableau.h"

void renverser(tableau_entiers *t);

#endif /* _OPERATIONS_TABLEAU_H_ */
```

— d'un fichier «corps» ou « implémentation » operations_tableau.c :

```c
#include "operations_tableau.h"

/* Permute les entiers p et q
   Données-résultats : p, q : entiers
   Post-condition : les valeurs de p et q ont été permutées
 */
void permuter(int *p, int *q) {
  int aux;

  aux = *p;
  *p = *q;
  *q = aux;
}

/* renverse l'ordre des elements du tableau T
   l'algorithme utilisé est le suivant
   soit n le nombre de valeurs dans le tableau T
   on permute l'élément d'indice 1 de T avec l'élément d'indice n
   on permute l'élément d'indice 2 de T avec l'élément d'indice n-1
   ...
   on permute l'élément d'indice i1 de T avec l'élément d'indice i2,
     avec i1<i2 et i1+i2=n+1
   ...
   jusqu'à la moitié du tableau c'est àdire lorsque i1>=i2
 */
void renverser(tableau_entiers *t) {
  /* Indices de parcours du tableau */
  int i1, i2;

  i1 = 0;
  i2 = t->taille - 1;

  while (i1 < i2) {
    /* Effectuer la permutation des éléments d'indices i1 et i2 de t */
    permuter(&(t->tab[i1]), &(t->tab[i2]));

    /* Incrémenter i1 et décrémenter i2 */
    i1 = i1 + 1;
    i2 = i2 - 1;
  }
}
```