Perfectionnement à la programmation en C

Licence 2 Informatique – Examen 2024 – 2025

jeudi 22 mai 2025

Deux feuilles A4 de notes manuscrites personnelles autorisées, tout autre document interdit.

Ce devoir est constitué de 17 exercices. Le barême est donné à titre indicatif par partie. Voici quelques conseils :

— lire l'intégralité du sujet avant de commencer

— respecter les **conventions** étudiées. Une fonction qui "marche" peut être incorrecte

— lorsqu'il est demandé de définir une fonction, seuls les paramètres principaux sont précisés. Il est ainsi possible d'**ajouter d'autres paramètres**

— les seules fonctions externes, types et macro-définitions qu'il est autorisé d'utiliser sont ceux apportés par les **en-têtes `stdio.h`**, **`stdlib.h`** et **`assert.h`**

— il est autorisé de déclarer des **types** et de définir des **fonctions intermédiaires** si besoin est

---

Rappel de représentation de la mémoire

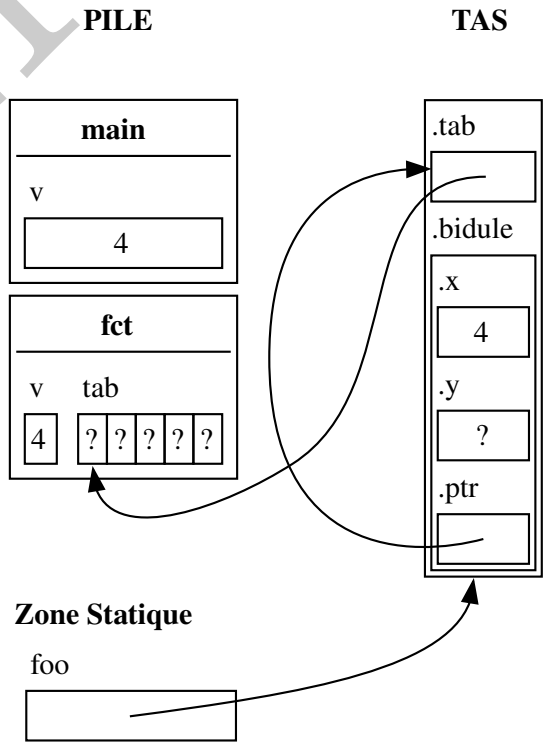
Dans un schéma de la mémoire, on distingue jusqu'à 3 zones : la pile (toujours présente pendant l'exécution), la zone statique et le tas. Il n'est pas nécessaire de représenter une zone si elle n'est pas utilisée.

```c
struct A {
  int **tab;
  struct B {
    int x, y;
    int ***ptr;
  } bidule;
};

// Variable globale
struct A *foo;

void fct(int v) {
  int *tab[5];
  foo = malloc(sizeof(struct A));
  foo->tab = tab;
  foo->bidule.ptr = &foo->tab;
  foo->bidule.x = v;
  // Schéma de la mémoire à cette
      ligne
  free(foo);
}

int main(void) {
  int v = 4;
  f(v);
  return 0;
}
```



Exemple de schéma pour le code ci-contre.

Analyse de code *(11,5 points)*

Voici un court code manipulant une chaine de caractères via l'adresse de son premier caractère et sa taille. Cette méthode permet de s'affranchir du caractère nul de fin de chaine dont on ne se préoccupera pas dans les 4 exercices suivants.

```
1 #include <stdio.h>
2
3 // Représentation d'une chaine par son adresse et sa taille
4 typedef struct {
5     char *data; // Pointeur vers le premier caractère d'une chaine
6     int len;    // Nombre de caractères
7 } Str;
8
9 // Affiche la chaine représentée par un objet de type Str
10 void putstr(Str s) {
11     for (int i = 0; i < s.len; i++)
12         printf("%c", s.data[i]);
13     return;
14 }
15
16 // Renvoie la sous chaine de `len` caractères commençant à l'indice
       `start`
17 Str substr(Str s, int start, int len) {
18     Str ret;
19     ret.data = &s.data[start]; // ou s.data + start
20     ret.len = len;
21     return ret;
22 }
23
24 int main(void) {
25     Str mot = {
26         .data = "Bonjour",
27         .len = 7
28     };
29     // Construction de la sous-chaine `on` de `B[on]jour`
30     Str sous = substr(mot, 1, 2);
31     // Affichage d'une sous chaine de `on` ... ?
32     putstr(substr(sous, 2, 4));
33     return 0;
34 }
```

Exercice 1. Pour chaque instruction de la fonction **`main`**, **identifier** les éventuels effets secondaires. **Justifier** également si les fonctions **`putstr`** et **`substr`** sont à effets secondaires ou non.

Exercice 2. **Représenter** par un schéma l'état de la mémoire au moment où l'exécution atteint la première instruction de la fonction **`putstr`** (ligne 11) appelée depuis le **`main`** (ligne 32).

Note : en cas d'incertitude sur l'emplacement mémoire d'une donnée, ***indiquer clairement les hypothèses*** *justifiant le schéma (toute réponse cohérente avec ces hypothèses sera valorisée).*

Exercice 3. **Donner en justifiant** (avec ou sans l'aide de votre schéma) ce que produit le programme dans son flux standard de sortie.

Exercice 4. Supposons maintenant que la variable `sous` est construite ainsi : `Str sous = substr(mot, 3, 4);`. **Indiquer et justifier** l'erreur potentielle. **Proposer** un mécanisme pour se prémunir d'une telle erreur en **justifiant** ce choix. **Corriger** le code de la fonction responsable.

Bases *(10 points)*

Exercice 5. **Écrire** une fonction `tasser` paramétrée par une chaine de caractères et qui la modifie en supprimant les répétitions de caractères. Par exemple, `"aaa11aAA;;"` est changé en `"a1aA;"`.

Exercice 6. **Écrire et justifier** le prototype d'une fonction `tableau_ajouter_fin` paramétrée par deux tableaux d'entiers et qui ajoute tous les éléments du second tableau à la fin du premier, en modifiant la taille de ce dernier (on supposera que le premier tableau a été alloué dynamiquement). La fonction devra gérer les éventuelles erreurs. **Donner** un exemple d'utilisation de cette fonction.

Exercice 7. **Écrire** une fonction `fichier_vers_chaine` à gestion d'erreurs paramétrée par un nom de fichier et qui renvoie la chaine de caractères contenant l'intégralité de ce fichier. La fonction doit permettre à la fonction appelante de distinguer entre les erreurs rencontrées. *Note : On pourra lire le fichier plusieurs fois si nécessaire.*

Modularisation *(6 points)*

Exercice 8. Soit un projet constitué de quatre modules `A`, `B`, `C`, `D` et d'un module principal `Main`. Il figure dans ce projet les inclusions suivantes :

- `Main.c` inclut `B.h` et `C.c`
- `A.h` inclut `B.h` et `C.h`
- `A.c` inclut `D.h`
- `B.c` inclut `C.h`
- `D.h` inclut `A.h` et `B.h`

Tracer le graphe d'inclusions (étendues) du projet.

Exercice 9. **Expliquer** si le projet contient une ou plusieurs incohérences/erreurs

Exercice 10. Voici un brouillon incomplet et surtout erroné de `Makefile` pour ce projet.

```
1 Prog: Main.o A.o B.o C.o D.o
2     gcc -Wall -o Prog Main.o
3
4 Main.o: Main.c B.h C.h
5     gcc -o Main.o Main.c
6
7 A.o: A.c B.h
8     gcc -o A.o A.c
9
10 B.o: B.c C.h
11     gcc -o B.o B.c
12 ...
```

En supposant que le projet ne présente aucune incohérence, **donner** les erreurs présentes dans ce `Makefile`. (On ne s'interessera pas aux règles de production de `C.o` et `D.o`)

Exercice 11. Après une compilation (qu'on supposera sans erreur), on modifie le fichier `B.h`. Sachant qu'on a conservé les fichiers objets, quels sont les fichiers à produire à nouveau à l'aide de `gcc/clang` ? **Justifier**.

Techniques avancées *(8,5 points)*

Exercice 12. On souhaite dans un programme représenter un membre de l'université. Tous les membres ont un identifiant d'au plus 20 caractères (comme `henri.derycke`). On distingue les étudiants qui ont un numéro d'étudiant à 6 chiffres et les membres du personnel qui ont un matricule de 10 caractères (on ne peut pas être les deux à la fois). **Définir** un type permettant de représenter les données d'un membre en minimisant l'utilisation de la mémoire.

Exercice 13. **Écrire** une fonction `rotation` qui effectue une rotation circulaire à droite sur un mot de 64 bits. La fonction prend en paramètre un mot de **64 bits** et un entier `n` compris entre 0 et 63, et renvoie le mot obtenu en décalant tous les bits vers la droite de `n` positions, en réinjectant les bits de poids faible dans les bits de poids fort.

Par exemple sur **8 bits**, la rotation circulaire de 3 bits à droite du mot `01110010` est le mot `01001110` : `01110010 -> 01001110`

Exercice 14. **Écrire** une fonction `appliquer` paramétrée par un tableau d'entiers de type `int` et un pointeur sur une fonction `fun` à type de retour `int` et à un paramètre de type `int`. La fonction applique la fonction `fun` à chaque élément du tableau en modifiant son contenu.

Exercice 15. **Écrire** un court programme utilisant la fonction `appliquer` précédente pour doubler la valeur de chaque élement d'un tableau.

Bonus – à traiter uniquement après les autres exercices *(4 points)*

Exercice 16. **Écrire** une fonction `inverser` permettant d'inverser (au sens binaire : 0 devient 1, 1 devient 0) une sous-partie des bits d'un mot de **64 bits**, à partir d'une position (0 désignant le bit de poids faible) et d'une taille. **Équiper** la fonction de pré-assertions.

Par exemple, le résultat de la fonction `inverser` sur le mot `0...01100101`, la position `2` et la taille `4` est le mot `0...01011001`.

Exercice 17. **Définir** un type permettant de représenter la donnée d'un dictionnaire générique, c'est-à-dire un ensemble de paires `(clé,valeur)` où les clés sont des chaines de caractères et les valeurs sont de type générique. On ne s'intéressera pas aux considérations algorithmiques. **Donner** le prototype d'une fonction permettant récupérer à partir d'une valeur une des clés qui lui est associée.