Travaux dirigés
Programmation en C
—IMAC Première Année—

Cette feuille d'exercices vous permet de revoir les notions vues en travaux dirigés cette année.

▸ Exercice 1. *Compilation*

1. *Écrire la commande pour la compilation d'un fichier* **toto.c** *en s'assurant qu'il respecte la norme C Ansi avec le niveau de warning maximal. Le programme generé s'appellera* **titi**.
2. *Quel est le nom par défaut donné aux programmes par gcc ?*
3. *Quelle est la commande à exécuter pour lancer le programme* **titi** *?*

▸ Exercice 2. *Fonctions*

1. *Écrire une fonction* **float min(float m1, float m2)** *qui renvoie le minimum des réels passés en paramètres.*
2. *Écrire un* **main** *qui lit trois flottants sur l'entrée standard et affiche le minimum des trois.*

▸ Exercice 3. *Échanges*

1. *Écrire une fonction* **swap** *qui échange les valeurs de deux variables entières passées en paramètres.*
2. *Lorsque l'on veut afficher une variable, on utilise la fonction* **printf** *et on lui passe entre autres la variable à afficher. Lorsque l'on veut lire la valeur d'une variable sur l'entrée standard, on utilise la fonction* **scanf** *mais on passe cette fois l'adresse de la variable. Pourquoi ?*

▸ Exercice 4. *Pointeurs*

1. *Écrire un programme déclarant une variable réelle* **v** *et l'initialisant à 0. Déclarer un pointeur de type correspondant à cette variable et le faire pointer sur* **v**. *Modifier la valeur de* **v** *indirectement en utilisant le pointeur.*

2. *Écrire et appeler la fonction* **somdif** *qui calcule la somme et la différence de deux entiers passés en paramètres. Comme la fonction doit renvoyer deux valeurs (la somme et la différence) il n'est pas possible d'utiliser le mécanisme de valeur de retour (*__return__*), on utilisera alors deux paramètres supplémentaires de type pointeur.*

3. *Écrire un programme déclarant une variable réelle* **v** *et l'initialisant à 0. Déclarer un pointeur* **pv** *de type correspondant à cette variable et le faire pointer sur* **v**. *Déclarer un autre pointeur* **ppv** *de type approprié pour pouvoir pointer sur le pointeur* **pv**. *Faire pointer* **ppv** *sur* **pv**. *Modifier la valeur de* **v** *indirectement en utilisant le pointeur* **ppv**. *Faire un schéma de la mémoire pour bien comprendre.*

▸ **Exercice 5.** ***Réels - Allocation***

```
void *calloc(size_t nmemb, size_t size);
void *malloc(size_t size);
void free(void *ptr);
void *realloc(void *ptr, size_t size);
```

1. *Écrire une fonction* **double ecart_type(double *tableau, int taille)** *qui renvoie l'écart type*[(1)] *d'un ensemble de valeurs contenues dans un tableau passé en paramètre (on suppose que l'on dispose de la fonction* **double sqrt(double x)** *qui renvoie la racine carré de x).*

2. *Écrire une fonction* **main** *qui affiche l'écart type d'un ensemble de valeurs saisies au clavier par l'utilisateur. La saisie s'arrête quand l'utilisateur entre la valeur -1.*

▸ **Exercice 6.** ***Chaînes de caractères - Allocation***

1. *Écrire la fonction* **toupper** *qui prend en argument une chaine de caractères et transforme toutes les minuscules de cette chaîne en majuscules. La fonction ne renvoie rien.*

2. *Ré-écrire la fonction* **toupper** *sans utiliser la notation* [] *ni d'appel à d'autres fonctions telles que* **strlen**.

3. *Écrire une fonction* **main** *qui lit une chaine de caractères sur l'entrée standard. La chaîne est stockée dans un tableau statique de 256 caractères.*

4. *Écrire un* **main** *qui lit une chaine de caractères sur l'entrée standard. La chaîne est stockée dans un tableau dynamique de 256 caractères.*

5. *Citer les avantages et inconvenients des deux méthodes précédentes.*

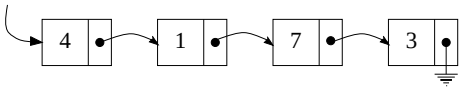
6. *Écrire la fonction* **toupper2** *qui prend en argument une chaîne de caractères et renvoie une chaîne qui est une copie de la chaîne passée en paramètre mais avec toutes les minuscules transformées en majuscules.*

(1) Rappel : $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$ où $\bar{x}$ est la moyenne de l'echantillon.

▸ **Exercice 7.** ***Structures***

1. *Définir un type* **Etudiant** *qui est une structure contenant un pointeur sur caractère (une chaîne de caractères...) et un entier.*
2. *Écrire une fonction* **Etudiant *nouvel_etudiant(char *nom, int age)** *qui crée un nouvel* **Etudiant** *de nom* **nom** *et d'age* **age**.
3. *Écrire un* **main** *qui demande à l'utilisateur combien d'étudiants il veut enregistrer. Crée un tableau de taille suffisante pour stocker ce nombre d'étudiants. L'utilisateur entre ensuite le nom et l'age de chaque étudiant et le programme affiche le nom du plus jeune d'entre eux.*

▸ **Exercice 8.** ***Listes simplement chaînées d'entiers***



L'objectif est de manipuler des listes chainées d'entiers. On considère les types **Element** *et* **Liste** *suivant :*

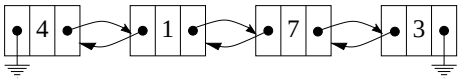
```
typedef struct element{
  int val;
  struct element *suivant;
}Element;

typedef Element * Liste;
```

Le champ suivant contient l'adresse du prochain **Element**. *On utilise l'adresse NULL pour signifier la fin de la liste.*

1. *Répondre aux questions suivantes :*
   - *a quoi correspond le type* **Liste** *?*
   - *imaginons une fonction qui manipule une liste, dans quel cas va-t'on passer une liste en paramètre et dans quel cas va-t'on passer un pointeur sur liste ?*
2. *Écrire une fonction* **void ajouter(Liste *l, int v)** *qui ajoute un élément de contenant* **v** *au début de la liste* **l**.
3. *Écrire une fonction* **int longueur(Liste l)** *qui compte le nombre d'éléments contenus dans la liste.*
4. *Écrire une fonction* **void afficher(Liste l)** *qui affiche les valeurs entières contenues dans la liste.*
5. *Écrire une fonction* **void détruire(Liste *l)** *qui libère la liste* **l**.
6. *En supposant que l'on ne dispose pas du type* **Liste**, *ré-écrire les prototypes des fonctions précédentes.*

▸ **Exercice 9.** ***Listes doublement de chaînées de chaînes de caractères***



1. *Définir les types* **Cellule** *et* **Liste** *relatifs à une liste doublement chainée de chaînes de caractères.*
2. *Écrire la fonction* **Cellule *nouvelle_cellule(char *str)**. *On dispose pour cela des fonctions :*
   - **char *strcpy(char *dest, char *src);**
   - **int strlen(char *str);**
3. *Sans utiliser la notation* [], *écrire la fonction* **int mon_strcmp(char *s1, char *s2)** *qui renvoie :*
   - *0 si les deux chaînes sont égales,*
   - *-1 si s1 est infèrieure à s2,*
   - *1 si s1 est supérieure à s2,*

   au sens lexicographique.
4. *Écrire la fonction* **void ajouter_trie(Liste *l, char *chaine)** *qui ajoute une cellule contenant* **chaine** *dans la liste* **l**. *On suppose que* **l** *est triée du plus petit au plus grand avant l'appel de la fonction et* **ajouter_trie** *conserve cette propriété !*
5. *Écrire la fonction* **void liberer_cellule(Cellule *c)**.
6. *Écrire la fonction* **void supprimer_cellule(Liste *l, Cellule *c)**.
7. *Écrire la fonction* **void liberer_liste(Liste *l)**.
8. *Écrire un* **main** *qui demande à l'utilisateur un entier* **n**. **n** *chaînes de caractères sont ensuite lues sur l'entrée standard et ajoutées dans une liste chaînée. La liste est alors affichée puis libérée.*