Rattrapages C

Code C

14 mai 2023

Version 1

Fabrice BOISSIER <fabrice.boissier@epita.fr>

Copyright

Ce document est destiné à une utilisation interne à EPITA.

Copyright © 2022/2023 Fabrice BOISSIER

La copie de ce document est soumise à conditions :

- ▷ Il est interdit de partager ce document avec d'autres personnes.
- ▷ Vérifiez que vous disposez de la dernière révision de ce document.

Table des matières

1 Consignes Générales

Les informations suivantes sont très importantes :

Le non-respect d'une des consignes suivantes entraînera des sanctions pouvant aller jusqu'à la multiplication de la note finale par 0.

Ces consignes sont claires, non-ambiguës, et ont un objectif précis. En outre, elles ne sont pas négociables.

N'hésitez pas à demander si vous ne comprenez pas une des règles.

Consigne Générale 0 : Vous devez lire le sujet.
Consigne Générale 1 : Vous devez respecter les consignes.
Consigne Générale 2 : Vous devez rendre le travail dans les délais prévus.

Consigne Générale 3 : Le travail doit être rendu dans le format décrit à la section Format de Rendu.

Consigne Générale 4 : Le travail rendu ne doit pas contenir de fichiers binaires, temporaires, ou d'erreurs (`*~`, `*.o`, `*.a`, `*.so`, `*#*`, `*core`, `*.log`, `*.exe`, binaires, ...).

Consigne Générale 5 : Dans l'ensemble de ce document, la casse (caractères majuscules et minuscules) est très importante. Vous devez strictement respecter les majuscules et minuscules imposées dans les messages et noms de fichiers du sujet.

Consigne Générale 6 : Dans l'ensemble de ce document, `login.x` correspond à votre login (donc `prenom.nom`).

Consigne Générale 7 : Dans l'ensemble de ce document, `nom1-nom2` correspond à la combinaison des deux noms de votre binôme (par exemple pour Fabrice BOISSIER et Mark ANGOUSTURES, cela donnera `boissier-angoustures`).

Consigne Générale 8 : Dans l'ensemble de ce document, le caractère ␣ correspond à une espace (s'il vous est demandé d'afficher ␣␣␣, vous devez afficher trois espaces consécutives).

Consigne Générale 9 : Tout retard, même d'une seconde, entraîne la note non négociable de 0.

Consigne Générale 10 : La triche (échange de code, copie de code ou de texte, ...) entraîne **au mieux** la note non négociable de 0.

Consigne Générale 11 : En cas de problème avec le projet, vous devez contacter le plus tôt possible les responsables du sujet aux adresses mail indiquées.

Conseil : N'attendez pas la dernière minute pour commencer à travailler sur le sujet.

Consigne Exceptionnelle : Tout code dans les headers (`.h`) sera sanctionné par un 0 (excepté les prototypes, constantes, et globales, si ceux-ci sont nécessaires).

2 Format de Rendu

Responsable(s) du projet :	**Fabrice BOISSIER** `<fabrice.boissier@epita.fr>`
Balise(s) du projet :	**`[RATT][C]`**
Nombre d'étudiant(s) par rendu :	1
Procédure de rendu :	Devoir sur Moodle
Nom du répertoire :	`login.x-RATT-C`
Nom de l'archive :	`login.x-RATT-C.tar.bz2`
Date maximale de rendu :	14/05/2023 23h42
Durée du projet :	2 semaines
Architecture/OS :	Linux - Ubuntu (x86_64)
Langage(s) :	C
Compilateur/Interpréteur :	**`/usr/bin/gcc`**
Options du compilateur/interpréteur :	**`-W -Wall -Werror -std=c99 -pedantic`**

Les fichiers suivants sont requis :

`AUTHORS`	contient le(s) nom(s) et prénom(s) de(s) auteur(s).
`README`	contient la description du projet et des exercices, ainsi que la façon d'utiliser le projet.

Votre code sera testé automatiquement, vous devez donc scrupuleusement respecter les spécifications pour pouvoir obtenir des points en validant les exercices.
Chaque fichier sera compilé indépendamment, vous pouvez donc mettre un **`main`** par fichier C.

L'arborescence attendue pour le projet est la suivante :

```
login.x-RATT-C/
login.x-RATT-C/AUTHORS
login.x-RATT-C/README
login.x-RATT-C/src/
login.x-RATT-C/src/is_even.c
login.x-RATT-C/src/my_calc.c
login.x-RATT-C/src/my_factures.c
login.x-RATT-C/src/my_pintree.c
login.x-RATT-C/src/my_transpose.c
```

3 Aide Mémoire

Le travail doit être rendu au format ***.tar.bz2***, c'est-à-dire une archive **bz2** compressée avec un outil adapté (voir **`man 1 tar`** et **`man 1 bz2`**).
Tout autre format d'archive (zip, rar, 7zip, gz, gzip, ...) ne sera pas pris en compte, et votre travail ne sera pas corrigé (entraînant la note de 0).

Pour générer une archive *tar* en y mettant les dossiers *folder1* et *folder2*, vous devez taper :

```
tar cvf MyTarball.tar folder1 folder2
```

Pour générer une archive *tar* et la compresser avec GZip, vous devez taper :

```
tar cvzf MyTarball.tar.gz folder1 folder2
```

Pour générer une archive *tar* et la compresser avec BZip2, vous devez taper :

```
tar cvjf MyTarball.tar.bz2 folder1 folder2
```

Pour lister le contenu d'une archive *tar*, vous devez taper :

```
tar tf MyTarball.tar.bz2
```

Pour extraire le contenu d'une archive *tar*, vous devez taper :

```
tar xvf MyTarball.tar.bz2
```

Pour générer des exécutables avec les symboles de debug, vous devez utiliser les flags **`-g -ggdb`** avec le compilateur. N'oubliez pas d'appliquer ces flags sur *l'ensemble* des fichiers sources transformés en fichiers objets, et d'éventuellement utiliser les bibliothèques compilées en mode debug.

```
gcc -g -ggdb -c file1.c file2.c
```

Pour produire des exécutables avec les symboles de debug, il est conseillé de fournir un script **`configure`** prenant en paramètre une option permettant d'ajouter ces flags aux **`CFLAGS`** habituels.

```
./configure
cat Makefile.rules
CFLAGS=-W -Wall -Werror -std=c99 -pedantic
./configure debug
cat Makefile.rules
CFLAGS=-W -Wall -Werror -std=c99 -pedantic -g -ggdb
```

Dans ce sujet précis, vous ferez du code en C et des appels à des scripts shell qui afficheront les résultats dans le terminal (donc des flux de sortie qui pourront être redirigés vers un fichier texte).

4 Exercice 1 - is even

Nom du(es) fichier(s) :	**is_even.c**
Répertoire :	**login.x-RATT-C/src/is_even.c**
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est d'afficher si un nombre est pair ou impair.

Vous devez écrire une fonction qui prendra un paramètre, et affichera selon ce paramètre s'il est pair ou impair.
Si le nombre est pair, vous devez écrire le mot **even** dans le terminal et renvoyer 0.
Si le nombre est impair, vous devez écrire le mot **odd** dans le terminal et renvoyer 1.

```
int is_even(int number)
```

Prototype de la fonction

```
$ ./is_even 3
odd
$ echo $?
1
$ ./is_even 6
even
$ echo $?
0
$ ./is_even 0
even
$ echo $?
0
```

Cas général

5 Exercice 2 - Calculatrice

Nom du(es) fichier(s) :	**my_calc.c**
Répertoire :	**login.x-RATT-C/src/my_calc.c**
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est de créer une mini calculatrice en C.

Vous devez écrire une fonction qui prendra trois paramètres (deux nombres, puis l'opérateur), et affichera le résultat de l'opération désignée.
Vous devez implémenter les 5 opérations suivantes : l'addition (symbole **+**), la soustraction (symbole **-**), la multiplication (lettre **x**), la division entière (symbole **/**), et le reste de la division euclidienne (symbole **%**).
À la fin du calcul, votre programme doit renvoyer 0.

```
int my_calc(int num1, int num2, char op)
```

Prototype de la fonction

```
$ ./my_calc 1 3 +
4
$ echo $?
0
$ ./my_calc 144 362 -
-218
$ echo $?
0
$ ./my_calc 6 7 x
42
$ echo $?
0
$ ./my_calc 13 3 /
4
$ echo $?
0
$ ./my_calc 69 2 %
1
$ echo $?
0
```

Cas général

Si le troisième paramètre n'est pas un des 5 caractères reconnus (**+**, **-**, **x**, **/**, **%**), vous devez indiquer le message suivant, et renvoyer 3.

Wrong␣parameters.
Usage:␣./my_calc␣number␣number␣operator
Operator␣might␣be␣:␣+␣-␣x␣/␣%

```
$ ./my_calc 1 2 A
Wrong parameters.
Usage: ./my_calc number number operator
Operator might be : + - x / %
$ echo $?
3
$ ./my_calc 1 2 3
Wrong parameters.
Usage: ./my_calc number number operator
Operator might be : + - x / %
$ echo $?
3
```

Cas d'erreur 1 : mauvais paramètres

Si le deuxième paramètre donné à la division est 0, vous devez renvoyer 4 et écrire le message d'erreur suivant.

Division␣by␣0␣is␣forbidden.

```
$ ./my_calc 1 0 /
Division by 0 is forbidden.
$ echo $?
4
```

Cas d'erreur 2 : division par 0

Si plusieurs des problèmes précédents sont rencontrés simultanément, vous devez les gérer dans cet ordre de priorité : le mauvais caractère en opérateur est prioritaire sur la division par 0.

```
$ ./my_calc 1 0 A
Wrong parameters.
Usage: ./my_calc number number operator
Operator might be : + - x / %
$ echo $?
3
$ ./my_calc 1 0 /
Division by 0 is forbidden.
$ echo $?
4
```

Cas d'erreurs : ordre des erreurs

ATTENTION

Il est formellement interdit d'utiliser le programme **bc** ou tout autre programme ou bibliothèque de calcul.

6 Exercice 3 - Transposition de matrice

Nom du(es) fichier(s) :	**my_transpose.c**
Répertoire :	**login.x-RATT-C/src/my_transpose.c**
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est de transposer une matrice.

Vous devez écrire une fonction nommée **my_transpose** qui prendra en paramètre un fichier contenant une matrice, et affichera dans le terminal la version transposée.

```
int my_transpose(char *filename)
```

Prototype de la fonction

```
$ cat file1.txt
123
456
789
$ cat file2.txt
0123
4567
$ cat file3.txt
01
23
45
67
$ cat empty.txt
$ cat file0.txt
0
```

Exemples de fichiers d'entrée

```
$ ./my_transpose file1.txt
147
258
369
$ ./my_transpose file2.txt
04
15
26
37
$ ./my_transpose file3.txt
0246
1357
$ ./my_transpose file0.txt
0
```

Cas général

Deux cas d'erreur doivent être gérés avant tout affichage : si la matrice est vide (ou n'existe pas), et si la matrice contient autre chose que des nombres.
Si la matrice est vide ou que le fichier n'existe pas, il faut indiquer le message suivant et retourner 1.

Empty␣matrix

```
$ ./my_transpose empty.txt
Empty matrix
$ echo $?
1
$ rm -f non-existent
$ ./my_transpose non-existent
Empty matrix
$ echo $?
1
```

Cas d'erreur 1

Si la matrice contient des caractères autres que des nombres, il faut indiquer le message suivant et renvoyer 2.

Incorrect␣matrix

```
$ cat file_error.txt
123
abc
789
$ ./my_transpose file_error.txt
Incorrect matrix
$ echo $?
2
```

Cas d'erreur 2

7 Exercice 4 - Sapin

Nom du(es) fichier(s) :	**my_pintree.c**
Répertoire :	**login.x-RATT-C/src/my_pintree.c**
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est d'afficher un sapin en ASCII art, dont la taille varie selon le paramètre donné.

Vous devez écrire une fonction qui prendra un paramètre, et affichera selon ce paramètre un sapin. Dans le cas général, affichez le sapin, et renvoyez 0.

```
int my_pintree(int number)
```

Prototype de la fonction

```
$ ./my_pintree 4
    /\
   /  \
  /    \
 /      \
/        \
----------
    ||
$ echo $?
0
$ ./my_pintree 5
     /\
    /  \
   /    \
  /      \
 /        \
/          \
------------
     ||
$ echo $?
0
```

Cas général

De façon précise, voici les spécifications pour les cas 1, 4, et 5 :

```
 /\     1 espace / 0 espace  \
/  \    0 espace / 2 espaces \
____    4 _
 ||     1 espace 2 |
```

Cas général 1

```
    /\          4 espaces / 0 espace  \
   /  \         3 espaces / 2 espaces \
  /    \        2 espaces / 4 espaces \
 /      \       1 espace  / 6 espaces \
/        \      0 espace  / 8 espaces \
__________      10 _
    ||          4 espaces 2 |
```

Cas général 4

```
     /\         5 espaces / 0 espace   \
    /  \        4 espaces / 2 espaces  \
   /    \       3 espaces / 4 espaces  \
  /      \      2 espaces / 6 espaces  \
 /        \     1 espace  / 8 espaces  \
/          \    0 espace  / 10 espaces \
____________    12 _
     ||         5 espaces 2 |
```

Cas général 5

Dans le cas où le paramètre 0 est donné, vous retournerez 0 et afficherez :

```
$ ./my_pintree 0
/\
||
$ echo $?
0
```

Cas 0

8 Exercice 5 - Factures

Nom du(es) fichier(s) :	**my_factures.c**
Répertoire :	**login.x-RATT-C/src/my_factures.c**
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est d'implémenter une facturation par listes chaînées.

Vous devez écrire une structure **facture** contenant une chaîne de caractères qui servira de nom, une chaîne de caractères qui servira de prénom, une chaîne de caractères qui servira d'adresse, un pointeur vers une liste chaînée, et un entier représentant le total.

Vous devez également écrire une structure **facture_liste** qui représentera chaque élément de la facture, donc qui contiendra un pointeur vers l'élément suivant, une chaîne de caractère représentant le nom de l'élément, un entier représentant la quantité achetée, et un entier représentant le prix unitaire. Les prix seront tous des entiers.

Enfin, vous devez implémenter quelques fonctions permettant de manipuler ces factures.

```
facture *create_facture(char *nom, char *prenom, char *adr);
int empty_facture(facture *facture);
void delete_facture(facture *facture);

void modify_name(facture *facture, char *nom);
void modify_surname(facture *facture, char *prenom);
void modify_adress(facture *facture, char *adresse);

void add_article(facture *facture, char *nom_art, int prix);
void remove_article(facture *facture, char *nom_art);

void print_facture(facture *facture);
```

Prototype des fonctions

facture *create_facture(char *nom, char *prenom, char *adr)

Cette fonction crée une nouvelle facture vide en lui assignant un nom, un prénom, et une adresse. La facture étant vide, il faut mettre le total à 0. La fonction renvoie un pointeur vers la structure allouée en mémoire. S'il y a un problème de mémoire, il faut renvoyer un pointeur **NULL**. Si le nom, le prénom, ou l'adresse est **NULL**, il faut également renvoyer un pointeur **NULL**.

`int empty_facture(facture *facture)`

Cette fonction supprime les éléments contenus dans la facture (mais pas la facture elle-même). Elle libère donc la mémoire allouée par les éléments contenus, puis remet le total à 0. La fonction doit renvoyer le nombre d'éléments supprimés. Si la factue existe mais ne contient aucun élément, alors il faut renvoyer 0. Si le paramètre donné est **`NULL`**, alors il ne faut rien faire et retourner -1.

`void delete_facture(facture *facture)`

Cette fonction supprime la facture, elle libère donc la mémoire allouée par la structure de la facture, mais également de chacun des éléments contenus. Si le paramètre donné est **`NULL`**, alors il ne faut rien faire.

`void modify_name(facture *facture, char *nom)`

Cette fonction modifie le nom associé à la facture. Attention aux problèmes de mémoire : il faut copier les caractères contenus dans le nom donné en paramètre, et éventuellement libérer puis allouer de nouveau un espace mémoie pour le nom dans la facture. Si l'un ou l'autre des paramètres donnés est **`NULL`**, alors il ne faut rien faire.

`void modify_surname(facture *facture, char *prenom)`

Cette fonction modifie le prénom associé à la facture. Attention aux problèmes de mémoire : il faut copier les caractères contenus dans le prénom donné en paramètre, et éventuellement libérer puis allouer de nouveau un espace mémoie pour le prénom dans la facture. Si l'un ou l'autre des paramètres donnés est **`NULL`**, alors il ne faut rien faire.

`void modify_adress(facture *facture, char *adresse)`

Cette fonction modifie l'adresse associée à la facture. Attention aux problèmes de mémoire : il faut copier les caractères contenus dans l'adresse donnée en paramètre, et éventuellement libérer puis allouer de nouveau un espace mémoie pour l'adresse dans la facture. Si l'un ou l'autre des paramètres donnés est **`NULL`**, alors il ne faut rien faire.

`void add_article(facture *facture, char *nom_art, int prix)`

Cette fonction ajoute un article à la facture donnée en paramètre, puis recalcule le nouveau total. Si la facture ou le nom de l'article sont **`NULL`**, alors il ne faut rien faire.
L'ajout de l'article implique d'ajouter un **`facture_liste`** en plus dans la facture. Si un élément **`facture_liste`** contient déjà le même nom d'article et le même prix, alors il suffit d'augmenter sa quantité. Si aucun élément dans la liste n'a le même nom et le même prix, alors il faut ajouter un nouvel élément dans cette liste et lui mettre sa quantité à 1.

void remove_article(facture *facture, char *nom_art)

Cette fonction supprime un article à la facture donnée en paramètre, puis recalcule le nouveau total. Si la facture ou le nom de l'article sont **NULL**, alors il ne faut rien faire. La suppression d'un article implique soit de réduire la quantité de 1 à un élément existant, soit de le supprimer de la liste s'il est à 1. Comme les prix pour un élément peuvent varier, vous supprimerez n'importe quel article ayant le même nom de la liste.

void print_facture(facture *facture)

Cette fonction écrit sur la sortie standard la facture en imprimant le nom, prénom, et adresse de la personne, puis chacun des articles avec sa quantité et son prix, puis, le total de la facture.

Le format attendu est le suivant :

```
--Facture--\n
Nom␣:␣[nom de la personne]\n
Prenom␣:␣[prénom de la personne]\n
Adresse␣:␣[adresse de la personne]\n
--\n
Articles\n
[quantité]␣*␣[nom de l'article]␣@␣[prix unitaire]\n
--\n
Total␣:␣[total de la facture]\n
----\n
```

Ce qui donnerait par exemple pour une facture concernant *Fabrice BOISSIER* à l'adresse *42 rue Voltaire*, ayant acheté 3 pommes à 1€ chacune et 4 bananes à 2€ chacune :

```
$ ./facture
--Facture--
Nom : BOISSIER
Prenom : Fabrice
Adresse : 42 rue Voltaire
--
Articles
3 * pommes @ 1
4 * bananes @ 2
--
Total : 7
----
```

Exemple