

Votre numéro d'anonymat :

--	--	--

Eléments de programmation 2 – 1I002

Examen du 11 mai 2015 (2 heures)

Seule la fiche mémo est autorisée

*Les calculatrices, baladeurs et autres appareils électroniques sont interdits. Les téléphones mobiles doivent être éteints et rangés dans les sacs. Le barème sur 65 points n'a qu'une valeur indicative. La dernière question de cet examen est une question bonus. Vous pouvez donc avoir une bonne note sans la traiter ou bien vous pouvez la traiter pour compenser d'éventuelles erreurs faites dans vos réponses aux autres questions. Avant de commencer, vérifiez que votre copie contient **25 pages** avec 20 questions.*

1 Compilation (5 pts)

Question 1 (3 points)

On considère le programme `compil.c` suivant

```
#include <stdio.h>

int main() {
    int a, b;

    a = 17;
    printf("%d\n", a);

    return 0;
}
```

On essaie de le compiler avec la commande

```
gcc -Wall -Werror -o compil compil.c
```

En lançant cette commande de compilation, obtient-on un programme exécutable ? Si oui, comment ce fichier exécutable s'appelle-t-il ? Si non, qu'est-ce qui empêche le compilateur de le produire ?

Réponse :

Question 2 (2 points)

Le programme suivant compile-t-il ? Si oui, qu'affiche-t-il ? Si non, expliquez pour quelle raison la compilation n'est pas possible.

```
#include <stdio.h>

#define entier      int
#define principale main
#define affiche     printf
#define renvoie     return

entier principale() {
    affiche("Vive_la_France!\n");
    renvoie 0;
}
```

Réponse :

2 Pointeurs, tableaux et structures - Questions de cours (9 pts)

Question 3 (4 points)

Lesquelles des manipulations de variables suivantes sont correctes ? Si elles ne le sont pas, comment peut-on les corriger ?

1. **float** a; **int** *pa = &a;
2. **int** b = 17; **int** *pb = b;
3. **int** c[1]; **int** *pc = c; *c = 42;
4. **int** d = 51; **float** e; **float** *pe = &e; *pe = d;

Toute réponse « non, incorrect » sans proposition de correction sera considérée comme fausse.

Réponse :

Réponse (continué) :

Question 4 (2 points)

Cochez les bonnes réponses dans le tableau suivant :

Les pointeurs sont indispensables pour	vrai	faux
– modifier la valeur d'un paramètre effectif dans la fonction appelée		
– écrire des programmes récursifs		
– appeler des fonctions		
– développer des algorithmes de tri		

Question 5 (3 points)

Lesquels des programmes **A** et **B** suivants compilent ? Si oui, qu'affiche le programme ? Si non, pourquoi la compilation du programme n'est-elle pas possible ?

Programme A :

```
#include <stdio.h>

int *f() {
    int a[5] = {1, 2, 3, 4, 5};
    return a;
}

void g(int *a) {
    int i;
    for (i=0; i<5; i++) {
        printf("%d\n", a[i]);
    }
}

int main() {
    int *b;
    b = f();
    g(b);
    return 0;
}
```

Programme B :

```
#include <stdio.h>

struct s {
    int t[5];
};

struct s f() {
    struct s a = { { 1, 2, 3, 4, 5} };
    return a;
}

void g(int *a) {
    int i;
    for (i=0;i<5;i++) {
        printf("%d\n", a[i]);
    }
}

int main() {
    struct s b;
    b = f();
    g(b.t);
    return 0;
}
```

Réponse :

3 Récursivité (4 pts)

Question 6 (4 points)

Pour n un entier positif ou nul, on peut calculer la puissance n -ième d'un réel x avec la formule suivante :

$$x^n = \begin{cases} (x^2)^{n/2} & \text{si } n \text{ est pair} \\ x \cdot (x^2)^{(n-1)/2} & \text{sinon} \end{cases}$$

Évidemment, $x^0 = 1$ quel que soit x .

Écrivez une fonction puissance **récursive** qui calcule et renvoie la puissance n -ième d'un réel x , pour n un entier, supposé positif ou nul.

Réponse :

4 Boucles, chaînes de caractères (10 pts)

Question 7 (4 points)

Écrivez une fonction `motdepasse` qui génère une chaîne de n caractères aléatoire, composée de lettres majuscules entre A et Z.

Votre fonction prendra un tableau de caractères et l'entier n en paramètres. Elle modifiera le contenu du tableau de caractères, en mettant dans les n premières cases des lettres aléatoires, suivies du caractère spécial `'\0'` dans la case suivante. Vous pouvez supposer que l'appel à `srand` (pour initialiser le générateur de nombre aléatoires) a déjà été fait en début de la fonction `main` et que le tableau passé en paramètre a une taille suffisante pour stocker une chaîne de caractères contenant n lettres.

Réponse :

Question 8 (2 points)

Écrivez un programme complet (avec inclusion des bibliothèques, `main` etc.) qui déclare un tableau de caractères de la bonne taille, initialise le générateur de nombres aléatoires, appelle ensuite la fonction `motdepasse` de la question qui précède et qui affiche cette chaîne de caractères aléatoire. La longueur de la chaîne de caractères à générer sera mise en **#define**.

Si jamais vous n'avez pas su donner de réponse à la question précédente, vous pouvez supposer la fonction `motdepasse` définie et fonctionnelle.

Réponse :

Question 9 (4 points)

Écrivez une fonction `comparechaines` **impérative** qui prend en paramètre deux tableaux de caractères correspondant à des chaînes de caractères, les compare et qui renvoie une valeur codant un booléen indiquant si elles sont égales ou non.

Deux chaînes de caractères sont considérées égales si et seulement si elles sont de même longueur et qu'à chaque indice de lettre, elles contiennent le même caractère. La chaîne vide est considérée égale à une autre chaîne vide.

Votre fonction ne devra pas prendre de paramètre(s) indiquant la taille des tableaux de caractères passés en argument. Elle devra être efficace ; en particulier, elle ne devra pas faire de comparaisons qui ne servent à rien. Vous ne devez pas utiliser de fonction de la bibliothèque `string.h`.

Réponse :

5 Fonctions et pointeurs (14 pts)

Question 10 (2 points)

La fonction `time` fournie par la bibliothèque `time.h` renvoie le nombre de secondes depuis 1er janvier 1970, minuit, heure de Londres.

Écrivez une fonction `secondesaujourd'hui` qui ne prend aucun paramètre et qui, en se servant de la fonction `time`, renvoie le nombre de secondes qui se sont écoulées depuis minuit (heure de Londres) à l'instant du jour où la fonction est exécutée.

Vous remarquerez qu'indépendamment des années bissextiles, depuis le 1er janvier 1970, il y a toujours un nombre entier de jours qui se sont écoulés jusqu'à minuit de chaque jour. En plus, chaque jour dure évidemment $24 \times 60 \times 60$ secondes.

Réponse :

Question 11 (4 points)

Écrivez une fonction `heuresminutessecondes` qui étant donné le nombre de secondes écoulées depuis minuit pour un instant du jour, retourne les heures, minutes et secondes correspondant à cet instant.

Par exemple, étant donné 17, correspondant à l'instant 17 secondes après minuit, la fonction retournera 0 heures, 0 minutes et 17 secondes. Pour $86399 = 24 \times 60 \times 60 - 1$, elle retournera 23 heures, 59 minutes et 59 secondes.

Réponse :

Réponse (continué) :

Question 12 (2 points)

Écrivez une fonction `afficheTemps` qui prend en paramètre le nombre de secondes écoulées depuis minuit pour un instant du jour, se sert de la fonction `heuresminutessecondes` programmée à la question qui précède et qui affiche le temps correspondant à cet instant sous la forme `HHh MMm SSs`. Par exemple, votre fonction affichera `5h 17m 42s` ou `17h 9m 0s`.

Toute réponse qui ne fait pas intervenir d'appel à la fonction `heuresminutessecondes` sera considérée comme fausse.

Si vous n'avez donné aucune réponse à la question précédente, donnez un prototype pour la fonction `heuresminutessecondes` et supposez que la fonction est correctement définie.

Réponse :

Question 13 (1 point)

Écrivez une fonction `affichehorloge` qui, en se servant des fonctions `secondesaujourd'hui` et `affichetemps`, programmées ci-dessus, affiche le temps courant.

Toute réponse qui ne fait pas intervenir d'appels aux deux fonctions `secondesaujourd'hui` et `affichetemps` sera considérée comme fausse.

Si vous n'avez pas donné de réponse aux questions correspondantes, donnez un prototype pour chacune des deux fonctions `secondesaujourd'hui` et `affichetemps` et supposez qu'elles sont correctement définies.

Réponse :

Question 14 (5 points)

On considère le programme suivant :

```
#include <stdio.h>

int a;
float b;
char c;

void h() {
    printf("%c\n", c);
}

void f(int x) {
    a = a + x;
    printf("%f\n", b);
}

float g(char d) {
    int t;
    c = d;
```

No. anonymat :

```
printf("%d\n", a);
t = c - 'A';
f(t);
return 1664.0 * b;
}

int main() {
    char c;
    float e;
    a = 17;
    b = 42.0;
    c = 'X';
    f(42);
    e = g('B');
    h();
    printf("%c->_%f->_%d\n", c, e, a);
    return 0;
}
```

Donnez un programme qui n'utilise pas de variables globales et qui a le même comportement que le programme ci-dessus.

Réponse :

Réponse (continué) :

6 Problème : gestion de trains (23 pts)

Le but de cet exercice est de modéliser la gestion de trains au niveau « passagers », telle que la fait la SNCF quand vous achetez un billet de train.

Un train sera décrit par un ensemble de données, rassemblées dans une structure définie ci-dessus. Ces données comprennent :

- Le numéro du train, indiqué par un entier
- Le type de train, donné sous la forme d'un entier : TGV = 1, Corail = 2, TER = 3, train de nuit = 4, ICE = 5, Eurostar = 6, train spécial = 7. Ces constantes entières seront définies avec des **#define**.
- La gare de départ, indiquée par une chaîne de caractères.
- La gare d'arrivée, indiquée par une chaîne de caractères.
- L'heure de départ, donné comme un entier, indiquant les secondes depuis minuit.
- L'heure d'arrivée, donné comme un entier, indiquant les secondes depuis minuit.
- Le prix du trajet en Euro avec décimales.

Les gares de départ et d'arrivée des trains internationaux porteront, dans les champs correspondants de la structure, le nom de la ville en français. Pour permettre un affichage en gare et une impression de billets internationaux, on définira une deuxième structure, contenant deux champs : le nom d'une gare en français et le nom correspondant dans la langue du pays où se trouve la gare. Ainsi, on pourra faire correspondre Londres à London, Moscou à Moskva, Cologne à Köln etc.

Les trains circulant sur le réseau que nous considérons et les correspondances des villes seront reflétés dans deux tableaux définis globalement.

On partira donc du programme suivant :

```
#include <stdio.h>

#define TRAIN_TGV 1
#define TRAIN_CORAIL 2
#define TRAIN_TER 3
#define TRAIN_NUIT 4
#define TRAIN_ICE 5
#define TRAIN_EUROSTAR 6
#define TRAIN_SPECIAL 7

#define NOMBRE_TRAINS 8
#define NOMBRE_GARES_INTERNATIONALES 5

struct train {
    int    numero;
    int    type;
    char    gare_depart[64];
    char    gare_arrivee[64];
    int    heure_depart;
    int    heure_arrivee;
    float    prix;
};
```

```
struct traduction {
    char nom_francais[64];
    char nom_international[64];
};

struct train trains[NOMBRE_TRAINS] = {
    { 6420, TRAIN_TGV, "PARIS", "LYON", 28800, 36120, 65.00 },
    { 5948, TRAIN_EUROSTAR, "LONDRES", "PARIS", 16920, 28080, 240.50 },
    { 1664, TRAIN_CORAIL, "LYON", "NICE", 38160, 53940, 89.75 },
    { 9973, TRAIN_SPECIAL, "NICE", "MOSCOU", 57600, 20640, 542.90 },
    { 1242, TRAIN_ICE, "COLOGNE", "NANCY", 29820, 34800, 95.50 },
    { 3662, TRAIN_TER, "NANCY", "PARIS", 36060, 51180, 72.00 },
    { 4201, TRAIN_CORAIL, "NANCY", "COURTRAI", 36540, 55380, 112.50 },
    { 1001, TRAIN_NUIT, "PARIS", "VIENNE", 83220, 25440, 280.00 }
};

struct traduction gares_internationales[NOMBRE_GARES_INTERNATIONALES] = {
    { "LONDRES", "LONDON" },
    { "MOSCOU", "MOSKVA" },
    { "COLOGNE", "KOELN" },
    { "COURTRAI", "KORTRIJK" },
    { "VIENNE", "WIEN" }
};
```

Question 15 (4 points)

Écrivez une fonction `testgareinternationale` qui prend en paramètre :

- une chaîne de caractères indiquant le nom d'une gare en français,
- un tableau de structures `struct traduction` ainsi que
- la taille de ce tableau de structures

et qui détermine si la gare figure dans la liste des gares internationales avec une traduction ou non.

Si la gare figure dans la liste des gares internationales, la fonction `testgareinternationale` renvoie l'indice de la correspondance « gare en français/ gare en langue étrangère » dans le tableau donné en paramètre. Si la gare n'y figure pas, la fonction renvoie `-1`.

Votre fonction `testgareinternationale` pourra se servir de la fonction `comparechaines` programmée à la question 9. Cette fonction prend deux chaînes des caractères en paramètre et renvoie une valeur codant un booléen indiquant si les deux chaînes sont égales ou non.

Votre fonction devra être efficace et ne devra pas faire de comparaisons qui ne servent à rien.

Réponse :

Réponse (continué) :

Question 16 (3 points)

Écrivez une fonction `affiche_typed_train` qui prend en paramètre un entier décrivant un type de train et qui affiche le type de train correspondant.

- Pour un TGV, la fonction affichera TGV.
- Pour un Corail, la fonction affichera CORAIL.
- Pour un TER, la fonction affichera TER.
- Pour un train de nuit, la fonction affichera CNL.
- Pour un ICE, la fonction affichera ICE.
- Pour un Eurostar, la fonction affichera EUROSTAR.
- Pour un train spécial, la fonction affichera SPECIAL.
- Pour tout autre train, la fonction affichera AUTRE.

Votre solution ne devra rien supposer sur la valeur des constantes symboliques définies par `#define`, `TRAIN_TGV` à `TRAIN_SPECIAL`.

Réponse :

Réponse (continué) :

Question 17 (4 points)

Programmez une fonction

```
void affichetrain(struct train t, int taille,
                 struct traduction gares_internationales[]);
```

qui prend en paramètre une structure décrivant un train `t`, un tableau de traductions de noms de gare ainsi que la taille de ce tableau et qui, en se servant des fonctions `affichetypetrain` (Question 16), `affichetemps` (Question 12), `testgareinternationale` (Question 15), affiche les informations disponibles sur ce train `t`.

L’affichage sera fait sur le modèle :

TGV 6420 PARIS 8h 0m 0s LYON 10h 2m 0s 65.00 EUR.

Pour un train partant ou arrivant à une gare qui ne se trouve pas en France, la fonction affichera d’abord le nom en français, suivi du nom dans la langue étrangère correspondante entre parenthèses. Aucune parenthèse ne devra être affichée pour les trains partant ou arrivant à une gare en France. *Vous devez traiter et le cas où la gare de départ se trouve hors de France et le cas où c’est la gare d’arrivée qui a un nom en langue étrangère.*

Ainsi obtiendra-t-on pour des train internationaux, l’affichage suivant :

ICE 1242 COLOGNE (KOELN) 8h 17m 0s NANCY 9h 40m 0s 95.50 EUR resp.
CORAIL 4201 NANCY 10h 9m 0s COURTRAI (KORTRIJK) 15h 23m 0s 112.50 EUR.

Même si vous n’avez pas donné de réponse à l’une des questions 16, 12 ou 15, vous pouvez supposer que toutes les fonctions `affichetypetrain`, `affichetemps` et `testgareinternationale` sont définies et fonctionnelles.

Réponse :

```
void affichetrain(struct train t, int taille,
                 struct traduction gares_internationales[]) {
```

Réponse (continué) :

Question 18 (2 points)

Écrivez une fonction `int differencetemps(int t1, int t2)` qui, pour deux instants t_1 et t_2 , donnés sous la forme de secondes écoulées depuis minuit, retourne la différence de temps en secondes. Cette différence de temps est toujours positive ou nulle (t_2 correspond à un instant postérieur à t_1 , les deux instants ont lieu le même jour ou deux jours consécutifs).

Vous remarquerez qu'il ne suffit pas de soustraire t_1 de t_2 : la différence de temps entre $t_1 = 86340$, correspondant à 23h59m00s, et $t_2 = 60$, correspondant à 00h01m00s, est de 120 secondes et non de $t_2 - t_1 = 60 - 86340 = -86280$ secondes. Cette dernière valeur est négative.

Réponse :

```
int differencetemps(int t1, int t2) {
```

Question 19 (5 points)

Écrivez une fonction

```
int cherchetrainindirect(char gare_depart[],
                        char gare_arrivee[],
                        int heure_depart,
                        struct train trains[],
                        int trains_ignores[],
                        int taille);
```

qui prend en paramètre :

- le nom d'une gare de départ,
- le nom d'une gare d'arrivée,
- une heure de départ,
- un tableau de structures décrivant des trains,
- un tableau de valeurs codant des booléens indiquant les trains à ignorer dans le tableau des trains et
- la taille des deux tableaux en paramètre.

La fonction renvoie l'indice du train partant pour la gare d'arrivée de la gare de départ le plus tôt après l'heure de départ passée en paramètre, parmi les trains qui ne sont pas marqués comme « à ignorer » dans le tableau de booléens. Si aucun tel train n'existe, la fonction renvoie -1 .

Si la i -ième case du tableau `trains_ignores` est 0, le train à la i -ième du tableau `trains` est à considérer, sinon il est à ignorer.

La gare de départ et la gare d'arrivée sont censées être données sous leur nom français.

Votre fonction pourra se servir des fonctions `comparechaines` (Question 9) et `differencetemps` (Question 18). Même si vous n'avez pas su programmer l'une de ces fonctions, vous pouvez les supposer toutes les deux définies et fonctionnelles.

Réponse :

```
int cherchetrainindirect(char gare_depart[], char gare_arrivee[],
                        int heure_depart, struct train trains[],
                        int trains_ignores[], int taille) {
```

Réponse (continué) :

Question 20 (5 points)

Cette question est une question bonus. Vous pouvez obtenir une bonne note même sans la traiter. Vous pouvez également la traiter afin de compenser d'éventuelles erreurs faites dans vos réponses aux questions qui précèdent.

On veut maintenant écrire une fonction **récursive** qui permette de trouver une correspondance, c-à-d une liste de trains avec des changements dans des gares intermédiaires. Évidemment, on voudra savoir à la fois si une telle correspondance existe et laquelle des correspondances est la plus courte en termes de temps de trajet complet.

Écrivez donc une fonction **récursive**

```
int cherchecorrespondance(int correspondance[],
                          int *nb_changements,
                          int *heure_arrivee_destination,
                          char gare_depart[],
                          char gare_arrivee[],
                          int heure_depart,
                          struct train trains[],
                          int trains_ignores[],
                          int taille);
```

qui prend en paramètre :

- un tableau `correspondance`, qu'elle va modifier pour y mettre les indices des trains pris pour la correspondance la plus rapide (en temps),
- un pointeur sur un entier `nb_changements`, où elle mettra le nombre de trains pris pour faire le trajet,
- un pointeur sur un entier `heure_arrivee_destination`, où elle mettra l'heure d'arrivée à destination pour la correspondance la plus rapide,
- une chaîne de caractères indiquant la gare de départ,
- une chaîne de caractères indiquant la gare d'arrivée,
- l'heure de départ désirée pour la correspondance,
- le tableau des trains du réseau,
- un tableau de valeurs codant des booléens indiquant si le train d'un certain indice doit être ignoré pour établir la correspondance ainsi que
- un entier `taille` pour la taille des tableaux `correspondance`, `trains` et `trains_ignores`.

La fonction rend une valeur codant un booléen égal à « vrai » si et seulement si une correspondance existe entre les gares de départ et d'arrivée.

Remarquons que la fonction que vous devez écrire accomplit plusieurs tâches en même temps : d'une part, elle détermine si une correspondance existe entre deux gares, soit en prenant un train direct, soit avec des changements à des gares intermédiaires. D'autre part, si une ou plusieurs de telles correspondances existent, elle en détermine celle qui fait arriver le voyageur à sa destination le plus rapidement possible. En plus, pour la correspondance qui minimise le temps de trajet, elle retourne la liste des trains à prendre, en modifiant le tableau `correspondance` et la valeur des variables pointées par `nb_changements` et `heure_arrivee_destination`.

La fonction que vous devez programmer pourra procéder de la façon suivante :

- La fonction teste d'abord si un train direct existe entre les gares de départ et d'arrivée. Pour cela, elle se sert de la fonction `cherchetraindirect`, programmée à la question 19. Si un tel train direct existe, elle met l'indice de ce train dans la case d'indice 0 du tableau `correspondance`,

met le nombre de changements à 0 et copie l'heure d'arrivée du train direct dans la variable pointée par `heure_arrivee_destination`.

- Si aucun train direct n'a été trouvé, la fonction considérera alors tous les trains pour déterminer
 - si le train n'est pas marqué comme « à ignorer »,
 - que le train part de la gare de départ et
 - qu'une correspondance existe pour la destination finale depuis la gare d'arrivée de ce train. Pour déterminer si cette correspondance existe à partir de cette première gare intermédiaire, la fonction fait un appel récursif. Pour cet appel récursif, la fonction marque le train qu'elle est en train de considérer comme « à ignorer »¹ et passe le sous-tableau de `correspondance` commençant à la case d'indice 1 à la fonction appelée. Après l'appel récursif, la fonction marque le train à nouveau comme « à prendre ».

Parmi tous les trains satisfaisant ces trois conditions, la fonction détermine celui qui résulte en un trajet de durée minimale.

- Finalement, si au moins une correspondance a été trouvée et que le premier train commençant la correspondance la plus rapide est ainsi connu, la fonction met l'indice de ce premier train dans la case d'indice 0 du tableau `correspondance` et répète l'appel récursif pour avoir le sous-tableau commençant à la case d'indice 1 correctement rempli. La valeur pointée par `nb_changements` doit ajustée en conséquence.

Assurez-vous que votre fonction `cherchecorrespondance` renvoie bien une valeur codant un booléen qui indique si une correspondance a été trouvée entre les gare de départ et d'arrivée ou non.

Réponse :

```
int cherchecorrespondance(int correspondance[],
                          int *nb_changements,
                          int *heure_arrivee_destination,
                          char gare_depart[], char gare_arrivee[],
                          int heure_depart, struct train trains[],
                          int trains_ignores[], int taille) {
```

1. Le tableau des trains « à ignorer » sert à éviter, pour une recherche de train « Paris–New-York », une correspondance infinie « Paris–Lyon–Paris–Lyon–... », prise toujours dans l'espoir de finir par trouver un train pour New-York qui part de Paris ou de Lyon.

Réponse (continué) :

Réponse (continué) :