ESIR-SYS1 TP 10-11-12 C: Émissions mondiales de CO2 (TP Non Noté, 0.5 point bonus possible)

François Taïani (`ftaiani.ouvaton.org`)*

1 Préliminaires

1.1 Contexte

Le CO_2 est un des principaux gaz à effet de serre (abrégés en GES en français, *Greenhouse Gases* en anglais, abrégés en GHG), à l'origine des importants changements que subit le climat de notre planète.

Ce TP se concentre sur les émissions humaines de CO_2, mais d'autres gaz à effet de serre d'origine humaines (le méthane en particulier, très présent dans l'agriculture) contribuent aussi fortement au changement climatique.

Pour en apprendre plus sur ces thèmes, vous êtes fortement encouragé(e)s à consulter les ressources qui suivent :

- Le site Bon Pote, un site de vulgarisation sur le changement climatique, s'appuyant régulièrement sur les rapport du GIEC (voir ci-dessous).
- Les rapports du GIEC (Groupe d'experts intergouvernemental sur l'évolution du climat, *Intergovernmental Panel on Climate Change* ou IPCC en anglais), l'organisme intergouvernemental établie par l'ONU pour d'évaluer l'ampleur, les causes et les conséquences du changement climatique en cours.

  Les rapports du GIEC synthétisent les connaissances scientifiques sur des questions liées au climat, et sont généralement volumineux (plusieurs centaines de pages). Des versions simplifiées (*Summary for Policymakers*) sont généralement disponibles qui résument de façon succincte (30-40 pages) les points clés. Les rapports suivants peuvent vous intéresser :

  - Changement climatique 2021, Les bases scientifiques physiques, Résumé à l'intention des décideurs, GIEC, 2021, Cambridge University Press.
  - Climate Change 2022: Impacts, Adaptation and Vulnerability, Summary for Policymakers, IPCC, Geneva, Switzerland, `doi:10.1017/9781009325844.001`.
  - Climate Change 2023, Synthesis Report, Summary for Policymakers, IPCC, Geneva, Switzerland, `doi:10.59327/IPCC/AR6-9789291691647.001`

1.2 But du TP

Le but de ce TP est de lire et d'analyser une fichier au format CSV (*comma separated values*) documentant les émissions mondiales de CO_2, un des principaux gaz à effet de serre. Une partie de l'analyse consistera à ordonner les pays selon différentes métriques. Le TP commence donc par la réalisation d'un tri à bulles, une procédure très simple de tri qui nous suffira pour cet exercice.

1.3 Préambule

Créez un répertoire `tp_10_11_12` dans le répertoire correspondant à vos TP en programmation C (`tp_C` par exemple).

*francois.taiani@irisa.fr

2 Travail demandé

2.1 Étape 0: Tri à bulles (`step0_bubble_sort.c`)

Un tri à bulles consiste à parcourir les éléments d'un tableau `T` en échangeant les pairs d'éléments contigus `T[i]` et `T[i+1]` qui ne sont pas dans le bon ordre, et à répéter ce parcours jusqu'à ce que plus aucun échange ne soit nécessaire (ce qui indique que le tableau est trié). (Vous pouvez consulter l'article Wikipedia[1] sur le tri à bulles si la procédure n'est pas claire.)

En supposant que votre programme contienne un tableau d'entiers non triés comme le tableau ci-dessous, implémentez un tri à bulles qui trie ce tableau en place puis imprime le résultat.

```
int array_to_be_sorted[] = {9,2,1,15,25,27,20,0,14,9,2,12,21,40,23,5,17,29,22,30};
```

Conseils et remarques :

- Vous aurez besoin d'une variable pour indiquer si un échange a été nécessaire lors d'un parcours du tableau.
- Vous pouvez utiliser une boucle de contrôle de type `do .. while` pour garantir qu'une première itération ait bien lieu.
- Les valeurs `true` et `false` et le type `bool` sont disponibles en incluant l'entête `stdbool.h`. Il est possible de s'en passer (en utilisant `0` et `1`), mais leur utilisation rend le code plus lisible.

2.2 Étape 1: Lecture d'un fichier `*.csv` (`step1_read_from_csv.c`)

Vous trouverez sur le site du cours le fichier `owid-co2-data-excerpt.csv`, un fichier `csv` (*Comma Separated Values*) extrait des données proposées sur le site `https://github.com/owid/co2-data`. Ce fichier comprend une première ligne d'entête (*header*) indiquant les titres de chaque colonne, suivi d'une liste d'entrées (*records*) pour chaque pays du globe. Chaque entrée contient des informations sur le pays concerné (son code iso, son nom, sa population), ainsi que sur ses émissions. Le fichiers `owid-co2-codebook-except.csv`, aussi fourni, documente la signification de chaque colonne, ainsi que la provenance des données.

L'objectif de cette étape est de lire ce fichier ligne à ligne, d'en extraire des informations pour chaque pays, et de produire une sortie ressemblant à l'impression suivante :

```
No data available on the consumption CO2 caused by Afghanistan in 2019.
Albania caused 5.851000 million tons of CO2 in 2019.
No data available on the consumption CO2 caused by Algeria in 2019.
No data available on the consumption CO2 caused by Andorra in 2019.
No data available on the consumption CO2 caused by Angola in 2019.
No data available on the consumption CO2 caused by Anguilla in 2019.
No data available on the consumption CO2 caused by Antigua and Barbuda in 2019.
Argentina caused 166.598999 million tons of CO2 in 2019.
Armenia caused 5.472000 million tons of CO2 in 2019.
No data available on the consumption CO2 caused by Aruba in 2019.
...
```

2.2.1 Jalon 1a

En vous référant au cours, commencez par écrire un programme qui ouvre le fichier en lecture, et lit puis imprime chaque ligne en utilisant `fgets`. Comme pour le TP précédent, pré–définissez une chaîne de caractères suffisamment grande pour lire chaque ligne (par exemple de 1024 caractères). Pensez bien à tester que `fgets` a correctement fonctionné.

Conseils et remarques :

- Pour vos tests et la mise au point, vous pouvez créer un fichier de données plus petit en utilisant `head -10` et une redirection vers un fichier de test (par exemple `test.csv`).

[1] `https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles`

- Commencez de façon incrémentale par ne lire qu'une ou deux lignes.
- Vous aurez ensuite besoin d'une boucle `while` pour tout lire. Vous pouvez directement tester le retour de `fgets` dans le test de la boucle `while`, ou utiliser une boucle infinie (`while (true)`) dont vous sortez avec une instruction `break` quand `fgets` indique la fin du fichier.

2.2.2 Jalon 1b

En utilisant la fonction `sscanf` extrayez maintenant de chaque ligne les informations concernant un pays dans des variables appropriées.

Pour le code ISO et le nom du pays, prédéfinissez deux chaînes suffisamment grandes (par exemple 10 caractères pour le code iso, et 50 pour le nom). Pensez à utiliser des `float` pour les nombres à virgule (*co2*, etc.), et des entiers pour les nombres sans virgule (*year*, *population*). Vous pouvez pour l'instant réutiliser les mêmes variables pour chaque pays, en écrasant leurs valeurs précédentes.

Pour les chaînes de caractères, le format `%s` ne va ici pas bien fonctionner (du fait des espaces dans les noms, et de la mauvaise prise en compte des virgules comme séparateur). Pour éviter tout ennui, utilisez la syntaxe `%[^,]` dans la chaîne de formatage qui indique une chaîne de caractères ne contenant aucune virgule. (Voir `man sscanf` pour plus de détails.)

Attention à réaliser un passage par référence (opérateur `&`) pour les variables contenant des nombres (les chaînes sont déjà des pointeurs).

La colonne *consumption_ co2* pose un problème : la donnée n'est pas connue pour de nombreux pays, et apparaît dans ce cas avec un `?` dans le fichier de données. Pour cette quantité–là, il faut donc d'abord

- lire la donnée dans une chaîne définie pour cela (30 caractères devraient largement suffire),
- puis tester si la chaîne lue est `"?"`. Dans ce cas considérez que *consumption_ co2* vaut par convention -1. Sinon, utiliser `sscanf` de nouveau pour extraire la valeur lue.

Conseils et remarques :

- Il vous faudra ignorer la première ligne, qui contient les titres des colonnes.
- Lors de chaque utilisation de `sscanf` vérifiez bien que toutes les variables attendues sont lues avec succès en vérifiant la valeur de retour de la fonction (voir `man sscanf`). Si un problème est détecté, affichez un message d'erreur, éventuellement avec la chaîne qui pose problème, et arrêtez le programme. (Vous pouvez utiliser `exit(0)` pour sortir du programme en indiquant qu'un problème est survenu. Vous aurez besoin de l'entête `stdlib.h` pour utiliser `exit`.)
- Vous ne pouvez pas tester l'égalité entre deux chaîne avec l'opérateur `==`, qui va comparer leurs adresses (pointeurs). Pour tester si la chaîne lue pour *consumption_ co2* est `"?"`, tester simplement si le premier caractère de cette chaîne est égal à `'?'`.
- L'utilisation de chaînes de longueur fixe permet de simplifier le TP, mais est en général à proscrire pour des raisons de sécurité et de robustesse. `sscanf` possède un mécanisme pour allouer lui–même des variables de chaîne de taille suffisante, que nous n'utiliserons pas pour simplifier l'exercice.
- Pensez à ajouter un petit affichage comme celui démontré ci-dessus pour vérifier que vous lisez bien des valeurs appropriées.

2.3 Étape 2: Utilisation d'un type composite (`step2_use_a_struct.c`)

Nous allons maintenant utiliser un type composite (`struct` en C) pour réunir les informations lues dans une seule variable.

2.3.1 Préliminaires

Un `struct` permet de définir un regroupement de différentes données. Par exemple le code suivant permet de définir le type composite `student` qui regroupe une chaîne de caractère de 20 octets, et un entier.

```
typedef struct {
  char  name[20] ;
  int   age ;
```

```
} student ;
```

Ce nouveau type `student` peut être utilisé pour définir des variables qui contiennent à la fois une chaîne et un entier. Par exemple le code suivant déclare la variable `dark`, de type `student`, et l'initialise avec la chaîne `"Vador"` (pour le champ `name`) et l'entier 128 (pour le champ `age`). En mémoire chaque variable de type `student` utilise 20 octets pour `name`, plus 4 octets pour `age`. Ces 24 octets sont alloués de façon contiguë.

```
student dark = {"Vador",128};
```

Les champs d'une variable composite sont accessibles en utilisant l'opérateur `.`, comme par exemple :

```
printf("%s is %d years old.\n", dark.name, dark.age) ;
```

Comme avec tout type on peut manipuler des pointeurs vers un type composite, comme `student * ptr_dark ;`. Dans ce cas, la syntaxe `ptr_dark->name` est un raccourci syntaxique pour `(*ptr_dark).name`.

```
student * ptr_dark = &dark ;
printf("%s and ", (*ptr_dark).name) ;
printf("%s "    ,   ptr_dark->name) ;
printf("are the same thing.\n");
```

2.3.2 Travail de l'étape

Modifiez votre programme de l'étape 1 pour :

- définir un type composite contenant toutes les informations relatives à un pays ;
- utiliser une variable de ce type pour stocker les informations extraites d'une ligne donnée.

Comme pour l'étape précédente, vérifiez avec des impressions que vous extrayez bien les données du fichier.

Conseils et remarques :

- Vous aurez besoin de passer certains champs de votre variable composite par référence. De même que vous pouvez prendre l'adresse d'une variable composite avec `&`, vous pouvez aussi prendre l'adresse de l'un de ses champs, par exemple `&dark.name`.

2.4 Étape 3: Tableau de variables composites (`step3_array_of_structs.c`)

Définissez maintenant un tableau de variables composites (le tableau des pays), et peuplez ce tableau avec les données issues du fichier. Vérifiez que les données sont lues correctement en imprimant des données contenues dans le tableau.

Conseils et remarques :

- Le fichier contient 216 lignes : une ligne d'entête, et 215 lignes d'enregistrements (*records*).
- Vous pouvez définir la taille du tableau à remplir (215 donc) dans une constante (voir le TP 9 pour la syntaxe), et utiliser une boucle `for` plutôt que `while` pour remplir le tableau.
- Pour vérifier que tout le fichier a été lu, vous pouvez vérifier après la boucle `for` qu'un appel à `fgets` renvoie bien `NULL`.

2.5 Étape 4: Tri du tableau (`step4_sorting_array_of_structs.c`)

Adaptez maintenant le code de l'étape 0 pour triez le tableau des pays en fonction du CO_2 émis annuellement par chaque pays basé sur leur consommation[2].

[2] En particulier les produits manufacturés sont pris en compte dans leur pays de consommation plutôt que de production par cette métrique.

Réalisez cette procédure de tri dans une fonction séparée de votre `main`, à laquelle vous passez le tableau des pays et sa taille.

Pour rendre votre code plus lisible factorisez aussi dans une fonction la partie de votre boucle qui analyse chaque ligne pour peupler une éléments du tableau. Cette seconde fonction doit prendre en paramètre :

- la ligne courante lue depuis le fichier ;
- un pointeur vers l'enregistrement du tableau à remplir avec les informations de la ligne.

Imprimez ensuite les 15 pays les plus émetteurs selon cette métrique.

Modifiez ensuite cette version de votre programme pour trier selon d'autres métriques (*co2*, *cumulative_co2*).

2.6 Étape 5: CO_2 de consommation par habitant (`step5_adding_co2_per_capita.c`)

La quantité de GES (Gaz à effet de serre) émis par un pays est liée à sa population. Il est donc naturel de s'intéresser à la quantité de CO_2 émise par habitant.

Ajoutez un champ à votre type composite pour stocker le CO_2 de consommation émis par habitant dans chaque pays.

Calculez ensuite dans votre boucle cette quantité. Si le CO_2 de consommation n'est pas connu, considérez aussi que le CO_2 de consommation par habitant vaut -1.

Triez maintenant selon cette nouvelle métrique, et afficher les 15 pays les plus émetteur une fois ramené à leur population.

Conseils et remarques :

- Les émissions du fichier sont en millions de tonnes de CO_2. Utilisez des tonnes pour les émissions par habitant pour éviter des quantités trop petites.

2.7 Étape 6: Compilation multi–fichiers (`step6_factoring_out_struct_read_sort.c`)

En utilisant les indications du début de la séance sur l'édition de liens ("*Linkers and Loaders and Dynamic Libraries*"), déplacez le code de votre fonction d'analyse de ligne et de votre fonction de tri dans un fichier séparé (`step6_country_array.c`) du fichier principal contenant la méthode `main` (que vous appellerez `step6_factoring_out_struct_read_sort.c`). Compilez ensuite de façon incrémentale votre programme, comme dans le TP9.

Conseils et remarques :

- Votre type composite devra être défini dans votre fichier d'entête `step6_country_array.h`.

2.8 Étape Bonus: Bibliothèque partagée

Si vous avez terminé en avance, vous pouvez essayer d'emballer le code objet du fichier `step6_country_array.c` dans une bibliothèque partagée (*shared library*) appelée `libcountry_array.so`.

Conseils et remarques :

- Vous devrez passer les options `-fPIC` et `-shared` à `gcc` pour créer cette bibliothèque. Voir le cours sur l'édition de lien pour la commande complète.
- Vous devrez ensuite utiliser les options `L.` et `-lcountry_array` pour compiler votre exécutable. `-L.` indique de chercher des bibliothèques partagées dans le répertoire courant, et `-l...` indique de quelles bibliothèques (en plus des bibliothèques standard) votre programme a besoin.
- Utiliser `nm` et `ldd` sur l'exécutable et la bibliothèque partagée pour vérifier que les symboles non définis dans l'exécutable sont bien présents dans la bibliothèque, et que l'exécutable contient bien l'information de dépendance vis à vis de la bibliothèque.
- `ldd` ne devrait pas trouver votre bibliothèque, ce qui devrait aussi empêcher votre exécutable de se lancer (avec une erreur du type `"libcountry_array.so: cannot open shared object file: No such file or directory"`). Pour aider `ldd` à trouver votre bibliothèque, ajoutez le répertoire courant (`./`) à la variable `LD_LIBRARY_PATH`. Si `LD_LIBRARY_PATH` n'existe pas, pensez à

la définir par variable d'environnement (avec `export`). (Attention, ajouter `./` à `LD_LIBRARY_PATH` est usuellement considéré comme une vulnérabilité de sécurité. Ne rendez pas ce changement permanent.)

3 Soumission et attendus pour obtenir 0,5 point bonus

3.1 Soumission

- Pour les étapes 6 (Compilation Multi–Fichiers) et Bonus (Bibliothèque partagée), indiquez dans un fichier texte les opérations de compilation que vous avez utilisées. Emballez ces éventuels fichiers texte et l'ensemble de vos codes source C (`step?_*.[ch]`) dans une archive `.zip` (par exemple avec la commande `zip`), et soumettez votre archive sur le site Moodle du cours.
- Attention à respecter la date de soumission indiquée sur le site Moodle du cours. Les rendus en retard ne seront pas comptabilisés.

3.2 Attendus

Pour obtenir 0,5 point bonus, votre TP, sans être parfait, doit marcher convenablement.

4 Collaboration et entraide

Le TP ne donnant lieu qu'à 0,5 point bonus vous êtes fortement encouragé(e)s à collaborer avec vos camarades pour comprendre et progresser sur ce TP.

Attention cependant à ne pas simplement copier/coller du code que vous ne comprendriez pas : une telle approche ne vous permettra pas d'acquérir les compétences nécessaires à votre réussite lors du contrôle de TP sur table. À la fin du TP, vous devez être capables de reproduire une solution fonctionnelle au problème qui a été posé.

— Fin du sujet de TP —