A Merlin et Alexandre, mes deux raisons de vivre...

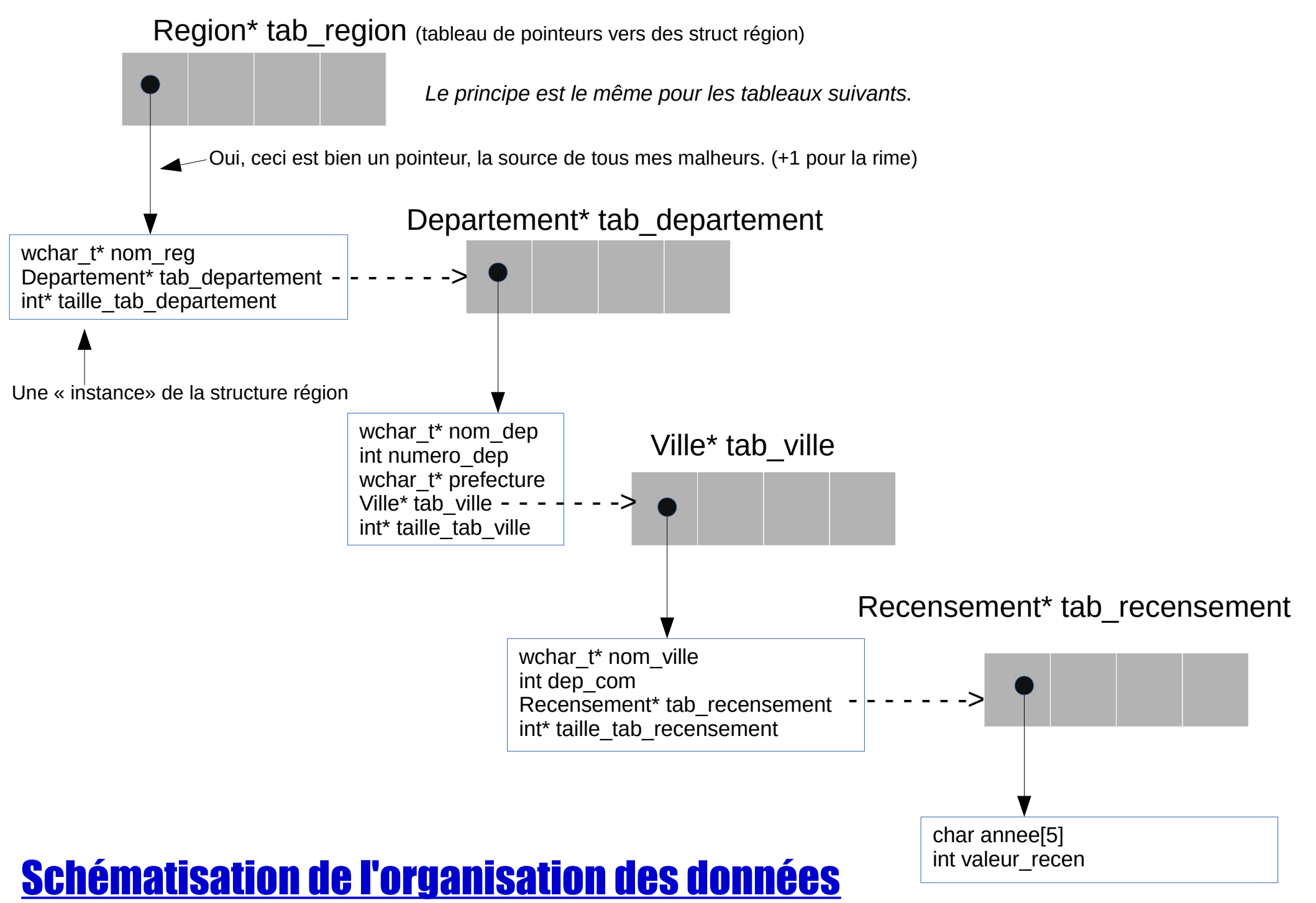
Ce document va vous fournir des explications claires et complètes sans pour autant vous prendre pour des neuneus.

Si jamais une ambiguïté ou une erreur subsistent, merci de me l'indiquer. Si la lecture de ce qui va suivre suscite chez vous une envie irrésistible de me poser une question, bah posez la. Si vous êtes sérieusement bloqués, prévenez moi. Mon week-end est de toute façon consacré ~~au culturisme et à la fornication~~ au C et au droit.

Si malgré tout vous avez des soucis avec la manipulation des données, mettez en commentaire dans votre code les accès aux structures lors des appels des fonctions (ou en dehors) et je remplacerai après. Si tel est le cas, il serait plus aisé que je travaille sur un code qui compile et qui fonctionne. Même si tout le programme n'est pas assemblé, testez vos fonctions avec des données inventées pour le test. Merci d'avance.

[Étant presque mort de fatigue, il y a sûrement des fautes de français qui se baladent.]

Bisous.



Schématisation de l'organisation des données

Tout part d'un tableau de Region* qui est déclaré dans le main.

Cas généraux

Voici les différentes manières d'accéder aux champs des structures en fonction de leur type, que ce soit un tableau de région, département…

Dans le cas d'un(e) :

Int : (tab_xx + i)->monInt

Chaine (tab_xx + i)->maChaine (Rappel, en C ~~chaine1 = chaine2 et chaine1 == chaine2~~. Il faut utiliser les fonctions strcpy et strcmp de string.h)

Int* : *((tab_xx + i)->monInt**[cette étoile ne compte pas hein ;-)]*)

Le wchar a un fonctionnement qui lui est propre. Je vous invite à lire le turotiel d'OpenClassRooms qui vous expliquera clairement et rapidement tous ce qui est nécessaire afin de manipuler les wide char.

http://openclassrooms.com/courses/mettez-des-accents-dans-vos-programmes-avec-le-type-wchar-t

Le cas particulier de l'accès aux tableaux de tableaux

Les cas généraux, appliqués au tableau de région sont très simple mais dès le moment où l'on veut accéder aux champs des sous-tableaux, cela devient délicat.

Exemple :

(tab_region +i)->tab_departement [fonctionne]
(tab_region +i)->(tab_departement +i)->nom_dep [ne fonctionne pas]

Il faut passer par une variable intermédiaire. Appliquée à notre cas :

Departement* tab_departement_tmp = (tab_region + i)->tab_departement
Et ensuite : (tab_departement_tmp + i)->nom_dep
(tab_departement_tmp + i)->tab_ville
…

Il faut après remettre à jour la variable de base : (tab_region + i)->tab_departement = tab_departement_tmp

Attention, si on effectue plusieurs modifications à la suite, il faut penser à réaffecter tab_departement_tmp et ensuite le principe reste le même.

Vous l'aurez compris cela devient vite la merde, mais j'ai pas mieux à proposer pour le moment.

Exemple : Accès à l'année de chaque recensement.

(Bon, à la base, je voulais écrire l'algorithme mais il est trop complexe pour moi qui est à moitié en train de dormir. Et puis si ça se trouve on aura pas besoin de partir du tableau de régions et de brasser tous les recensements de toutes les cases de tous les sous tableaux donc j'écrirai ~~cette merde~~ cet algorithme fort intéressant plus tard.)

Tester le chargement et mémoire du fichier + écriture en fin de programme va peut-être être difficile sans utiliser les accès aux struct.

Pour ce qui est de l'interface, les appels des fonctions qui font parties de la gestion des régions, départements etc... peuvent être plus facilement mis en commentaire si besoin. Il suffit de tester les affichages de données avec du texte random à la place. Les fonction qui affichent les données d'une région, d'un département… sont déjà codées dans leurs fichiers respectifs. L'affichage est pourri voir inexistant pour certaines, mais c'est provisoire (jusqu'à à peu près cet après midi).

Pour me faire pardonner (ou pas), je vais aborder le cas des pointeurs de pointeurs

void* supprimerRegion(Region** tab_region, Region* region_supp, int** taille_tab_region);

On a Région** et int **.

*« Region**, dafuq !? »*

Il n'y a pas de quoi paniquer : prenons le int**. A la base, taille_tab_région est de type int*. taille_tab_region contient donc l'adresse de la case mémoire qui contient l'int. On accède à l'int en faisant *taille_tab_region.

Le int** veut dire qu'on veut l'adresse de l'adresse. En effet, taille_tab_region contient une adresse. Mais cette adresse est stockée dans une case mémoire qui a sa propre adresse ! On recherche donc l'adresse de la case mémoire qui contient l'adresse de la case qui contient la valeur (hé hé).

En C on récupère une adresse avec l'esperluette. L'appel de la fonction se fera de la manière suivante : tab_region = supprimerRegion(&tab_region, tab_region + i, &taille_tab_region);

Il est à noter que les fonctions appelées dans le main et les fonction qui gèrent le menu admin devront prendre en paramètre un Region** tab_region et un int** taille_tab_region.