Nous avons tout d'abord choisi d'implémenter 3 structures :

- Struct fb : qui est la structure d'une zone libre avec la taille de la zone et un pointeur vers la prochaine zone libre.
- Struct ab : qui est la structure d'une zone allouée avec la taille de cette zone.
- Struct header : qui est la structure du bloc principal qui contient la taille de la zone mémoire ainsi qu'un pointeur vers la première zone libre (ou NULL s'il n'y a aucune zone libre dans la mémoire).

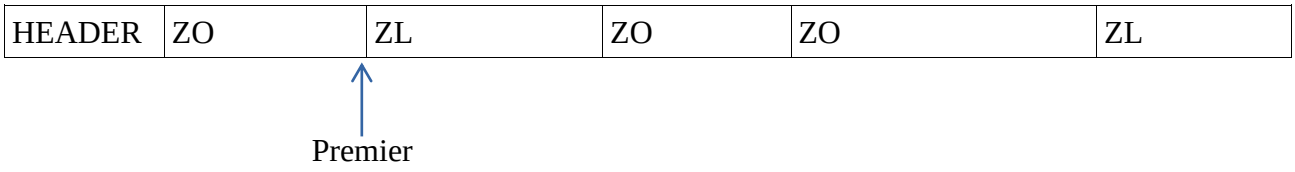
Mem_init : nous avons choisi d'implémenter notre fonction en initialisant le header et les blocs suivant le header à NULL afin de pouvoir les libérer et ensuite préparer la zone mémoire pour l'allocation.

Mem_show : Dans cette fonction, on parcourt la liste des zones (libres/allouées) en commençant à header et en changeant notre pointeur tout en affichant à chaque fois qu'on le change la zone précédemment vue. Si c'était une zone libre, on affiche la zone libre et sa taille sinon on récupère la taille de la zone allouée et on l'affiche tout en prenant en compte le fait qu'il peut y avoir plusieurs zones allouées à côté et dans ce cas on en affiche une puis on prend la taille de l'autre et on l'affiche à nouveau et ainsi de suite jusqu'à ce qu'on tombe à nouveau sur une zone libre.

Mem_alloc : Tout d'abord, on aligne la taille rentrée pour toujours avoir des adresses de taille un multiple de 8. Ensuite, on entoure la zone libre qu'on va transformer en zone allouée. Pour cela, on prend deux pointeurs : Prev et After qui vont permettre de faire les branchements correctement entre les futures zones libres. Dans le cas où la zone qu'on va allouer est la première zone libre dans ce cas on utilise un troisième pointeur : Premier qui pointe toujours vers la première zone libre en mémoire.

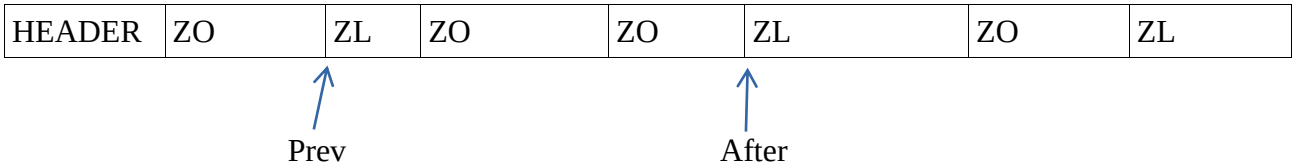
Cas où la taille de la zone libre où on va allouer est strictement supérieure à la taille demandée :

Cas ou Prev = After :



Dans ce cas-là, on décale Premier de la taille de la zone mémoire allouée.

Cas ou Prev != After :



Dans ce cas-là, on crée un autre pointeur vers une zone libre nommée tmp. Ce pointeur permet de garder en mémoire le next d'After, car après on change After en lui donnant comme adresse son adresse initiale plus la valeur de la taille allouée (et de la taille de la structure ab). On peut ainsi, grâce à la sauvegarde remettre le next d'After à l'endroit pointé par tmp. On donne comme adresse à Prev→next la valeur de l'adresse pointée par le nouvel After.

Cas où la taille de la zone libre où on va allouer est égale à la taille demandée :

Cas ou After = Prev :

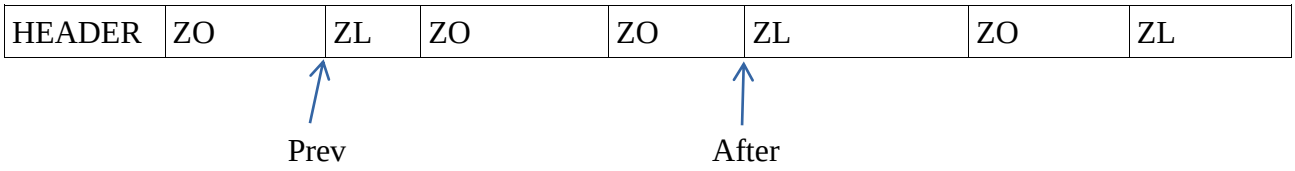
On remplace juste la zone ou on va allouer par une zone allouée. Dans ce cas-là, on réorganise les pointeurs comme il le faut pour que header pointe au bon endroit.

Cas ou After != Prev :

On fait juste en sorte que le next de Prev soit égal au next d'After afin de transformer la zone libre en zone allouée.

Dans le cas où on alloue une zone dans laquelle, après l'allocation, la taille restante est pas assez grande pour accueillir une structure fb, on alloue alors toute la zone à l'utilisateur.

Mem_free : Pour cette fonction, nous vérifions tout d'abord que la zone que nous voulons libérer n'est pas NULL.
Ensuite, nous récupérons dans header le premier bloc de la zone mémoire pour pouvoir récupérer l'adresse de la 1re zone libre. Nous avons ensuite 2 autres pointeurs sur adresse : Prev et After qui nous servent à borner la zone que l'on veut libérer.



On créer une variable new_fb qui pointe sur l'adresse de la nouvelle zone libre (qui est l'adresse de la zone que l'on veut libérer).
Pour libérer, nous gérons 2 cas différents :

- Si l'adresse de la nouvelle zone libre + sa taille est égale à l'adresse d'After, on fusionne alors ces deux zones. Si elle est inférieure, alors le pointeur de new_fb pointe sur l'adresse d'After et sinon il pointe sur NULL s'il n'y a pas de zone libre après.
- Si l'adresse de la zone précédente + sa taille est égale à l'adresse de new_fb, on fusionne alors ces deux zones. Si elle est inférieure, alors le pointeur pointe sur new_fb et sinon la 1re zone libre équivaut à new_fb.

Tests :

- Un premier test consiste à allouer des zones de taille aléatoire et de mémoriser leurs adresses dans un tableau. Ensuite, nous libérons un nombre aléatoire de zone occupée en choisissant aléatoirement dans le tableau les zones à libérer.
- Un second test consiste à allouer 100 zones de taille aléatoire et de les stocker dans un tableau. Nous libérons ensuite aléatoirement 80 zones puis en réallouons de nouveau 80, mais de tailles différentes cette fois-ci.
- Un troisième test consiste à allouer des zones de taille aléatoire. On libère ensuite toutes les adresses qui sont stockées dans le tableau à une position paire puis ceux qui sont dans une position impaire. Et on fait ensuite la même chose même en libérant les zones à une position impaire puis paire.