TP n°6 : gestion de la mémoire et structures autoréférentes

En cas de difficulté, demandez un indice ou une explication à l'enseignant. Chaque exercice doit faire l'objet d'une **validation expérimentale poussée**.

Exercice 1 (acquisition)

Vous devez définir une structure **strBuf** permettant le stockage d'une chaîne de caractères dans un bloc mémoire dont la capacité augmente au fur et à mesure des besoins.

Cette structure doit pouvoir être exploitée par trois fonctions:

void initBuf(strBuf* buf, int capa) : initialise une variable de type **strBuf** avec la capacité de stockage initiale demandée.

void appendBuf(strBuf* buf, char* s) : ajoute une chaîne s à la fin de la chaîne contenue dans le buffer pointé par buf.

char* readBuf(strBuf* buf) : récupère l'adresse de la chaîne de caractères contenue dans le buffer pointé par buf.

Compétences : **Gestion dynamique de la mémoire en C, Chaînes de caractères.**

Exercice 2 (acquisition)

Vous devez définir une structure **strStack** représentant une pile de chaînes de caractères basée sur une structure chaînée, i.e., constituée de maillons reliés ensembles comme dans l'exemple de pile d'entiers développé en cours. Vous devez évidemment définir également la structure représentant les maillons.

La structure **strStack** doit pouvoir être exploitée par trois fonctions:

void initStack(strBStack* p) : initialise une pile vide.

void push(strStack* p, char* s) : empile une *copie* de la chaîne s.

void pop(strStack* p, char* dest) : dépile une chaîne et en recopie le contenu à l'adresse **dest**.

Compétences : **Structures chaînées et autoréférentes en C, Chaînes de caractères.**

Exercice 3 (consolidation)

Reprenez l'exercices 2 en ajoutant les fonctions suivantes :

void reinitStack(strStack* p) : vide complètement une pile existante en libérant toute la mémoire utilisée.

char* toString(strStack* p) : produit (dans un nouveau bloc mémoire) et retourne une chaîne contenant la représentation de toutes les chaînes empilées dans la pile pointée par p, chacune encadrée par des crochets, et séparées par des virgules. Par exemple si **"toto"** et **"titi"** ont été empilés, la chaîne **"[toto],[titi]"** devra être retournée.

Que se passe t-il si on répète un million de fois la séquence suivante (en supposant la pile p correctement définie et initialisée) ?

```
push(p,"toto");push(p,"titi");
printf("%s\n",toString(p));
reinitStack(p);
```

Modifiez le code pour corriger le défaut.

Compétence : **Structures chaînées et autoréférentes en C, Chaînes de caractères.**

Exercice 4 (consolidation)

Réalisez l'implantation complète d'une file d'entiers de type FIFO (le prochain élément à sortir est celui qui est le plus ancien dans la file).

Cette file doit être basée sur une structure chaînée de maillons contenant les entiers stockés. La capacité ne doit pas être limitée (si ce n'est par la mémoire disponible, aucune vérification n'est demandée à ce niveau). L'ajout et le retrait d'une valeur doivent se faire en temps constant (sans boucle ni récursivité). L'exploitation de cette structure ne doit occasionner aucune fuite mémoire.

Vous pouvez compléter cet exercice en réalisant une fonction qui crée et retourne une tableau dynamique contenant toutes les valeurs présentes dans la file.

Compétences : **Structures chaînées et autoréférentes en C.**

Exercice 5 (consolidation)

En utilisant uniquement malloc et free, trouvez un moyen déterminer expérimentalement, à 1 Ko près, la quantité de mémoire disponible dans le tas.

Compétences : **Gestion dynamique de la mémoire en C.**