I2010 : langage C (9)

Tableaux de chaînes de caractères et allocation dynamique

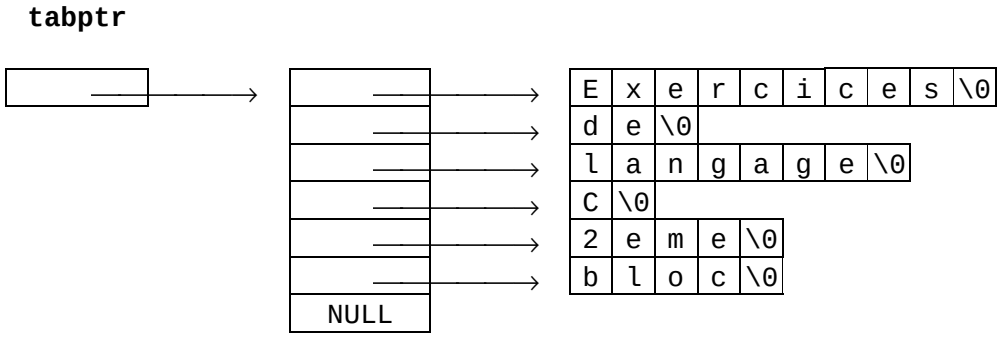
Remarque préliminaire

Quand la taille d'une zone mémoire est connue lors de l'écriture du programme, il est inutile de l'allouer dynamiquement.

L'allocation dynamique ne se justifie que si la taille dépend d'une valeur qui ne sera connue qu'à l'exécution.

Tableaux de chaînes de caractères

Ci-dessous, le dessin d'un tableau de pointeurs vers des chaînes de caractères



1. Ecrivez les définitions nécessaires pour réserver et initialiser ces zones mémoires (sans allocation dynamique).

2. Ecrivez les définitions et les instructions à écrire pour réserver et initialiser ces zones dynamiquement.

3. Dressez un tableau qui donne le type et le contenu des variables suivantes

	type	**contenu**
`tabptr[1]`		
`tabptr[1][1]`		
`tabptr[2][4]`		
`tabptr[6]`		
`*tabptr[4]`		
`*(tabptr[4]+1)`		
`*(tabptr[4])+1`		
`(*tabptr[4])+1`		
`*tabptr[4]+1`		

4. Si la déclaration et les instructions suivantes sont ajoutées

```
char ** adr_tab ;

adr_tab = tabptr ;
```

Complétez le tableau suivant

	type	**contenu**
`*adr_tab`		
`**adr_tab`		
`*(adr_tab+3)`		
`*(*(adr_tab+5))`		
`*adr_tab+4`		
`**adr_tab+2`		
`*(*adr_tab+3)`		

Comme expliqué dans le syllabus en ligne, les arguments d'un programme sont connus par l'intermédiaire d'un tableau tel que celui dont il est question ci-dessus.

Si, dans le shell, l'utilisateur introduit la commande suivante :

```
bonjour quel temps fait-il aujourd'hui ?
```

et que la fonction main du programme bonjour est déclarée comme suit :

```
int main (int argc, char **argv) ;
```

dessinez les contenus des variables `argc` et `argv`

5. Après chaque instruction, précisez le contenu de la variable modifiée et écrivez ce que le programme doit afficher

```
char ** adr_tab ;
int i;

adr_tab = tabptr ;

printf("adr_tab : %p\t%s\n",adr_tab,*adr_tab);


for (i=0;i<6;i++){
        printf("++adr_tab : %p",++adr_tab);
        printf("\t%s\n",*adr_tab);
}
```

```
adr_tab = tabptr ;
printf("*adr_tab : %p\t%s\n",*adr_tab,*adr_tab);


for (i=0;i<4;i++){
        printf("++*adr_tab : %p",++*adr_tab);
        printf("\t%s\n",*adr_tab);
}
```

6. Reprenez le programme écrit à la séance 8 et modifiez-le afin qu'il
    - reçoive deux arguments sur la ligne de commande, deux nombres entiers M et N
    - lise N mots d'au plus M caractères
    - les mette en table,
    - tant que la fin de fichier n'est pas rencontrée
        - lise un mot d'au plus M caractères
        - regarde si le mot est contenu dans la table
        - affiche le mot suivi de « présent » ou « absent » suivant le cas.
    - affiche le nombre de fois qu'un mot lu n'a pas été trouvé dans la table

Pour cet exercice, il vous est demandé d'utiliser la fonction `fgets` pour lire un mot et de refuser les lignes vides et les lignes trop logues.