Travaux Pratiques N°10

Allocation dynamique de mémoire

I. Allocation dynamique de mémoire

L'allocation dynamique consiste à réserver manuellement de l'espace en mémoire pour une variable ou un tableau.

L'allocation dynamique de mémoire est basée principalement sur les deux méthodes malloc et free de la bibliothèque<stdlib.h>.

- malloc (« Memory ALLOCation », c'est-à-dire « Allocation de mémoire ») : demande au système d'exploitation la permission d'utiliser de la mémoire ;
- free(« Libérer ») : permet d'indiquer à l'OS que l'on n'a plus besoin de la mémoire qu'on avait demandée. La place en mémoire est libérée, un autre programme peut maintenant s'en servir au besoin.

II. Allocation de mémoire pour une seule variable

Exemple :

```
void main ( ) {
int* memoireAllouee = NULL;
memoireAllouee = malloc (sizeof (int));
/* Allocation de la mémoire*/
        printf("Quel age avez-vous ? ");
scanf("%d", memoireAllouee);
printf ("Vous avez %d ans\n", *mémoireAllouee);
free (memoireAllouee); /* Libération de mémoire */
}
```

III. Allocation de mémoire pour un tableau

On peut essentiellement déplacer un pointeur dans un plan mémoire à l'aide des opérateurs d'addition, de soustraction, d'incrémentation et de décrémentation. Le déplacement ne peut se faire que d'un nombre de cases mémoire multiple du nombre de cases réservées en mémoire pour la variable sur laquelle il point (n*sizeof (type)).

Exemple :

Solution 1	**Solution 2**
int n , i ; float *tab; scanf("%d", &n); **/*réservation de n * 4 octes*/** tab =(float *) malloc (n* sizeof (float)); **/*remplissage du tab par n reels*/** printf("\n Entrez les %d données à trier\n",n); for (i=0; i<n; i++) scanf("%f", tab+i); /* tab+ i⇔&tab[i] */ **/*affichage du tab */** printf("\n Voici les éléments \n"); for (i=0; i<n ; i++) printf("% .2f ", *(tab+ i)) ; /* *(tab+ i) ⇔ tab [i] */ **/*libération de la mémoire réservée*/** free (tab);	int n ; float *tab, *p; scanf ("%d", &n); **/*réservation de n * 4 octes*/** tab =(float *) malloc (n* sizeof (float)); **/*remplissage du tab par n reels*/** printf("\n Entrez les %d données à trier\n",n); for (p=tab; p<tab+n; p++) scanf("%f", p); **/*affichage du tab */** printf("\n Voici les éléments \n"); for (p=tab; p<tab+n; p++) printf("% .2f ", *p); **/*libération de la mémoire réservée*/** free (tab);

IV. Travail demandé

Exercice 1 : Fréquence

Soit T un tableau contenant n éléments de type entier et x un entier quelconque.
Écrire une fonction **int frequence (int * T , int n , int x)** qui retourne le nombre d'apparitions de x dans le tableau T.

Exercice 2 : Occurrence de 0

Soit T un tableau contenant n éléments de type entier,
Écrire une fonction **void SuppOccZero (int * T , int *n)** qui permet d'effacer toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants.
Rq: Le nombre d'éléments doit changer!!

Exercice 3 : Inverse

On souhaite écrire un programme C qui lit la dimension N d'un tableau T de type int remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.
Pour cefait on vous demande d'écrire les fonctions c suivantes:

void remplir (int *t, int n)
void affiche (int *t, int n)
void inverser (int *t, int n)

Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.