Corrigé série 10.1 : Allocation dymanique

Exercice 1 : Débordement mémoire [Facile]

Ecrire un programme qui **alloue** dynamiquement des emplacements pour des tableaux d'entiers dont la taille est fournie en donnée. Les allocations se poursuivront jusqu'à ce que l'on aboutisse à un débordement de mémoire. L'exécution se présentera ainsi :

```
Taille souhaitée ? 6000
Allocation bloc numéro : 1
Allocation bloc numéro : 2
…
Mémoire insuffisante - arrêt de l'exécution !!!
```

i) Version avec **do while**

```
#include <stdio.h>
#include <stdlib.h>

#define TYPE int
int main()
{
  long taille;
  int  nbloc = 1;
  TYPE  *adresseMemoire = NULL;
  printf("Taille souhaitee ? "); scanf("%ld", &taille);
  do
  {
    adresseMemoire = malloc(taille*sizeof(TYPE));
    if (adresseMemoire != NULL)
    {
      printf("Allocation bloc numero : %d\n", nbloc);
      nbloc++;
    }
  }while(adresseMemoire!=NULL);

  printf("Memoire insuffisante\n");
  system("pause");
  return 0;
}
```

ii) Version avec **while**

```
…
  while( malloc(taille * sizeof(int))) != NULL)
  {
     printf("Allocation bloc numéro : %d\n", nbloc);
     nbloc++;
  }
…
}
```

Exercice 2 : Allocation dynamique d'un tableau à **une** dimension **[Facile]**

Ecrire un programme qui alloue dynamiquement un tableau de nombres entiers.

a. La taille du tableau est saisie au clavier.
b. Le tableau est alloué dynamiquement.
c. Le tableau est rempli avec des valeurs aléatoires entre 32 et 126,
d. Parcourir le tableau et afficher les valeurs et les adresses du tableau.
    i. Avec un formalime tableau
    ii. Avec un formalisme pointeur
    iii. Avec un pointeur incrémenté
e. Parcourir ce tableau et l'afficher caractère par caractère **[Avancé]**
f. Libèrez la mémoire

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    int *tab=NULL, *ptr=NULL;
    int N;
    int i;

    /* a) Saisie de la taille */
    printf("Entrez la taille du tableau");
    scanf("%d", &N);
    /* b) Allocation dynamique */
    tab = malloc(N*sizeof(int));
    if(tab==NULL)    {
        printf("Problème allocation mémoire\n");
        exit(1);
    }
    /* c) Remplissage */
    srand(time(NULL));
    for(i=0;i<N;i++)
        tab[i]=32+rand()%(127-32);
    /* di) Affichage formalisme tableau */
    for(i=0;i<N;i++)
        printf("%d ",tab[i]);
    printf("\n");
    /* dii) Affichage formalisme pointeur */
    for(i=0;i<N;i++)
        printf("%d ",*(tab+i));
    printf("\n");
    /* diii) Affichage pointeur incrémenté */
    ptr=tab;
    for(i=0;i<N;i++)
        printf("%d ",*ptr++);
    printf("\n");
    /* e) Affichage byte par byte */
    char* ptrC=(char*)tab;
    for(i=0;i<N*sizeof(int);i++){
        printf("%3d/%c\n",*ptrC,*ptrC);
        ptrC++;
    }
    printf("\n");
    /* f) Désallocation(libération mémoire) */
    free(tab);
    tab=ptr=NULL;
    ptrC=NULL;
    return 0;
}
```

Exercice 4 : Allocation dynamique d'un tableau à **deux** dimensions **[Moyen]**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int** ptrtab=NULL, *ptr=NULL;
    int i, j, M, N;
    // a) Saisie de la taille
    printf("Dimensions du tableau (M N): ");
    scanf("%d %d", &M, &N);

    // b) Allocation dynamique
    ptrtab = malloc(M*sizeof(int*));
    if(ptrtab==NULL)
    {
        printf("Problème allocation mémoire\n");
        exit(1);
    }
    for(i=0;i<M;i++)
    {
        ptrtab[i] = malloc(N*sizeof(int));
        if(ptrtab[i]==NULL)
        {
            printf("Problème allocation mémoire\n");
            exit(1);
        }
    }
    //c) Remplissage
    srand(time(NULL));
    for(i=0;i<M;i++)
        for(j=0;j<N;j++)
            ptrtab[i][j]= 32 + (rand()%(127-32));
    // d) Affichage
    for(i=0;i<M;i++)
    {
        for(j=0;j<N;j++)
            printf("%3d ",ptrtab[i][j]);    //formalisme tableau
          //printf("%3d ",*(*(ptrtab+i)+j));//formalisme pointeur
          //printf("%3d ",*ptr++); //formalisme pointeur incrémenté
        printf("\n");
    }
    printf("\n");

    /* Désallocation */
    for(i=0;i<M;i++)
        free(ptrtab[i]);
    free(ptrtab);
    ptrtab=NULL;
    system("Pause");
    return 0;
}
```

Exercice 5 : Allocation dynamique d'un tableau à **trois** dimensions **[Avancé]**

Même exercice que le 2) mais avec un tableau à **trois** dimensions.

Exercice 6 : Allocation dynamique et manipulation de chaînes de caractères **[Avancé]**

Ecrire un programme qui lit 10 phrases d'une longueur maximale de 200 caractères au clavier et qui les mémorise dans un tableau de pointeurs sur **`char`** en réservant dynamiquement l'emplacement en mémoire pour les chaînes. Ensuite, l'ordre des phrases est inversé en modifiant les pointeurs et le tableau résultant est affiché.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
 // Déclarations
 char intro[500]; //chaîne pour l'introduction des données
 const int N=10 ;
 char *texte[N];   //Tableau des pointeurs sur les N chaînes
 char *pAide;     //pointeur d'aide pour l'échange des pointeurs
 int i,j;

 // Saisie des données et allocation dynamique de mémoire
 printf("Introduire %d phrases\n", N);

 for (i=0; i<N; i++)
 {
      /* Lecture d'une phrase */
      printf("Phrase %d : ", i);
      gets(intro);
      /* Réservation de la mémoire */
      texte[i] = malloc(strlen(intro)+1);
      /* S'il y a assez de mémoire, ... */
      if (texte[i] !=NULL)
           // copier la phrase à l'adresse fournie par malloc
           strcpy(texte[i], intro);
      else
      {
           printf("\aPas assez de mémoire \n");
           exit(-1);           // quitter le programme.
      }
 }



 /* Afficher le tableau donné */
```

```
 puts("Contenu du tableau donné :");
 for (i=0; i<N; i++)   puts(texte[i]);

 /* a) Inverser l'ordre des phrases avec le pointeur pAide */
 for (i=0, j=N-1 ; i<j ; i++, j--)
 {
      pAide    = texte[i];
      texte[i] = texte[j];
      texte[j] = pAide;
 }

// Afficher le tableau résultat
 puts("Contenu du tableau résultat :");
 for (i=0; i<N; i++)   puts(texte[i]);

// Libérer la mémoire
 for (i=0; i<N; i++)
 {
   free(texte[i]);
   texte[i]=NULL;
 }
 system("pause");
 return 0;
}
```