```
/*****************************************************************************
Programme:      Epreuve du 09/01/2004
Auteur:         Stéphane Bressani
Description:    Recherche du nombre de nombre pair et impair dans 2 tableau
                en utilisant des pointeurs
Version:        1.0     09/01/2004
*****************************************************************************/
#pragma hdrstop
#pragma argsused

//----------------------------   INCLUDE   ----------------------------------

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

//----------------------------   DEFINES   ----------------------------------

#define MAX 99                  // taille du tableau
#define SIZETAB1 6              // limite des nombres aléatoires
#define SIZETAB2 10             // limite des nombres aléatoires

//----------------------------   TYPES   ------------------------------------

typedef struct                  // pour les résultats
{
   int pair;
   int impair;
}TResult;

//----------------------------   PROTOTYPES   -------------------------------

int NbrPair(int *,int);
int NbrImpair(int *,int);
void AfficheTab(int *,int);
void AfficheResultat(TResult *);

/*****************************************************************************
FONCTION:       NbrPair()
DESCRIPTION:    Cherche les nombres paire
PARAMETRE 1:    Le pointeur sur le tableau
PARAMETRE 2:    La taille du tableau
RETOUR:         Le nombre de nombres paire
*****************************************************************************/

int NbrPair (int *Ptr,int taille)
{
   int i,pair = 0;

   for (i= 0; i < taille;i++)
   {
      if ((*(Ptr + i) %2) != 0)   // si le pointeur MOD 2 est pas egal a 0
      {       // rien car impair !
      }
      else
         pair++;    // sinon en incrémente :)
   }
   return pair;
}

/*****************************************************************************
FONCTION:       NbrImpair()
DESCRIPTION:    Cherche les nombres impaire
PARAMETRE 1:    Le pointeur sur le tableau
PARAMETRE 2:    La taille du tableau
RETOUR:         Le nombre de nombres impaire
*****************************************************************************/

int NbrImpair (int *Ptr,int taille)
{
   int i,impair = 0;

   for (i= 0; i < taille;i++)
   {
```

```
      if ((*(Ptr + i) %2) != 0)    // si le pointeur MOD 2 est pas egal a 0
         impair++;     // alors en incrémente :)
   }
   return impair;
}


/*****************************************************************************
FONCTION:        AfficheTab()
DESCRIPTION:     Affiche le contenu d'un tableau
PARAMETRE 1:     Le pointeur sur le tableau
PARAMETRE 2:     La taille du tableau
RETOUR:          Rien
*****************************************************************************/
void AfficheTab (int *Ptr,int taille)
{
   int i;

   for (i= 0; i < taille;i++)
      printf("%3d",*(Ptr + i));    // affiche le contenu du tableau
}

/*****************************************************************************
FONCTION:        AfficheResultat()
DESCRIPTION:     Affiche les résultats
PARAMETRE 1:     Le pointeur sur le tableau
RETOUR:          Rien
*****************************************************************************/
void AfficheResultat (TResult *Ptr)
{
   printf("\n le nombre de pair = %3d",Ptr->pair);
   printf("\n le nombre d'impair = %3d",Ptr->impair);
}

/*****************************************************************************
FONCTION:        main()
DESCRIPTION:     Programme principal
*****************************************************************************/
int main(int argc, char* argv[])
{
   int Tab1[6], *PTab1 = &Tab1[0]; // tab1 et pointeurs
   int Tab2[10], *PTab2 = &Tab2[0]; // tab 2 et pointeurs
   int i;
   TResult ResultTab1, *PResultTab1 = & ResultTab1; //pointeur des Resultat tab1
   TResult ResultTab2, *PResultTab2 = & ResultTab2; //pointeur des Resultat tab2

   randomize();
   do
   {
      clrscr();
      printf("\n");
//------------------------ remplit les 2 tableaux ----------------------------
      for (i=0;i<SIZETAB1;i++)
        Tab1[i] = random(MAX);

      for (i=0;i<SIZETAB2;i++)
        Tab2[i] = random(MAX);
//----------------- Traitement du tableau avec les 6 nombres ------------------
      printf("\n Tableau de 6 nombres     :  ");
      AfficheTab(PTab1,SIZETAB1);
      ResultTab1.pair = NbrPair(PTab1,SIZETAB1);
      ResultTab1.impair = NbrImpair(PTab1,SIZETAB1);
      AfficheResultat(PResultTab1);
//----------------- Traitement du tableau avec les 10 nombres -----------------
      printf("\n\n\n\n Tableau de 10 nombres     :  ");
      AfficheTab(PTab2,SIZETAB2);
      ResultTab2.pair = NbrPair(PTab2,SIZETAB2);
      ResultTab2.impair = NbrImpair(PTab2,SIZETAB2);
      AfficheResultat(PResultTab2);

      printf("\n\n\n Nouvel essai ? [O/N]\n");
   }while(toupper(getch())=='O');
   return 0;
}
```

```
/*****************************************************************************
Programme:      Epreuve du 09/01/2004
Auteur:         Stéphane Bressani
Description:    Recherche du nombre de nombre pair et impair dans 2 tableau
                en utilisant des pointeurs *** -> avec l'allocation <- ***
Version:        2.0     09/01/2004
*****************************************************************************/
#pragma hdrstop
#pragma argsused

//----------------------------   INCLUDE   ----------------------------------

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

//----------------------------   DEFINES   ----------------------------------

#define MAX 99                  // taille du tableau
#define SIZETAB1 6              // limite des nombres aléatoires
#define SIZETAB2 10             // limite des nombres aléatoires

//----------------------------   TYPES   ------------------------------------

typedef struct                  // pour les résultats
{
   int pair;
   int impair;
}TResult;

//----------------------------   PROTOTYPES   -------------------------------

int NbrPair(int *,int);
int NbrImpair(int *,int);
void AfficheTab(int *,int);
void AfficheResultat(TResult *);

/*****************************************************************************
FONCTION:       NbrPair()
DESCRIPTION:    Cherche les nombres paire
PARAMETRE 1:    Le pointeur sur le tableau
PARAMETRE 2:    La taille du tableau
RETOUR:         Le nombre de nombres paire
*****************************************************************************/

int NbrPair (int *Ptr,int taille)
{
   int i,pair = 0;

   for (i= 0; i < taille;i++)
   {
      if ((*(Ptr + i) %2) != 0)   // si le pointeur MOD 2 est pas egal a 0
      {       // rien car impair !
      }
      else
         pair++;     // sinon en incrémente :)
   }
   return pair;
}

/*****************************************************************************
FONCTION:       NbrImpair()
DESCRIPTION:    Cherche les nombres impaire
PARAMETRE 1:    Le pointeur sur le tableau
PARAMETRE 2:    La taille du tableau
RETOUR:         Le nombre de nombres impaire
*****************************************************************************/

int NbrImpair (int *Ptr,int taille)
{
   int i,impair = 0;

   for (i= 0; i < taille;i++)
   {
```

```
      if ((*(Ptr + i) %2) != 0)    // si le pointeur MOD 2 est pas egal a 0
         impair++;     // alors en incrémente :)
   }
   return impair;
}

/*****************************************************************************
FONCTION:        AfficheTab()
DESCRIPTION:     Affiche le contenu d'un tableau
PARAMETRE 1:     Le pointeur sur le tableau
PARAMETRE 2:     La taille du tableau
RETOUR:          Rien
*****************************************************************************/
void AfficheTab (int *Ptr,int taille)
{
   int i;

   for (i= 0; i < taille;i++)
      printf("%3d",*(Ptr + i));    // affiche le contenu du tableau
}

/*****************************************************************************
FONCTION:        AfficheResultat()
DESCRIPTION:     Affiche les résultats
PARAMETRE 1:     Le pointeur sur le tableau
RETOUR:          Rien
*****************************************************************************/
void AfficheResultat (TResult *Ptr)
{
   printf("\n le nombre de pair = %3d",Ptr->pair);
   printf("\n le nombre d'impair = %3d",Ptr->impair);
}

/*****************************************************************************
FONCTION:        main()
DESCRIPTION:     Programme principal
*****************************************************************************/
int main(int argc, char* argv[])
{
   int *PTab1; // pointeur du tableau 1 de 6 caractère
   int *PTab2; // pointeur du tableau 2 de 10 caractère
   int i;
   TResult *PResultTab1; //pointeur des Resultat tab1
   TResult *PResultTab2; //pointeur des Resultat tab2
//---------- réservation en mémoire avec initialisation des pointeurs ----------
   PTab1 = malloc(sizeof(int) * SIZETAB1);
   PTab2 = malloc(sizeof(int) * SIZETAB2);
   PResultTab1 = malloc(sizeof(TResult));
   PResultTab2 = malloc(sizeof(TResult));

   randomize();
   do
   {
      clrscr();
      printf("\n");
//------------------------ remplit les 2 tableaux ---------------------------
      for (i=0;i<SIZETAB1;i++)
         *(PTab1 + i) = random(MAX);

      for (i=0;i<SIZETAB2;i++)
         *(PTab2 + i) = random(MAX);

//----------------- Traitement du tableau avec les 6 nombres ------------------
      printf("\n Tableau de 6 nombres    :  ");
      AfficheTab(PTab1,SIZETAB1);
      PResultTab1->pair = NbrPair(PTab1,SIZETAB1);
      PResultTab1->impair = NbrImpair(PTab1,SIZETAB1);
      AfficheResultat(PResultTab1);
//----------------- Traitement du tableau avec les 10 nombres -----------------
      printf("\n\n\n\n Tableau de 10 nombres    :  ");
      AfficheTab(PTab2,SIZETAB2);
      PResultTab2->pair = NbrPair(PTab2,SIZETAB2);
      PResultTab2->impair = NbrImpair(PTab2,SIZETAB2);
      AfficheResultat(PResultTab2);
```

```
    printf("\n\n\n Nouvel essai ? [O/N]\n");
  }while(toupper(getch())=='O');
  return 0;
}
```