Corrigé série 8.4 : Les structures

Exercice 1 : Temps

Définir une structure capable de mémoriser le temps d'un marathon (hh:mm:ss).
Déclarer une variable t1 et lui affecter le temps de 5h 03 min et 17 sec.
Ecrire une fonction qui affiche le temps de la variable t1.

Appel : `afficherTemps(t1);` -> 05:03:17

Ecrire une fonction qui retourne un temps et qui reçoit les heures, minutes et secondes en paramètres.

Appel :
```
t2 = initialiserTemps(2, 2, 57);
afficherTemps(t2);
```
-> 02:02:57

Utiliser un `typedef` pour remplace `struct…` par un type `TempsTy`.

Corrigé : sans typedef

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

struct Temps {
    unsigned char hh;
    unsigned char mm;
    unsigned char ss
};

void afficherTemps(struct Temps t)
{
    printf("%02d:%02d:%02d\n", t.hh, t.mm, t.ss);
}

struct Temps initialierTemps(int h, int m, int s)
{
    struct Temps t;
    t.hh=h;
    t.mm=m;
    t.ss=s;
    return t;
}

int main()
{

    struct Temps t1 = {0, 0, 0};
    struct Temps t2 = {0, 0, 0};
    t1.hh=5; t1.mm=3; t1.ss=17;
    afficherTemps(t1);
    t2=initialierTemps(2,2,57);
    afficherTemps(t2);

    return 0;
}
```

Corrigé : avec typedef

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

typedef struct                  // dans la portée globale (avant main())
{
    unsigned char hh;
    unsigned char mm;
    unsigned char ss
} TempsTy;

void afficherTemps(TempsTy t)   // défini ou déclaré (prototype) avant main()
{
    printf("%02d:%02d:%02d\n", t.hh, t.mm, t.ss);
}

TempsTy initialierTemps(int h, int m, int s)
{
    TempsTy t;
    t.hh=h;
    t.mm=m;
    t.ss=s;
    return t;
}


int main()
{


    TempsTy t1 = {0, 0, 0};
    TempsTy t2 = {0, 0, 0};
    t1.hh=12; t1.mm=14; t1.ss=18;
    afficherTemps(t1);
    t2=initialierTemps(23,56,1);
    afficherTemps(t2);

    return 0 ;

}
```

Exercice 2 : Nombres complexes [MOYEN]

Compléter le programme donné:

- définir le type `ComplexTy` avec 2 membres de type réel: `reelle`, `imaginaire`
- ajouter le code à scanf pour que la saisie d'un nombres complexe ne se fasse qu'au format (r,i) et encapsuler ce code dans une fonction **`saisieComplexe().`**
- ajouter les fonctions appelées pour calculer la somme, et la différence de deux nombres complexes.

  `sommeComplexe()` : (a1 + i b1)**+**(a2 + i b2) = (a1**+**a2) + i (b1**+**b2)

  `differenceComplexe()` : (a1 + i b1)**-**(a2 + i b2) = (a1**-**a2) + i (b1**-**b2)

- implémenter une fonction **`afficheComplexe(…)`** qui affiche un nombre complexe sous la forme suivante : **5.00 + -3.00 i** .

```
int main()
{
  ComplexeTy c1, c2, resultat;
  c1 = saisieComplexe();
  c2 = saisieComplexe();

  resultat = sommeComplexe(c1, c2);
  printf("\nSomme des deux nombres      : ");
  afficheComplexe(resultat);

  resultat = differenceComplexe(c1, c2);
  printf("\nDifference des deux nombres : ");
  afficheComplexe(resultat);

  return 0;
}
```

Corrigé

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

typedef struct
{
  float reelle ;
  float imaginaire;
} ComplexeTy;

// Prototypes des fonctions
ComplexeTy saisieComplexe();
ComplexeTy sommeComplexe(ComplexeTy a, ComplexeTy b);
ComplexeTy differenceComplexe(ComplexeTy a, ComplexeTy b);
void       afficheComplexe(ComplexeTy c);


int main()
{
  ComplexeTy c1, c2, resultat;
  c1=saisieComplexe();
  c2=saisieComplexe();

  resultat = sommeComplexe(c1, c2);

  printf("\nSomme des deux nombres      : ");
  afficheComplexe(resultat);
  resultat = differenceComplexe(c1, c2);
  printf("\nDifference des deux nombres : ");
  afficheComplexe(resultat);

  system("Pause");
  return 0;
}

ComplexeTy saisieComplexe()
{
  ComplexeTy nouveau;
  int status;

  printf("Donner la valeur d'un nombre complexe (r,i): ");
  status = scanf("(%f, %f)", &nouveau.reelle, &nouveau.imaginaire);
  fflush(stdin);
  while (status != 2)
  {
    printf("ERREUR: Donner la valeur d'un nombre complexe (r,i): ");
    status = scanf("(%f, %f)", &nouveau.reelle, &nouveau.imaginaire);
    fflush(stdin);
  };
  return nouveau;       // COPIE de la variable renvoyée à main
}
```

```
ComplexeTy sommeComplexe(ComplexeTy a, ComplexeTy b)
{
  ComplexeTy c;
  c.reelle     = a.reelle     + b.reelle;
  c.imaginaire = a.imaginaire + b.imaginaire;

  return c;
}

ComplexeTy differenceComplexe(ComplexeTy a, ComplexeTy b)
{
  ComplexeTy c;
  c.reelle     = a.reelle     - b.reelle;
  c.imaginaire = a.imaginaire - b.imaginaire;

  return c;
}

void afficheComplexe(ComplexeTy c)
{
  printf("%.2f + %.2f i\n", c.reelle, c.imaginaire);
}
```

Exercice 3 : tableau de structure

Gestion d'une liste de personnes. Créez un type **`PersonTy`** pour une personne, caractérisée par son nom (20 caractères), son prénom (20 caractères) et son année de naissance.

- Déclarer un tableau de 10 personnes et initialisez-le avec les données des données des sept personnes suivantes:

  "Knuth", "Donald", 1938
  "Kernighan", "Brian", 1942
  "Ritchie", "Dennis", 1941
  "Thompson", "Ken", 1943
  "Minsky", "Marvin", 1927
  "Dijkstra", "Edsger", 1930
  "Stroustrup", "Bjarne", 1950

- Afficher toutes les personnes âgées de moins de 80 ans.
- Trouver les personnes qui seront classées en tête de liste et en queue de liste selon leur nom.
- Qui sont ces personnes et qu'ont-elles faites ?

Sortie de ce programme :

```
Brian Kernighan (76)
Dennis Ritchie (77)
Ken Thompson (75)
Bjarne Stroustrup (69)


Le premier: Dijkstra
Le dernier: Thompson

Process returned 0 (0x0)   execution time : 0.010 s
Press any key to continue.
```

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

typedef struct
{
   char nom[20];
   char prenom[20];
   int  anNaissance;
}
PersonTy;

int main()
{
    int i;
    const int N=7;

    PersonTy tab[10] = {{ "Knuth",     "Donald",     1938},
                        { "Kernighan", "Brian",      1942},
                        { "Ritchie",   "Dennis",     1941},
                        { "Thompson",  "Ken",        1943},
                        { "Minsky",    "Marvin Lee", 1927},
                        { "Dijkstra",  "Edsger",     1930},
                        { "Stroustrup","Bjarne",     1950}
                       };

    for(i=0;i<N;i++)
    {
        if (2019-tab[i].anNaissance < 80)
        {
            printf("%s %s %d ans\n", tab[i].prenom,
                                     tab[i].nom,
                                2019-tab[i].anNaissance);
        }
    }

    int indicePlusPetit=0;
    int indicePlusGrand=0;
    for(i=1;i<N;i++)
    {
        if ( strcmp(tab[i].nom, tab[indicePlusPetit].nom )<0 )
            indicePlusPetit = i;
        if ( strcmp(tab[i].nom, tab[indicePlusGrand].nom )>0 )
            indicePlusGrand = i;
    }
    printf("\n");
    printf("Le premier: %s \n",  tab[indicePlusPetit].nom);
    printf("Le dernier: %s  \n", tab[indicePlusGrand].nom);

    return 0;
}
```

Les informations ci-dessous proviennent de Wikipédia.

Marvin Minsky, 1927-2016

Un des fondateurs de l'intelligence artificielle. Il a travaillé dans le domaine des sciences cognitives et de l'intelligence artificielle. Il est également cofondateur, avec l'informaticien John McCarthy du Groupe d'intelligence artificielle du Massachusetts Institute of Technology (MIT) et auteur de nombreuses publications aussi bien en intelligence artificielle qu'en philosophie comme *La Société de l'Esprit* (1986).
Inventeur de l' « ultimate machine ».

Edsger Dijkstra, 1930-2002

Mathématicien et informaticien néerlandais. Il reçoit en 1972 le prix Turing pour ses contributions sur la science et l'art des langages de programmation et au langage Algol.

Constatant les dégâts provoqués par l'usage incontrôlé de l'instruction goto en programmation, il rédige en 1968 pour les *Communications of the ACM* un article qu'il nomme « *A Case against the GOTO Statement* » (« Un Procès contre l'instruction GOTO »).
Dijkstra avait joué un rôle important dans le développement du langage Algol à la fin des années 1950 et développé ensuite « la science et l'art des langages de programmation »

Dijkstra, connu pour son caractère difficile et son intransigeance, était réputé pour ses aphorismes, lesquels résumaient sa vision de la science informatique.

- « Tester un programme peut démontrer la présence de bugs, jamais leur absence. »
- « Se demander si un ordinateur peut penser est aussi intéressant que de se demander si un sous-marin peut nager. »
- « La programmation par objets est une idée exceptionnellement mauvaise qui ne pouvait naître qu'en Californie. »
- « It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration »
- «The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence »

Donald Knuth, 1938

Il est l'auteur d'une centaine d'articles et d'une dizaine de livres sur **l'algorithmique** et les mathématiques discrètes ; les 3 premiers volumes avec la partie déjà publiée du volume 4A (5 fascicules) de *The Art of Computer Programming* (TAOCP) demeurent des ouvrages de référence, ce qui est exceptionnel dans une science comme l'informatique, qui évolue très rapidement.

Afin d'avoir une bonne qualité de mise en page pour la deuxième édition de son TAOCP, Knuth a créé deux logiciels libres, par la suite largement utilisés en typographie professionnelle et en mathématiques, TeX et Metafont. Son intérêt pour la typographie l'a également poussé à créer la police *Computer Modern*, police par défaut de TeX.

Note :
Les versions de TeX convergent vers Pi : 3, 3.1, 3.14, 3.1416, etc
Les versions de Metafont convergent vers e.

Dennis Ritchie, 1941-2011

Est un des pionniers de l'informatique moderne, inventeur du langage C et codéveloppeur de Unix avec Thompson.
Il promeut le langage C et rédige notamment le livre de référence *The C Programming Language*.

Brian Kernighan, 1942

Résumé des réalisations

- *Hello, world*, (Bonjour, monde), un programme initialement écrit par Brian Kernighan de Bell Labs dans "un tutoriel d'introduction au B"
- *The C Programming Language*, le premier livre sur le C avec Dennis Ritchie, son créateur
- awk, avec Alfred V. Aho et Peter Weinberger, et son livre *The AWK Programming Language*
- Le langage de composition pic pour troff
- Le langage de composition eqn pour troff avec Lorinda Cherry

Ken Thompson", 1943

Concepteur des systèmes UNIX et Plan 9 ainsi que des langages B et Go.

Il y travaille sur les systèmes d'exploitation à temps partagé, notamment Multics, puis Unix à partir de 1969 et plus tard Plan 9. En 1970, il met au point le langage B, précurseur du C.

Il participe à de nombreuses évolutions du système Unix : portage sur PDP-11, réécriture en langage C, et introduction des *pipes*.

Bjarne Stroustrup, 1950

Informaticien, écrivain et professeur de sciences informatiques danois. Il est connu pour être l'auteur du langage de programmation C++