
TD 8

Structures et pointeurs

Exercice n  1

```
#include <stdio.h>

int Jeu[52];

typedef struct {
    int numero;
    char couleur;
    char figure;
    int PH;
} Carte;

/* la main */
typedef struct {
    Carte Cards[13];
    int TPH;
    int TPL;
    int TPD;
    int Distrib[4];
    char TypeJeu;
} MainB;

/* la ligne */
typedef struct {
    MainB M1;
    MainB M2;
} LigneB;

/* la donne */
typedef struct {
    MainB Nord;
    MainB Est;
    MainB Sud;
```

```

MainB Ouest;
} DonneB;

// Codage des couleurs
char CodeCouleur(int coul)
{
    char result;

    switch(coul){
    case 0 :
        result = 'T';
        break;

    case 1:
        result = 'K';
        break;

    case 2:
        result = 'C';
        break;

    case 3:
        result = 'P';
        break;
    }

    return result;
}

// codage des figures
char CodeFigure (int fig)
{
    char result;

    switch(fig)
    {
    case 0:
        result = '2';
        break;

    case 1:
        result = '3';
        break;

    case 2:
        result = '4';

```

```

        break;

    case 3:
        result = '5';
        break;

    case 4:
        result = '6';
        break;

    case 5:
        result = '7';
        break;

    case 6:
        result = '8';
        break;

    case 7:
        result = '9';
        break;

    case 8:
        result = 'X';
        break;

    case 9:
        result = 'V';
        break;

    case 10:
        result = 'D';
        break;

    case 11:
        result = 'R';
        break;

    case 12:
        result = 'A';
        break;
    }
    return result;
}

```

```

// définition de la valeur en P
int ValPH(char f)

```

```

{
    int result;

    switch(f)
    {
        case 'A':
            result = 4;
            break;

        case 'R':
            result = 3;
            break;

        case 'D':
            result = 2;
            break;

        case 'V':
            result = 1;
            break;

        default:
            result = 0;
    }

    return result;
}

void InitCarte(Carte * pc, int num)
{
    char figure, couleur;
    int ph;

    figure = CodeFigure(num % 13);
    couleur = CodeCouleur(num / 13);

    ph = ValPH(figure);

    pc->numero = num;
    pc->couleur = couleur;
    pc->figure = figure;
    pc->PH = ph;
}

void AfficheEnClair(Carte C)
{

```

```

switch(C.figure)
{
case 'A':
    printf("As");
    break;

case 'R':
    printf("Roi");
    break;

case 'D':
    printf("Dame");
    break;

case 'V':
    printf("Valet");
    break;

default:
    printf("%c",C.figure);
}

printf("_de_");

switch(C.couleur)
{
case 'T':
    printf("Trèfle");
    break;

case 'K':
    printf("Carreau");
    break;

case 'C':
    printf("Coeur");
    break;

case 'P':
    printf("Pique");
    break;
}

printf("\n");
}

void AfficheMain(MainB M)

```

```

{
    int i;

    for(i = 0; i < 13; i++){
        printf("%d\n", (M.Cards[i]).numero);
        AfficheEnClair(M.Cards[i]);
    }
    printf("Total des points H: %d\n", M.TPH);
    printf("Distribution: \n");

    for(i = 0; i < 4; i++)
        printf("%d, ", M.Distrib[i]);
    printf("\n");
}

void Shuffle()
{
    int i, j, s, h;

    for(i = 0; i < 52; i++)
        Jeu[i] = -1;

    for(i = 51; i >= 0; i--)
    {
        j = rand() % (i+1);
        for(h = 0, s = 0; s <= j; h++)
            if (Jeu[h] < 0)
                s++;
        //printf(" i = %d | t j = %d | t h = %d | t s = %d\n", i, j, h, s);
        Jeu[--h] = i;
    }
}

void EchangeCartes(MainB *M, int i, int j)
{
    Carte C;

    C.couleur = (M->Cards[i]).couleur;
    C.figure = (M->Cards[i]).figure;
    C.numero = (M->Cards[i]).numero;
    C.PH = (M->Cards[i]).PH;
    (M->Cards[i]).couleur = (M->Cards[j]).couleur;
    (M->Cards[i]).figure = (M->Cards[j]).figure;
    (M->Cards[i]).numero = (M->Cards[j]).numero;
    (M->Cards[i]).PH = (M->Cards[j]).PH;
    (M->Cards[j]).couleur = C.couleur;
    (M->Cards[j]).figure = C.figure;
}

```

```

    (M->Cards[j]).numero = C.numero;
    (M->Cards[j]).PH = C.PH;
}

void TrieCartes(MainB *M)
{
    int i, j;

    for(i = 0; i < 13; i++)
        for (j = 0; j < 12 - i; j++)
            if ((M->Cards[j]).numero > (M->Cards[j+1]).numero)
                EchangeCartes(M, j+1, j);
}

// Initialisation d'une main
void InitMain(MainB *M, int * t)
{
    int i;

    for(i = 0; i < 4; i++)
        M->Distrib[i] = 0;

    M->TPH = 0;

    for ( i = 0; i < 13; i++) {
        InitCarte(&(M->Cards[i]), t[i]);

        switch((M->Cards[i]).couleur)
        {
            case 'T':
                (M->Distrib[0])++;
                break;

            case 'K':
                (M->Distrib[1])++;
                break;

            case 'C':
                (M->Distrib[2])++;
                break;

            case 'P':
                (M->Distrib[3])++;
                break;
        }
        M->TPH += ValPH((M->Cards[i]).figure);
    }
}

```

```

    }
}

void DealAndSort(DonneB * D)
{
    Shuffle ();
    InitMain (&(D->Nord) , Jeu);
    TrieCartes (&(D->Nord));
    InitMain (&(D->Est) , Jeu+13);
    TrieCartes (&(D->Est));
    InitMain (&(D->Sud) , Jeu+26);
    TrieCartes (&(D->Sud));
    InitMain (&(D->Ouest) , Jeu+39);
    TrieCartes (&(D->Ouest));
}

main ()
{
    int i;
    DonneB D;

    DealAndSort(&D);
    AfficheMain (D.Nord);
    AfficheMain (D.Est);
    AfficheMain (D.Ouest);
    AfficheMain (D.Sud);

}

```

Exercice n° 2

```

#include <stdio.h>

int Jeu[52];

typedef struct {
    int numero;
    char couleur;
    char figure;
    int PH;
} Carte;

/* la main */
typedef struct {

```

```

    Carte Cards[13];
    int TPH;
    int TPL;
    int TPD;
    int Distrib[4];
    char TypeJeu;
} MainB;

/* la ligne */
typedef struct {
    MainB M1;
    MainB M2;
} LigneB;

/* la donne */
typedef struct {
    MainB Nord;
    MainB Est;
    MainB Sud;
    MainB Ouest;
} DonneB;

// Codage des couleurs
char CodeCouleur(int coul)
{
    char result;

    switch(coul){
    case 0 :
        result = 'T';
        break;

    case 1:
        result = 'K';
        break;

    case 2:
        result = 'C';
        break;

    case 3:
        result = 'P';
        break;
    }

    return result;
}

```

```

}

// codage des figures
char CodeFigure (int fig)
{
    char result;

    switch(fig)
    {
        case 0:
            result = '2';
            break;

        case 1:
            result = '3';
            break;

        case 2:
            result = '4';
            break;

        case 3:
            result = '5';
            break;

        case 4:
            result = '6';
            break;

        case 5:
            result = '7';
            break;

        case 6:
            result = '8';
            break;

        case 7:
            result = '9';
            break;

        case 8:
            result = 'X';
            break;

        case 9:
            result = 'V';

```

```

        break;

    case 10:
        result = 'D';
        break;

    case 11:
        result = 'R';
        break;

    case 12:
        result = 'A';
        break;
    }
    return result;
}

// définition de la valeur en P
int ValPH(char f)
{
    int result;

    switch(f)
    {
        case 'A':
            result = 4;
            break;

        case 'R':
            result = 3;
            break;

        case 'D':
            result = 2;
            break;

        case 'V':
            result = 1;
            break;

        default:
            result = 0;
    }

    return result;
}

```

```

void InitCarte(Carte * pc, int num)
{
    char figure , couleur;
    int ph;

    figure = CodeFigure(num % 13);
    couleur = CodeCouleur(num / 13);

    ph = ValPH(figure);

    pc->numero = num;
    pc->couleur = couleur;
    pc->figure = figure;
    pc->PH = ph;
}

```

```

void AfficheEnClair(Carte C)
{
    switch(C.figure)
    {
        case 'A':
            printf("As");
            break;

        case 'R':
            printf("Roi");
            break;

        case 'D':
            printf("Dame");
            break;

        case 'V':
            printf("Valet");
            break;

        default:
            printf("%c",C.figure);
    }
}

```

```

printf("_de_");

```

```

switch(C.couleur)
{
    case 'T':
        printf("Trèfle");
}

```

```

        break;

    case 'K':
        printf("Carreau");
        break;

    case 'C':
        printf("Coeur");
        break;

    case 'P':
        printf("Pique");
        break;
    }

    printf("\n");
}

void AfficheMain(MainB M)
{
    int i;

    for(i = 0; i < 13; i++){
        printf("%d", (M.Cards[i]).numero);
        AfficheEnClair(M.Cards[i]);
    }
    printf("Total des points H: %d\n", M.TPH);
    printf("Distribution: {");

    for(i = 0; i < 4; i++)
        printf("%d, ", M.Distrib[i]);
    printf("}\n");
}

void Shuffle()
{
    int i, j, s, h;

    for(i = 0; i < 52; i++)
        Jeu[i] = -1;

    for(i = 51; i >= 0; i--)
    {
        j = rand() % (i+1);
        for(h = 0, s = 0; s <= j; h++)
            if (Jeu[h] < 0)
                s++;
    }
}

```

```

        //printf(" i = %d\t j = %d\t h = %d\t s = %d\n", i, j, h, s);
        Jeu[--h] = i;
    }
}

void EchangeCartes(MainB *M, int i, int j)
{
    Carte C;

    C.couleur = (M->Cards[i]).couleur;
    C.figure = (M->Cards[i]).figure;
    C.numero = (M->Cards[i]).numero;
    C.PH = (M->Cards[i]).PH;
    (M->Cards[i]).couleur = (M->Cards[j]).couleur;
    (M->Cards[i]).figure = (M->Cards[j]).figure;
    (M->Cards[i]).numero = (M->Cards[j]).numero;
    (M->Cards[i]).PH = (M->Cards[j]).PH;
    (M->Cards[j]).couleur = C.couleur;
    (M->Cards[j]).figure = C.figure;
    (M->Cards[j]).numero = C.numero;
    (M->Cards[j]).PH = C.PH;
}

void TrieCartes(MainB *M)
{
    int i, j;

    for(i = 0; i < 13; i++)
        for (j = 0; j < 12 - i; j++)
            if ((M->Cards[j]).numero > (M->Cards[j+1]).numero)
                EchangeCartes(M, j+1, j);
}

// Initialisation d'une main
void InitMain(MainB *M, int * t)
{
    int i;

    for(i = 0; i < 4; i++)
        M->Distrib[i] = 0;

    M->TPH = 0;

    for ( i = 0; i < 13; i++) {
        InitCarte(&(M->Cards[i]), t[i]);
    }
}

```

```

switch ((M->Cards [ i ] ) . couleur )
{
case 'T':
(M->Distrib [0] ) ++;
break;

case 'K':
(M->Distrib [1] ) ++;
break;

case 'C':
(M->Distrib [2] ) ++;
break;

case 'P':
(M->Distrib [3] ) ++;
break;
}
M->TPH += ValPH ((M->Cards [ i ] ) . figure );
}
}

```

```

void DealAndSort (DonneB * D)
{
Shuffle ();
InitMain (&(D->Nord) , Jeu );
TrieCartes (&(D->Nord) );
InitMain (&(D->Est) , Jeu +13);
TrieCartes (&(D->Est) );
InitMain (&(D->Sud) , Jeu +26);
TrieCartes (&(D->Sud) );
InitMain (&(D->Ouest) , Jeu +39);
TrieCartes (&(D->Ouest) );
}

```

```

main ()
{
int i;
DonneB D;

DealAndSort (&D);
AfficheMain (D. Nord);
AfficheMain (D. Est);
AfficheMain (D. Ouest);
AfficheMain (D. Sud);
}

```