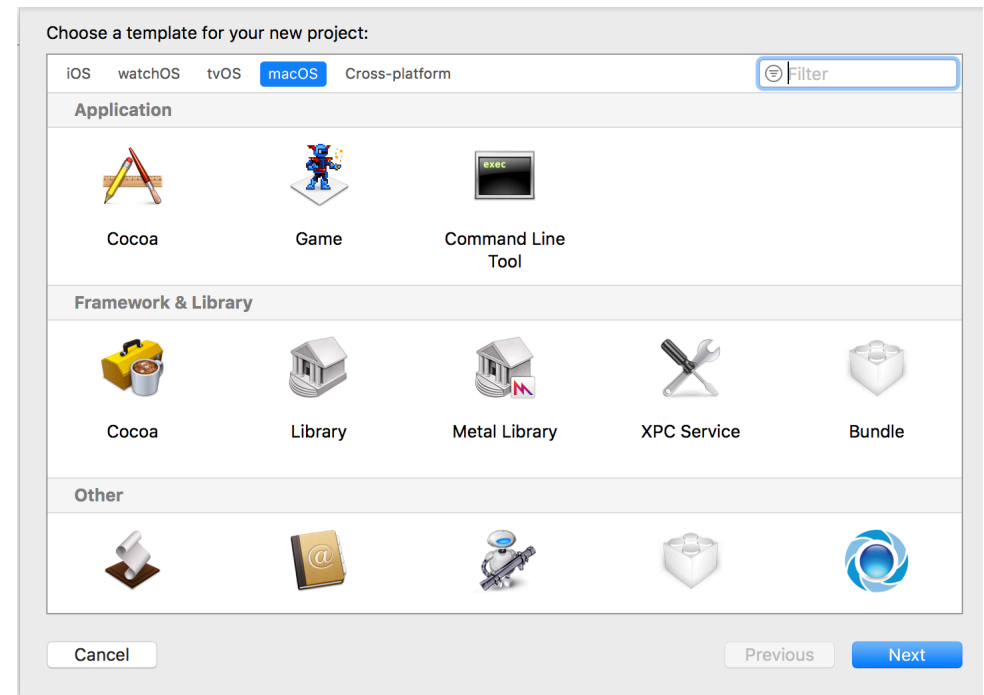




- The first line of the program `#include <stdio.h>` is a preprocessor



Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

# Première Programme en C

## Exercice – 1

```
1  #include <stdio.h>
2
3  int main() {
4      /* Ma première programme en C */
5      printf("Bonjour le monde! \n");
6
7      return 0;
8  }
```

# Définir constant

## Exercice – 2

Avec **# define** préprocesseur

```
1 #include <stdio.h>
2
3 #define LONGUEUR 10
4 #define LARGEUR 5
5 #define NOUVELLE_LIGNE '\n'
6
7 int main() {
8     int region;
9
10
11     region = LONGUEUR * LARGEUR;
12     printf("valeur de la région : %d", region);
13     printf("%c", NOUVELLE_LIGNE);
14
15     return 0;
16 }
```

## Exercice – 3

Avec **const** motsclé

```
1 #include <stdio.h>
2
3 int main() {
4
5     const int LONGUEUR = 10;
6     const int LARGEUR = 5;
7     const char NOUVELLE_LIGNE = '\n';
8     int region;
9
10     region = LONGUEUR * LARGEUR;
11     printf("valeur de la région : %d", region);
12     printf("%c", NOUVELLE_LIGNE);
13
14     return 0;
15 }
```

# Les structures de contrôle en C

Les décisions - if then else

Exercice – 3

```
1  #include <stdio.h>
2
3  int main () {
4
5      /* définition de variable locale */
6      int a = 10;
7
8      /* vérifier la condition booléenne en utilisant if */
9
10     if( a < 20 ) {
11         /* si la condition est vraie, imprimez le texte suivant */
12         printf("a est inférieur à 20\n" );
13     }
14
15     printf("La valeur de a est: %d\n", a);
16
17     return 0;
18 }
```

# Les structures de contrôle en C

## Exercice – 4 Les décisions – “if” then “else if”

```
1 #include <stdio.h>
2
3 int main () {
4
5     /* définition de variable locale */
6     int a = 100;
7
8     /* vérifier la condition booléenne */
9     if( a == 10 ) {
10         /* si la condition est vraie, imprimez le texte suivant */
11         printf("La valeur de a est 10\n" );
12     } else if( a == 20 ) {
13         /* si autre si la condition est vraie */
14         printf("La valeur de a est 20\n" );
15     } else if( a == 30 ) {
16         /* si autre si la condition est vraie */
17         printf("La valeur de a est 30\n" );
18     } else {
19         /* si aucune des conditions n'est vraie */
20         printf("Aucune des valeurs ne correspond\n" );
21     }
22
23     printf("La valeur exacte d'un est: %d\n", a );
24
25     return 0;
26 }
```

## Imbriqué « if »

## Exercice – 5

```
1 #include <stdio.h>
2
3 int main () {
4
5     /* définition de variable locale*/
6     int a = 100;
7     int b = 200;
8
9     /* vérifier la condition booléenne */
10    if( a == 100 ) {
11
12        /* Si la condition est vraie, vérifiez les points suivants */
13        if( b == 200 ) {
14            /* si la condition est vraie, imprimez le texte suivant */
15            printf("La valeur de a est 100 et b est 200\n" );
16        }
17    }
18
19    printf("La valeur exacte d'un est : %d\n", a );
20    printf("La valeur exacte de b est : %d\n", b );
21
22    return 0;
23 }
```

# Les structures de contrôle en C

## Exercice – 6

### Les décisions – “switch”

```
1 #include <stdio.h>
2
3 int main () {
4
5     /* définition de variable locale */
6     char grade = 'B';
7
8     switch(grade) {
9         case 'A' :
10             printf("Excellent!\n" );
11             break;
12         case 'B' :
13             printf("Trés Bien!\n" );
14             break;
15         case 'C' :
16             printf("Bien joué\n" );
17             break;
18         case 'D' :
19             printf("Tu es passé\n" );
20             break;
21         case 'F' :
22             printf("Mieux vaut réessayer\n" );
23             break;
24         default :
25             printf("Grade invalide\n" );
26     }
27
28     printf("Votre note est %c\n", grade );
29
30     return 0;
31 }
```

## Exercice – 7

```
1 #include <stdio.h>
2
3 int main () {
4
5     /* local variable definition */
6     int a = 100;
7     int b = 200;
8
9     switch(a) {
10
11         case 300:
12             printf("This is part of outer switch\n");
13             break;
14         case 400:
15             printf("This is part of outer switch\n");
16             break;
17         case 100:
18             printf("This is part of outer switch\n");
19             break;
20             switch(b) {
21                 case 200:
22                     printf("This is part of inner switch\n");
23                     break;
24                 case 2000:
25                     printf("This is part of inner switch\n");
26                     break;
27                 case 20000:
28                     printf("This is part of inner switch\n");
29                     break;
30             }
31         case 500:
32             printf("This is part of outer switch\n");
33             break;
34         default:
35             printf("This is part of default switch\n");
36     }
37
38     printf("Exact value of a is : %d\n", a );
39     printf("Exact value of b is : %d\n", b );
40
41     return 0;
42 }
43 }
```

## Les itérations

### Exercice – 8 – While

```
1  #include <stdio.h>
2
3  int main () {
4
5      /* définition de variable locale */
6      int a = 10;
7
8      /* pendant l'exécution de la boucle */
9      while( a < 20 ) {
10         printf("valeur de a: %d\n", a);
11         a++;
12     }
13
14     return 0;
15 }
```

### Exercice – 9 – for

```
1  #include <stdio.h>
2
3  int main () {
4
5      int a;
6
7      /* pour l'exécution de la boucle */
8      for( a = 10; a < 20; a = a + 1 ){
9         printf("valeur de a: %d\n", a);
10     }
11
12     return 0;
13 }
```

### Exercice – 10 – do while

```
1  #include <stdio.h>
2
3  int main () {
4
5      /* définition de variable locale */
6      int a = 10;
7
8      /* faire une boucle d'exécution */
9      do {
10         printf("valeur de a: %d\n", a);
11         a = a + 1;
12     }while( a < 20 );
13
14     return 0;
15 }
```



## Imbriqué for

### Exercice – 11

```
1- /**
2  * Programme C pour imprimer la table de multiplication de 1 à 5
3  */
4  #include <stdio.h>
5
6  int main()
7  {
8      /* Déclaration de variable de compteur de boucle */
9      int i, j;
10
11     /* Boucle extérieure */
12     for(i=1; i<=10; i++)
13     {
14         /* Boucle intérieure */
15         for(j=1; j<=5; j++)
16         {
17             printf("%d\t", (i*j));
18         }
19
20         /* Imprimer une nouvelle ligne */
21         printf("\n");
22     }
23
24     return 0;
25 }
```

## Exercice – 12

### Break

```
1 #include <stdio.h>
2
3 int main () {
4     /* définition de variable locale */
5     int a = 10;
6
7     /* l'exécution de la boucle while */
8     while( a < 20 ) {
9
10        printf("valeur de a: %d\n", a);
11        a++;
12
13        if( a > 15) {
14            /* terminer la boucle en utilisant l'instruction break */
15            break;
16        }
17    }
18    return 0;
19 }
20 }
```

## Sortir de boucle

### Exercice – 13

### Continue

```
1 #include <stdio.h>
2
3 int main () {
4     /* définition de variable locale */
5     int a = 10;
6
7     /* faire une boucle d'exécution */
8     do {
9
10        if( a == 15) {
11            /* ignorer l'itération */
12            a = a + 1;
13            continue;
14        }
15
16        printf("valeur de a: %d\n", a);
17        a++;
18    } while( a < 20 );
19
20    return 0;
21 }
22
23 }
```

## Exercice – 14

### goto

```
1 #include <stdio.h>
2
3 int main () {
4     /* définition de variable locale */
5     int a = 10;
6
7     /* faire une boucle d'exécution */
8     LOOP:do {
9
10        if( a == 15) {
11            /* ignorer l'itération */
12            a = a + 1;
13            goto LOOP;
14        }
15
16        printf("valeur de a: %d\n", a);
17        a++;
18    }while( a < 20 );
19
20    return 0;
21 }
22
23 }
```

# Définition et Déclaration de une fonction

## Exercice – 15

```
1  #include <stdio.h>
2
3  /* déclaration de fonction */
4  int max(int num1, int num2);
5
6  int main () {
7
8      /* définition de variable locale */
9      int a = 100;
10     int b = 200;
11     int ret;
12
13     /* appeler une fonction pour obtenir la valeur maximale */
14     ret = max(a, b);
15
16     printf( "La valeur maximale est: %d\n", ret );
17
18     return 0;
19 }
20
21 /* fonction renvoyant le maximum entre deux nombres */
22 int max(int num1, int num2) {
23
24     /* déclaration de variable locale */
25     int result;
26
27     if (num1 > num2)
28         result = num1;
29     else
30         result = num2;
31
32     return result;
33 }
```

## Exercice – 16

### Appeler par valeur

# Appeler une fonction

## Exercice – 17

### Appeler par référence

```

1  #include <stdio.h>
2
3  /* déclaration de fonction */
4  void swap(int x, int y);
5
6  int main () {
7
8      /* définition de variable locale */
9      int a = 100;
10     int b = 200;
11
12     printf("Avant l'échange, valeur de a : %d\n", a );
13     printf("Avant l'échange, valeur de b : %d\n", b );
14
15     /* appeler une fonction pour échanger les valeurs */
16     swap(a, b);
17
18     printf("Après échange, valeur de a : %d\n", a );
19     printf("Après échange, valeur de b : %d\n", b );
20
21     return 0;
22 }
23
24 /* définition de fonction pour échanger les valeurs */
25 void swap(int x, int y) {
26
27     int temp;
28
29     temp = x; /* enregistrer la valeur de x */
30     x = y;    /* mettre y en x */
31     y = temp; /* mettre temp en y */
32
33     return;
34 }

```

```

,-----
Avant l'échange, valeur de a : 100
Avant l'échange, valeur de b : 200
Après échange, valeur de a : 100
Après échange, valeur de b : 200

```

```

1  #include <stdio.h>
2
3  /* déclaration de fonction */
4  void swap(int *x, int *y);
5
6  int main () {
7
8      /* définition de variable locale */
9      int a = 100;
10     int b = 200;
11
12     printf("Avant l'échange, valeur de a : %d\n", a );
13     printf("Avant l'échange, valeur de b : %d\n", b );
14
15     /* appeler une fonction pour échanger les valeurs.
16      * &a indique un pointeur vers a i.e. adresse de la variable a et
17      * &b indique un pointeur vers b i.e. adresse de la variable b
18     */
19     swap(&a, &b);
20
21     printf("Après échange, valeur de a : %d\n", a );
22     printf("Après échange, valeur de b : %d\n", b );
23
24     return 0;
25 }
26
27 /* définition de fonction pour échanger les valeurs */
28 void swap(int *x, int *y) {
29
30     int temp;
31     temp = *x; /* enregistrer la valeur à l'adresse x */
32     *x = *y;    /* mettre y en x */
33     *y = temp;  /* mettre temp en y */
34
35     return;
36 }

```

# Tableaux

## Exercice – 18

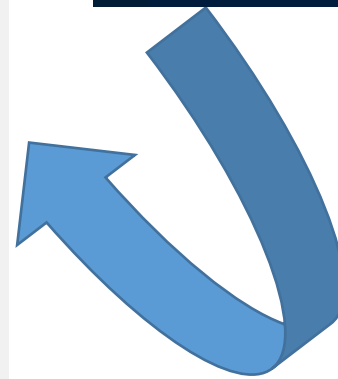
```
1  #include <stdio.h>
2
3  int main () {
4
5      int n[ 10 ]; /* n est un tableau de 10 nombres entiers */
6      int i,j;
7
8      /* initialise les éléments du tableau n à 0 */
9      for ( i = 0; i < 10; i++ ) {
10         n[ i ] = i + 100; /* définir l'élément à l'emplacement i à i + 100 */
11     }
12
13     /* afficher la valeur de chaque élément de tableau */
14     for ( j = 0; j < 10; j++ ) {
15         printf("Élément[%d] = %d\n", j, n[j] );
16     }
17
18     return 0;
19 }
```

# Tableaux Multidimensionnelle

## Exercice – 19

```
1  #include <stdio.h>
2
3  int main () {
4
5      /* un tableau avec 5 lignes et 2 colonnes */
6      int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6}, {4,8} };
7      int i, j;
8
9      /* afficher la valeur de chaque élément de tableau */
10     for ( i = 0; i < 5; i++ ) {
11
12         for ( j = 0; j < 2; j++ ) {
13             printf("a[%d][%d] = %d\n", i, j, a[i][j] );
14         }
15     }
16
17     return 0;
18 }
```

a[0][0]: 0  
a[0][1]: 0  
a[1][0]: 1  
a[1][1]: 2  
a[2][0]: 2  
a[2][1]: 4  
a[3][0]: 3  
a[3][1]: 6  
a[4][0]: 4  
a[4][1]: 8



# Passer un tableaux dans un fonctions

## Technique -1

```
/* Paramètres formels en tant que pointeur */  
void maFonction(int *param) {  
    .  
    .  
    .  
}
```

## Technique -2

```
/* Paramètres formels en tant que tableau de taille */  
void maFonction(int param[10]) {  
    .  
    .  
    .  
}
```

## Technique -3

```
/* Paramètres formels en tant que tableau non dimensionné */  
void maFonction(int param[]) {  
    .  
    .  
    .  
}
```

## Exercice – 20

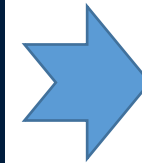
```
double obtenirMoyenne(int arr[], int size) {  
    int i;  
    double avg;  
    double sum = 0;  
  
    for (i = 0; i < size; ++i) {  
        sum += arr[i];  
    }  
    avg = sum / size;  
    return avg;  
}
```

```
#include <stdio.h>  
  
/* function declaration */  
double obtenirMoyenne(int arr[], int taille);  
  
int main () {  
  
    /* un tableau int avec 5 éléments */  
    int balance[5] = {1000, 2, 3, 17, 50};  
    double avg;  
  
    /* passe le pointeur vers le tableau en tant qu'argument */  
    avg = obtenirMoyenne( balance, 5 );  
  
    /* afficher la valeur renvoyée */  
    printf( "La valeur moyenne est: %f ", avg );  
    |  
    return 0;  
}
```

# Retourner un tableaux de fonction

- ne peux pas renvoyer un tableau entier en tant qu'argument à une fonction
- peut renvoyer un pointeur vers un tableau en spécifiant le nom du tableau sans index

```
1  #include <stdio.h>
2
3  /* fonction pour générer et renvoyer des nombres random */
4  int * getRandom( ) {
5
6      static int r[10];
7      int i;
8
9      /* mettre la graine */
10     srand( (unsigned)time( NULL ) );
11
12     for ( i = 0; i < 10; ++i) {
13         r[i] = rand();
14         printf( "r[%d] = %d\n", i, r[i]);
15     }
16
17     return r;
18 }
19
20 /* fonction principale à appeler au-dessus de la fonction définie */
21 int main () {
22
23     /* un pointeur vers un int */
24     int *p;
25     int i;
26
27     p = getRandom();
28
29     for ( i = 0; i < 10; i++ ) {
30         printf( "*(p + %d) : %d\n", i, *(p + i));
31     }
32
33     return 0;
34 }
```



```
r[0] = 313959809
r[1] = 1759055877
r[2] = 1113101911
r[3] = 2133832223
r[4] = 2073354073
r[5] = 167288147
r[6] = 1827471542
r[7] = 834791014
r[8] = 1901409888
r[9] = 1990469526

*(p + 0) : 313959809
*(p + 1) : 1759055877
*(p + 2) : 1113101911
*(p + 3) : 2133832223
*(p + 4) : 2073354073
*(p + 5) : 167288147
*(p + 6) : 1827471542
*(p + 7) : 834791014
*(p + 8) : 1901409888
*(p + 9) : 1990469526
```

Exercice – 21



# Pointeur vers un tableau

```
1  #include <stdio.h>
2
3  int main () {
4
5      /* un tableau avec 5 éléments */
6      double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
7      double *p;
8      int i;
9
10     p = balance;
11
12     /* afficher la valeur de chaque élément de tableau */
13     printf( "Valeurs de tableau à l'aide du pointeur\n" );
14
15     for ( i = 0; i < 5; i++ ) {
16         printf( "(p + %d) : %f\n", i, *(p + i) );
17     }
18
19     printf( "Valeurs de tableau utilisant la balance comme adresse\n" );
20
21     for ( i = 0; i < 5; i++ ) {
22         printf( "(balance + %d) : %f\n", i, *(balance + i) );
23     }
24
25     return 0;
26 }
```

```
Valeurs de tableau à l'aide du pointeur
*(p + 0) : 1000.000000
*(p + 1) : 2.000000
*(p + 2) : 3.400000
*(p + 3) : 17.000000
*(p + 4) : 50.000000
Valeurs de tableau utilisant la balance comme adresse
*(balance + 0) : 1000.000000
*(balance + 1) : 2.000000
*(balance + 2) : 3.400000
*(balance + 3) : 17.000000
*(balance + 4) : 50.000000
```

Exercice – 22

# Les pointeurs, c'est quoi?

## Exercice – 23

```
1  #include <stdio.h>
2
3  int main () {
4
5      int var = 20; /* déclaration de variable réelle */
6      int *ip;      /* déclaration de variable de pointeur */
7
8      ip = &var; /* adresse de stockage de var dans la variable de pointeur */
9
10     printf("Adresse de var variable: %x\n", &var );
11
12     /* adresse stockée dans la variable pointeur */
13     printf("Adresse stockée dans la variable ip: %x\n", ip );
14
15     /* accéder à la valeur en utilisant le pointeur */
16     printf("Valeur de variable *ip: %d\n", *ip );
17
18     return 0;
19 }
```



Adresse de var variable: 12666c84  
Adresse stockée dans la variable ip: 12666c84  
Valeur de variable \*ip: 20

# Tableaux de pointeurs

Est que on a bien compris le  
tableaux ?

## Exercice – 24

```
1 #include <stdio.h>
2
3 const int MAX = 4;
4
5 int main () {
6
7     char *names[] = {
8         "Nicolas",
9         "Sanah",
10        "Marie",
11        "Julie"
12    };
13
14    int i = 0;
15
16    for ( i = 0; i < MAX; i++) {
17        printf("Valeur des noms [%d] = %s\n", i, names[i] );
18    }
19
20    return 0;
21 }
```

```
Valeur des noms [0] = Nicolas
Valeur des noms [1] = Sanah
Valeur des noms [2] = Marie
Valeur des noms [3] = Julie
```

Alors pointer de tableaux

## Exercice – 25 ensuite !!

```
1 #include <stdio.h>
2
3 const int MAX = 3;
4
5 int main () {
6
7     int var[] = {10, 100, 200};
8     int i, *ptr[MAX];
9
10    for ( i = 0; i < MAX; i++) {
11        ptr[i] = &var[i]; /* affecter l'adresse de l'entier. */
12    }
13
14    for ( i = 0; i < MAX; i++) {
15        printf("Valeur de var[%d] = %d\n", i, *ptr[i] );
16    }
17
18    return 0;
19 }
```

```
Valeur de var[0] = 10
Valeur de var[1] = 100
Valeur de var[2] = 200
```

# Envoyer un pointer dans un fonction en C

## Exercice – 26

Passer pointer dans un fonction

```
1  #include <stdio.h>
2  #include <time.h>
3
4  void getSeconds(unsigned long *par);
5
6  int main () {
7
8      unsigned long sec;
9      getSeconds( &sec );
10
11     /* imprime la valeur réelle*/
12     printf("Nombre de secondes: %ld\n", sec );
13
14     return 0;
15 }
16
17 void getSeconds(unsigned long *par) {
18     /* obtenir le nombre actuel de secondes */
19     *par = time( NULL );
20     return;
21 }
```

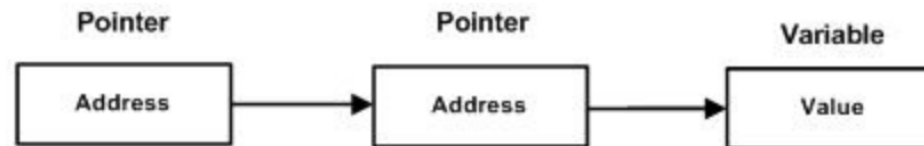
## Exercice – 27

Passer pointer de tableaux dans un fonction

```
1  #include <stdio.h>
2
3  /* déclaration de fonction */
4  double obtenirMoyenne(int *arr, int size);
5
6  int main () {
7
8      /* un tableau int avec 5 éléments */
9      int balance[5] = {1000, 2, 3, 17, 50};
10     double avg;
11
12     /* passe le pointeur vers le tableau en tant qu'argument */
13     avg = obtenirMoyenne( balance, 5 );
14
15     /* afficher la valeur renvoyée */
16     printf("La valeur moyenne est: %f\n", avg );
17     return 0;
18 }
19
20 double obtenirMoyenne(int *arr, int size) {
21
22     int i, sum = 0;
23     double avg;
24
25     for (i = 0; i < size; ++i) {
26         sum += arr[i];
27     }
28
29     avg = (double)sum / size;
30     return avg;
31 }
```

# pointer de pointer

## Exercice – 28



```
1  #include <stdio.h>
2
3  int main () {
4
5      int var;
6      int *ptr;
7      int **pptr;
8
9      var = 3000;
10
11     /* prendre l'adresse de var */
12     ptr = &var;
13
14     /* prendre l'adresse de ptr en utilisant l'opérateur & */
15     pptr = &ptr;
16
17     /* prendre la valeur en utilisant pptr */
18     printf("Valeur de var = %d\n", var );
19     printf("Valeur disponible à *ptr = %d\n", *ptr );
20     printf("Valeur disponible à **pptr = %d\n", **pptr);
21
22     return 0;
23 }
```

Valeur de var = 3000  
Valeur disponible à \*ptr = 3000  
Valeur disponible à \*\*pptr = 3000

## Exercice – 29

### Accéder le Structures

# Structures en C

## Exercice – 30

### Structures comme les argument de fonctions

```
1 #include <stdio.h>
2 #include <string.h>
3
4 struct Livres {
5     char titre[50];
6     char auteur[50];
7     char sujet[100];
8     int livre_id;
9 };
10
11 int main( ) {
12
13     struct Livres Libre1;      /* Déclare Book1 de type Livre */
14     struct Livres Libre2;      /* Déclare Book2 de type Livre */
15
16     /* book 1 specification */
17     strcpy( Libre1.titre, "Programmation C");
18     strcpy( Libre1.auteur, "Nuha Ali");
19     strcpy( Libre1.sujet, "Tutoriel de programmation C");
20     Libre1.livre_id = 6495407;
21
22     /* book 2 specification */
23     strcpy( Libre2.titre, "Facturation télécom");
24     strcpy( Libre2.auteur, "Zara Ali");
25     strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
26     Libre2.livre_id = 6495700;
27
28     /* print Book1 info */
29     printf( "Book 1 titre : %s\n", Libre1.titre);
30     printf( "Book 1 auteur : %s\n", Libre1.auteur);
31     printf( "Book 1 sujet : %s\n", Libre1.sujet);
32     printf( "Book 1 livre_id : %d\n", Libre1.livre_id);
33
34     /* print Book2 info */
35     printf( "Book 2 titre : %s\n", Libre2.titre);
36     printf( "Book 2 auteur : %s\n", Libre2.auteur);
37     printf( "Book 2 sujet : %s\n", Libre2.sujet);
38     printf( "Book 2 livre_id : %d\n", Libre2.livre_id);
39
40     return 0;
41 }
```

```
Libre titre : Programmation C
Libre auteur : Nuha Ali
Libre sujet : Tutoriel de programmation C
Libre livre_id : 6495407
Libre titre : Facturation télécom
Libre auteur : Zara Ali
Libre sujet : Didacticiel de facturation télécom
Libre livre_id : 6495700
```

```
1 #include <stdio.h>
2 #include <string.h>
3
4 struct Livres {
5     char titre[50];
6     char auteur[50];
7     char sujet[100];
8     int livre_id;
9 };
10
11 /* function declaration */
12 void imprimerLibre( struct Livres livre );
13
14 int main( ) {
15
16     struct Livres Libre1;      /* Déclare Book1 de type Livre */
17     struct Livres Libre2;      /* Déclare Book2 de type Livre */
18
19     /* book 1 specification */
20     strcpy( Libre1.titre, "Programmation C");
21     strcpy( Libre1.auteur, "Nuha Ali");
22     strcpy( Libre1.sujet, "Tutoriel de programmation C");
23     Libre1.livre_id = 6495407;
24
25     /* book 2 specification */
26     strcpy( Libre2.titre, "Facturation télécom");
27     strcpy( Libre2.auteur, "Zara Ali");
28     strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
29     Libre2.livre_id = 6495700;
30
31     /* print Book1 info */
32     imprimerLibre( Libre1 );
33
34     /* Print Book2 info */
35     imprimerLibre( Libre2 );
36
37     return 0;
38 }
39
40 void imprimerLibre( struct Livres livre ) {
41
42     printf( "Libre titre : %s\n", livre.titre);
43     printf( "Libre auteur : %s\n", livre.auteur);
44     printf( "Libre sujet : %s\n", livre.sujet);
45     printf( "Libre livre_id : %d\n", livre.livre_id);
46 }
```

## Exercice – 31

### Pointers de Structure

## Structures en C

```
1 #include <stdio.h>
2 #include <string.h>
3
4 struct Livres {
5     char titre[50];
6     char auteur[50];
7     char sujet[100];
8     int livre_id;
9 };
10
11 /* function declaration */
12 void imprimerLibre( struct Livres *libre );
13
14 int main( ) {
15
16     struct Livres Libre1;      /* Déclare Book1 de type Livre */
17     struct Livres Libre2;      /* Déclare Book2 de type Livre */
18
19     /* book 1 specification */
20     strcpy( Libre1.titre, "Programmation C");
21     strcpy( Libre1.auteur, "Nuha Ali");
22     strcpy( Libre1.sujet, "Tutoriel de programmation C");
23     Libre1.livre_id = 6495407;
24
25     /* book 2 specification */
26     strcpy( Libre2.titre, "Facturation télécom");
27     strcpy( Libre2.auteur, "Zara Ali");
28     strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
29     Libre2.livre_id = 6495700;
30
31     /* print Book1 info */
32     imprimerLibre( &Libre1 );
33
34     /* Print Book2 info */
35     imprimerLibre( &Libre2 );
36
37     return 0;
38 }
39
40 void imprimerLibre( struct Livres *libre ) {
41
42     printf( "Libre titre : %s\n", libre->titre);
43     printf( "Libre auteur : %s\n", libre->auteur);
44     printf( "Libre sujet : %s\n", libre->sujet);
45     printf( "Libre livre_id : %d\n", libre->livre_id);
46 }
```

```
Libre titre : Programmation C
Libre auteur : Nuha Ali
Libre sujet : Tutoriel de programmation C
Libre livre_id : 6495407
Libre titre : Facturation télécom
Libre auteur : Zara Ali
Libre sujet : Didacticiel de facturation télécom
Libre livre_id : 6495700
```