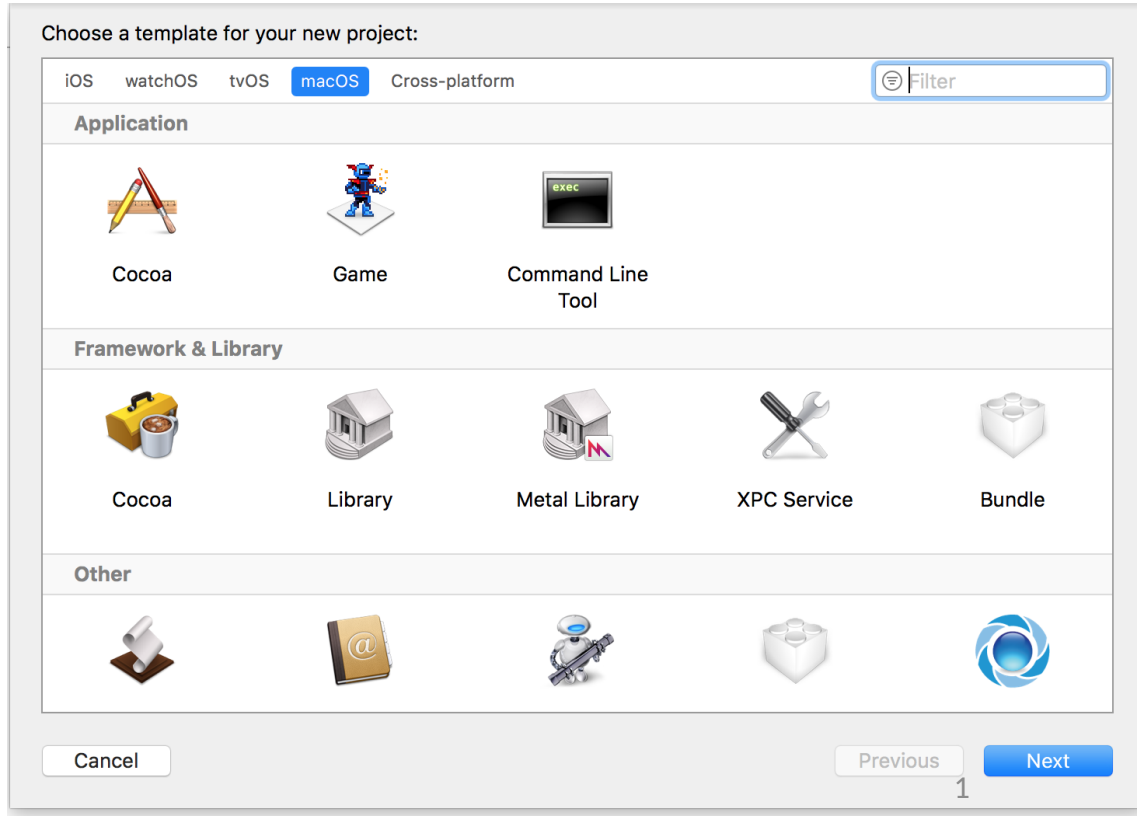
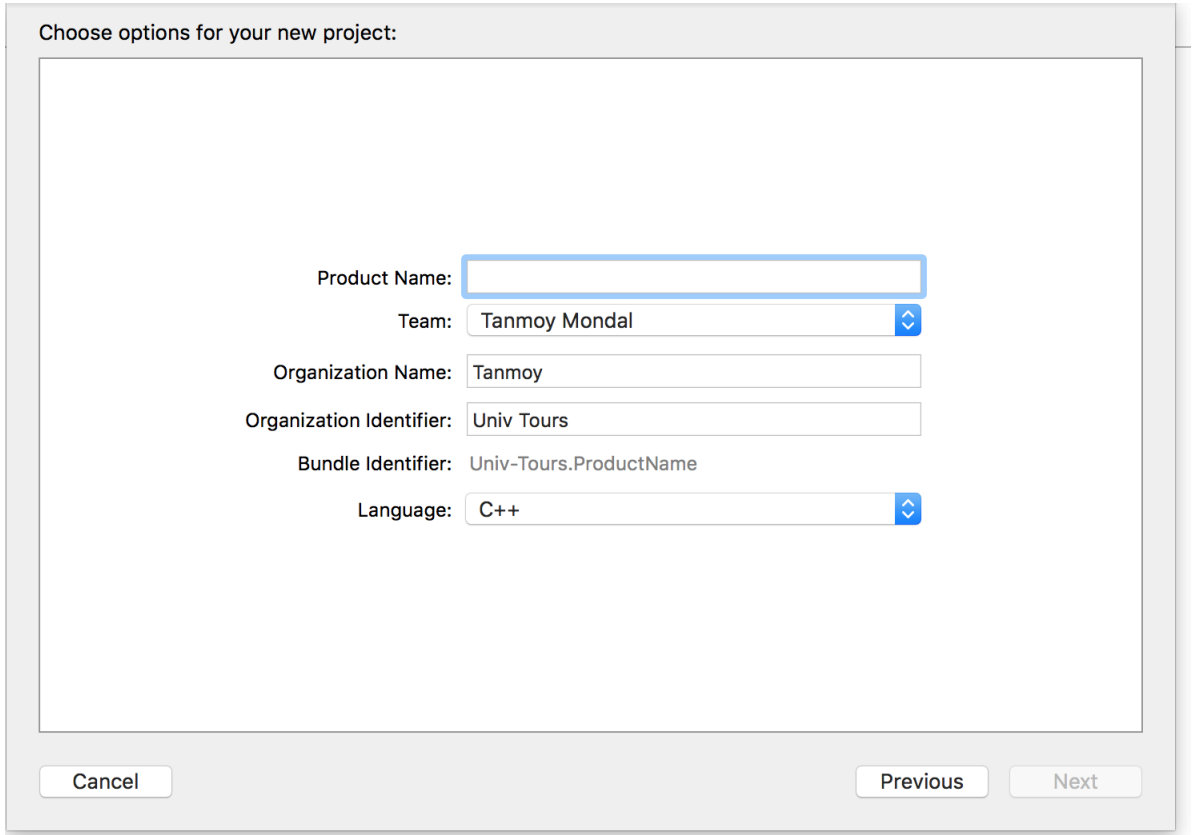
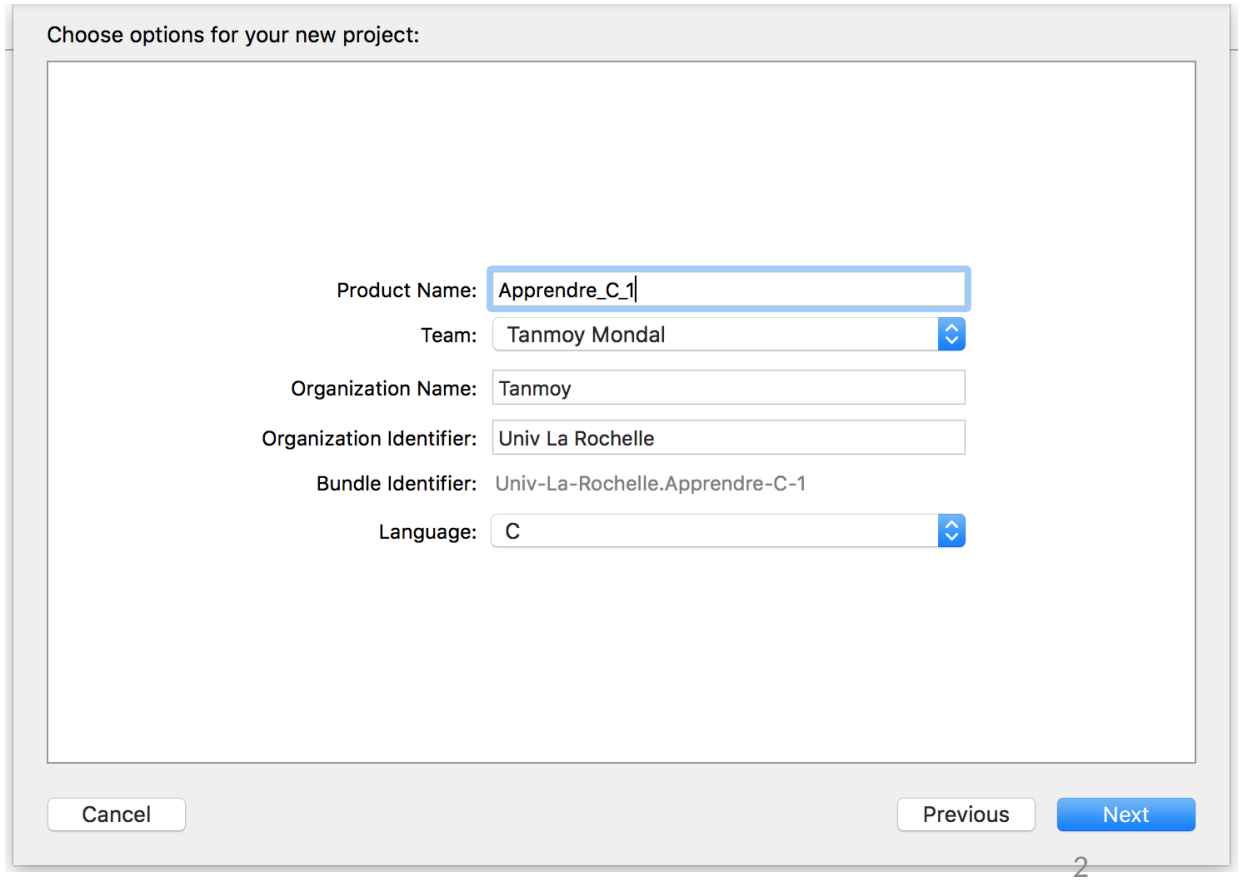
- The first line of the program *#include <stdio.h>* is a preprocessor







Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Première Programme en C

```c
#include <stdio.h>

int main() {
   /* Ma première programme en C*/
   printf("Bonjour le monde! \n");

   return 0;
}
```

Types de base

4 types de base, les autres types seront dérivés de ceux-ci.

Type	Signification	Codage en mémoire	Peut être
char	Caractère unique	1 octet	signed, unsigned
int	Nombre entier	2 ou 4 octets	Short, long, signed, unsigned
float	Nombre réel simple	4 octets	
double	Nombre réel double précision	8 octets	long

Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Préprocesseur

- Le préprocesseur effectue un prétraitement du programme source avant qu'il soit compilé
- Ce préprocesseur exécute des instructions particulières appelées **directives**
- Ces directives sont identifiées par le caractère **#** en tête

Inclusion de fichiers

```
#include <nom-de-fichier>      /* répertoire  standard */
#include "nom-de-fichier"      /* répertoire  courant */
```

La gestion des fichiers (**stdio.h**) /* Entrees-sorties standard */

Les fonctions mathématiques (**math.h**)

Taille des type entiers (**limits.h**)

Limites des type réels (**float.h**)

Traitement de chaînes de caractères (**string.h**)

Le traitement de caractères (**ctype.h**)

Utilitaires généraux (**stdlib.h**)

Date et heure (**time.h**)

Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Définir constant

Avec # define preprocesseur

```c
#include <stdio.h>

#define LONGUEUR 10
#define LARGEUR  5
#define NOUVELLE_LIGNE '\n'

int main() {

   int region;

   region = LONGUEUR * LARGEUR;
   printf("valeur de la région : %d", region);
   printf("%c", NOUVELLE_LIGNE);

   return 0;
}
```

Avec const motsclé

```c
#include <stdio.h>

int main() {

   const int  LONGUEUR = 10;
   const int  LARGEUR = 5;
   const char NOUVELLE_LIGNE = '\n';
   int region;

   region = LONGUEUR * LARGEUR;
   printf("valeur de la région : %d", region);
   printf("%c", NOUVELLE_LIGNE);

   return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Les structures de contrôle en C

Alternative: **`if-else`**

Choix Multiple: **`switch-case`**

Itérations: **`for, while, do-while`**

Rupture de Contrôle: **`break, continue, return`** … **`goto`**

Les structures de contrôle en C

Les décisions - if then else

Pas de then en C

Le bloc " **else** " est optionnel.

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* vérifier la condition booléenne en utilisant if */

   if( a < 20 ) {
      /* si la condition est vraie, imprimez le texte suivant */
      printf("a est inférieur à 20\n" );
   }

   printf("La valeur de a est: %d\n", a);

   return 0;
}
```

*** Tout ce qui est 0 est faux**
*** Tout ce qui est != de 0 est vrai**

```c
if(1)
    printf("ceci sera toujours affiche\n");
if(0)
    printf("ceci ne sera jamais affiche\n");
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Les structures de contrôle en C

Les décisions – "if" then "else if"

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 100;

   /* vérifier la condition booléenne */
   if( a == 10 ) {
      /* si la condition est vraie, imprimez le texte suivant */
      printf("La valeur de a est 10\n" );
   } else if( a == 20 ) {
      /* si autre si la condition est vraie */
      printf("La valeur de a est 20\n" );
   } else if( a == 30 ) {
      /* si autre si la condition est vraie */
      printf("La valeur de a est 30\n" );
   } else {
      /* si aucune des conditions n'est vraie */
      printf("Aucune des valeurs ne correspond\n" );
   }

   printf("La valeur exacte d'un est: %d\n", a );

   return 0;
}
```

Imbriqué « if »

```c
#include <stdio.h>

int main () {

   /* définition de variable locale*/
   int a = 100;
   int b = 200;

   /* vérifier la condition booléenne */
   if( a == 100 ) {

      /* Si la condition est vraie, vérifiez les points suivants */
      if( b == 200 ) {
         /* si la condition est vraie, imprimez le texte suivant */
         printf("La valeur de a est 100 et b est 200\n" );
      }
   }

   printf("La valeur exacte d'un est : %d\n", a );
   printf("La valeur exacte de b est : %d\n", b );

   return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Les structures de contrôle en C

Les décisions – “switch”

```c
#include <stdio.h>

int main () {

   /* définition de variable locale*/
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
         printf("Trés Bien!\n" );
         break;
      case 'C' :
         printf("Bien joué\n" );
         break;
      case 'D' :
         printf("Tu es passé\n" );
         break;
      case 'F' :
         printf("Mieux vaut réessayer\n" );
         break;
      default :
         printf("Grade invalide\n" );
   }

   printf("Votre note est  %c\n", grade );

   return 0;
}
```

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;

   switch(a) {

      case 300:
         printf("This is part of outer switch\n");
         break;
      case 400:
         printf("This is part of outer switch\n");
         break;
      case 100:
         printf("This is part of outer switch\n");
         break;
         switch(b) {
            case 200:
               printf("This is part of inner switch\n");
               break;
            case 2000:
               printf("This is part of inner switch\n");
               break;
            case 20000:
               printf("This is part of inner switch\n");
               break;
         }
      case 500:
         printf("This is part of outer switch\n");
         break;
      default:
        printf("This is part of dafult switch\n");

   }

   printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );

   return 0;
}
```

Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Les itérations

– While

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* pendant l'exécution de la boucle*/
   while( a < 20 ) {
      printf("valeur de a: %d\n", a);
      a++;
   }

   return 0;
}
```

– for

```c
#include <stdio.h>

int main () {

   int a;

   /* pour l'exécution de la boucle */
   for( a = 10; a < 20; a = a + 1 ){
      printf("valeur de a: %d\n", a);
   }

   return 0;
}
```

– do while

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* faire une boucle d'exécution */
   do {
      printf("valeur de a: %d\n", a);
      a = a + 1;
   }while( a < 20 );

   return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Imbriqué for

```c
/**
 * Programme C pour imprimer la table de multiplication de 1 à 5
 */
#include <stdio.h>

int main()
{
    /* Déclaration de variable de compteur de boucle */
    int i, j;

    /* Boucle extérieure*/
    for(i=1; i<=10; i++)
    {
        /* Boucle intérieure*/
        for(j=1; j<=5; j++)
        {
            printf("%d\t", (i*j));
        }

        /* Imprimer une nouvelle ligne */
        printf("\n");
    }

    return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Sortir de boucle

Break

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* l'exécution de la boucle while */
   while( a < 20 ) {

      printf("valeur de a: %d\n", a);
      a++;

      if( a > 15) {
         /* terminer la boucle en utilisant l'instruction break */
         break;
      }
   }
   return 0;
}
```

Continue

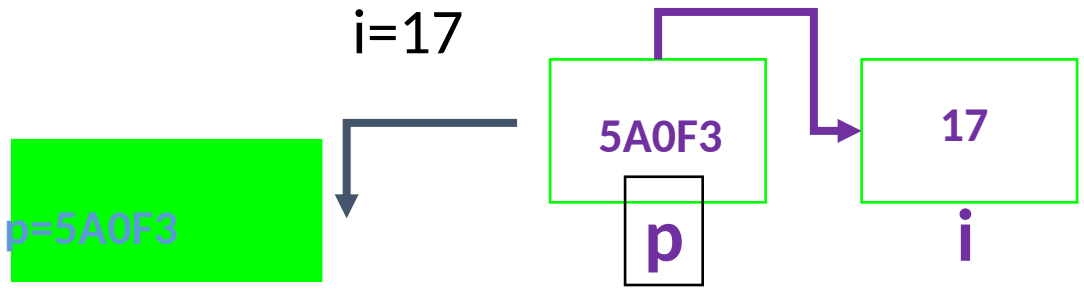
```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* faire une boucle d'exécution*/
   do {

      if( a == 15) {
         /* ignorer l'itération */
         a = a + 1;
         continue;
      }

      printf("valuer de a: %d\n", a);
      a++;

   } while( a < 20 );

   return 0;
}
```

goto

```c
#include <stdio.h>

int main () {

   /* définition de variable locale */
   int a = 10;

   /* faire une boucle d'exécution */
   LOOP:do {

      if( a == 15) {
         /* ignorer l'itération*/
         a = a + 1;
         goto LOOP;
      }

      printf("valeur de a: %d\n", a);
      a++;

   }while( a < 20 );

   return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Les pointeurs, c'est quoi?




```
1  int    *ip;    /* pointeur vers un integer */
2  double *dp;    /* pointeur vers un double */
3  float  *fp;    /* pointeur vers un float */
4  char   *ch     /* pointeur vers un character */
```

```
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 }
```

Déclaration de Pointeurs

Le symbole * est utilisé entre le type et le nom du pointeur

- Déclaration d'un entier: `int i;`
- Déclaration d'un pointeur vers un entier: `int *p;`

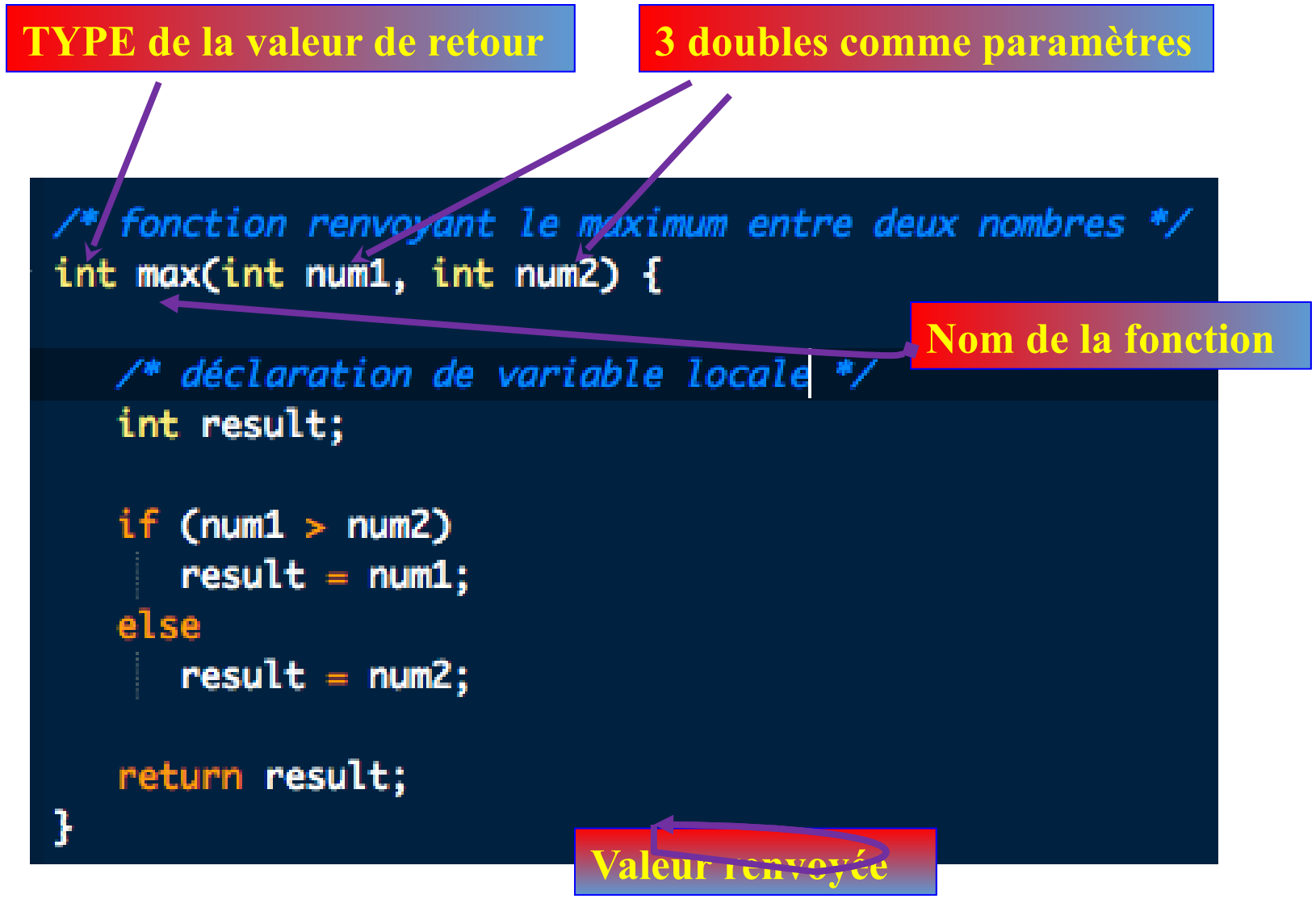
Exemples de déclarations de pointeurs

```
int *pi;          /*  pi est un pointeur vers un int
                      *pi désigne le contenu de l'adresse */
float *pf;        /* pf est un pointeur vers un float */

char c, d, *pc;   /* c et d sont des char*/
                  /* pc est un pointeur vers un char */
double *pd, e, f; /* pd est un pointeur vers un double*/
                  /* e et f sont des doubles */
double **tab;     /* tab est un pointeur pointant sur un pointeur qui
                                  pointe sur un flottant double */
```

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


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Déclarer une fonction



Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Définition et Déclaration de une fonction

```c
#include <stdio.h>

/* déclaration de fonction*/
int max(int num1, int num2);

int main () {

   /* définition de variable locale */
   int a = 100;
   int b = 200;
   int ret;

   /* appeler une fonction pour obtenir la valeur maximale */
   ret = max(a, b);

   printf( "La valeur maximale est: %d\n", ret );

   return 0;
}

/* fonction renvoyant le maximum entre deux nombres*/
int max(int num1, int num2) {

   /* déclaration de variable locale */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Appeler une fonction

Appeler par valeur

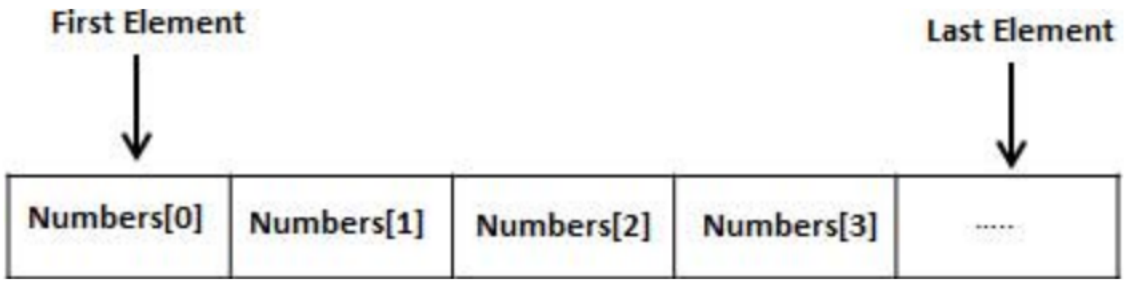
```c
#include <stdio.h>

/* déclaration de fonction*/
void swap(int x, int y);

int main () {

   /* définition de variable locale */
   int a = 100;
   int b = 200;

   printf("Avant l'échange, valeur de a : %d\n", a );
   printf("Avant l'échange, valeur de b : %d\n", b );

   /* appeler une fonction pour échanger les valeurs */
   swap(a, b);

   printf("Après échange, valeur de a : %d\n", a );
   printf("Après échange, valeur de b : %d\n", b );

   return 0;
}

/* définition de fonction pour échanger les valeurs */
void swap(int x, int y) {

   int temp;

   temp = x; /* enregistrer la valeur de x */
   x = y;    /* mettre y en x */
   y = temp; /* mettre temp en y */

   return;
}
```

```
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
```

Appeler par référence

```c
#include <stdio.h>

/* déclaration de fonction*/
void swap(int *x, int *y);

int main () {

   /* définition de variable locale */
   int a = 100;
   int b = 200;

   printf("Avant l'échange, valeur de a : %d\n", a );
   printf("Avant l'échange, valeur de b : %d\n", b );

   /* appeler une fonction pour échanger les valeurs.
      * &a indique un pointeur vers a i.e. adresse de la variable a et
      * &b indique un pointeur vers b i.e. adresse de la variable b
   */
   swap(&a, &b);

   printf("Après échange, valeur de a : %d\n", a );
   printf("Après échange, valeur de b : %d\n", b );

   return 0;
}

/* définition de fonction pour échanger les valeurs */
void swap(int *x, int *y) {

   int temp;
   temp = *x;    /* enregistrer la valeur à l'adresse x */
   *x = *y;      /* mettre y en x */
   *y = temp;    /* mettre temp en y */

   return;
}
```

Tableaux




```
double balance[10];
```

ou

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

ou

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

```c
#include <stdio.h>

int main () {

   int n[ 10 ]; /* n est un tableau de 10 nombres entiers */
   int i,j;

   /* initialise les éléments du tableau n à 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 100; /* définir l'élément à l'emplacement i à i + 100 */
   }

   /* afficher la valeur de chaque élément de tableau */
   for (j = 0; j < 10; j++ ) {
      printf("Élément[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

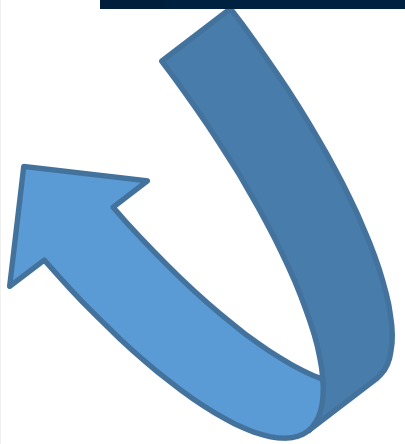
Tableaux Multidimensionnelle

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

```c
int a[3][4] = {
    {0, 1, 2, 3} ,   /*  initialiseurs pour l'index de ligne par 0 */
    {4, 5, 6, 7} ,   /*  initialiseurs pour l'index de ligne par 1 */
    {8, 9, 10, 11}   /*  initialiseurs pour l'index de ligne par 2 */
}
```

```c
#include <stdio.h>

int main () {

   /* un tableau avec 5 lignes et 2 colonnes*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* afficher la valeur de chaque élément de tableau */
   for ( i = 0; i < 5; i++ ) {

      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }

   return 0;
}
```

```
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
```



Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

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[]) {
   .
   .
   .
}
```

```
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;
}
```

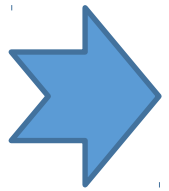
```
#include <stdio.h>

/* 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;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

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

```c
#include <stdio.h>

/* fonction pour générer et renvoyer des nombres random */
int * getRandom( ) {

   static int  r[10];
   int i;

   /* mettre la graine */
   srand( (unsigned)time( NULL ) );

   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf( "r[%d] = %d\n", i, r[i]);
   }

   return r;
}

/* fonction principale à appeler au-dessus de la fonction définie */
int main () {

   /* un pointeur vers un int */
   int *p;
   int i;

   p = getRandom();

   for ( i = 0; i < 10; i++ ) {
      printf( "*(p + %d) : %d\n", i, *(p + i));
   }

   return 0;
}
```



```
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
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Pointeur vers un tableau

```c
#include <stdio.h>

int main () {

   /* un tableau avec 5 éléments */
   double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
   double *p;
   int i;

   p = balance;

   /* afficher la valeur de chaque élément de tableau */
   printf( "Valeurs de tableau à l'aide du pointeur\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(p + %d) : %f\n",  i, *(p + i) );
   }

   printf( "Valeurs de tableau utilisant la balance comme adresse\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(balance + %d) : %f\n",  i, *(balance + i) );
   }

   return 0;
}
```

```
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
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Tableaux de pointers

Est que on a bien compris le tableaux ?

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int i;

   for (i = 0; i < MAX; i++) {
      printf("Valeur de var[%d] = %d\n", i, var[i] );
   }

   return 0;
}
```

Alors pointer de tableaux ensuite !!

```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int i, *ptr[MAX];

   for ( i = 0; i < MAX; i++) {
      ptr[i] = &var[i]; /* affecter l'adresse de l'entier. */
   }

   for ( i = 0; i < MAX; i++) {
      printf("Valeur de var[%d] = %d\n", i, *ptr[i] );
   }

   return 0;
}
```

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


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Tableaux de pointers

Est que on a bien compris le tableaux ?

```c
#include <stdio.h>

const int MAX = 4;

int main () {

   char *names[] = {
      "Nicolas",
      "Sanah",
      "Marie",
      "Julie"
   };

   int i = 0;

   for ( i = 0; i < MAX; i++) {
      printf("Valeur des noms [%d] = %s\n", i, names[i] );
   }

   return 0;
}
```

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


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Envoyer un pointer dans un fonction en C

Passer pointer dans un fonction

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

void getSeconds(unsigned long *par);

int main () {

   unsigned long sec;
   getSeconds( &sec );

   /* imprime la valeur réelle*/
   printf("Nombre de secondes: %ld\n", sec );

   return 0;
}

void getSeconds(unsigned long *par) {
   /* obtenir le nombre actuel de secondes */
   *par = time( NULL );
   return;
}
```

Passer pointer de tableaux dans un fonction

```c
#include <stdio.h>

/* déclaration de fonction */
double obtenirMoyenne(int *arr, int size);

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\n", avg );
   return 0;
}

double obtenirMoyenne(int *arr, int size) {

   int  i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = (double)sum / size;
   return avg;
}
```


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

pointer de pointer


```c
#include <stdio.h>

int main () {

   int  var;
   int  *ptr;
   int  **pptr;

   var = 3000;

   /* prendre l'adresse de var */
   ptr = &var;

   /* prendre l'adresse de ptr en utilisant l'adresse de l'opérateur & */
   pptr = &ptr;

   /* prendre la valeur en utilisant pptr */
   printf("Valeur de var = %d\n", var );
   printf("Valeur disponible à *ptr = %d\n", *ptr );
   printf("Valeur disponible à **pptr = %d\n", **pptr);

   return 0;
}
```

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


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/

Structures en C

Accéder le Structures

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

struct Livres {
   char  titre[50];
   char  auteur[50];
   char  sujet[100];
   int   livre_id;
};

int main( ) {

   struct Livres Libre1;        /* Déclare Book1 de type Livre */
   struct Livres Libre2;        /* Déclare Book2 de type Livre */

   /* book 1 specification */
   strcpy( Libre1.titre, "Programmation C");
   strcpy( Libre1.auteur, "Nuha Ali");
   strcpy( Libre1.sujet, "Tutoriel de programmation C");
   Libre1.livre_id = 6495407;

   /* book 2 specification */
   strcpy( Libre2.titre, "Facturation télécom");
   strcpy( Libre2.auteur, "Zara Ali");
   strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
   Libre2.livre_id = 6495700;

   /* print Book1 info */
   printf( "Book 1 titre : %s\n", Libre1.titre);
   printf( "Book 1 auteur : %s\n", Libre1.auteur);
   printf( "Book 1 sujet : %s\n", Libre1.sujet);
   printf( "Book 1 livre_id : %d\n", Libre1.livre_id);

   /* print Book2 info */
   printf( "Book 2 titre : %s\n", Libre2.titre);
   printf( "Book 2 auteur : %s\n", Libre2.auteur);
   printf( "Book 2 sujet : %s\n", Libre2.sujet);
   printf( "Book 2 livre_id : %d\n", Libre2.livre_id);

   return 0;
}
```

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

Structures comme les argument de fonctions

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

struct Livres {
   char  titre[50];
   char  auteur[50];
   char  sujet[100];
   int   livre_id;
};

/* function declaration */
void imprimerLibre( struct Livres libre );

int main( ) {

   struct Livres Libre1;        /* Déclare Book1 de type Livre */
   struct Livres Libre2;        /* Déclare Book2 de type Livre */

   /* book 1 specification */
   strcpy( Libre1.titre, "Programmation C");
   strcpy( Libre1.auteur, "Nuha Ali");
   strcpy( Libre1.sujet, "Tutoriel de programmation C");
   Libre1.livre_id = 6495407;

   /* book 2 specification */
   strcpy( Libre2.titre, "Facturation télécom");
   strcpy( Libre2.auteur, "Zara Ali");
   strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
   Libre2.livre_id = 6495700;

  /* print Book1 info */
   imprimerLibre( Libre1 );

   /* Print Book2 info */
   imprimerLibre( Libre2 );

   return 0;
}

void imprimerLibre( struct Livres libre ) {

   printf( "Libre titre : %s\n", libre.titre);
   printf( "Libre auteur : %s\n", libre.auteur);
   printf( "Libre subjet : %s\n", libre.sujet);
   printf( "Libre livre_id : %d\n", libre.livre_id);
}
```

Structures en C

Pointers de Structure

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

struct Livres {
   char  titre[50];
   char  auteur[50];
   char  sujet[100];
   int   livre_id;
};

/* function declaration */
void imprimerLibre( struct Livres *libre );

int main( ) {

   struct Livres Libre1;        /* Déclare Book1 de type Livre */
   struct Livres Libre2;        /* Déclare Book2 de type Livre */

   /* book 1 specification */
   strcpy( Libre1.titre, "Programmation C");
   strcpy( Libre1.auteur, "Nuha Ali");
   strcpy( Libre1.sujet, "Tutoriel de programmation C");
   Libre1.livre_id = 6495407;

   /* book 2 specification */
   strcpy( Libre2.titre, "Facturation télécom");
   strcpy( Libre2.auteur, "Zara Ali");
   strcpy( Libre2.sujet, "Didacticiel de facturation télécom");
   Libre2.livre_id = 6495700;

  /* print Book1 info */
   imprimerLibre( &Libre1 );

   /* Print Book2 info */
   imprimerLibre( &Libre2 );

   return 0;
}

void imprimerLibre( struct Livres *libre ) {

   printf( "Libre titre : %s\n", libre->titre);
   printf( "Libre auteur : %s\n", libre->auteur);
   printf( "Libre subjet : %s\n", libre->sujet);
   printf( "Libre livre_id : %d\n", libre->livre_id);
}
```

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


Le matériel pédagogique est tiré de TutorialPoint: https://www.tutorialspoint.com/cprogramming/