

Choose options for your new project:

Product Name:

Team: Tanmoy Mondal

Organization Name: Tanmoy

Organization Identifier: Univ Tours

Bundle Identifier: Univ-Tours.ProductName

Language: C++

[Cancel](#) [Previous](#) [Next](#)

Choose options for your new project:

Product Name: Apprendre\_C++

Team: Tanmoy Mondal

Organization Name: Tanmoy

Organization Identifier: Univ La Rochelle

Bundle Identifier: Univ-La-Rochelle.Apprendre-C--

Language: C++

[Cancel](#) [Previous](#) [Next](#)

# Première Programme en C++

## Exercice – 1\_1

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Bonjour Le Monde";
6     return 0;
7 }
```

## Exercice – 1\_2

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Taille de char : " << sizeof(char) << endl;
6     cout << "Taille de int : " << sizeof(int) << endl;
7     cout << "Taille de short int : " << sizeof(short int) << endl;
8     cout << "Taille de long int : " << sizeof(long int) << endl;
9     cout << "Taille de float : " << sizeof(float) << endl;
10    cout << "Taille de double : " << sizeof(double) << endl;
11    cout << "Taille de wchar_t : " << sizeof(wchar_t) << endl;
12
13    return 0;
14 }
```

```
Taille de char : 1
Taille de int : 4
Taille de short int : 2
Taille de long int : 8
Taille de float : 4
Taille de double : 8
Taille de wchar_t : 4
```

# Définir constant

## Exercice – 1\_2

Avec **# define** préprocesseur

```
1 #include <iostream>
2 using namespace std;
3
4 #define LONGUEUR 10
5 #define LARGEUR 5
6 #define NOUVELLE_LIGNE '\n'
7
8 int main() {
9     int region;
10
11     region = LONGUEUR * LARGEUR;
12     cout << "valeur de la région :" << region;
13     cout << NOUVELLE_LIGNE;
14     return 0;
15 }
```

## Exercice – 1\_3

Avec **const** motsclé

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
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    cout << "valeur de la région :" << region;
12    cout << NOUVELLE_LIGNE;
13    return 0;
14 }
```

## Exercice – 2\_1

```
1 // i/o example
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
8     int i;
9     cout << "Please enter an integer value: ";
10    cin >> i;
11    cout << "The value you entered is " << i;
12    cout << " and its double is " << i*2 << ".\n";
13    return 0;
14 }
```

## Exercice – 2\_2

```
1 // cin with strings
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main ()
7 {
8     string mystr;
9     cout << "What's your name? ";
10    getline (cin, mystr);
11    cout << "Hello " << mystr << ".\n";
12    cout << "What is your favorite team? ";
13    getline (cin, mystr);
14    cout << "I like " << mystr << " too!\n";
15    return 0;
16 }
```

## Exercice – 2\_3

```
1 // echo machine
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main ()
7 {
8     string str;
9     do {
10        cout << "Enter text: ";
11        getline (cin,str);
12        cout << "You entered: " << str << '\n';
13    } while (str != "goodbye");
14 }
```

# Eléments à dans fichier .hpp et .cpp

## Fichier .hpp

- Tout les # define
- Toutes les variables globales
- Déclaration de classe

```
#include <iostream>
using namespace std;

class Box {
public:
    double length;           // Length of a box
    double breadth;          // Breadth of a box
    double height;           // Height of a box

    // Member functions declaration
    double getVolume(void);
    void setLength( double len );
    void setBreadth( double bre );
    void setHeight( double hei );
};


```

## Fichier .cpp

- Mettre the code actuel
  - Le fonctions main ()
    - Tous les choses on écrire dans main()
- Mettre la définition de fonctions

```
// Member functions definitions
double Box::getVolume(void) {
    return length * breadth * height;
}

void Box::setLength( double len ) {
    length = len;
}
void Box::setBreadth( double bre ) {
    breadth = bre;
}
void Box::setHeight( double hei ) {
    height = hei;
}

// Main function for the program
int main() {
    Box Box1;                      // Declare Box1 of type Box
    Box Box2;                      // Declare Box2 of type Box
    double volume = 0.0;            // Store the volume of a box here

    // box 1 specification
    Box1.setLength(6.0);
    Box1.setBreadth(7.0);
    Box1.setHeight(5.0);

    // box 2 specification
    Box2.setLength(12.0);
    Box2.setBreadth(13.0);
    Box2.setHeight(10.0);

    // volume of box 1
    volume = Box1.getVolume();
    cout << "Volume of Box1 : " << volume << endl;

    // volume of box 2
    volume = Box2.getVolume();
    cout << "Volume of Box2 : " << volume << endl;
    return 0;
}
```

- Essayer à chercher sur le internet des exemples pour mieux comprendre ces quoi à mettre dans quel fichier (.cpp ou .hpp)

# Class et Objet en C++

## Exercice – 2\_4

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Boitre {
6     public:
7         double longeur; // Length of a box
8         double largeur; // Breadth of a box
9         double hauteur; // Height of a box
10 }
11
12 int main() {
13     Boitre Boit1; // Declare Box1 of type Box
14     Boitre Boit2; // Declare Box2 of type Box
15     double volume = 0.0; // Store the volume of a box here
16
17     // box 1 specification
18     Boit1.longeur = 5.0;
19     Boit1.largeur = 6.0;
20     Boit1.hauteur = 7.0;
21
22     // box 2 specification
23     Boit2.longeur = 10.0;
24     Boit2.largeur = 12.0;
25     Boit2.hauteur = 13.0;
26
27     // volume of box 1
28     volume = Boit1.longeur * Boit1.largeur * Boit1.hauteur;
29     cout << "Volume de Boit1 : " << volume << endl;
30
31     // volume of box 2
32     volume = Boit2.longeur * Boit2.largeur * Boit2.hauteur;
33     cout << "Volume de Boit2 : " << volume << endl;
34
35 }
```

## Exercice – 2\_5 Variables et Fonctions dans class

```
#include <iostream>
using namespace std;

class Box {
public:
    double length; // Length of a box
    double breadth; // Breadth of a box
    double height; // Height of a box

    // Member functions declaration
    double getVolume(void);
    void setLength( double len );
    void setBreadth( double bre );
    void setHeight( double hei );
};

// Member functions definitions
double Box::getVolume(void) {
    return length * breadth * height;
}

void Box::setLength( double len ) {
    length = len;
}
void Box::setBreadth( double bre ) {
    breadth = bre;
}
void Box::setHeight( double hei ) {
    height = hei;
}

// Main function for the program
int main() {
    Box Box1; // Declare Box1 of type Box
    Box Box2; // Declare Box2 of type Box
    double volume = 0.0; // Store the volume of a box here

    // box 1 specification
    Box1.setLength(6.0);
    Box1.setBreadth(7.0);
    Box1.setHeight(5.0);

    // box 2 specification
    Box2.setLength(12.0);
    Box2.setBreadth(13.0);
    Box2.setHeight(10.0);

    // volume of box 1
    volume = Box1.getVolume();
    cout << "Volume of Box1 : " << volume << endl;

    // volume of box 2
    volume = Box2.getVolume();
    cout << "Volume of Box2 : " << volume << endl;
    return 0;
}
```

# Héritage

## Exercice – 2\_6 Héritage Simple

```
#include <iostream>
using namespace std;

// Base class
class Shape {
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }

protected:
    int width;
    int height;
};

// Derived class
class Rectangle: public Shape {
public:
    int getArea() {
        return (width * height);
    }
};

int main(void) {
    Rectangle Rect;

    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    return 0;
}
```

Try it

```
#include <iostream>
using namespace std;

// Base class Shape
class Shape {
public:
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }

protected:
    int width;
    int height;
};

// Base class PaintCost
class PaintCost {
public:
    int getCost(int area) {
        return area * 70;
    }
};

// Derived class
class Rectangle: public Shape, public PaintCost {
public:
    int getArea() {
        return (width * height);
    }
};

int main(void) {
    Rectangle Rect;
    int area;

    Rect.setWidth(5);
    Rect.setHeight(7);

    area = Rect.getArea();

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() << endl;

    // Print the total cost of painting
    cout << "Total paint cost: $" << Rect.getCost(area) << endl;

    return 0;
}
```

Try it

## Exercice – 2\_7 Héritage Multiple

Pour tous les diapositives suivantes, utiliser les syntaxes de c++ à la place de C pour faire les mêmes programmes

# 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 }
```

## Exercice – 6

```
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("Très 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 }
```

## Les structures de contrôle en C Les décisions – “switch”

## 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
39 printf("Exact value of a is : %d\n", a );
40 printf("Exact value of b is : %d\n", b );
41
42 return 0;
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
5     /* définition de variable locale */
6     int a = 10;
7
8     /* l'exécution de la boucle while */
9     while( a < 20 ) {
10
11         printf("valeur de a: %d\n", a);
12         a++;
13
14         if( a > 15) {
15             /* terminer la boucle en utilisant l'instruction break */
16             break;
17         }
18     }
19     return 0;
20 }
```

## Sortir de boucle Exercice – 13 Continue

```
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
11         if( a == 15) {
12             /* ignorer l'itération */
13             a = a + 1;
14             continue;
15         }
16
17         printf("valuer de a: %d\n", a);
18         a++;
19
20     } while( a < 20 );
21
22     return 0;
23 }
```

## Exercice – 14 goto

```
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     LOOP:do {
10
11         if( a == 15) {
12             /* ignorer l'itération */
13             a = a + 1;
14             goto LOOP;
15         }
16
17         printf("valeur de a: %d\n", a);
18         a++;
19
20     }while( a < 20 );
21
22     return 0;
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

```
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 }
```

## 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     * &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 }
```

```
.-----
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
```

# Tableaux

## Exercice – 18

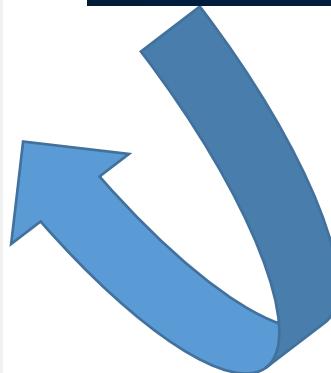
```
1 #include <stdio.h>
2
3 int main () {
4
5     int n[ 10 ]; /* n est un tableau de 10 nombres entiers */
6     int i,j;
7
8     /* initialise les éléments du tableau n à 0 */
9     for ( i = 0; i < 10; i++ ) {
10         n[ i ] = i + 100; /* définir l'élément à l'emplacement i à i + 100 */
11     }
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 tableau de fonction

- ne peut 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( "*(%d + %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("%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("%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 pointers

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 }
```

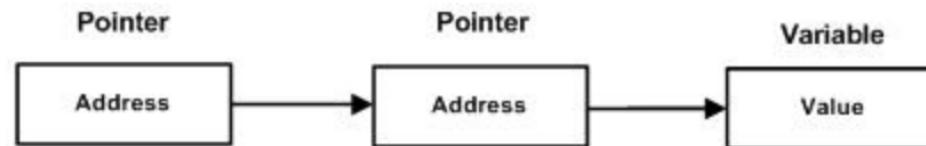
## Exercice – 27

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'adresse de 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

```
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, "Tutorial 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 }
```

## Structures en C

```
Libre titre : Programmation C
Libre auteur : Nuha Ali
Libre sujet : Tutorial 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
```

## 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 /* 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, "Tutorial 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 }
```

# Exercice – 31

# Structures en C

## Pointers de Structure

```
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, "Tutorial 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 : Tutorial 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