GRANDE ÉCOLE D'**INGÉNIEUR GÉNÉRALISTE**

# Programmation Embarquée

## 2021-2022

www.hei.fr

# VI. Fonctions

## 1. Les procédures:

Une **procédure** est un **ensemble d'instructions** portant un nom.
Pour définir une procédure, on utilise la syntaxe :

```
void nomprocedure()
{
/*
instructions
*/
}
```

Une procédure est une nouvelle instruction, il suffit pour l'exécuter d'utiliser son nom. Par exemple, pour exécuter (on dit aussi appeler ou invoquer) une procédure appelée ***pr***, il suffit d'écrire ***pr();***

# VI. Fonctions

## 1. Les procédures:

**Les deux programmes suivants font la même chose:**

Le premier est écrit sans procédure:

```
#include<stdio.h>

int main ()
{
   printf("Bonjour\n") ;
   return 0;
}
```

Et dans le deuxième, le **printf** est placé dans la procédure **afficheBonjour** et cette procédure est appelée depuis le **main**:

```
#include<stdio.h>

void afficheBonjour ()
{
   printf ("Bonjour\n");
}

int main ()
{
   afficheBonjour () ;
   return 0 ;
}
```

# VI. Fonctions

## 1. Les procédures:

Vous pouvez définir autant de procédures que vous le voulez et vous pouvez même appeler des procédures depuis des procédures!

Testez l'affichage du programme à gauche

### a) Variables locales:

Une procédure est un bloc d'instructions et est sujette aux mêmes règles que main. Il est donc possible de déclarer des variables :

```
void nomprocedure(){
        /*
        declaration de variables
        */
        /*
        instructions
        /*
}
```

```
#include<stdio.h>
void afficheBonjour(){
   printf("Bonjour\n") ;
}
void afficheUn(){
   printf("1\n");
}
void afficheDeux()
{
   printf("2\n");
}
void afficheUnEtDeux(){
   afficheUn();
   afficheDeux();
}
void afficheAuRevoir(){
   printf("Au revoir\n");
}
int main(){
   afficheBonjour();
   afficheUnEtDeux();
   afficheAuRevoir();
   return 0;
}
```

# VI. Fonctions

## 1. Les procédures:

Variables locales:

Attention, ces variables ne sont accessibles que dans le corps de la procédure, cela signifie qu'elle naissent au moment de leur déclaration et qu'elles sont détruites une fois la dernière instruction de la procédure exécutée. C'est pour cela qu'on les appelle des variables locales. Une variable locale n'est visible qu'entre sa déclaration et l'accolade fermant la procédure.

Par exemple, ce code est illégal :

```
#include<stdio.h>

void maProcedure()
{
   char a=b;
}
int main ()
{
   char b = 'k' ;
   printf("%c ,%c\n",a,b) ;
   return 0;
}
```

FAUX ! !

La variable b est déclarée dans la procédure main, et n'est donc visible que dans cette même procédure. Son utilisation dans ***maProcedure*** est donc impossible.
De même, la variable a est déclarée dans ***maProcedure***, elle n'est pas visible dans le ***main***.

# VI. Fonctions

## 1. Les procédures:

Passage de paramètres:

si la procédure/fonction reçoit, comme paramètre, la valeur d'une donnée (plus précisément, une copie de celle-ci), alors on parle de ***transmission (ou passage) par valeur***.

si la procédure/fonction reçoit, non pas la valeur de la donnée comme paramètre, mais son adresse, alors on parle de ***transmission (ou passage) par adresse***. **A voir dans le prochain cours!**

**Passage par valeur (call by value):**

La procédure/fonction appelée reçoit une copie de la valeur de l'objet passé comme paramètre effectif. Cette copie est affectée au paramètre formel correspondant. La procédure/fonction travaille donc sur un duplicata, et non sur l'original de la valeur transmise.

```c
#include <stdio.h>

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

int main()
{
    int a=3, b=5; /* variables locales à main */
    printf("Valeur1:%d\tValeur2 : %d\n", a, b);
    show(a, b);    /* appelle show: passage par valeur */
    printf("Valeur1:%d\tValeur2:%d\n",a,b);
    return 0;
}
void show(int x, int y)/* définition de show */
{
  x++ ;
  y++ ;
  printf("Valeur1:%d\tValeur2:%d\n",x,y);
}
```

**Quel est l'affichage du programme ci-haut?**

# VI. Fonctions

## 2. Les fonctions:

Une fonction est une portion de programme composée d'une ou plusieurs instructions et devant accomplir une certaine tâche. On distingue 2 cas :

les fonctions prédéfinies des bibliothèques (telles que **printf** ou **scanf),** livrées avec le compilateur, qui sont intégrées au programme seulement lors de l'édition des liens,
les fonctions personnelles que le programmeur écrit lui-même.

Un programme C comprend habituellement plusieurs fonctions personnelles, dont une au moins, nommée **main.**
**Rappel:** main est la fonction principale de tout programme C. C'est par elle que débute toujours l'exécution du programme.

Un programme C peut être considéré comme une collection de fonctions. Les fonctions, y compris **main**, peuvent être disposées dans un ordre quelconque.

Les fonctions renvoient au programme (plus précisément à la fonction appelante) le résultat de leur travail, cela sous forme de valeur réutilisable.

# VI. Fonctions

## 2. Les fonctions:

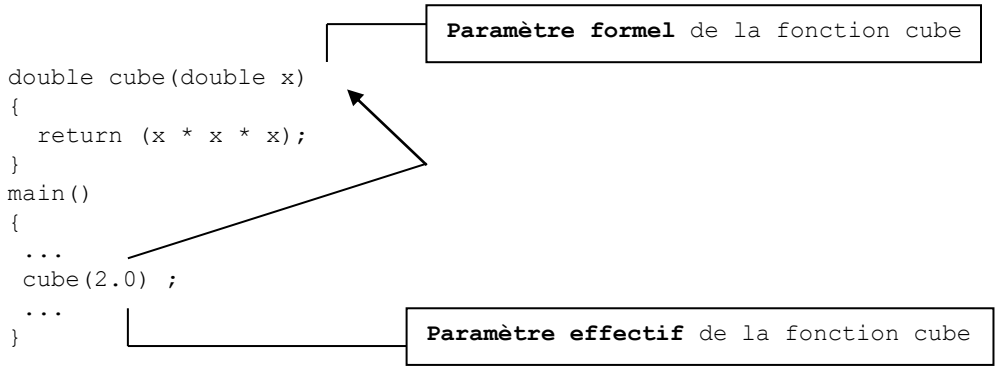
Définition:

On définit une fonction avec la syntaxe suivante :

```
typeValeurDeRetour nomFonction (listeParametres)
          {
          /*
          instructions
          */
          }
```

**Exemple:**

```
int carre(int i)
{
   return i*i ;
}
int somme(int a ,int b)
{
   return a + b;
}
```




**Rappel :** Les copies des valeurs des variables passées comme paramètres effectifs sont modifiées dans la fonction appelée, sans que cela se répercute sur les valeurs originales des variables, dans la fonction appelante. Les valeurs de **a** et **b** ne sont pas affectées

# VI. Fonctions

## 2. Les fonctions:

Déclaration d'une fonction: `type_de_donnee Nom_De_La_Fonction(type1 argument1, type2 argument2, ...)`

La définition d'une fonction doit être globale, c'est-à-dire se faire hors de toute fonction.
Même si la déclaration est parfois facultative (par exemple quand les fonctions sont définies avant la fonction ***main*** et dans le bon ordre), elle seule permet au compilateur de vérifier que le nombre et le type des paramètres utilisés dans la définition concordent bien avec le prototype.

Lorsque le prototype d'une fonction n'est pas déclaré, le compilateur suppose que le type de retour est un *int*. C'est ce qu'il appelle une déclaration implicite (implicit declaration of function)
Cela peut causer des erreurs quand le type d'une fonction non déclarée est *double* par exemple.

```
#include <stdio.h>
int main(){
   double a = 2.5, b = 5;
   printf("%f\n", somme(a,b));
   return 0;
}
double somme(double a ,double b){
   return a + b;
}
```

FAUX ! !

```
#include <stdio.h>
          double somme (double,double);
          int main(){
             double a = 2.5, b = 5;
             printf("%f\n", somme(a,b));
             return 0;
          }
          double somme(double a ,double b){
             return a + b;
          }
```

VRAI !

# VI. Fonctions

## 2. Les fonctions:

Une fonction communique une valeur au sous-programme (portion de programme ) appelant. Cette valeur s'appelle valeur de retour, ou valeur retournée.

Invocation:
La syntaxe pour appeler une fonction est :

```
v=nomfonction (parametres);
```

L'instruction ci-dessus place dans la variable v la valeur retournée par la fonction *nomfonction* quand on lui passe les paramètres *parametres*.

Exemples:

```
v=carre (2);
v=somme (2,3);
```

# VI. Fonctions: TDs

1. Ecrire une procédure qui calcule le carré d'un nombre. ce nombre doit être saisi en paramètre à la procédure. Ecrire le programme principal correspondant.

2. Ecrire une fonction qui calcule l'aire d'un rectangle dont la longueur et la largeur sont saisies en paramètres. Le résultat sera utilisé  dans la fonction main pour calculer le volume du parallélépipède rectangle correspondant.

3. Ecrire une fonction qui retourne le maximum dans un tableau. Ecrire le programme principal qui demande à l'utilisateur de saisir un tableau de taille N (max 10) puis affiche le maximum en appelant la fonction.

4. Faite le même exercice que 3 mais pour le minimum, puis la moyenne.

5. une suite arithmétique est définie sous la forme suivante : Un+1= Un + R
   on donne U0 =5 et R=2. Ecrire une fonction qui calcule Un où N est saisie à partir du clavier. Ecrire le programme principal qui demande à l'utilisateur de rentrer N et appelle la fonction.