Les bonnes pratiques avec les fonctions

plan

- Déclaration
- Définition
- Fonctions sans valeur de retour
  - définition
  - utilisation
- Fonctions avec valeur de retour
  - définition
  - utilisation

Rappel

- **Prototype** (dans le fichier .h):
  - c'est la **déclaration** de la fonction
  - `typeDeRetour nomFonction(`

    `typeParam1 nomParam1, typeParam2 nomParam2, etc) ;`

Rappel

- **Définition**(dans le fichier .c):
  - C'est le **code** de la fonction
  - Les paramètres sont considérés comme des variables correctement initialisées.
  - ```
    typeDeRetour nomFonction(
    typeParam1 nomParam1, typeParam2 nomParam2
    , etc)
    {
      code de la fonction
    }
    ```
    ```

Rappel

- **Définition**(dans le fichier .c):

Sans retour

```
void nomFonction(
typeParam1 nomParam1, typeParam2 nomParam2
, etc)
{
   code de la fonction, pas de return à la fin
}
```

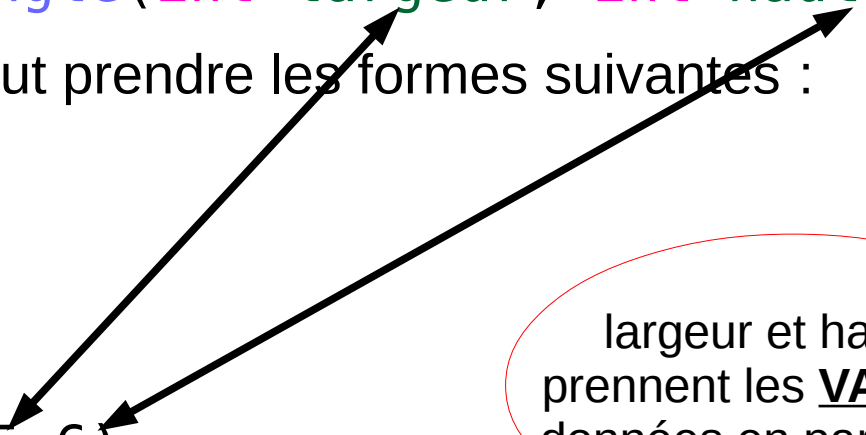
Utilisation

- S'il existe une fonction ayant pour prototype :

```
void afficheUnRectangle(int largeur, int hauteur) ;
```

- L'appel à cette fonction peut prendre les formes suivantes :

```
int v1,v2 ;
v1=5 ;
v2=6
afficheUnRectangle(5,6) ;
afficheUnRectangle(v1,6) ;
afficheUnRectangle(5,v2) ;
afficheUnRectangle(v1,v2) ;
```



largeur et hauteur prennent les **VALEURS** données en paramètres

Rappel

- **Définition**(dans le fichier .c):

Avec retour

```
typeDeRetour nomFonction(
typeParam1 nomParam1, typeParam2 nomParam2
, etc)
{
  typeDeRetour retour ;
   code de la fonction
  return retour ;
}
```

Utilisation

- Fonction avec retour
- s'il existe une fonction ayant pour prototype :

```
float calculLongueur(float x1, float y1, float x2, float y2) ;
```

- L'appel à cette fonction peut prendre la forme suivante :

```
float resultatCalcul ;
resultatCalcul = calculLongueur(1.5, 2.0, 3.8, 4.7) ;
if (resultatCalcul >0)
{
  printf("ca avance\n") ;
}
else
{
  printf("ca ne va pas dans le bon sens\n") ;
}
```

Utilisation

- Fonction avec retour
- s'il existe une fonction ayant pour prototype :

```
float calculLongueur(float x1, float y1, float x2, float y2) ;
```

- L'appel à cette fonction peut prendre la forme suivante :

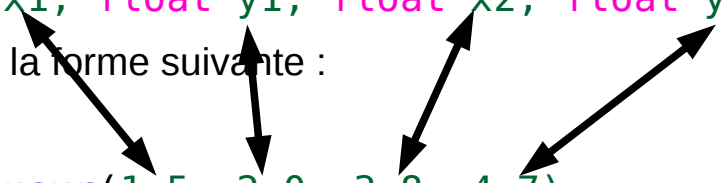
```
float resultatCalcul ;
resultatCalcul = calculLongueur(1.5, 2.0, 3.8, 4.7) ;
if (resultatCalcul >0)
{
  printf("ca avance\n") ;
}
else
{
  printf("ca ne va pas dans le bon sens\n") ;
}
```

Utilisation

- Fonction avec retour
- s'il existe une fonction ayant pour prototype :

```
float calculLongueur(float x1, float y1, float x2, float y2) ;
```

- L'appel à cette fonction peut prendre la forme suivante :



```
float resultatCalcul ;
resultatCalcul = calculLongueur(1.5, 2.0, 3.8, 4.7) ;
if (resultatCalcul >0)
{
  printf("ca avance\n") ;
}
else
{
  printf("ca ne va pas dans le bon sens\n") ;
}
```

Utilisation

- Fonction avec retour
- s'il existe une fonction ayant pour prototype :

```
float calculLongueur(float x1, float y1, float x2, float y2) ;
```

- L'appel à cette fonction peut prendre la forme suivante :

```
float resultatCalcul ;
resultatCalcul = calculLongueur(1.5, 2.0, 3.8, 4.7) ;
if (resultatCalcul >0)
{
  printf("ca avance\n") ;
}
else
{
  printf("ca ne va pas dans le bon sens\n") ;
}
```

Si une fonction retourne une valeur, il faut **OBLIGATOIREMENT** une variable du même type pour la récupérer via une affectation.