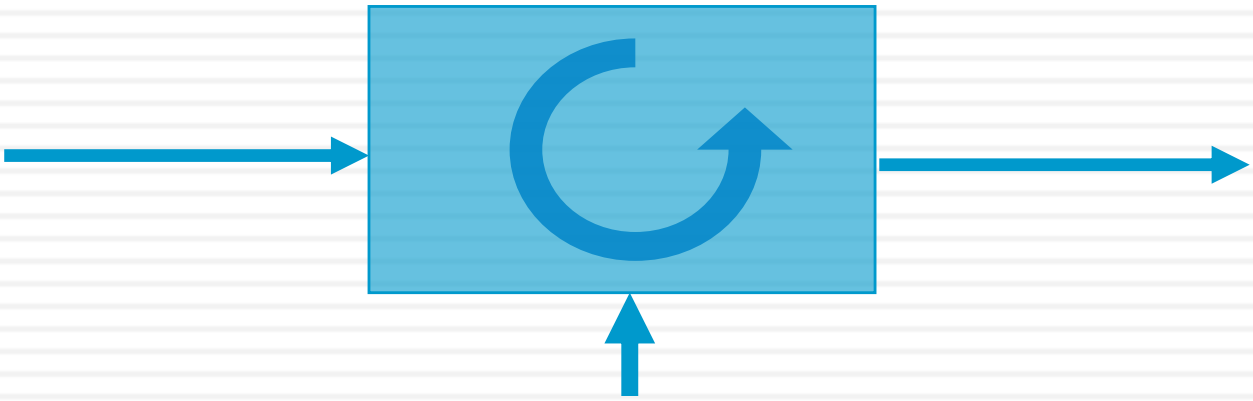
R1.01
DEV1

Les fonctions y : f(x)



Au cours précédent

- Les enregistrements
- Les énumérations

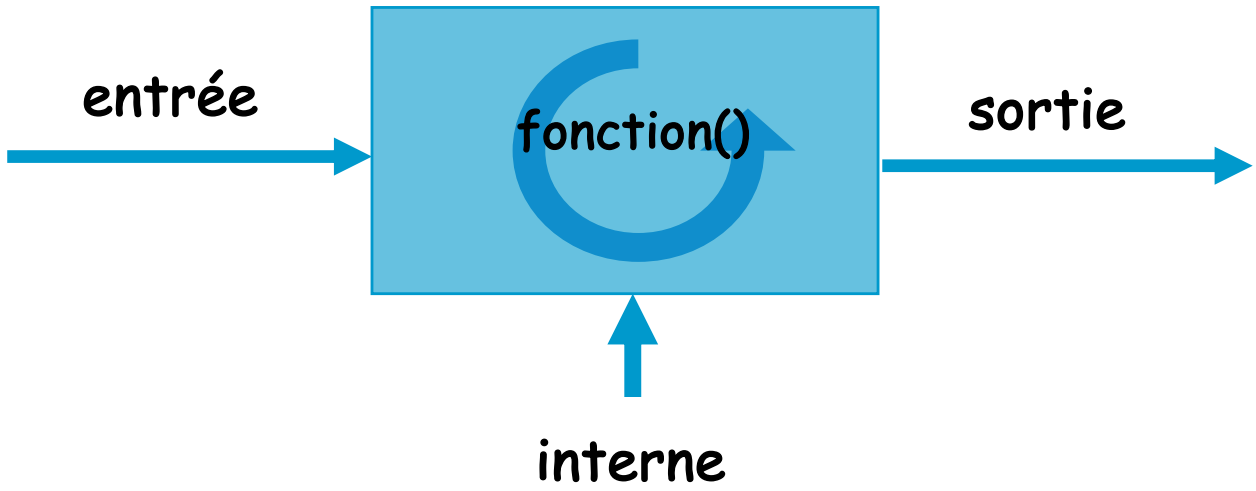
- BOGGLE
- DS

Sommaire

- Les fonctions
- Les appels de fonctions
- Les passages de paramètres
- Les valeurs de retour

Les fonctions

Ensemble d'instructions regroupées sous un nom, qui réalise un traitement particulier dans une fonction lorsqu'on l'appelle et qui peut retourner un mais un seul résultat à la fonction appelante.



Les fonctions

Pourquoi ?

➢ programmation modulaire

- pour organiser le code
- pour utiliser du code existant
  - ➢ fonctions perso.
  - ➢ bibliothèque standard
  - ➢ bibliothèque tiers (Qt, …)
- quand on doit réaliser un même traitement plusieurs fois

Exemple : sqrt()

On veut calculer la racine carrée de plusieurs nombres et arrêt du programme avec 0.

function

sqrt

<cmath> <ctgmath>

C90 C99 C++98 C++11

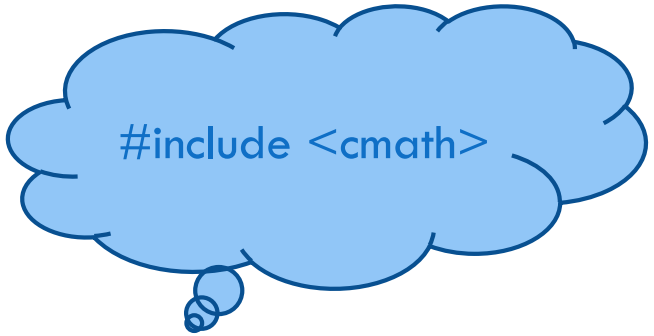
```
double sqrt (double x);      float sqrt (float x);long double sqrt (long double x);      double sqrt (T x);          //
```

Compute square root

Returns the ***square root*** of *x*.

Exemple : sqrt()


```
double nb ;
cout << "Entrez un nb (0 pour quitter) : " ;
cin >> nb ;
while( nb > 0)
{
    cout << "La racine de " << nb << " est " << sqrt(nb) << endl ;
    cout << "Entrez un nb (0 pour quitter)" ;
    cin >> nb ;
}
```



appel de la fonction sqrt() de la bibliothèque cmath

Définition d'une fonction

Une fonction possède :

- un nom,
- des variables dont les paramètres optionnels,
- des instructions,
- un début et une fin.

Une fonction ne peut pas s'exécuter indépendamment d'une autre fonction.

Le point d'entrée d'un programme est la fonction main().

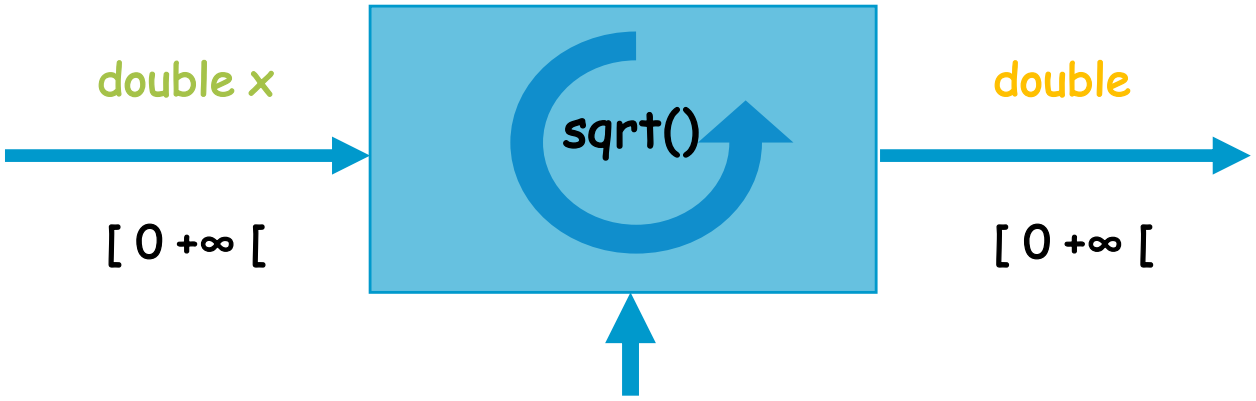
Exemple : sqrt()


```
double sqrt ( double x ) ; // déclaration
```

type de retour

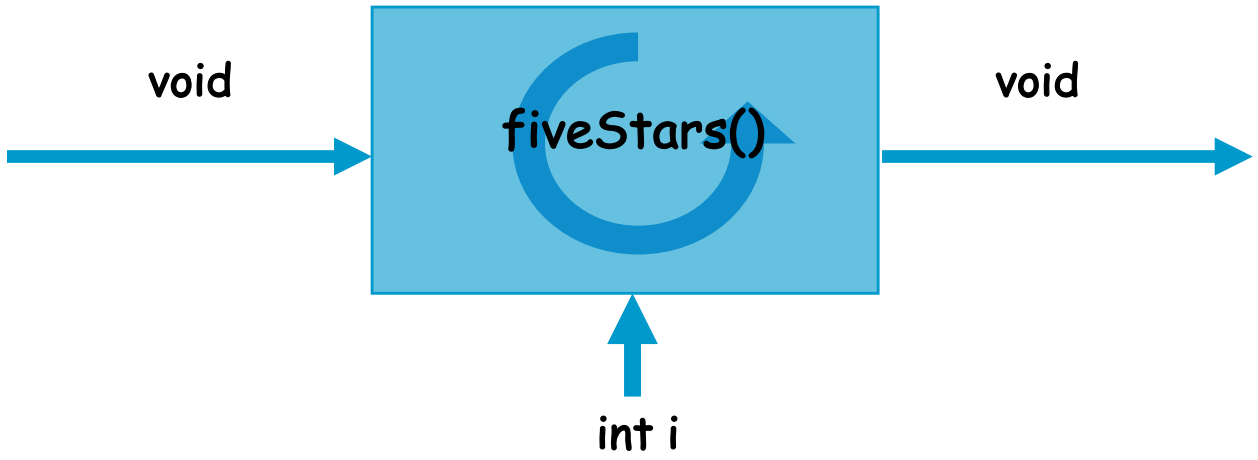
nom

paramètre
type + nom



Exemple de fonction perso

On veut une fonction qui dessine 5 étoiles ***** puis saute une ligne



*****↵

Exemple de fonction perso

On veut une fonction qui dessine 5 étoiles ***** puis saute une ligne

```
// Draw one line with five stars
void fiveStars()
{
    for(int i=0 ; i<5 ; i ++)
    {
        cout << "*" ;
    }
    cout << endl ; // saut de ligne
}
```

commentaire de la fonction en anglais

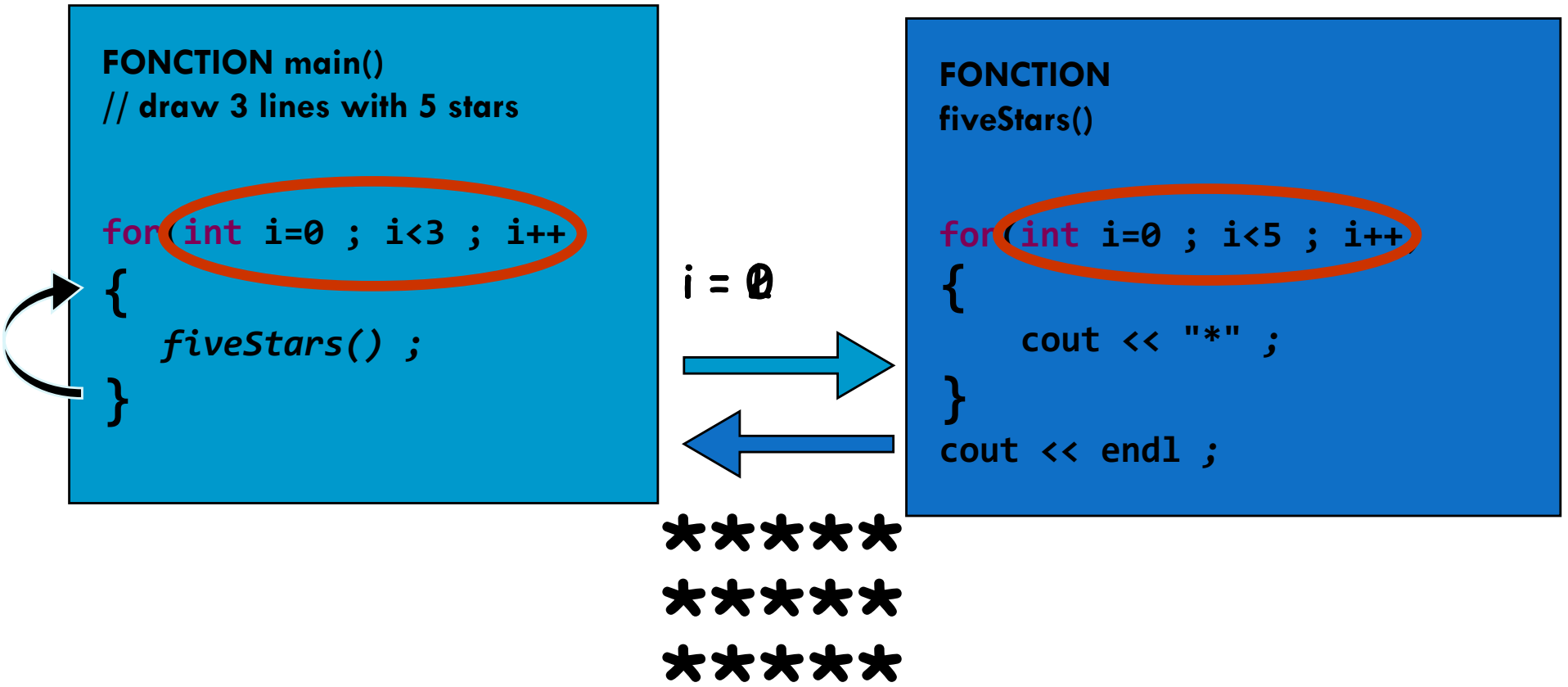
Appel d'une fonction

On fait appel à la fonction fiveStars ( ) pour dessiner 3 lignes de 5 étoiles

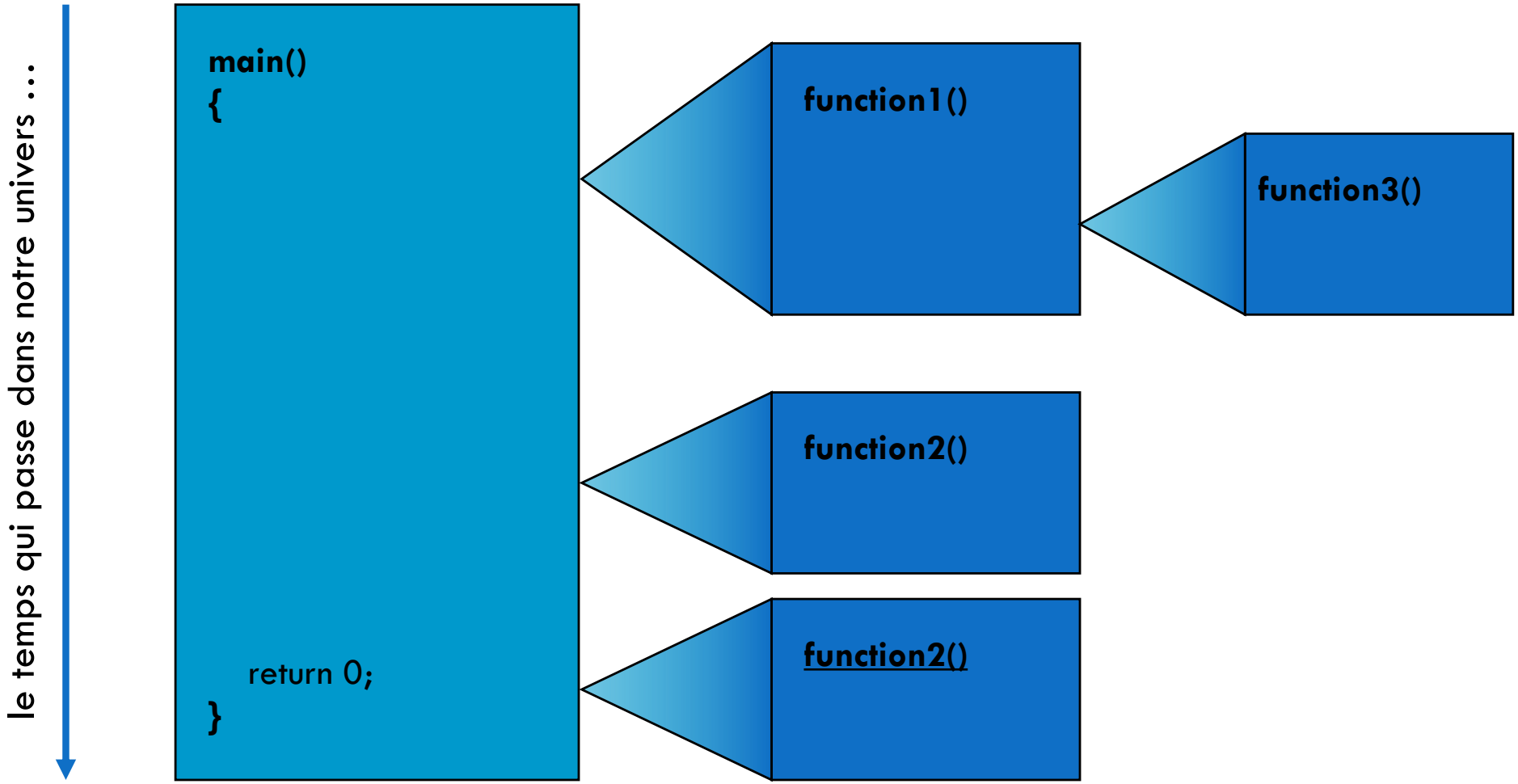
```
// Draw 3 lines with 5 stars

int main()
{
   for(int i=0 ; i<3 ; i ++)
   {
      fiveStars() ; // appel de la fonction
   }
   return 0 ;
}
```

Appel d'une fonction

variables locales



Les appels de fonctions imbriquées

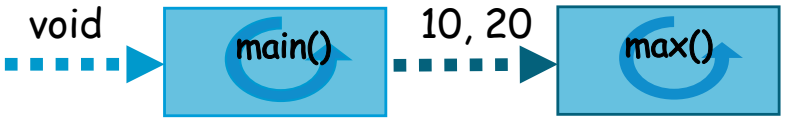


Les fonctions dans un programme


```
// Cfonction/6_max(2+3)/main.cpp
// cours sur les fonctions : ex max()
// EC 01/02/03
#include <iostream>          // cout, cin

// compute max of 2 integers
// nb1 : one integer
// nb2 : another one
// return the max value between nb1 and nb2
int max (int nb1, int nb2)
{
      return (nb1>nb2 ? nb1: nb2) ;
}

// Test of max() function
int main(void)
{
      cout << max(  10,  20) << endl ;
      cout << max( -10, -20) << endl ;
      cout << max(  10,  10) << endl ;

      return 0;
}
```

fichier source main.cpp avec entête et inclusion de la bibliothèque de flux d'E/S de l'espace de nom standard

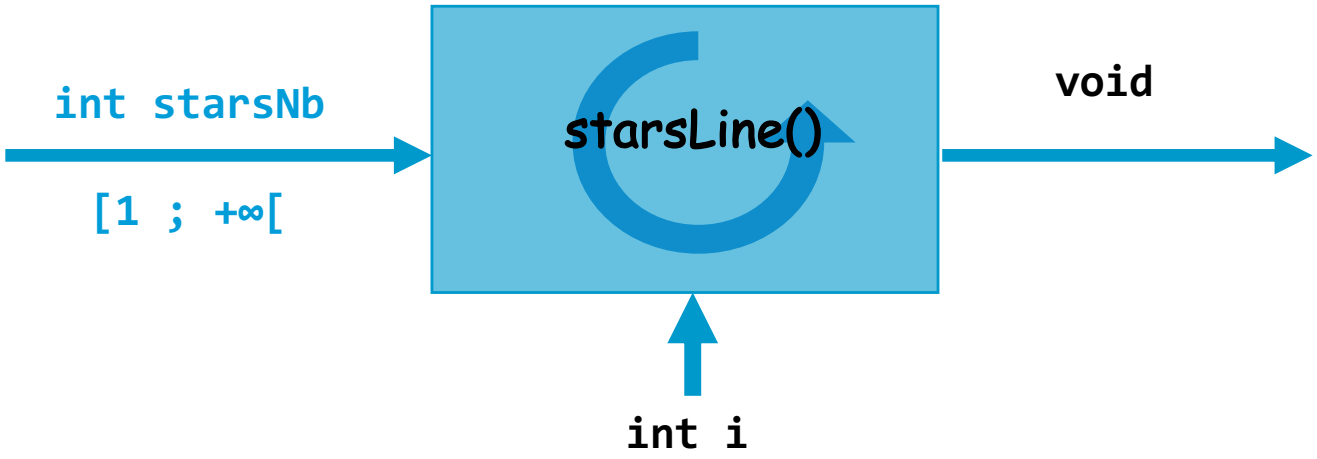
définition de la fonction `max()` appelée par la fonction `main()`



définition de la fonction principale `main()`
→ point d'entrée de l'application

Exemple de fonction avec paramètre

On demande le nombre d'étoiles qu'on veut afficher par ligne.



Exemple de fonction avec paramètre

On veut une fonction qui dessine un nb fini d'étoiles.

```
// draw a line with some stars
// starsNb : nb of stars by line

void starsLine(int starsNb)
{
    for(int i=0 ; i<starsNb ; i++)
    {
        cout << "*" ;
    }
    cout << endl ;  // saute 1 ligne
}
```

commentaire de la fonction

Appel d'une fonction avec paramètre

On fait appel à la fonction starsLine( ) pour dessiner 3 lignes de nb étoiles.

```
// draw 3 lines with nb stars

int main()
{
   int nb ;
   cout << "nb étoiles ? " ;
   cin >> nb ;

    for(int i=0 ; i<3 ; i++)
    {
        starsLine(nb) ; // appel de la fonction paramétrée
    }
    return 0 ;
}
```

Fonction avec passage de paramètres


```
FONCTION main()
// draw 3 lines with nb stars


cout << "nb étoiles ? " ;
cin >> nb ;

for(int i=0 ; i<3 ; i++)
{
  starsLine(nb) ;
}
```

paramètre réel ou effectif

starsNb = nb

COPIE

```
FONCTION
starsLine(int starsNb)
```

paramètre formel

```
for(int i=0 ; i<starsNb ; i++)
{
    cout << "*" ;
}
cout << endl ;
```

Les paramètres formels

Les paramètres placés dans la définition d'une fonction sont les paramètres formels.

Ils servent à décrire le traitement à réaliser par la fonction indépendamment des valeurs traitées.

Les paramètres formels sont des variables locales à la fonction, et à ce titre ils sont déclarés dans le prototype de la fonction. On parle de la signature de la fonction.

Les paramètres réels ou effectifs

Les paramètres placés dans l'appel d'une fonction sont les paramètres réels ou effectifs.

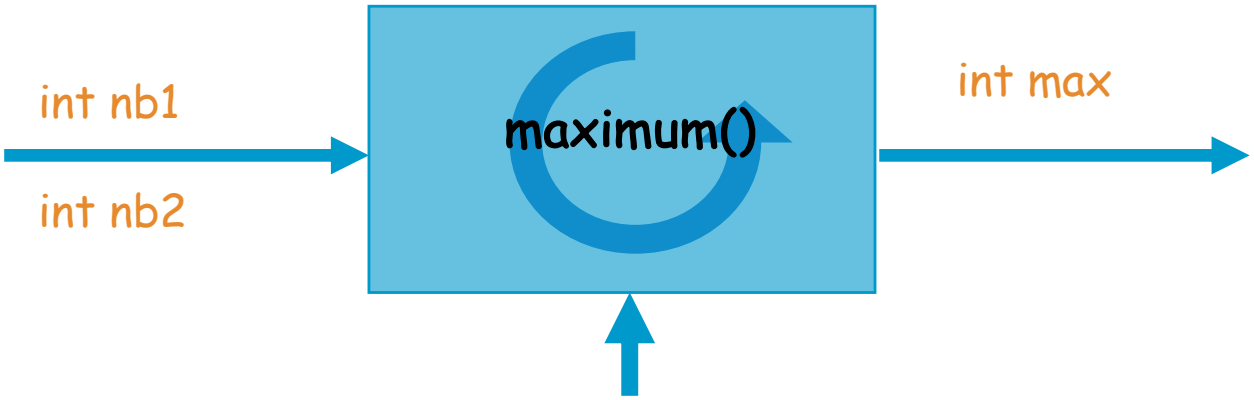
Lorsqu'ils sont de type *donnée*, ils contiennent effectivement les valeurs sur lesquelles sera effectué le traitement de la procédure.

Lors de l'appel, leur valeur est recopiée dans les paramètres formels correspondants.

Un paramètre effectif en *donnée* peut être soit une variable du programme appelant, soit une valeur littérale (ex : 10) , soit le résultat d'une expression (ex : 3*nb).

Exemple de fonction avec retour

On veut le maximum de 2 entiers.



Exemple de fonction avec retour

On veut une fonction qui renvoie le maximum de 2 entiers.

```
// returns the greater of 2 integers
// nb1 : one integer
// nb2 : another one
// returns the greater of nb1 and nb2

int maximum(int nb1, int nb2)
{
    int max ;
    if (nb1 > nb2)
        max = nb1 ;
    else
        max = nb2 ;
    return max ;          // renvoie la valeur de max

    // return nb1>nb2 ? nb1:nb2 ;
}
```

commentaire de la fonction

Appel d'une fonction avec valeur de retour

On fait appel à la fonction maximum( ) qui renvoie le maximum de 2 entiers.

```
int main()
{
    int nb1, nb2 ;
    cout <<"Entrez 2 entiers : " ;
    cin >> nb1 ;
    cin >> nb2 ;
                                        // appel de la fonction
    int max = maximum(nb1, nb2) ;       // et affectation de la
                                        // valeur de retour à max
    cout << "max = " << max ;

    // cout << "max = " << maximum(nb1, nb2) ;
}
```

Exemple de fonction avec retour

variables locales à la fonction main()

```
FONCTION PRINCIPALE
int main()


int nb1, nb2 ;
cout << Entrez 2 entiers : " ;
cin >> nb1 ;
cin >> nb2 ;

int max = maximum(nb1, nb2) ;

cout << "max = " + max ;
```

nb1, nb2

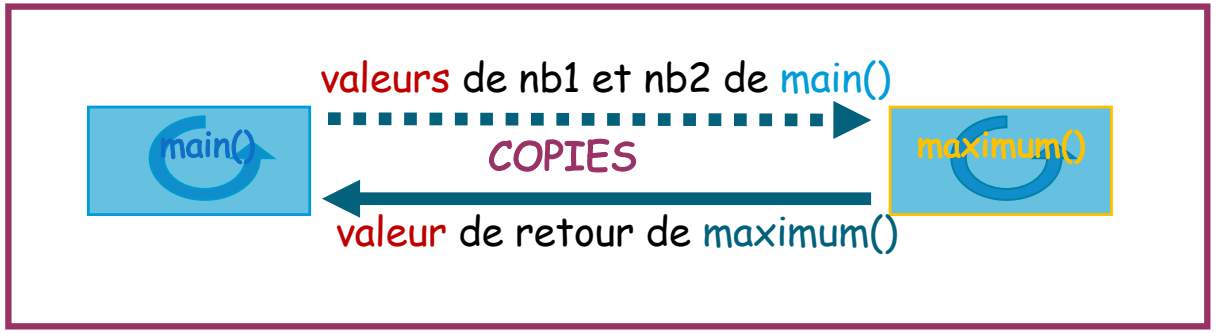
COPIES

valeur de max

variables locales à la fonction max()

```
FONCTION
int maximum(int nb1,
            int nb2)

int max ;

if (nb1>nb2)
  max = nb1 ;
else
  max = nb2 ;

return max ;
```

Les fonctions

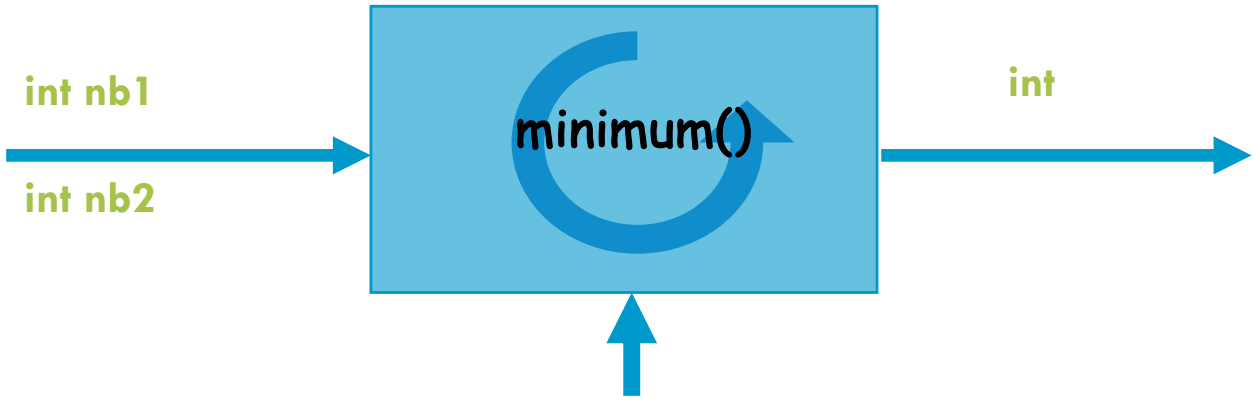


Copie de la valeur des paramètres réels `nb1, nb2` de la fonction appelante `main()` vers les paramètres formels `nb1, nb2` de la fonction de traitement `maximum()`.

Copie de la valeur de retour de la fonction de traitement `maximum()` vers le paramètre `max` de la fonction appelante `main()`.

Exemple de fonction avec retour

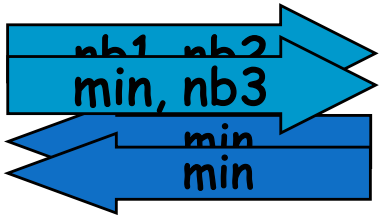
On veut le minimum de 3 entiers en utilisant la fonction `minimum(nb1,nb2)` .



Les fonctions

ex : minimum de 3 entiers

```
FONCTION PRINCIPALE
int main()
{
   int nb1, nb2, nb3, min ;

   cout << "Entrer 3 entiers : " ;
   cin >> nb1 >> nb2 >> nb3 ;

   min = minimum (nb1, nb2) ;
   min = minimum (min, nb3) ;

   cout << "min = " << min ;

   // cout << " min = "  <<
   // minimum( minimum(nb1, nb2), nb3 ) ;
}
```




```
FONCTION
int minimum (int nb1, int nb2)
{
   int min ;

   if(nb1>nb2)
     min = nb2 ;
   else
     min = nb1 ;

   return min ;
}
```