

# Procédure et fonction

## \* programme principale

void main() { // procédure qui retourne void vide / rien

# “ Lecture d’imagination

# // Affichage In tab lesson

"Tri du las leau

"Affichage du tableau (après le tri.)

 Bloco  
republicano

- pourquoi ne pas créer un seul segment de code puis faire appel à ce segment en cas de besoin

→ le gain des procédures et finiti c'est la  
réalisation de segment de code

void main() {

//affichage du tableau horizontal

(for( $i = 0; i < N; i++$ )  
printf("%f ", T[i]);

//tri

„affiche

void afficheTab(float T[], int N)  
{  
int i;  
for( $i = 0; i < N; i++$ )  
printf("%f ", T[i])  
}

void main() {  
T[] = { ... };  
N = 15;

affichage

afficheTab(T, N); & appel

„tris

„afficheTab(T, N); & appel

Definite:

NonProcedure( parameters ) {

The diagram consists of two main parts: the word "Procedure" written in blue cursive at the top right, and the word "inH()" written in blue cursive at the bottom left. A curved arrow, also in blue, originates from the letter "e" in "Procedure" and points downwards and to the left, ending near the opening parenthesis of "inH()".

$$u = \text{abs}(y)$$

Nomfonction (paramètres) {

corps } return         valeur\_type de retour }

void

int student  
type de retour  
flot

```

#include <stdio.h>
- mesfonctis.h -
    // procédure d'affichage d'un tableau de réel
void afficheTab(float T[], int N) {
    ;
}

// procédure de lecture d'un tableau de réel
void lectureTab(float T[], int N) {
    ;
}

// ...
;

}

Programme procédurale

```

```

#include <stdio.h>
#include "r;mesfonction.h"
void main() {
    float T[100];
    int N=10;
    // Lecture
    lectureTab(T,N);
    // affichage
    afficheTab(T,N);
    // Tri par sélection
    triparselection(T,N);
    // affichage après le tri
    printf("Le tableau après le tri est :");
    // affichage
    afficheTab(T,N)
}

```

## Syntaxe

- le nom doit commencer par une lettre ou - / les noms  
• r " ne doit pas contenir des espaces      n'importe quelles qui sont appliquées au nom de variable
- lectureTab ( )  
lecture\_tab()  
lectureTab( )      → erreur
- lecture ( — )      → erreur

```
void lectureTab ( int T[], int taille ) {  
    // T et taille sont des variables locales à  
    // la procédure ( elles ne sont pas accessible depuis l'extérieur  
    // de la procédure )  
    int i ;  
    for ( i = 0 ; i <= taille ; i++ ) {  
        printf (" L'elt d'indice %d ", i );  
        scanf ("%d", &T[i] );  
    }  
}
```

// utilisation (appel)

void main () {

int Tab [100];

int N = 10;

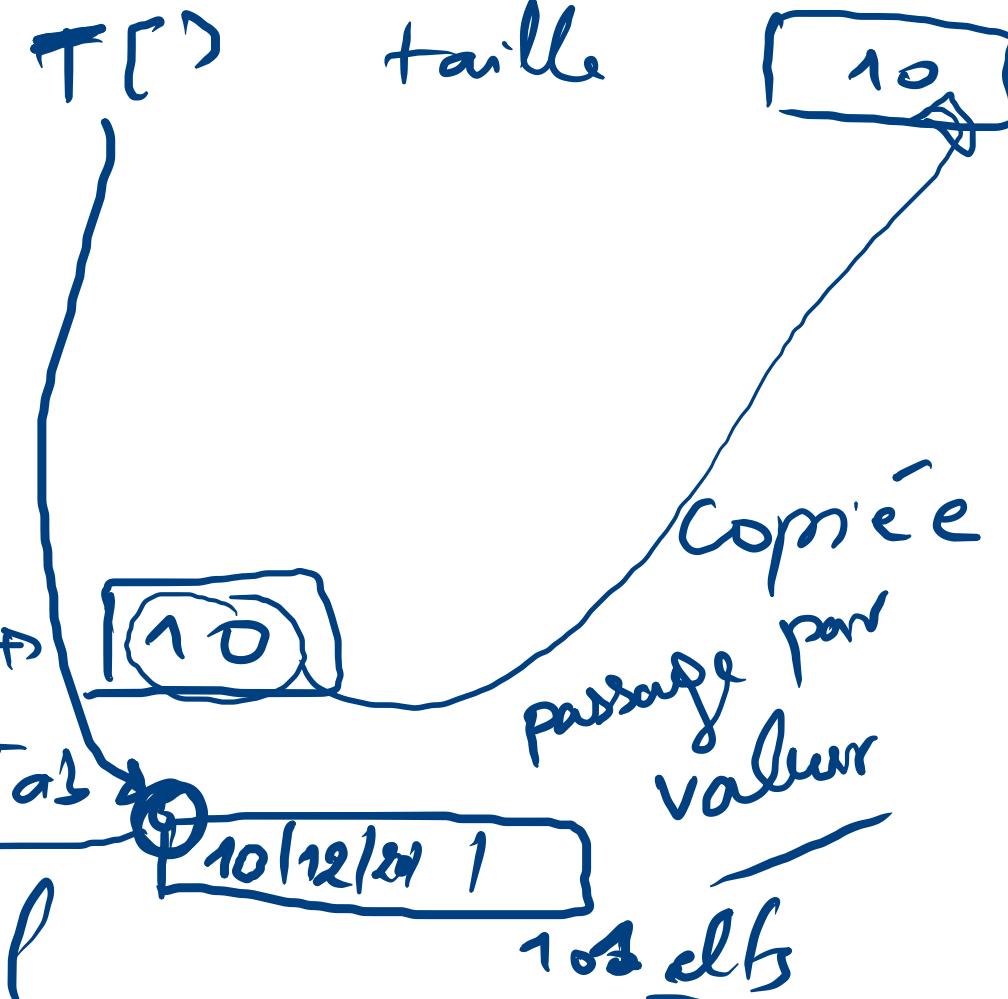
// lecture

lectureTab (Tab, N); // appel

lectureTab (Tab, 20); // appel correct

// affichage

afficheTab (Tab, N);



## Passage par valeur et par adresse

```
void main() {
```

```
    int x = 20;
```

```
    incrementer(x);
```

→ printf("%d", x);

qui elle est la valeur  
affidree ici ??

✓ | 21(6)  
✓ | 20(3) △  
manzraf

```
void incrementer(int a){
```

```
    a++;
```

E A01



passage par valeur  
(copie de la  
valeur)

① passage par adresse  
 ② fonction qui me retourne la valeur  
 incrementée

③ ancien de "void incrementer"

int incrementer (int a) { fonction

$a + i$  ————— a  
 return a; i

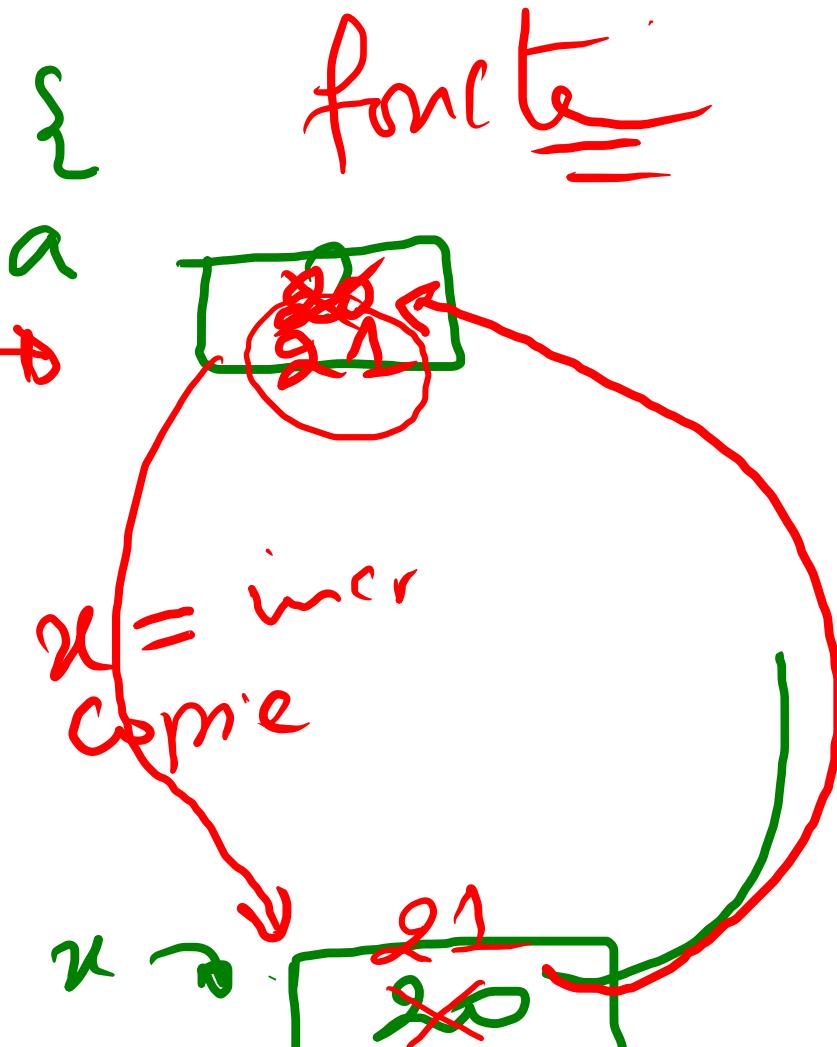
// principal

void main () {

int x = 20;

x = incrementer (x); // copie

printf ("%d", x);  $\Rightarrow$  21



② passage par adresse

séances prochaines