

# Introduction au C

## Passage de paramètres

University of Geneva  
www.miralab.ch

## Rappel : variables globales

### Rappel

Sont visibles de toutes les routines

### Limitations

Protection d'accès quasi impossible

N'importe quelle routine peut les modifier

Le transfert d'information se fait par nom variable

Les routines appelante et appelée doivent utiliser le même  
label de variable

Implicite

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Solution

8

Passage **explicite** de paramètres

Liste d'arguments passés à l'appel

Déclaration indique la liste des arguments

```
bool exemple(int arg1, float arg2);
```

Référence peut passer des variables locales

```
int i;
```

```
float f;
```

```
bool résultat = exemple(i, f);
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Indépendance des labels

9

Définition utilise ses propres labels de variables

```
bool exemple(int vari, float varf)
{
    if(vari > 0 && varf > 0.0f)
        return true;
    return false;
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

10

## Appel d'une fonction paramétrée

Respect des types des paramètres, sinon

soit erreur de compilation

soit un type-cast implicite

Respect du nombre de variable

Respect de l'ordre des variables

Rappel

Respect du type de valeur retournée

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

11

```
//déclaration
void rien2special(double d);

void appelante(void)
{
    int vali = 20;
    char * valc = NULL;
    rien2special(vali); // OK implicitement convertie en double
    rien2special(valc); //pas OK : pas de conversion implicite,
                        //erreur de compilation
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

12

```
//déclaration
void rien2special(double d, int i);

void appelante(void)
{
    int vi = 20;
    double vd = 10.5;
    rien2special(vd ,vi); // OK
    rien2special(vd);     // erreur
    rien2special(vi, vd); // OK, type-cast
    rien2special(vd, vd); // OK, type-cast
}
```

13

## Exemple

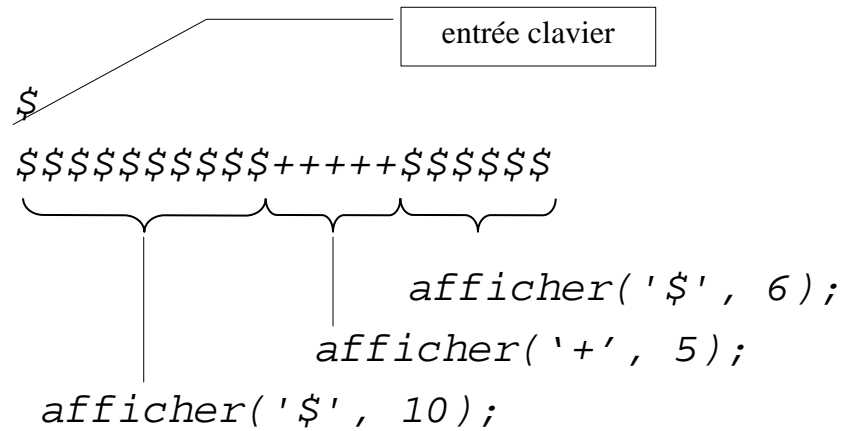
Procédure qui affiche  $n$  fois 1 caractère  $c$

```
void afficher(char c, int n)
{
    int i;
    for(i=0; i < n; ++i)
        cout << c;
}

int main()
{
    char c;
    cin >> c;
    afficher(c, 10); // afficher 10 fois le caractère c
    afficher('+', 5); // afficher 5 fois '+'
    afficher(c, 6); // afficher 6 fois le caractère c
}
```

## Exécution

14



University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## 2 manières de « passer » les paramètres

15

Passage par valeur

Copie des valeurs

Passage par référence

Passage du pointeur

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

# Passage par valeur

16

## 1. Les paramètres sont passés par valeur

La valeur de la variable du programme appelant est copiée dans une nouvelle variable locale à la routine à chaque appel

La durée de vie des variables correspond à la durée de l'exécution de la fonction

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

17

```
void afficher(int n,
              char c)
{
    int i;
    for(i=0; i<n; ++i)
        cout << c;
}

void main()
{
    int nm=10;
    char cm='z';
    afficher(nm, cm);
}
```

Correspond à :

```
void main()
{
    int nm=10;
    char cm='z';
    {
        int n=nm;
        char c=cm;
        int i;
        for(i=0; i<n; ++i)
            cout << c;
    }
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

18

## 2. Les paramètres sont passés par valeur

Les modifications à l'intérieur de la fonction ne changent pas la valeur de la variable du programme appelant

19

```
void affich(int n)
{
    n += 10;
    cout << "n=" << n;      n=11
}

void main()
{
    int n=1;
    affich(n);
    cout << "n=" << n;      n=1
}
```

20

## Passage par valeur : résumé

Passage par valeur :

Copie des valeurs

Pas de modification des variables utilisées à l'appel

seul la copie des valeurs est modifiée durant l'exécution de la fonction

21

## Comment modifier un paramètre ?

En passant son pointeur

Soit la procédure

```
void affich(int n); //déclaration
affich(numero);    //appel
```

Pour modifier *n* dans *affich()* et obtenir sa nouvelle valeur dans la routine appelante

```
void affich(int * n); //déclaration
affich(&numero);     //appel
```



22

```

void affich(int* nn)
{
    *nn += 10;
    cout << "n=" << *nn;
}
void main()
{
    int n=1;
    int* pn = &n;
    affich(pn);
    /*pn += 10
    cout << "n=" << n;
}

```

n=11

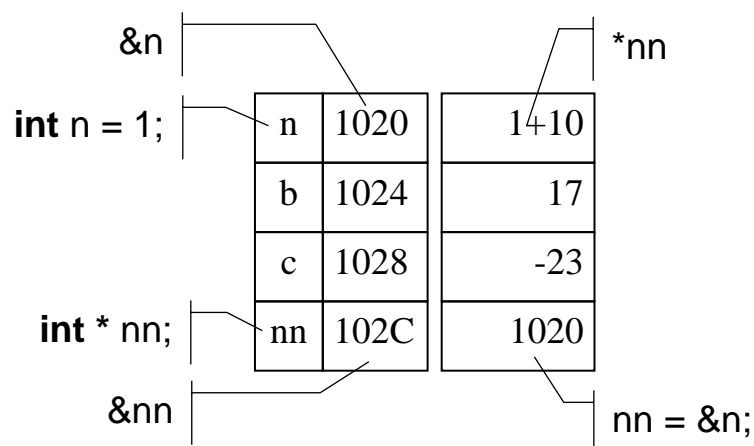
n=11

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

23

## Représentation mémoire



University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Passage par référence en C++

C++ offre une alternative aux pointeurs

Permet de modifier une variable à travers une routine

Utilisation de & après le type dans la déclaration

Pour indiquer un passage par référence

Aperçu de la syntaxe

```
void affich(int & n); //déclaration
affich(numéro);      //appel
```

```
void affich(int & n)
{
    n += 10;
    cout << "n=" << n;      n=11
}

void main()
{
    int n=1;
    affich(n);
    cout << "n=" << n;      n=11
}
```

## Exercice

University of Geneva  
www.miralab.ch

## Exercice 6

27

Reprendre l'exercice 5, en utilisant des paramètres aux fonctions de calcul de distance.

Tout pixel d'une image est caractérisé par un couple de coordonnées  $(x,y)$ . On peut donc calculer des distances entre pixels. Les distances les plus courantes sont (pour deux pixels  $P(x_p,y_p)$  et  $Q(x_q,y_q)$ ):

- distance de Manathan :  $d1(P,Q) = |x_p - x_q| + |y_p - y_q|$
- distance euclidienne :  $d2(P,Q) = [(x_p - x_q)^2 + (y_p - y_q)^2]^{1/2}$
- distance de l'échiquier :  $dinf(P,Q) = \text{Max}(|x_p - x_q|, |y_p - y_q|)$

Ces distances sont reliées par la propriété :

$$dinf(P,Q) \leq d2(P,Q) \leq d1(P,Q)$$

Écrire un programme définissant les structures nécessaires définir les points et les fonctions permettant de calculer les distances de **Manathan** et **euclidienne** (echiquier en option ☺) + un petit programme *main()* utilisant ces fonctions.

PS: <math.h> fournit 2 fonctions intéressantes : `abs()` et `sqrt()` ☺

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Exercice 7

28

Calculer son indice de masse corporelle (IMC) permet de chiffrer une surcharge ou une insuffisance pondérale. L'IMC indique également quel poids nous convient pour une santé optimale. Souvent utilisé par les médecins pour dépister un problème pondéral, il fournit aussi à ceux qui veulent éloigner une maladie liée au poids (hypertension, diabète, cancer...) une aide précieuse pour identifier le nombre de kilos à perdre ou à gagner.

L'IMC se calcule en divisant le poids [exprimé en kilogrammes] par la taille au carré [exprimée en mètres] :

$$\text{IMC} = \text{Poids} / (\text{Taille})^2$$

On parle de surcharge pondérale lorsque le IMC dépasse 25 et d'insuffisance pondérale lorsque le IMC est inférieur à 19.

1. Créer une fonction qui calcul l'IMC à partir de 2 paramètres (poids [en kg] et sa Taille [en mètre])
2. Ecrire la fonction main demandant à l'utilisateur de saisir les valeurs et d'afficher le résultat en fonction de l'IMC :
  - IMC < 19 : « Mince alors ! »
  - 19 ≤ IMC ≤ 25 : « Vous avez un poids idéal pour votre taille, félicitations ! »
  - IMC > 25 : « Faites du sport et mangez équilibré ! »

## Exercice 8

29

### Boîte à outils 3D

Définir une structure permettant de manipuler les points 3D : point3D

Définir une structure pour les vecteurs (utilisant 2 points) : vecteur

Écrire les fonctions/procédures suivantes (en utilisant les paramètres)

saisirPoint : permet de saisir les coordonnées d'un point par l'utilisateur

vect\_construc : construit un vecteur à partir de 2 points

vect\_norme : retourne la norme d'un vecteur

$$|\mathbf{V1}| = (x1.x1 + y1.y1 + z1.z1)^{1/2}$$

vect\_add : ajoute 2 vecteurs (résultat dans un 3<sup>ème</sup> vecteur)

vect\_mulScal : multiplication par un scalaire (modifie le vecteur en paramètre)

## Exercice 8 ... suite

30

vect\_prodScal :

produit scalaire:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos(\phi) = \mathbf{a}' \cdot \mathbf{b}$$

$$= a_1 b_1 + a_2 b_2 + a_3 b_3$$

$\phi$  : angle entre  $\mathbf{a}$  et  $\mathbf{b}$



le produit est = 0  
si  $\phi$  est droit !

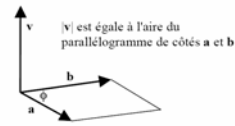


\* calcul d'un angle:  $\cos(\phi) = \mathbf{a} \cdot \mathbf{b} / |\mathbf{a}| |\mathbf{b}|$   
 $\Rightarrow \phi = \arccos(\mathbf{a} \cdot \mathbf{b} / |\mathbf{a}| |\mathbf{b}|)$

vect\_prodVect :

Produit vectoriel  
 $\mathbf{v} = \mathbf{a} \times \mathbf{b}$  est un vecteur orthogonal à  $\mathbf{a}$  et à  $\mathbf{b}$   
 de norme :  $|\mathbf{v}| = |\mathbf{a}| |\mathbf{b}| \sin(\phi)$

$$\begin{aligned} v_1 &= a_2 b_3 - a_3 b_2 \\ v_2 &= a_3 b_1 - a_1 b_3 \\ v_3 &= a_1 b_2 - a_2 b_1 \end{aligned}$$



vect\_angle : angle entre 2 vecteurs

calcul d'un angle:  $\cos(\phi) = \mathbf{a} \cdot \mathbf{b} / |\mathbf{a}| |\mathbf{b}|$   
 $\Rightarrow \phi = \arccos(\mathbf{a} \cdot \mathbf{b} / |\mathbf{a}| |\mathbf{b}|)$

University of Geneva  
[www.miralab.ch](http://www.miralab.ch)

LCI  
 Introduction au langage C

## Tableaux

University of Geneva  
[www.miralab.ch](http://www.miralab.ch)

## Les tableaux unidimensionnels

32

### Problème

Comment mémoriser un grand nombre de valeurs ?  
Sans utiliser un grand nombre d'identificateurs de variables ?

### Solution

Nom de variable unique  
Référencer valeurs par 1 numéro ou 1 adresse

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Définitions

33

### Tableau ou variable indexée

Variable permettant de mémoriser plusieurs valeurs

### Indice ou index

Numéro qui permet d'accéder à une valeur particulière du tableau

### Composante ou entrée du tableau

Valeur particulière pour un index donné

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Déclaration d'un tableau

34

### Préciser

Sa dimension = le nombre d'éléments contenus dans le tableau

### Exemples

```
int    nb[200];
char   tab[10];
float  prix[30];
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Indexage

35

En C, les indices vont de 0 à n-1

... et non pas de 1 à n

```
#define NOMBRE 20
int main()
{
    int i;
    int nb[NOMBRE];
    for(i = 0; i < NOMBRE; ++i)
        cin >> nb[i];
    for(i = NOMBRE-1; i >= 0; --i)
        cout << nb[i] << endl;
}
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

36

## On utilise un tableau

Les composantes sont des entiers=compteur

L'indice = la lettre dont on compte l'occurrence

Pour mapper ['a'..'z'] en [0..25] on soustrait le caractère par le code ASCII de 'a'

'a' - 'a' = 0

'z' - 'a' = 25

37

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

#define MAXLETTRE 26

int main() {
    char c;
    int freq[MAXLETTRE];

    // initialisation du tableau
    for(c = 'a'; c <= 'z'; ++c) {
        freq[c-'a'] = 0;
    }

    // lecture et comptage
    do {
        cin >> c; // lit un caractère
        if (c >= 'a' && c <= 'z')
            freq[c-'a'] += 1;
    }
    while (c != '.');

    // affichage du resultat
    for(c = 'a'; c <= 'z'; ++c) {
        cout << c << " " << freq[c-'a'] << endl;
    }
    return(0);
}
```

```
C:\d:\StefWork_vc7\Test\ex1\Debug\ex1.exe
un chasseur sachant chasser sans son chat.
a 6
b 0
c 4
d 0
e 2
f 0
g 0
h 4
i 0
j 0
k 0
l 0
m 0
n 4
o 1
p 0
q 0
r 2
s 8
t 2
u 2
v 0
w 0
x 0
y 0
z 0
Press any key to continue_
```



## Initialisation à la déclaration

38

Donner la liste des valeurs

entre accolades { }

séparées par des virgules ,

```
int    nb[2] = { 29, 34 };
```

```
char  tab[3] = { 'x', 'y', 'z' };
```

```
float prix[1] = { 22.0f };
```

```
string str[2] = { "bon", "test" };
```

## Pointeurs et tableaux

39

L'utilisation d'index peut être réalisée sous forme de pointeur

Un tableau déclaré

```
int  tab[4];
```

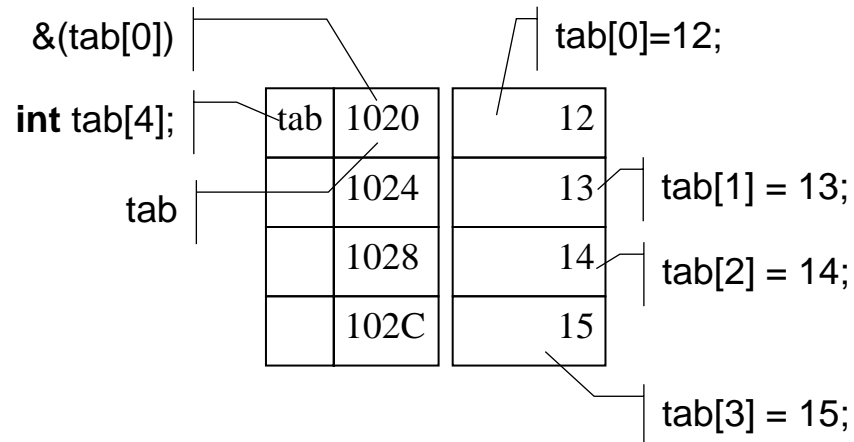
Définit un bloc mémoire de 5 entiers rangés consécutivement et nommés

```
tab[0], tab[1], tab[2], tab[3]
```

« tab[i] » désigne l'objet placé à la i<sup>ème</sup> position à partir du début

## Représentation mémoire

40



University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Relation tableau/pointeur

41

Soit un pointeur *pta* déclaré par

```
int * pta;
```

Initialisé à

```
pta = tab; //équivalent à  
pta=&(tab[0]);
```

*pta* pointe vers le 1<sup>er</sup> élément du tableau

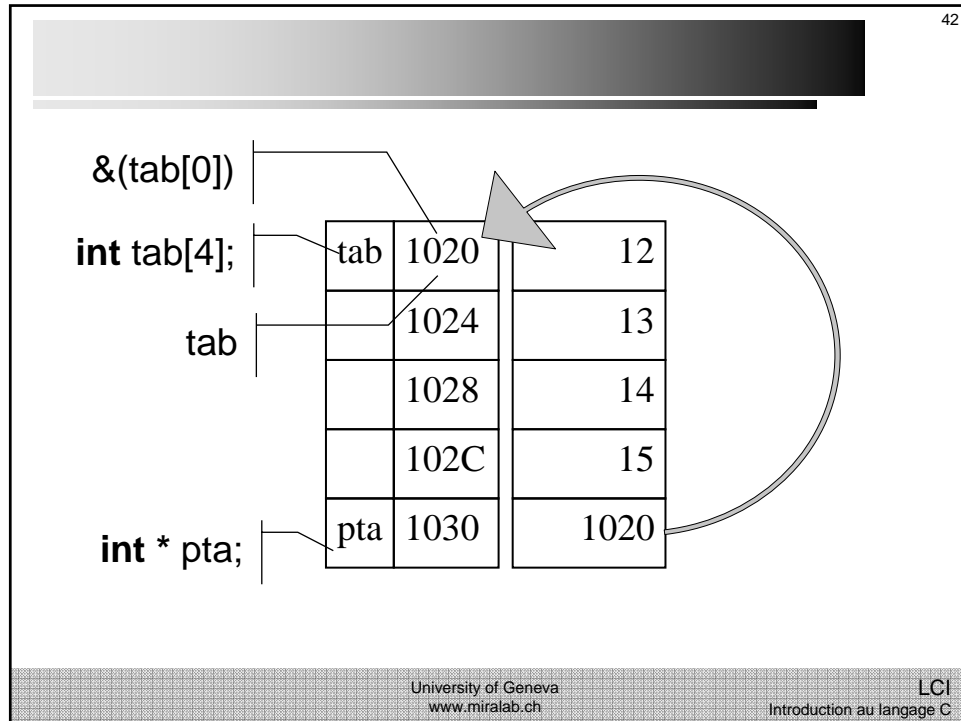
```
int z = *pta;
```

équivalent à

```
int z = tab[0];
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C



43

## Relation tableau/pointeur

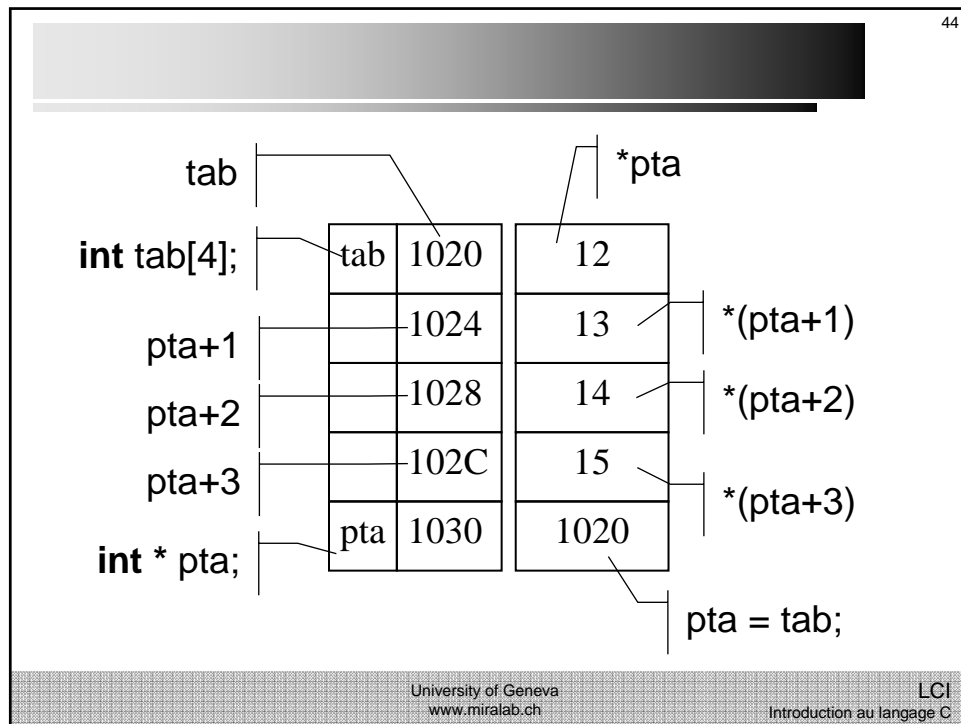
$pta+i$  pointe vers l'élément en  $i^{\text{ème}}$  position dans le tableau

$*(pta+i)$  correspond au contenu de  $tab[i]$

$pta+i$  à l'adresse de  $tab[i]$  soit  $\&(tab[i])$

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C



45

Tableaux à 2 dimensions .... la semaine prochaine ☺

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Exercices (tableaux)

University of Geneva  
www.miralab.ch

### Exercice 9 : tableaux

47

On déclare un tableau d'entiers A dimension 10

Deux sous-programmes, rempliA et impA, permettent de remplir et imprimer le tableau (sur la console).

Déclarer une référence vers l'élément 5 du tableau et incrémenter sa valeur

Déclarer un pointeur vers l'élément 7, incrémenter le pointeur et la valeur pointée

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C