

## Rappels

2

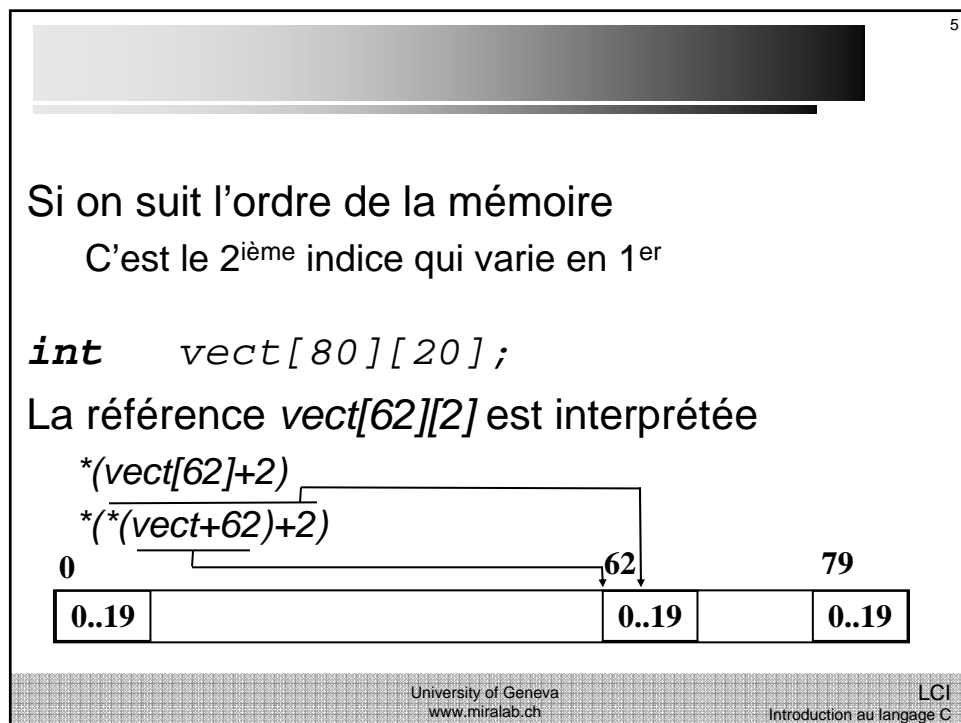
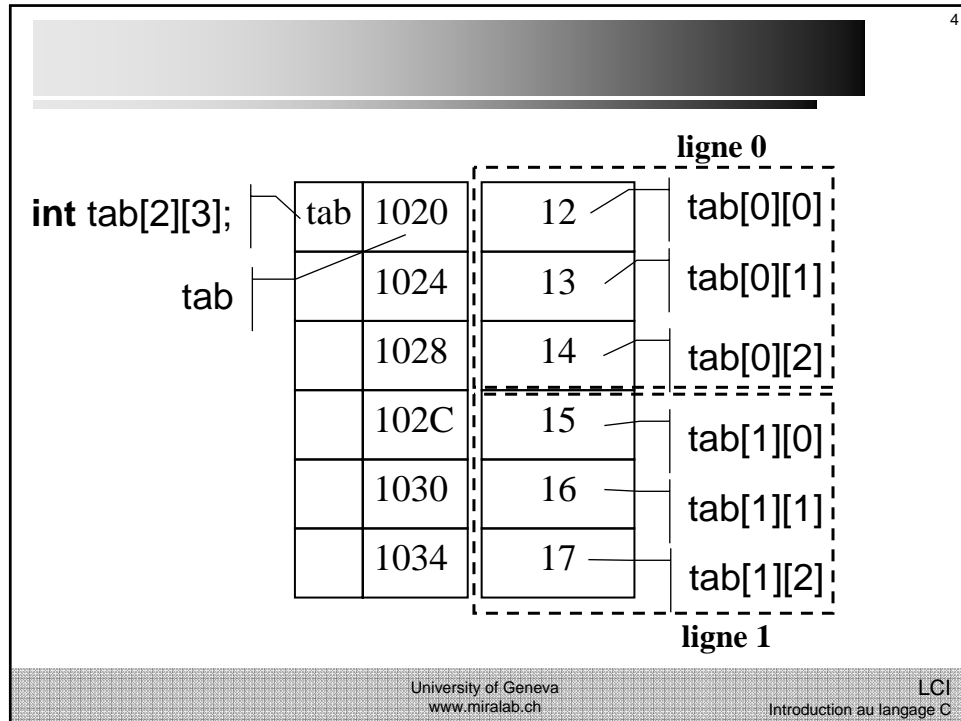
- Fonctions et procédures
- Passage de paramètres
- Tableaux

## Les tableaux multidimensionnels

3

2 dimensions, ou +, se déclarent  
de façon similaire aux tableaux  
unidimensionnels  
en précisant les dimensions de chaque  
coordonnée entre crochets `[]`

```
int    vect[80][20];  
char   tab[10][11];  
float  prix[30][2][50];
```



## Initialisation à la déclaration

6

Similaire au cas unidimensionnel

Chaque dimension séparée par accolades {}

```
int vect1[2][3]= { {0, 1, 5},
                  {7, 3, 2} };

int vect2[3][2][4]=
{
    {{0,1,2,3}, {1,2,3,4}},
    {{5,6,7,8}, {7,8,9,1}},
    {{3,4,5,6}, {2,3,4,5}}
};
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

7

Accès à un élément

```
vect1[i][j]
```

Exemples

```
vect1[0][1] = 5
int x = vect1[0][1] * vect[0][0]
```

**ATTENTION :**

Il n'existe pas de « protection » concernant les indices des tableaux, ie il est possible d'écrire ou de lire des cases qui ne sont pas allouées !!!  
= crash ☹

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Tableau et argument de fonction

8

Un tableau peut être passé comme argument d'une fonction

Soit par indice

Soit par pointeur

2 déclarations différentes possibles

```
void traiter(int *t);
```

```
void traiter(int t[]);
```

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

9

On peut modifier le contenu des tableaux passés en arguments dans la fonction

Les tableaux **ne** sont **pas** passés par valeur  
... toujours par référence !!!

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Résumé : les tableaux

Permet de regrouper un grand nombre de valeurs sous la même variable

La mémoire doit être « allouée » avant l'utilisation du tableau (comme pour les variables)

Stocke toutes les valeurs à la suite (même pour un tableau à 2 dimensions)

Déclaration

*type nomdevariable [nb d'elements];*

On accède à la xx<sup>ème</sup> valeurs d'un tableau par `tableau[xx]` (affectation ou utilisation)

Les indices sont numérotés de **0** à **n-1**

## Boîte à trucs

Portée et persistance des variables

Fichier header

# Portée d'une variable

12

## Définition

= région d'un programme où une variable peut être référencée

## Variables locales

Portée = bloc de contrôle contenant la déclaration  
e.g. après la déclaration

## Variables globales

Portée = modules ou programme  
e.g. après la déclaration, souvent placée dans fichier .h

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

# Exemple

13

	z	somme	dernier	adernier	somme	premier_terme	
int N = 500;							//début de portée de N
int premier_terme(int somme) {							//début de portée de somme
int dernier= 1;							//début de portée de dernier
do {							
int adernier = dernier;							//déb. de p. de adernier
dernier = somme;							
somme = adernier + dernier;							
} while (somme <= N);							//fin de portée de adernier
}							//fin de portée de dernier,somme
int main() {							
int somme=1;							//début de portée de somme
somme = premier_terme(somme);							
cout<<"Premier terme de la suite de " <<"Fibonacci supérieur à " << N <<" vaut "<< somme <<endl;							// fin de portée de somme
}							// fin de portée de N,
							// et de la fonction premier_terme()

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## Variable temporaire

14

Conserve sa valeur seulement dans la portée de la variable

Mémoire associée est

Dynamiquement allouée en début de portée

Libérée en fin de portée par le compilateur

Durée de vie limitée

## Variable statique

15

Conserve sa valeur hors de sa portée

Mot clef *static* lors de la déclaration

Indique ce type de variable

**int** tempsomme; //variable temporaire

**static int** statsomme; //variable statique

Initialisée qu'une seule fois !

Exemple : conserver le résultat dans un fonction tout au long de l'exécution du programme

... à utiliser avec modération !!! Préférer les paramètres.

## Variable globale et persistance

*pour information !!*

16

### La persistance d'une variable globale

Correspond à une allocation mémoire statique

Elle est initialisée une seule fois

Elle conserve sa valeur hors de sa portée

#### Différence

Pas besoin du mot clef *static*

Ce mot clef a une autre signification dans ce cas

### Statique

Indique que sa visibilité reste confinée au module

Les autres modules n'ont donc pas accès à cette variable

### Non-statique

Accessible par les autres modules

Lors de l'édition des liens

Re-déclaration (.h) utilise mot clef *extern*

## Autres mots clefs : *const*

17

Indique une constante

La valeur ne peut pas être modifiée après  
l'initialisation

```
const int n = 2;
```

```
n = 4; //erreur de compilation
```

Utile pour protéger des paramètres de toute  
modification lors d'appel de fonction



18

```

void erreur(const string & msg)
{
    cerr << "Erreur : " << msg << endl;
    msg = "c'est pas gagné!"; //erreur
}

```

### Caractéristiques

Passage par référence (rappel C++)

évite de copier la string = optimisation

Paramètre constant

assure que *msg* ne sera pas modifiée

19

## Appel en utilisant const ...

```

void erreur(const string& msg);

void main(void)
{
    string test = "si ca marche!";
    erreur(test);
}

```

**... maintenant vous connaissez  
Les bases de C++ ☺**

**Appliquons ces connaissances !**

University of Geneva  
www.miralab.ch

## Décomposition en fonction

ou l'art de découper un programme

21

- On ne décompose pas un programme juste pour faire plaisir à l'enseignant ou à l'assistant ☺
- La décomposition en fonction doit être réfléchie et correspondre à un besoin
  - Vais-je réutiliser ce bloc d'instruction ?
  - Avec quels paramètres variants ?
  - Comment simplifier mon problème ?
  - Comment éviter d'écrire 10 fois le même code ?

University of Geneva  
www.miralab.ch

LCI  
Introduction au langage C

## La réutilisabilité

22

- La réutilisabilité n'est pas un concept nouveau
- Les types de données à stocker sont toujours construite sur les mêmes bases (tables, listes, ensembles)
- La plupart des traitements comportent des actions atomiques telles que l'insertion, la recherche, le tri ... qui sont des pbs résolus en informatique
- Les bibliothèques (systèmes, mathématiques, etc...) sont des bons exemples de réutilisabilité et sont couramment utilisées par les programmeurs

⇒ Utilisation de librairies

## Décomposition

23

... ça ne s'apprend pas !!!

L'expérience permet de concevoir des décompositions plus « optimale »

Néanmoins

Si on souhaite définir un module, celui-ci ne doit pas être « fixe » en affichant un message défini à l'utilisateur par exemple, mais simplement retourner un résultat

Un bloc d'instructions doit être conçu afin d'être réutilisable

## Exemple : exercice 7

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 ! »

Variables globales ?  
Affichage avec l'utilisateur dans une fonction ?

### Proposition

Demander à l'utilisateur son poids et sa taille
Calculer l'IMC
En fonction de l'IMC afficher un message différent

## Solution

26

```
#include <iostream>
#include <math.h>

using namespace std;

float calculIMC(float poids, float taille) {
    return (poids / (taille*taille));
}

int main() {
    float p,t;

    cout << "Saisir votre poids (kg): ";
    cin >> p;
    cout << "Saisir votre taille (m): ";
    cin >> t;

    cout << "Votre IMC est de " << calculIMC(p,t) << endl;
    return(0);
}
```

IMC = fonction prenant 2 paramètres (poids et taille) et retournant l'IMC

Saisie des valeurs

Calcul et affichage du résultat

## 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(xp,yp) et Q(xq,yq)):

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

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() ☺

## Exercice 8

28

### Boite à outils 3D

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

Définir une structure pour les vecteurs : 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

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

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 9 : tableaux

29

On déclare un tableau d'entiers A dimension 10

Deux sous-programmes, remplitA 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