

Introduction à la programmation en C++

Les tableaux statiques

Nicolas Audebert

Vendredi 29 septembre 2017



Supports de cours

Site du cours : <http://imagine.enpc.fr/~monasse/Info/>

Slides : <https://nicolas.audebert.at/teaching/>

Rendus de TP et des exercices

Les rendus se font sur **Educnet**.

1. Le code rendu **doit compiler**.
2. Le code rendu doit **être propre** (indentation, noms de variables clairs).
3. Le code rendu doit **être commenté** (réponses aux questions, fonctionnement du code).

Rappels

Déclaration et définition

Spécificités des tableaux

La librairie Imagine++

TP

Déclaration

```
type nom_de_variable;
```

Affectation

```
nom_de_variable = valeur;
```

Type

Une variable ne possède qu'un seul et unique type.

Nativement, C++ connaît

int, **char**, **float**, **double**, **bool**.

Initialisation

Une variable déclarée n'ayant subi aucune affectation est dite non initialisée et peut prendre n'importe quelle valeur.

Expression booléenne

Une expression booléenne est une affirmation pouvant se réduire à une valeur de type `bool` pouvant être soit `true`, soit `false`.

if(...) then ... else ...

```
if(expression booléenne){  
    ...  
} else {  
    ...  
}
```

switch

Cf. chapitre 3.

La règle des accolades

Une variable n'existe que dans le bloc (et les sous-blocs), défini par des accolades, dans lequel elle a été déclarée.

Comparaison Python/C++

```
# Python
a = 3
b = 6
if a < b:
    # Début du sous-bloc
    c = a + b
    print(c)
    # Qui s'arrête ici
print(c)

// C++
int a = 3, b= 6;
if(a < b){
    // Début du sous-bloc
    int c = a + b;
    cout << c << endl;
    // Qui s'arrête ici
}
cout << c << endl; // Erreur
```

Variables globales

À utiliser avec parcimonie.

Boucle while (tant que)

```
while(expression){...}
```

```
do {...} while (expression);
```

Boucle for (pour)

```
for (initialisation; expression; iteration) { ... }
```

Signature

La **signature** d'une fonction est sa spécification au sens mathématique. Elle contient son nom, le type de valeur retournée (ou **void**) et les types de ses arguments.

Exemple

```
double puissance (double valeur, int exposant);
```

Passage par référence

Par défaut, les arguments d'une fonction lui sont passés par copie lorsque celle-ci est appelée. Si l'on veut modifier un ou plusieurs arguments, il faut les passer par référence avec le symbole **&**.

Rappels

Déclaration et définition

Spécificités des tableaux

La librairie Imagine++

TP

Les tableaux permettent

1. d'**éviter la multiplication** des variables,
2. de donner une **structure** aux données (par exemple, coordonnées d'un vecteur),
3. de **parcourir** rapidement un ensemble d'éléments.

```
double coor_x, coor_y, coor_z
```

```
14.2  0.65  1.0
```

```
double coordonnees[3];
```

0	1	2
14.2	0.65	1.0

Les tableaux statiques sont caractérisés par deux propriétés :

- ▶ Le **type** de leurs éléments,
- ▶ Leur **taille** (qui est donc fixée à l'avance par une constante).

Conséquences

- ▶ Un tableau ne peut contenir des éléments que d'**un seul et même type**,
- ▶ La taille d'un tableau statique **ne peut pas changer**.

Déclaration

```
type nom_tableau[taille];
```

Initialisation

```
int mon_tableau[10];  
// Initialisation manuelle à l'aide d'une boucle  
for(int i=0; i < 10; i++){  
    mon_tableau[i] = 5;  
}  
  
// Déclaration et initialisation directe  
double tableau_reel[5] = {2, 3.2, 9.76, 6, 1000};  
  
// Déclaration puis initialisation directe  
bool tableau_bool[3];  
tableau_bool = {true, true, false};
```

Comparaison avec les listes Python

Les tableaux en C++ sont plus proches des tableaux numpy que des listes Python.

```
# Python
tab1 = [0,0,0,0,0]
tab1[2] = 5

tab2 = ["test", 10, True]

t = 6
tab3 = [0 for i in range(t)]
tab3.append(100)
```

```
// C++
int tab1[5] = {0,0,0,0,0};
tab1[2] = 5

bool tab1 = {"a", 10, True};
// Erreur ! Tous les éléments
↪ doivent être booléens

int t = 6
int tab3[t]; // Erreur !

const int t = 6;
int tab3[t]; // OK: t constant
tab3.append(100) // ERREUR
// Un tableau ne change PAS de
↪ taille
```

Si n est la taille du tableau, alors les indices vont de 0 à $n-1$.

Attention

Tenter d'accéder à un élément hors de ces bornes résultera systématiquement en une erreur lors de l'exécution du programme.

Exemple

```
const int n = 100; // Taille du tableau (constante)
char tab[n]; // Déclaration du tableau
tab[0] = 'a'; // OK
tab[10] = 'd'; // OK
tab[n-1] = 'k'; // OK
tab[n] = 'f'; // ERREUR
tab[-1] = 'z'; // ERREUR
```

Les tableaux occupent de l'espace en mémoire (une case mémoire par élément du tableau, initialisé ou non). Il est donc nécessaire de les utiliser à bon escient.

```
// Calcul de 2^99
// Le tableau stocke tous les
  ↳ résultats temporaires
int t[100];
t[0] = 1;
for(int i = 1; i < 100; i++){
    t[i] = t[i-1] * 2;
}
cout << t[99] << endl;
```

```
// Calcul de 2^99
// (sans tableau)
int r = 1;

for(int i = 1; i < 100; i++){
    r *= 2;
}
cout << r << endl;
```

Rappels

Déclaration et définition

Spécificités des tableaux

La librairie Imagine++

TP

Une fonction peut manipuler un tableau dans sa signature :

```
void affiche(int t[5]){
    for(int i=0; i<5; i++){
        cout << t[i] << " ";
    }
    cout << endl;
}
```

```
void affiche(int t[], int
    ↪  taille){
    for(int i=0; i<taille; i++){
        cout << t[i] << " ";
    }
    cout << endl;
}
```

La première solution ne traite que le cas où la taille du tableau est toujours la même pour tous les tableaux à traiter. La seconde solution est à préférer car elle réutilisable avec des tableaux de différentes tailles, **en passant celle-ci en argument.**

Passage par référence

- ▶ Un tableau est **toujours** implicitement passé par référence (il ne faut donc pas rajouter de `&`).
- ▶ Une fonction ne peut pas retourner de tableau.

```
const int taille = 10;
double tab[taille];
init(tab);
affiche(tab);
// 0 0 0 0 0 0 0 0 0 0
```

```
void init(double t[], int
↪  taille){
    for(int i=0; i<taille; i++){
        t[i] = 0;
    }
}
```

On ne peut pas copier directement des tableaux entre eux.

```
int t1[4] = {1,2,3,4}, t2[4];  
t2 = t1 ; // ERREUR : pas d'affectation avec le = pour les  
↪ tableaux
```

Seule solution : itérer sur les éléments.

```
int t1[4] = {1,2,3,4}, t2[4];  
for(int i = 0; i < 4; i++){  
    t2[i] = t1[i];  
}
```

De même, pour tester l'égalité entre deux tableaux.

Rappels

Déclaration et définition

Spécificités des tableaux

La librairie Imagine++

TP

La librairie Imagine++ contient des fonctions toutes prêtes pour réaliser de nombreuses opérations :

- ▶ **Common**
fonctions et classes basiques (Timer, Color...)
- ▶ **LinAlg**
algèbre linéaire (inversion de matrices...)
- ▶ **Graphics**
affichage (fenêtre 2D/3D, dessin...)
- ▶ **Images**
classe Image et traitements d'image

Exemple d'un programme simple avec Imagine++

```
#include<iostream>
#include <Imagine/Graphics.h>
using namespace std;
using namespace Imagine;

int main(){
    int xc = 128, yc = 128, t = 0, rayon; // init variables
    openWindow(256, 256); // Ouverture de la fenetre
    while (true) { // Boucle principale
        rayon = 10 * cos(t/1000); // mise a jour du rayon
        fillCircle(xc, yc, rayon, RED); // Affichage du disque
        milliSleep(20); // Temporisation
        fillCircle(xc, yc, rayon, WHITE); // Effacement du
        ↪ disque
        t++; // incremente le temps
    }
    endGraphics();
    return 0;
}
```

Il est possible d'ouvrir et de travailler avec plusieurs fenêtres graphiques.

```
// Première fenêtre
openWindow(256,256);
fillCircle(128,128,50,RED);

// Seconde fenêtre
openWindow(256,256);
fillCircle(128,128,50,BLUE);

// Impossible de revenir
↪ dessiner dans
// la première fenêtre car
// elle n'a pas de nom
```

```
// Première fenêtre
Window window1 =
↪ openWindow(256,256);
fillCircle(128,128,50,RED);
// Seconde fenêtre
Window window2 =
↪ openWindow(256,256);
fillCircle(128,128,50,BLUE);

setActiveWindow(window1);
fillCircle(128,128,50,GREEN);
setActiveWindow(window2);
fillCircle(128,128,50,BLACK);

// Fermeture d'une fenêtre
closeWindow(window1);
```

Le site du cours → Installation Imagine++ → Instructions

<http://imagine.enpc.fr/~monasse/Imagine++/>

Imagine++

Main Page	Related Pages	Modules	Namespaces	Classes	Files	Examples	
-----------	---------------	---------	------------	---------	-------	----------	--

Imagine++ Libraries - version 4.2.0

The most up to date version of this documentation should be on the website: <http://imagine.enpc.fr/~monasse/Imagine++/>

Introduction

Imagine++ is a set of libraries developed at the Imagine group (<http://imagine.enpc.fr>). Initially designed for students and beginners, and though it is still easy enough for them, Imagine++ is now used daily by Imagine researchers. It consists in different modules. Four of them are publicly available:

- The **Common Library**, providing basic types and utilities.
- The **LinAlg Library**, providing linear algebra types and algorithms
- The **Graphics Library**, providing convenient 2D and 3D displays
- The **Images Library**, providing image containers and algorithms

Some useful links:

- The Imagine++ home page is <http://imagine.enpc.fr/~monasse/Imagine++>
- A quick start guide
- This C++ course for beginners (in French...) makes extensive use of Imagine++, especially of the Graphics module: <http://imagine.enpc.fr/~monasse/Info/>
- For any question, feel free to contact monasse@imagine.enpc.fr

Installation

- **Windows installation**

Le site du cours → Installation Imagine++ → Instructions

<http://imagine.enpc.fr/~monasse/Imagine++/>

void	Imagine::drawRect (const IntPoint2 &p, int w, int h, const Color &col, int penWidth=1, bool xorMode=false) Rectangle (IntPoint2). More...
void	Imagine::drawString (int x, int y, const std::string &s, const AlphaColor &col, int fontSize=12, double alpha=0, bool italic=false, bool bold=false, bool underlined=false, bool xorMode=false) String. More...
void	Imagine::drawString (const IntPoint2 &p, const std::string &s, const AlphaColor &col, int fontSize=12, double alpha=0, bool italic=false, bool bold=false, bool underlined=false, bool xorMode=false) String (IntPoint2). More...
void	Imagine::enableMouseTracking (bool en) Mouse tracking. More...
void	Imagine::endGraphics () Terminate graphics application. More...
void	Imagine::fillCircle (int xc, int yc, int r, const AlphaColor &col, bool xorMode=false) Filled Circle. More...
void	Imagine::fillCircle (const IntPoint2 &c, int r, const AlphaColor &col, bool xorMode=false) Filled Circle (IntPoint2). More...
void	Imagine::fillEllipse (int x, int y, int w, int h, const AlphaColor &col, bool xorMode=false) Filled Ellipse. More...
void	Imagine::fillEllipse (const IntPoint2 &p, int w, int h, const AlphaColor &col, bool xorMode=false) Filled Ellipse (IntPoint2). More...

Le site du cours → Installation Imagine++ → Instructions

<http://imagine.enpc.fr/~monasse/Imagine++/>

```
void Imagine::fillCircle ( int      xc,  
                          int      yc,  
                          int      r,  
                          const AlphaColor & col,  
                          bool      xorMode = false  
                        )
```

Fills a circle.

Parameters

xc,yc center
r radius
col AlphaColor or Color
xorMode XOR drawing (default=off). Used twice, recovers the original content

```
fillCircle(330,43,30,YELLOW); // filled circle
```

Examples:

Graphics/test/example.cpp, and Graphics/test/test.cpp.

Rappels

Déclaration et définition

Spécificités des tableaux

La librairie Imagine++

TP

Mastermind

- ▶ Utilisation des tableaux
- ▶ Algorithmie
- ▶ Fonctions graphiques

Rendu

À rendre **avec l'exercice individuel** au plus tard le **4/10** sur **Educnet**.

