

# Information, Calcul, Communication (partie programmation) : Exceptions / Synthèse

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle  
Faculté I&C

# Rappel du calendrier

	MOOC	décalage / MOOC	exercices prog. 1h45 Jeudi 8-10	cours prog. 45 min. Jeudi 10-11
1 11.09.25	--	-1	prise en main	Bienvenue/Introduction
2 18.09.25	1. variables	0	variables / expressions	variables / expressions
3 25.09.25	2. if	0	if – switch	if – switch
4 02.10.25	3. for/while	0	for / while	for / while
5 09.10.25	4. fonctions	0	fonctions (1)	fonctions (1)
6 16.10.25		1	fonctions (2)	fonctions (2)
- 23.10.25				
7 30.10.25	5. tableaux (vector)	1	vector	vector
8 06.11.25	6. string + struct	1	array / string	array / string
9 13.11.25		2	structures	structures
10 20.11.25	7. pointeurs	2	pointeurs	pointeurs
11 27.11.25		-	entrées/sorties	entrées/sorties
12 04.12.25		-	erreurs / exceptions	erreurs / exceptions
13 11.12.25		-	révisions	théorie : sécurité
14 18.12.25	8. étude de cas	-	révisions	Révisions

# Objectifs du cours d'aujourd'hui

- ▶ Rappels sur les exceptions
- ▶ Synthèse de la partie programmation du cours
- ▶ Étude(s) de cas (ou questions ?) sur les exceptions ?

# Exceptions

- ▶ anticiper des erreurs d'utilisation d'une portion de programme (typiquement d'une fonction)
- ▶ principe de base:  
prévoir une erreur à un endroit et de la gérer à un **autre** endroit
- ▶ throw d'un côté  
try/catch de l'autre
- ▶ bien comprendre les déroulements possibles
- ▶ ne pas en abuser (utiliser à bon escient)



# Qu'avons nous vu en programmation ?

Quoi ?

Programmer, c'est **décomposer** une **tâche** à automatiser en une **séquence d'instructions** (**traitements**) et des **données**

Algorithme	S.D.A.
Traitements	Données
Expressions & Opérateurs Structures de contrôle Fonctions	Variables  Portée Chaînes de caractères Tableaux statiques Tableaux dynamiques Structures Pointeurs  Entrées/Sorties Exceptions

Il faut que vous **soyez au clair** sur chacune des notions ci-contre :

1. Qu'est-ce que c'est ?
2. À quoi ça sert ?
3. Comment ça s'utilise ?

# FONDAMENTAUX

## 1. **déclarez avant d'utiliser**

- ▶ variables (pensez à les initialiser)

```
int i(0);
```

```
vector<double> v;
```

- ▶ fonctions  **prototype**

```
double sin(double x);
```

```
bool cherche_valeur(Listechaine l, Valeur v);
```

## 2. **passages** (par valeur, par référence[, par référence constante])

## 3. choisir le **type** approprié

## 4. **modularisez** / décomposez / pensez « atomique »

4.1 **conception** (qu'est ce qu'on veut ?) *Q VOI !!*

4.2 **implémentation** (comment ça se réalise ?)

4.3 **syntaxe** (comment ça s'écrit ?)

4.4 **tests** (où sont mes fautes, comment pourrais-je les tester ?)



# Proposition de méthode de révision

- ▶ prendre le tableau synthétique du [transparent 5](#)
- ▶ prendre les fiches résumé
- ▶ et pour chacun des points, se demander si on sait :
  - ▶ de quoi ça parle ?
  - ▶ ce que ça veut dire ?
  - ▶ l'utiliser ?

👉 se focaliser sur les **concepts**.


Les détails de syntaxe (comment ça s'écrit) peuvent être **ensuite** rapidement retrouvés dans la fiche résumé, si on sait ce qu'on cherche (c'est-à-dire si on a le concept)

▶ pratiquer, pratiquer, pratiquer

par soi-même, seul(e) (sans « tricher » = sans aide, sans corrigé, ...)

Noter ses points de blocage/ses difficultés et revoir ensuite les concepts, puis exercices correspondants

# Conseils pour examen(s)

- 
- ▶ **lisez** la donnée et faites ce qui est demandé  
ni plus (perte de temps), ni moins (perte de points)
  - ▶ montrez **ce que vous savez**, pas ce que vous ne savez pas :
    1. ne perdez pas du temps à essayer de répondre à une question dont vous savez que vous ne connaissez pas le sujet (perte de temps, voire de points)
    2. et cf point ci-dessus : ne pas répondre à un aspect demandé alors que vous le connaissez est une perte de points
  - ▶ **gérer** correctement **votre temps**
    1. ne ratez pas des questions auxquelles vous savez répondre
    2. ne perdez pas trop de temps sur une question qui ne rapporte pas beaucoup de points





③ [5 points] Qu'affiche l'extrait de code ci-dessous ? Si nécessaire, mettez « ??? » pour toute valeur inconnue ou « **Segmentation fault** » si vous pensez que le programme plante (à cet endroit).

Justifiez vos réponses (vous pouvez aussi annoter le code).

```
int i(3);
int* p(&i);
int** q(&p);

**q = 25;
p = &i;

int j(i - 10);

cout << "a. " << *p << endl;
cout << "b. " << j << endl;

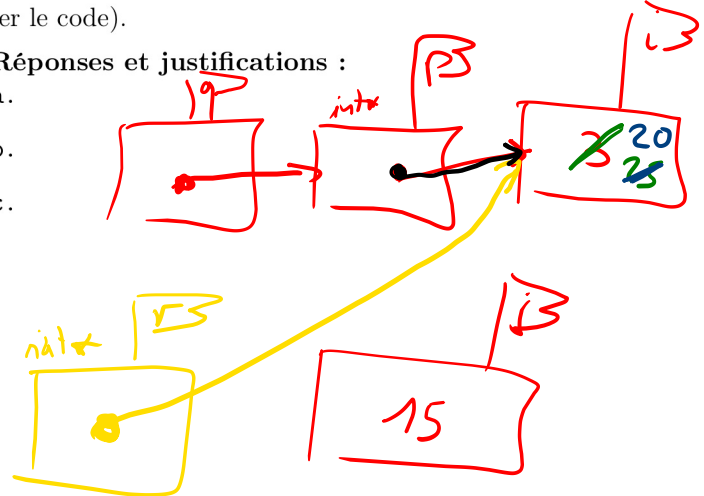
i = *p;

int* r(*q);

i = i - *r + 20;
*q = &i;
cout << "c. " << *r - **q << endl;
```

Réponses et justifications :

- a.  
b.  
c.



Ne pas écrire dans cette zone.

$$\text{if} (p == \text{nullptr}) \text{ throw "Erreur"s;}$$

④ [2.5 points] Si un nombre décimal  $q$  s'écrit 0.0001 en binaire à virgule fixe, comment s'écrit  $\log_2(q)$  en binaire entier signé sur 6 bits ? Justifiez votre réponse.

Réponse et justification :

suite au dos ➡