Information, Calcul, Communication (partie programmation) : Structures de contrôle en C++ (2) : boucles / itérations

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectifs de la leçon d'aujourd'hui

- Rappels :
  - **structures de contrôle** en C++ : boucles, itérations
  - et sur la portée
- Complément sur les structures de contrôle : les **sauts**
- Etudes de cas
- Questions

Rappel du calendrier

		MOOC	décalage / MOOC	exercices prog. 1h45 Jeudi 9-11	cours prog. 45 min. Jeudi 11-12
1	23.09.21	--	-1	prise en main	Bienvenue/Introduction
2	30.09.21	1. variables	0	variables / expressions	variables / expressions
3	07.10.21	2. if	0	if – switch	if – switch
4	14.10.21	3. for/while	0	for / while	for / while
5	21.10.21	4. fonctions	0	fonctions (1)	fonctions (1)
6	28.10.21		1	fonctions (2)	fonctions (2)
7	04.11.21	5. tableaux (vector)	1	vector	vector
8	11.11.21	6. string + struct	1	array / string	array / string
9	18.11.21		2	structures	structures
10	25.11.21	7. pointeurs	2	pointeurs	pointeurs
11	02.12.21		-	entrées/sorties	entrées/sorties
12	09.12.21		-	erreurs / exceptions	erreurs / exceptions
13	16.12.21		-	révisions	théorie : sécurité
14	23.12.21	8. étude de cas	-	**Examen final (2h45)**	
					(ne sont pas sur le MOOC)

Les différentes structures de contrôle

On distingue 3 types de structures de contrôle :

les branchements conditionnels : *si ... alors ...*

Si $\Delta = 0$
  $x \leftarrow -\frac{b}{2}$
Sinon
  $x \leftarrow \frac{-b-\sqrt{\Delta}}{2}$, $y \leftarrow \frac{-b+\sqrt{\Delta}}{2}$

les boucles conditionnelles : *tant que ...*

Tant que pas arrivé
  avancer d'un pas

Répéter
  poser la question
jusqu'à réponse valide

les itérations : *pour ... allant de ... à ...* , *pour ... parmi ...*

$$x = \sum_{i=1}^{5} \frac{1}{i^2}$$

$x \leftarrow 0$
Pour i de 1 à 5
  $x \leftarrow x + \frac{1}{i^2}$

Jean-Cédric Chappelier
& Jamila Sam

Boucles et itérations

Les boucles permettent la mise en œuvre **répétitive** d'un traitement.

La répétition est **contrôlée** par une **condition de continuation**.

- boucles conditionnelles *a priori*

```
while (condition) {
    Instructions
}
```

- boucles conditionnelles *a posteriori*

```
do {
    Instructions
} while (condition);
```

- itérations générales (« à la C »)

```
for (initialisation ; condition ; mise_à_jour)
```

- itérations sur des ensembles (C++11)

```
for (déclaration : ensemble)
```

☞ plus tard (tableaux)

Rappels sur la portée

La **portée** d'une variable c'est l'ensemble des lignes de code où cette variable est accessible / est définie / existe / a un sens.

- les variables déclarées à l'intérieur d'un bloc sont appelées **variables locales** (au bloc). Elles ne sont accessibles qu'à l'intérieur du bloc.
- les variables déclarées en dehors de tout bloc (même du bloc `main(){}`) seront appelées **variables globales** (au programme). Elles sont accessibles dans l'ensemble du programme.
- en cas d'ambiguïté : résolution à la portée la plus proche.

Conseils :

1. Ne jamais utiliser de variables globales (sauf peut être pour certaines constantes).
2. Déclarer les variables au plus près de leur utilisation.
3. Evitez d'utiliser le même nom pour des variables différentes.

Jean-Cédric Chappelier
& Jamila Sam

Sauts : `break` et `continue`

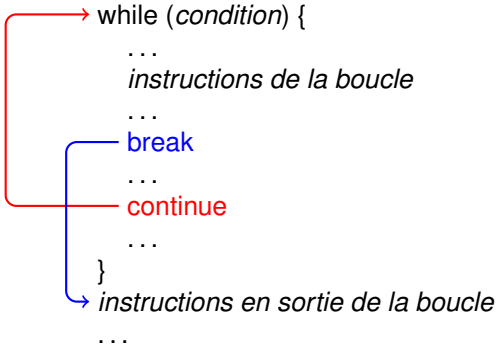
C++ fournit deux instructions prédéfinies, `break` et `continue`, permettant de contrôler de façon plus fine le déroulement d'une boucle.

- Si l'instruction `break` est exécutée au sein du bloc intérieur de la boucle, l'exécution de la boucle est interrompue (quelque soit l'état de la condition de contrôle) ;
- Si l'instruction `continue` est exécutée au sein du bloc intérieur de la boucle, l'exécution du bloc est interrompue et la condition de continuation est évaluée pour déterminer si l'exécution de la boucle doit être poursuivie.
  Dans le cas d'un `for` la partie mise à jour est également effectuée (avant l'évaluation de la condition).

Conseil : En toute rigueur on n'aurait **pas besoin** de ces intructions, et tout bon programmeur évite de les utiliser.

> Pour la petite histoire, un bug lié à une mauvaise utilisation de `break;` a conduit à l'effondrement du réseau téléphonique longue distance d'AT&T, le 15 janvier 1990. Plus de 16'000 usagers ont perdu l'usage de leur téléphone pendant près de 9 heures. 70'000'000 d'appels ont été perdus.
> [P. Van der Linden, *Expert C Programming*, 1994.]

Instructions `break` et `continue`



Instruction `break` : exemple

Exemple d'utilisation de `break` :
une **mauvaise** (!) façon de simuler une boucle avec condition d'arrêt

```
while (true) {
  Instruction 1;
  ...
  if (condition d'arrêt)
      break;
}
autres instructions;
```

Question : quelle est la bonne façon d'écrire le code ci-dessus ?

Instruction `continue` : exemple

Exemple d'utilisation de `continue` :

```
{
  int i(0);
  while (i < 100) {
    ++i;
    if ((i % 2) == 0) continue;
    // la suite n'est exécutée que pour les
    // entiers pairs ?/impairs ?
    Instructions;
    ...
  }
}
```

Question : quelle est une meilleure façon d'écrire le code ci-dessus ?
(on suppose que *`Instructions; ...`* ne modifie pas la valeur de `i`)

Jean-Cédric Chappelier
& Jamila Sam

Les structures de contrôle

les branchements conditionnels : *si ... alors ...*

```
if (condition)
    instructions
.....................
if (condition 1)
    instructions 1
...
else if (condition N)
    instructions N
else
    instructions N+1
```

```
switch (expression) {
   case valeur:
       instructions;
       break;
   ...
   default:
       instructions;
}
```

les boucles conditionnelles : *tant que ...*

```
while (condition) {
    Instructions
}
```

```
do {
    Instructions
} while (condition);
```

les itérations : *pour ... allant de ... à ...* , *pour ... parmi ...*

```
for (initialisation ; condition ; increment)
    instructions
```

```
for (déclaration : valeurs)
    instructions
```

les sauts : `break;` et `continue;`

Note : `instructions` représente une instruction élémentaire ou un bloc.
`instructions;` représente une suite d'instructions élémentaires.

Jean-Cédric Chappelier
& Jamila Sam

Etudes de cas

- lien avec ICC : début du calcul des $\binom{n}{k}$ (version programmation dynamique)
- jeu de devinette : trouver (par dichotomie) le nombre imaginé