

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 ?

```
do {  
    Instruction 1;  
    ...  
} while (not (condition d'arrêt));  
autres instructions;
```

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`)

```
for (int i(1); i < 100; i += 2) {  
    Instructions;  
    ...  
}
```

Etude de cas : les $\binom{n}{k}$

Comment commencer le calcul des $\binom{n}{k}$ tel que vu en cours ICC ?

(version « programmation dynamique » ; on ne fait ici que le début, nous continuerons lorsque nous aurons vu les tableaux en C++)

Rappel du problème :

- ▶ on nous donne n et k
- ▶ on veut calculer $\binom{n}{k}$ par deux boucles sur le « triangle de Pascal » :

A Pascal's triangle diagram with indices i and j . The vertical axis is labeled i with a red arrow pointing downwards from 0 to n . The horizontal axis is labeled j with a blue arrow pointing from 0 to k . The triangle contains the following values:

	0						
	1						
	1	1					
	1	2	1				
	1	3	3	1			
	1	4	6	4	1		
	1	5	10	10	5	1	

Etude de cas : les $\binom{n}{k}$

- ▶ On va donc commencer par une boucle en i (de 1 en 1) :
`for (int i(...); i ...; ++i)`
- ▶ Dans laquelle on aura une boucle en j (de 1 en 1) :
`for (int j(...); j ...; ++j)`

Quelles bornes pour i ?

☞ entre 0 et n

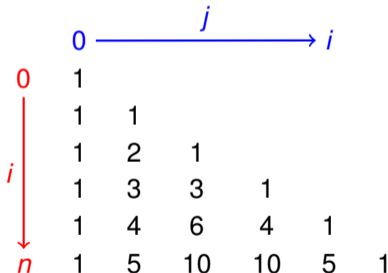
Quelles bornes pour j ?

☞ entre 0 et i

On a donc à ce stade :

```
for (int i(0); i <= n; ++i) {
    for (int j(0); j <= i; ++j) {

    }
}
```



Etude de cas : les $\binom{n}{k}$

Peut-on faire mieux ?

➡ Oui : il **n'**est **pas** nécessaire d'aller au delà de k pour la boucle en j

Donc veut donc que j soit inférieur à i , mais sans non plus dépasser k .

Comment cela s'écrit-il ?

$(j \leq i) \text{ or } (j \leq k)$

ou bien

$(j \leq i) \text{ and } (j \leq k)$

?

or ou and ?

Il est souvent difficile de correctement choisir entre la conjonction `and` et la disjonction `or`. Quelques pistes :

- ▶ tout d'abord que veut-on : que la condition soit vraie ou qu'elle soit fausse ?
 - ☞ Ici on veut *continuer* tant qu'elle est *vraie*
 - ▶ parfois il est plus facile de passer par la contraposée (sa négation) :
 - ☞ Ici on veut *s'arrêter* dès que la condition est *fausse* ;
dès que j est plus grand que i ou k (...donc sa négation : et)
 - ▶ tester pour un cas ambigu.
 - ☞ Ici : tester dans un cas où justement j est entre k et i pour i plus grand que k (puisque c'est bien ces cas là que nous voulons optimiser)
dans ce cas ($k < j < i$) :
 - ▶ $(j \leq i) \text{ or } (j \leq k)$ est vraie (`true or false`), la boucle continuera donc ; ce qui n'est pas ce que nous voulons ;
 - ▶ $(j \leq i) \text{ and } (j \leq k)$ est fausse (`true and false`), la boucle aura donc été arrêtée ; ce que nous voulons.
- ☞ Ici c'est donc bien « $(j \leq i) \text{ and } (j \leq k)$ »

Etude de cas : les $\binom{n}{k}$

On obtient donc :

```
for (int i(0); i <= n; ++i) {  
    for (int j(0); (j <= i) and (j <= k); ++j) {  
        // ...  
    }  
}
```

On pourrait aussi écrire (avec `#include <algorithm>`) :

```
for (int i(0); i <= n; ++i) {  
    for (int j(0); j <= min(i, k); ++j) {  
        // ...  
    }  
}
```

Reste à voir quoi mettre dans la boucle...

...ce que nous aborderons plus tard une fois vus les tableaux.

Devinette

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

int main()
{
    int min(1);
    int max(100);

    cout << "Pensez à un nombre entre "
         << min << " et " << max
         << endl;

    // boucle conditionnelle a posteriori
    do {
        const int pivot( (min+max) / 2 );

        cout << "Votre nb est il < > = à "
             << pivot << " ? ";

        char rep;
        cin >> rep;
```

```
        if (rep == '>') {
            min = pivot;
            if (max - min == 1) min = max;
        } else if (rep == '<') {
            max = pivot;
        } else {
            min = max = pivot;
        }
    } while (min < max);

    cout << "J'ai trouvé : " << min
         << endl;

    return 0;
}
```