8

Boucle infinies

Dans une boucle for, que se passe-t-il si la condition ne devient jamais fausse? Par exemple :

```
for (int i = 0; i >=0; i++){ cout << i << endl; }
```

Cette boucle sera exécutée pour `i = 0, 1, 2, 3, 4, 5, 6, ...`

Si la condition est toujours vraie (ne devient jamais fausse), les instructions se trouvant dans la boucle seront répétées indéfiniment et la variable `i` prendra toutes les valeurs positives que le type `int` peut représenter.

Quelle est la plus grande valeur positive que le type `int` peut prendre (représenter)? Pour pouvoir comprendre la réponse à cette question, vous devez tout d'abord comprendre ce qui suit : quelles sont les combinaisons (de 0 et 1) possibles sur 3 bits?

Binaire	Décimal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Nombre de combinaisons = $2^{\text{nombre de bits}}$. Par exemple, nombre de combinaisons sur 3 bits = $2^3 = 8$ combinaisons possibles. Maintenant, quelle est la plus grande valeur positive qu'on peut représenter sur 3 bits? La plus grande valeur positive qu'on peut représenter sur 3 bits est : 111 (en binaire).

$111_{(2)} = 7_{(10)}$

On peut obtenir cette valeur de la manière suivante : $2^{\text{nombre de bits}} - 1$

Pour 3 bits, on obtient : $2^3 - 1 = 8 - 1 = 7$

Pourquoi on soustrait 1 dans cette formule ($2^{\text{nombre de bits}} - 1$)? Parce que les valeurs commence à partir de zéro. Par exemple, sur 3 bits, on a 8 combinaisons possibles. La première est prise par le zéro, donc, le plus grand nombre qu'on peut représenter sur 3 bits et 7 et non pas 8 (0, 1, 2, 3, 4, 5, 6 et 7).

Supposons maintenant que le type `'int'` (entier) prend 1 octet comme espace mémoire. La question est : quelle est la plus grande valeur positive qu'on peut représenter sur 1 octet (i.e., 8 bits)?

$2^{\text{nombre de bits}} - 1 = 2^8 - 1 = 256 - 1 = 255$

Comme vous le savez déjà les entiers peuvent être positifs ou négatifs. Donc, par exemple, pour pouvoir représenter les nombres négatifs dans le cas d'un espace mémoire de 3 bits :

Binaire			Décimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

une combinaison sera prise par le zéro et il nous reste 7 combinaisons; 4 pour les nombres négatifs et 3 pour les nombres positifs, comme suit :

```
-4,   -3,   -2,   -1,   0,   1,   2,   3
```

Reprenons le même exemple, mais supposons que les entiers sont représentés sur 1 octet (i.e., 8 bits). On a 2^8 combinaisons possibles, i.e., 256 combinaisons. Une combinaison sera prise par le zéro. Il nous reste 255 combinaisons 128 pour les nombres négatifs et 127 pour les nombres positifs:

```
-128, -127,  .....,      -2,   -1,   0,   1,   2, ...,      126,  127
```

En général, avec 4 octets (32 bits), on a 2^{32} valeurs (combinaisons) différentes, c'est-à-dire `4,294,967,296: 0 à 4,294,967,295.` Cependant, un nombre entier peut être négatif ou positif, donc, on divise 2^{32} par deux pour représenter les nombres négatifs et positifs. $(2^{32} / 2) = 2^{31} = 2{,}147{,}483{,}648$. On obtiendra, ainsi, 2^{31} valeurs pour les nombres positifs et 2^{31} valeurs pour les nombre négatifs. Pour représenter le zéro, on prend une des valeurs des nombres positifs. Par conséquent, les valeurs possibles qu'un nombre entier (32 bits) peut prendre : `-2,147,483,648 à 2,147,483,647`

Revenons à notre exemple de boucle: `for (int i = 0; i >= 0; i++){ cout << i << endl; }`

On a dit que cette boucle sera exécutée pour `i = 0, 1, 2, 3, 4, 5, 6, ...` Si la condition est toujours vraie, les instructions se trouvant dans la boucle seront répétées indéfiniment et la variable `i` prendra toutes les valeurs positives que le type `int` peut représenter (de 0 jusqu'à la plus grande valeur positive). Pour mieux comprendre, copier et lancer le programme suivant :

```
int main(){
	for (char i = 0; i >= 0; i ++){
		cout << "i = " << (int)i << endl;
	}
}
```

La notion d'infini n'existe pas en Informatique. En exécutant le programme ci-dessus, vous avez bien remarqué que les entiers sont limités. Pour voir combien d'octets votre machine réserve pour les entiers (en mémoire), écrivez le morceau de code suivant :

```
int main(){
	cout << "sizeof(int) = " << sizeof(int) << endl;
}
```

Vous pouvez obtenir la taille des autres types (`double, char, bool, ...`) en modifiant `int` par le type désiré dans le programme ci-dessus.