Les tableaux de bits

Vous avez déjà rencontré la classe `std::bitset` dans le chapitre [Logique binaire et calcul booléen](#), pour afficher une séquence de bits.

main.cpp

```
#include <iostream>
#include <bitset>

int main() {
    std::cout << "0b" << std::bitset<8>(0b101010) << std::
endl;
    std::cout << "0b" << std::bitset<8>(42) << std::endl;
}
```

affiche :

```
0b00101010
0b00101010
```

Ce chapitre détaille l'utilisation de cette classe `std::bitset` et les notions de *flag* et *mask*. Les notions vues dans les chapitres [Logique binaire et calcul booléen](#) et [\[Aller plus loin\] L'algèbre de Boole](#) seront utilisées, n'hésitez pas à les relire si nécessaire.

Créer un `std::bitset`

La classe `std::bitset` est une classe template prenant en argument le nombre de bits. La taille d'un `std::bitset` est donc déterminée à la compilation (tout comme `std::array`).

```
std::bitset<TAILLE>
```

La méthode la plus simple pour créer un `std::bitset` est de lui fournir une valeur entière lors de l'initialisation, de préférence en utilisant une

représentation binaire (avec le préfixe `0b`) ou hexadécimale (avec le préfixe `0x`). Si aucune valeur n'est fournie, `std::bitset` est initialisé avec la valeur nulle.

main.cpp

```
#include <bitset>

int main() {
    const std::bitset<8> b;
    const std::bitset<8> b8 { 0b101010 };
    const std::bitset<16> b16 { 0xA1B2 };
}
```

Pour faciliter la création de `std::bitset` à partir d'une entrée utilisateur (flux standard, fichier, etc), il est également possible d'initialiser un `std::bitset` à partir d'une chaîne de caractères. Cette chaîne peut être constituée des caractères "0" et "1", ou de n'importe quelle paire de caractères. Dans ce dernier cas, il faut fournir à `std::bitset` plusieurs arguments supplémentaires. Dans l'ordre :

- la position dans la chaîne du premier caractère à lire (les caractères précédents sont ignorés) ;
- le nombre de caractères à lire ;
- le caractère qui représente "0" ;
- le caractère qui représente "1".

Notez qu'avec une littérale chaîne de caractères, il ne faut pas fournir la position du premier, la littérale est lue depuis le début.

main.cpp

```
#include <iostream>
#include <bitset>

int main() {
    const std::bitset<8> b1 { "101010" };
    std::cout << "0b" << b1 << std::endl;

    const std::bitset<8> b2 { "ABAABB", 6, 'A', 'B' };
    std::cout << "0b" << b2 << std::endl;
}
```

```
const std::string s { "BABBBBA" };
const std::bitset<8> b3 { s, 0, 6, 'A', 'B' };
std::cout << "0b" << b3 << std::endl;
}
```

affiche :

```
0b00101010
0b00010011
0b00101110
```

Afficher un std::bitset

Lors de l'affichage d'un `std::bitset`, les bits sont affichés en utilisant les caractères 0 et 1 et le nombre de caractères correspondra à la taille du `std::bitset`, quelque soit les caractères et leur nombre, utilisés pour initialiser le `std::bitset`.

Avec `to_string`: utiliser d'autres caractères.

Mask et flag

tester un bit : `mask`, `flag`, opérateur ET bit à bit, `test()` forcer un bit : OU bit à bit

tester plusieurs bit : `count`, `all`, `any`, `none` (cf algo)

Plusieurs fois des données de même type. Accès avec un indice, partant de 0. Taille fixé à la compilation ou à l'exécution : `bitset` à la compilation (`vector<bool>` à l'exécution).

Accès a un élément : `[]` Validation taille : `assert` connaitre la taille : `size`

Exercices

A partir de la représentation binaire d'un nombre (42 =

0b0000000000101010)

- Compter le nombre de 1 dans la représentation
- trouver la plus longue chaîne de 1 dans la représentation

Chapitre précédent	Sommaire principal	Chapitre suivant
------------------------------------	------------------------------------	----------------------------------