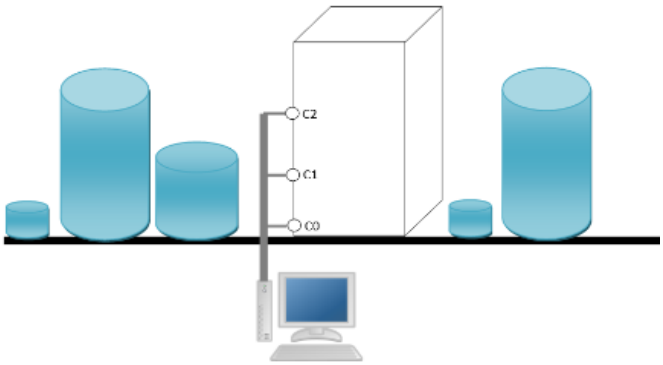
COMMENT EXTRAIRE DES BITS OU OCTETS UTILES D'UNE VARIABLE EN C++ ? [1]

Un programme C++ peut être amené à devoir extraire certains bits ou ensemble de bits d'un mot.

Le principe consiste à utiliser les masques et les décalages.

Prenons l'exemple suivant pour illustrer ces propos.

Dans un entrepôt, on est amené à effectuer le tri de colis. Ces derniers sont classés en trois catégories : les petits, les moyens et les grands. Pour effectuer le tri, on a mis en place des capteurs qui permettent de récupérer la taille des boites comme le montre l'exemple ci-contre.



Les capteurs sont connectés à une carte d'entrées/sorties du PC. Afin de tester la validité du tri, il faut réaliser un programme de test permettant d'afficher la nature de la boite qui est passée dans le détecteur en tenant compte de l'octet récupéré sur la carte d'entrées/sorties.

La fonction de lecture du port d'entrées/sorties (que vous n'avez pas à étudier ici) permet de récupérer un octet organisé de la façon suivante :

octet état capteurs :

?	?	?	C2	C1	C0	?	?

Quand le capteur concerné est actionné, le bit est à 1.

Exemple 1 : EXTRACTION D'UN BIT PRECIS

On récupère un octet sur le port qui vaut 0x3A soit en binaire 0011 1010 et on veut connaître l'état du bit C2.

Principe	Codage en C++

octetLu : 0 0 1 1 1 0 1 0
 	
Pour isoler le bit C2, on fait un **ET_BIT_A_BIT** avec un octet dont tous les bits sont à 0 sauf celui correspondant à la position de C2
 	
masque : 0 0 0 1 0 0 0 0
 	
On obtient:
 	
bitC2 : 0 0 0 1 0 0 0 0
 	
En faisant un décalage de 4 bits à droite, on obtient :
 	
bitC2 : 0 0 0 0 0 0 0 1
 	
Maintenant, la variable bitC2 vaudra 1 si C2 valait 1 et 0 sinon.
	`// Simulation de l'extraction du bit C2`
	`// Définition des variables de type octet positif`
	`unsigned char octetLu, masque, bitC2;`
	
	`// on simule ici le résultat fourni par le capteur`
	`// 0x permet d'écrire de l'hexadécimal`
	`// On ne peut pas saisir du binaire en C++`
	`octetLu = 0x3A;`
	
	`masque = 0x10; // seul le bit associé à C2 est à 1`
	`// on pouvait aussi écrire masque=16;`
	
	`bitC2 = octetLu & masque; // ET BIT A BIT`
	
	`bitC2 >>= 4; // Décalage de 4 bits à droite`
	
	`cout << "bit C2=" <<(int) bitC2 << endl;`

[1] Utile notamment pour l'exercice 9 sur les fonctions

Exemple 2 : EXTRACTION D'UN GROUPE DE BITS

Supposons maintenant que l'on souhaite récupérer une variable (un octet) dont les 3 bits de poids faible regroupent l'état des 3 capteurs C2, C1 et C0

Variable à récupérer :		0	0	0	0	0	C2	C1	C0

Si on reprend l'exemple précédent (octetLu=0x3A), on doit récupérer %00000110 (binaire) ou 0x06 (hexadécimal) ou 6 (décimal)

Principe	Codage en C++

octetLu : 0 0 1 1 1 0 1 0
 	
Pour isoler les 3 bits C2,C1 et C0, on fait un ET_BIT_A_BIT avec un octet dont tous les bits sont à 0 sauf ceux associés à C2, C1 ou C0
 	
masque : 0 0 0 1 1 1 0 0
 	
On obtient:
 	
capteurs : 0 0 0 1 1 0 0 0
 	
En faisant un décalage de 2 bits à droite, on obtient :
 	
capteurs : 0 0 0 0 0 1 1 0
 	
Maintenant, la variable capteurs vaudra 6 (ou 0x06 en hexa) si le mot lu était 0x3A
	`// Simulation de l'extraction des 3 bits C2,C1 et C0`
	`// Définition des variables de type octet positif`
	`unsigned char octetLu, masque, capteurs;`
	
	`// on simule ici le résultat fourni par le capteur`
	`// 0x permet d'écrire de l'hexadécimal`
	`// On ne peut pas saisir du binaire en C++`
	`octetLu = 0x3A;`
	
	`masque = 0x1C; //les bits associés à C2,C1,C0 sont à 1`
	`// on pouvait aussi écrire masque=28;`
	
	`capteurs = octetLu & masque; // ET BIT A BIT`
	
	`capteurs >>= 2; // Décalage de 2 bits à droite`
	
	`cout << "capteurs =" <<(int) capteurs << endl;`

ANNEXE :

Pour forcer un bit à 1, faire un OU BIT A BIT avec 1

Pour forcer un bit à 0, faire un ET BIT A BIT avec 0

Table de vérité du ET

a	b	a.b
0	0	0
0	1	0
1	0	0
1	1	1

Si a est un bit

a.0 ➔ 0 (force à 0 le bit)

a.1 ➔ a (ne change pas le bit)

Table de vérité du OU

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

Si a est un bit

a+0 ➔ a (ne change pas le bit)

a+1 ➔ 1 (force à 1 le bit)