**Au sujet de l’affichage de la grille à l’écran.**

┌─┬─┬─┬─┬─┬─┬─┬─┬─┐

│ │ │ │ │ │ │ │ │ │ 0

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 1

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 2

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 3

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 4

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 5

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 6

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 7

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

│ │ │ │ │ │ │ │ │ │ 8

└─┴─┴─┴─┴─┴─┴─┴─┴─┘

On m’a demandé si on doit tout taper les char(xxx) dans la déclaration du tableau.

Il y a de l’économie d’écriture en faisant des copier-coller.

Si vous regardez la présentation conceptuelle de la grille « vide » (j’ai mis des espaces au lieu de chiffres), vous remarquerez qu’il y a des choses qui se répètent. Dans ce « dessin » de type char, il n’y a que quatre sortes de rangées. La première et la dernière rangée sont uniques. Cependant, les autres reviennent à 9 et 8 fois chaque.

Si on décortique chaque type de rangées, là aussi on peut voir de l’économie d’écriture (en faisant un peu de copier-coller, encore une fois).

**Rangée du haut**

┌─┬─┬─┬─┬─┬─┬─┬─┬─┐

┌ ─┬ ─┬ ─┬ ─┬ ─┬ ─┬ ─┬ ─┬ ─ ┐

Il y des exceptions au début et à la fin, mais il y a un pattern de 2 caractères qui se répète 8 fois.

**Rangées où se trouveront les chiffres.**

│ │ │ │ │ │ │ │ │ │

Je mets un X pour indiquer l’espace, pour la rendre plus visible.

│X│X│X│X│X│X│X│X│X│

│X │X │X │X │X │X │X │X │X │

Il y a une exception à la fin mais il y a un pattern de 2 caractères qui se répète 9 fois.

**Rangées entre les rangées où se trouvent les chiffres.**

├─┼─┼─┼─┼─┼─┼─┼─┼─┤

├ ─┼ ─┼ ─┼ ─┼ ─┼ ─┼ ─┼ ─┼ ─ ┤

Il y des exceptions au début et à la fin, mais il y a un pattern de 2 caractères qui se répète 8 fois.

**Rangée du bas**

└─┴─┴─┴─┴─┴─┴─┴─┴─┘

└ ─┴ ─┴ ─┴ ─┴ ─┴ ─┴ ─┴ ─┴ ─ ┘

Il y des exceptions au début et à la fin, mais il y a un pattern de 2 caractères qui se répète 8 fois.

On peut donc créer assez aisément un tableau de caractères pour générer le dessin.

Il faudra faire attention lorsqu’on voudra mettre les chiffres aux bons endroits.

N’oubliez pas que le tableau de type uneGrille a short comme type de base. Ce sont des numériques. Le tableau qui contient le dessin a char comme type de base.

Les valeurs de 0 à 9 et les caractères de ‘0’ à ‘9’ ne sont pas codés pareillement. Le type short a 16 bits et le type char en a 8,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Short | En binaire |  | char | En binaire | En décimal |
| 0 | 0000000000000000 |  | ‘0’ | 00110000 | 48 |
| 1 | 0000000000000001 |  | ‘1’ | 00110001 | 49 |
| 2 | 0000000000000010 |  | ‘2’ | 00110010 | 50 |
| 3 | 0000000000000011 |  | ‘3’ | 00110011 | 51 |
| 4 | 0000000000000100 |  | ‘4’ | 00110100 | 52 |
| 5 | 0000000000000101 |  | ‘5’ | 00110101 | 53 |
| 6 | 0000000000000110 |  | ‘6’ | 00110110 | 54 |
| 7 | 0000000000000111 |  | ‘7’ | 00110111 | 55 |
| 8 | 0000000000001000 |  | ‘8’ | 00111000 | 56 |
| 9 | 0000000000001001 |  | ‘9’ | 00111001 | 57 |
|  |  |  |  |  |  |

Comment convertir de short à char ?

Il y a plusieurs trucs que l’on peut utiliser.

Par exemple :

char T[10] = { ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’ } ;

Quelle est la valeur obtenue si on demande T[4] ?

Si K contient une valeur entre 0 et 9, inclusivement, quelle est la valeur de l’expression suivante : **char( 48 + K )** ?

Si on déclare la variable Z comme suit :

**char Z ;**

**Z = ‘0’ + K ;**

**cout << Z ;**

**Et si on voulait faire la conversion inverse ?**

Avec le tableau **T[ ]** ci-dessus, faites faire une recherche dans le tableau pour le caractère voulu, l’indice où on le trouve est sa valeur numérique. (Rechercher ‘9’ devrait nous donner l’indice 9.)

L’expression **short( caractere – ‘0’ )** vous donne la valeur numérique (pas son encodage) du caractère. Ditto pour **short( caractere – 48 )**.

On peut aussi utiliser des opérations bit-à-bit.

Si vous comparez les bits 0, 1, 2 et 3 des short et des char pour nos dix chiffres, vous verrez que c’est pareil.

Convertir un chiffre (char) en son « équivalent » short.

short masque15 = 15 ; // 00001111

short N = caractere & masque15 ; // ‘0’ <= caractere <= ‘9’

Convertir un short en son « équivalent » chiffre (char).

short masque48 = 48 ; // 00110000

char caractere = char( K | masque48 ) ; // 0 <= K <= 9

**Remarque #1**

Plusieurs des conversions sont facile à concevoir si on connait comment les codes sont fait pour chacun des types. C’est un genre de connaissance utile.

**Remarque #2**

Pour en revenir à l’affichage du dessin, l’utilisation d’un tableau de type char n’est pas obligatoire. La relativement grande régularité dans les termes ferait qu’on pourrait s’en tirer avec l’utilisation de boucles. Il faut juste faire attention aux cas d’exceptions.