Les formats

Jean-Christophe Dubacq

S1 2016

1 Les séquences de codage

1.1 Les formats complexes

1.1.1 Types simples ou composés

Q1 Identifiez dans les types suivants lesquels sont susceptibles d'êtres des types de base et lesquels sont plutôt des types construits par assemblage :

- Un nombre entier positif ou nul
- Un nombre complexe
- Un point dans l'espace
- Un nombre avec un très grand nombre de chiffres non fixé à l'avance
- Un intervalle
- Une date
- Un étudiant (nom, prénom, date de naissance)
- Un caractère
- Une chaîne de caractères

Un nombre entier positif ou nul est en général un type simple, de même que un caractère. Pour tout le reste, ce sont des types composés. Un nombre complexe, un point dans l'espace sont des tableaux de nombres (2 ou 3), de même qu'un intervalle. Une date, c'est souvent une série de nombres, même si on peut se fixer un point de référence et compter par exemple le nombre de secondes depuis ce point de référence. Une chaîne de caractères ou un nombre avec un très grand nombre de chiffres non fixé à l'avance, c'est une liste d'élements simples (caractères ou chiffres). Quant à l'étudiant, ce sont des types de données hétérogènes, donc forcément composées.

1.1.2 Une date

Q2 Décrivez à partir de quels éléments on peut composer une donnée qui représente un moment précis de la journée. *Voir plus loin pour une correction possible. Le fuseau horaire se discute (et dans ce cas, c'est une chaîne de caractères ou un code dans une liste).*

Q3 Discutez les éléments précis selon que l'on considère qu'un moment est pris à la seconde près ou beaucoup plus précis. *Le nombre de secondes peut être soit un entier, soit un flottant.*

1.2 La représentation en mémoire

1.2.1 Stockage d'une date (suite)

Une date est composée des éléments suivants :
— Une année (disons de -2 milliards à +2 milliards)
— Un mois, un jour du mois
— Un fuseau horaire qui est une « adresse »
— Une heure, une minute (entiers)
— Un nombre de secondes qui est un flottant simple précision

Q4 Dites quels sont les types de base du C à utiliser pour coder cette information, d'après les limites connues de stockage pour chaque type. Utilisez les tailles les plus petites possibles.
Un `int` 32 bits pour l'année, des entiers très courts (donc `char` ou `unsigned char`) pour mois, jour du mois, heure, minute. Une adresse pour le fuseau horaire qui est en fait une chaîne de caractères. Et un `float` pour les secondes.

Q5 Donnez leur noms à la fois dans un modèle 32 bits et un modèle 64 bits.
Voir dans le tableau, mais `int` remplit bien sauf pour ILP64 (il s'appelle `_int32`.

Q6 Si on avait voulu aller de -5 milliards à +5 milliards d'années, quel type aurait-on dû utiliser ?
Un entier sur 64 bits donc un `long int` (voire un `long long int` sous Microsoft).

Q7 Si les données sont dans l'ordre indiqué dans un type composé, précisez à quel moment les contraintes d'alignement provoquent des « trous » dans la structure.
On commence par 4 octets, puis 2 fois un octet. Puisqu'une adresse fait 4 ou 8 octets, il faut là un trou qui fait 2 octets (alignement sur 4 octets) ou 6 octets (alignement sur 8 octets).
Après, on a encore 2 octets (des `char` suffisent pour heures et minutes), suivi d'un `float` donc sur 4 octets. Encore 2 octets de trous sont donc nécessaires.

Q8 Quelle est la taille totale de la structure (avec les trous) ?
En 32 bits pour une adresse : 4+2+2+4+2+2+4=20 octets.

En 64 bits pour une adresse : 4+2+2+8+2+2+4=24 octets.
Si on veut aller jusqu'à 5 milliards, il faut rajouter 4 octets à chaque fois.

1.2.2 Stockage d'une date (suite)

Q9 On veut stocker la date du 24 décembre -4 à 21h45 dans un système 32 bits. Calculez les valeurs à stocker en hexadécimal (hormis l'adresse du fuseau horaire). Vous prendrez comme valeur d'adresse pour le fuseau horaire 0x12345678.
L'an -4 c'est `0xFFFFFFFC`*. Le mois, c'est* `0xC`*, le jour c'est* `0x18`*, l'heure c'est* `0x15` *et les minutes c'est* `0x0x2D`*. Les secondes c'est 0, et 0 en float c'est* `0x00000000`*.*

Q10 Écrivez, les uns après les autres, les octets qui composent cette date si on est dans un système 32 bits *big-endian*
`FF FF FF FC 0C 18 ?? ?? 12 34 56 78 15 2D ?? ?? 00 00 00 00`*. Les* `??` *sont du bourrage (ça peut être n'importe quelle valeur).*

Q11 Écrivez, les uns après les autres, les octets qui composent cette date si on est dans un système 32 bits *little-endian*
`FC FF FF FF 0C 18 ?? ?? 78 56 34 12 15 2D ?? ?? 00 00 00 00`*. Les* `??` *sont du bourrage (ça peut être n'importe quelle valeur).*

1.3 La compression

1.3.1 Information intrinsèque

Q12 Si vous lancez une pièce de monnaie 20 fois en l'air, est-ce que vous avez plus de chance de tomber sur 20 fois face (F), 10 fois face-pile (FP), ou sur FPFPPFFFPPPPFPFFPPFP ?

Q13 Quelle est la quantité d'information contenue dans la suite binaire 110111001001 ? Et dans la suite binaire 000000000000 ? Et dans la suite binaire 010101010101 ?
C'est la même probabilité pour les suites, et la même quantité de bits (12 bits).

Q14 Est-ce qu'on pourrait écrire certaines de ces suites de façons plus courtes ? Proposez-en.
Par exemple, 12 fois 0, 6 fois 01. Et pour le premier, $A1\overline{A}1$*, avec* $A = B0B$*, avec* $B = CC$*, avec* $C = 1$*. Donc par exemple* $AC\overline{A}C$*,* $A = B\overline{C}B$*,* $B = CC$*,* $C = 1$*. Pas vraiment plus cours* `:-/`*.*

1.3.2 Compression RLE

Q15 Voilà des suites à compresser avec la compression RLE :
— 000011110010100001111111
— 1112112111112213122111311222111132132 11
— 0110100110010110

1.3.3 Protocole de fax

Q16 Un fax est une suite de points noirs et blancs. En utilisant les notations du RLE binaire données dans le cours, décrivez l'image à droite. On part en haut à gauche de l'image, on part à droite et on reprend à la fin de chaque ligne à la ligne suivante à gauche.