Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes embarqués 1 : TP.04 Introduction au language C

Horner Frédéric / Pharisa Valentin, HEIA Fribourg le 06.11.2017

Synthèse

Non-acquis :
Le type enum, le pragma once avec le #ifndef, les différences fondamentales entre les classes Java et les structures du C

Acquis mais à exercer :
Le #define, les extern et static

Parfaitement acquis :
L'utilisation des entêtes

Questions

1. **Peut-on se passer des fichiers d'entête en C ? Si oui, comment ? Si non, pour quelle raison ?**
   Oui, c'est une convention. On utilise les « includes » pour appeler les entêtes dans plusieurs fichiers différents. Sans l'utilisation des entêtes et donc des « includes », nous sommes forcés de tout taper à main et ce dans chaque fichier.
2. **Quelle est l'utilité du pragma #pragma once dans les fichiers d'entête ? Doit-il être accompagné d'une autre directive ? Si oui, laquelle ?**
   Le « pragma once » est une directive utilisée afin d'éviter l'inclusion multiple de fichiers d'entête. Dans notre TP, il est toujours accompagné du #ifndef.
3. **Que faut-il placer dans un fichier d'entête ?**
   Les fonctions et variables qui seront utilisées par la suite dans l'ensemble de nos structures
4. **Quelle est l'utilité des mots-clef extern et static ?**
   Extern : fonction utilisable dans toutes les structures
   Static : fonction utilisable uniquement dans la structure actuelle
5. **Comment faut-il procéder pour définir une constante en C ?**
   Il faut mentionner le « const »
   Exemple : int const a = 2 ;
6. **Quelle(s) différence(s) existe-t-il entre les instructions #define MAX 10 et const int MAX=10; ?**
   #define = macro soit un enchainement d'instructions, on peut y ranger des valeurs mais aussi des instructions, ne prend pas de place en mémoire et intervient à la phase de préprocessing
   const = variable qui garde une valeur fixe, prend aussi de la place en mémoire et intervient lors de la phase de compilation
7. **Comment peut-on définir une énumération en C ? Quelle est son utilité ?**
   Le type « enum » nous permet de définir nos propres types ainsi que leurs valeurs. Cela ressemble à une structure en C ou une classe en Java. On créé un objet et on l'initialise.
   Ex : enum jour {LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI, DIMANCHE};
   enum jour j1, j2;
   j1 = LUNDI;

j2 = MARDI;

8. **Quelle(s) différence(s) existe-t-il entre une structure en C (struct S{}) et une classe en Java (class C{}) ?**
   Il n'y a pas d'héritage entre structure, autrement c'est un peu flou.
9. **Comment faut-il procéder pour définir un tableau en C ? Peut-on lui donner des valeurs initiales lors de sa définition ?**
   Comme en Java, on déclare le type, le nom et on précise sa taille. Oui, on peut lui donner des valeurs lors de sa définition.
   Ex : int tableau[4] = {0,1,2,3} ;
10. **Comment faut-il procéder pour obtenir le nombre d'éléments contenus dans un tableau ?**
    Cela n'est pas possible par défaut en C. On peut en revanche savoir la taille de ce dernier avec une boucle « for ».

Remarques

Le C est nouveau, nous espérons voir un peu de théorie durant le cours. Nous voudrions parler des éléments non-acquis du TP4.

Nombre d'heures passées

Pharisa Valentin : 5h- 6h

Horner Frédéric : 2h – 3h

Feedback

L'introduction au C a été intéressante mais très rapide. Nous sommes directement partis dans le vif du sujet- Il y a des similarités entre le Java et le C mais certaines choses restent encore floues et devront être traités ultérieurement en classe. Nous avons trouvé le résultat sympa avec ces digits défilants sur notre beaglebone.