Electronique Numérique

Systèmes à microprocesseur

S4 : Notion de Langage C

Frédéric Giamarchi
IUT GEII Nîmes
Centre Spatial Universitaire (Montpellier Nîmes)
Université de Montpellier
frederic.giamarchi@umontpellier.fr

Langage C et langage C++ (CPP)

AVANTAGES	INCONVENIENTS
Langage universel: il est utilisé dans de nombreux domaines et sur de nombreux processeurs	
Langage compact: il repose sur un nombre fini de fonctions et d'opérateurs simples	Certes compact mais peut rapidement devenir **incompréhensible** si les codes sources ne sont pas correctement commentées
Langage structuré, et **extensible** (rattachement de bibliothèques)	
Proche machine: il permet la manipulation de bits dans les données	
Langage indépendant du processeur donc normalement portable.	

Typage des variables

- Une variable en C est caractérisée par :
  - Son adresse : endroit où elle est stockée en mémoire
  - Son type : nature de l'information qu'elle contient
  - Son nom : nom donnée à la variable dans le programme
  - Sa valeur : contenu de la variable

int k = -5; char c = 'g'; float x = -45,32 uint8_t = 75;

- Chaque type de variable est codé sur un certain nombre d'informations binaires, limitant la valeur maximale qu'elle peut contenir.

Types de variables

type	variable	Taille	Plage de valeur	Langage C
bool	binaire	1 bit	0, 1 ou true, false	
char	caractère	8 bits	-128 à 127	classique
int8_t	entier signé	8 bits	-128 à 127	embarqué
uint8_t	entier non signé	8 bits	0 à 255	embarqué
int	entier	16 bits	-32 768 à 32 767	classique
int16_t	entier signé	16 bits	-32 768 à 32 767	embarqué
uint16_t	entier non signé	16 bits	0 à 65 535	embarqué
long	entier	32 bits	-2 147 483 648 à 2 147 483 647	classique
int32_t	entier signé	32 bits		embarqué
uint32_t	entier non signé	32 bits	0 à 4 294 967 295	embarqué
float	réel	32 bits	+/- 3,4^-38 à +/- 3,4^+38	

Constantes et tableau 1D

- Il est également possible de déclarer des constantes

```
#define   M     10
#define   PI    3,14
#define   NOM   "Julien"
```

- Les tableaux sont des regroupements indexés de N variables d'un même type, l'indice allant de 0 à N-1.
- Ils sont caractérisés par leur nom et le nombre de variables qu'ils peuvent stocker.

```
char tab[10] = "Bonjour";          uint8_t notes[5] = {10, 15, 12, 8, 13};
```

Opérateurs arithmétiques et logiques

Opérateurs	action
+, -, *, /	Opérateurs arithmétique de base
%	Reste de la division entière
==	Test d'égalité
!=	Test de différence
<, >, <=, >=	Test de comparaison
!, ~	Négation, complément
&	ET logique
|	OU logique
&&	Test avec ET logique
||	Test avec OU logique

Structure de Contrôle

- if (expression vraie) else if else

```
if ( x == 10)
{
        k = 1;
}
else if ( x < 5)
{
        k = -1;
}
else
{
        k = 0;
}
```

Si la première condition est vraie,
le premier bloc d'instructions est exécuté.

Sinon teste la condition suivante,
et exécute le bloc associé.

Sinon le dernier bloc d'instructions est exécuté.

Structure de Contrôle

- switch (variable) case break

```
switch (etat)
{
        case 0:
                k = 1;
        break;
        case 1:
                k = 2;
        break;
        default:
                etat = 0;
        break;
}
```

Sélectionne les actions à faire en fonction de la valeur d'une variable (etat)

switch → définit la variable à tester (etat)

case x → si variable (etat) vaut x

break → permet de sortir du switch

default → action en cas d'erreur sur la variable (etat)

Structure de Contrôle

- for (initialisation; condition à tester; incrémentation)

```
for (x = 0; x < 10; x++)
{
        k = k + 2;
}
```

Permet de répéter un certain nombre de fois des actions.

On définit le nombre de répétition par la gestion d'une variable.

On doit donner une valeur à cette variable au départ, puis définir la condition limite et indiquer comment on progresse de la valeur initiale à la la valeur finale.

Structure de Contrôle

- while (expression vraie)

```
while (1)
{
        liste des actions
}
```

Tant que l'expression est vraie, la liste des actions est exécutée en boucle.

- do while (expression vraie)

```
do
{
        liste des actions
}
      while ( x != 0 );
```

Exécute le contenu puis teste la condition.

Si la condition est vraie, alors reprend au début de la liste, sinon passe à la suite du programme

Liens

- Mbed home page
- Mbed library
- Mbed compiler
- Extra tutorials concerning Mbed and Nucleo boards