C

YuGiOhJCJ

22 août 2009

Table des matières

1 Avant propos...

Cette documentation a été rédigée par YuGiOhJCJ. Vous lisez actuellement la version 20090822 qui est gratuite. Si vous souhaitez utiliser une partie de cette documentation pour vos créations, veuillez d'abord me contacter à yugiohjcj@1s.fr. La version la plus récente de ce document est disponible à l'adresse http ://yugiohjcj.1s.fr/. Cette publication peut contenir certaines erreurs. N'hésitez pas à me les rapporter pour que j'effectue une correction.

2 Introduction

Le langage C est un langage de programmation impératif. Il est normalisé par l'ISO[1]. Quelques brouillons de ces normes peuvent être téléchargés sur http ://www.open-std.org/.

3 Le compilateur

Pour transformer le code source (écrit en langage C par le développeur) en exécutable (lancé par l'utilisateur), un compilateur doit être utilisé.

Par exemple, avec le compilateur gcc[2], voici la ligne de commande permettant la compilation :

```
$ gcc -o hello_world hello_world.c
```

4 Le préprocesseur

Avant de générer l'exécutable, les directives qui commencent par un «#» sont analysées par le préprocesseur.

Voici quelques directives destinées au préprocesseur :

directive	description
#include	inclure un fichier
#define	définir une constante
#ifndef	créer une condition «si la constante n'est pas définie»
#endif	mettre fin à une condition

5 Les en-têtes

Les fichiers en-têtes (ou «headers») ont une extension «.h». Ils contiennent des prototypes de fonctions.

Pour ajouter le fichier d'en-tête «stdio.h» qui contient le prototype de la fonction *printf*, par exemple, on procède ainsi :

```
#include <stdio.h>
```

[1] ISO : International Organization for Standardization, http ://www.iso.org/

[2] gcc : GNU Compiler Collection, http ://gcc.gnu.org/

6 Les commentaires

Il est possible d'ajouter des commentaires dans la source. Ceux-ci n'ont aucun effet sur le programme ; ils sont là pour donner des indications dans le fichier source.

Un commentaire sur une seule ligne :

```
// Exemple de programme en C
```

Un commentaire sur plusieurs lignes :

```
 /*----------------------------*
  | Exemple de programme en C   |
  | YuGiOhJCJ <yugiohjcj@1s.fr> |
  | http://yugiohjcj.1s.fr/     |
  *----------------------------*/
```

7 Les variables

On peut affecter des valeurs à des noms. Ce sont des variables. Celles-ci sont typées. C'est à dire qu'il faut préciser, lors de sa déclaration dans le code source du programme, le type de la variable. Voici quelques exemples de déclarations :

```
// Un entier
int a;
// Un caractère
char b;
// Une chaîne de caractères (taille : 24 caractères)
char c[24];
// Un nombre à virgule
float d;
```

Voici comment affecter une valeur :

```
// Un entier
a = 2;
// Un caractère
b = 'x';
// Un nombre à virgule
d = 3.14;
```

Pour les chaînes de caractères, il est nécessaire de faire appel à la fonction *strcpy*.

```
// Une chaîne de caractères
strcpy(c, "Ma chaîne de caractères");
```

Il est possible ensuite d'afficher les valeurs de ces variables à l'aide de la fonction *printf* :

```
printf("%d, %c, %s, %f", a, b, c, d);
```

8 Les constantes

Les constantes sont des variables qui ne peuvent être modifiées dans le programme. Il existe deux méthodes pour déclarer une constante.

La première utilise le mot clé «const» :

```
const int entier = 2;
```

La deuxième utilise la directive «#define» pour le préprocesseur :

```
#define entier 2
```

9 Les structures

Une structure regroupe des données. Par exemple, une personne peut être définie par une structure dont les membres sont trois variables qui représentent son nom, son prénom et son âge.

```
typedef struct personne{
char nom[20];
char prenom[20];
int age;
} Personne;
```

Il est possible ensuite d'affecter des valeurs aux membres de la structure :

```
Personne p1;
strcpy(p1.nom, "TOTO");
strcpy(p1.prenom, "Toto");
p1.age = 25;
```

10 Les énumérations

Une énumération est un ensemble fini de valeurs. Par exemple, un sexe peut être défini par une énumération dont les membres sont deux valeurs : masculin et féminin.

```
typedef enum sexe{
masculin,
feminin
} Sexe;
```

Il est possible ensuite d'affecter des valeurs à une énumération :

```
Sexe s1;
s1 = masculin;
```

11 Les structures conditionnelles

Comme beaucoup de langages de programmation, on dispose de structures conditionnelles :

```
if ( 1 == 2 ) {
printf("Ce texte ne s'affiche pas car la condition est fausse.");
}
else{
printf("Étant donné que la condition du if est fausse, "
"c'est le bloc de else qui est exécuté.");
}
while ( 1 == 1 ) {
printf("Ce texte s'affiche en continu car la condition est vraie.");
}
switch ( a ) {
case 1:
printf("Si la variable a est égale à 1.");
break;
case 2:
printf("Si la variable a est égale à 2.");
break;
default:
printf("Dans les autres cas.");
break;
}
for(b=0;b<10;b++){
printf("Passé dans la boucle %d fois.", b);
}
```

12 Les fonctions

Il existe un grand nombres de fonctions. Par exemple, la fonction *printf* est faite pour afficher du texte. Si l'on souhaite afficher le texte «Hello World» à l'écran, il faut faire un appel de fonction :

```
printf("Hello World!\n");
```

La fonction *main* est indispensable dans un programme. C'est elle qui est appelée en premier.

Parfois, il est nécessaire de créer sa propre fonction. Celle-ci peut posséder des arguments et peut retourner une valeur. Voici une fonction qui additionne deux valeurs :

```
int additionner(int premier, int deuxieme){
int somme;
somme = premier + deuxieme;
```

```
return somme;
}
```

Il est maintenant possible de l'appeler à partir de la fonction *main* :

```
int main(void){
int solution, a, b;
a = 2;
b = 3;
solution = additionner(a, b);
printf("%d", solution);
return(0);
}
```

13 Un exemple

```
 /*-------------------------------*
  | Exemple de programme en C     |
  | YuGiOhJCJ <yugiohjcj@1s.fr>   |
  | http://yugiohjcj.1s.fr/       |
  *-------------------------------*/

#include <stdio.h>

int main(void){
printf("Hello World!\n");
return(0);
}
```