CI 1: Les variables

l.cuvillon@unistra.fr

Télécom Physique Strasbourg — October 14, 2021

Consignes

- Utiliser votre compte ENT (ou le compte local si indisponible) sous Linux.
- On vous demande de **créer un fichier source par exercice**.

1 Compilation

Exercice 1

Ecrire le code source d'`helloworldc+` puis le compiler et l'exécuter.

- On utilisera l'éditeur de texte `gedit` pour les fichiers sources (ou `geany` ou autre si disponible) et `gcc` comme compilateur. On a intérêt à activer l'affichage du numéro de ligne (qu'indique le compilateur en cas de problème), la colorisation de la syntaxe, les parenthèses actives dans les options de `gedit`. Voir les diapositives 45 à 55 du cours.

  On testera la compilation avec les 2 méthodes suivantes :

  – un appel direct au compilateur `gcc` et les options `-Wall -ansi -pedantic-errors` (cas d'un fichier source C nommé "helloworld.c"), puis exécution :

Exercice 1
Command Line

```
$ gcc -Wall -ansi -pedantic-errors helloworld.c -lm -o helloworld
$ ./helloworld

Hello world!
```

  L'option `-o` est suivie du nom souhaité pour l'exécutable qui peut être quelconque.
  L'option `-Wall` donne tous les warnings de compilation (très souvent des erreurs).
  L'option `-ansi -pedantic-errors` force l'utilisation de la version ansi du C.

  Le programme est exécuté avec un `./` devant le nom, qui indique que le fichier exécutable est situé dans le répertoire courant.

  – l'utilitaire de compilation `make` et son fichier de configuration `makefile` suivant à créer dans le répertoire de travail :

Makefile

```
CFLAGS= -Wall -ansi -pedantic-errors
LDLIBS= -lm
```

  Il suffit alors d'utiliser la commande `make` suivie du nom d'un quelconque fichier source (sans le `.c`) pour lancer que `make` compile le fichier avec les options adéquates:

Exercice 1
Command Line

```
$ make helloworld
cc -Wall -ansi -pedantic-errors helloworld.c -lm -o helloworld
$ ./helloworld

Hello world!
```

`Make` est particulièrement utile lorsque le programme est composé de plusieurs fichiers sources à compiler.

2 Les variables en C

2.1 Variable : Type et format d'affichage

Exercice 2

1. Ecrire un programme C qui déclare 4 variables de type adéquat pour être initialisées avec les valeurs suivantes : 98 (un entier naturel), 82.6, 185.3e2 et le caractère 'a'.
2. Afficher le contenu de ces variables avec printf() et le bon format:

   ```
   printf("%.. %... ", var1, var2, ...);
   ```

3. Si on utilise %c au lieu de %d pour afficher la variable contenant la valeur 98, qu'obtient-on? Quel est le nom du code standardisé qui permet de prédire cet affichage.
4. Et si on utilise %d pour afficher la valeur de l'expression 'a'+1, qu'obtient-on? Avec %c ?
5. Ajouter ce code à votre programme. Quelle valeur est affichée? Pourquoi?

   ```
   char var = 127;
   var = var +1; /
   printf("%d", var);
   ```

Note: Le type **char** stocke en mémoire un entier signé sur 256 valeurs. Le format précisé, i.e. %c ou %i,d, indique au compilateur si il faut ou non interprété la valeur de la variable comme une valeur numérique ou un caractère d'imprimerie par le biais du code ASCII.

2.2 Printf/Scanf

Exercice 3

En vous inspirant de l'exemple de l'addition de 2 entiers du cours, écrire un programme qui :

- demande à l'utilisateur de saisir les dimensions des 2 cotés orthogonaux d'un triangle rectangle
- affiche l'aire du triangle.

Note: Choisir le type de variables le plus adapté et générique pour les dimensions.