



Licence d'informatique
Module de Pratique du C

Premiers travaux dirigés de pratique du C

Octobre 2004
révision de septembre 2006

1 Premiers abords du langage C

Ces premiers exercices abordent les notions d'identificateurs du langage, les valeurs entières, les expressions et les opérateurs simples.

Exercice 1 (Identificateurs)

Indiquer si les identificateurs suivants sont valides :

foo-1	_foo_bar	3cavaliers
foo.	tête	__A__
-	a	while

Exercice 2 (Validité d'expressions)

On suppose les variables entières suivantes définies et affectées des valeurs indiquées :

A=20 B=5 C=-10 D=2 X=12 Y=15

Évaluez les expressions valides parmi les expressions suivantes :

5*X+2*3*B/4	A == B = 3	A += X + 2	A != C *= -D
A %= D++	A %= ++D	(X++) * A+C	A-A == B-B
!(X-D+C) D	A && B !0	C=12--	(1<<(2<<1))==(1<<2)<<1)

Corriger les expressions invalides en utilisant le parenthésage. On se référera au tableau 1 qui liste les opérateurs et leurs priorités.

Exercice 3 (Typage entiers et conversion)

Soient les déclarations suivantes :

```
long int i = 15;  
char c = 'A';  
short int j = 10;
```

Évaluez et donnez le type des expressions valides suivantes :

c+1	c+i	c+j
3*c+2*c	2*c+(i+10)/j	2*c+(i+10.0)/j
j=(i+10.0)/j	c = 666	j *= 3.14

2 Premiers programmes en C

Ces exercices ont pour but de vous familiariser avec la syntaxe du langage C :

- les structures de contrôle : if, while, for...
- quelques fonctions d'entrée/sortie : printf(), scanf(), getchar()...
- la structure de base d'un programme
- le codage des entiers positifs en binaire.

TAB. 1 – Opérateurs du langage C

16	() [] -> .	G
15	++ -- (postfixé)	D
14	! ~ ++ -- (préfixé) - (unaire)	D
	* (indirection) & (adresse) sizeof	D
13	* (multiplication) / %	G
12	+ -	G
11	<< >>	G
10	< <= > >=	G
9	== !=	G
8	& (et bit à bit)	G
7	^	G
6		G
5	&&	G
4		G
3	? :	D
2	= += -= *= /= %= >>= <<= &= ^= =	D
1	,	G

Exercice 4 (Factorielle)

Question 4.1 Donner le code d'une fonction

```
unsigned int factorielle (unsigned int n) ;
```

qui calcule $n!$. On développera une version itérative et une version récursive de la fonction.

Question 4.2 Donner le code d'un programme qui lit un entier et affiche sa factorielle.

Exercice 5 (Grande ligne)

Soit la fonction C `maxl_line()` qui lit un texte sur l'entrée standard et retourne la longueur de la plus grande ligne du texte.

On pourra utiliser la fonction `getchar()` de la bibliothèque standard qui renvoie le code ASCII du caractère lu sur `stdin`, la valeur `EOF` en fin de fichier ou cas d'erreur.

Question 5.1 Donnez le prototype de la fonction `maxl_line()`.

Question 5.2 Donnez la définition de la fonction `maxl_line()`.

Question 5.3 Donnez le code d'un programme qui lit son entrée standard et affiche la longueur de la plus longue ligne.

Exercice 6 (Parenthésage d'une expression)

On lit sur l'entrée standard (`stdin`) un texte, terminé par `EOF`, dont on veut vérifier que le parenthésage est correct. On sortira sur la sortie standard (`stdout`) un message pour le résultat. De plus, la commande se terminera sur un succès si et seulement si l'expression est bien parenthésée.

Exercice 7 (Exemples de macros)

Définir à l'aide de macros `cpp` les « constantes » `VRAI` et `FAUX`.

Écrire ensuite, le plus simplement possible, une macro `cpp` qui teste si un caractère `c` passé en paramètre est un chiffre ou une lettre majuscule : `CHIFROUMAJ(c)`.

Exercice 8 (Word count)

Donnez un programme C qui compte le nombre de mots du texte fourni en entrée.

Exercice 9 (Conversion caractères/entiers)

On calcule la valeur d'un entier entré sur `stdin` sous forme d'une suite de caractères (terminée par un caractère autre qu'un chiffre).

L'arithmétique « classique » peut s'appliquer sur le type `char`. La valeur numérique d'un caractère est le code ASCII de ce caractère (65 pour 'A', 66 pour 'B' ...).

Exercice 10 (Format des entiers positifs)

On veut calculer le nombre de bits sur lequel sont codés les `unsigned int`. Le format des données n'est pas précisé dans les spécifications du langage C, il peut dépendre de la machine ou du compilateur.

L'opérateur arithmétique `x << lg` calcule un décalage sur `x` d'une longueur `lg` bits vers les bits de poids fort. Les `lg` bits de poids forts sont perdus, ceux de poids faible sont mis à 0.

On remarquera qu'on peut utiliser cet opérateur pour une multiplication (ou une division avec `>>`) par 2, 4..., à condition de faire attention aux pertes éventuelles d'informations.

Exercice 11 (Conversion décimal/binaire)

Définissez une fonction qui affiche sur la sortie standard la représentation binaire d'un entier non signé fourni en paramètre.