INFO 301 – **2008/2012**

Pratique du C: Fiche de TD 2

Langage C

octobre 2008

1 Identificateurs, types simples, opérateurs, priorité et conversion implicite

Question 1.1 : *Indiquer si les identificateurs suivants sont valides :*

foo-1	*_foo_bar*	*3cavaliers*
foo.	*tête*	*___A_*
_	*a*	*3*

Question 1.2 : *En supposant :*

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

évaluer les expressions valides parmi les expressions suivantes :

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

En utilisant le parenthésage, corriger les expressions invalides.

Question 1.3 : *En supposant faites les déclaration :*

```
long int A = 15 ;
char B = 'A' ;
short int C = 10 ;
```

évaluer et donner le type des expressions valides suivantes :

$B + 1$	$B + A$	$B + C$
$3 * B + 2 * B$	$2 * B + (A + 10)/C$	$2 * B + (A + 10.0)/C$
$B = 2 * B + (A + 10.0)/C$	$C = 666$	$B* = 3.14$

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

2 Instructions de contrôle

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

Question 2.1 : *On entre sur* `stdin` *une valeur* `unsigned int`*. On sort sur* `stdout` *sa traduction en binaire.*

Question 2.2 : *Vous avez déjà remarqué que le clavier d'un téléphone outre la touche #, ressemblait à :*

1	*2*	*3*
	ABC	*DEF*
4	*5*	*6*
GHI	*JKL*	*MNO*
7	*8*	*9*
PQRS	*TUV*	*WXYZ*

Cette disposition permet à l'utilisateur de taper du texte. Par exemple, presser une fois la touche 7 *correspond à la lettre* P *alors que presser cette touche* 4 *fois correspond à la lettre* S*. La touche # du téléphone sert à séparer les lettres et peut être omise lorsqu'il est clair qu'une lettre se termine et l'autre commence.*

Ainsi, la séquence `777666222559996#6668866#8244466` *correspond au mot* `rockymountain`*.*

Donnez un programme qui permet de traduire une suite tapée sur le clavier d'un téléphone en un texte.

Question 2.3 : *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.*

Question 2.4 : *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.*

Question 2.5 : *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'`*...).*

Question 2.6 : *La racine carrée par défaut d'un entier positif* n *est un entier positif* p *qui vérifie :*

$$p^2 \leq n < (p+1)^2.$$

Construiser un programme qui calcul la racine carrée par défaut d'un entier.

Vous pouvez tester deux approches : la recherche linéaire de la racine par défaut et la recherche dichotomique.