[LC020]

Les symboles utilisés en C

Délimiteurs

/*	*/	de commentaires	(normalement non imbriquables)
"	"	de chaînes de caractères	(non imbriquables)
'	'	de caractère isolé	(non imbriquables)
<	>	de nom de fichiers	(lors de l'inclusion d'un fichier d'entête) (non imbriquables)
{	}	de blocs	(un bloc = instruction composée de plusieurs instructions)
(	)	d'arguments de fonctions	(obligatoires même si pas d'arguments)
(	)	d'expression	(parenthèses)
[	]	d'indice de tableau	(un seul indice à la fois)
	;	fin d'instruction	(obligatoire)
#include <...>↵		inclusion d'un fichier d'entête	(à partir du répertoire des fichiers d'entête standard)
#include "..."↵		inclusion d'un fichier d'entête	(à partir du répertoire courant)
#define	↵	définition d'une macro substitution	
	\↵	prolongement à la ligne suivante	

Opérateurs (voir détails plus loin)

.unaires	-	+	!	~	&	*	sizeof()	++	--
.arithmétiques	*	/	%	+	-	<<	>>		
.de comparaisons	>	>=	<	<=	==	!=			
.logiques et booléens	&	∧	|	&&	||				
.d'affectations	=	+=	-=	*=	/=	%=	<<=	>>=	&=	|=	∧=

Identificateurs (noms de variables ou de fonctions)

- Suite de caractères dont le premier est une lettre ou _ et les suivants sont une lettre, un chiffre, ou _
- Nombre de caractères quelconques mais certains compilateurs ne considèrent que les huit premiers pour les objets internes et les six premiers pour les objets externes
  (TURBO C : jusqu'à 32 caract.) (IBM RTPC C : jusqu'à 64 caract.) (Code Warrior 1.1 : jusqu'à 255 caract.)
- Distinction entre minuscules et majuscules
  (traditionnellement : minuscules pour variables et fonctions ; majuscules pour MACROS)
- Éviter d'utiliser un mot clé réservé, ou un mot commençant par _ (symbole de soulignement)

Liste des mots clés réservés (K&R 2.3) (TCU p315) (RTPCC p7.8) (ANSI)

	K&R	ANSI	TC	RTPC		K&R	ANSI	TC	RTPC		K&R	ANSI	TC	RTPC		TC
asm			*	*	float	*	•	•	•	signed		*	•			*
auto	*	•	•	•	for	*	•	•	•	sizeof	*	•	•	•		*
break	*	•	•	•	fortran					static	*	•	•	•		*
case	*	•	•	•	generic					struct	*	•	•	•		*
cdecl			*		goto	*	•	•	•	switch	*	•	•	•		*
char	*	•	•	•	huge			*		typedef	*	•	•	•		*
class					if	*	•	•	•	union	*	•	•	•		*
const		*	•		inline					unsigned	*	•	•	•		*
continue	*	•	•	•	int	*	•	•	•	void		*	•			*
default	*	•	•	•	interrupt			*		volatile		*	•			*
defined					long	*	•	•	•	while	*	•	•	•		*
do	*	•	•	•	near			*								*
double	*	•	•	•	parsal			*								*
else	*	•	•	•	pragma			*								*
enum		*	•	•	public											*
entry	*	•	•		register	*	•	•	•							*
extern	*	•	•	•	return	*	•	•	•							*
far			*		short	*	•	•	•							*

Différents types de données et leurs tailles

[références (K&R 2.6) (TCU p230) (RTPC C Annexe A-1)]

La seule chose imposée par le langage C du point de vue de la taille des différents types de données, est une relation . C'est à dire que l'on a :

sizeof(**char**) ≤ sizeof(**short**) ≤ sizeof(**int**) ≤ sizeof(**long**) pour les types entiers
et sizeof(**unsigned** *type*) = sizeof(*type*) pour chacun des mêmes types entiers

sizeof(**float**) ≤ sizeof(**double**) ≤ sizeof(**long double**) pour les types réels

Dans un programme où l'exécution dépend de la taille des données, il est conseillé d'utiliser l'opérateur **sizeof** (avec un nom de type en argument, ou bien suivi d'un nom de variable) pour déterminer cette taille.

Le type énuméré (**enum**) est en général équivalent au type **int** (en ce qui concerne la taille), mais certains compilateur autorisent une taille variable (indiquée ci-dessous par *) suivant le nombre de valeurs possibles.

Voici quelques exemples de tailles (en octets) avec les bornes de représentation correspondantes :

	TURBO C		**CODE WARRIOR** (version 1.1) sur **PowerMac**		**cc sur HP/UX**	
unsigned char	1	0 à 255	1	0 à 255	1	0 à 255
char	1	-128 à 127	1	-128 à 127	1	-128 à 127
unsigned short	2	0 à 65535	2	0 à 65535	2	0 à 65535
short	2	-32768 à 32767	2	-32768 à 32767	2	-32768 à +32767
enum	2	-32768 à 32767	*	(intervalle variable avec la taille)	4	-2147483648 à +2147483647
unsigned int	2	0 à 65535	4	0 à 4294967295	4	0 à 4294967295
int	2	-32768 à 32767	4	-2147483648 à +2147483647	4	-2147483648 à +2147483647
unsigned long	4	0 à 4294967295	4	0 à 4294967295	4	0 à 4294967295
long	4	-2147483648 à +2147483647	4	-2147483648 à +2147483647	4	-2147483648 à +2147483647
float	4	$-3,4.10^{38}$ à $3,4.10^{38}$	4	$-3,4.10^{38}$ à $3,4.10^{38}$	4	$-3,4.10^{38}$ à $3,4.10^{38}$
double	8	$-1,8.10^{308}$ à $1,8.10^{308}$	8	$-1,8.10^{308}$ à $1,8.10^{308}$	8	$-1,8.10^{308}$ à $1,8.10^{308}$
long double	10	$-1,2.10^{4932}$ à $1,2.10^{4932}$	8	$-1,8.10^{308}$ à $1,8.10^{308}$	16	$-1,2.10^{4932}$ à $1,2.10^{4932}$
pointeur	2	(pointeurs courts)	4		4	
	4	(pointeurs longs et maximaux)				

[LC021]

OPÉRATEURS en langage C

Priorité - Associativité - Ordre d'évaluation

			associativité
spéciaux		`( )` `[ ]` `->` `.`	**gauche**
unaires		`-` moins, `+` plus, `!` non logique (non booléen), `~` complément, `&` adresse, `*` él^t pointé par (indirection), **`sizeof()`** taille de, `++` incrém., `--` décrém.	**droite**
binaires	multiplicatifs	`*` mult., `/` div., `%` modulo	**gauche**
	additifs	`+` add., `-` soust.	**gauche**
	de décalage	`<<` à gauche, `>>` à droite	**gauche**
	de comparaisons	`>` sup., `>=` sup ou égal, `<` inf., `<=` inf ou égal	**gauche**
	de comparaisons	`==` égal à, `!=` différent de	**gauche**
	et bit à bit	`&`	**gauche**
	ou exclusif bit à bit	`^`	**gauche**
	ou bit à bit	`|`	**gauche**
	et booléen	`&&`	**gauche**
	ou booléen	`||`	**gauche**
d'expressions conditionnelles		`? :`	**gauche**
d'affectations		`=` `+=` `-=` `*=` `/=` `%=` `<<=` `>>=` `&=` `^=` `|=` `&&=` `||=`	**droite**
d'expressions composées		`,` **la virgule** sert de séparateur d'expressions composées (ou de séparateur d'arguments).	**gauche**

priorité décroissante

Exemples :

Associativité gauche :

tab[i].nom signifie `(tab[i]).nom`

x / y * z signifie `(x/y)*z`

x >> 5 << 2 signifie `(x>>5)<<2`

Associativité droite :

! pt ++ signifie `!(pt++)`

k = c += 3 signifie `k=(c+=3)`

☞ **Mais :** L'ordre d'évaluation est `ALEATOIRE ⚠` (non imposé par le langage)