Pratique du C Premiers pas Pratique du C Premiers pas **Details pratiques** Équipe pédagogique : Pratique du C Francesco De Comité G 1 (info) Premiers pas Sedoglavic G 2 (info) Alexandre Mikaël Salson G 3 (info) Adrien Poteaux G 4 (info) Samy Meftali G 1 (miage) Licence Informatique — Université Lille 1 Jean-François Roos G 2 (miage) Pour toutes remarques : Alexandre.Sedoglavic@univ-lille1.fr Semestre 5 — 2012-2013 Toutes les informations (emploi du temps, semainier, documents, etc.) sont disponibles à l'url : http://www.fil.univ-lille1.fr/portail Licence \rightarrow S5 info \rightarrow PDC. 101 101 131 131 3 900 4 D > 4 B > 4 B > 4 B > 9 9 9 www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf V58 (12-09-2011) Conçu aux laboratoires Bell par D. Ritchie pour développer Pratique du C Trust the programmer le système d'exploitation UNIX (des langages A et B ont existé mais ne sont plus utilisés) : Maximes; le ► C n'est lié à aucune architecture particulière ; Le langage C n'a pas été conçu pour faciliter sa lecture ► C est un langage typé qui fournit toutes les instructions (contrairement à Ada par exemple). nécessaires à la programmation structurée; Un concours annuel (International Obfuscated C Code C est un langage compilé. Contest — www.ioccc.org) récompense d'ailleurs le En 1983, l'ANSI décida de normaliser ce langage et définit la programme le plus illisible. norme ANSI C en 1989. Elle fut reprise intégralement en 1990 par l'ISO. Par exemple, le cru 2001 présentait le programme suivant : Les principes historiques de C sont : m(char*s,char*t) { 1. Trust the programmer. return *t-42?*s?63==*t|*s==*t&&m(s+1,t+1): 2. Make it fast, even if it is not guaranteed to be portable. !*t:m(s,t+1)||*s&&m(s+1,t); 3. Keep the language small and simple. main(int c,char **v) { return!m(v[1],v[2]); } **4.** Don't prevent the programmer from doing what needs to be done. 5. Provide (preferably) only one (obvious) way to do an operation. (D) (@) (E) (E) (900 4 D F 4 D F 4 E F 4 E F 9 9 9 9 www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf Trust the programmer Trust the programmer Premiers pas Maximes; le programme et sa Le langage C n'a pas été conçu pour faciliter sa lecture Le langage C n'a pas été conçu pour faciliter sa lecture (contrairement à Ada par exemple). (contrairement à Ada par exemple). Un concours annuel (International Obfuscated C Code Un concours annuel (International Obfuscated C Code Contest — www.ioccc.org) récompense d'ailleurs le Contest — www.ioccc.org) récompense d'ailleurs le programme le plus illisible. programme le plus illisible. Un autre exemple de 1999 : Vous ne validerez pas cet enseignement si vous suivez ces exemples. #include <stdio.h> int 0,o,i;char*I="";

V-2 (12-09-2011)

Premiers pas

Maximes; le premier programme et sa compilation

V58 (12-09-2011)

Pratique du C Premiers pas

Maximes; le programme et sa compilation

 $main(1){0\&=1\&1?*I:~*I,*I++||(1=2*getchar(),i+=0>8}$

?o:0?0:o+1,o=0>9,0=-1,I="t8B~pq'",1>0)?main(1/2):

printf("%d\n",--i);}

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

Par contre, vous l'aurez réussi si vous les comprenez sans

problèmes.

```
Pratique du C
Premiers pas
                                                                              Pratique du C
Premiers pas
             Make it fast, even if it is not guaranteed to
                                                                                          Keep the language small and simple
             be portable.
Maximes; le
              Les compilateurs traîtent les commandes C en fonction des
                                                                             premier
programme et sa compilation
                                                                             programme et sa compilation
                                                                                           Les 32 mots-clefs de l'ANSI C
              spécificités de l'architecture (implantation des types au plus
                                                                                              les spécificateurs de type :
              De plus, on peut faire appel à l'assembleur pour des tâches
                                                                                                 char double enum float
                                                                                                                                   int
                                                                                                                                               long
              critiques. Par exemple, dans le code du noyau Linux :
                                                                                                                                               void
                                                                                                 short signed struct union
                                                                                                                                   unsigned
              static inline int flag_is_changeable_p(u32 flag){ u32 f1,f2;
                                                                                              les qualificateurs de type :
                                                                                                                                     const
                                                                                                                                               volatile
                      asm("pushfl\n\t"
                                                                                              les instructions de contrôle :
                          "pushfln\t"
                                                                                                 break case continue default do
                                                                                                                                              else
                          "popl %0\n\t"
                                                                                                         goto if
                                                                                                                            switch while
                          "movl %0,%1\n\t"
                          "xorl %2,%0\n\t"
                                                                                              spécificateurs de stockage :
                          "push1 %0\n\t"
                                                                                                 auto register static extern typedef
                          "popfl\n\t"
                          "pushfl\n\t"
                                                                                              autres :
                                                                                                                                     return
                                                                                                                                                sizeof
                          "popl %0\n\t"
                          "popfl\n\t"
                          : "=&r" (f1), "=&r" (f2)
                          : "ir" (flag)); return ((f1^f2) & flag) != 0;
                                                                                                                              18 18 18 18 18 1 8 00 0
              }
V58 (12-09-2011)
                                                 www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                              www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf
 Pratique du C
             Keep the language small and simple
                                                                                          Don't prevent the programmer from doing
 Premiers pas
                                                                                          what needs to be done: C est un langage
Maximes; le
                                                                                          de bas niveau
                                                                             premier
programme et sa
programme et sa
              Les 40 opérateurs de l'ANSI C
compilation
                les opérateurs
                                                                                           Il n'est pas rare d'entendre dire que C est un assembleur de
                                    ->
                                                                                           haut niveau i.e. un assembleur typé qui offre des structures de
                                                    *()
                             ___
                                          (type)
                                                                                           contrôle élaborées et qui est — relativement — portable (et
                            sizeof
                                            *
                                                                                           porté) sur l'ensemble des architectures.
                                           >>
                                                    <<
                                                                                           Ce langage est pensé comme un assembleur portable :
                      >
                              <
                                    <=
                                           >=
                                                   ==
                              &
                                                   &&
                                                                                           son pouvoir d'expression est une projection des fonctions
                    <<=
                                                                                           élémentaires d'un microprocesseur idéalisé et suffisament
                      ?:
                                    +=
                                                    *=
                                           -=
                                                                                           simple pour être une abstraction des architectures réelles.
                     /=
                                           !=
                                                  >>=
              Et c'est tout!!!
```

4 m > V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf V58 (12-09-2011)

Don't prevent the programmer from doing

Pratique du C Premiers pas

Maximes; le programme et sa compilation

what needs to be done: C est un langage de bas niveau

C est un langage de bas niveau, il

- permet de manipuler des données au niveau du processeur (sans recourir à l'assembleur);
- ▶ ne gère pas la mémoire (ramassage de miettes);
- ▶ ne prévoit pas d'instruction traitant des objets composés comme des chaînes de caractères, des structures, etc. (pour comparer deux chaînes, il faut utiliser une fonction);
- ▶ ne fournit pas d'opération d'entrée-sortie dans le langage.

C utilise des bibliothèques pour ces tâches.

Maximes; le premier programme et sa compilation

www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf

Pratique du C dans le cursus de formation

L'ambition du cours est de se comprendre à plusieurs niveaux (dans l'ordre chronologique) :

- 1. C comme langage de programmation.
- 2. Relations entre C et architecture.
- 3. Relations entre C et os.

sans pour autant faire les cours se trouvant dans la même filière informatique du LMD à Lille :

Architecture élémentaire	(info 202)
Pratique du C	(info 301)
Pratique des systèmes	(info 305)

Pratique du C Premiers pas Maximes; le programme et sa compilation return 0 ; /* valeur de retour (0 i.e. EXIT_SUCCESS) */ V58 (12-09-2011) Pratique du C Premiers pas Principe de la compilation élémentaire 1. Édition du fichier source : fichier texte contenant le Maximes; le premier programme et sa 2. Traitement par le préprocesseur : le fichier source est compilation 3. La compilation : le fichier engendré par le 4. L'assemblage : transforme le code assembleur en un 5. L'édition de liens : afin d'utiliser des librairies de V58 (12-09-2011) Pratique du C Premiers pas Maximes; le programme et sa compilation

```
Le premier programme et sa compilation
 En fin de cours, les détails du code suivant seront limpides :
 /* ceci est un commentaire */
#include <stdio.h>
 main
 (int argc, char **argv)
 printf("Salut le monde \n") ;
```

▶ include est une directive au préprocesseur pour incorporer ce qui permet l'usage de la fonction printf de la bibliothèque standard;

programme — nécessite un éditeur de texte (emacs, vi).

traité par un *préprocesseur* qui fait des transformations

préprocesseur est traduit en assembleur i.e. en une suite

purement textuelles (remplacement de chaînes de

caractères, inclusion d'autres fichiers source, etc).

d'instructions associées aux fonctionnalités du

fichier objet i.e. compréhensible par le processeur

fonctions déjà écrites, un programme est séparé en

plusieurs fichiers source. Une fois le code source assemblé, il faut lier entre eux les fichiers objets. L'édition de liens produit un fichier exécutable.

microprocesseur (faire une addition, etc).

▶ les instructions se terminent par un point-virgule;

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

Pratique du C Premiers pas

Maximes; le

programme et sa compilation

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

Outils utilisés en TP

Vous êtes libres d'utiliser vos outils préférés...

... à condition que ceux-ci soient :

- emacs ou vi pour l'édition de vos fichiers sources;
- gcc pour la compilation. Il s'agit du gnu C compiler et on peut y adjoindre certains drapeaux. Par exemple, % gcc -Wall -ansi -pedantic foo.c indique que vous désirez voir s'afficher tous les avertissements du compilateur (très recommandé);
- pdb pour l'exécution pas à pas et l'examen de la mémoire (ddd est sa surcouche graphique).

Une séance sera consacrée à la compilation séparée et à certains outils de développement logiciels.

Pratique du C Premiers pas

Les constantes et identificateurs

Le premier programme et sa compilation

```
En fin de cours, les détails du code suivant seront limpides :
/* ceci est un commentaire */
#include <stdio.h>
int
main
(int argc, char **argv)
printf("Salut le monde \n") ;
return 0 ; /* valeur de retour (0 i.e. EXIT_SUCCESS) */
```

- la fonction main est imposée pour produire un exécutable (qui commence par exécuter main). Elle est définie par l'en-tête de la fonction : type de retour, nom, argument; les accolades contiennent les instructions composant la fonction. L'instruction return est une instruction de retour à la fonction appelante;
- ▶ appel de la fonction printf déclarée dans stdio.h — avec une chaîne en paramètre.

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

Programmation sans filet ⇒ maîtrise indispensable du langage

En C, le programmeur est censé maîtriser parfaitement le langage et son fonctionnement.

Les tests d'erreurs et de typages ne sont fait qu'au moment de la compilation i.e. rien n'est testé lors de l'exécution (les convertions de types, l'utilisation d'indices de tableau supérieurs à sa taille, etc).

De plus, le compilateur est laxiste : il vous laisse faire tout ce qui a un sens (même ténu) pour lui.

Un programme peut donc marcher dans un contexte (certaines données) et provoquer des erreurs dans un autre.

```
10 × 40 × 45 × 45 × 5 × 90 0
www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf
```

Les constantes numériques

Les entiers machines : on peut utiliser 3 types de notations:

- ▶ la notation décimale usuelle (66, −2);
- ▶ la notation octale (commençant par un 0 (en C, la constante 010 est différente de 10));
- ▶ la notation hexadécimale (commençant par un 0x (en C, la constante 0×10 est égale à 16));

Les réels machines : ne sont pas en précision infinie et sont notés par :

- ► *mantisse* −273.15, 3.14 et
- ► exposant indiqué par la lettre e : 1.4e10, 10e 15.

Pratique du C Premiers pas Pratique du C Premiers pas Les constantes (non) numériques Identificateur : un nom associé à de l'espace mémoire Les identificateurs servent à manipuler les obiets du langage i.e. manipuler de l'espace mémoire auquel on donne un nom. Ils désignent de la mémoire contenant des données (des Les caractères se répartissent en deux types : Les constantes et variables, etc.) ou de la mémoire contenant du code à ▶ imprimable spécifié par '. Ainsi, 'n' correspond à exécuter (des fonctions). l'entier 110 (et par le biais d'une table au caractère n); Ils ne peuvent pas commencer par un entier (mais peuvent ▶ non imprimable qui sont précédés par \. Ainsi, '\n' les contenir). C distingue les majuscules des minuscules : (x correspond à un saut de ligne. et X sont deux identificateurs différents). On peut aussi utiliser pour ces caractères la Certaines règles de bon usage sont à respecter; notation '\ ξ ' avec ξ le code ASCII octal associé (cf. ▶ les caractères non ASCII — i.e. non portables — ne % man ascii) devraient pas être utilisés (pas d'accent); les identificateurs débutant par un blanc_souligné sont propres au système d'exploitation et ne devraient pas être utilisés par un programmeur en dehors de l'OS; il vaut mieux choisir des identificateurs parlant. www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011) Pratique du C Pratique du C Les entiers machines Les entiers machines Premiers pas Premiers pas Il y a 8 types associés aux entiers machines : Ces types sont modifiables par les attributs short et long : ▶ int est le type de base des entiers signés et correspond ▶ short int est un type plus *court* (codé sur 16 bits) et représentant les entiers dans $[-2^7, 2^7]$. au mot machine (historiquement 16 bits ou actuellement 32 bits, bientôt 64). Les entiers ▶ long int est un type plus long (codé sur 32 bits — Types : tailles de Types : tailles de représentables sont donc dans l'intervalle $[-2^{31}, 2^{31}]$. pour des raisons historiques) et représentant les entiers la représentation Ce type est modifiable par un attribut unsigned des objets dans $[-2^{31}, 2^{31}]$. unsigned int est le type de base des entiers non ▶ long long int est un type encore plus long (codé signés codés sur le même nombre d'octets (donc sur 64 bits) et représentant les entiers dans $[-2^{63}, 2^{63}]$. compris entre $[0, 2^{32} - 1]$). La taille dépend de l'architecture de la machine et peut varier. 4 m > 4 D F 4 D F 4 E F 4 E F 9 9 9 9 www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf Pratique du C Les entiers machines Les entiers machines Premiers pas Premiers pas On indique comment typer une constante en utilisant les suf-Pour connaître le nombre d'octets associés à un type, on utilise le mot clef du langage sizeof. Par exemple, u ou U unsigned (int ou long) 550u int I ou L long 123456789L main Types : tailles de (void) ul ou UL unsigned long 12092UL la représentation des objets return sizeof(long long int); On peut manipuler en C des entiers plus grand en employant des représentations non spécifiées par le langage (tableaux, listes chaînées — voir la librairie gnu multiprecision (GMP) est un programme qui retourne le nombre d'octets codant le par exemple). type long long int. % # sur les machines de tp % gcc foo.c ; ./a.out % echo \$?

Les constantes et

V58 (12-09-2011)

la représentation des objets

V58 (12-09-2011)

Pratique du C

Types : tailles de la représentation des objets

```
%
```

10148143131 3 999

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

Pratique du C Premiers pas Pratique du C Premiers pas Le type flottant (float) et les "booléens" Le type char Il existe deux types pour le codage des réels en C : Le langage C associe aux caractères le type char ▶ float pour les flottants simple précision 32 bits généralement codé sur 1 octet. Le type char est généralement ; généralement signé de -128 à 127. double pour les flottants double précision 64 bits Types : tailles de la représentation Historiquement, le code ASCII nécessitait 7 bits. Types : tailles de la représentation généralement. Le reste du monde utilisant des accents, l'ISO définit une des objets des objets On ne peut pas les modifier (unsigned, short) comme les foultitude de codage sur 1 octet : le code ASCII de base autres types si ce n'est pour : jusqu'à 127 et le reste à partir de 128. Le type char est modifiable par l'attribut unsigned pour ▶ long double qui correspond à un codage sur 12 octets. coder les entiers de 0 à 255. Il faut bien garder à l'esprit que le type char représente des Les booléens sont — sémantiquement — représentés par les entiers dont la correspondance avec des lettres de l'alphabet entiers: est faîte en dehors du langage par une table (voir valeur logique fausse : valeur nulle 0; % man ascii). valeur logique vraie : tout entier ≠ 0. ARKARIATE TOOR V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf Pratique du C Pratique du C Pour déclarer une variable, il faut faire suivre le nom du type Premiers pas Premiers pas Schématiquement, une variable correspond à un par un identificateur. Par exemple : emplacement en mémoire. Dans le code source, ce dernier int i; /* pas tr\'es */ est manipulé par l'entremise de l'identificateur de la variable. /* explicite */ int j, k; Avant utilisation, toutes les variables doivent être : short int s; /* ces identificateurs :-(*/ ▶ soit *définies localement* ce qui correspond à l'allocation float f; d'une zone mémoire (segment de pile); double d1,d2; ▶ soit définies globalement ce qui correspond : Bien qu'il soit vivement conseillé de découpler déclaration et Variable et Variable et initialisation, on peut affecter des variables à la déclaration : déclaration de déclaration de ▶ à l'allocation d'une zone mémoire (segment de variable variable Caractère : char nom =' A '; données): au stockage de cette adresse dans la table; Chaîne de caractères : char *foo =" bar "; ▶ soit *déclarées* extern ce qui correspond à : Entier machine : int nom = 666; une variables définies dans un autre fichier source; ► Flottant machine : float nom = 3.14; la création d'une entrée dans la table des symboles ; ▶ mais pas à son allocation : l'adresse est inconnue à Implicitement, nous venons de nous servir de 2 opérateurs : la virgule permet de définir une suite; ► (l'adresse est résolue à l'édition de liens). l'opérateur d'affectation =. V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf Une expression correspond à la composition d'identificateurs Pratique du C Pratique du C Qualificatif précisant le type Premiers pas Premiers pas et d'opérateurs. Elle se termine par un point virgule. L'action d'un opérateur sur un identificateur peut avoir 2 types de conséquences : retourner la valeur de l'expression; On peut modifier les types en précisant le codage machine à un effet latéral portant sur l'identificateur. l'aide des mots-clefs signed, unsigned, short, long. Par exemple, l'affectation foo = 2 provoque : $[-2^{31}, 2^{31}]$ $[-2^7, 2^7]$ short int ▶ l'effet latéral : l'entier 2 est affecté à la variable foo ; Variable et $[0, 2^{32}]$ long int $[-2^{63}, 2^{63}]$ unsigned int déclaration de • et retourne la valeur qui vient d'être affectée. variable unsigned short int $[0, 2^8]$ unsigned long int $[0, 2^{64}]$ Construction On peut donc avoir une expression du type : d'expression en C On peut aussi modifier les flottants par les mots-clefs bar = foo = 2; double et long double. L'opérateur ++ provoque : l'effet latéral : incrémente l'expression ; • et retourne la valeur qui vient d'être obtenue. foo = ++bar; (D) (A) (B) (B) (B) (900 V58 (12-09-2011) www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

Pratique du C Premiers pas Construction d'expression en C V58 (12-09-2011) Pratique du C Premiers pas Conversion explicite V58 (12-09-2011) Pratique du C Premiers pas Division Opérateurs

```
Comment déterminer la sémantique d'une
expression?
 Il faut maîtriser l'action des opérateurs :
```

opérateurs arithmétiques classiques : + addition - soustraction * multiplication / division % reste de la division

- ▶ les opérateurs relationnels >, <, <=, >=, ==
- ▶ les opérateurs logiques booléens && et, || ou, ! non
- ▶ les opérateurs logique bit à bit & et, | ou inclusif, ~ ou
- ▶ les opérateurs d'affectation composée +=, -=, /=,
- les opérateurs d'incrémentation et de décrémentation
- ▶ l'opérateur conditionnel ternaire foo = x>=0? x :-x.
- conversion de type char foo = (char) 48.14;

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

```
Considérons la situation suivante :
```

```
int foo = 2;
unsigned char bar = 3;
double var = foo + bar ;
```

Les opérateurs ne pouvant agir que sur des données de types homogènes, il y a 2 conversions de type dans cet exemple :

- ▶ l'opérateur + provoque si nécessaire la conversion d'une des opérandes après son évaluation;
- ▶ l'opérateur = provoque si nécessaire la conversion de l'opérande de droite dans le type de l'identificateur de gauche après son évaluation et avant son affectation à cet identificateur.

En cas de doute, il faut utiliser la conversion de type explicite:

```
int foo = 2;
unsigned char bar = 3:
double var = foo / bar ; var = ((double) foo / (double) bar)
                                      www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
```

Pratique du C Premiers pas

```
Syntaxe :
                        \Rightarrow expression<sub>1</sub> / expression<sub>2</sub>
► Sémantique : comme l'addition
```

- pas de distinction entre division entière ou réelle;
- ▶ division entière ⇔ expression₁ et expression₂ entières;
- cas de la division entière :
 - opérandes positives : arrondi vers 0 (13/2 = 6);
 - une opérande négative : dépend de l'implantation ; 13/-2=-6 ou -7.

Modulo

V58 (12-09-2011)

- Syntaxe : ⇒ expression₁ % expression₂
- Sémantique :
 - expression₁ et expression₂ entières;
 - reste de la division entière ;
 - si un opérande négatif : signe du dividende en général;
 - Assurer que b * (a / b) + a % b soit égal à a.

Priorité et ordre d'évaluation des opérateurs

```
L'instruction X *= Y + 1; n'est pas équivalente à
X = X * Y + 1;
```

```
16 | () [] ->
15 | ++ -- (postfixé) -- (préfixé) -- (unaire)
14 | ! -- (préfixé) -- (unaire)
      * (indirection) & (adresse) sizeof
    11
10
     & (et bit à bit)
     &&
5
     ||
?:
3
                -= *= /= %= >>= <<= &= ^=
```

Pour l'opérateur o, la priorité *G* indique que l'expression $\exp_1 \circ \exp_2 \circ \exp_3$ est évaluée de gauche à droite.

```
4 D > 4 B > 4 B > 4 B > 90 C
www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf
```

Opérateurs arithmétiques usuels

Addition

Pratique du C Premiers pas

Construction

Opérateurs

Opérateurs

d'expression en C

- Syntaxe : \Rightarrow expression₁ + expression₂
- Sémantique :
 - évaluation des expressions et calcul de l'addition;
 - retourne la valeur de l'addition :
 - ordre indéterminé d'évaluation des expressions;
 - conversion éventuelle d'une des opérandes après

Soustraction

```
Syntaxe : l'opérateur peut être utilisé de manière unaire
  ou binaire :
                         \Rightarrow - expression
           \Rightarrow expression<sub>1</sub> - expression<sub>2</sub>
```

Multiplication

- Syntaxe : \Rightarrow expression₁ * expression₂
- Sémantique : voir addition.

10 × 40 × 45 × 45 × 5 × 90 0 www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf

Opérateurs de comparaison

Svntaxe :

 \Rightarrow expression₁ opérateur expression₂ où opérateur est l'un des symboles :

opérateur	ur sémantique	
>	strictement supérieur	
<	strictement inférieur	
>=	supérieur ou égal	
<=	inférieur ou égal	
==	égal	
!=	différent	

- Sémantique :
 - évaluation des expressions puis comparaison;
 - valeur rendue de type int (pas de type booléen);
 - vaut 1 si la condition est vraie :
 - vaut 0 si la condition est fausse;
- ▶ Ne pas confondre : test d'égalité (==) et affectation (=)

```
Pratique du C
Premiers pas
                                                                                              Pratique du C
Premiers pas
                Opérateurs logiques
                                                                                                              Ou logique
                                                                                                                 Syntaxe :
                                                                                                                                        \Rightarrow expression<sub>1</sub> || expression<sub>2</sub>
                  Et logique
                                                                                                                 Sémantique : expression<sub>1</sub> est évaluée et :
                    Syntaxe :
                                                                                                                      1. si valeur non nulle : l'expression || rend 1;
                               ⇒ expression<sub>1</sub> && expression<sub>2</sub>
                                                                                                                      2. si valeur nulle : expression2 est évaluée et
                                                                                                                          2.1 si valeur nulle : l'expression || rend 0;
                    ► Sémantique : expression<sub>1</sub> est évaluée et :
                                                                                                                          2.2 si valeur non nulle : l'expression || rend 1.
                         1. si valeur nulle : l'expression && rend 0 ;
                         2. si valeur non nulle : expression<sub>2</sub> est évaluée et
                                                                                                                 ▶ Remarque : expression<sub>2</sub> non évaluée si expression<sub>1</sub> vraie
                             2.1 si valeur nulle : l'expression && rend 0;
                             2.2 si valeur non nulle : l'expression && rend 1.
                                                                                                              Non logique
                                                                                                                 Syntaxe :
                                                                                                                                        \Rightarrow ! expression
                             expression<sub>2</sub> non évaluée si expression<sub>1</sub> fausse
                                                                                                                 ► Sémantique : expression est évaluée et :
Opérateurs
                       Utile: (n != 0) && (N / n == 2)
                                                                                             Opérateurs
                                                                                                                      1. valeur nulle : l'opérateur ! délivre 1;
                       Désagréable : (0) && (j = j - 1).
                                                                                                                      2. valeur non nulle : l'opérateur ! délivre 0.
                                                           ARTHUR TO ARTHUR TO AREA
                                                                                                                                                        40 > 40 > 43 > 43 > 3 990
V58 (12-09-2011)
                                                           www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                                                        www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf
  Pratique du C
                                                                                              Pratique du C
                Opérateurs de traitement des bits
  Premiers pas
                 Non bit à bit
                                                                                                              Décalage à gauche
                                                                                                                 Syntaxe :
                    Syntaxe :
                                                                                                                                        \Rightarrow expression<sub>1</sub> << expression<sub>2</sub>
                                           ⇒ ~ expression
                                                                                                                 Sémantique :
                    ► Sémantique :
                          ▶ évaluation de expression ⇒ type entier;

    évaluation de expression<sub>1</sub> et expression<sub>2</sub>;

    doivent être de valeur entière, positive pour expression2;

                          calcul du non bit à bit sur cette valeur :
                          rend une valeur entière.
                                                                                                                       ▶ expression₁ décalée à gauche de expression₂ bits en
                                                                                                                         remplissant les bits libres avec des zéros.
                  Et bit à bit
                   Syntaxe :
                                           \Rightarrow expression<sub>1</sub> & expression<sub>2</sub>
                                                                                                              Décalage à droite
                    ► Sémantique : évaluation de expression<sub>1</sub> et expression<sub>2</sub>
                                                                                                                 Syntaxe :
                                                                                                                                        \Rightarrow expression<sub>1</sub> >> expression<sub>2</sub>
                       qui doivent être de valeur entière.
                                                                                                                 ► Sémantique : voir décalage à gauche :
                  Ou bit à bit
                                                                                                                       ▶ si expression₁ unsigned : décalage logique
Opérateurs
                                                                                             Opérateurs
                                                                                                                         les bits rentrants à droite sont des zéros :
                   Syntaxe :
                                           \Rightarrow expression<sub>1</sub> | expression<sub>2</sub>
                                                                                                                       ▶ si expression<sub>1</sub> signée : décalage arithmétique
                  Ou exclusif bit à bit
                                                                                                                         les bits rentrants à droite sont égaux au bit de signe.
                    Syntaxe :
                                           \Rightarrow expression<sub>1</sub> \hat{} expression<sub>2</sub>
                    Sémantique : voir et bit à bit.
                                                            (D) (@) (E) (E) (900
                                                                                                                                                        4□ F 4∰ F 4 분 F 4 분 F 900
V58 (12-09-2011)
                                                           www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                                                        www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf
                                                                                                              Attention à l'usage des instructions composées et des
  Pratique du C
                                                                                              Pratique du C
                                                                                               Premiers pas
                                                                                                              variables :
                                                                                                              #include <stdio.h>
                  Instruction composée (du bon usage des accolades).
                                                                                                              char foo = 'c' :
                    ► Syntaxe : instruction-composée :
                                                                                                              int main(void){
                                                                                                                printf(" %c ",foo) ;
                                                                                                                 char foo = 'a' ; \*on n'utiliser qu'un nom de variable*\
                                    liste-de-déclarationsoption
                                                                                                                printf(" %c ",foo) ;
                                    liste-d'instructionsoption
                                   } liste-de-déclarations :
                                                                                                                char foo = 'b' ; \*mais c'est une tr\'es mauvaise id\'ee*\
                                                                                                                printf(" %c ",foo);
                               ⇒ déclaration
                               ⇒ liste-de-déclarations déclaration
                                                                                                                printf(" %c \n",foo) ;
                       liste-d'instructions:
                                                                                                                return 0 ;
                               \Rightarrow instruction
                               ⇒ liste-d'instructions instruction
                                                                                                              % gcc InstructionsComposees.c : a.out
                                                                                                               caba
                       Une expression isolée n'a pas de sens.
Instructions
                                                                                             Instructions
                                                                                                              De toutes façons :
usuelles
                                                                                             usuelles
                                                                                                              %gcc -Wall -ansi -pedantic InstructionsComposees.c
                                                                                                              InstructionsComposees.c: In function 'main':
                                                                                                              InstructionsComposees.c:9: warning:
```

40) 40) 42) 42) 2 990

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

V58 (12-09-2011)

48 + 43 + 43 + 3 99 C

ISO C89 forbids mixed declarations wand in Good file 1.fr/~sedoglav/C/CoursO2.pdf

```
Pratique du C
Premiers pas
                                                                                          Pratique du C
Premiers pas
                                                                                                            Syntaxe : instruction-conditionnelle :
                   Sémantique des instructions composées : 2 objectifs
                                                                                                                      \Rightarrow if ( expression ) instruction<sub>1</sub>
                        1. Grouper un ensemble d'instructions en une seule
                                                                                                                      \Rightarrow if ( expression ) instruction<sub>1</sub>
                           instruction:
                                                                                                                               else instruction
                        2. Déclarer des variables accessibles seulement à l'intérieur
                                                                                                            ► Remarques sur la syntaxe :
                           de instruction-composée

    expression doit être parenthésée;

                                           ⇒ Structure classique de blocs
                                                                                                                 pas de mot clé then;
                                                                                                                 ▶ ambiguïté du else :
                   Remarques
                                                                                                                       if (a > b) if (c < d) u = v; else i = j;
                         pas de séparateur dans liste-d'instructions :
                                                                                                                    Règle : relier le else au premier if de même niveau
                                           le ; est un terminateur pour les expressions
                                                                                                                    d'imbrication n'ayant pas de else associé
                        ▶ accolades ({}) correspondant au begin end de Pascal.
                                                                                                                       if (a > b) { if (c < d) u = v; } else i = j;
Instructions
usuelles
                                                                                         Instructions de
                                                                                         contrôle
                                                         40 > 40 > 43 > 43 > 3 990
V58 (12-09-2011)
                                                        www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                                                 www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf
  Pratique du C
                                                                                                        Instruction à choix multiples
  Premiers pas
                                                                                           Premiers pas
                                                                                                         Syntaxe:
                   Sémantique :
                                                                                                         switch ( expression )
                        évaluation de expression;
                        ▶ si valeur non nulle : exécution de instruction₁ :
                        ▶ si valeur nulle : exécution de instruction<sub>2</sub> si elle existe.
                                                                                                               case expr-cste<sub>1</sub>: liste-d'instructions<sub>1 option</sub>
                                                                                                                                                               break<sub>option</sub>;
                   ► Remarques sur la sémantique :
                                                                                                               case expr-cste<sub>2</sub> : liste-d'instructions<sub>2 option</sub>
                         ▶ if : teste égalité à zéro de expression ;

    expression : pas forcément une comparaison ;

                                                                                                                                                               break option;

    expression : comparable à zéro ;

                               if (a != 0) { ... }
                               if (a) { ... }
                                                                                                               case expr-cste_n: liste-d'instructions_{n \ option}
                                                                                                                                                               break option;
                                                                                                               default : liste-d'instructions<sub>option</sub>
Instructions de
                                                                                         Instructions de
                                                                                                         }
                                                                                         contrôle
                                                         4 D > 4 B > 4 E > 4 E > 9 Q P
                                                                                                                                                 4□ F 4∰ F 4 분 F 4 분 F 900
V58 (12-09-2011)
                                                        www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                                                 www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf
                                                                                                         Exemple utilisant le break :
  Pratique du C
                                                                                          Pratique du C
  Premiers pas
                                                                                          Premiers pas
                                                                                                             int nb = 1:
                  1. Évaluation de expression;
                                                                                                             switch(nb){
                  2. Résultat comparé avec expr-cste<sub>1</sub>, expr-cste<sub>2</sub>, etc.;
                                                                                                                case 1 : printf("un"); break;
                                                                                                                case 2 : printf("dos"); break;
                  3. Première expr-cste; égale à expression : exécution de
                                                                                                                case 3 : printf("tres"); break;
                      liste-d'instructions correspondante;
                                                                                                                default : printf("erreur: pas dans la chanson\n");
                  4. Instruction break : termine l'exécution du switch;
                  5. Si aucune expr-cste; égale à expression : exécution de
                                                                                                         Exemple n'utilisant pas le break :
                      liste-d'instructions de l'alternative default si celle-ci
                                                                                                             switch(c){
                      existe, sinon on ne fait rien.
                                                                                                                case '0':
                                                                                                                case '1':
                 Remarques:
                                                                                                                case '2':
                                                                                                                case '3':

    expr-cste; : valeur connue à la compilation (constante);

                                                                                                                case '4':
                   expr-cste; : pas deux fois la même valeur;
                                                                                                                case '5':
                                                                                                                case '6':
                   s'il n'y a pas de break à la fin d'un case : exécution
                                                                                                                case '7':
                      des liste-d'instruction des case suivants:
                                                                                                                case '8':
Instructions de
                                                                                         Instructions de
                                                                                                                case '9': nb_chiffres++; break;
                   ▶ l'alternative default est optionnelle.
                                                                                         contrôle
```

4 D F 4 B F 4 B F B F 99 C

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

V58 (12-09-2011)

4 m + www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

default: nb_non_chiffres++;

```
Pratique du C
Premiers pas
                                                                                                                                                                                                            Pratique du C
Premiers pas
                                   Instructions itératives
                                                                                                                                                                                                                                              Instruction do ... while
                                       Trois instructions d'itération : instruction-itérative :
                                                                                                                                                                                                                                                     ► Syntaxe : ⇒ do instruction
                                                        ⇒ instruction-while
                                                                                                                                                                                                                                                                            while ( expression );
                                                        ⇒ instruction-do
                                                                                                                                                                                                                                                     ► Sémantique : boucle avec test en fin d'itération
                                                        ⇒ instruction-for
                                       Instruction while
                                                                                                                                                                                                                                                                                                           instruction
                                           ► Syntaxe : ⇒ while ( expression ) instruction
                                           ► Sémantique : boucle avec test en début d'itération
                                                                                                                                                                                                                                                                                                     expression != 0? non
                                                                                                                                                                                                                                                                                                                                                      fin du do
                                                                                                 expression != 0?
                                                                                                                 oui
                                                                                                                                              fin du while
                                                                                                                                                                                                                                                     Exemple : somme des n = 10 premiers entiers
                                                                                                  instruction
                                                                                                                                                                                                                                                           int n = 10; int i = 1; int somme = 0;
                                                                                                                                                                                                                                                           if (i <= n) do {
                                                                                                                                                                                                                                                                     somme = somme + i;
                                                                                                                                                                                                                                                                     i = i + 1;
                                           Exemple : somme des n = 10 premiers entiers
Instructions de
                                                                                                                                                                                                          Instructions de
                                                                                                                                                                                                                                                               } while (i <= n);</pre>
contrôle
                                                                                                                                                                                                         contrôle
                                                  int n = 10; int i = 1; int somme = 0;
                                                  while (i <= n) { somme = somme+i; i = i+1; }
                                                                                                                                                                                                                                                                                                                                         40 > 40 > 43 > 43 > 3 990
V58 (12-09-2011)
                                                                                                                                                                                                                                                                                                                                         www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf
                                                                                                                                www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
    Pratique du C
                                                                                                                                                                                                             Pratique du C
     Premiers pas
                                                                                                                                                                                                             Premiers pas
                                                                                                                                                                                                                                              Instruction for (suite)
                                                                                                                                                                                                                                                     Remarques
                                       Instruction for
                                                                                                                                                     expression1

    expression<sub>1</sub> et expression<sub>3</sub> : valeurs inutilisées

                                        Syntaxe:
                                                                                                                                                                                                                                                                                                        ⇒ effet latéral : expressions d'affectation
                                         \Rightarrow for ( expression<sub>1 option</sub> ;
                                                                                                                                                                                                                                                                 expression<sub>1</sub>: instruction d'initialisation;
                                                                                                                                               \overline{\text{expression2} \,!\!= 0\,?} \underbrace{\text{non}}

    expression<sub>3</sub>: instruction d'itération;

                                                      expression_{2 option};
                                                                                                                                                 oui
                                                                                                                                                                             fin du for

    expression<sub>2</sub>: expression de test (arrêt quand nulle);

                                                                                                                                                                                                                                                                 ▶ instruction : corps de la boucle.
                                                                                                                                                       instruction
                                                      expression<sub>3 option</sub>)
                                                                                                                                                                                                                                                     Exemple : somme des n = 10 premiers entiers
                                                                                                                                                                                                                                                           int i; int n = 10; int somme = 0;
                                                        instruction
                                                                                                                                                     expression3
                                                                                                                                                                                                                                                           for (i = 0; i \le n; i = i + 1)
                                          Sémantique :
                                                                                                                                                                                                                                                                     somme = somme + i;
                                         itération bornée
                                                                                                                                                                                                                                                     ▶ Même exemple avec un corps de boucle vide
                                                                                                                                                                                                                                                           int i ; int n ; int somme ;
                                                                                                                                                                                                                                                           for (i=0,n=10,somme=0; i< n; somme=somme+(i=i+1));
 Instructions de
                                                                                                                                                                                                          Instructions de
                                                                                                                                                                                                         contrôle
                                                                                                                                 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 
                                                                                                                                                                                                                                                                                                                                         4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 
V58 (12-09-2011)
                                                                                                                                www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)
                                                                                                                                                                                                                                                                                                                                         www.fil.univ-lille1.fr/~sedoglay/C/Cours02.pdf
     Pratique du C
                                                                                                                                                                                                             Pratique du C
     Premiers pas
                                       Attention à la confusion avec d'autres langages :
                                                                                                                                                                                                              Premiers pas
                                                                                                                                                                                                                                               Instruction goto (possible mais à proscrire) :
                                                                                                                                                                                                                                                     Syntaxe :
                                                                                                                                                                                                                                                                                                      ⇒ goto identificateur ;
                                       #include <stdio.h>
                                                                                                                                                                                                                                                     ► Sémantique :
                                       int main(void){
                                                                                                                                                                                                                                                                 ► toute instruction est étiquetable;
                                                                                                                                                                                                                                                                 si on la fait précéder d'un identificateur suivi du signe :
                                           int foo :
                                                                                                                                                                                                                                                                 et d'un identificateur : étiquette;
                                           for( foo=0 ; foo<10 ; foo++)
                                                                                                                                                                                                                                                                 poto : transfère le contrôle d'exécution à l'instruction
                                                                printf("%d\n",foo);
                                                                                                                                                                                                                                                                      étiquetée par identificateur.
                                           for( int bar=0 ; bar<10 ; bar++)</pre>
                                                                                                                                                                                                                                                     ► Remarques :
                                                                printf("%d\n",bar);
                                                                                                                                                                                                                                                                 • étiquettes visibles que dans la fonction où elles sont
                                          return 0 ;
                                                                                                                                                                                                                                                                 s'utilise pour sortir de plusieurs niveaux d'imbrication;

    permet d'éviter des tests répétitifs ;

                                       La compilation donne :
                                                                                                                                                                                                                                                                               for (...) { for (...) {
                                      % gcc for.c
                                                                                                                                                                                                                                                                                         if (catastrophe) goto erreur;
                                      for.c: In function 'main':
 Instructions de
                                      for.c:9: 'for' loop initial declaration used outside C99 mode
                                                                                                                                                                                                                                                                               erreur: printf("c'est la cata...");
 contrôle
                                                                                                                                 4 D F 4 B F 4 B F B F 99 C
                                                                                                                                                                                                                                                                                                                                          ( D ) ( D ) ( E ) ( E ) ( E ) 900
```

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

V58 (12-09-2011)

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

Pratique du C Premiers pas Pratique du C Premiers pas Rupture de contrôle Instruction continue: Syntaxe : \Rightarrow continue; ► Sémantique : Instruction break: uniquement dans une instruction for, while ou do; Syntaxe : instruction : \Rightarrow break ; provoque l'arrêt de l'itération courante; Sémantique : passage au début de l'itération suivante; ▶ provoque l'arrêt de la première instruction for, while, ▶ équivalent goto suite avec : do ou switch englobante; while(...) { do { for(...) { reprend l'exécution à l'instruction suivant l'instruction terminée. suite: ; suite: ; } while(...); Instructions de contrôle Instructions de contrôle

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf V58 (12-09-2011)

V58 (12-09-2011)

www.fil.univ-lille1.fr/~sedoglav/C/Cours02.pdf

suite: ;