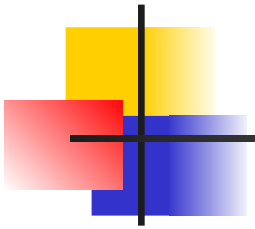


Types d’opérateurs

- Opérateur d’affectation
- Opérateurs arithmétiques
- Opérateurs de comparaison
- Opérateurs booléens
- Opérateurs de manipulation de bits
- …

Rappel sur la logique booléenne

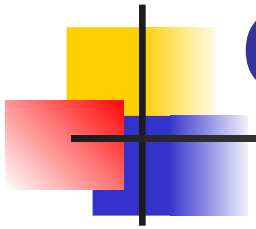
- Valeurs possibles d'une variable booléenne
  - VRAI ou FAUX
- Opérateurs logiques
  - ET, OU, NON
- Expression booléenne
  - Suite de variables booléennes et d'opérateurs
  - Valeur : VRAI ou FAUX
  - Exemple
    - Si A est VRAI et B est FAUX
      - A ET B FAUX
      - A OU B VRAI
      - NON A est FAUX

Les opérateurs de comparaison

- Opérateurs
  - < ⇒ inférieur
  - > ⇒ supérieur
  - <= ⇒ ≤, inférieur ou égal
  - >= ⇒ ≥, supérieur ou égal
  - == ⇒ =, égal
  - != ⇒ ≠, différent
- Exemples
  - `5 < 3`
  - `3 == 3`
  - `A != B`

  } expressions

Conditions simples



- Syntaxe
  - expr1 op expr2
- Exemple
  - `2 < 5` valeur de l'expression : VRAI
  - `3 > 7` valeur de l'expression : FAUX
  - `1 == 5` valeur de l'expression : FAUX

Conditions composées

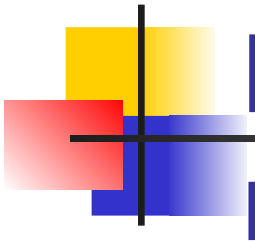
- Plusieurs conditions simples reliées entre elles par des opérateurs logiques
- Exemple
  - 3<5 ET 5<8 — VRAI
  - 2>6 OU 7==3 OU 4<9 — VRAI
  - 4<9 ET 5>9 — FAUX
  - 9==9 OU 5>5 — VRAI

Opérateurs logiques

- En C, il existe 3 opérateurs logiques
  - ET `&&`
  - OU `||`
  - NON `!`
- Exemples
  - `a && b`
  - `3 < 5 && 8 < 7`
  - `2 > 4 || 4 < 8`
  - `!a`

Valeur d'une expression logique

- **En C, la valeur d'une expression vaut**
  - **1 si l'expression est VRAI**
  - **0 si l'expression est FAUX**
- Exemples
  - `(5<3)` vaut 0
  - `(5<3 || 3<9)` vaut 1
  - `!0` vaut 1



Les opérateurs de manipulation de bits

- Le langage C permet de travailler directement sur le motif binaire d'une valeur
- Il permet notamment
  - l'opération ET bit à bit
  - l'opération OU bit à bit
  - l'opération OU exclusif bit à bit
  - Le complément à 1 bit à bit
  - Le décalage à gauche
  - Le décalage à droite du motif

Rappel

Dans les exemples suivants, les nombres sont non-signés

```
  0100 1100 (76)
ET 0000 1111 (15)
-----------------
  0000 1100 (12)
```

```
  0100 1100 (76)
OU 0000 1111 (15)
-----------------
  0100 1111 (79)
```

```
   0100 1100 (76)
OU EX 0000 1111 (15)
-------------------
   0100 0011 (67)
```

0100 1100 (76)

complément à 1

1011 0011 (179)

0100 1100 (76)

décalage à gauche

0100 11000 (152)

décalage à droite

00100 110 0 (38)

Les opérateurs en C

- Bit à bit
  - ET : `op1 & op2`
  - OU : `op1 | op2`
  - OU EX : `op1 ^ op2`
  - complément à 1: `~op1`
- Décalage
  - Gauche `op1 << op2` (décalé à gauche de op2 bits)
  - Droite `op1 >> op2` (décalé à droite de op2 bits)

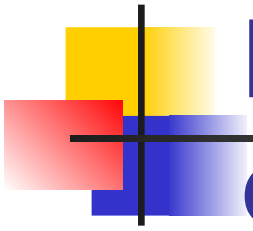
Exemple

```
int main(void)
{
    unsigned char a,b,resultat;
    a = 76; /* 0x4C */
    b = 15; /* 0x0F */
    resultat = a & b;/* 12 */    /* 0x0C  */
    resultat = a | b;/* 79 */    /* 0x4F  */
    resultat = a ^ b;/* 67 */    /* 0x43  */
    resultat = ~a;   /* 179  */      /* 0xB3  */
    resultat = a << 1;   /* 152  */      /* 0x98  */
    resultat = a >> 1;   /* 38  */       /* 0x26  */
}
```

Les opérateurs d'affectation élargie

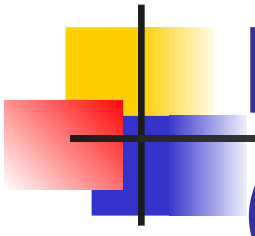
- Ces opérateurs permettent de réaliser en même temps:
  - une opération
  - une affectation

`+= , -=`	`a += 3;`	`a = a + 3;`
`*= , /=`	`a *= 5;`	`a = a * 5;`
`%=`	`a %= 3;`	`a = a % 3;`
`&= , |= , ~=`	`a |= 2;`	`a = a | 2;`
`<<= , >>=`	`a <<= 4;`	`a = a << 4;`



Les opérateurs d'incrément et de décrément

- incrémentation
  - `++`
- décrémentation
  - `--`
- Exemple
  - `i++;` équivalent `i = i+1;`
  - `i--;` équivalent `i = i-1;`
- Possibilité de pré-incrémentation
  - `++i;` équivalent `i = i+1;`
- Possibilité de post-incrémentation
  - `i++;` équivalent `i = i+1;`



Pré et Post (in|dé)crémentation

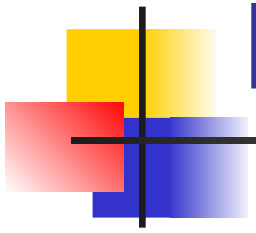
```
int a;
int i = 2;
a = i++;
```

La valeur de i est d'abord affectée à a puis incrémentée ensuite
i sera égal à 3
a sera égal à 2

```
int a;
int i = 2;
a = ++i;
```

La valeur de i est d'abord incrémentée puis affectée à a ensuite
i sera égal à 3
a sera égal à 3

L'opérateur `sizeof`



- Cet opérateur permet de connaître la taille en octets d'un objet.
- Son unique opérande peut être
  - un type
  - une expression
- Exemple

```
int n;
sizeof(int );/*a pour valeur  4 */
sizeof( n );  /*a pour valeur  4 */
sizeof(char);/*a pour valeur 1  */
```

Priorité des opérateurs

Opérateurs	Associativité
() [] -> .	De gauche à droite
! ~ ++ -- (type) * & sizeof	De droite à gauche
* / %	De gauche à droite
+ -	De gauche à droite
<< >>	De gauche à droite
< > <= >=	De gauche à droite
== !=	De gauche à droite
&	De gauche à droite
^	De gauche à droite
|	De gauche à droite
&&	De gauche à droite
||	De gauche à droite
?:	De droite à gauche
= += -=	De droite à gauche
,	De gauche à droite