

Variables et opérateurs en C

Séance 2

de l'ECUE « introduction à la programmation »

Bruno Bouzy

bruno.bouzy@parisdescartes.fr

Opérateurs arithmétiques

- $*$, $+$, $-$: multiplication, addition, soustraction
- $/$: division réelle pour des `float` ou `double`
- $/$: quotient de la division entière pour des `int`
- $\%$: reste de la division entière pour des `int`

Affectation, opérateurs de comparaison, opérateurs logiques

- `=` : affectation
(cf séance 1...)
- `==`, `!=` : égalité, inégalité
(prochaine séance...)
- `<`, `<=` : inférieur strict, inférieur ou égal
(prochaine séance...)
- `!`, `&&`, `||` : non logique, et logique, ou logique
(prochaine séance...)

Opérateurs d'incrémentement

- `++` : incrémentement
- `--` : décrémentation
- `a++`; **et** `++a`; sont équivalents à `a = a+1`;
- `a--`; **et** `--a`; sont équivalents à `a = a-1`;
- `expression(a++)`; **est équivalent à:**
`expression(a)`; `a=a+1`;
- `expression(++a)`; **est équivalent à:**
`a=a+1`; `expression(a)`;

Opérateurs et affectation simultanée

- $+=$, $-=$, $*=$, $/=$: opérateurs fusionnés avec l'affectation
- Exemples:
 - $x += a$; est équivalent à: $x = x + a$;
 - $n /= 2$; est équivalent à $n = n/2$;

Priorité (ou précédence) des opérateurs

- Par priorité décroissante:

()

- (unaire)

!

* / %

+ -

< <= > >=

== !=

&&

||

Exemple 1 (1/2)

- Donner la sortie de:

```
printf("-4-3*2-1 = %d\n", -4-3*2-1);
```

```
printf("-5/2 = %d\n", -5/2);
```

```
printf("2*-5/2 = %d\n", 2*-5/2);
```

```
printf("1+3*8%3 = %d\n", 1+3*8%3);
```

Exemple 1 (2/2)

- Sortie:

$$-4 - 3 * 2 - 1 = \mathbf{-11}$$

$$-5 / 2 = \mathbf{-2}$$

$$2 * -5 / 2 = \mathbf{-5}$$

$$1 + 3 * 8 \% 3 = \mathbf{1}$$

Exemple 2 (1/4)

- Donner la sortie de:

```
int n1 = 20, x1;  
x1 = n1++;  
printf("x1 = %d\n", x1);
```

```
int n2 = 20, x2;  
x2 = ++n2;  
printf("x2 = %d\n", x2);
```

Exemple 2 (2/4)

```
int n1=20, x1;
```

n1
int 


x1
int 

```
x1 = n1;
```

n1
int 

x1
int 

```
n1 = n1+1;
```


n1
int 

x1
int 

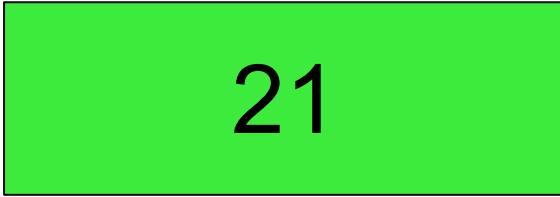
Exemple 2 (3/4)


```
int n2=20, x2;
```

n2
int 

x2
int 

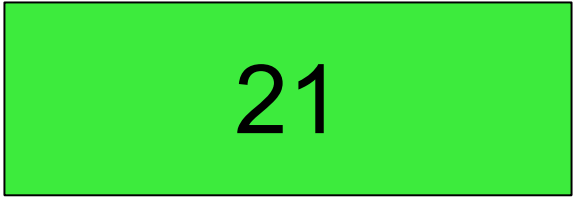
```
n2 = n2+1;
```

n2
int 

x2
int 

```
x2 = n2;
```

n2
int 

x2
int 

Exemple 2 (4/4)

- Sortie:

x1 = 20

x2 = 21

Exemple 3 (1/2)

- Donner la sortie de:

```
int X1, A=5, B=10, C=1;  
X1 = (2*A+3)*B+4;  
printf("X1 = %d\n", X1);
```

```
int X2=3, Y=4;  
X2 *= Y+1;  
printf("X2 = %d\n", X2);
```

Exemple 3 (2/2)

- Sortie:

X1 = 134

X2 = 15

Exemple 4 (1/3)

- $*$, $+$, $-$: multiplication, addition, soustraction

```
int a=3, b=5, c=7, d=11;
```

```
printf("a + b * c = %d.\n", a + b * c);
```

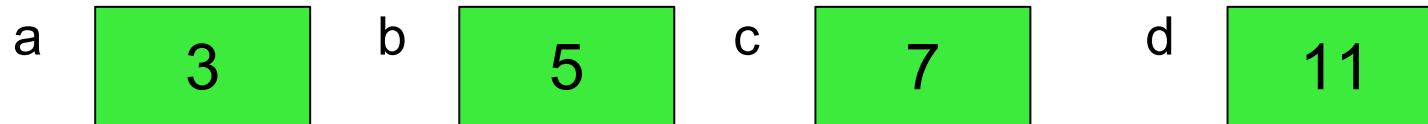
```
printf("a * b + c = %d.\n", a * b + c);
```

```
printf("a + c / b = %d.\n", a + c / b);
```

```
printf("c / a + b = %d.\n", c / a + b);
```

Exemple 4 (2/3)

- En mémoire...



Exemple 4 (3/3)

- Sortie écran:

$$a + b * c = \mathbf{38}.$$

$$a * b + c = \mathbf{22}.$$

$$a + c / b = \mathbf{4}.$$

$$c / a + b = \mathbf{7}.$$

Exemple 5 (1/2)

- $*$, $+$, $-$: multiplication, addition, soustraction

```
int a=3, b=5, c=7, d=11;
```

```
printf("a + b * c - d = %d.\n", a + b * c - d );
```

```
printf("(a + b)*c - d = %d.\n", (a + b)*c - d );
```

```
printf("a + (b*c) - d = %d.\n", a + (b*c) - d );
```

```
printf("a + b*(c - d) = %d.\n", a + b*(c - d) );
```

Exemple 5 (2/2)

- Sortie écran:

$$a + b * c - d = \mathbf{27.}$$

$$(a + b) * c - d = \mathbf{45.}$$

$$a + (b * c) - d = \mathbf{27.}$$

$$a + b * (c - d) = \mathbf{-17.}$$

Exemple 6 (1/2)

- **Division entière**

```
int a=3, b=5, c=7, d=11;
```

```
printf("b/a = %d. ", b / a);
```

```
printf("reste de b/a = %d.\n", b%a);
```

```
printf("c/a = %d. ", c / a);
```

```
printf("reste de c/a = %d.\n", c%a);
```

```
printf("d/a = %d. ", d / a);
```

```
printf("reste de d/a = %d.\n", d%a);
```

Exemple 6 (2/2)

- Sortie écran:

b/a = **1**. reste de b/a = **2**.

c/a = **2**. reste de c/a = **1**.

d/a = **3**. reste de d/a = **2**.

Exemple 7 (1/2)

- Division réelle

```
float e=3, f=5, g=7, h=11;  
  
printf("f/e = %f.\n", f/e);  
  
printf("g/e = %f.\n", g/e);  
  
printf("h/e = %f.\n", h/e);
```

Exemple 7 (2/2)

- Sortie écran:

$$f/e = \mathbf{1.666667.}$$

$$g/e = \mathbf{2.333333.}$$

$$h/e = \mathbf{3.666667.}$$

Présentation du source (1/3)

```
// premierProg.c
#include <stdio.h>
int main() {
    printf("Bonjour.\n");
    return (0);
}
```

- Un ; marque la fin d'une instruction.
- Une ligne par instruction.
- Les { } marquent le début et la fin d'un « bloc ».
- Un bloc regroupent des instructions.
- Indentation: décalage des lignes du même bloc.

Présentation du source (2/3)

```
// premierProg.c
#include <stdio.h>
int main() { printf("Bonjour.\n"); return (0); }
```

```
// premierProg.c #include <stdio.h>
int main() { printf("Bonjour.\n"); return (0); }
```

```
// premierProg.c
#include <stdio.h>
int main()
{
    printf("Bonjour.\n");
    return (0);
}
```

- **Compilable ? Lisible ?**

Présentation du source (3/3)

```
#include <stdio.h>
int main ( ) { printf ( "Bonjour.\n"
) ; return ( 0 ) ; }
```

```
#include <stdio.h>
int
main()
{
    printf(
    "Bonjour.\n")
    ;
    return
    (0)
    ;
}
```

- **Compilable ? Lisible ?**

Résumé de la séance 2

- Opérateurs
 - Arithmétiques, (booléens à la séance 3)
 - Affectation, (de comparaison à la séance 3)
 - In(dé)crémentement
 - Précédence, priorité
- Exemples
 - $+$, $-$, $*$, $x++$, $++x$, ...
 - Division entière, division réelle
- Présentation du source