# Les boucles en C

Séance 4

de l'ECUE « introduction à la programmation »

Bruno Bouzy

bruno.bouzy@parisdescartes.fr

# Boucles

- Pour un traitement itératif, répétitif.
- Jeu du sumo, définition
- 3 types de boucles:

  `for` `while` `do while`
- Exemple
- Jeu du sumo, solution
- Suite numérique

# Jeu du sumo, définition (1/5)

- Exécution (1/3)

Le sumo se déplace
vers la droite.
Le sumo peut
sortir du terrain
par la droite.
Une sortie du terrain
termine le jeu.

```
Bonjour.
---O---
mouvement (g/d) ? d
----O--
mouvement (g/d) ? d
-----O-
mouvement (g/d) ? d
------O
mouvement (g/d) ? d
Droite :-)
Au revoir.
```

# Jeu du sumo, définition (2/5)

- Exécution (2/3)

Le sumo se déplace
aussi vers la gauche.
Il peut sortir du
terrain par la gauche.

```
Bonjour.
---O---
mouvement (g/d) ? g
--O----
mouvement (g/d) ? g
-O-----
mouvement (g/d) ? g
O------
mouvement (g/d) ? g
Gauche :-)
Au revoir.
```

# Jeu du sumo, définition (3/5)

- Exécution (3/3)

Le sumo peut aller

à droite ou à gauche

tant qu'il reste

sur le terrain.

```
Bonjour.
---O---
mouvement (g/d) ? g
--O----
mouvement (g/d) ? d
---O---
mouvement (g/d) ? d
----O--
mouvement (g/d) ? d
-----O-
mouvement (g/d) ? d
------O
mouvement (g/d) ? d
Droite :-)
Au revoir.
```

# Jeu du sumo, définition (4/5)

- Entrée sorties:
  - L'utilisateur (U) entre `'g'` ou `'d'` au clavier.
  - Le sumo est représenté avec un `'O'`
  - Un espace est représenté avec un `'-'`
  - Le terrain est représenté par une suite de 7 caractères dont 6 `'-'` et 1 `'O'`.
  - Si l'U tape `'g'` le sumo va à gauche.
  - Si l'U tape `'d'` le sumo va à droite.
  - Le programme affiche le terrain.

# Jeu du sumo, définition (5/5)

- L'exécution est répétitive:
  - Tant que le sumo n'est pas éjecté,
    - Affichage du terrain avec le sumo.
    - Demande de l'action: `'g'` ou `'d'` ?
    - Mise a jour de la position du sumo.
  - Quand le sumo est éjecté,
    - Le programme affiche le vainqueur.
    - L'exécution se termine.

# Comment programmer le jeu du sumo en C ?

- Avec des:
  - entrées sorties
  - variables
  - conditionnelles: `if else`
  - **boucles** (séance 4 aujourd'hui)
    - `for`
    - `while`
    - `do while`

# for (1/11)

```
int x; printf("v initiale ? ");
scanf("%d", &x);
int y; printf("increment ? ");
scanf("%d", &y);
int n; printf("n iterations ? ");
scanf("%d", &n);

int i;
for (i=0; i<n; i++) {
  x += y;
  printf("i = %d, x = %d\n", i, x);
}
```

# for (2/11)

- Exécution:

```
Tapez une valeur initiale: 0
Tapez un increment: 3
Tapez un nombre d'iterations: 4
i = 0, x = 3
i = 1, x = 6
i = 2, x = 9
i = 3, x = 12
```

# for (3/11)

- `i=0;`
  - `i` : variable de boucle
  - Valeur initiale de `i : 0`
- `i<n;`
  - Condition pour continuer
  - Testée au début de chaque itération
  - `n` : nombre d'itérations (`i=0, i=1, , i=n-1`)
- `i++;`
  - Incrémentation de `i`
  - Exécutée à la fin de chaque itération

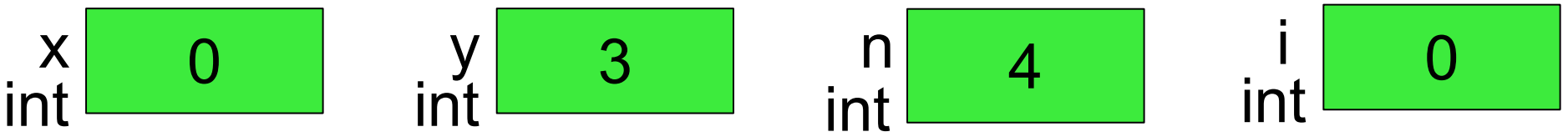
# for (4/11)

```
{
    x += y;
    printf("i = %d, x = %d\n", i, x);
}
```

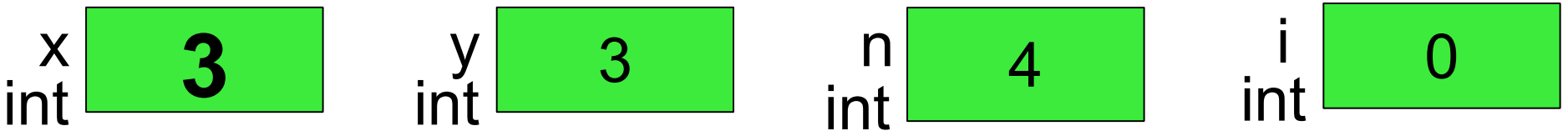
- Le bloc exécuté à chaque itération

# for (5/11)

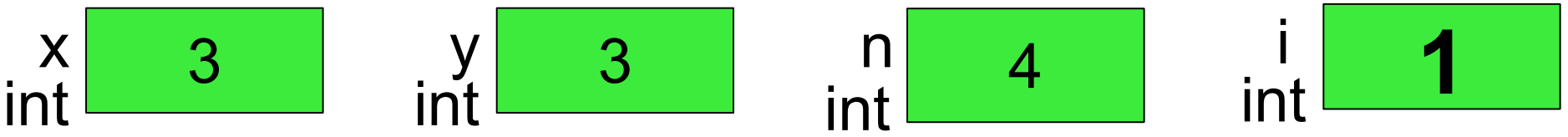
- **Avant** d'exécuter le bloc pour la 1ère fois:




- **Après** la 1ère exécution du bloc:

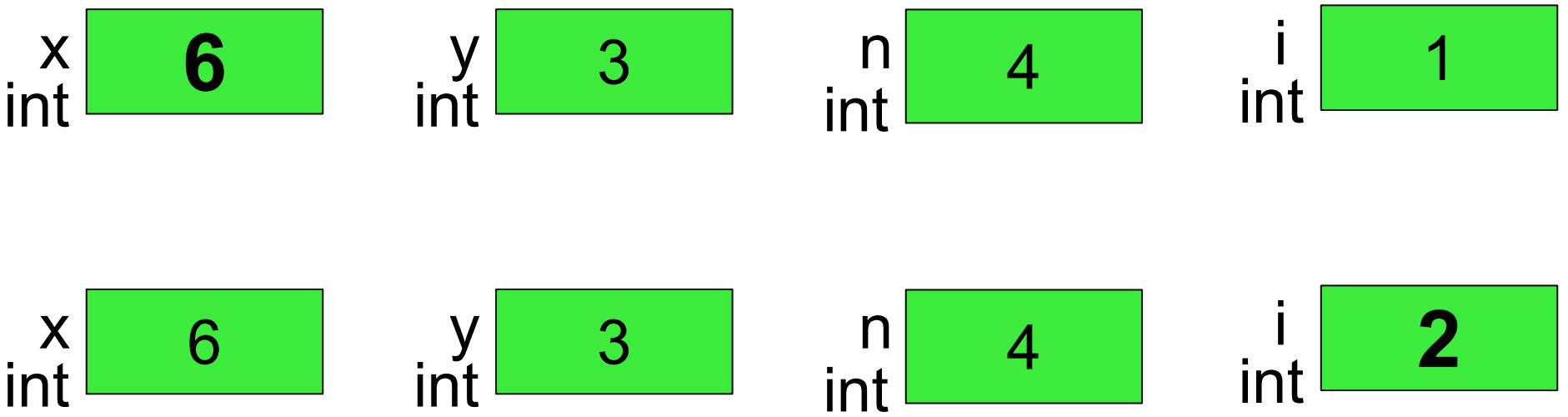



- Après l'exécution de la **fin du bloc**:



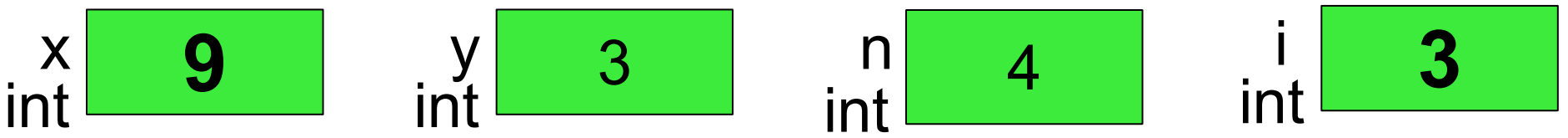
# for (6/11)

- Est-ce que `i<n` ? **`1<4`** ? oui: on continue!



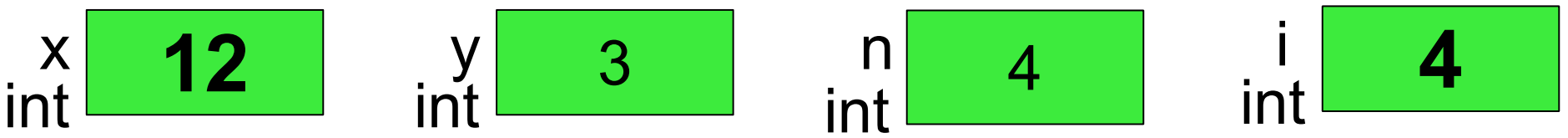
# for (7/11)

- Est-ce que `i<n` ? **`2<4`** ? oui: on continue!



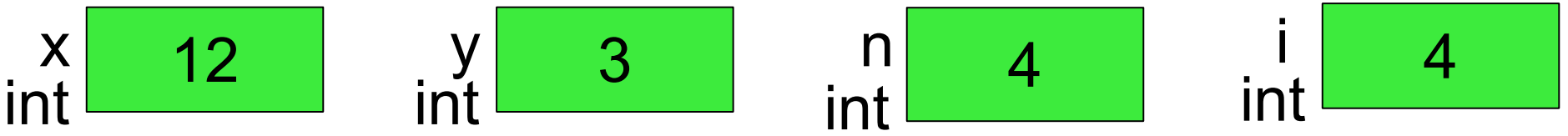
# for (8/11)

- Est-ce que `i<n` ? **`3<4`** ? oui: on continue!



# for (9/11)

- Est-ce que `i<n` ? **`4<4`** ? **non: on s'arrête!**



# while

- `while` peut faire la même chose que `for`

```
int i = 0;
while (i<n) {
   x += y;
   printf("i = %d, x = %d\n", i, x);
   i++;
}
```

- Initialisation `i=0;` faite avant le `while`
- Condition testée avant l'exécution du bloc
- Incrémentation faite dans le bloc, à la fin.

# do while

- La condition est testée à la fin
- On exécute au moins une fois le bloc

```
int i = 0;
do {
   x += y;
   printf("i = %d, x = %d\n", i, x);
   i++;
} while (i<n);
```

# for (10/11)

- On peut forcer la sortie avec un `break;`

```
int i;
for (i=0; i<n; i++) {
  x += y;
  printf("i = %d\n", i);
  if (i>n/2) break;
  printf("x = %d, y = %d\n", x, y);
}
```

- Lorsque `i>n/2` on sort de la boucle

# for (11/11)

- On peut effectuer les itérations sans faire les instructions situées après un `continue;`

```
int i;
for (i=0; i<n; i++) {
  x += y;
  printf("i = %d\n", i);
  if (i>n/2) continue;
  printf("x = %d, y = %d\n", x, y);
}
```

- Lorsque `i>n/2` on n'effectue pas l'instruction située après le `continue`.

# Exemple (1/2)

- Que fait le programme suivant ?

```
int main() {
  char c1, c2;
  printf("bonjour.\n");
  do {
    printf("q: quitter,\n");
    printf("autre: continuer.\n");
    scanf("%c", &c1);
    scanf("%c", &c2);
  } while (c1!='q');
  printf("au revoir.\n");
}
```

# Exemple (2/2)

```
bonjour.
q: quitter,
autre: continuer.
d
q: quitter,
autre: continuer.
s
q: quitter,
autre: continuer.
q
au revoir.
```

- Réponse:
  - Affichage du menu tant que le caractère lu au clavier est différent de `q`.

# Jeu du sumo, solution (1/6)

- Les variables importantes:
  - `int x;` position du sumo `-3 <= x <= +3`
  - `char m;` mouvement (`'g'` gauche, `'d'` droite)

# Jeu du sumo, solution (2/6)

- L'affichage:

```
----O--
```

- Le code C:

```
int i;
for (i=-3; i<=3; i++) {
  if (i==x) printf("O");
  else printf("-");
}
printf("\n");
```

# Jeu du sumo, solution (3/6)

- L'entrée au clavier:

```
mouvement (g/d) ? d
```

- Le code C:

```
char m, bidon;
printf("mouvement (g/d) ? ");
scanf("%c", &m);
scanf("%c", &bidon);
```

# Jeu du sumo, solution (4/6)

- La mise a jour de la position du sumo:
  - Si `'g'` alors `x` est décrémenté (`x=x-1`)
  - Sinon si `'d'` alors `x` est incrémenté (`x=x+1`)
- Le code C:

```
if (m=='g') x--;
else if (m=='d') x++;
```

# Jeu du sumo, solution (5/6)

- Code C d'une itération du jeu:

```
int i;                              // affichage
for (i=-3; i<=3; i++) {
  if (i==x) printf("O");
  else printf("-");
}
printf("\n");

char m, bidon;                        // entree
printf("mouvement (g/d) ? ");
scanf("%c", &m); scanf("%c", &bidon);

if (m=='g') x--;                    // mise a jour
else if (m=='d') x++;
```

# Jeu du sumo, solution (6/6)

- Code C de plusieurs itérations du jeu:

```
int x=0;
do {
  int i; for (i=-3; i<=3; i++) {
    if (i==x) printf("O");
    else printf("-");
  } printf("\n");

  char m, bidon; printf("mouvement (g/d) ? ");
  scanf("%c", &m); scanf("%c", &bidon);

  if (m=='g') x--;
  else if (m=='d') x++;
} while ((x>=-3) && (x<=3));

if (x<-3) printf("Gauche :-)\n");
else      printf("Droite :-)\n");
```

# Suite numérique (1/5)

- Ecrire un programme affichant les `N` premières valeurs de la suite numérique $U_n$ définie par:

$$U_0 = 0$$

$$U_{n+1} = U_n + 4/(2n+1) \text{ si } n \text{ est pair}$$

$$U_{n+1} = U_n - 4/(2n+1) \text{ si } n \text{ est impair}$$

# Suite numérique (2/5)

- Quelles variables C utiliser ?
  - Une variable pour `Un`, par exemple `u`.
  - Une variable pour l'index `n`, par exemple `n`.
- A ne pas faire :
  - Une variable par terme de la suite,

    par exemple `u0`, `u1`, `u2` etc.
- Les mathématiques sont différentes de la programmation C:

  pas de variable avec indice en programmation C

# Suite numérique (3/5)

```
#include <stdio.h>

#define N_ITERATIONS_LEIBNIZ      200

int main() {
  float u=0;
  int n;
  for (n=0; n<N_ITERATIONS_LEIBNIZ; n++) {
    if (n%2==0)
      u = u + 4.0/(2*n+1);
    else
      u = u - 4.0/(2*n+1);
    printf("u%2d = %.6f\n", n+1, u);
  }
  return 0;
}
```

# Suite numérique (4/5)

```
u 1 = 4.000000    ...               ...
u 2 = 2.666667    u30 = 3.108268
u 3 = 3.466667    ...               u100 = 3.131593
u 4 = 2.895238    u40 = 3.116596    ...
u 5 = 3.339683    ...
u 6 = 2.976046    u50 = 3.121594
...               ...               u195 = 3.146721
u10 = 3.041840    u60 = 3.124927    u196 = 3.136491
...               ...               u197 = 3.146669
u20 = 3.091624    u70 = 3.127307    u198 = 3.136543
                                    u199 = 3.146618
...               ...               u200 = 3.136593
```

# Suite numérique (5/5)

```
#include <stdio.h>
#define N_ITERATIONS_LEIBNIZ      200
int main() {
float u=0;
int n;
for (n=0; n<N_ITERATIONS_LEIBNIZ; n++) {
if (n%2==0)
u = u + 4.0/(2*n+1);
else
u = u - 4.0/(2*n+1);
printf("u%2d = %.6f\n", n, u);
}
return 0;
}
```

- Compilable ? Lisible ?

# Résumé de la séance 4

- Jeu du sumo, exécution, définition
- Types de boucles

  `for` `while` `do while`
- Exemple, affichage répétitif
- Jeu du sumo, solution, code C
- Suite numérique, code C