

Corrigé des exercices du Cours 10

Fonctions récursives
Graphisme

Vincent Lepetit

`vincent.lepetit@epfl.ch`

Qu'affichent les programmes suivants ?

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}

...

cout << fact(3) << endl;
```

B.

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}

...

d(3);
```

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}


...

cout << fact(3) << endl;
```

A.


```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

 `cout << fact(3) << endl;`

n = 3

A.



```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

n = 3

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}

...

cout << fact(3) << endl;
```

n = 3

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

n = 3

n = 3

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    → if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

n = 3

n = 3

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```




$\underbrace{\hspace{1.5cm}}_2$

...

```
cout << fact(3) << endl;
```

n = 3

n = 2



```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

n = 3

n = 2

```
int fact(int n)
{
  cout << "n = " << n << endl;
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

n = 3

n = 2

```
int fact(int n)
{
  cout << "n = " << n << endl;
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1);
}
```

...

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
```

n = 2

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```



1


...

```
cout << fact(3) << endl;
```

n = 3

n = 2

n = 1



```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
```

n = 1

```
int fact(int n)
{
  cout << "n = " << n << endl;
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

n = 3

n = 2

n = 1


```
int fact(int n)
{
  cout << "n = " << n << endl;
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
```

n = 1

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

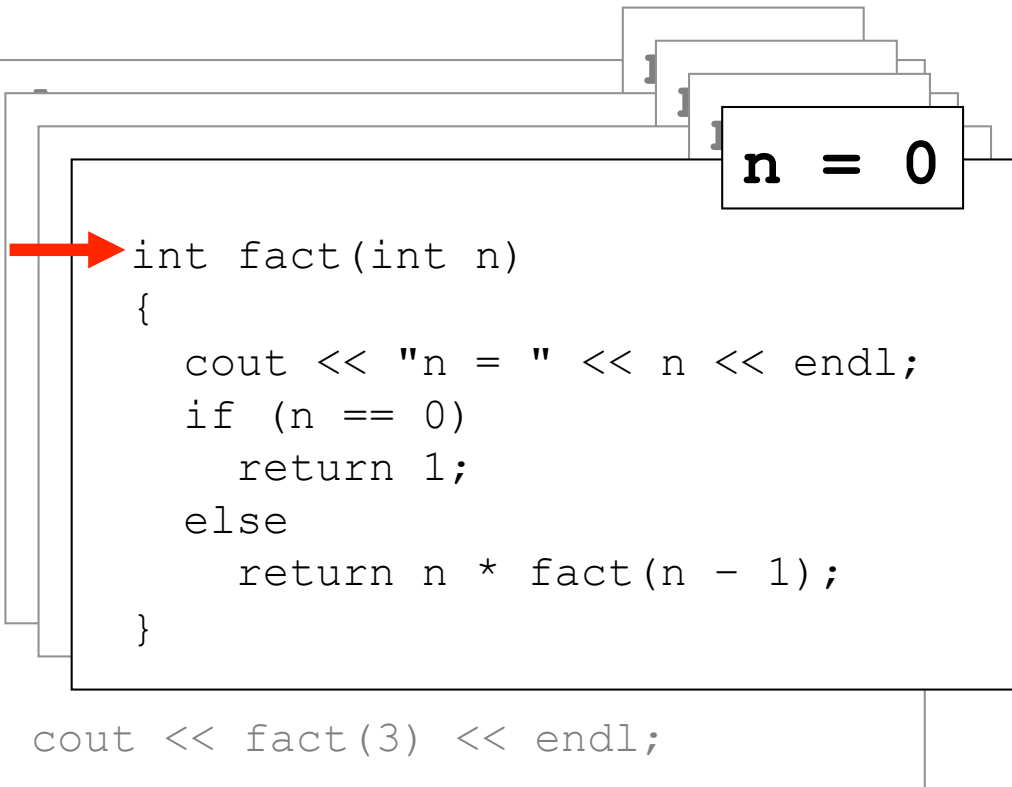
 $\underbrace{\text{fact}(n - 1)}_0$

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
```

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}

cout << fact(3) << endl;
```



```
n = 3
n = 2
n = 1
```

n = 0

```
int fact(int n)
{
  cout << "n = " << n << endl;
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
n = 0
```

n = 0

```
int fact(int n)
{
    cout << "n = " << n << endl;
    → if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
n = 0
```

n = 0

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        → return 1;
    else
        return n * fact(n - 1);
}
```

```
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
n = 0
```

n = 1

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        → return n * fact(n - 1);
}
```

cout << fact(3) << endl;

```
n = 3
n = 2
n = 1
n = 0
```

n = 2

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        → return n * fact(n - 1);
}
```

...

2
cout << fact(3) << endl;

```
n = 3
n = 2
n = 1
n = 0
```

n = 3

A.


```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        → return n * fact(n - 1);
}
...
        2
        6
cout << fact(3) << endl;
```

```
n = 3
n = 2
n = 1
n = 0
```

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...


 `cout << fact(3) << endl;`

```
n = 3
n = 2
n = 1
n = 0
```

A.

```
int fact(int n)
{
    cout << "n = " << n << endl;
    if (n == 0)
        return 1;
    else
        return n * fact(n - 1);
}
```

...

 cout << fact(3) << endl;



6

n = 3

n = 2

n = 1

n = 0

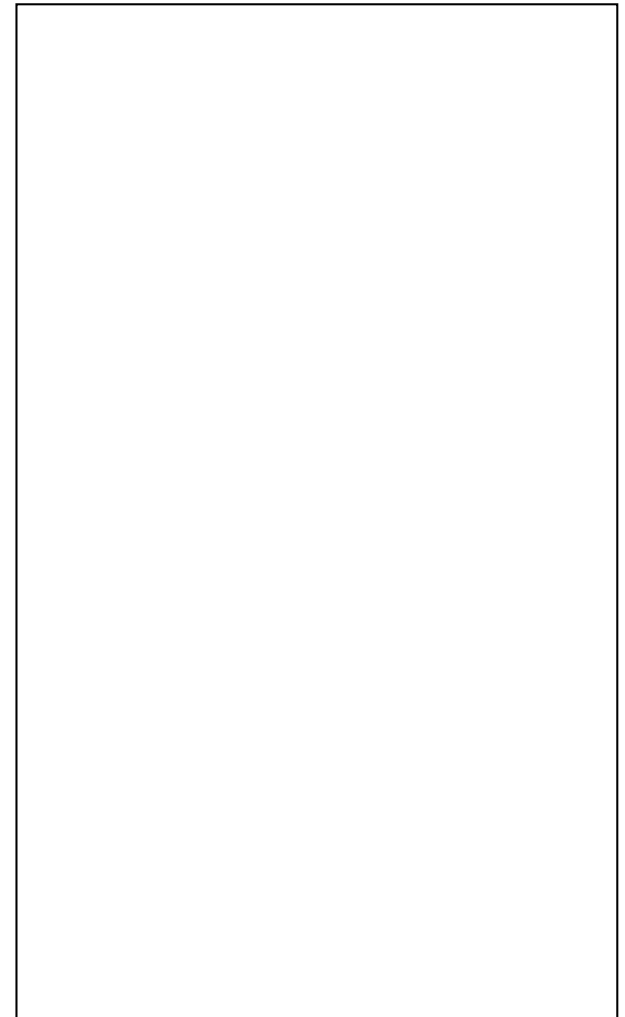
6

B.

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}

...

d(3);
```

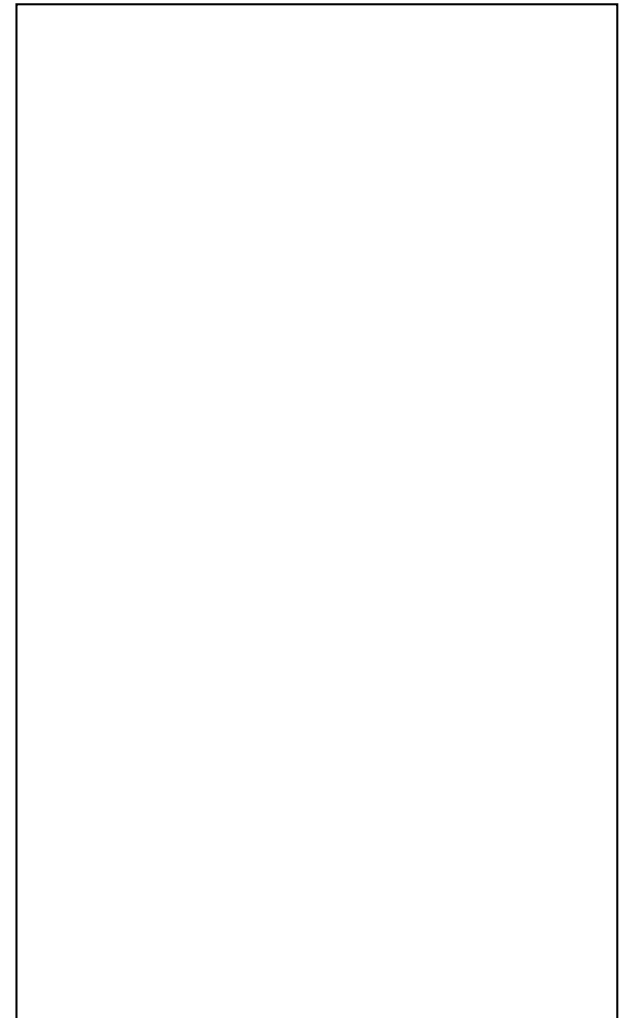


B.

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}


...

→ d(3);
```



n = 3

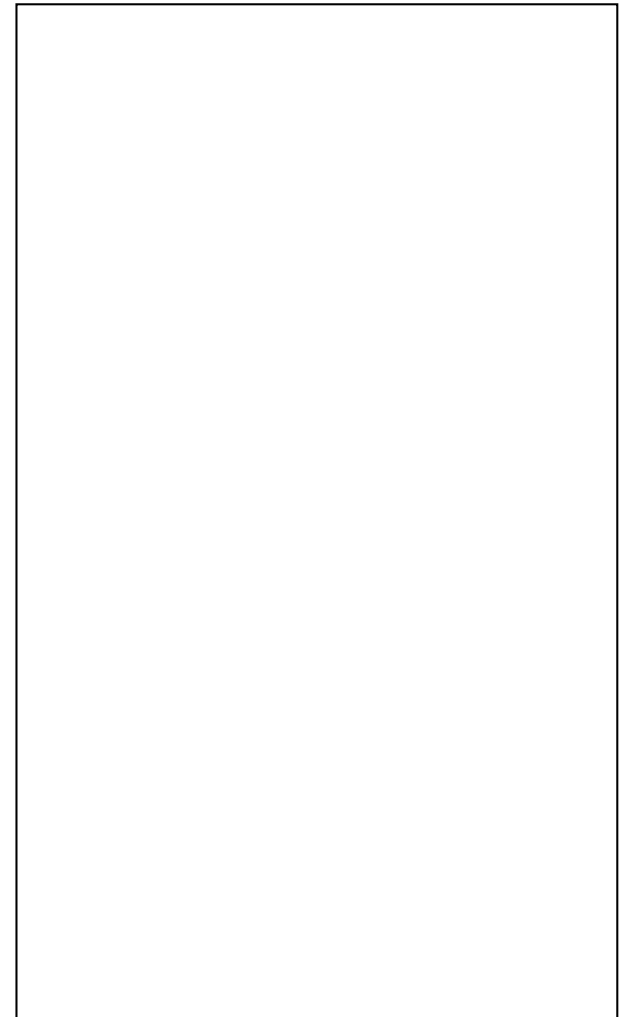
B.



```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}

...

d(3);
```



n = 3

B.

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
  if (n > 0)
```

```
  {
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
  }
```

```
}
```


```
...
```

```
d(3);
```

```
n = 3
```

n = 3

B.


```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
         d(n - 1);
        d(n - 1);
    }
}

...

d(3);
```

n = 3

n = 2



```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}
```

...

```
d(3);
```

```
n = 3
```

n = 2

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
if (n > 0)
```

```
{
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
}
```

```
}
```


```
...
```

```
d(3);
```

```
n = 3
```

```
n = 2
```

n = 2

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
         d(n - 1);
        d(n - 1);
    }
}
```

...

```
d(3);
```

```
n = 3
n = 2
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}
```

```
d(3);
```

```
n = 3
n = 2
```

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
  if (n > 0)
```

```
  {
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
  }
```

```
}
```

```
n = 1
```

```
d(3);
```

```
n = 3
```

```
n = 2
```

```
n = 1
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        → d(n - 1);
        d(n - 1);
    }
}
```

d(3);

```
n = 3
n = 2
n = 1
```

```
void d(int n)
{
  cout << "n = " << n << endl;
  if (n > 0)
  {
    d(n - 1);
    d(n - 1);
  }
}
```

```
d(3);
```

```
n = 3
n = 2
n = 1
```

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
if (n > 0)
```

```
{
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
}
```

```
}
```

n = 0

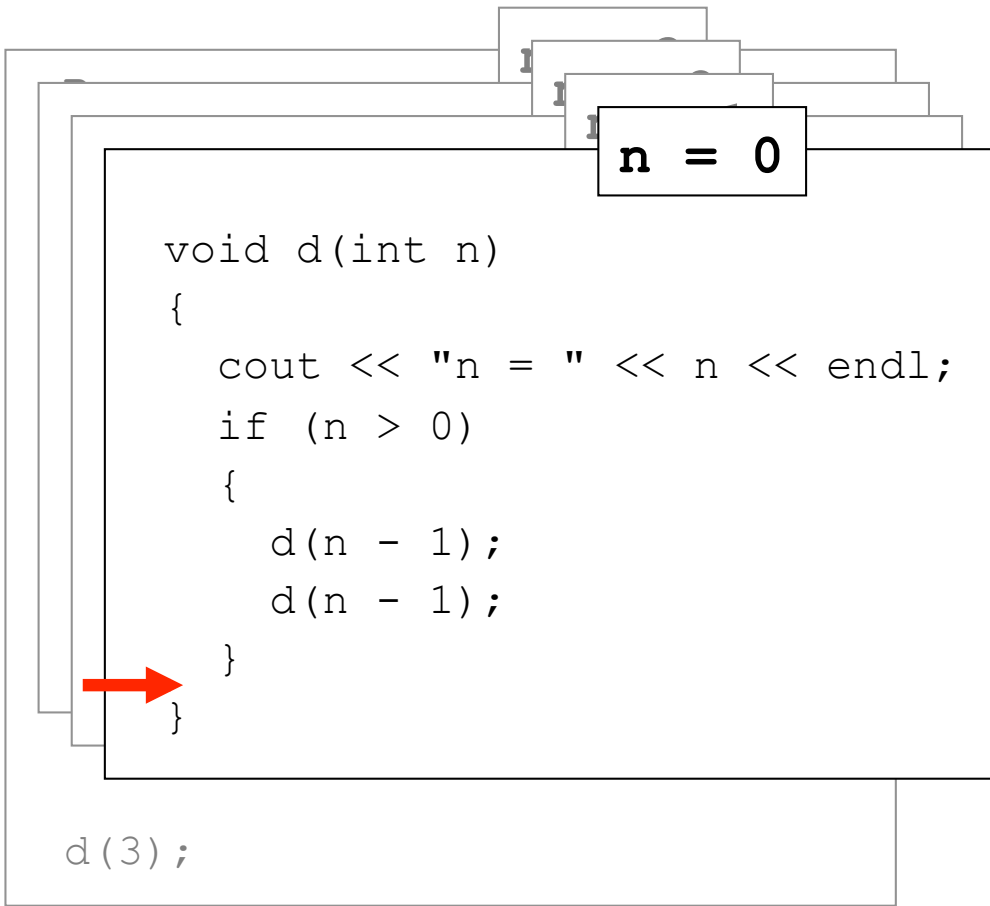
```
d(3);
```

```
n = 3
```

```
n = 2
```

```
n = 1
```

```
n = 0
```



```
n = 3
n = 2
n = 1
n = 0
```


n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        → d(n - 1);
        d(n - 1);
    }
}
```

d(3);

```
n = 3
n = 2
n = 1
n = 0
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
         d(n - 1);
    }
}
```

d(3);

```
n = 3
n = 2
n = 1
n = 0
```

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}
```

```
d(3);
```

```
n = 3
n = 2
n = 1
n = 0
```

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
if (n > 0)
```

```
{
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
}
```

```
}
```

n = 0

```
d(3);
```

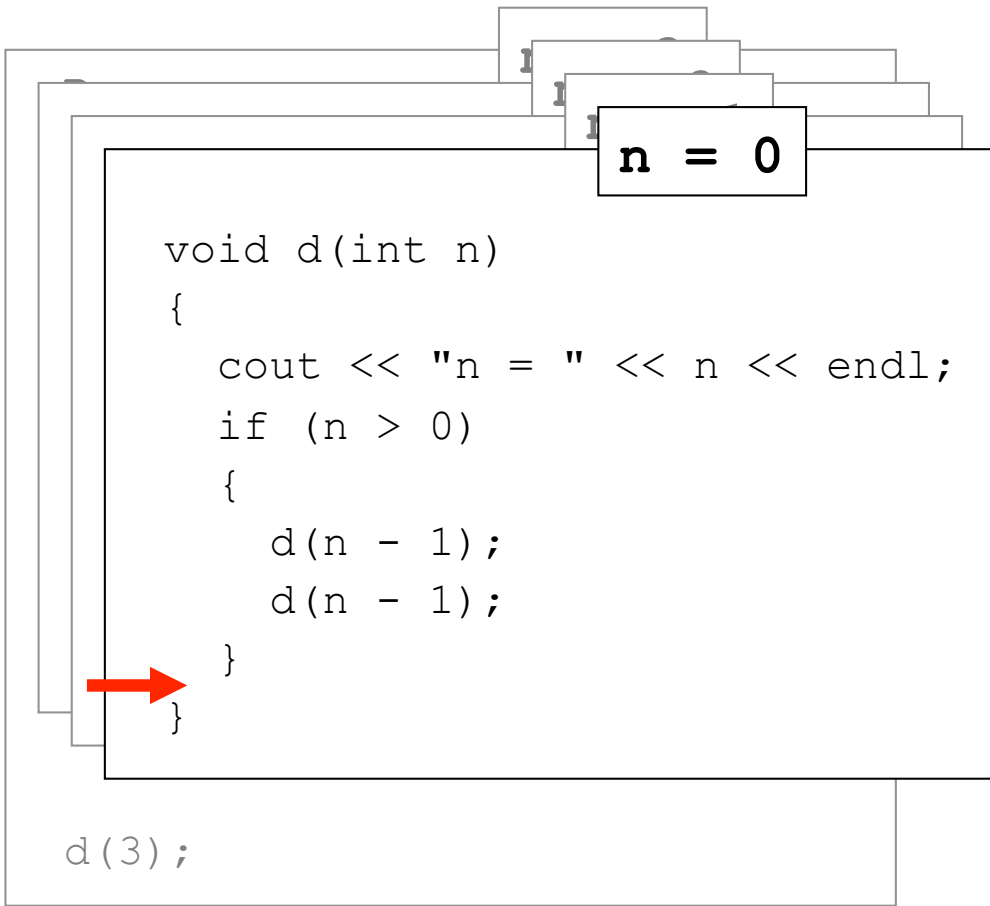
```
n = 3
```

```
n = 2
```

```
n = 1
```


```
n = 0
```

```
n = 0
```



```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
         d(n - 1);
    }
}
```

d(3);

```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}
```



```
d(3);
```

```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 2


```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        → d(n - 1);
        d(n - 1);
    }
}
```

...

```
d(3);
```

```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 2

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
         d(n - 1);
    }
}
```

...

```
d(3);
```

```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 1

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}
```

```
d(3);
```

```
n = 3
n = 2
n = 1
n = 0
n = 0
```

n = 1

```
void d(int n)
```

```
{
```

```
→ cout << "n = " << n << endl;
```

```
  if (n > 0)
```

```
  {
```

```
    d(n - 1);
```

```
    d(n - 1);
```

```
  }
```

```
}
```

```
d(3);
```

```
n = 3
```

```
n = 2
```

```
n = 1
```

```
n = 0
```

```
n = 0
```

```
n = 1
```

B.

```
void d(int n)
{
    cout << "n = " << n << endl;
    if (n > 0)
    {
        d(n - 1);
        d(n - 1);
    }
}

...

d(3);
```

```
n = 3
n = 2
n = 1
n = 0
n = 0
n = 1
n = 0
n = 0
n = 0
n = 0
n = 1
n = 0
n = 0
```

flock of birds

A partir de quelques règles très simples modélisant le déplacement d'un individu, on obtient un comportement de groupe complexe. Nous allons considérer un modèle simplifié où un individu tend à s'éloigner des zones trop peuplées, et à se rapprocher des zones peu peuplées.

Un individu sera défini par la structure suivante :

```
struct Oiseau
{
    float x, y;
    float dx, dy;
};
```

où x et y désignent les coordonnées dans le plan 2D de l'oiseau, et dx et dy son vecteur vitesse.

Un essaim sera défini par la structure suivante:

```
struct Essaim
{
    Oiseau * oiseaux;
    int nb_oiseaux;
};
```

où `oiseaux` est un pointeur sur un tableau d'Oiseaux, et `nb_oiseaux` le nombre d'éléments de ce tableau.

Ecrivez la fonction

```
Essaim * alloue_essaim(int n)
```

qui permet d'allouer une structure `Essaim` comportant `n` individus.

Ecrivez la fonction

```
Essaim * alloue_essaim(int n)
```

qui permet d'allouer une structure `Essaim` comportant `n` individus.

```
Essaim * alloue_essaim(int n)
```

```
{
```

```
    Essaim * e = new Essaim;
```

```
    e->nb_oiseaux = n;
```

```
    e->oiseaux = new Oiseau[n];
```

```
    return e;
```

```
}
```

Ecrivez la fonction

```
void initialise_oiseau(Oiseau * o, int longueur, int hauteur)
```

qui initialise l'individu pointé par `o`.

Les coordonnées x et y seront tirées aléatoirement entre 0 et longueur, et 0 et hauteur respectivement.

Le vecteur vitesse défini par les champs dx et dy devra avoir une norme égale à 2, et une direction aléatoire.

Pour cela, on tirera un angle α aléatoirement entre 0 et 2Pi , et dx sera initialisée à $2\cos\alpha$, dy à $2\sin\alpha$. La valeur de Pi peut être obtenue avec la constante `M_PI`.

Ecrivez la fonction

```
void initialise_oiseau(Oiseau * o, int longueur, int hauteur)
```

qui initialise l'individu pointé par `o`.

Les coordonnées `x` et `y` seront tirées aléatoirement entre 0 et `longueur`, et 0 et `hauteur` respectivement.

Le vecteur vitesse défini par les champs `dx` et `dy` devra avoir une norme égale à 2, et une direction aléatoire. Pour cela, on tirera un angle α aléatoirement entre 0 et 2Pi , et `dx` sera initialisée à $2 \cos \alpha$, `dy` à $2 \sin \alpha$. La valeur de Pi peut être obtenue avec la constante `M_PI`.

```
void initialise_oiseau(Oiseau * o, int longueur, int hauteur)
{
    o->x = rand() % longueur;
    o->y = rand() % hauteur;

    float angle = float(rand()) / RAND_MAX * 2 * M_PI;
    o->dx = 2 * cos(angle);
    o->dy = 2 * sin(angle);
}
```

Ecrivez la fonction

```
void initialise_essaim(Essaim * e,  
                        int longueur, int hauteur)
```

qui appelle la fonction

```
void initialise_oiseau(Oiseau * o,  
                       int longueur, int hauteur)
```

pour chacun des individus de l'essaim pointé par e.

Ecrivez la fonction

```
void initialise_essaim(Essaim * e, int longueur, int hauteur)
```

qui appelle la fonction

```
void initialise_oiseau(Oiseau * o, int longueur, int hauteur)
```

pour chacun des individus de l'essaim pointé par e.

```
void initialise_essaim(Essaim * e, int longueur, int hauteur)
{
    for(int i = 0; i < e->nb_oiseaux; i++)
        initialise_oiseau(e->oiseaux + i, longueur, hauteur);
}
```

ou

```
void initialise_essaim(Essaim * e, int longueur, int hauteur)
{
    for(int i = 0; i < e->nb_oiseaux; i++)
        initialise_oiseau(&(e->oiseaux[i]), longueur, hauteur);
}
```

Ecrivez la fonction

```
void affiche_essaim(SimpleWindow * window, Essaim * e)
```

qui dessine chacun des individus de `e` dans la fenêtre pointée par `window`. Un individu sera simplement représenté par un point grâce à la fonction `drawPoint`.

Ecrivez la fonction

```
void affiche_essaim(SimpleWindow * window, Essaim * e)
```

qui dessine chacun des individus de `e` dans la fenêtre pointée par `window`. Un individu sera simplement représenté par un point grâce à la fonction `drawPoint`.

```
void affiche_essaim(SimpleWindow * window, Essaim * e)
```

```
{
```

```
    for(int i = 0; i < e->nb_oiseaux; i++)
```

```
        window->drawPoint(int(e->oiseaux[i].x), int(e->oiseaux[i].y));
```

```
}
```