

Le titre de votre TIPE :  
avec éventuellement un sous titre

Présentation de **Prénom-Composé NOM**

travail réalisé avec **binôme**

# Plan

1. Première section : le haut des slides  
À propos de l'en-tête
2. Le corps des diapositives
3. Faire des animations
4. Troisième section : le bas des slides
5. Gérer les annexes de code

# Titre d'une slide avant la sous-section

Ici on n'a pas encore de titre de sous-section dans le badeau du haut.

## Titre d'une slide dans la sous-section

Ici on a un titre de sous-section, contrairement à la slide 1

Regarder le code ici pour référencer une slide avec `\label` et la citer avec son numéro grâce à `\ref`

# Ce qui apparaît dans l'en-tête

Dans la **première ligne** :

- la version courte du titre, précisée en option de `\title`  
( *en option = entre crochets, avant les accolades* )
- la version courte du nom, voire des initiales, redéfinir la commande `\newcommand{\initiales}{Petit Nom}`
- la version courte de la date, précisée en option de `\date`

Dans la **deuxième ligne** :

- le numéro et le titre de la section, sauf si le numéro est nul  
*s'il est précisé en option de `\section` le titre court est utilisé*
- le numéro et le titre de la sous-section, sauf si le numéro est nul  
*s'il est précisé en option de `\subsection` le titre court est utilisé*

1. Première section : le haut des slides

2. Le corps des diapositives

- Équations et formules

- Environnements description et minipage

- Minipages et inclusions (code, image)

- Une diapositive avec du pseudo-code

- Astuces pour s'étaler en largeur

3. Faire des animations

4. Troisième section : le bas des slides

5. Gérer les annexes de code

# Suite d'équations avec align\*

$$\begin{aligned} a &= b + c + d * \sum_{i=1}^n x_i \\ &= b + c + d * \sum_{i=1}^n (z_i - y_i) \\ &\leq B + c + d * \sum_{i=1}^n (z_i - y_i) \end{aligned}$$

*ici une petite explication*

# Formules centrées, éventuellement encadrées

Voilà une formule juste centrée car entre `$$` et `$$`

$$A = \sum_{i=1}^n a_i + b_i$$

Voilà une formule encadrée avec `\fbox` et centrée

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$

Voilà une formule encadrée en couleurs avec `\fcolorbox` et centrée

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$

NB : le deuxième argument de `\fcolorbox` fixe la couleur du fond, je déconseille de l'utiliser avec une couleur franche, ainsi on évitera ça :

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E \text{ et même } \Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$



## Exemples d'utilisation de l'environnement description

On peut par exemple décrire le problème...

entrées : première donnée  
          deuxième donnée  
          dernière donnée

sortie : le résultat

## Exemples d'utilisation de l'environnement minipage

Là, à gauche, une colonne qui fait presque la moitié de la slide, dans laquelle je peux écrire plein de choses et dont je peux constater la largeur complète grâce à `\hrule`

Là, à droite, une colonne qui fait presque la moitié de la slide, dans laquelle je peux écrire plein de choses et dont je peux constater la largeur grâce à `\hrule` que j'ai mis cette fois en début de paragraphe

NB : par défaut, l'alignement vertical des minipages est "centré", ce qu'on peut modifier avec l'option `[t]`. Ici on est entre `\begin{minipage}[t]{0.64\textwidth}` et `\end{minipage}`

Ici une petite largeur donc peu de mots font vite une grande hauteur, Remarquez que bien que plus haute, le haut de cette minipage est aligné avec celui de celle de gauche

## Attention avec minipage sous beamer

Comme on peut le voir ici, la somme des largeurs indiquées dans les minipage doit être moins que `\textwidth`, sans quoi la dernière minipage est renvoyée à la ligne comme on peut le voir ici où on a juxtaposé une `{minipage}{0.6\textwidth}` et une `{minipage}{0.4\textwidth}`.



## Du texte et une image côte à côte

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc.



## Du texte et du code C côte à côte

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque.

```
1  int main(){
2      int un = 1;
3      int deux = 2;
4      printf("%d\n", un + deux);
5      return 0;
6  }
```

# Une image et du code Ocaml côte à côte



```
1  type nb =  
2      | Int of int  
3      | Flo of float  
4  
5  let add (x:nb)(y:nb):nb =  
6      match x,y with  
7      | Int a, Int b -> Int(a + b)  
8      | Int a, Flo b ->  
9          ↪ Flo((float_of_int a) +. b)  
10     | Flo a, Int b -> Flo( a +.  
11         ↪ (float_of_int b))  
12     | Flo a, Flo b -> Flo(a +. b)
```

## Deux images côte à côte



Une première image : un plan



Une deuxième image : la même

# Pseudo-code avec algorithm du package algo2e

---

## Algorithme 1 : Dijkstra

---

**Entrée** : Un graphe pondéré  $G = (S, A, c)$  où  $c : A \rightarrow \mathbb{R}^+$ ,  $s \in S$

**Sortie** : La distance de  $s$  à chaque sommet de  $S$

Initialiser  $\delta[u]$  à  $+\infty$  pour  $u \in S$  ;

Initialiser  $\pi[u]$  à ?? pour  $u \in S$  ;

$\delta[s] \leftarrow 0$  ;

$\pi[s] \leftarrow s$  ;

$\text{todo} \leftarrow \{s\}$  ;

**tant que**  $\text{todo} \neq \emptyset$  **faire**

    Soit  $u \in \text{todo}$  minimisant  $\delta[u]$  ;

**pour tout**  $v \in \text{voisin}(u)$  **faire**

        Relacher( $G, u, v, \delta, \pi, \text{todo}$ )

    Ôter  $u$  de  $\text{todo}$  ;

**retourner**  $\delta$

---



## Moins de marge (ou plus ?)

Ce paragraphe est entre `\begin{adjustwidth}{-1.5 em}{-1.5em}` et `\end{adjustwidth}` ce qui lui permet de s'étaler d'un bout à l'autre de la slide. En fait on a réduit les marges à gauche et à droite de 1.5em, on peut aussi augmenter les marges (réduire la largeur du texte donc) en mettant des valeurs positives, cf paragraphe juste après

Ce paragraphe est entre

`\begin{adjustwidth}{2 em}{5em}` et

`\end{adjustwidth}` ce qui lui permet de s'étaler sur une largeur plus petite, et décalée vers la gauche. En fait on a augmenté la marge à gauche de 2em, et celle à droite de 5em

# Plan

1. Première section : le haut des slides

2. Le corps des diapositives

3. Faire des animations

- Sur une diapo quelconque

- Avec des images

- Avec du tikz

- Animation et numérotation

4. Troisième section : le bas des slides

5. Gérer les annexes de code

## Pourquoi des animations ?

- Faire des animations sur une diapositive permet que son contenu arrive progressivement, de manière synchrone avec votre discours. Cela évite que votre public lise une formule compliquée encadrée en fin de diapositive au lieu de vous écouter expliquer le début de la diapositive.
- Dans le cas où vous avez fait une simulation dynamique, cela permet de simuler un petit dessin animé dans un fichier pdf autorisé aux concours.

## Formules au fur et à mesure avec `\pause`

On reprend la suite d'équations présentée plus haut, mais grâce à des commandes `\pause`, celles-ci apparaissent au fur et à mesure.

## Formules au fur et à mesure avec `\pause`

On reprend la suite d'équations présentée plus haut, mais grâce à des commandes `\pause`, celles-ci apparaissent au fur et à mesure. Voilà une formule juste centrée car entre `$$` et `$$`

$$A = \sum_{i=1}^n a_i + b_i$$

## Formules au fur et à mesure avec `\pause`

On reprend la suite d'équations présentée plus haut, mais grâce à des commandes `\pause`, celles-ci apparaissent au fur et à mesure. Voilà une formule juste centrée car entre `$$` et `$$`

$$A = \sum_{i=1}^n a_i + b_i$$

Voilà une formule encadrée avec `\fbox` centrée

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$

## Formules au fur et à mesure avec `\pause`

On reprend la suite d'équations présentée plus haut, mais grâce à des commandes `\pause`, celles-ci apparaissent au fur et à mesure. Voilà une formule juste centrée car entre `$$` et `$$`

$$A = \sum_{i=1}^n a_i + b_i$$

Voilà une formule encadrée avec `\fbox`et centrée

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$

Voilà une formule encadrée en couleurs avec `\fcolorbox` et centrée

$$\Delta_u^{\text{early}}(E, T) \geq 0 \text{ if } u \in E$$

## Exemple avec `\only`

N'apparaît que sur la slide "1", grâce à `\only<1>{...}`,  
puis disparaît et laisse sa place.

partie là tout le temps, car hors d'un `\only<1>`



## Exemple avec `\only`

partie là tout le temps, car hors d'un `\only<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à  
`\only<2-4>\{...\}`,  
puis disparaît et laisse sa place.

N'apparaît que sur la slide "2", grâce à `\only<2>\{...\}`,  
puis disparaît et laisse sa place.

## Exemple avec `\only`

partie là tout le temps, car hors d'un `\only<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à  
`\only<2-4>\{...\}`,  
puis disparaît et laisse sa place.

Apparaît de la slide "3" à la fin, grâce à `\only<3->\{...\}`, avant sa  
place n'est pas reversée

## Exemple avec `\only`

partie là tout le temps, car hors d'un `\only<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à  
`\only<2-4>\{...\}`,  
puis disparaît et laisse sa place.

Apparaît de la slide "3" à la fin, grâce à `\only<3->\{...\}`, avant sa  
place n'est pas reversée

## Exemple avec `\onslide`

N'apparaît que sur la slide "1", grâce à `\onslide<1>\{...\}`,  
mais ici sa place est reversée, pas réutilisée.

`partie là tout le temps`, car hors d'un `\onslide<1>`

## Exemple avec `\onslide`

partie là tout le temps, car hors d'un `\onslide<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à `\only<2-4>\{...\}`,  
mais ici sa place est reversée, pas réutilisée.

N'apparaît que sur la slide "2", grâce à `\onslide<2>\{...\}`,  
mais ici sa place est reversée, pas réutilisée.

## Exemple avec `\onslide`

partie là tout le temps, car hors d'un `\onslide<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à `\only<2-4>\{...\}`,  
mais ici sa place est reversée, pas réutilisée.

Apparaît de la slide "3" à la fin, grâce à `\onslide<3->\{...\}`, mais sa place est reversée, pas réutilisée.

## Exemple avec `\onslide`

partie là tout le temps, car hors d'un `\onslide<1>`

N'apparaît que sur les slides "2", "3" et "4", grâce à `\only<2-4>\{...\}`,  
mais ici sa place est reversée, pas réutilisée.

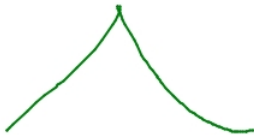
Apparaît de la slide "3" à la fin, grâce à `\onslide<3->\{...\}`, mais sa place est reversée, pas réutilisée.

## Images animées - exemple - le début à la main

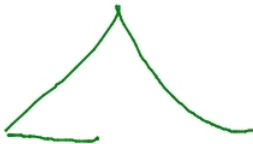




## Images animées - exemple - le début à la main



## Images animées - exemple - le début à la main



## Images animées - exemple - le tout avec `\foreach`

Image 1



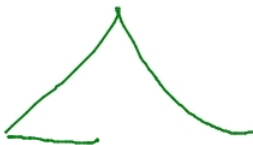
## Images animées - exemple - le tout avec `\foreach`

Image 2



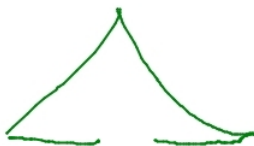
## Images animées - exemple - le tout avec `\foreach`

Image 3



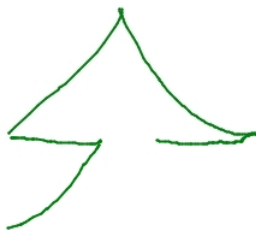
## Images animées - exemple - le tout avec `\foreach`

Image 4



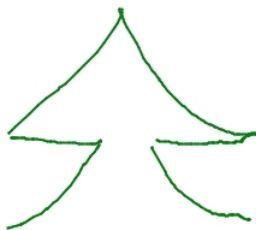
## Images animées - exemple - le tout avec `\foreach`

Image 5



## Images animées - exemple - le tout avec `\foreach`

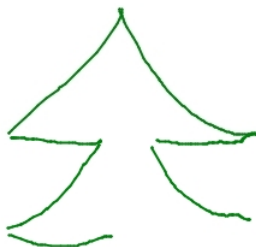
Image 6





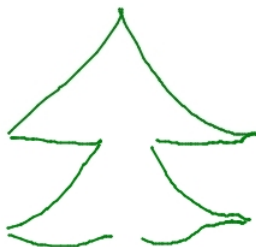
## Images animées - exemple - le tout avec `\foreach`

Image 7



## Images animées - exemple - le tout avec `\foreach`

Image 8



## Images animées - exemple - le tout avec `\foreach`

Image 9



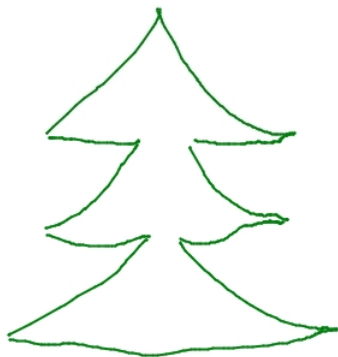
## Images animées - exemple - le tout avec `\foreach`

Image 10



## Images animées - exemple - le tout avec `\foreach`

Image 11



## Images animées - explications

- On nomme les images à afficher successivement avec des noms comme `im1.jpg`, `im2.jpg`... où la numérotation suit bien sûr l'ordre dans lequel doivent apparaître ces images.
- On peut alors faire des copier-coller de `includegraphics{im1.jpg}` en changeant le numéro, ou bien utiliser `\foreach` pour faire une boucle
- Attention le poids du PDF de présentation augmente très vite, or vous êtes limités en taille pour le PDF à envoyer aux concours (à 5Mo je crois). Vous pouvez commenter la slide qui inclut les 11 images du sapin, recompiler, et observer que la taille du PDF a réduit d'un volume équivalent à la somme des 11 images (un peu plus même). Donc il faut avoir peu d'images (pas 50) ou des petites images, ou bien se mettre au tikz...

## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe





## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

\*

On peut utiliser `\draw<n->` pour animer directement une figure tikz.

**Attention**, la figure est remplacée automatiquement à chaque étape en fonction de la place occupée par la figure à cette étape, donc si on veut voir le point bouger dans l'exemple ci-dessous, il faut un cadre fixe



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :





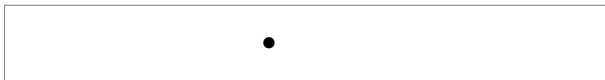
## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



## Exemple avec tikZ - v1

De plus, cet ajustement automatique rend parfois les animations désagréables, comme juste avant les deux derniers points qui ont tout fait bouger. Il faut donc mettre un cadre qui englobe la superposition de toutes les étapes. Exemple ci-dessous :



# Animation et numérotation

Comme on peut le voir sur les slides précédentes, il y a par défaut un seul numéro pour les différents slides issues des animations d'une seule "frame". Autrement dit on a un numéro de slide pour chaque occurrence d'un environnement `frame`.

Dans la plupart des cas ce comportement par défaut est opportun, mais dans le cas où l'on veut pouvoir faire référence à une étape précise de l'animation, comme dans un déroulé d'algorithme par exemple, on peut ajouter des numéros. Pour cela, on incrémente *manuellement* le compteur `framenumber` grâce à la commande `\addtocounter{framenumber}{1}`.

# Numéroter chaque étape d'une animation avec `\pause`

Si on utilise des `\pause`, il suffit de mettre une seule fois l'incréméntation `\addtocounter{framenum}{1}` juste après le `\begin{frame}` car elle est ré-évaluée pour chaque slide. Comme elle est évaluée la première fois alors que le `\begin{frame}` incrémente déjà automatiquement ce compteur, on compense en décrémentant avant le `\begin{frame}`. Exemple ici :

► étape 1



## Numéroter chaque étape d'une animation avec `\pause`

Si on utilise des `\pause`, il suffit de mettre une seule fois l'incrémementation `\addtocounter{framenum}{1}` juste après le `\begin{frame}` car elle est ré-évaluée pour chaque slide. Comme elle est évaluée la première fois alors que le `\begin{frame}` incrémente déjà automatiquement ce compteur, on compense en décrémentant avant le `\begin{frame}`. Exemple ici :

- ▶ étape 1
- ▶ étape 2

## Numéroter chaque étape d'une animation avec `\pause`

Si on utilise des `\pause`, il suffit de mettre une seule fois l'incrémement `\addtocounter{framenum}{1}` juste après le `\begin{frame}` car elle est ré-évaluée pour chaque slide. Comme elle est évaluée la première fois alors que le `\begin{frame}` incrémente déjà automatiquement ce compteur, on compense en décrémentant avant le `\begin{frame}`. Exemple ici :

- ▶ étape 1
- ▶ étape 2
- ▶ étape 3

## Numéroter chaque étape d'une animation avec `\pause`

Si on utilise des `\pause`, il suffit de mettre une seule fois l'incréméntation `\addtocounter{framenum}{1}` juste après le `\begin{frame}` car elle est ré-évaluée pour chaque slide. Comme elle est évaluée la première fois alors que le `\begin{frame}` incrémente déjà automatiquement ce compteur, on compense en décrémentant avant le `\begin{frame}`. Exemple ici :

- ▶ étape 1
- ▶ étape 2
- ▶ étape 3

## Numéroter chaque étape d'une animation avec `\only`

Avec des `\only` ça marche pareil : il suffit de mettre une seule fois `\addtocounter{framenum}{1}` juste après le `\begin{frame}` et une seule fois `\addtocounter{framenum}{-1}` juste avant.

Exemple ici :

- ▶ étape 1

## Numéroter chaque étape d'une animation avec `\only`

Avec des `\only` ça marche pareil : il suffit de mettre une seule fois `\addtocounter{framenum}{1}` juste après le `\begin{frame}` et une seule fois `\addtocounter{framenum}{-1}` juste avant.

Exemple ici :

- ▶ étape 2 et suivantes

## Numéroter chaque étape d'une animation avec `\only`

Avec des `\only` ça marche pareil : il suffit de mettre une seule fois `\addtocounter{framenum}{1}` juste après le `\begin{frame}` et une seule fois `\addtocounter{framenum}{-1}` juste avant.

Exemple ici :

- ▶ étape 2 et suivantes
- ▶ étape 3

# Plan

1. Première section : le haut des slides
2. Le corps des diapositives
3. Faire des animations
4. Troisième section : le bas des slides
  - Pour les références
  - Numérotation
5. Gérer les annexes de code

## Petit bandeau de citation d'une référence

Se fait à la main en utilisant la commande `\bandeauREF`.

En plus il faut ajuster à la main un `\vspace` pour le forcer à être bien en bas... (`\vfill`) ne marche pas).



# La numérotation des diapo

Compte les diapo sans multiplicité : si le contenu d'un diapo arrive petit à petit parce qu'on a utilisé des

Ne compte pas la page de titre, ni les pages de plan.

Compte le nombre total de slides

Si vous avez n slides bonus, vous pouvez fausser (mais rectifier) le nombre total de slides avec `\addtocounter\{framenum\}\{-n\}`

# La dernière vraie slide

Elle est donc numérotée 34/ 34

# Slide bonus qui ne compte pas

Elle est donc numérotée 35/ 34

# Encore une slide bonus qui ne compte pas

Elle est donc numérotée 36/ 34

# Plan

1. Première section : le haut des slides
2. Le corps des diapositives
3. Faire des animations
4. Troisième section : le bas des slides
5. Gérer les annexes de code

## Pourquoi inclure son code ?

La notice du TIPE impose depuis la session 2024 de produire l'intégralité de son code dans la présentation.

Afin d'éviter des captures d'écrans ou des copier-coller de texte sans mise en forme ni coloration syntaxique, on peut utiliser le package `minted` pour inclure du code depuis ses fichiers sources.

Cela a aussi l'avantage de la *robustesse* : si vous modifiez votre code, il suffit de recompiler et c'est le code modifié qui sera importé. Ainsi mettre à jour ses slides est aisé.

## Comment inclure son code ?

La syntaxe pour inclure l'intégralité d'un fichier est la suivante.

```
\inputminted[] {langage} {chemin/fichier.extension}
```

langage désigne la syntaxe selon laquelle le code doit être coloré, par exemple ocaml, c, sql, python...

La syntaxe pour inclure un extrait du fichier (de la ligne n1 à la ligne n2) est la suivante.

```
\inputminted[firstline=n1, lastline=n2,  
→ firstnumber=1] {langage} {chemin/fichier.extension}
```

Sans l'option firstnumber=1, les numéros de lignes indiqués sont les numéros de ligne dans le fichier.

## Code trop long ?

Votre code peut être trop long à différents égards.

- ▶ Si un fichier est trop long pour tenir sur une slide, s'il a trop de lignes et seules les premières sont visibles sur la slide, il suffit de placer la commande `\inputminted` en dehors d'un environnement `frame`, c'est-à-dire pas entre `\begin{frame}` et `\end{frame}`.
- ▶ Si le fichier est tellement long que, avec la solution précédente, le code du fichier est étalé sur une quinzaine de slides parmi lesquelles il est difficile de naviguer, alors il peut être pertinent de faire des imports par extraits, entrecoupés de `\subsection` ou `\subsubsection` qui placeront ainsi des repères, et des liens cliquables dans la table des matières.
- ▶ Si le code est constitué de trop nombreux fichiers, alors on peut là aussi organiser les imports des différents fichiers en `\subsection` ou `\subsubsection`, ou bien utiliser une boucle `\foreach`.

Les slides suivantes illustrent les solutions proposées ici.



## Annexe : Code du TIPE

Fichier court `test.ml`

Fichier long avec découpage auto `test_long.ml`

Fichier long avec découpage manuel `test_long.ml`

Série de fichiers importée avec une boucle

# Fichier court sur une slide

```
1  type nb =  
2    | Int of int  
3    | Flo of float  
4  
5  let add (x:nb) (y:nb):nb =  
6    match x,y with  
7    | Int a, Int b -> Int(a + b)  
8    | Int a, Flo b -> Flo((float_of_int a) +. b)  
9    | Flo a, Int b -> Flo( a +. (float_of_int b))  
10   | Flo a, Flo b -> Flo(a +. b)
```

```
1  type nb =
2    | Int of int
3    | Flo of float
4
5  let add_v1 (x:nb)(y:nb):nb =
6    match x,y with
7    | Int a, Int b -> Int(a + b)
8    | Int a, Flo b -> Flo((float_of_int a) +. b)
9    | Flo a, Int b -> Flo( a +. (float_of_int b))
10   | Flo a, Flo b -> Flo(a +. b)
11
12  let add_v2 (x:nb)(y:nb):nb =
13    match x,y with
14    | Int a, Int b -> Int(a + b)
15    | Flo a, Flo b -> Flo(a +. b)
16    | Int a, Flo b -> Flo((float_of_int a) +. b)
17    | Flo a, Int b -> Flo( a +. (float_of_int b))
```

```
18
19 let add_v3 (x:nb)(y:nb):nb =
20     match x,y with
21     | Flo a, Int b -> Flo( a +. (float_of_int b))
22     | Flo a, Flo b -> Flo(a +. b)
23     | Int a, Int b -> Int(a + b)
24     | Int a, Flo b -> Flo((float_of_int a) +. b)
25
26 let add_v4 (x:nb)(y:nb):nb =
27     match x,y with
28     | Int a, Int b -> Int(a + b)
29     | Int a, Flo b -> Flo((float_of_int a) +. b)
30     | Flo a, Flo b -> Flo(a +. b)
31     | Flo a, Int b -> Flo( a +. (float_of_int b))
32
33 let add_v5 (x:nb)(y:nb):nb =
34     match x,y with
```

```
35 | Int a, Flo b -> Flo((float_of_int a) +. b)
36 | Int a, Int b -> Int(a + b)
37 | Flo a, Int b -> Flo( a +. (float_of_int b))
38 | Flo a, Flo b -> Flo(a +. b)
```

# La première version

```
1  type nb =  
2    | Int of int  
3    | Flo of float  
4  
5  let add_v1 (x:nb)(y:nb):nb =  
6    match x,y with  
7    | Int a, Int b -> Int(a + b)  
8    | Int a, Flo b -> Flo((float_of_int a) +. b)  
9    | Flo a, Int b -> Flo( a +. (float_of_int b))  
10   | Flo a, Flo b -> Flo(a +. b)
```

## La deuxième et la troisième version

```
1  let add_v2 (x:nb)(y:nb):nb =
2      match x,y with
3      | Int a, Int b -> Int(a + b)
4      | Flo a, Flo b -> Flo(a +. b)
5      | Int a, Flo b -> Flo((float_of_int a) +. b)
6      | Flo a, Int b -> Flo( a +. (float_of_int b))
7
8  let add_v3 (x:nb)(y:nb):nb =
9      match x,y with
10     | Flo a, Int b -> Flo( a +. (float_of_int b))
11     | Flo a, Flo b -> Flo(a +. b)
12     | Int a, Int b -> Int(a + b)
13     | Int a, Flo b -> Flo((float_of_int a) +. b)
```

Voilà trois fichiers à la suite.

Fichier test.ml

```
1  type nb =  
2    | Int of int  
3    | Flo of float  
4  
5  let add (x:nb)(y:nb):nb =  
6    match x,y with  
7    | Int a, Int b -> Int(a + b)  
8    | Int a, Flo b -> Flo((float_of_int a) +. b)  
9    | Flo a, Int b -> Flo( a +. (float_of_int b))  
10   | Flo a, Flo b -> Flo(a +. b)
```

Fichier test\_bis.ml

```
1  type nb_bis =  
2    | Int of int  
3    | Flo of float  
4
```



```
5  let add (x:nb)(y:nb):nb =  
6    match x,y with  
7    | Int a, Int b -> Int(a + b)  
8    | Int a, Flo b -> Flo((float_of_int a) +. b)  
9    | Flo a, Int b -> Flo( a +. (float_of_int b))  
10   | Flo a, Flo b -> Flo(a +. b)
```

Fichier test\_ter.ml

```
1  type nb_ter =  
2    | Int of int  
3    | Flo of float  
4  
5  let add (x:nb)(y:nb):nb =  
6    match x,y with  
7    | Int a, Int b -> Int(a + b)  
8    | Int a, Flo b -> Flo((float_of_int a) +. b)  
9    | Flo a, Int b -> Flo( a +. (float_of_int b))  
10   | Flo a, Flo b -> Flo(a +. b)
```