

Presentation template

Ivan Trepakov

University

Literate Haskell sample

```
-- >>> fact 5  
-- 120
```

```
fact 0 = 1  
fact n = n * fact (n - 1)
```

Пример слайда с кириллицей

Теорема Пифагора

Основная формулировка содержит алгебраические действия — в прямоугольном треугольнике, длины катетов которого равны a и b , а длина гипотенузы — c , выполнено соотношение:

$$a^2 + b^2 = c^2.$$

Пример программы

```
# Вычисление факториала числа n
def fact(n):
    if (n==1 or n==0):
        return 1
    else:
        return n * fact(n - 1)
```

Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX

Second column (centered)

- Markdown lists
- With beautiful math: $x^n + y^n = z^n$
- And easy **Markdown** styles

Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX
- Beamer allows you to flexibly animate slides with `\uncover<X>` and `\only<X>`

Second column (centered)

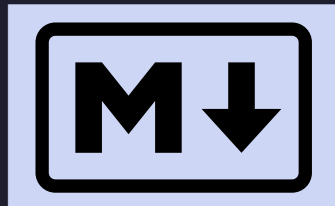
- Markdown lists
- With beautiful math: $x^n + y^n = z^n$
- And easy **Markdown** styles

Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX
- Beamer allows you to flexibly animate slides with `\uncover<X>` and `\only<X>`
- For images it is better to use vector graphics, e.g. in `.svg` which is automatically converted into `.pdf` via `Makefile` magic

Second column (centered)

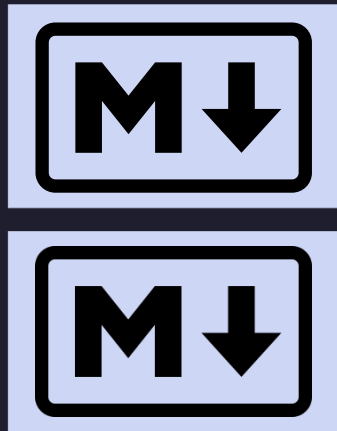


Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX
- Beamer allows you to flexibly animate slides with `\uncover<X>` and `\only<X>`
- For images it is better to use vector graphics, e.g. in `.svg` which is automatically converted into `.pdf` via `Makefile` magic
- You can also use `.png` or `.jpg` but they usually look worse than `.svg/.pdf`

Second column (centered)

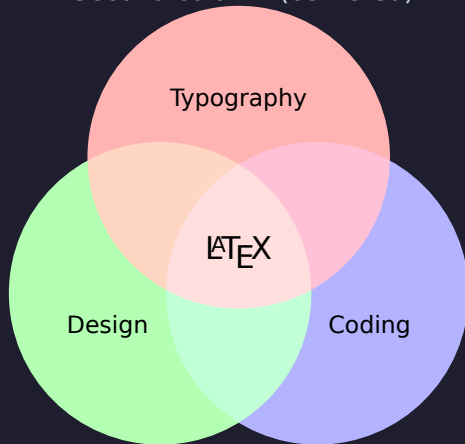


Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX
- Beamer allows you to flexibly animate slides with `\uncover<X>` and `\only<X>`
- For images it is better to use vector graphics, e.g. in `.svg` which is automatically converted into `.pdf` via `Makefile` magic
- You can also use `.png` or `.jpg` but they usually look worse than `.svg/.pdf`
- Or you can dive deep into TikZ

Second column (centered)



Sample slide

First column

- You can use all Markdown features and directly embed \LaTeX
- Beamer allows you to flexibly animate slides with `\uncover<X>` and `\only<X>`
- For images it is better to use vector graphics, e.g. in `.svg` which is automatically converted into `.pdf` via `Makefile` magic
- You can also use `.png` or `.jpg` but they usually look worse than `.svg/.pdf`
- Or you can dive deep into TikZ
- Links can also be embedded as QR codes into presentation with \LaTeX

Second column (centered)



<https://texample.net/tikz/examples/>

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```


Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Incremental code highlighting

Beamer macros

- `\onslide<X>` macro can be used inside of code listings to provide custom animations
- `\setbeamercovered` macro controls how the elements are displayed when they are supposed to be hidden
- In this example `\setbeamercovered{transparent=40}` makes elements dimmed instead of being hidden completely

Code sample

```
# Factorial example
def factorial(n):
    if n < 2:
        return 1
    else:
        return n * factorial(n-1)
```

Summary

- Pandoc = Markdown + \LaTeX
- Please use this template and never open ~~Google Slides~~ PowerPoint ever again

Thank you for your attention