Asignatura
Título de la Práctica

Nombre del autor

30 de noviembre de 2020

1. Introducción

Esta es una introducción de la práctica. Aquí se puede poner de qué trata la práctica, qué es lo que se va a hacer, cómo se va a llevar a cabo y con qué tecnologías.

2. Estructura del documento

Una bondad de utilizar LaTeX como generador de documentos frente a otras herramientas es la posibilidad de estructurar el código del documento completo en distintos archivos para poder trabajar mejor. Esto se realiza mediante un paquete llamado `subfiles`.

La manera más común de utilizar `subfiles` es utilizar la siguiente línea en el documento que incluye el `subfile`:

```
\subfile{Prueba_subfiles.tex}
```

Y el resultado de este comando es el siguiente:

Esta es una prueba de `subfile`

Este archivo `Prueba_subfiles.tex` contiene lo siguiente:

```
\documentclass[../../main.tex]{subfile}

\begin{document}

  \textcolor{green!50!blue!50!}{\textit{Esta es una prueba de \texttt{subfile}}}

\end{document}
```

En cada `subfile` hay que incluir la ruta relativa desde el `subfile` hasta el archivo principal (`main.tex` en esta plantilla) y decir que el documento es de clase `subfile`.

Se hace más estructurado el proyecto LaTeX si se trabaja con carpetas y subcarpetas, todas organizadas según las distintas secciones del documento.

Así, el uso que se recomienda de los `subfiles` es el de un `subfile` por apartado en el documento[1]. De manera que en el archivo padre se tenga algo como lo siguiente:

```
\section{Estructura del documento}
  \label{sec:Estructura del documento}
  \subfile{Estructura_del_documento/Estructura_del_documento.tex}
```

Sin embargo, estas tres líneas de código se han resumido en un comando `\mysection`, que recibe dos parámetros. El equivalente al código anterior sería:

```
\mysection{Estructura del documento}{Estructura_del_documento}
```

Como se puede observar, el primer parámetro es el título que se mostrará en el documento (y el nombre de la etiqueta); y el segundo parámetro es el nombre del archivo que tiene el contenido de la sección (sin extensión).

Al igual que existe `\mysection` en el archivo `packages/titlesconfig.sty`, existen otros comandos como `\mysubsection` y `\mysubsubsection`, los cuales tienen la misma utilidad pero con otros niveles de sección en el documento.

El ejemplo más claro de cómo utilizar `subfiles` y de cómo estructurar el código LaTeX es el presente documento y su código. Analícese detenidamente para mayor comprensión.

3. Ecuaciones

Usar ecuaciones en LaTeX es bastante sencillo. Si se quiere poner una ecuación en línea con el texto, se pone `$\mathrm{e}^{\mathrm{i}\pi} + 1 = 0$`, y quedaría como $\mathrm{e}^{\mathrm{i}\pi} + 1 = 0$.

Si se prefiere poner la ecuación en un nuevo párrafo y centrada, se debe poner la misma expresión pero con dos símbolos $ de apertura y cierre. Es decir:

[1] También es conveniente utilizar un `subfile` cuando se incluya un dibujo con `tikzpicture`, una tabla, una gráfica con `pgfplots`...

```
$$\sum_{n = 1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$
```

El resultado es:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Si se quiere referenciar una ecuación es necesario utilizar el entorno `equation`. Por ejemplo, el siguiente código:

```
\begin{equation}
  \label{eq:matrix_fibonacci}
  \begin{pmatrix}
    F_{n + 1} & F_n \\
    F_n & F_{n - 1}
  \end{pmatrix} = \begin{pmatrix}
    1 & 1 \\
    1 & 0
  \end{pmatrix} ^ n
\end{equation}
```

Genera la siguiente ecuación:

$$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \tag{1}$$

Esta ecuación puede ser referenciada utilizando `\eqref{eq:matrix_fibonacci}`, y queda como (1).

4. Figuras

Para incluir una figura, en esta plantilla de LaTeX se dispone de un comando llamado `\figcaption` que recibe tres parámetros:

1. Nombre del archivo de imagen (puede ser con o sin extensión, aunque se recomienda con extensión).
2. Texto del pie de figura.
3. Tamaño de la imagen (siendo 1 el ancho de la hoja).

Por ejemplo, este código:

```
\figcaption{Lab.png}{Sucesión de Fibonacci.}{0.9}
```

Genera la siguiente figura:

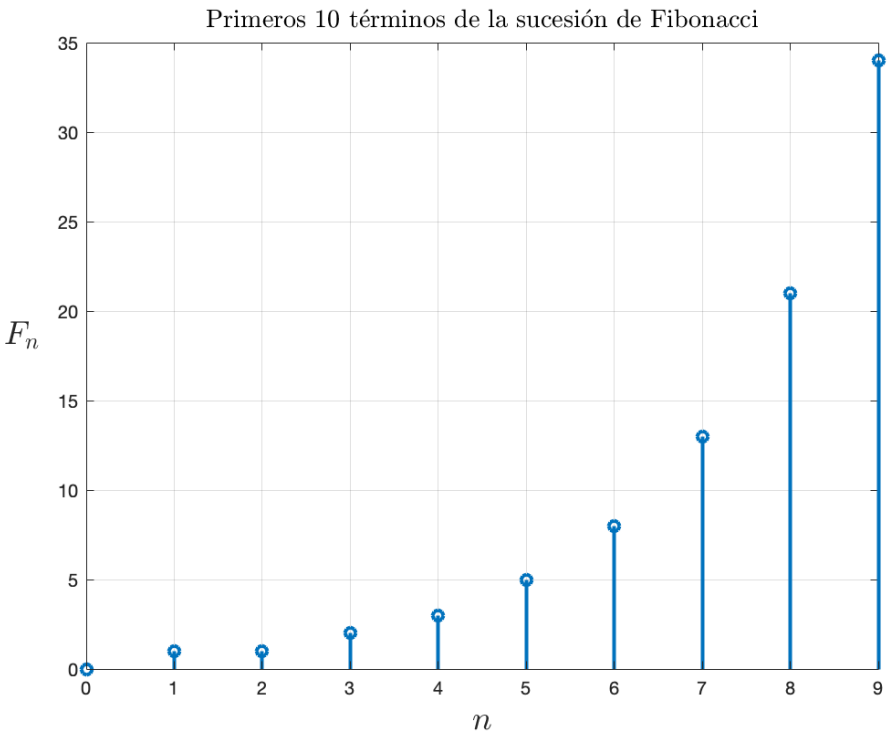


Fig. 1. Sucesión de Fibonacci.

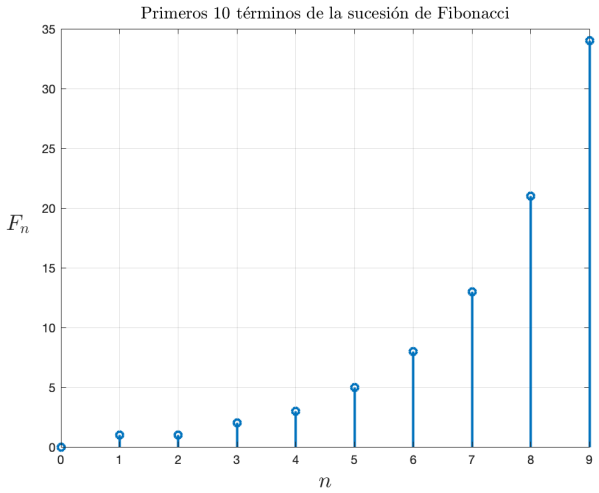
Para referenciar esta figura, basta con utilizar el comando `\figref{Lab.png}`, indicando el nombre del archivo de imagen, y dará lugar al siguiente hipervínculo: Fig. 1. Si se quiere cambiar el nombre del tipo de referencia, se debe cambiar en el archivo `packages/references.sty`.

El formato del pie de foto se puede modificar en el archivo `main.tex`, donde se dice:

```
\usepackage[font=small, labelfont=bf, labelsep=period]{caption}
```

Más información al respecto puede encontrarse en la documentación del paquete `caption`.

Si se quiere incluir una imagen que no se va a referenciar o que no va a tener pie de figura, se puede utilizar el comando `\fignocaption`, que solo necesita dos parámetros (el nombre del archivo y el tamaño de la imagen). De manera que `\figcaption{Lab.png}{0.5}` muestra la siguiente imagen:



Cabe mencionar que la estructura de directorios de la plantilla está pensada para incluir las imágenes en la carpeta `images/`. Si se desea cambiar el nombre de esta carpeta o su ubicación, se deberá cambiar el código `packages/figureconfig.sty`, en la parte en la que se dice:

```
% Ruta al directorio de imágenes
\graphicspath{{./images/}}
```

Y poner lo que sea conveniente.

5. Código fuente

Para incluir código fuente se utiliza el paquete `minted`, el cual tiene soporte para gran cantidad de lenguajes de programación y lenguajes de marcado. El código puede ser incluido directamente en el código LaTeX, de manera que:

```
\begin{minted}[bgcolor=lightgray]{html}
<!doctype html>
<html>
  <head>
    <title>Hello, world!</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
\end{minted}
```

Genera el siguiente código coloreado sintácticamente:

```
<!doctype html>
<html>
  <head>
```

```
    <title>Hello, world!</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

El parámetro `bgcolor=lightgray` indica que el color de fondo sea gris claro. Este color `lightgray` está definido en el archivo `packages/mycommands.sty`.

Al entorno `minted` se le pueden añadir muchos parámetros, entre ellos `linenos` (para que se indique el número de cada linea del código a la izquierda) y `fontsize` (para indicar el tamaño de letra, este tamaño suele no especificarse o ponerse como `fontsize=\footnotesize` o como `fontsize=\scriptsize`).

Una utilidad muy buena de `minted` es la posibilidad de insertar código fuente desde un archivo existente, sin necesidad de copiar el código en el archivo de LaTeX. Además, se puede indicar desde qué línea hasta qué linea se debe incluir. Para ello, en esta plantilla existe el comando `codecaption`, el cual requiere de cuatro parámetros:

1. Ruta al fichero.
2. El lenguaje de programación o marcado utilizado.
3. Texto del pie de código.

A este comando se le pueden insertar muchas opciones, como `bgcolor`, `linenos`, `fontsize`, `firstline` y `lastline`. Por ejemplo, con la siguiente instrucción:

```
\codecaption[
  bgcolor=lightgray,
  firstline=28,
  fontsize=\footnotesize,
  lastline=36
]
{Lab.m}
{\matlab}
{Función para generar la sucesión de Fibonacci.}
```

Se obtiene:

```
function F = fibonacci(n)

    if n < 2
        F = n;
    else
        F = fibonacci(n - 1) + fibonacci(n - 2);
    end

end
```

Código 1. Función para generar la sucesión de Fibonacci.

Al igual que antes, para referenciar este código, se puede utilizar el comando configurado como `\coderef{Lab.m}`, indicando la ruta del archivo, de manera que se obtiene el hipervínculo: Código 1.

De nuevo, en esta plantilla, el código fuente se debe poner en la carpeta `src/`. Si se quiere elegir otra carpeta independiente de la plantilla se ha de modificar el archivo `packages/codeconfig.sty`, donde dice:

```
% Ruta relativa al código fuente
\newcommand*{\sourcedir}{../src/}
```

Es probable que sea necesario poner una ruta absoluta y evitar nombres de carpetas y archivos con espacios o caracteres raros.

Si no se quiere referenciar el código o escribir un pie de código, se puede utilizar el comando `\codenocaption`, el cual requiere los mismos argumentos que el comando anterior, a excepción del texto del pie de código. Por ejemplo:

```
\codenocaption[bgcolor=lightgray, fontsize=\footnotesize]{main.c}{c}
```

Muestra el siguiente código en C:

```c
#include <stdio.h>

int main() {
  printf("Hello, world!\n");

  return 0;
}
```

A la hora de elegir el lenguaje para que `minted` coloree el código, cabe mencionar que no todos funcionan bien al 100 %, puede que los *lexers* no estén actualizados a las últimas tendencias de los lenguajes. Por este motivo, se han añadido algunos *lexers* personalizados en la carpeta `lexers/`. Están escritos en Python y hacen uso de expresiones regulares. Para utilizarlos, basta con poner `\dockerfile` en lugar de `dockerfile`, `\json` en lugar de `json`, `\yaml` en lugar de `yaml` y `\matlab` en lugar de `matlab` (este último comando simplemente cambia el *lexer* de `matlab` por el de

`octave`, porque funciona mejor).

6. Conclusiones

En este apartado se debe poner qué se ha conseguido con la práctica, qué se ha aprendido, qué problemas han surgido y cómo se han solucionado. Se pueden incluir opiniones, observaciones y comentarios relacionados con el desarrollo de la práctica.

Respecto a la plantilla de LaTeX implementada, es bastante básica y trata de minimizar la necesidad de conocer el lenguaje LaTeX para poder generar un documento de calidad con este lenguaje y sin tener que pasar más tiempo investigando en cómo trabajar en LaTeX que en escribir el informe.