Enunciados y programación

Composición de textos científicos

17 de noviembre de 2021

1. Definición y numeración de enunciados

El paquete más usado para dar formato a los enunciados matemáticos y sus pruebas es `amsthm`. Por consiguiente, comenzamos incluyendo en la cabecera

```
\usepackage{amsthm}
```

Según la documentación hay que hacerlo después de que esté cargado el paquete `amsmath`.

Es importante que sepas que utilizando el `\documentclass` con el estilo de una editorial o quizá la plantilla de tu TFG, este paquete podría estar cargado de antemano y los comandos indicados en esta sección podrían tener interacciones indeseadas.

El uso de `amsthm` es muy sencillo. En la cabecera, después de haberlo cargado y antes del `\begin{document}`, escribiremos `\newtheorem` con dos argumentos entre llaves, el primero indica el nombre del entorno que queremos usar y el segundo el nombre que aparecerá en el documento. Por ejemplo, para un artículo en español podemos emplear:

```
\newtheorem{theorem}{Teorema}
```

que crea un entorno llamado `theorem` que mostrará la palabra "Teorema" cuando se invoque a él. Es decir, con

```
\begin{theorem}
 Los poliedros convexos cumplen $V+C=A+2$.
\end{theorem}
```

conseguiremos:

Teorema 1. *Los poliedros convexos cumplen* $V + C = A + 2$.

El nombre del entorno es convencional, si queremos ser originales

```
\newtheorem{un_teorema}{Teorema}
```

tendría el mismo efecto siempre que abramos con `\begin{un_teorema}` los enunciados de los teoremas y los cerremos con `\end{un_teorema}`.

La numeración es automática y consecutiva, así

```
\begin{theorem}
 Si fuera un mapa en un toro, $V+C=A$.
\end{theorem}
```

genera

Teorema 2. *Si fuera un mapa en un toro,* $V + C = A$.

Para un documento largo la numeración anterior es demasiado simple, y, si el documento es muy largo, poco útil. Las razones son las mismas que en la numeración de ecuaciones. Por ejemplo, si uno tuviera que buscar el Teorema 503 en un volumen gigantesco seguramente estaría un buen rato pasando páginas. Por ello, en un libro lo más habitual es que el número de un enunciado incorpore alguna información acerca del lugar donde se ubica. Por ejemplo, los teoremas anteriores podrían numerarse con **1.1** y **1.2** para indicar que están en la primera sección. Recordemos que las secciones en LaTeX se indican mediante `\section{`*título*`}`. Por ejemplo, en este documento la primera sección se creó con

```
\section{Definición y numeración de enunciados}
```

y también existen, en la clase `article`, `\subsection` y `\subsubsection` que actúan de la misma forma para crear subsecciones y lo que corresponde al palabro "subsubsecciones". Dicho sea de paso, el formato `book` que, como su nombre indica, se aplica a libros, incorpora todavía dos divisiones más amplias: `\chapter` y `\part`.

Para que los teoremas, o enunciados en general, incorporen el número de sección, basta añadir `[section]` al final de la definición del enunciado en la cabecera. Esto es:

```
\newtheorem{theorem}{Theorem}[section]
```

Si no hubiera secciones porque no hemos empleado el comando `\section` entonces la numeración sería **0.1** y **0.2** porque el contador de secciones está internamente a cero, lo cual quizá te parezca conveniente para una sección introductoria. Los nombres de otras divisiones se pueden emplear de manera análoga en la numeración de enunciados en lugar de `section` y cada una incluye las numeraciones de las divisiones que las contienen (en formato `book` hasta `chapter`). Con ello, si en un documento como este empleamos

```
\newtheorem{theorem}{Teorema}[subsubsection]
```

antes del número del teorema aparecerá el número de sección, subsección y subsubsección (por ese orden). De esta forma un teorema podría llamarse **Teorema 2.3.5.7** y en formato `book` se transformaría en **Teorema 1.2.3.5.7** si está en el capítulo 1. Por supuesto, rara vez se da tanto detalle sobre la ubicación de un teorema y en formato `article` el argumento `[section]` es más común que cualquier otro.

Con `\newtheorem` podemos definir tantos entornos como deseemos para lemas, corolarios, observaciones o enunciados de nuestra propia cosecha. Si lo hacemos como antes, por ejemplo

```
\newtheorem{lemma}{Lema}
```

o

```
\newtheorem{lemma}{Lema}[section]
```

la numeración será independiente, es decir, habrá Lema 1 (o 1.1) y también Teorema 1 (o 1.1). Algunos textos de matemáticas (una minoría) siguen esta política. Personalmente me parece poco adecuada porque los resultados son difíciles de localizar: cerca del Teorema 20 puede estar el Lema 42. La manera de impedirlo es forzar a que la numeración de los lemas (y otros enunciados) continúen la misma numeración especificada en los teoremas. Para indicar que un enunciado depende de un entorno anteriormente definido, digamos `theorem` en nuestro caso, la estructura es

`\newtheorem{`nombre del entorno`}[theorem]{`nombre en el texto`}`

Con ello el nuevo entorno continuará la numeración de `theorem` y de cualquier entorno ligado a él de la misma forma. Dicho esto, es común ver en la cabecera de un documento LaTeX en español cosas como:

```
\newtheorem{proposition}[theorem]{Proposición}
\newtheorem{lemma}[theorem]{Lema}
\newtheorem{corollary}[theorem]{Corolario}
```

Siempre habiendo definido antes `theorem`. Por ejemplo, con esto el código

```
\begin{theorem}
 Si $a$ y $n$ son coprimos, $n$ divide a $a^{\varphi(n)}-1$.
\end{theorem}
\begin{corollary}
 Si $p$ es primo, $a^p-a$ es siempre múltiplo de $p$.
\end{corollary}
```

daría la numeración consecutiva deseada:

Teorema 3. *Si a y n son coprimos, n divide a $a^{\varphi(n)} - 1$.*

Corolario 4. *Si p es primo, $a^p - a$ es siempre múltiplo de p.*

Hay una variante que tiene algún sentido para observaciones o corolarios, pero que, en mi experiencia, no es demasiado usada en la práctica. Si escribimos `[theorem]` al final, en vez de en medio, entonces la numeración incluirá la del último enunciado de tipo `theorem`. Por ejemplo, si con las líneas anteriores en la cabecera incluimos allí también

```
\newtheorem{remark}{Observación}[theorem]
```

Entonces, salvo el tipo de letra, las líneas

```
\begin{proposition}
 Para $a,b\in\mathbb{R}$ se tiene $ab=ba$.
\end{proposition}

\begin{remark}
 También se cumple para números complejos.
\end{remark}

\begin{remark}
 No se cumple en general para matrices.
\end{remark}
```

producirán:

Proposición 5. *Para $a, b \in \mathbb{R}$ se tiene $ab = ba$.*

Observación 5.1. También se cumple para números complejos.

Observación 5.2. No se cumple en general para matrices.

Hay coherencia en la sintaxis del comando con lo visto anteriormente: cuando poníamos `[section]` se incorporaba al enunciado el número de sección y ahora que ponemos `[theorem]` se incorpora el de teorema, a pesar de que las secciones y teoremas son para nosotros, e internamente para LaTeX, objetos bien distintos.

Seguramente te hayas percatado de que el ejemplo anterior es un poco tramposo porque el tipo de letra de las observaciones es diferente del de la proposición, tanto en el nombre como en el contenido, cuando el comportamiento por defecto es que todos los enunciados tengan el mismo formato. Hay un mínimo control sobre ello empleando:

`\theoremstyle{`*estilo*`}`

donde el *estilo* es `plain` (por defecto), `remark` o `definition`. Es posible crear nuevos estilos, pero no lo veremos aquí[1]. Al incluir un `\theoremstyle`

[1] Un sustituto más poderoso que veremos más adelante es la definición de un nuevo entorno no asociado a `amsthm`.

en la cabecera, los `\newtheorem` definidos a continuación hasta un nuevo `\theoremstyle` se verán afectados por dicho estilo. Así en la cabecera de este documento aparece:

```
\theoremstyle{remark}
\newtheorem{remark}{Observación}[theorem]
```

Como muestran los ejemplos anteriores, `plain` pone el nombre y número en negrita y el texto en cursiva, mientras que `remark` pone el nombre y número en cursiva y el texto en letra normal. Por otro lado, `definition` pone el nombre y número en negrita y el texto en letra normal.

2. Referencias y pruebas

Siendo las matemáticas deductivas, continuamente en nuestros documentos habrá referencias a enunciados anteriores. Las etiquetas de teoremas y otros enunciados se definen con `label`, como las de las ecuaciones. Por ejemplo, si al enunciar nuestra proposición escribimos

```
\begin{proposition}\label{conmut}
 Para $a,b\in\mathbb{R}$ se tiene $ab=ba$.
\end{proposition}
```

entonces con

```
 Por la Proposición~\ref{conmut}
```

se obtiene "Por la Proposición 5".

El entorno para las demostraciones es `proof`. Por ejemplo, la prueba del primer teorema:

Demostración. Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo. □

viene de la fuente

```
\begin{proof}
 Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo.
\end{proof}
```

El nombre "*Demostración*" depende de `babel` y si cambiamos de idioma se modificará consecuentemente. De esta forma, si cambiamos `spanish` por `english` cuando se carga `babel` en la cabecera de la fuente de este documento se obtiene "*Proof*" y si lo cambiamos por `german` se obtiene "*Beweis*".

A veces, la prueba está lejos del teorema y queremos hacer referencia a él. La manera habitual de hacerlo es emplear el entorno `proof` con un parámetro en la forma `{proof}[...]` que sustituye "Demostración" por lo que escribamos entre los corchetes. Por ejemplo, con

```
\begin{proof}[Demostración de la Proposición~\ref{conmut}]
 Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo.
\end{proof}
```

obtenemos

Demostración de la Proposición 5. Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo. □

No hay restricciones sobre el texto que sustituye a "Demostración" y podemos usar `\ref` para referirnos a partes de un texto, como `\section` o `\subsection`, que hayamos etiquetado con `\label`. Por ejemplo, si al introducir esta sección hemos escrito

```
\section{Referencias y pruebas}\label{re_pr}
```

entonces con

```
\begin{proof}[Prueba del resultado de la sección \ref{re_pr}]
 Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo.
\end{proof}
```

obtenemos

Prueba del resultado de la sección 2. Se sigue de la conmutativa porque $\mathbb{R}$ es un cuerpo. □

3. Definición de comandos

Ya habíamos visto algún ejemplo simple de definición de comandos. La estructura básica es:

`\newcommand{`*nombre*`}{`*significado*`}`

Así con `\newcommand{\R}{\mathbb{R}}` conseguíamos abreviar la expresión `\mathbb{R}` por `\R` lo cual será muy conveniente si estamos escribiendo un texto de análisis real. Lo ortodoxo y muy recomendable es poner las definiciones de los comandos en la cabecera para favorecer la legibilidad de la fuente y darles ámbito global, pero no es estrictamente obligatorio.

Alguna vez querremos redefinir un comando presente en LaTeX porque la abreviatura que hemos elegido coincide con un comando que no usamos o dicho comando se emplea internamente con algún propósito y queremos que personalizarlo. Un ejemplo de esto último es `\qedsymbol` que es un comando, normalmente de uso interno, que da el símbolo que indica el final de una demostración. Por defecto está definido como un cuadrado sin rellenar. Para redefinir este u otro comando se utiliza `\renewcommand`. Por ejemplo si quisiéramos ser originales y que en vez del cuadrado apareciera un círculo, emplearíamos:

```
\renewcommand{\qedsymbol}{$\bigcirc$}
```

Para que el cuadrado se rellene una posibilidad es

```
\renewcommand{\qedsymbol}{$\blacksquare$}
```

Dicho sea de paso, como `\rule{...}{...}` da una línea de cierto ancho y alto sustituyendo los puntos suspensivos por longitudes iguales tendremos control sobre el tamaño del cuadrado relleno. Así, sustituyendo `$\blacksquare$` por `\rule{4mm}{3mm}` obtendremos un rectángulo apaisado de 4×3 milímetros.

Incidiendo sobre lo dicho antes, un `\renewcommand{\qedsymbol}` que esté dentro de un entorno `proof` solo actuará sobre el símbolo de la demostración correspondiente, dejando el resto sin cambios.

Este uso de `\newcommand` y `\renewcommand` es equivalente a lo que conseguiríamos con un editor básico reemplazando cadenas de caracteres. Algo más complicado (aunque no fuera del alcance de editores avanzados) es definir o redefinir comandos con argumentos. La estructura es ahora

`\newcommand{`*nombre*`}[`n^o*. argumentos*`]{`*significado*`}`

Los argumentos se indican en el *significado* con `#1`, `#2`, etc.

Por ejemplo, supongamos que voy a escribir muchos ejemplos de integrales racionales entre cero y uno para los alumnos de Cálculo I y para ello me gustaría definir un comando en el que bastase introducir numerador y denominador. Una posibilidad es

```
\newcommand{\intr}[2]{\int_0^1\frac{#1}{#2}\;dx}
```

Con ello,

```
\[
 \intr{x+1}{x^2+2},\qquad \intr{x}{x-2},\qquad
 \intr{1}{x+2}.
\]
```

da lugar a:

$$\intr{x+1}{x^2+2},\qquad \intr{x}{x-2},\qquad \intr{1}{x+2}.$$

Hay todavía otra forma más general de `\newcommand` (y `\renewcommand`) que responde a

`\newcommand{`*nombre*`}[`n^o*. arg.*`][`*opcional*`]{`*significado*`}`

Con ello `#1` tomará el valor de *opcional* a no ser que se especifique otra cosa como un parámetro entre corchetes. Volviendo al ejemplo anterior, imaginemos que casi todas las integrales que quiero escribir tienen numerador $x+1$, pero hay algunas excepciones. Definiríamos

```
\newcommand{\inte}[2][x+1]{\int_0^1\frac{#1}{#2}\;dx}
```

y con

```
\inte{x^2+2},\qquad \inte{x+7},\qquad \inte[x^2+1]{x+1}.
```

conseguiríamos:

$$\int_0^1 \frac{x+1}{x^2+2}\,dx, \qquad \int_0^1 \frac{x+1}{x+7}\,dx, \qquad \int_0^1 \frac{x^2+1}{x+1}\,dx.$$

En el último caso `[x^2+1]` es el argumento opcional que cambia el valor $x+1$ por defecto de `#1`.

4. Definición de entornos

Recordemos que un entorno es algo que está en un bloque

\begin{*nombre*}...\end{*nombre*}

Crear nuevos entornos es más complicado que crear nuevos comandos, pero la idea sobre su definición o redefinición es similar. La plantilla básica es:

\newenvironment{*nombre*}{*código inicial*}{*código final*}

y también hay un `\renewenvironment` con estructura análoga. Su efecto es que se copiará el código inicial, después lo que hayamos escrito dentro del entorno y después el código final. Por ejemplo, si me quiero inventar una nueva forma de matriz que en vez de paréntesis tenga dobles llaves, una posibilidad es definir un nuevo entorno `BBmatrix` de la siguiente forma:

```
\newenvironment{BBmatrix}
{\left\{\left\{\begin{matrix}}
{\end{matrix}\right\}\right\}}
```

Si escribimos por ejemplo (en modo matemáticas)

```
\begin{BBmatrix}
 2 & 1 \\ 3& 4
\end{BBmatrix}
```

el resultado será:

$$\left\{\left\{\begin{matrix} 2 & 1 \\ 3 & 4 \end{matrix}\right\}\right\}$$

Los entornos, igual que los comandos, admiten argumentos y argumentos opcionales. Los argumentos no opcionales, al menos en mi experiencia, no

son tan comunes. Quizá una razón para ello es que, por razones técnicas, el uso de los argumentos está limitado al código inicial[2].

Para ilustrarlo, supongamos que tenemos que elaborar una lista de ejercicios y tareas que incorporan una indicación de la dificultad. Ensayemos diferentes versiones. Para practicar supongamos que queremos el texto del ejercicio o tarea en letra inclinada y el nombre en versalita. Una posibilidad muy simple es:

```
\newenvironment{ej_a}[2]{\textsc{#1}. (Dificultad: #2).\sl}{}
```

Al escribir:

```
\begin{ej_a}{Ejercicio}{0}
 Calcula $1+1$.
\end{ej_a}

\begin{ej_a}{Tarea}{$\star\star\star$}
 Halla una fórmula para $1^2+2^2+\dots+n^2$.
\end{ej_a}
```

obtendremos:

EJERCICIO. (Dificultad: 0). *Calcula* $1+1$.

TAREA. (Dificultad: $\star\star\star$). *Halla una fórmula para* $1^2+2^2+\cdots+n^2$.

Si vamos a escribir muchos ejercicios y ocasionalmente una tarea, convendría definir en su lugar:

```
\newenvironment{ej_b}[2][Ejercicio]{\textsc{#1}.
(Dificultad: #2).\sl}{}
```

y entonces se conseguiría el mismo resultado que antes ahora con

```
\begin{ej_b}{0}
 Calcula $1+1$.
\end{ej_b}

\begin{ej_b}[Tarea]{$\star\star$}
 Halla una fórmula para $1^2+2^2+\dots+n^2$.
\end{ej_b}
```

Antes de pasar a algo más complicado, vamos con un poco de ajuste fino del entorno. Si tecleamos

[2]Hay maneras indirectas de evitar esta restricción; por ejemplo con el paquete `xparse` o definiendo en el código inicial un comando que guarde el argumento y que se use en el código final.

```
Resuelve la siguiente lista de ejercicios fáciles:
\begin{ej_b}{0}
 Calcula $1+1$.
\end{ej_b}

\begin{ej_b}{0}
 Calcula $2+2$.
\end{ej_b}
```

el resultado dista seguramente de lo que teníamos en mente:

Resuelve la siguiente lista de ejercicios fáciles: EJERCICIO. (Dificultad: 0). *Calcula* $1 + 1$.

EJERCICIO. (Dificultad: 0). *Calcula* $2 + 2$.

Y más todavía si no dejamos la línea en blanco entre los entornos. Lo habitual en las listas de ejercicios es que estén separados algo más que el interlineado habitual. Una alternativa es:

```
\newenvironment{ej_c}[2][Ejercicio]{\par\textsc{#1}.
(Dificultad: #2).\sl}{\medskip}
```

El efecto es que incluye un cambio de línea inicial (`\par`) y un salto extra al final. El resultado sería:

Resuelve la siguiente lista de ejercicios fáciles:

EJERCICIO. (Dificultad: 0). *Calcula* $1 + 1$.

EJERCICIO. (Dificultad: 0). *Calcula* $2 + 2$.

Con un `\medskip` tras de `\par` también separaríamos de la línea inicial.

Ahora vamos a una situación más realista en la que deseamos numerar automáticamente los ejercicios. Con esto, estamos cerca de lo que hacíamos con los enunciados usando `amsthm`, pero ahora tenemos más libertad permitiendo argumentos. Los enunciados que hemos visto antes son un caso particular de la definición de entornos.

Sin entrar en muchos detalles, aquí están los comandos más relevantes:

Plantilla	**Efecto**
`\newcounter{`*nombre*`}`	Crea el contador *nombre*
`\the`*nombre*	Muestra el valor de *nombre*
`\refstepcounter{`*nombre*`}`	Incrementa *nombre*
`\setcounter{`*nombre*`}{`*valor*`}`	Asigna *valor* a *nombre*

Los contadores por defecto tienen asignado inicialmente el valor cero y entonces el último comando solo se emplea si queremos romper la numeración natural.

En nuestro caso, creamos un contador `ejer` que se incremente al llamar al entorno y se muestre al lado del primer argumento. Para ello incluimos en la cabecera:

```
\newcounter{ejer}
\newenvironment{ej_d}[2][Ejercicio]{\refstepcounter{ejer}%
\par\textsc{#1 \theejer}. %
(Dificultad: #2).\sl}{\medskip}
```

Seguro que te estás preguntando a qué vienen los `%` al final de las líneas. No son necesarios, podríamos haber escrito todo en una línea como antes, pero si miras fuentes de usuarios avanzados verás esto muy a menudo. El propósito es asegurarse de que no se cuela ningún espacio de fin de línea que no vemos. En un editor si veo `\theejer}.` no estoy seguro de si hay un salto de línea justo tras el punto o hay un espacio (que es lo que quiero).

Con o sin `%` el resultado de usar `ej_d` como antes es:

Ejercicio 1. (Dificultad: 0). *Calcula* $1 + 1$.

Ejercicio 2. (Dificultad: 0). *Calcula* $2 + 2$.

Las divisiones de un documento, como `section` y `subsection` (también `page`), tienen sus propios contadores. Imaginemos que quisiéramos retocar el entorno anterior para que mostrase también el número de sección. Lo más lógico sería emplear:

```
\newenvironment{ej_e}[2][Ejercicio]{\refstepcounter{ejer}%
\par\textsc{#1 \thesection.\theejer}. %
(Dificultad: #2).\sl}{\medskip}
```

El resultado con ello sería:

Ejercicio 4.3. (Dificultad: 0). *Calcula* $1 + 1$.

Ejercicio 4.4. (Dificultad: 0). *Calcula* $2 + 2$.

El comando `\newcounter` admite un parámetro entre corchetes que es otro contador de manera que *nombre* se pone a cero cuando ese segundo contador varía. Esto que suena tan raro es para permitir cosas como `\newcounter{ejer}[section]` que en el ejemplo anterior haría que la cuenta se reiniciase cuando pasamos a otra sección. En otro caso, el primer ejercicio en la siguiente sección aparecería numerado con un segundo número diferente de 1.

Un apunte final es que si buscas en la documentación verás que hay varias maneras de incrementar un contador, la elegida aquí, `\refstepcounter` debe el "`ref`" de su nombre a que es posible utilizar `\label` en los entornos definidos para que `\ref` refleje su valor.