Manuel de la classe Prepamath v. 2.28b du 2021/05/06

Table des matières

Avant-propos

Classe prepamath

La maquette de la collection Prepamath a été implémentée dans une classe LaTeX. Il s'agit d'un fichier nommé `prepamath.cls` *que vous n'êtes autorisé ni à modifier, ni à diffuser*.

L'auteur saisissant en LaTeX n'aura donc pas à se préoccuper outre mesure de la mise en forme de son ouvrage : il lui suffira d'utiliser les commandes et environnements fournis par la classe. En pratique, saisir au début du document :

```
\documentclass[<options>]{prepamath}
```

faire ensuite preuve de bon sens…

Bien entendu l'usage de commandes LaTeX qui influeraient sur la maquette doit à tout prix être évité. Parmi ces commandes proscrites, on trouve :

- `\enlargethispage`[1] ;
- les commandes de sauts verticaux (`\bigskip`, `\medskip`, etc.) ;
- les changements intempestifs de graisse, taille, etc.
- les puces fantaisistes ;
- etc.

Installation

L'auteur de la classe a pris soin de rendre l'installation la plus simple possible. Pour un auteur, avant la version 2.28, seul le fichier `prepamath.cls` était nécessaire. Depuis la version 2.28 il faut disposer des cinq fichiers graphiques :

- `PictoAudio.eps` ;
- `PictoNathanLive.eps` ;
- `PictoProg.eps` ;
- `PictoVideo.eps` ;
- `PictoWeb.eps`

Il est possible d'installer ces fichier dans le répertoire où se situe le fichier source principal devant être compilé mais on peut également les installer dans un arborescence `texmf` de sa machine. L'emplacement exact et les procédures post-installation dépendent de la distribution TeX installée mais en notant `[TEXMF]`

1. Bien entendu, autant l'auteur se doit de ne jamais utiliser ce type de commande, autant le metteur en page pourra les employer lorsque le besoin s'en fera sentir. Il en va de même de `\pagebreak`, `\newline`, etc. Ces commandes seront revues lors du chapitre consacré à la mise en page.

la racine de l'installation, le fichier de classe et les fichiers graphiques devraient logiquement se trouver dans le répertoire :

`[TEXMF]/tex/latex/prepamath/`

La classe a été initialement conçue sous un système TEX datant de 2007, la version 2.0 sous un système TEX datant de 2014 et la version 2.28 datant de 2021. En théorie, la compilation devrait fonctionner sur toutes les distributions LATEX mais plus cette dernière est ancienne, plus on prend un risque d'incompatibilité.

Un fichier `prepamath.tar.gz` est fourni aux auteurs. Il faudra faire attention au fait que ce fichier n'est pas accompagné des fontes commerciales nécessaires à une version définitive du document. Si vous ne possédez pas ces fontes commerciales il faudra compiler sans l'option de classe `realfonts` (voir chapitre 2, section 1.4 page 10).

Si la fonte pour l'euro (fonte d'Adobe) n'était pas installée sur votre système, il faudra l'activer avec la procédure décrite au chapitre 3 section 1 page 43.

Dialogue avec l'auteur de la classe

Cette classe en est à sa version 2.28b. Elle évolue en fonction des remarques des utilisateurs. Voici les possibilités de contact :

Jean-Côme Charpentier
68bis Avenue de la Côte de Beauté
17570 SAINT AUGUSTIN
05 46 06 22 37
06 37 25 17 60
`Jean-Come.Charpentier@wanadoo.fr`

Chapitre 1

Packages

Plan du chapitre

1 Packages de la classe

La classe `prepamath` déclare un certain nombre de packages. Il est donc inutile (voire fautif) de les déclarer à nouveau avec un `\usepackage`. Voici les packages appelés directement ou indirectement par la classe.

Package	date	Version
amsbsy.sty	1999/11/29	v1.2d
amsfonts.sty	2013/01/14	v3.01
amsgen.sty	1999/11/30	v2.0
amsmath.sty	2018/12/01	v2.17b
amsopn.sty	2016/03/08	v2.02
amssymb.sty	2013/01/14	v3.01
amstext.sty	2000/06/29	v2.01
array.sty	2018/12/30	v2.4k
babel.sty	2019/06/03	v3.32
calc.sty	2017/05/25	v4.3
ccaption.sty	2011/08/07	v3.2c
cellspace.sty	2019/03/11	v1.8.1
color.cfg	2016/01/02	v1.6
colortbl.sty	2018/12/12	v1.0d
dvips.def	2017/06/20	v3.1d
expl3.sty	2019-07-01	
expl3-code.tex	2019-07-01	
fancyhdr.sty	2019/01/31	v3.10
fancyvrb.sty	2019/01/15	
fontenc.sty	2005/09/27	v1.99g
frenchb.ldf	2019/03/30	v3.5e
geometry.sty	2018/04/16	v5.8
graphics.sty	2017/06/25	v1.2c
graphicx.sty	2017/06/01	v1.1a
helvet	2005/04/12	v9.2a
ifluatex.sty	2016/05/16	v1.4
ifpdf.sty	2018/09/07	v3.3

COURS

Package	date	Version
iftex.sty	2013/04/04	v0.2
ifthen.sty	2014/09/29	v1.1c
ifvtex.sty	2016/05/16	v1.6
ifxetex.sty	2010/09/12	v0.6
infwarerr.sty	2016/05/16	v1.4
inputenc.sty	2018/08/11	v1.3c
l3keys2e.sty	2019-05-28	
lstlang1.sty	2019/02/27	v1.8b
lstmisc.sty	2019/02/27	v1.8b
listings.cfg	2019/02/27	v1.8b
listings.sty	2019/02/27	v1.8b
listingsutf8.sty	2016/05/16	v1.3
ltxcmds.sty	2016/05/16	v1.23
mathptmx	2005/04/12	v9.2a
mathtime.sty	1999/03/29	v1.1
microtype.sty	2019/02/28	v2.7b
microtype-pdftex.def	2019/02/28	v2.7b
microtype.cfg	2019/02/28	v2.7b
multicol.sty	2018/12/27	v1.8v
multido.sty	2004/05/18	
multido.tex	2010/05/14	v1.42
numprint.sty	2012/08/20	v1.39
pdfescape.sty	2016/05/16	v1.14
pdftexcmds.sty	2018/09/10	v0.29
prepamath.cls	2021/05/06	v2.28b
pst-3d.sty	2009/07/28	
pst-3d.tex	2018/12/22	v1.01
pst-all.sty	2008/01/01	
pst-calculate.sty	2019/01/24	v0.02
pst-coil.sty	2010/02/01	
pst-coil.tex	2015/05/13	v1.07
pst-eps.sty	2005/05/20	
pst-eps.tex	2006/11/04	v1.00
pst-fill.sty	2005/09/12	
pst-fill.tex	2007/03/10	v1.01
pst-fp.tex	2019/05/11	v2.97
pst-grad.sty	2004/07/15	
pst-grad.tex	2006/11/27	v1.06
pst-node.sty	2010/04/22	
pst-node.tex	2018/08/31	v1.91
pst-plot.sty	2010/04/22	
pst-plot.tex	2018/08/31	v1.91
pst-text.sty	2018/12/28	

Package	date	Version
pst-text.tex	2018/12/22	v1.01
pst-tree.sty	2009/01/25	
pst-tree.tex	2017/02/18	v1.13
pst-xkey.tex	2005/11/25	v1.6
pstricks.sty	2018/12/21	v0.69
pstricks.tex	2019/05/11	v2.97
pstricks-add.sty	2018/02/04	v0.16
pstricks-add.tex	2007/03/10	v1.01
scalefnt.sty	1997/09/28	
siunitx.sty	2018/05/17	v2.7s
shellesc.sty	2016/06/07	v0.02a
stringenc.sty	2016/05/16	v1.11
0 t1enc.def	2018/08/11	v2.0j
tabularx.sty	2016/02/03	v2.11b
textcomp.sty	2018/08/11	v2.0j
translator.sty	2019-05-31	v1.12a
trig.sty	2016/01/03	v1.10
ts1enc.dfu	2018/10/05	v1.2f
xcolor.sty	2016/05/11	v2.12
xkeyval.sty	2016/05/11	v2.7a
xkeyval.tex	2006/11/18	v2.7a
xparse.sty	2019-05-28	
xspace.sty	2014/10/28	v1.13
xstring.sty	2019/02/06	v1.83

Cette liste indique par conséquent un grand nombre de commandes proposées automatiquement : toutes les commandes de ces packages. On se reportera, le cas échéant, aux documentations correspondantes.

Le tableau ci-dessus donne les versions et dates des packages testés lors de l'élaboration de cette documentation. Il est tout à fait possible que la classe puisse fonctionner avec des versions plus anciennes de ces packages mais il faut bien être conscient que plus l'écart de version sera important, plus le risque d'erreur sera grand.

2 Packages de l'utilisateur

Rien n'interdit de charger d'autres packages (avec la commande `\usepackage`) pour ses propres besoin mais il ne faut pas utiliser de packages qui modifient la mise en page du document, soit de façon globale, soit de façon ponctuelle (pour modifier l'aspect des listes, des en-têtes de page, etc.). Les packages permettant

d'introduire des éléments de détails typographiques seront sans doute toujours permis. Par exemple, le package `slashbox` qui permet d'avoir des cellules de tableau partagées en diagonale ne pose aucun problème. Un exemple ultérieur montre également l'utilisation du package `mathrsfs` pour obtenir des anglaises rondes dans les formules de mathématiques. Pour les manuels de chimie, l'utilisation de `chemfig`, par exemple, ne pose pas de problème.

Il est possible que l'appel de certains packages entraînent des incompatibilités avec la classe. La meilleure façon de procéder est de contacter l'auteur. Il ne faut surtout pas essayer de modifier la classe pour permettre l'utilisation de ce package !

De façon générale, les auteurs ont l'obligation (plus ou moins stricte) de se conformer à la maquette de la collection, donc aux spécifications de la classe. Même si l'utilisation d'un package est techniquement possible, il vaut mieux en informer l'auteur de la classe.

De la même façon, les fontes utilisées pour les ouvrages de la collection sont indiquées dans la classe et il n'est pas permis d'en utiliser d'autres. Dans la version 2.28b (2021/05/06) de la classe, la fonte romaine est Times, les fontes sans empattement sont Helvetica (normal et condensé), Block (normal et condensé) et Din (normal et condensé) et la fonte à chasse fixe est Computer Modern (cmtt). Les fontes Block, Din et Helvetica condensé sont des fontes commerciales ; si ces fontes ne sont pas disponibles, ce sera la fonte Helvetica appelée par le package `helvet` qui sera utilisée. Le fait de composer son ouvrage avec les fontes libres ou les fontes commerciales va entraîner des changements sensibles. En particulier, cela peut influer sur les coupures de lignes de texte écrits avec ces fontes (par exemple les titres de section). Tout cela pour dire que les commandes du type `\enlargethispage`, `\vspace`, `\pagebreak[...]`, voire encore plus directif avec `\newpage` dans le but d'améliorer la mise en page ne sont pas d'une grande utilité si elles ne sont pas utilisées lors d'une composition avec les fontes commerciales. Elles sont même plutôt nuisibles.

1 QCM Permissions et interdictions

1 min — Corrigé p. 6

Dans un document prepamath, je peux :

a utiliser n'importe quel package b utiliser du Comic Sans pour le texte
c lire ce manuel

2 Packages interdits

★★★ — 5 min — Corrigé p. 6

Les auteurs réunis

Donner la liste des packages qui posent problème avec la classe `prepamath`.

CORRIGÉS

1 QCM Permissions et interdictions

Énoncé p. 5

Les choix [a] et [b] sont absolument proscrits (relisez le chapitre). Le choix [c] est évident : si vous regardez cet exercice, c'est que vous lisez le manuel (c'est bien).

2 Packages interdits

Énoncé p. 5

Les auteurs réunis

Cette liste n'existe pas encore et ne sera jamais complète. On notera que certains packages seront refusés parce qu'ils cassent la maquette et non pas pour une incompatibilité technique avec la classe.

Par exemple, les mises en page de type théorème sont prises en charge par la classe donc le package `amsthm` pose un problème. Même si ce dernier ne posera aucun problème de compilation (je suppose), il est *a priori* interdit à un auteur de changer la maquette de présentation de la sorte. Au minimum, il faudra en parler à l'éditeur au préalable… préparez bien votre argumentaire !

Chapitre 2

Utilisation de la classe

Plan du chapitre

1 Options de classe

Dans cette version 2.28b, la classe `prepamath` n'offre pas une grande variété d'options. On a uniquement cinq catégories d'options pour gérer :

1 Le codage d'entrée ;
2 L'affichage des repères photographiques ;
3 L'utilisation de `pstricks` ;
4 Les fontes ;
5 Les liens hypertextes.

1.1 Codage d'entrée

Lorsqu'on utilise un éditeur de texte pour taper son document, on le fait avec un certain codage d'entrée. Celui-ci peut un peu dépendre du système d'exploitation mais en dernier lieu, c'est l'éditeur de texte qui dicte sa loi.

Sous Windows, on trouve assez fréquemment le codage ANSI Windows. Si votre éditeur utilise ce codage, il faut indiquer l'option `ansinew` comme paramètre optionnel de la commande `\documentclass`. En d'autres termes, la première ligne de code de votre document doit être :

```
\documentclass[ansinew]{prepamath}
```

Sous les anciens systèmes Mac (OS 9 et antérieurs), le codage était assez souvent quelque chose de spécifique à Apple. L'option de classe à indiquer est `applemac`.

Sous Linux, le codage (en France) a été très longtemps soit Latin1, soit Latin9. On choisira les options `latin1` ou `latin9` selon le cas.

Pour ceux qui n'utilisent que les caractères ASCII, c'est-à-dire que les caractères de code inférieur à 127, c'est-à-dire uniquement les caractères qu'on peut rencontrer dans un texte anglais basique – en particulier aucun caractère accentué – il existe l'option `ascii`.

Certains auteurs sont adeptes de ce mode de fonctionnement mais pour ceux qui ne connaissent pas, je déconseille l'utilisation du codage `ascii` : tous les caractères accentués se saisissent « à la TEX ». Pour faire comprendre la chose, voici un exemple où la saisie de :

```
D\‘es no\"el o\‘u un z\’ephir ha\"i me v\^et de
gla\c cons w\"urmiens
```

donne le résultat :

Dès noël où un zéphir haï me vêt de glaçons würmiens

Le codage qui a le vent en poupe depuis plusieurs années et qui – tout le monde l'espère – finira par détrôner tous les autres est le codage Unicode. Plus particulièrement la façon UTF-8 de coder en utilisant Unicode. Pour cela, la classe `prepamath` accepte l'option `utf8`.

⚠ ATTENTION

Le codage `utf8` est le codage par défaut, c'est-à-dire que si on n'indique rien, c'est le codage UTF-8 qui sera utilisé.

Pour ceux qui commence à s'affoler devant toutes ces informations techniques, je propose un petit test. Copier ce qui suit dans un fichier `test.tex`. Attention : pas de copier-coller, il faut entrer le texte directement avec votre clavier en utilisant votre éditeur de texte.

```
\documentclass[<codage>]{prepamath}
\title{Essai de codage}
\author{<votre nom>}

\begin{document}
Dès Noël, où un zéphyr haï me vêt de glaçons würmiens,
je dîne d'exquis rôtis de bœuf au kir, à l'aÿ d'âge mûr,
et cætera.
\end{document}
```

Le `<codage>` prendra une des valeurs suivantes :

- `ansinew`
- `applemac`
- `ascii`
- `latin1`
- `latin9`

- `utf8`

Il suffit alors d'observer – éventuellement attentivement – le texte produit par la compilation pour savoir quel est le codage effectivement utilisé par votre éditeur de texte.

Dans le cas, assez extraordinaire, où aucun de ces codages ne serait utilisé par l'éditeur dont vous vous servez, adressez-vous à l'auteur de la classe pour que l'ajout de ce point exotique puisse se faire dans de bonnes conditions !

⚠ ATTENTION

Lorsqu'on a choisi un codage, il faut que *tous* les fichiers sources utilisés partagent ce même codage.

1.2 Repères photographiques

Ce manuel a été créé avec ses repères photographiques. Il s'agit de différentes lignes et cercles qui permettent d'assurer un bon alignement des différentes pages avant découpe et reliure. En l'occurrence, il s'agit de quatre repères situés au quatre coins de ce qui sera la page physique de l'ouvrage final.

Les repères photographiques n'ont d'intérêt que pour l'imprimeur. Les auteurs ont raisonnablement le droit de s'en moquer. L'option permettant d'obtenir ces repères est `crop` et celle permettant de ne pas en avoir est `nocrop`. Si on n'indique rien, c'est l'option `nocrop` qui sera utilisée.

Lorsqu'on demande les repères photographiques, on a également, sur chaque page (en dehors de la zone physique finale), le titre de l'ouvrage, sa version, la date, la page logique et la page physique précédée d'un signe dièse. La page logique est le nombre effectivement indiqué au niveau du foliotage et le numéro physique correspond au nombre de pages total depuis le début du document.

1.3 Packages PSTricks

Un point important et qui diffère notablement des toutes premières versions de la classe `prepamath` est l'utilisation de la bibliothèque `pstricks`. Cette bibliothèque permet de construire des graphiques en les codant directement au niveau du source du document.

Le fait de coder les graphiques dans la classe fait qu'il n'y a plus besoin de tous les fichiers graphiques qui étaient nécessaires avec l'ancienne classe mais cela interdit la compilation par `pdflatex`. L'option de classe `nopstricks` permet de ne plus utiliser `pstricks`, ce qui va permettre de compiler le document avec `pdflatex` mais, en contrepartie, tout ce qui est graphique va disparaître (en-tête, fond coloré de multiples éléments, . . .). La présence des fichiers de la bibliothèque `pstricks` est obligatoire, même si l'option de classe `nopstricks` a été précisée.

S'il faut les fichiers de la bibliothèque `pstricks`, on peut se demander quel intérêt à compiler sous `pdflatex` plutôt que sous `latex`. Il y a en fait deux raisons qui peuvent pousser à utiliser `pdflatex` :

- on ne sait pas faire autrement ;
- on doit inclure des fichiers graphiques `.png`, `.jpg` ou `.pdf`.

Pour le premier point, essayez quand même de vous renseignez pour savoir comment passer par la chaîne de production `dvi`→`ps`→`pdf`.

Pour le second point, si vous ne savez pas comment convertir (proprement) un fichier graphique en PostScript, gardez `pdflatex` avec l'option `nopstricks`. En revanche, comme la compilation finale se fera avec `latex`, il faudra donner l'ensemble des fichiers graphiques à la personne en charge de cette compilation finale pour qu'elle fasse la conversion en PostScript.

À propos des formats des fichiers graphiques, il convient de faire quelques mises au point importantes.

1 Les formats `png` et `jpg` sont des formats bitmap, c'est-à-dire que le fichier mémorise la couleur des points de façon individuelle (avec en plus des outils de compression pour alléger la taille de ces fichiers). Sauf si l'image est très petite ou si le fichier est de taille très importante, on court au devant de grandes désillusions : une image très jolie à l'écran sera absolument horrible à l'impression. À chaque fois que c'est possible, il faut absolument privilégier les formats vectoriels (`svg`, `pdf` et `ps` sont des formats vectoriels… normalement).

2 Les outils de conversion graphique sont plus ou moins performants. En particulier, les deux poids lourds du domaine – `photoshop` pour Windows et `gimp` sur toutes les plates-formes – ne conviennent absolument pas pour faire des conversions car ils commencent par transformer l'image d'entrée en format bitmap interne… et après c'est fini, on ne peut plus revenir en arrière.

1.4 Fontes

Par défaut, un document de classe `prepamath` utilisera des fontes disponibles sur n'importe quelle distribution TEX. En l'occurrence la fonte Times par l'intermédiaire du package `mathptmx`, la fonte Helvetica par l'intermédiaire du package `helvet` et de la fonte Computer Modern cmtt (fonte à chasse fixe).

Pour que le document utilise les fontes commerciales, il faut préciser l'option `realfonts`. Il faut évidemment avoir ces fontes ! Cela signifie les fichiers `.pfb`, `.tfm`, `.vf`, `.fd` et `.map` aux bons endroits. On notera que ces fichiers ont été créés exprès pour cette classe [2] et qu'on ne trouvera aucun fichier « prêt à l'emploi » sur Internet.

2. Remerciements à Sébastien Mengin et Michel Bovani.

1.5 Liens hypertextes

Tout ce qui tourne autour des liens avec le monde externe sera développé à la section 8 du chapitre2 (page 33).

En ce qui concerne les liens, il n'y a qu'à ajouter l'option de classe : il n'y a rien à modifier dans le document lui-même : les éléments qui peuvent être sujets à un lien hypertexte le seront de façon automatiques (liens entre exercices et corrigés, tables des matières générale et de chapitres, index, etc.).

Évidemment, il n'y a strictement aucun intérêt à composer un document avec l'option `hyperref` si ce document est destiné à être imprimé.

2 Préambule

Le préambule d'un document est tout ce qui se trouve entre le `\documentclass` et le `\begin{document}`. Comme dans n'importe quel document on peut demander l'utilisation de packages supplémentaires avec des `\usepackage`. On a suffisamment mis en garde précédemment pour ne pas y revenir maintenant.

Le préambule est également le lieu où l'on peut définir ses propres commandes. Ce n'est pas forcément le mieux (d'habitude, il vaut mieux se faire son propre package) mais pour des besoins ponctuels, même les puristes ne vous en voudront pas trop. Outre le fait de pouvoir économiser de la frappe en jouant un rôle de raccourci, ces commandes personnelles sont aussi la seule façon de distinguer fond et forme. Plus la distinction sera nette, plus le travail final sera facile… et le travail final sur un document est une tâche très longue. Le squelette de document fourni indique l'inclusion d'un fichier `commandes.tex` qui permet de rassembler les commandes personnelles nécessaires à l'ouvrage traité. Il est préférable d'utiliser ce fichier plutôt que de placer ces commandes directement dans le préambule.

Dans le préambule des documents `prepamath`, il y a deux renseignements qui sont obligatoires. Ne pas les donner conduira à une erreur de compilation. Ces renseignements sont :

- le titre de l'ouvrage ;
- le ou les auteurs de l'ouvrage.

On peut aussi donner d'autres renseignements sur l'ouvrage :

- la date (par défaut la date courante) ;
- la version (par défaut 1.0) ;
- l'ISBN (par défaut X-XXXX-XXXX-X).

On donne ces renseignements en appelant respectivement les macros suivantes (chacune d'entre elle prenant le renseignement proprement dit en argument) :

- `\title`
- `\author`
- `\date`
- `\version`
- `\ISBN`

Par exemple, le préambule complet de ce manuel est :

```
\input{commandes}

\title{Manuel de la classe Prepamath}
\author{Jean-Côme Charpentier}
\version{0.9}
```

3 Parties du document

3.1 Prolégomènes et compagnie

Comme avec les classes standards `book` et `report`, on dispose des commandes `\frontmatter`, `\mainmatter` et `\backmatter` pour indiquer respectivement la partie initiale, de la partie principale et de la partie finale d'un document.

Techniquement, dans la partie initiale, les chapitres ne seront pas numérotés, les en-têtes seront très différents de ceux de la partie principale et le foliotage se fera en chiffre romain minuscule. Traditionnellement, cette partie se compose de l'avant-propos et de la table des matières (composée grâce à la commande `\tableofcontents`). Les tables des matières de début de chapitre sont composées de façon automatique : aucune commande ne doit être saisie à ce niveau.

Dans la partie principale, contrairement à ce qui se passait pour la partie initiale, les commandes `\subsection` et inférieures sont autorisées. On a le droit à six niveaux des classes standards, à savoir :

- `\part`
- `\chapter`
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`

La commande `\part` ne devrait être utilisée que lorsque le manuel propose des thèmes rassemblant plusieurs chapitres. Depuis la version 2.2, la classe permet d'utiliser également des thèmes avec la commande `\thema` (voir un peu plus loin).

Enfin, le foliotage se fera en numérotation arabe.

Pour la partie finale, rien n'est encore fixé et on préconise de ne pas s'en servir. De même, la commande `\subparagraph` existe dans la classe mais c'est pour l'instant hors maquette donc il faudrait ne pas l'utiliser ou bien le préciser clairement auprès de l'éditeur.

3.2 Chapitres

Un livre `prepamath` est découpé en chapitres. Pour débuter un chapitre on utilise la commande `\chapter` tout comme dans les classes standards mais sa syntaxe est différente. La commande demande un argument obligatoire : le titre du chapitre. Elle accepte également un argument optionnel qui permet de gérer les onglets d'un chapitre.

⚠ ATTENTION

Les commandes de section de la classe `prepamath` n'ont pas de forme étoilée. Cela n'aurait pas de sens. Les chapitres non numérotés sont ceux de la partie `\frontmatter`.

Un chapitre peut proposer un ou plusieurs thèmes. Ces thèmes se spécifient avec la commande `\thema`. Pour spécifier plusieurs thèmes, il suffit de les indiquer en les séparant avec des virgules. La présence d'une virgule indique à la classe qu'il y a plusieurs thèmes et donc d'écrire « Thèmes » au pluriel. Si par hasard un thème contient une virgule dans son titre et qu'on ne veut pas de ce pluriel, il suffit de « protéger » cette virgule en la plaçant entre accolade. Par exemple :

```
\thema{L'eau{,} l'air{,} la vie}
```

Normalement, la commande `\thema` doit suivre immédiatement la commande `\chapter` (sinon, le ou les thèmes appartiendront à un onglet plutôt qu'à un chapitre).

Les chapitres peuvent être divisés en sections, sous-sections, etc. grâce aux commandes de la liste page précédente. Dans la partie principale du document, ils sont également divisés en onglets. Les titres des onglets sont repris dans les en-têtes des pages paires et dans des onglets sur le bord extérieur des pages impaires. Les onglets sont une structure totalement indépendante des autres commandes de sectionnement. Pour passer à l'onglet suivant, il faut utiliser la commande `\thumb` ou son équivalent `\onglet`.

Par défaut, les chapitres acceptent trois onglets nommés « Cours », « Énoncés » et « Corrigés ». En réalité, cette valeur par défaut est donnée par la commande `\setthumb` avec les différents titres séparés par des virgules. On indique également la couleur de base de l'onglet avec une lettre (R, B, Y, O et G pour respectivement les couleurs rouge, bleu, jaune, orange et vert). Cette couleur sera celle utilisée, entre autres, pour les en-têtes, les pieds de page et les onglets. La couleur sera indiquée avant le titre de l'onglet et en sera séparée par une barre oblique. Si un titre est terminé par une étoile, les pages de cet onglet auront un fond coloré. C'est le cas par défaut de l'onglet « Corrigés ». Ainsi, la classe `prepamath` exécute la commande suivante :

```
\setthumb{R/Cours,O/Interros,G/Corrig\'es*}
```

En utilisant cette commande, on peut modifier les onglets de tout le document. On peut également modifier les onglets d'un seul chapitre en utilisant le paramètre

optionnel de la commande `\chapter`. Par exemple, le chapitre en cours a été introduit avec l'instruction :

```
\chapter[R/Généralités,O/Détails,G/Typographie]
        {Utilisation de la classe}
```

Encore une fois, changer le nom des onglets ou les couleurs revient à modifier la maquette donc il faut prévenir l'éditeur et avoir une bonne raison de le faire !

3.3 Exercices

La classe prévoit trois types d'exercice [3] :

- l'exercice-type (ou les exercices type), d'habitude en ouverture de chapitre ;
- les QCM ;
- les exercices qui, avec les QCM, sont plutôt réservés à l'onglet « Énoncés » mais rien n'est figé dans la classe à ce niveau là. En revanche, pour l'instant, la maquette exige que ce soit bien au niveau de l'onglet « Énoncés » que l'on retrouve ces types d'exercices.

Ces trois structures sont des environnements. On a ainsi, dans l'ordre, les environnements `exointro`, `qcm` et `exo`.

L'environnement le plus simple est `exointro`. Il ne demande qu'un seul argument qui est le nom du lycée de provenance de cet exercice. Le corps de l'environnement sera l'énoncé de l'exercice et il n'y a *a priori* aucune restriction sur ce qu'il est possible d'y faire.

La correction de l'exercice-type, traditionnellement à la fin de l'onglet « Cours », est saisie à l'intérieur d'un environnement `solintro`. Cet environnement ne demande aucun argument.

Il peut y avoir plusieurs exercices type. Dans ce cas, il faudra autant d'environnement `exointro` que d'exercices et, bien sûr, autant d'environnement `solintro` qu'il y a eu d'exercices.

L'environnement `qcm` demande deux arguments : le titre de l'exercice et le temps supposé de composition. Les choix du QCM seront saisis avec la commande `\choix` qui affiche le numéro de question dans un cadre.

Chaque nouvel exercice remet à zéro le compteur de choix. Si on veut mettre plusieurs QCM indépendants dans un même exercice, il va falloir réinitialiser le compteur de façon explicite en utilisant la commande `\resetchoix`. Voici un exemple complet :

```
\begin{qcm}{Niveau du manuel}{1}
  \begin{enumerate}
  \item Vous trouvez que ce manuel est :
    \begin{tabular}{ll}
```

3. Ce n'est pas vrai, il y en a cinq mais les deux derniers types seront vus dans le chapitre traitant des devoirs de synthèse.

```
      \choix ignoble & \choix trop fouillis \\[6pt]
      \choix pas mal & \choix merveilleux
    \end{tabular}
  \item Vous préconisez :\resetchoix
    \begin{tabular}{ll}
      \choix une lapidation & \choix un blâme \\[6pt]
      \choix un bon-point & \choix une statue
    \end{tabular}
  \end{enumerate}
\end{qcm}
```

qui donne le résultat :

1 QCM Niveau du manuel — 1 min — Corrigé p. 18

1 Vous trouvez que ce manuel est : [a] ignoble [b] trop fouillis [c] pas mal [d] merveilleux

2 Vous préconisez : [a] une lapidation [b] un blâme [c] un bon point [d] une statue

On reviendra sur le cadre en haut à droite qui indique un numéro de page.

La mise en page des choix est laissé à l'appréciation de l'auteur (la classe ne fixe rien en la matière).

L'environnement `exo` pour produire des exercices « normaux » demande quatre paramètres. Dans l'ordre :

- le titre ;
- le lycée de provenance ;
- la difficulté (0, 1, 2 ou 3) ;
- le temps supposé de résolution de l'exercice.

L'intérieur de l'environnement contient l'énoncé de l'exercice et est entièrement libre. Par exemple, en saisissant :

```
\begin{exo}{Petite récréation}{Lycée Fermat}{3}{99}
  Voici quelques problèmes pour vous amusez une vie entière.
  \begin{enumerate}
  \item Retrouvez la démonstration du grand théorème de
    Fermat. L'équation
    \[a^n + b^n = c^n\]
  avec $a$, $b$ et $c$ entiers strictement positifs et $n$
  entier strictement supérieur à~2 n'a pas de solution.

  Il est sous-entendu que les travaux de Wiles ne devront pas
```

```
  être consultés !
  \item Si vous avez encore un peu de temps, le problème des
    zéros de la fonction $\zeta$ de Riemann vous attend.
  \end{enumerate}
\end{exo}
```

donne le résultat :

2 Petite récréation

★★★ 99 min Corrigé p. 18

Lycée Fermat

Voici quelques problèmes pour vous amusez une vie entière.

1 Retrouvez la démonstration du grand théorème de Fermat. L'équation

$$a^n + b^n = c^n$$

avec a, b et c entiers strictement positifs et n entier strictement supérieur à 2 n'a pas de solution.

Il est sous-entendu que les travaux de Wiles ne devront pas être consultés !

2 Si vous avez encore un peu de temps, le problème des zéros de la fonction ζ de Riemann vous attend.

Pour les environnements `qcm` et `exo`, les solutions doivent être indiquées ensemble au niveau de l'onglet « Corrigés ». La classe automatise tout le processus de la façon suivante :

1 Chaque exercice (QCM ou exercice classique) doit être *immédiatement* suivi d'un environnement `corrige` où figure le texte de la correction.

2 Pour afficher l'ensemble des corrections, il suffit d'appeler la commande `\AfficheCorrige`.

Dans ce manuel, l'auteur a un peu menti : Lorsque les deux exemples de QCM et d'exercice ont été tapés, ils ont été chacun immédiatement suivis de leur environnement `corrige`. La saisie avait été :

```
\begin{qcm}{Niveau du manuel}{1}
  <texte montré précédemment>
\end{qcm}
\begin{corrige}
  \begin{enumerate}
  \item Les réponses \choix[a] et \choix[d] sont trop
    extrêmes pour être honnêtes.  Seules les réponses
    \choix[b] et \choix[c] sont acceptées.
  \item Voir ci-dessus.
  \end{enumerate}
\end{corrige}
...
```

```
\begin{exo}{Petite récréation}{Lycée Fermat}{3}{99}
  <texte montré précédemment>
\end{exo}
\begin{corrige}
  \begin{enumerate}
  \item Dès que vous aurez trouvé, comparez vos résultats
    avec ceux de Wiles.
  \item La taille de ce manuel est trop petite pour y
    faire tenir la solution. C'est bien dommage.
  \end{enumerate}
\end{corrige}
```

Cela montre au passage la façon d'obtenir les lettres encadrées de façon non automatique, ce qui servira beaucoup pour les corrigés de QCM : la commande `\choix` accepte un argument optionnel qui permet d'indiquer ce que l'on veut à l'intérieur du cadre.

À propos des choix type QCM, la maquette (et la classe) n'impose rien sur la façon de les placer dans la page mais il est bon de se souvenir que l'environnement `tabularx` permet de disposer du matériel réparti régulièrement sur la largeur de page. On verra cela à la section 6 page 25.

Pour afficher les corrections, il faut utiliser la commande `\AfficheCorrige`. En la saisissant ici, on obtient le résultat :

1 QCM Niveau du manuel

Énoncé p. 16

1 Les réponses [a] et [d] sont trop extrêmes pour être honnêtes. Seules les réponses [b] et [c] sont acceptées.

2 Voir ci-dessus.

2 Petite récréation

Énoncé p. 17

Lycée Fermat

1 Dès que vous aurez trouvé, comparez vos résultats avec ceux de Wiles.

2 La taille de ce manuel est trop petite pour y faire tenir la solution. C'est bien dommage.

On peut maintenant expliquer le numéro de page dans le cadre en haut à droite : c'est la page de correction. Les plus perspicaces des lecteurs auront remarqué qu'il y a aussi un numéro de page au niveau des solutions qui correspond à la page où l'exercice a été posé.

3.4 Organisation des fichiers

L'organisation des chapitres en un onglet « Cours » puis en deux onglets « Énoncés » et « Corrigés » qui seront intriqués au niveau du code source impose plus ou moins la structure suivante pour l'ensemble du document :

- un document maître ;
- un fichier source pour l'onglet « Cours » de chaque chapitre ;
- un fichier source pour les onglets « Énoncés » et « Corrigés » de chaque chapitre.

Cela donne le squelette suivant pour un document maître :

```
\documentclass[crop,realfonts]{prepamath}
\graphicspath{{figures/}}

\input{commandes}
\title{Test de maquette}
\author{Jean-Côme Charpentier}

\begin{document}

\frontmatter

\include{src/avant-propos}

\tableofcontents

\include{src/methode}

\mainmatter
\include{src/01-Cours}
\include{src/01-Enonces}
% Même chose pour les autres chapitres

\devoirsynthese{Bac blanc}
               {Baccalauréat 2008, Académie de Toulouse}
\include{src/bac-blanc}
\end{document}
```

Une telle organisation n'est bien sûr pas une obligation mais elle est quand même fortement recommandée. D'abord, c'est une bonne façon de s'y retrouver, surtout en fin de travail, et, d'autre part, cela permet aux metteurs en pages d'avoir une organisation semblable quel que soit l'auteur.

Les auteurs ou metteurs en page familiers de LaTeX auront peut-être envie de travailler avec `\input` plutôt que `\include`. Le faire avec `\include` permet de réaliser des compilations partielles tout en gardant un foliotage correct et les accès à toutes les références croisées du texte.

On remarquera au passage qu'il n'y a pas de commande `\usepackage`, celles-ci étant avantageusement placées dans le fichier `commandes`[4]. On remarquera également qu'il y a une commande `\graphicspath` qui permet de spécifier un répertoire de recherche pour les fichiers graphiques. Là aussi, ce n'est pas une obligation mais on recommande chaudement de suivre cette directive : mettre tous les fichiers graphiques (`eps` normalement) dans un sous-répertoire `figures` et placer cette commande en préambule du document maître.

4. On peut bien entendu vouloir le contraire : un package personnel qui contiendra des commandes personnelles. Le choix est affaire de goût.

4 Listes

On retrouve les trois types de listes classiques (à savoir environnements `itemize`, `enumerate` et `description`). Les listes à puces sont vraiment des listes à puces et non pas des listes à tirets. Les listes numérotées utilisent des numéros dans un cadre à fond coloré. Les listes de description ne présentent aucune différence avec celles des classes standards.

Il va de soi que la mise en page de ces listes fait partie de la maquette de la collection et que les packages de type `enumitem` ou `enumerate` permettant de modifier leur aspect sont interdits. Pour le reste, tout fonctionne comme avec les classes standards : chaque item est introduit par la commande `\item` et il est possible de faire des références croisées sur le numéro d'item avec le mécanisme `\label-\ref-\pageref`.

Il y a déjà eu des exemples de listes dans ce manuel mais pour bien fixer les choses, voici un exemple avec le code correspondant. Lorsqu'on tape :

```
\begin{itemize}
\item premier item ;
\item deuxième item.
\end{itemize}
\begin{enumerate}
\item Premier item.
\item Deuxième item.
\end{enumerate}
\begin{description}
\item[first] Premier item.
\item[second] Deuxième item.
\end{description}
```

on obtient le résultat :

- premier item ;
- deuxième item.

1. Premier item.
2. Deuxième item.

first Premier item.

second Deuxième item.

Dans les documents `prepamath`, on sera amené à utiliser un autre type de liste : les choix d'un QCM. Comme l'organisation des réponses possibles est assez libre, la classe ne fige pas la mise en page dans un environnement mais ne donne que la commande `\choix`. Cette commande utilise le compteur LaTeX `choix`, on peut donc modifier la numérotation des QCM en redéfinissant la commande `\thechoix` même si c'est très fortement déconseillé.

Voici le résultat obtenu avec cette commande. Le source :

DÉTAILS

```
Que préférez-vous \choix Les maths \choix la physique
                  \choix la chimie
```

donne le résultat :

Que préférez-vous [a] Les maths [b] la physique [c] la chimie

Le compteur `choix` est remis à zéro à chaque changement d'exercice, à chaque changement de solution, de section (ou sous-section ou…). Parfois, cela ne suffit pas. Typiquement lorsque dans un même exercice, on va avoir plusieurs QCM. Pour remettre le compteur à zéro, la classe propose la commande `\resetchoix`. Par exemple :

```
\begin{enumerate}
\item $f(x)=0$ a \choix une solution
  \choix deux solutions \choix trois solutions
\item L'équation $f(x)=g(x)$ admet \resetchoix
  \choix une \choix deux ou \choix trois solutions
\end{enumerate}
```

donne le résultat :

1. $f(x) = 0$ a [a] une solution [b] deux solutions [c] trois solutions
2. L'équation $f(x) = g(x)$ admet [a] une [b] deux ou [c] trois solutions

5 Théorèmes et compagnie

La classe `prepamath` propose un certain nombre de structures de type théorème. Il s'agit d'environnements acceptant un argument optionnel (normalement pour un titre). L'exemple de base est :

```
\begin{thm}
  Tout ce qui est rare est cher.\par
  Un cheval bon marché est rare.
\end{thm}
```

qui donne le résultat :

Théorème 1

Tout ce qui est rare est cher.
Un cheval bon marché est rare.

L'utilisation de l'argument optionnel :

```
\begin{thm}[théorème de Fermat]
  L'équation entière
  \[a^n + b^n = c^n\]
  n'a pas de solution non triviale pour $n>2$.
\end{thm}
```

donne le résultat :

Théorème 2 : théorème de Fermat

L'équation entière

$$a^n + b^n = c^n$$

n'a pas de solution triviale pour $n > 2$.

La classe définit les environnements `thm`, `thms`, `coro`, `coros`, `defin`, `defins`, `prop` et `props` pour, respectivement, un théorème, des théorèmes, un corollaire, des corollaires, une définition, des définitions, une propriété et des propriétés.

La commande `\DefineTheoremLike` permet de définir de nouvelles structures similaires. Cette commande demande deux paramètres : le nom du nouvel environnement et son titre. Il y a également un argument optionnel qui permet d'indiquer le nom du compteur qui sera utilisé. Par défaut, le nom du compteur est le nom de l'environnement. La marche typographique de Prepamath fait que les structures au noms différents auront des compteurs indépendants. En revanche, lorsqu'on définit, par exemple, les environnements `defin` et `defins`, le premier est défini sans argument optionnel (donc avec un compteur `defin`) mais le deuxième utilise l'argument optionnel de `\DefineTheoremLike` afin d'utiliser également le compteur `defin` au lieu d'un compteur `defins`.

L'exemple d'utilisation présenté ci-dessous ne devrait pas être reproduit tel quel : normalement, la commande de création d'une structure de type théorème ne devrait se trouver que dans le préambule d'un document[5]. Le source :

```
\DefineTheoremLike{lm}{Lemme}
\begin{lm}[Lemme du cheval]
  Un cheval bon marché est rare.\par
  Un cheval bon marché est cher.
\end{lm}
```

donne le résultat :

Lemme 1 : Lemme du cheval

Un cheval bon marché est rare.
Un cheval bon marché est cher.

Ces structures présentent un filet de couleur à gauche, sur toute la hauteur du texte.

Un deuxième type de structures, celles du type des exemples, fonctionne de la même façon sans ce filet. Une autre différence est que, pour cette version de la classe, les structures de type théorème ne peuvent pas être coupées sur deux pages alors que les structures de types exemple peuvent l'être. La dernière différence est que ces structures ne possèdent pas de numérotation automatique.

La classe `prepamath` met à disposition les environnements `Remarque`, `Exemple`, `Remarques` et `Exemples` pour, respectivement, une remarque, un exemple, plusieurs remarques et plusieurs exemples. Par exemple, le code :

5. Ou, nettement préférable, dans le fichier `commandes.tex` ou dans un package personnel.

```
\begin{Exemple}[exemple idiot]
  $3^3+4^3$ n'est pas un cube parfait.
\end{Exemple}
```

donne le résultat :

Exemple (exemple idiot) : $3^3 + 4^3$ n'est pas un cube parfait.

En réalité, les environnements pour plusieurs remarques ou plusieurs exemples n'ont pas le même comportement que pour un seul. Les environnements `Exemple` et `Remarque` ne font pas de saut de ligne à la suite du mot « Exemple » ou « Remarque ». Les environnements `Exemples` et `Remarques` ont un saut de ligne après le mot « Exemples » ou « Remarques ». Par exemple, le code

```
\begin{Remarques}[évidentes]
  À propos du théorème de Fermat-Wiles :
  \begin{itemize}
  \item L'exemple précédent n'était qu'un cas très
    particulier du théorème de Fermat-Wiles.
  \item Ce théorème sert un peu trop dans les différents
    exemples. On fera mieux dans la suite du manuel.
  \end{itemize}
\end{Remarques}
```

donne le résultat :

Remarques (évidentes)

À propos du théorème de Fermat-Wiles :

- L'exemple précédent n'était qu'un cas très particulier du théorème de Fermat-Wiles.
- Ce théorème sert un peu trop dans les différents exemples. On fera mieux dans la suite du manuel.

Comme pour les environnements avec filet, il est possible de définir de nouveaux environnements de ce type avec les commandes `\DefineExampleLike` et `\DefineExampleLikeWithPar` qui fonctionnent exactement comme la commande `\DefineTheoremLike` (sauf qu'il n'y a pas d'argument optionnel puisqu'il n'y a pas de compteur). Dans un cas comme dans l'autre, il faudra absolument en discuter avec l'éditeur. La première sert pour les environnements sans saut de ligne et la deuxième pour les environnements avec saut de ligne.

Ces deux commandes demandent deux arguments obligatoires (le nom de l'environnement et le titre). Il n'y a pas d'argument optionnel puisqu'il n'y a pas de numérotation.

6 Tableaux

La classe `prepamath` fait appel à plusieurs packages gérant les tableaux. On trouve ainsi `array`, `cellspace`, `colortbl` et `tabularx`. Ce manuel ne prétend pas reprendre l'ensemble des fonctionnalités offertes par ces packages et on se reportera, le cas échéant, aux différentes documentations correspondantes.

On supposera que les fonctionnalités du package `array` sont connues. Si ce n'est pas le cas, il faudrait absolument regarder sa documentation !

Le package `colortbl` (appelé via le package `xcolor`) nécessite quelques précaution d'emploi puisque les couleurs sont régies de façon extrêmement stricte par la maquette. En clair, vous n'avez le droit de n'utiliser qu'une seule couleur pour les tableaux et il s'agit de la couleur `TabularColor`.

La classe offre un raccourci très utile avec la commande `\tabulartitle`. Elle sert à obtenir une ligne de tableau en jaune – normalement la ligne de titre – sans se soucier des détails techniques. Voici un exemple d'utilisation. Le code :

```
\begin{tabular}{|*{3}{c|}}
  \hline\multicolumn{3}{|c|}{pré-titre}\\\hline
  \tabulartitle $f$ & $f'$ & $F$ \\\hline
  $1$ & $0$ & $x + k$ \\
  $x$ & $1$ & $x^2/2 + k$ \\
  $x^2$ & $2x$ & $x^3/3 + k$ \\\hline
\end{tabular}
```

donne le résultat :

pré-titre		
f	f'	F
1	0	$x+k$
x	1	$x^2/2+k$
x^2	$2x$	$x^3/3+k$

En fait, dans la mesure du possible, il faudrait que les tableaux aient une ligne de titre utilisant la commande `\tabulartitle`.

Pour les autres commandes permettant d'obtenir des cases de tableaux en couleur, il faudra utiliser la couleur `TabularColor`. Voici un exemple typique :

```
\begin{tabular}{|>{\columncolor{TabularColor}}c|*{2}{c}|}
  \hline
  \tabulartitle $\times$ & 1 & 2 \\\hline
                         1 & 1 & 2 \\
                         2 & 2 & 1 \\\hline
\end{tabular}
```

qui donne le résultat :

$\times$	1	2
1	1	2
2	2	1

DÉTAILS

Comme ce type de présentation est très courant, la classe offre trois raccourcis permettant de spécifier des colonnes avec fond coloré. Il s'agit des spécificateurs de colonnes R, L, C qui fonctionnent exactement comme les spécificateurs habituels r, l et c mais avec le fond coloré.

L'exemple précédent pouvait ainsi être obtenu avec la syntaxe un peu plus courte :

```
\begin{tabular}{|C|*{2}{c}|}
  \hline
  \tabulartitle $\times$ & 1 & 2 \\\hline
                         1 & 1 & 2 \\
                         2 & 2 & 1 \\\hline
\end{tabular}
```

qui donne toujours le résultat :

×	1	2
1	1	2
2	2	1

En plus des spécificateurs R, L, C pour les colonnes colorées, on trouve également les spécificateurs P, M, B qui demandent un paramètre et qui agissent exactement comme les spécificateurs p, m et b (m et b du package `array`). On a également le spécificateur x (minuscule) pour une colonne de type X avec couleur (voir `tabularx` pour le spécificateur X).

Le fonctionnement de base de l'environnement `tabularx` est supposé connu. On va donner une utilisation pratique de ce package dans le cadre d'ouvrage Prepamath. N'oubliez pas que le placement des petits cadres de choix (pour les QCM par exemple) posent souvent des problèmes d'alignement. Les tableaux `tabularx` permettent d'obtenir des placements qui tiennent compte de la largeur de la page et qui vont donc proposer des mises en page cohérentes. À ce propos, l'utilisation du multicolonnage est dans l'immense majorité des cas une très mauvaise idée. Voici une présentation « classique » :

```
L'équation $x^2+bx+c=0$ où $c<0$\par
\begin{tabularx}{\linewidth}
                {@{}*{4}{l@{ }>{\raggedright}X@{ }}@{}}
\choix & n'a aucune solution &
\choix & a une unique solution &
\choix & a deux solutions &
\choix & a trois solutions
\end{tabularx}\resetchoix

La courbe $C$ admet pour asymptote la droite d'équation\par
\begin{tabularx}{\linewidth}{@{}XXXX@{}}
\choix $x=2$   & \choix $y=3$   &
\choix $y=x+3$ & \choix $y=-x+3$
\end{tabularx}
```

qui donne le résultat :

L'équation $x^2 + bx + c = 0$ où $c < 0$

[a] n'a aucune solution [b] a une unique solution [c] a deux solutions [d] a trois solutions

La courbe C admet pour asymptote la droite d'équation

[a] $x = 2$ [b] $y = 3$ [c] $y = x + 3$ [d] $y = -x + 3$

Dans les fonctionnalités de `tabularx`, on a la commande `\tabularxcolumn` qui permet de redéfinir le comportement des colonnes de type `X`. Par défaut, ces colonnes se comporte comme des paragraphes dont on alignerait les premières lignes. On peut vouloir un alignement sur la dernière ligne ou, plus fréquemment un alignement centré. La définition par défaut d'une colonne `X` est :

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

Pour obtenir des alignements centrés (par exemple) il suffit de demander :

```
\renewcommand{\tabularxcolumn}[1]{m{#1}}
```

Combiné avec la commande `\newcolumntype` du package `array`, on peut obtenir tous les effets désirés :

```
\renewcommand{\tabularxcolumn}[1]{m{#1}}
\newcolumntype{d}{>{\raggedright\arraybackslash}X}
\newcolumntype{e}{>{\centering\arraybackslash}X}
\newcolumntype{f}{>{\raggedleft\arraybackslash}X}
\begin{tabularx}{0.7\linewidth}{|X|d|e|f|}
  \hline
  Ceci est un texte justifié &
  Ceci est un texte au fer à droite &
  texte centré &
  Ceci est un texte au fer à gauche \\\hline
\end{tabularx}
```

qui donne :

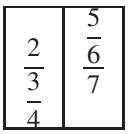
Ceci est un texte justifié	Ceci est un texte au fer à droite	texte centré	Ceci est un texte au fer à gauche

Le dernier package – `cellspace` – est assez méconnu et c'est bien dommage. Il automatise proprement la résolution d'un problème très fréquent avec les tableaux comportant des filets horizontaux : le fait que le texte touche à ces filets. Par exemple dans le tableau suivant :

```
\begin{tabular}{|c|c|}
  \hline
  $\dfrac{2}{\dfrac{3}{4}}$ & $\dfrac{\dfrac{5}{6}}{7}$
  \\\hline
\end{tabular}
```

le résultat est assez pitoyable :

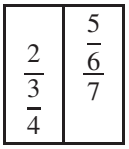
DÉTAILS



La solution immédiate (et propre) est de préfixer les motifs du tableau par la lettre Z capitale. Cela donne :

```
\begin{tabular}{|Zc|Zc|}
  \hline
  $\dfrac{2}{\dfrac{3}{4}}$ & $\dfrac{\dfrac{5}{6}}{7}$
  \\\hline
\end{tabular}
```

avec le résultat bien plus acceptable :



Normalement, le préfixe devrait être un S capital mais l'extension `siunitx` a pensé qu'elle était seule au monde et a introduit une incompatibilité entre ces deux extensions en se servant également de cette lettre S.

Pour les motifs composés (par exemple `p{3cm}`) il faudra mettre l'ensemble du motif entre accolade à la suite de la lettre Z (`Z{p{3cm}}`). Ce préfixe de motif ne fonctionne pas dans les tableaux mathématiques (`array`) mais on peut toujours se débrouiller pour utiliser un tableau non mathématique.

7 Cadres

La classe `prepamath` propose plusieurs types de cadres. On trouve ainsi les environnements : `aretenir`, `attention`, `anecdote`, `methode` et `problematique`.

Tous ces blocs peuvent apparaître partout dans l'ouvrage mais n'auront pas forcément le même aspect si la page a un fond coloré ou non.

Le code :

```
\begin{aretenir}
  Environnement \texttt{aretenir}.
\end{aretenir}
\begin{attention}
  Environnement \texttt{attention}\\
  sur plusieurs lignes\\
  pour mieux voir
\end{attention}
```

donne le résultat :

À RETENIR

Environnement `aretenir`.

ATTENTION

Environnement `attention`
sur plusieurs lignes
pour mieux voir

On a la possibilité de définir de nouveaux environnements de ce type avec les deux commandes :

- `\DefineWarningLike` pour les structures sans barre ;
- `\DefineWarningLikeWithBar` pour les structures avec une barre.

Ces deux commandes ont la même syntaxe, à savoir trois paramètres donnant respectivement le nom de l'environnement, le logo et le titre de la structure. La classe donne accès aux logos suivants :

- `\LogoExclamation` .....  ;
- `\LogoBulb` .....  ;
- `\LogoAnecdote` .....  ;
- `\LogoSemiClock` .....  ;
- `\LogoQuestion` .....  ;
- `\LogoComputer` .....  ;
- `\LogoRightArrow` .....  ;
- `\LogoComputerOld` .....  .

Bien sûr, on peut produire ses propres logos mais il y a des normes de couleurs et de tailles bien fixées. Les auteurs familiers de la programmation PSTricks pourront regarder le code de ces différentes commandes dans le fichier `prepamath.cls`. Si besoin, la classe étoffera l'offre de logos disponibles. C'est d'ailleurs ce qui s'est passé en pratique, certains logos n'existant qu'à partir d'une certaine version de la classe.

Que ce soit pour la création d'un nouveau type de bloc ou pour la création d'un nouveau logo, il faut bien voir qu'on travaille alors hors maquette. Toute décision à ce niveau exige un accord de la part de l'éditeur… Je conseille de bien préparer son argumentaire ! Sauf raison vraiment très forte, ce type de demande devrait être refusé.

Il existe deux cadres un peu plus spéciaux réservés à l'informatique. Le premier est l'environnement `algorithme` dont la particularité est que les sauts de ligne et les espaces du source seront respectés. Le code :

```
\begin{algorithme}{}
  m $\gets$ (a + b) / 2
  Tant que b - a $>$ seuil
    Si f(a)*f(m) $>$ 0
      a $\gets$ m
    Sinon
      b $\gets$ m
    Fin Si
    m $\gets$ (a + b) / 2
  Fin Tant que
\end{algorithme}
```

donne le résultat :

```
  m ← (a + b) / 2
  Tant que b - a > seuil
    Si f(a)*f(m) > 0
      a ← m
    Sinon
      b ← m
    Fin Si
    m ← (a + b) / 2
  Fin Tant que
```

Il faudra faire attention à ce que les fontes utilisées ne proposent pas d'équivalence correcte pour certains symboles mathématiques et qu'il est plus prudent, voire obligatoire, de saisir ces derniers en mode mathématiques. Ainsi, si les symboles +, /, * on une transcription texte correcte, il n'en est pas de même du symbole « < » et il est alors nécessaire de le saisir en mode mathématique.

Comme on peut le voir sur l'exemple précédent, l'environnement `algorithme` demande un argument obligatoire. Si celui-ci est vide, on obtient un cadre sans cartouche et sans titre. Sinon, cet argument indique le titre de l'algorithme. Ainsi, le code :

```
\begin{algorithme}{Algorithme de seuil}
  m $\gets$ (a + b) / 2
  Tant que b - a $>$ seuil
    Si f(a)*f(m) $>$ 0
      a $\gets$ m
    Sinon
      b $\gets$ m
    Fin Si
    m $\gets$ (a + b) / 2
  Fin Tant que
\end{algorithme}
```

donne le résultat :

Algorithme de seuil

```
m ← (a + b) / 2
Tant que b - a > seuil
  Si f(a)*f(m) > 0
    a ← m
  Sinon
    b ← m
  Fin Si
  m ← (a + b) / 2
Fin Tant que
```

La classe propose également un environnement listing qui permet d'afficher des listings. Le fait de travailler en utf8 complique un peu le processus et ne permet pas facilement de saisir directement le code à l'intérieur de l'environnement. Le parti pris a été de copier le code dans un fichier externe et de l'inclure grâce à cet environnement. Supposons qu'on ait le fichier `pgcd.py` suivant :

```
def PGCD(a, b):
    if a % b == 0:
        return b
    else:
        return PGCD(b, a % b)

print("pgcd({},{}) = {}".format(45, 12, PGCD(45,12)))
```

Dans ce cas, le code suivant :

```
\begin{listing}{code PGCD}{pgcd.py}
\end{listing}
```

va donner le résultat :

code PGCD

```
def PGCD(a, b):
    if a % b == 0:
        return b
    else:
        return PGCD(b, a % b)

print("pgcd({},{}) = {}".format(45, 12, PGCD(45,12)))
```

Le premier argument de l'environnement est le titre placé dans le cartouche. Si cet argument est vide, il n'y aura ni titre, ni cartouche supérieur avec le logo (comme pour l'environnement `algorithme`).

Le deuxième argument est le nom du fichier. Si les fichiers sources sont tous placés dans un répertoire particulier, on peut redéfinir la macro `\DirListing`

DÉTAILS

pour le spécifier. Cela permettra de ne pas avoir à le répéter à chaque appel de l'environnement `listing`.

Il est possible de placer un texte introductif dans le cadre : il suffit de le spécifier dans le corps de l'environnement. Enfin, cet environnement accepte un argument optionnel entre crochet qui permet de spécifier ou de modifier les paramètres de l'extension `listings`. Les paramètres déclarées par la classe sont :

- inputencoding=utf8 ;
- showstringspaces=false ;
- basicstyle=\ttfamily ;
- keywordstyle=\color{DO} ;
- language=python ;
- columns=flexible ;
- keepspaces=true ;
- breaklines=true.

Voici un deuxième exemple qui utilise ces spécificités.

```
\begin{listing}[showstringspaces, keywordstyle=\color{DB}]
      {code PGCD}{pgcd.py}
  Ce code est bloqué sur les valeurs 45 et 12.
\end{listing}
```

va donner le résultat :

code PGCD

```
Ce code est bloqué sur les valeurs 45 et 12.
def PGCD(a, b):
    if a % b == 0:
        return b
    else:
        return PGCD(b, a % b)

print("pgcd({},{}) = {}".format(45, 12, PGCD(45,12)))
```

La coloration syntaxique est faite suivant le langage Python mais rien n'empêche d'utiliser nu autre langage. L'argument optionnel de l'environnement indiquant les paramètres du package `listingsutf8`, il suffit d'utiliser le mot-clé `language` (voir documentation du package `listings` pour plus de renseignements).

8 Commandes Nathan

Depuis la version 2.28, la classe permet l'utilisation de liens externes. Tout d'abord les liens hypertextes classiques, qui vont être activés avec l'option de classe `hyperref`, auxquels il faut ajouter les références à des outils hébergés sur Internet qu'on va appeler les « commandes Nathan ».

Au niveau du corps de document, on peut utiliser quatre commandes différentes permettant d'indiquer un outil particulier :

- `\NathanLiveAudio[texte doc]{texte list}{URL}`
- `\NathanLiveProg[texte doc]{texte list}{URL}`
- `\NathanLiveVideo[texte doc]{texte list}{URL}`
- `\NathanLiveWeb[texte doc]{texte list}{URL}`

Ces quatre commandes se comportent exactement de la même façon et ont la même syntaxe d'appel. Il y a un argument optionnel entre crochets droits et deux arguments obligatoires entre accolades.

Ces quatre commandes on une version étoilée qui vont afficher un pictogramme en lien avec leur nom. Ces pictogrammes, ainsi qu'un cinquième, peuvent être affichés de façon indépendantes avec les commandes suivantes (suivi de leur résultat) :

- `\PictoAudio` donne
- `\PictoProg` donne
- `\PictoVideo` donne
- `\PictoWeb` donne
- `\PictoNathanLive` donne

Voici ce que donne, par exemple, la commande `\NathanLiveVideo` :

```
\NathanLiveVideo{Ce texte sera dans la liste des vidéos}
                {\http://www/lesite.com}
```

sans étoile donc sans pictogramme, cela donnera le résultat :

Retrouvez le corrigé de cet exercice en vidéo

La même commande avec une étoile, c'est-à-dire avec un code du type :

```
\NathanLiveVideo*{Ce texte sera aussi dans la liste des vidéos}
                 {\http://www/lesite.com}
```

donnera le résultat :

Retrouvez le corrigé de cet exercice en vidéo

On peut remarquer l'emploi automatique du pictogramme dédié à l'outil indiqué (audio, programme, vidéo ou web). Dans les deux cas on a un texte par défaut.

DÉTAILS

Pour modifier le texte par défaut, il suffit d'utiliser l'argument optionnel de la commande. Par exemple, le code :

```
\NathanLiveVideo[Pour plus de renseignements : \PictoNathanLive]
                {Autre entrée de la liste des vidéos}
                {\http://www/lesite.com}
```

donnera le résultat :

Pou plus de renseignements : 

Dans les deux cas, on a spécifié une URL. En cas de composition avec l'option de classe `hyperref`, la ligne entière deviendra un lien hypertexte sur l'URL indiquée.

Voici les textes par défaut de ces quatre commandes. Le code :

```
\NathanLiveAudio{texte}{}
\NathanLiveProg{texte}{}
\NathanLiveVideo{texte}{}
\NathanLiveWeb{texte}{}
```

donnera le résultat :

Retrouvez le son dont il est question dans cet exercice

Retrouvez ce programme en ligne

Retrouvez le corrigé de cet exercice en vidéo

Retrouvez le site web

Le dernier point à voir concernant toutes ces commandes et celui des listes d'objets Nathan. Il y a deux types de listes. Un premier qui est destiné à être affiché en fin d'ouvrage. Pour cela, il suffit d'appeler une des commandes suivantes :

- `\ListOfAudio{texte}`
- `\ListOfProg{texte}`
- `\ListOfVideo{texte}`
- `\ListOfWeb{texte}`

En pratique, la seule qui soit autorisée pour l'instant est la liste des vidéos mais, dans le cas contraire, les commandes seront déjà prêtes.

L'argument obligatoire de ces commandes est un texte qui sera affiché en bas de page, après la liste de tous les objets.

Dans cette liste, on trouvera le texte qui a été indiqué en premier paramètre obligatoire dans la commande `\NathanLiveXxx`. Ainsi, avec les trois commandes appelées ci-dessus, la commande :

```
\ListOfVideo{Texte terminal\par sur plusieurs lignes}
```

donnera le résultat montré en fin de manuel.

À noter que les objets Nathan ne devraient apparaître que dans les chapitres « normaux » ou dans le chapitre « Avant-propos » de la partie `\frontmatter`.

Un deuxième type de liste répertorie tous les objets NathanLive de façon très sommaire en ne donnant que leur type (Audio, Prog, Video ou Web) et la page où ils apparaissent. Ces listes sont destinées à une vérification et non à une impression finale. Il existe deux commandes permettant de les produire :

- `\printNL` qui imprime le résultat en triple colonne.
- `\exportNL[fichier]` qui exporte son résultat dans un fichier externe dont le nom est optionnellement indiqué (par défaut, c'est le nom du fichier maître avec l'extension `.nl`.

Par exemple la commande

```
\exportNL
```

indiquée à ce niveau va produire le fichier `manuel.nl` dont voici le contenu :

```
----------------
liste des videos
----------------
Page 33
Page 33
Page 34
Page 34
--------------------
liste des programmes
--------------------
Page 34
----------------------
liste des pages audio
----------------------
Page 34
-------------------
liste des sites web
-------------------
Page 34
```

Les cinq commandes `\PictoXxx` pour obtenir un pictogramme isolé acceptent un argument optionnel qui permettra de créer une telle entrée dans ces listes. La valeur de l'argument ne peut être que `Audio`, `Prog`, `Video` ou `Web`.

En pratique, seule la commande `\PictoNathanLive` devrait être utilisée avec son argument optionnel.

DÉTAILS

9 Commandes diverses

9.1 Commandes mathématiques

La classe `prepamath` définit (ou redéfinit) un certain nombre de macros pour le mode mathématique. On trouve ainsi :

Macro	Signification
`\nep`	e Base du logarithme népérien. On le compose en romain.
`\im`	i Base de la partie imaginaire des nombres complexes.
`\ch`	ch Cosinus hyperbolique
`\sh`	sh Sinus hyperbolique
`\Re`	Re Partie réelle des nombres complexes (redéfinition)
`\Im`	Im Partie imaginaire des nombres complexes (redéfinition)
`\Arg`	arg Argument d'un nombre complexe
`\dd`	d Différentiel (composé en romain)
`\card`	card Cardinal d'un ensemble
`\PGCD`	pgcd Plus grand diviseur commun
`\PPCM`	ppcm Plus petit multiple commun
`\esp`	E Espérance mathématique
`\var`	V Variance
`\ent`	E Partie entière
`\Moy{x}`	$\overline{x}$ Moyenne
`\gvec{AB}`	$\overrightarrow{AB}$ Vecteur
`\N`	$\mathbb{N}$ Ensemble des entiers naturels
`\Z`	$\mathbb{Z}$ Ensemble des entiers relatifs
`\Q`	$\mathbb{Q}$ Ensemble des rationnels
`\R`	$\mathbb{R}$ Ensemble des réels
`\C`	$\mathbb{C}$ Ensemble des complexes

9.2 Touches de calculatrices

La classe `prepamath` offre la macro `\T` qui permet de placer n'importe quoi à l'intérieur d'une touche de calculatrice. Par exemple le code `\T{abc}` donne le résultat **abc**. Si le contenu doit être composé en mode mathématique, on emploiera la macro `\Tm`. Par exemple `\Tm{\mu}` donne μ.

Pour éviter une frappe fastidieuse, les touches les plus courantes ont leur propres macros. Il est très facile d'en définir d'autre. On donne la définition des macros `\TTA` qui affiche un « A » et `\Tcarre` qui affiche le carré :

```
\newcommand*\TTA{\T{A}\xspace}
\newcommand*\Tcarre{\Tm{x^2}\xspace}
```

On notera l'ajout du `\xspace` qui permet de ne pas avoir à ajouter de code particulier après la macro si on veut une espace.

Le tableau suivant dresse la liste de toutes les touches prédéfinies de la classe :

\Tpourcent %	\Tcalc CALC	\TTA A	\TTN N
\Tcarre x^2	\Trcl RCL	\TTB B	\TTO O
\Tcube x^3	\Tmem MEM	\TTC C	\TTP P
\Tpuissance x^n	\Tdel DEL	\TTD D	\TTQ Q
\Tlog log	\Tac AC	\TTE E	\TTR R
\Tln ln	\Tseconde SECONDE	\TTF F	\TTS S
\Tneg (−)	\Tshift SHIFT	\TTG G	\TTT T
\Tsin SIN	\Tmul ×	\TTH H	\TTU U
\Tcos COS	\Tdiv ÷	\TTI I	\TTV V
\Ttan TAN	\Tplus +	\TTJ J	\TTW W
\Tasin ASIN	\Tmoins −	\TTK K	\TTX X
\Tacos ACOS	\Tegal =	\TTL L	\TTY Y
\Tatan ATAN	\Texe EXE	\TTM M	\TTZ Z
\TAsin SIN^{-1}	\Tenter ENTER		
\TAcos COS^{-1}	\Tans ANS		
\TAtan TAN^{-1}			

Dans la version antérieure à 2.0 de la classe, les touches étaient plus complexes et pouvaient provoquer des débordements de capacités. Par exemple, pour être capable de construire cette page, ni une main memory (mémoire principale) de 3 000 000 (valeur par défaut sous TEXLive à l'époque), ni une main memory de 5 000 000 n'avaient suffit. Il avait fallu porter cette valeur à 8 000 000.

Aujourd'hui, sur une TEXLive, la valeur par défaut de la mémoire principale est 5 000 000 et elle suffit pour produire cette page.

10 Syntaxe LaTeXienne

Il est hors de question de faire un cours de LaTeX dans le cadre de ce manuel. On ne rappellera que les points les plus importants d'une « bonne » façon d'utiliser LaTeX, plutôt dans le cadre des questions de typographie.

La classe `prepamath` appelle le package `babel` avec l'option `frenchb`. En conséquence, le document va obéir automatiquement à un certain nombre de règles typographiques françaises. En particulier, les espaces entourant les ponctuations doubles sont gérées automatiquement (espace moyenne ou fine et insécable selon les circonstances et les endroits). Tout ce qu'il faut faire est de taper la ponctuation double normalement (pas d'ajout d'espace insécable « à la main » avec le caractère tilde `~`). Pendant qu'on est dans les ponctuations, on peut rappeler que les ponctuations simples (point et virgule) n'ont pas d'espace avant et une espace justifiante après, les parenthèses n'ont pas d'espace à l'intérieur, les guillemets doivent être saisis avec les commandes `\og` et `\fg` pour, respectivement, ouvrir et fermer les guillemets (pas de `<<` et de `>>` : la gestions des espaces seraient fautives). Il est également possible d'utiliser les caractères guillemets (français) accessibles directement mais uniquement sous les codages Latin1, Latin9 et Ansinew (ils sont rendus équivalents aux commandes `\og` et `\fg`). Enfin, les points de suspension sont indiqués avec la commande `\ldots` et non avec trois points saisis à la queue leu leu.

Il a été décidé de ne permettre aucun flottant dans la classe. Cette décision est révocable et la syntaxe ne devrait pas changer. C'est pourquoi il existe quand même les environnements `figure` et `table` classiques avec presque la même syntaxe, l'unique différence est que la commande `\caption` devient maintenant `\figcaption` ou `\tabcaption` selon l'environnement utilisé.

Comme un manuel est un ouvrage potentiellement important, il vaut mieux découper celui-ci en plusieurs fichiers :

- un document maître ;
- un ou deux fichiers par chapitres.

Le document maître n'étant alors qu'une suite de commande `\include` pour appeler les fichiers de chapitre un par un.

À RETENIR

Le manuel de l'ancienne classe préconisait de découper les chapitres en cours, exercices et corrigés. Comme la nouvelle classe demande que chaque correction suive immédiatement l'exercice correspondant, ce découpage n'est plus pertinent. On pourra néanmoins garder un découpage cours et exercices, les corrigés étant dans le même fichier que les énoncés.

Des commandes diverses destinées aux auteurs sont reprises de l'ancienne classe. En voici la liste exhaustive. Un certain nombre de ces commandes ont été gardées pour des raisons de compatibilité ascendante mais n'ont plus vraiment d'intérêt

avec la nouvelle maquette. Elles seront signalées par des astérisques et devront être évitées dans les nouveaux documents.

Macro		Exemple	Commentaire
\Arg		$\mathrm{Arg}(z) = \pi$	argument d'un complexe
\C		$\mathbb{C}$	ensemble complexes
\card		$card(\mathbb{N}) = \aleph_0$	cardinal d'un ensemble
\ch		$\mathrm{ch}(x)$	cosinus hyperbolique
\dd		$\int f(x)\,\mathrm{d}x$	d différentiel (en romain)
\ds			\displaystyle abrégé
\dvline	*		double ligne verticale dans un tableau
\ent		$\mathrm{E}(\pi) = 3$	partie entière
\esp		E	E pour espace vectoriel
\gvec		$\overrightarrow{x}$	vecteur
\Im		$\mathrm{Im}(z) = b$	partie imaginaire
\im		$a + ib$	en romain
\morestretch	*		augmente l'espacement vertical dans un tableau
\Moy		$\overline{x}$	moyenne
\N		$\mathbb{N}$	ensemble entiers naturels
\nep		e^x	en romain
\Par	*		fin de paragraphe avec un espacement vertical plus important
\PGCD		$\mathrm{pgcd}(a, b) = d$	PGCD de deux entiers
\PPCM		$\mathrm{ppcm}(a, b) = m$	PPCM de deux entiers
\Q		$\mathbb{Q}$	ensemble rationnels
\R		$\mathbb{R}$	ensemble réels
\Re		$\mathrm{Re}(z) = a$	partie réelle
\sh		$sh(x)$	sinus hyperbolique
\var		$\mathrm{V}(X)$	variable aléatoire
\Z		$\mathbb{Z}$	ensemble entiers relatifs

Des précautions particulières doivent être prises en raison de certains aspects graphique de la maquette et du fait que certaines structures ne peuvent pas être coupées (ou difficilement coupées).

Le titre des chapitres ne doit pas être trop long pour ne pas télescoper les en-têtes de la première page du chapitre. En réalité, la classe permet d'utiliser la commande \\ pour couper les lignes d'un titre. Cette coupure ne se fera qu'au niveau de la première page du chapitre. Elle est inhibée lors de l'écriture de la table des matières ainsi qu'au niveau des en-têtes des pages paires. Pour les auteurs qui n'ont pas accès aux fontes commerciales, il vaut mieux ne pas placer de coupure quitte à laisser déborder certains titres. En effet, une fois que la compilation se fera avec les fontes commerciales définitives, toute la mise en page risque de bouger et les coupures automatiques sont alors très gênantes.

De la même façon, les titres des onglets ne doivent pas être trop longs. Un exemple typique est le mot « Typographie » actuellement en cours d'utilisation qui montre clairement que la limite admissible a été atteinte avec la fonte en cours puisque les en-têtes des pages paires montrent ce titre au bord de la zone verte (avec les fontes commerciales). Ici, contrairement au titre de chapitre, il est réellement impossible de couper ces titres dans les onglets ou les en-têtes de pages impaires. De toutes façons, on ne devrait normalement jamais devoir modifier ces titres d'onglets : ils font partie de la maquette, les changer relève d'une décision éditoriale.

Les structures à fond coloré ou avec filet à gauche peuvent être coupées en utilisant les commandes `\Break` et `\BreakWithoutItem`. L'utilisation de la commande `\Break` en dehors de ces deux types d'environnements provoquera une erreur de compilation et n'aura aucune action sur le document. La commande `\Break` permet de couper ce type de bloc juste avant une commande `\item` (ce qui est généralement une bonne idée). S'il n'est vraiment pas possible de couper avant une commande `\item` ou si on ne se trouve pas dans une liste, il faut employer la commande `\BreakWithoutItem`.

Les formules hors-texte doivent être encadrées par les commandes `\[` et `\]` (ou bien placées dans des environnements `equation*`) mais jamais encadrées par des doubles dollars. L'utilisation des doubles dollars va rendre les espacements verticaux incohérents avec les autres structures mathématiques complexes et on se prive de certaines facilités éventuelles du package `amsmath`.

Des conseils du type de celui qui précède peuvent être trouvés dans le fichier `l2tabu.pdf` qui doit normalement exister sur n'importe quelle distribution TEX. Il est même possible que vous ayez le fichier `l2tabufr-heavy.pdf` (mieux si on est peu anglophone). Dans la négative, ce fichier se télécharge facilement sur Internet au niveau des miroirs CTAN. Il recense, entre autres, la liste des « péchés mortels » qu'il convient d'éviter lorsqu'on se sert de LATEX.

11 Marche typographique

La marche typographique de cette collection s'appuie en grande partie sur le *lexique des règles typographiques en usage à l'imprimerie nationale*.

En cas de doute, il faut absolument en référer soit au directeur de collection, soit à l'auteur de la classe. Ce qui suit n'est qu'une simple reprise de ce qu'avait énoncé Michel Bovani, l'auteur de la toute première version de la classe `prepamath`.

Pour souligner un mot à l'intérieur d'une phrase, on emploiera l'italique plutôt que le gras (certainement pas le soulignement). Dans l'immense majorité des cas, l'italique devrait être saisie avec la commande `\emph` et non avec `\textit`.

Les tirets seront des tirets demi-cadratins (obtenus avec `--` ou `\textendash`) et non des tirets cadratins (obtenus avec `---` ou `\textemdash`).

Voici quelques pièges classiques :

On doit écrire	On saisit	Commentaire
1er, 1re	`1\ier`, `1\iere`	Et non 1ier, 1iere
2nd, 2nde	`2\up{nd}`, `2\up{nde}`	Second et seconde. Les avis divergent fortement sur les différences entre second et deuxième. Je ne prendrai pas part au débat !
2e, 3e …	`2\ieme`, `3\ieme`	Et non 2ième, 3ième, …
c.-à-d.	`\mbox{c.-à-d.}`	Abréviation de c'est-à-dire. Je sais que je me bats contre des moulins à vent, surtout chez les mathématiciens et les informaticiens, mais l'abréviation latine *i.e.* n'est normalement pas employée dans un texte en français (c'est une norme anglo-saxonne). Devant la marée, je suppose que cet anglicisme va devenir une règle admise en typographie française dans peu de temps !
cf.	`cf.`	Abréviation de *confer*.
etc.	`etc.`	Et jamais etc. . . (ou etc.. . . ou autre bizarrerie)
p. 34, p 34-36	`p.~34`, `p~34--36`	Pour faire référence à une ou plusieurs pages. On notera l'utilisation de l'espace insécable et le trait d'union (pas de tiret cadratin). On notera aussi que l'abréviation pp. n'existe pas.
a priori	`\emph{a priori}`	En fait, il y a bagarre de typographes mais soit c'est en italique sans accent (c'est du latin), soit c'est en police droite avec un a accentué (ce serait du français).

Chapitre 3

Mise en page finale

Plan du chapitre

Normalement ce chapitre n'a pas à être lu par les auteurs. Cela dit, je ne vais pas être en mesure d'interdire de le faire et, d'autre part, cela ne fera sans doute pas de mal que les auteurs voient ce qui se passe après leur création.

Dans ce chapitre, il faudra distinguer les sources provenant d'auteurs ayant composé leur texte avec un version récente de la classe et les autres ayant utilisé la toute première version de cette classe, à savoir celle de Michel Bovani. La nouvelle classe a essayé autant que faire ce peut de limiter le travail d'adaptation mais n'a pas pu tout automatiser non plus. Dans un premier temps, on laissera ce problème de côté.

1 Fontes commerciales

La différence fondamentale entre une compilation chez un auteur et une compilation de mise en page finale est que cette dernière utilise des fontes commerciales :

- les fontes Times (texte et mathématiques) *via* `mathtime` au lieu de `mathptmx` ;
- les fontes Block Berthold, FF-Din et Helvetica Rounded qui n'existent absolument pas en version libre, pas même tout le système nécessaire à LaTeX pour pouvoir s'en servir (fichiers `fd`, `vf`, `tfm`, `map`).

Il faut donc que le metteur en page dispose d'un système LaTeX permettant de travailler avec ces fontes. Pour faciliter l'installation, le répertoire `fontes` provenant de la décompression du fichier `zip` contient l'intégralité des fichiers gérant les fontes commerciales. On a fait en sorte de respecter la TDS (TeX Directory Structure) : une simple recopie intégrale du contenu du répertoire `fontes` dans une arborescence TeX doit suffire à obtenir un système fonctionnel[6].

6. En fait, c'est le cas si la recopie se fait dans une arborescence personnelle à une emplacement adéquat. Si la recopie se fait dans une arborescence locale ou générale, il faut mettre à jour la base de données et le système de map. On le fait respectivement avec la commande `\texhash` ou `\mktexlsr` (ce sont des synonymes) et avec les commandes suivantes :

- `\updmap --enable Map=block.map`
- `\updmap --enable Map=F-DIN/FF-DIN.map`
- `\updmap --enable Map=helvetica-rounded.map`
- `\updmap --enable Map=mathtime.map`

La fonte euro d'Adobe n'est pas une fonte commerciale au sens stricte mais elle n'est pas distribuée de façon libre donc n'existe pas d'emblée sur une distribution TEXLive ou MikTEX. Pour l'installer le plus simplement possible, il suffit de lancer le script `getnonfree-fonts -a` dans une fenêtre DOS sous Windows, ou dans un terminal sous Linux ou Mac.

En plus des fontes commerciales, il faut que les packages utilisés ne soient pas trop anciens par rapport à ceux utilisés lors de la conception de la classe. En l'occurrence, la distribution utilisée a d'abord été une TEXLive 2007 (celle utilisée par le système Ubuntu Hardy Heron) avec une version récente de `frenchb` (en l'occurrence sa version 2). La version 2.25 de la classe a été réalisée sur une TEXLive 2019 (installée sur une machine Mint 17.2 Rafaela).

Travailler avec des versions plus récentes de ces logiciels ne devraient normalement pas poser de problème mais il devient risqué de travailler avec des versions plus anciennes. L'installation des packages est donc, éventuellement, accessoire. Sinon, on les trouvera dans le répertoire `TDS` issu de la décompression du fichier `zip`. Là aussi, la TDS est respectée et un simple transfert de ces fichiers dans une arborescence TEX suffit.

⚠ ATTENTION

Contrairement à l'installation des fontes commerciales qui peut se faire dans n'importe quelle arborescence TEX, il est très dangereux de placer les fichiers du répertoire `TDS` dans une arborescence autre que l'arborescence personnelle. Il y aurait un risque d'écrasement de fichiers importants d'une distribution TEX.

La version 2.22 de la classe qui date de février 2019 a été obligé de s'adapter à une incompatibilité introduite depuis peu entre les extensions `cellspace` et `siunitx` qui utilisent tous les deux la lettre « S » dans les motifs de tableau. Il faut maintenant remplacer le « S » de `cellspace` par un « Z ».

Lorsque tout le système TEX est installé, pour pouvoir faire un test de compilation, il faudra modifier légèrement l'appel de classe du fichier maître de l'auteur. En effet, pour utiliser les fontes commerciales, il faudra ajouter l'option de classe `realfonts`. Au niveau des options de classe, il sera bon d'ajouter l'option `crop` également pour avoir les repères photographiques sur chaque page.

C'est une étape un peu délicate et il vaut mieux vérifier immédiatement que tout se passe bien en essayant de compiler le fichier de test (voir fichier `testIL.tex` dans le répertoire `tests`). On rappelle qu'une compilation complète se passe avec la chaîne `latex` → `dvips` → `ps2pdf` ou quelque chose d'équivalent mais qui commence par `latex` et non pas par `pdflatex`[7].

ou bien en utilisant des boutons d'utilitaires graphiques dépendant de la distribution utilisée.

7. On pourra regarder le fichier `Makefile` qui, même s'il n'est pas exploitable sur votre système, indique les programmes à utiliser et leurs syntaxes d'appel. En particulier l'option `-t a4` du programme `dvips`

2 Coupures

D'une part, les auteurs n'ont normalement pas à se soucier des problèmes de mise en page, d'autre part, les fontes utilisées par les auteurs ne sont pas les fontes définitives. Ces deux éléments expliquent que l'ouvrage brut donné par les auteurs comportent de nombreuses zones à reprendre (débordements, coupures malheureuses, veuves et orphelines caricaturales, boîtes incomplètes), sans compter évidemment toutes les petites fautes ortho-typographiques inévitables.

Corriger tout cela est le travail du metteur en page en sachant qu'un correcteur lira et relira le résultat pour détecter tous les endroits fautifs ou inélégants.

Le texte, hors fautes ortho-typographiques n'a normalement pas à être modifié. Il ne reste alors plus que les commandes permettant de poser des coupures à certains endroits, demander éventuellement une fonte de taille moindre (par exemple pour un tableau qui ne loge pas en largeur), élargir ou rétrécir certaines espaces, modifier légèrement la taille des images *via* l'argument optionnel de la commande `\includegraphics` ou demander un agrandissement de la hauteur de page avec la commande `\enlargethispage`. Pour cette dernière commande, il faut bien voir que la maquette demande un ouvrage avec les dernières lignes de chaque page alignées donc il ne faudra pas exagérer la quantité d'agrandissement. Typiquement, on ne devrait pas dépasser `1\baselineskip`.

On n'insistera pas plus sur toutes les commandes LaTeX permettant de couper (ou d'interdire de couper) les lignes et les pages. La maquette demande plusieurs constructions qui ne peuvent se couper sans intervention humaine. Il s'agit des structures :

- de type théorème ;
- de type exemple avec ou sans saut de paragraphe ;
- de type attention ;
- d'exercice d'introduction de chapitre ;
- de correction des exercices d'introduction de chapitre.

Pour ces structures, la classe propose deux commandes permettant de poser une coupure de page. Soit la coupure de page (et donc la commande) est posée juste avant une commande `\item` et c'est la commande `\Break` qu'il faudra utiliser (attention au B en capitale), soit c'est n'importe où ailleurs et c'est la commande `\BreakWithoutItem` qu'il faudra utiliser.

Il faut être conscient que pour toutes ces structures, une coupures ajoute un certain espacement vertical. Par exemple pour mettre un cartouche avec des points de suspensions. Il faudra en tenir compte dans le choix du placement de ces commandes.

On ne fera pas l'insulte de rappeler que la pose de coupure de page doit se faire par chapitre, en allant du début du chapitre vers la fin et une fois que *tous* les autres problèmes de mise en page auront été résolus !

Cette partie du manuel indique les différences entre l'ancienne classe et la nouvelle et la façon d'adapter les sources.

3 Syntaxe de l'ancienne classe

Il existe quelques différences tout à fait mineures, qui ont été répertoriées dans le tableau page 39.

Les syntaxes (et les représentations) diffèrent notablement en ce qui concerne les exercices et leurs corrigés. On distinguera :

- les exercices introductifs de début de chapitre ;
- la correction des exercices introductifs ;
- les QCM ;
- les exercices ;
- les corrections des QCM et des exercices.

Avec l'ancienne classe, les exercices introductifs était appelés avec la syntaxe suivante :

```
\begin{exointro}{Lycée Montaigne, Paris}
   ...
\end{exointro}
```

qui est exactement la syntaxe de la nouvelle classe.

Une première différence est que l'ancienne classe donnait la possibilité d'avoir des exercices dans la partie cours (autre part qu'en tout début de chapitre) avec l'environnement `exocours`. Cette possibilité est à la fois interdite par la nouvelle maquette – il est interdit d'avoir des exercices autre part qu'en tout début de chapitre ou que dans l'onglet Énoncés – et par la nouvelle classe – la rencontre de cet environnement donnera une erreur de compilation (environnement inconnu).

Une deuxième différence est que pour poser une coupure de page à l'intérieur d'un exercice introductif, il fallait employer la commande `\breakexo`. Cette commande a été remplacée par les deux commandes `\Break` et `\BreakWithoutItem`. Ces deux dernières commandes ont moins de restrictions que l'ancienne commande `\breakexo`. Cette dernière commande ne peut pas être employée avec la nouvelle classe : il y aura une erreur de compilation à propos d'une commande inexistante.

Dans l'ancienne classe, c'est la commande `\lastsol` qui commençait la correction d'un exercice type. Avec la nouvelle classe, c'est radicalement différent. On doit employer l'environnement `solintro`. De même, les corrections des autres exercices de chapitre étaient introduites par la commande `\sol`, il faudrait les remplacer également par des environnementsolintro en ayant eu soin au préalable de bien vérifier que les exercices étaient effectivement introductifs et non des exercices en plein milieu du chapitre.

La syntaxe est très largement différente pour les QCM et exercices de l'onglet « Énoncés » ainsi que pour leurs corrections. Des exemples complets sont donnés à la sous-section 3.3 page 15.

Tout d'abord, dans l'ancienne maquette, il n'y avait pas vraiment de distinction entre QCM et exercice, la syntaxe pour désigner un exercice était :

```
\exo(<difficulté>)[<titre>]{<Lycée>}{<temps>}
```

où la `<difficulté>` et le `<titre>` était des arguments optionnels. Avec la nouvelle classe, ces quatre données sont obligatoires pour les exercices, tandis que les QCM demandent uniquement (et obligatoirement) le titre et le temps. Un exercice se compose avec l'environnement `exo` et un QCM avec l'environnement `qcm` dont les syntaxes (pour rappel) sont :

```
\begin{exo}{<titre>}{<lycée>}{<difficulté>}{<temps>}
  ...
\end{exo}
\begin{qcm}{<titre>}{<temps>}
  ...
\end{qcm}
```

C'est encore pire pour les corrections. Dans l'ancienne classe, une correction était appelée avec la commande `\correxo` qui demandait un seul argument obligatoire : le lycée. Ce lycée devait bien sûr correspondre au lycée indiqué dans l'énoncé de l'exercice. Avec la nouvelle classe, les auteurs sont obligés d'indiquer la correction des QCM ou des exercices :

- avec l'environnement `corrige` qui ne demande aucun argument ;
- immédiatement après l'énoncé de l'exercice.

Cette façon de faire évite un grand nombre d'erreurs puisque les données n'ont pas a être répétées (le lycée, par exemple, reste le même entre l'énoncé et le corrigé de façon automatique) et on est sûr que tel corrigé correspond effectivement à tel exercice ou QCM. L'affichage effectif se fait avec la commande `\AfficheCorrige`, cette dernière devant être immédiatement précédée de la commande `\thumb` (ou `\onglet`).

Avec l'ancienne classe, les auteurs étaient encouragés à découper chaque chapitre en trois fichiers : un fichier pour le cours, un fichier pour les énoncés des exercices et un fichier pour les corrections. Mettre les sources en conformité avec la nouvelle classe représente donc un travail important de copier coller entre fichiers pour insérer chaque correction à la bonne place. Il faut également modifier la syntaxe d'appel, ce qui peut être en partie automatisée par n'importe quel éditeur (remplacement automatique) mais en partie seulement puisqu'il va falloir aussi placer les `\end{qcm}` ou les `\end{exo}` en fin de QCM ou d'exercice.

Ce travail est plutôt la façon conseillée de travailler mais la classe propose un autre voie qui demande un peu moins de travail mais un peu plus de précautions. On garde la structure à trois fichiers par chapitre, on garde la syntaxe d'appel mais il faut quand même changer les commandes en pseudo-environnements. Dans ce

cas :

- on modifie les commandes `\exo` en `\oldexo` avec la même syntaxe d'appel (sans oublier que la maquette impose l'ensemble des arguments[8] ;
- on ajoute une commande `\endoldexo` à la fin de l'exercice ;
- pour les QCM, on modifie la commande d'appel par `\begin{oldqcm}` et on place un `\end{oldqcm}` en fin de QCM ;
- on ajoute le paramètre de titre aux commandes `\correxo` (dans l'ancienne maquette, seul le lycée était utilisé) ;
- on place une commande `\endcorrexo` en fin de correction ;
- pour les corrections de QCM, la commande devient `\corrqcm` et doit avoir le titre comme seul paramètre et on doit placer un `\endcorrqcm` en fin de correction.

En fait, la nouvelle classe permet d'être plus sûr de soi en plaçant la commande `\startcorr` juste avant la première correction. Avec cette commande, les paramètres des commandes `\correxo` et `\corrqcm` ne servent plus à rien (mais il faut quand même en respecter le nombre), la classe se chargeant de mémoriser les titres et les lycées pour les afficher au niveau des corrections. C'est beaucoup plus sûr. Il reste juste à bien vérifier que les corrections sont strictement dans le même ordre que les énoncés.

8. Au besoin, il faudra demander à l'auteur de préciser un titre et une difficulté.

Cette partie propose quelques conseils pour réussir une bonne mise en page. Il s'agit d'une partie destinée à s'enrichir petit à petit au fur et à mesure que des problèmes particuliers seront rapportés. On peut donc voir cela comme une sorte de FAQ.

Comment traiter les formules mathématiques venant d'un document Word converti ?

Certains auteurs, peu coutumiers de LaTeX utilisent visiblement des convertisseurs automatiques pour traduire des formules de mathématiques de Word en LaTeX. Si quelqu'un connaît un outil vraiment propre, qu'il le crie sur tous les toits. Pour l'instant, ce que j'ai vu ne ressemble à rien : c'est illisible et il y a une sur-utilisation des accolades qui finissent par rendre les formules insécables (passe encore) mais totalement figées en ce qui concerne les espacements horizontaux, ce qui pose d'énormes problèmes de débordements ou de boîtes incomplètes. Pour ces problèmes particuliers, je n'ai pas de solution miracle : c'est pénible et long à corriger. Il faudra demander à l'éditeur à quel point il est prêt à sacrifier du temps pour obtenir un résultat propre.

Quelle doit être la structure d'un document maître ? `\input` ou `\include` ?

Voici le source d'un fichier maître de la collection[9] :

```
\errorcontextlines=10
\documentclass[latin1,crop,realfonts]{prepamath}
\usepackage{mathrsfs}
\graphicspath{{figures/}}
\input{commandes}
\title{ILTES}
\author{Danièle Rodrigue}

\includeonly{src/03-Cours,src/03-Enonces}

\begin{document}

\frontmatter
\include{src/avant-propos}
\include{src/methode}

\mainmatter
\include{src/01-Cours}
\include{src/01-Enonces}
\include{src/02-Cours}
\include{src/02-Enonces}
\include{src/03-Cours}
\include{src/03-Enonces}
\include{src/04-Cours}
```

9. En réalité le code source est légèrement repris.

```
\include{src/04-Enonces}
\include{src/05-Cours}
\include{src/05-Enonces}
\include{src/06-Cours}
\include{src/06-Enonces}
\include{src/07-Cours}
\include{src/07-Enonces}
\include{src/08-Cours}
\include{src/08-Enonces}
\include{src/09-Cours}
\include{src/09-Enonces}
\include{src/10-Cours}
\include{src/10-Enonces}
\include{src/bac-blanc}
\end{document}
```

Le `\errorcontextlines=10` permet d'avoir des messages d'erreurs plus détaillés. Si cela effraie plus que cela n'aide, on peut tout à fait supprimer cette ligne.

Lors d'un travail sur un chapitre, on utilise la possibilité `\includeonly` qui ne fonctionne qu'avec des fichiers appelés par `\include`. Sur l'exemple, on est en train de travailler sur le chapitre 3 complet. Lorsqu'on doit faire soixante-dix-huit compilations de suite, on apprécie ce gain de temps ! Avec des `\input`, il faudrait compiler tout l'ouvrage à chaque fois. On pourrait aussi commenter les lignes `\input` mais on perd toutes les références croisées du livres (références aux pages, aux sections d'un autre chapitre, table des matières, etc.) ainsi que le foliotage.

Comment repérer les petits détails typographiques à corriger ?

La plupart de ces fautes demandent un certain entraînement de l'œil (espace sécable malheureux par exemple entre « page » et le numéro de page) et un minimum de culture (espace avant une parenthèse gauche mais pas après, points de suspensions saisis avec `\ldots`, etc.). Pour ce dernier point, la référence de base est l'ouvrage *Les règles typographiques en usage à L'Imprimerie Nationale*.

On ne préconisera jamais assez la lecture (attentive) du fichier `log…` éventuellement de façon automatisée si votre environnement de travail le permet. Au minimum, il permet de repérer les erreurs de compilations bien sûr, mais aussi les débordements de boîtes trop importants, les boîtes incomplètes inélégantes, les références inexistantes, l'obligation de recompiler pour résoudre les références croisées (dans certains cas, deux compilations ne suffisent pas).

Lors de la compilation, les lettre accentuées provoquent des erreurs ou ont un rendu bizarre.

Il s'agit toujours d'une inadéquation entre le codage du document et l'option de classe indiquant ce codage.

Il s'agit d'un problème fréquent. Notamment lorsque l'auteur et le metteur en page n'utilise pas le même environnement de travail et pas le même codage. En effet, si le metteur en page commence à modifier un document en modifiant son codage, il va y avoir des problèmes.

Une autre source d'erreur est que l'ensemble des fichiers sources n'a pas le même codage. On pourrait s'en sortir avec des commandes LaTeX et un éditeur capable de s'adapter au vol à n'importe quel format. Il est cependant bien plus simple (et bien plus sûr), avant de commencer le travail, de tout convertir dans le même codage.

Le document compilé déborde de la page et n'est pas au format A4

La classe demande que le document soit au format d'impression finale et réalise un décalage pour loger les repères photographiques. Il faut alors forcer `dvips` à produire un document au format A4 en lui indiquant l'option `-t a4`. Par exemple, la chaîne de compilation telle qu'on peut la voir dans les `Makefile` donnés en exemple est :

```
        latex '\nonstopmode \input <fichier>'
        dvips -t a4 -O 0in,-2in <fichier>.dvi
        ps2pdf -dPDFSETTINGS=/prepress -dEmbedAllFonts=true\
-sPAPERSIZE=a4 <fichier>.ps <fichier>.pdf
```

Avec cette syntaxe, on évite toutes les mauvaises surprises.

Lors de la compilation, j'ai une erreur :

```
! Undefined control sequence.
<recently read> \c@lor@to@ps
```

La compilation a eu lieu avec `pdflatex`. Il faut absolument compiler avec la chaîne de production `latex`→`dvips`→`ps2pdf` ou, si ce n'est pas possible ou si vous ne savez pas faire, ajouter l'option de classe nopstricks. Dans ce dernier cas, la mise en page sera très altérée mais la compilation se fera correctement.

Chapitre 4

Bac, Brevet ou Devoir de synthèse

Les ouvrages destinés à la classe terminale peuvent avoir une partie « Bac Blanc ». La maquette diffère sur quelques points par rapport aux chapitres « normaux » et une syntaxe un peu particulière devra être utilisée bien que la classe essaie d'être la plus homogène possible.

Depuis la version v.2.0, la classe `prepamath` utilise dorénavant la commande `\devoirsynthese` qui permet de composer des chapitres destinés au bac blanc pour la classe de terminale (ou de première), au brevet blanc pour la classe de troisième ou bien de devoir de synthèse pour les autres classes.

Un chapitre « Bac blanc », « Brevet blanc » ou « Devoir de synthèse » sera indiqué par la commande `\devoirsynthese` et donc pas avec la commande `\chapter` habituelle. Contrairement à la commande `\chapter`, il y aura deux paramètres obligatoires qui indiquent le type de chapitre puis le titre du chapitre. Le type de chapitre est assez strict, il ne peut s'agir que de :

- Bac blanc ;
- Brevet blanc ;
- Devoir de synthèse.

Pour le bac blanc et le brevet blanc, le titre est également assez strict. Par exemple, pour un bac blanc ou un brevet blanc, cela donnera :

- Baccalauréat YYYY, académie de XXXX ;
- Brevet blanc YYYY.

Enfin, contrairement aux chapitres « normaux », le bac blanc et le brevet blanc n'auront pas les mêmes onglets. Les onglets par défaut sont « Sujet » et « Corrigé » avec les couleurs respectives associées orange et vert. La partie « corrigé » se fera sur fond coloré.

```
\devoirsynthese{Bac blanc}
               {Baccalauréat 2009, académie de Poitiers}
```

On peut remarquer que seules les pages paires font référence à l'onglet en cours. Sur les pages impaires, il n'y a plus d'onglet et l'en-tête spécifie le titre du chapitre. Il ne faut pas regarder ce manuel comme exemple : l'énoncé et la correction des pseudo-exercices ne tiennent chacun que sur une seule page. Ce ne sera évidemment jamais le cas en pratique.

Une fois de plus, il s'agit d'une décision éditoriale donc les auteurs ne sont pas autorisés à modifier les onglets sans accords explicite de l'éditeur. Dans ce manuel, la syntaxe utilisée a été :

```
\devoirsynthese[R/Intro,O/Sujet,G/Corrigés*]
               {Chapitres spéciaux}
               {Bac, Brevet ou Devoir de synthèse}
```

ce qui fait que les pages paires de cette première partie montre une en-tête avec le titre « Intro » alors que les deux autres parties auront des onglets et des en-têtes ayant respectivement les titres « Sujet » et « Corrigés », la page d'onglet « Corrigés » étant de plus sur fond vert (en raison de l'astérisque final).

Il n'y a pas de numéro de chapitre pour les parties devoir de synthèse, il ne doit pas non plus y avoir de numéro de section, sous-section, etc. Pour cela, la classe propose les macros `\sectionnn`, `\subsectionnn`, `\subsubsectionnn`, `\paragraphnn`, et `\subparagraphnn`. En principe, la maquette ne prévoit que `\sectionnn`, pour des niveaux plus fin, il faudrait en discuter avec l'éditeur. Il s'agit d'une commande utile lorsqu'un exercice (et sa correction donc) comporte plusieurs parties.

Le fait que les devoirs de synthèses soient des chapitres non numérotés rend assez illogique leur utilisation ailleurs qu'en toute fin d'ouvrage. Il conviendra donc, sauf obligation à discuter avec l'éditeur, de placer la totalité des chapitres *avant* les devoirs de synthèse. Seules les annexes peuvent être placées après.

Dans un devoir de synthèse, les exercices n'ont ni la même syntaxe ni la même présentation que dans les autres chapitres. En premier lieu, il n'y a pas de QCM et les exercices sont composés en utilisant l'environnement `exobb`, la correction qui suit immédiatement chaque exercice est saisie dans l'environnement `corrigebb` et l'affichage des corrigés est effectué grâce à la commande `\AfficheCorrigeBB`. Normalement la commande `\AfficheCorrigeBB` doit absolument être précédée de la commande `\thumb` (ou `\onglet`).

⚠ ATTENTION

Dans un devoir de synthèse, il ne faut pas utiliser les environnements `exo` et `qcm` ni l'environnement `corrige`, ni la commande `\AfficheCorrige`.

L'environnement `corrigebb` ainsi que la commande `\AfficheCorrigeBB` ne demandent aucun argument (comme pour les commandes similaires des chapitres normaux). En revanche, l'environnement `exobb` n'a pas la même syntaxe que l'environnement `exo`. Ici, il n'y a plus qu'à spécifier le titre, un sous-titre éventuel et le barème.

Les trois arguments sont obligatoires. Voici un exemple avec deux exercices, la saisie de leurs corrections et l'affichage. Le titre et le sous-titre des exercices proposés ci-dessous doivent être la forme habituelle. Ainsi, normalement, les sous-titres ne peuvent être que « Pour tous les candidats » ou « Candidats n'ayant pas suivi l'enseignement de Spécialité » ou « Candidats ayant suivi l'enseignement de Spécialité » et le titre doit indiquer le numéro d'exercice et le nombre de points attribués séparés par un tiret demi-cadratin (`--`). Le code :

```
\begin{exobb}{Exercice 1 -- 4 points}{Pour tous les candidats}
             {1}{10}
  Donner une fonction $f$ telle que $f(x+y)=f(x)+f(y)$
\end{exobb}
\begin{corrigebb}
  N'importe quelle fonction linéaire (par exemple).
\end{corrigebb}
\begin{exobb}{Exercice 2 -- 4 points}{Pour tous les candidats}
             {2}{15}
  De l'oxygène et de l'hydrogène ?
\end{exobb}
\begin{corrigebb}
  Ça fait boum.
\end{corrigebb}
\thumb
\AfficheCorrigeBB
```

donne le résultat :

SUJET

1 Exercice 1 – 4 points

★ — 10 min — Corrigé p. 57

Pour tous les candidats

Donner une fonction f telle que $f(x+y) = f(x) + f(y)$

2 Exercice 2 – 4 points

★★ — 15 min — Corrigé p. 57

Pour tous les candidats

De l'oxygène et de l'hydrogène ?

1 Exercice 1 – 4 points

Énoncé p. 56

Pour tous les candidats

N'importe quelle fonction linéaire (par exemple).

2 Exercice 2 – 4 points

Énoncé p. 56

Pour tous les candidats

Ça fait boum.

Annexe A

Chapitres annexes

Plan de l'annexe

1 Présentation

Depuis la version 1.7 de la classe, il est possible d'avoir une ou plusieurs annexes à l'ouvrage. Ces annexes seront obligatoirement situées en fin de document, après un devoir de synthèse (bac blanc ou brevet blanc).

On indique que les chapitres à venir sont des annexes en utilisant la commande LaTeX standard `\appendix`.

2 Rôle

Ces chapitres peuvent avoir la structure habituelle des autres chapitres de l'ouvrage. Dans ce cas, on emploie la macro habituelle `\chapter` et seule la numérotation change : elle se fait avec des lettres capitales au lieu de chiffres arabes (s'il n'y a qu'une seule annexe, il n'y a pas de numérotation du tout). Les annexes peuvent également avoir une structure beaucoup plus simple et, dans ce cas, doivent être spécifiées avec la commande `\simplechapter`.

3 Syntaxe

Dans le cas des annexes à structure habituelle, on va retrouver toutes les composantes des chapitres « normaux » : possibilités de modifier la liste des onglets avec l'argument optionnel de la commande `\chapter`, utilisation de sections avec affichage d'une table des matières de chapitre, etc.

Dans le cas d'une annexe simple introduite par la commande `\simplechapter`, il n'y a plus d'argument optionnel, plus d'onglet, plus de table des matières de chapitre. En fait, ce type d'annexe est prévu pour faire une ou deux pages. Typiquement, on peut réserver une telle annexe à l'affichage de la table de Mendeleïev, à un formulaire mathématique, etc.

Par exemple, l'annexe présente a été spécifiée avec le code :

```
\appendix
\chapter{Chapitres annexes}
```

Dans cet exemple, on n'a pas utilisé d'onglet ce qui fait que seul le premier de la liste par défaut est utilisé (et indiqué dans la table des matières de l'ouvrage). En pratique, il aurait fallu utiliser une commande `\simplechapter` comme pour l'annexe qui suit et qui est spécifiée avec le code :

```
\simplechapter{Table de multiplication}
\vspace*{\stretch{1}}
\begin{center}
  \begin{tabular}{|c|*{10}{c}|}
    \cline{2-11}
    \multicolumn{1}{c|}{$\times$}
      & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\\hline
    0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
    ...
    8 & 0 & 8 & 16& 24& 32& 40& 48& 56& 64& 72\\
    9 & 0 & 9 & 18& 27& 36& 45& 54& 63& 72& 81\\\hline
  \end{tabular}
\end{center}
\vspace*{\stretch{1}}
```

4 Index

Une annexe très particulière est celle destinée à afficher l'index de l'ouvrage. Elle a été introduite dans la version 2.19 de la classe. Un index se fabrique de façon tout à fait habituelle à l'aide de la commande `\index` et son affichage effectif se fait grâce à la commande `\printindex`.

Attention : la commande `\printindex` se charge d'appeler toutes les commandes nécessaires à l'affichage de l'index. En particulier, il ne faut pas ajouter de titre de chapitre avec la commande `\chapter` ni avec la commande `\simplechapter`. En revanche, un index fait obligatoirement partie des annexes. Si la commande `\printindex` est appelée en dehors des annexes, une commande `\appendix` sera automatiquement appelée et on rentrera bien dans une zone d'annexes.

Un exemple d'index est donné en dernière annexe de ce manuel.

Annexe B

Table de multiplications

×	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

Annexe C

Index

Annexe D

Liste des vidéos

Texte terminal
sur plusieurs lignes