




L'extension rdarithm

Quelques commandes spécifiques à l'arithmétique.

Ce package contient quelques commandes destinées à faciliter la composition de documents de travail pour les élèves (chapitre Arithmétique - collège).

Les couleurs suivantes sont prédéfinies par l'extension :

 arithmcol1  arithmcol2  arithmcol3

Chargement de l'extension

De la manière la plus classique qui soit, on charge l'extension à l'aide de la commande :

```
\usepackage{rdarithm}
```

Aucune option de chargement n'est proposée.

Cette extension est dépendante des packages **tikz.sty**, **xfp.sty** et **listofitems.sty**.

Si l'extension **rdcrep.sty** est également chargée, l'utilisation de boîtes destinées à recevoir la réponse de l'élève devient possible.

Parfois, plusieurs compilations successives seront nécessaires avant d'obtenir la mise en forme souhaitée.

Remarque : en cas de changements importants dans un document, en particulier s'il est long, il est souvent préférable de supprimer les fichiers auxiliaires avant une nouvelle compilation (ce qui évite de devoir compiler successivement un grand nombre de fois avant de voir les différents nœuds correctement recalculés).

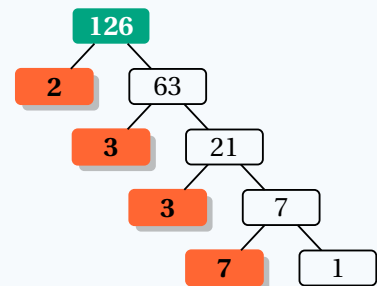
Modification des valeurs par défaut des clés

\setrdarithm{<paramètres>}

Cette commande permet de modifier les valeurs attribuées par défaut aux différentes clés de l'extension.

Il est également permis, à l'aide de cette commande, de créer des styles utilisateur pouvant être réinvestis à volonté.

```
% Nouveau style rdarithm
\setrdarithm{monstyle/.style={
  cell append style={rounded corners=2pt},
  prime append style={drop shadow, fill opacity=1},
  cell 1/.style={draw=none, fill=arithmcol2,
    text=white, font=\bfseries}
}
\DecompFPB[monstyle]{126}
```



Modulo

\xfpmod{<nombre entier>}{<nombre entier>}

Cette commande renvoie un nombre entier : le reste dans la division euclidienne du premier nombre passé en paramètre par le second.

Le reste dans la division euclidienne de 75 par 8 vaut `\xfpmod{75}{8}`.

Le reste dans la division euclidienne de 75 par 8 vaut 3.

Décomposition en produit de facteurs premiers

`\LOIFP`{*macro*}{*nombre entier*}

Cette commande renvoie, dans la commande donnée en premier paramètre, la liste, au format *list of items*, des facteurs premiers du nombre entier passé en second paramètre.

`\LOIFP{\maliste}{2310}`
`\showitems\maliste`

$\boxed{2}\boxed{3}\boxed{5}\boxed{7}\boxed{11}$

`\ListeFP`[*options*]{*nombre entier*}

Cette commande affiche la liste ordonnée des facteurs premiers qui entrent dans la décomposition du nombre entier qui lui est passé en paramètre.

Facteurs premiers pour 210 : `\ListeFP{210}`

Facteurs premiers pour 210 : 2, 3, 5, 7

`/rdarithm/separator`=*texte*

(initialement ,~)

Le séparateur entre chaque facteur premier peut être défini à l'aide de cette clé.

Facteurs premiers pour 210 :
`\ListeFP[separator=~~~]{210}`

Facteurs premiers pour 210 : 2 - 3 - 5 - 7

`\DecompFP`[*options*]{*nombre entier*}

Cette commande renvoie la décomposition en produit de facteurs premiers de l'entier qui lui est passé en paramètre (aucun mécanisme de contrôle n'est prévu pour s'assurer que le paramètre est bien un nombre entier).

On peut utiliser cette commande en mode mathématiques mais également hors du mode mathématiques (le résultat est composé à l'aide d'une commande `\ensuremath`).

`$126=\DecompFP{126}$`

$126 = 2 \times 3 \times 3 \times 7$

Cette commande ne prend en compte qu'une seule clé :

`/rdarithm/powers`=*none*|*right*|*bottom*

(par défaut *bottom*, initialement *none*)

Lorsque la valeur de cette clé n'est pas égale à *none*, les facteurs égaux, dans la décomposition en produit de facteurs premiers, sont regroupés et affichés à l'aide d'exposants.

`$126=\DecompFP{126}=\DecompFP[powers]{126}$`

$126 = 2 \times 3 \times 3 \times 7 = 2 \times 3^2 \times 7$

`\DecompFPA`[*options*]{*nombre entier*}

On utilisera cette commande pour afficher la décomposition d'un entier en produit de facteurs premiers sous la forme suivante :

Décomposons 1260 : `\DecompFPA{1260}`

Décomposons 1260 : $1260 = 2 \times \boxed{630}$
 $= 2 \times \boxed{2} \times \boxed{315}$
 $= 2 \times 2 \times \boxed{3} \times \boxed{105}$
 $= 2 \times 2 \times 3 \times \boxed{3} \times \boxed{35}$
 $= 2 \times 2 \times 3 \times 3 \times \boxed{5} \times 7$

Cette commande est composée en mode mathématiques (à l'aide d'un environnement **aligned**) et recourt à la bibliothèque **tikzmark** pour l'affichage des boîtes colorées et de traits. Si on ne souhaite aucune décoration, on emploiera la clé `/rdarithm/decorate` avec la valeur **false**.

`/rdarithm/decorate=true|false`

(par défaut **true**, initialement **true**)

Suivant sa valeur, cette clé permet d'activer ou de supprimer l'affichage des boîtes colorées et de traits.

```
\DecompFPA[decorate=false]{126}
```

$$\begin{aligned} 126 &= 2 \times 63 \\ &= 2 \times 3 \times 21 \\ &= 2 \times 3 \times 3 \times 7 \end{aligned}$$

`/rdarithm/colors=<liste de couleurs>`

(initialement `{arithmcol1, arithmcol2}`)

Cette clé permet de modifier les couleurs utilisées pour le coloriage des boîtes et le dessin des traits. L'extension utilise les couleurs dans l'ordre dans lequel elles ont été fournies et effectue une boucle sur les couleurs si le nombre d'étapes est supérieur au nombre de couleurs.

```
\DecompFPA[colors=red]{1260}%
\DecompFPA[colors={red, green,
blue}]{1260}
```

$$\begin{aligned} 1260 &= 2 \times 630 \\ &= 2 \times 2 \times 315 \\ &= 2 \times 2 \times 3 \times 105 \\ &= 2 \times 2 \times 3 \times 3 \times 35 \\ &= 2 \times 2 \times 3 \times 3 \times 5 \times 7 \end{aligned}$$

$$\begin{aligned} 1260 &= 2 \times 630 \\ &= 2 \times 2 \times 315 \\ &= 2 \times 2 \times 3 \times 105 \\ &= 2 \times 2 \times 3 \times 3 \times 35 \\ &= 2 \times 2 \times 3 \times 3 \times 5 \times 7 \end{aligned}$$

`/rdarithm/node opacity=<nombre entre 0 et 1>`

(initialement **0.25**)

L'opacité des boîtes colorées peut être modifiée à l'aide de cette clé.

```
\DecompFPA[node opacity=1]{126}
```

$$\begin{aligned} 126 &= 2 \times 63 \\ &= 2 \times 3 \times 21 \\ &= 2 \times 3 \times 3 \times 7 \end{aligned}$$

`/rdarithm/node style=<style TikZ>`

(initialement vide)

Le style des nœuds contenant les facteurs peut être altéré à l'aide de cette clé.

```
\DecompFPA[node opacity=1, node style={
text=white, font=\bfseries, inner sep
=3pt}]{126}
```

$$\begin{aligned} 126 &= 2 \times 63 \\ &= 2 \times 3 \times 21 \\ &= 2 \times 3 \times 3 \times 7 \end{aligned}$$

La clé `/rdarithm/powers`, vue précédemment, peut être utilisée pour afficher la forme condensée de la décomposition. Dans ce cas, les valeurs **right** ou **bottom** permettent de préciser la position de cet affichage.

```
\DecompFPA[powers=bottom]{126}%
\hspace*{1cm}%
\DecompFPA[powers=right]{126}
```

$$\begin{aligned} 126 &= 2 \times 63 \\ &= 2 \times 3 \times 21 \\ &= 2 \times 3 \times 3 \times 7 \\ &= 2 \times 3^2 \times 7 \end{aligned}$$

$$\begin{aligned} 126 &= 2 \times 63 \\ &= 2 \times 3 \times 21 \\ &= 2 \times 3 \times 3 \times 7 = 2 \times 3^2 \times 7 \end{aligned}$$

`/rdarithm/stop=<nombre entier>`

(initialement **0**)

L'utilisation de cette clé permet de n'afficher que la ou les première(s) étape(s) de la décomposition en produit de facteurs premiers.

```
\foreach \n in {1,...,4} {\DecompFPA[stop=\n]{2310}\hfill}
```

$$2310 = 2 \times 1155$$

$$2310 = 2 \times 1155 \\ = 2 \times 3 \times 385$$

$$2310 = 2 \times 1155 \\ = 2 \times 3 \times 385 \\ = 2 \times 3 \times 5 \times 77$$

$$2310 = 2 \times 1155 \\ = 2 \times 3 \times 385 \\ = 2 \times 3 \times 5 \times 77 \\ = 2 \times 3 \times 5 \times 7 \times 11$$

`/rdarithm/crep=true|false`

(par défaut true, initialement false)

Lorsque l'extension `rdcrep.sty` est chargée, cette clé permet d'englober les valeurs à afficher au sein d'une instruction `\tccrep`.

```
\setcrep{correction=false}
\DecompFPA[crep]{126}
```

$$126 = \quad \times \quad \\ = \quad \times \quad \times \quad \\ = \quad \times \quad \times \quad \times \quad$$

```
\setcrep{correction=true}
\DecompFPA[crep]{126}
```

$$126 = 2 \times 63 \\ = 2 \times 3 \times 21 \\ = 2 \times 3 \times 3 \times 7$$

`/rdarithm/crep width=<longueur>`

(initialement 3em)

La largeur des boîtes créées par la commande `\tccrep` est définie par la valeur de cette clé.

```
\DecompFPA[crep, crep width=2em]{126}
```

$$126 = \quad \times \quad \\ = \quad \times \quad \times \quad \\ = \quad \times \quad \times \quad \times \quad$$

`/rdarithm/crep style=<style tcb>`

(initialement voir ci-dessous)

Cette clé permet de définir le style `tcolorbox` passé en paramètre à la commande `\tccrep` produisant les boîtes. La valeur par défaut de ce style est donnée ci-contre.

```
seyes,
boxsep=1pt,
opacityback=0.6,
left=2pt, right=2pt
```

```
\DecompFPA[crep, crep style={}]{126}
```

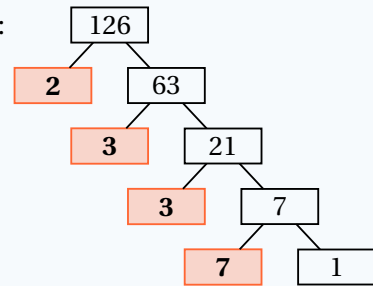
$$126 = \quad \times \quad \\ = \quad \times \quad \times \quad \\ = \quad \times \quad \times \quad \times \quad$$

`\DecompFPB[<options>]{<nombre entier>}`

On utilisera cette commande pour afficher la décomposition d'un entier en produit de facteurs premiers sous la forme suivante :

Décomposons 126 : `\DecompFPB{126}`

Décomposons 126 :



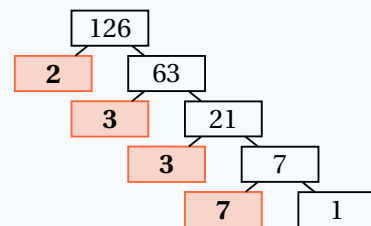
Avec cette commande, la décomposition d'un entier est encapsulée au sein d'un environnement **tikzpicture**. La construction de l'arbre affichant les différents facteurs est effectuée à l'aide d'une commande du type `\node ... child ... ;`.

`/rdarithm/level distance=<longueur>`

(initialement 8mm)

Cette clé permet de modifier la distance entre deux niveaux successifs de l'arbre.

`\DecompFPB[level distance=6mm]{126}`

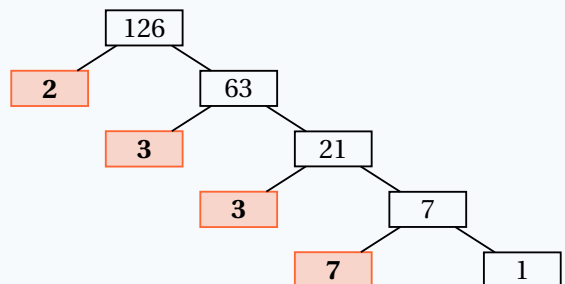


`/rdarithm/sibling distance=<longueur>`

(initialement 15mm)

Cette clé permet de modifier la distance entre deux facteurs sur un même niveau.

`\DecompFPB[sibling distance=25mm]{126}`

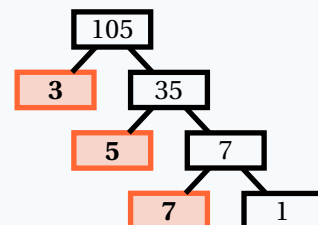


`/rdarithm/line width=<longueur>`

(initialement 0.7pt)

Largeur des lignes.

`\DecompFPB[line width=2pt]{105}`

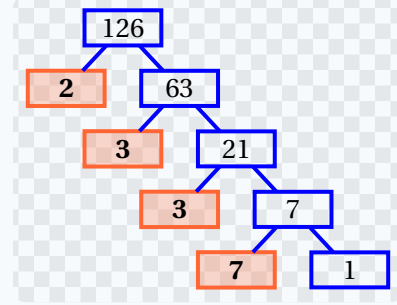


`/rdarithm/picture style=<style TikZ>`

(initialement vide)

Les paramètres globaux de l'environnement **tikzpicture** sous-jacent peuvent être modifiés à l'aide de cette clé.

```
\DecompFPB[line width=1.5pt,
picture style={
  draw=blue,
  show background rectangle,
  background rectangle/.style={
    rounded corners=1ex,
    pattern=checkerboard,
    pattern color=black!30,
    opacity=0.5
  }
}] {126}
```



`/rdarithm/cell style=<style TikZ>`

(initialement vide)

`/rdarithm/cell append style=<style TikZ>`

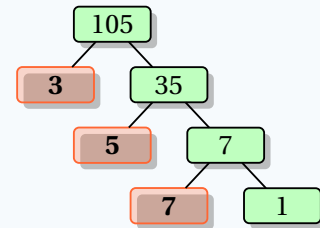
(initialement vide)

Au déclenchement de la commande `\DecompFPB`, tous les nœuds TikZ de l'arbre se voient affecter le style `rdarithm default cell style B` défini ci-contre.

```
\tikzset{
  rdarithm default cell style B/.style={
    draw, rectangle, minimum width=3em
  }
}
```

Il est permis de modifier ce style à l'aide des clés `/rdarithm/cell style` ou `/rdarithm/cell append style` (bien sûr, il serait aussi possible de modifier directement le style `rdarithm default cell style B` mais ce n'est pas recommandé).

```
\DecompFPB[cell style={
  rounded corners=2pt,
  fill=green!25,
  drop shadow}] {105}
```



`/rdarithm/prime style=<style TikZ>`

(initialement voir ci-dessous)

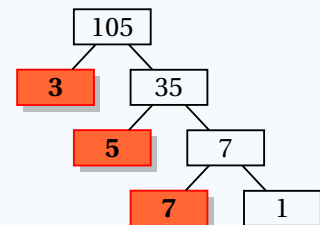
`/rdarithm/prime append style=<style TikZ>`

(initialement vide)

Tous les nœuds TikZ qui contiennent un facteur premier se voient attribuer un style supplémentaire défini ci-contre. Les clés `/rdarithm/prime style` ou `/rdarithm/prime append style` permettent de modifier directement ce style.

```
fill=arithmcoll,
fill opacity=0.25,
draw=arithmcoll,
text opacity=1,
node font=\bfseries
```

```
\DecompFPB[prime append style={
  draw=red,
  fill opacity=1,
  drop shadow}] {105}
```



`/rdarithm/cell 1=<style TikZ>`

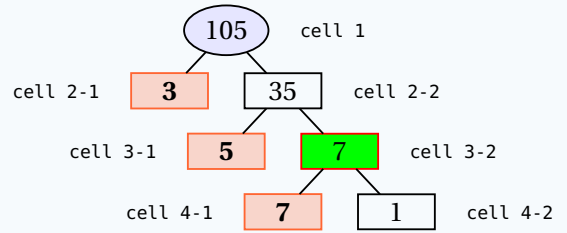
(initialement vide)

`/rdarithm/cell m-n=<style TikZ>`

(initialement vide)

En définissant ces styles TikZ, il est possible de modifier individuellement le style de chacun des nœuds de l'arbre. Le nœud qui contient le nombre initial est nommé `cell 1` tandis que les autres nœuds sont nommés sous la forme `cell m-n` où `m` désigne le numéro de rangée et `n` le numéro de colonne ($n = 1$ ou $n = 2$).

```
\DecompFPB[cell 1/.style={
  ellipse, fill=blue!10
},
cell 3-2/.style={
  fill=green,
  draw=red}]{105}
```

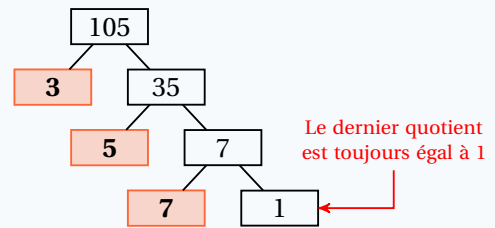


`/rdarithm/overlay`=*(code TikZ)*

(initialement vide)

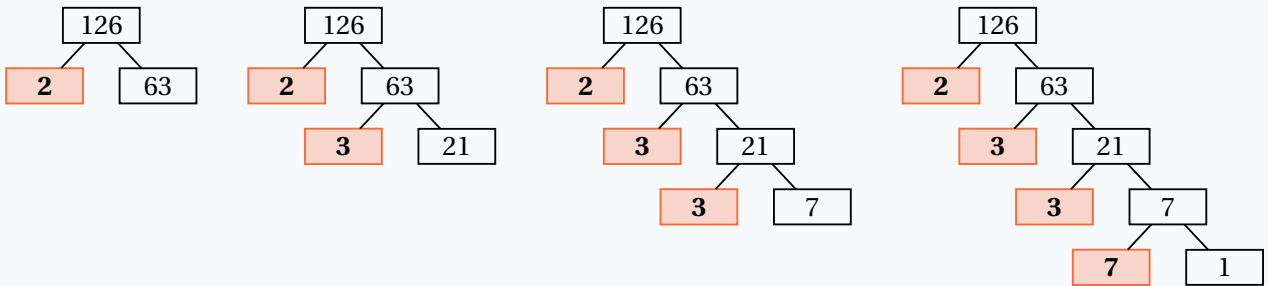
Des commandes TikZ arbitraires peuvent être ajoutées au code de l'environnement `tikzpicture` utilisé pour construire l'arbre. Le code graphique ajouté entre dans le calcul des dimensions de la boîte englobante de l'environnement final. Si on ne souhaite pas que ce soit le cas, on utilisera avec profit la clé `/tikz/overlay` (soit directement comme option des commandes de code TikZ soit par l'intermédiaire d'un environnement `\begin{scope}[overlay] ... \end{scope}`).

```
\DecompFPB[overlay={
  \draw[red, <-, >=stealth]
    (cell 4-2.east) -| ++(1cm,0.5cm)
    node[anchor=south,
    text width=2.5cm,
    align=center,
    node font=\scriptsize]
    {Le dernier quotient est
    toujours égal à 1} ;
}]{105}
```



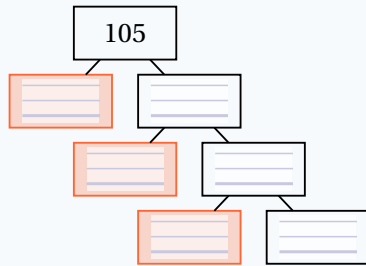
Comme avec la commande `\DecompFPA`, la clé `/rdarithm/stop` permet de préciser le numéro de l'étape où s'arrêter dans la décomposition.

```
\foreach \n in {1,...,4} {\DecompFPB[stop=\n]{126}\hfill}
```

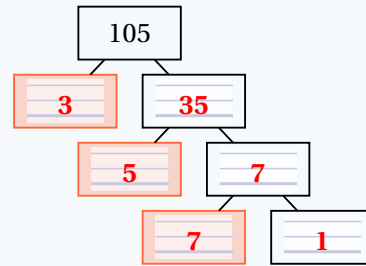


De la même façon, les clés `/rdarithm/crep`, `/rdarithm/crep width` et `/rdarithm/crep style` restent utilisables avec la commande `\DecompFPB`.

```
\setcrep{correction=false}
\DecompFPB[crep, level distance=9mm,
  sibling distance=17mm,
  cell style={inner sep=0pt,
    minimum height=7mm, minimum width=4em
  }]{105}
```



```
\setcrep{correction=true}
\DecompFPB[crep, level distance=9mm,
  sibling distance=17mm,
  cell style={inner sep=0pt,
    minimum height=7mm, minimum width=4em
  }]{105}
```



\DecompFPC[*<options>*]{*<nombre entier>*}

On utilisera cette commande pour afficher la décomposition d'un entier en produit de facteurs premiers sous la forme suivante :

```
Décomposons 126 : \DecompFPC{126}
```

Décomposons 126 :	126	2
	63	3
	21	3
	7	7
	1	

Avec cette commande, la décomposition d'un entier est encapsulée au sein d'un environnement **tikzpicture**. Le tableau est construit l'aide d'une commande de la forme **\matrix[...]** (**rdarithm**) {...} ;.

Les paramètres, par défaut, de la commande **\matrix** sont donnés ci-contre.

matrix of nodes,
nodes in empty cells,
inner sep=0pt, , outer sep=0pt

/rdarithm/column sep=*<longueur>*

(initialement 1mm)

Espacement inter colonnes.

```
\DecompFPC[column sep=1cm]{105}
```

105	3
35	5
7	7
1	

Avec cette commande, la clé **/rdarithm/level distance** reste utilisable et désigne l'espace inter rangées.

```
\DecompFPC[level distance=5mm]{105}
```

105	3
35	5
7	7
1	

La clé `/rdarithm/line width` modifie l'épaisseur de la ligne verticale.

```
\DecompFPC[line width=2pt]{105}
```

105		3
35		5
7		7
1		

`/rdarithm/label=none|left|above`

(par défaut above, initialement none)

Cette clé permet d'afficher directement, au sein de l'environnement, le nombre entier décomposé. Il peut être situé à gauche du tableau (`label=left`) ou bien au-dessus (`label=above`) de celui-ci.

La valeur `none` pour cette clé indique que le nombre ne doit pas être affiché.

```
\DecompFPC[label=left]{105}
```

105	105		3
	35		5
	7		7
	1		

```
\DecompFPC[label=above]{105}
```

	105	
105		3
35		5
7		7
1		

`/rdarithm/label extrasep=<longueur>`

(initialement 0mm)

Lorsque la clé `/rdarithm/label` a une valeur différente de `none`, la distance entre le nombre entier initial et le tableau est donnée par la somme de la valeur de la clé `/rdarithm/level distance` et de la valeur de la clé `/rdarithm/label extrasep`.

```
\DecompFPC[label,
  label extrasep=-3mm]{105}
```

	105	
105		3
35		5
7		7
1		

`/rdarithm/prefix label=<texte>`

(initialement vide)

`/rdarithm/suffix label=<texte>`

(initialement vide)

Lorsque la clé `/rdarithm/label` a une valeur différente de `none`, il est possible d'ajouter des chaînes de caractères (avant et après) au nombre entier affiché.

```
\DecompFPC[label,
  prefix label=On a :~,
  suffix label={$\,=\DecompFP
{105}$}]{105}
```

On a : 105 = 3 × 5 × 7

105		3
35		5
7		7
1		

`/rdarithm/label style=<style TikZ>`

(initialement vide)

Lorsqu'il est présent, le label est composé à l'intérieur d'un nœud dont le style est modifiable à l'aide de cette clé.

```
\DecompFPC[label=left,
  label style={fill=black!70,
    node font=\sffamily\bfseries,
    rounded corners=2pt,
    text=white},
  suffix label= :]{105}
```

105 :	105	3
	35	5
	7	7
	1	

Les clés `/rdarithm/cell style` et `/rdarithm/cell append style` permettent de modifier le style de chacune des cellules du tableau.

Au déclenchement de la commande `\DecompFPC`, toutes les cellules du tableau se voient affecter le style `rdarithm default cell style C` défini ci-contre.

```
\tikzset{
  rdarithm default cell style C/.style={
    rectangle, minimum width=3em,
    inner sep=3pt, outer sep=0pt
  }
}
```

```
\DecompFPC[cell style={
  fill=arithmcol2!20,
  rounded corners=2pt}]{105}
```

105	3
35	5
7	7
1	

Le style affecté aux cellules qui contiennent un facteur premier (colonne de droite) est modifiable à l'aide des clés `/rdarithm/prime style` et `/rdarithm/prime append style` (ce style s'applique après le style d'abord appliqué à toutes les cellules du tableau). La valeur par défaut de cette clé reste celle donnée [page 6](#). Mais, dans ce cas, il est légitime de s'interroger, en observant les tableaux précédents, sur, par exemple, l'absence de couleur dans les cellules contenant un facteur premier. Pour le comprendre, il faut se souvenir que le tableau est construit à l'aide d'une instruction du type `\matrix[matrix of nodes, ...] {...}` ;. Ainsi, si l'on souhaite agir directement sur le style des cellules, il conviendra de modifier le style `/tikz/nodes`.

```
\DecompFPC[prime style={
  nodes={
    text=arithmcol1,
    fill=arithmcol1!20
  }
}]{105}
```

105	3
35	5
7	7
1	

Le style individuel de chaque cellule peut être modifié à l'aide de la clé `/rdarithm/cell m-n` (où *m* désigne le numéro de rangée et *n* celui de colonne).

Les paramètres liés à l'environnement `tikzpicture` peuvent, quant à eux, être modifiés en ayant recours à la clé `/rdarithm/picture style`.

```
\DecompFPC[cell 4-1/.style={
    circle, minimum width=0pt,
    text=white, fill=arithmcol2,
    font=\bfseries},
    picture style={
        show background rectangle,
        background rectangle/.style={
            rounded corners=1ex,
            pattern=checkerboard,
            pattern color=black!30,
            opacity=0.5}},
    ]{105}
```

cell 1-1	105	3	cell 1-2
cell 2-1	35	5	cell 2-2
cell 3-1	7	7	cell 3-2
cell 4-1	1		cell 4-2

La clé `/rdarithm/overlay` permet d'ajouter des commandes TikZ à l'environnement `tikzpicture`.

```
\DecompFPC[level distance=4mm,
  overlay={
    \begin{pgfonlayer}{background}
      \fill[rounded corners=3pt, coulperso154!50]
        ([shift={({4pt,2pt})}]cell 1-2.north west)
        rectangle
        ([shift={{(-4pt,-2pt)}}]cell 4-2.south east) ;
    \end{pgfonlayer}
  }]{84}
```

84	2
42	2
21	3
7	7
1	

La clé `/rdarithm/stop` permet de préciser le numéro de l'étape où arrêter la décomposition.

```
\foreach \n in {1,...,4} {\DecompFPC[stop=\n]{126}\hfill}
```

126	2	126	2	126	2	126	2
		63	3	63	3	63	3
				21	3	21	3
						7	7
						1	

Les clés `/rdarithm/crep`, `/rdarithm/crep width` et `/rdarithm/crep style` restent autorisées avec la commande `\DecompFPC`.

```
\setcrep{correction=false}
\DecompFPC[crep,
  crep style={seyes},
  label, label extrasep=6pt]{105}
```

105

```
\setcrep{correction=true}
\DecompFPC[crep,
  crep style={seyes},
  label, label extrasep=6pt]{105}
```

105	
105	3
35	5
7	7
1	

`\LOIDiviseurs`{<macro>}{<nombre entier>}

Cette commande renvoie, dans la commande donnée en premier paramètre, la liste, au format *list of items*, des diviseurs du nombre entier passé en second paramètre.

```
\LOIDiviseurs{\maliste}{40}
\showitems\maliste
```

1 2 4 5 8 10 20 40

`\ListeDiviseurs`[<options>]{<nombre entier>}

Cette commande affiche la liste ordonnée des diviseurs du nombre entier qui lui est passé en paramètre.

```
Les diviseurs de 40 : \ListeDiviseurs{40}
```

Les diviseurs de 40 : 1, 2, 4, 5, 8, 10, 20, 40

Le séparateur entre chaque diviseur peut être défini à l'aide de la clé `/rdarithm/separator`.

```
Les diviseurs de 40 :
\ListeDiviseurs[separator=~~~]{40}
```

Les diviseurs de 40 : 1 - 2 - 4 - 5 - 8 - 10 - 20 - 40

`\TabDiviseurs`[<options>]{<nombre entier>}

Cette commande s'utilise exactement de la même façon que la commande `\DecompFPC` et accepte les mêmes clés à l'exception, bien entendu, des clés `/rdarithm/prime style` et `/rdarithm/prime append style`.

```
\TabDiviseurs[%
  label,
  level distance=5mm,
  cell 4-1/.style={
    circle, minimum width=0pt,
    text=white, fill=arithmcol2,
    font=\bfseries},
  picture style={
    show background rectangle,
    background rectangle/.style={
      rounded corners=1ex,
      pattern=checkerboard,
      pattern color=black!30,
      opacity=0.5}},
]{40}
```

		40	
cell 1-1	1	40	cell 1-2
cell 2-1	2	20	cell 2-2
cell 3-1	4	10	cell 3-2
cell 4-1	5	8	cell 4-2

```
\setcrep{correction=false}
\TabDiviseurs[crep,
  crep style={seyes},
  label, label extrasep=6pt]{40}
```

	40

```
\setcrep{correction=true}
\TabDiviseurs[crep,
  crep style={seyes},
  label, label extrasep=6pt]{40}
```

	40

`/rdarithm/cf=<nombre entier>`

(initialement 0)

Lorsqu'un nombre entier non nul est affecté à cette clé (*cf* = *common factors*), les styles définis par les clés `/rdarithm/cf style` ou `/rdarithm/gcd style` (voir ci-après) sont attribués aux cellules du tableau contenant les diviseurs communs à ce nombre et au nombre passé comme paramètre à la commande `\TabDiviseurs`.

```
\setrdarithm{dc/.style={
  label, label extrasep=6pt,
  prefix label=Diviseurs de~,
  level distance=5mm
}
}
\TabDiviseurs[dc, cf=75]{60}
\hspace*{2em}
\TabDiviseurs[dc, cf=60]{75}
```

Diviseurs de 60

1	60
2	30
3	20
4	15
5	12
6	10

Diviseurs de 75

1	75
3	25
5	15

`/rdarithm/cf style=<style TikZ>`

(initialement {fill=arithmcol3!50})

`/rdarithm/gcd style=<style TikZ>`

(initialement {fill=arithmcol3})

Lorsque la valeur de la clé `/rdarithm/cf` est différente de zéro, le style TikZ défini par la clé `/rdarithm/cf style` est appliqué à toutes les cellules du tableau qui contiennent un diviseur commun au nombre passé en paramètre à la commande `\TabDiviseurs` et au nombre défini par `/rdarithm/cf`.

Si ce diviseur commun est le plus grand, alors c'est le style défini par la clé `/rdarithm/gcd style` qui est appliqué.

```
% On définit préalablement les différents styles ci-dessous.
\tikzset{%
  % le handler /.cor style est défini par l'extension rdcrep.sty
  dcstyle/.cor style={fill=arithmcol3!50}{},
  pgcdstyle/.cor style={fill=arithmcol3}{}
}
\setrdarithm{dc/.style={
  label, label extrasep=6pt,
  prefix label=Diviseurs de~,
  cell append style={inner sep=0pt},
  cf style={dcstyle},
  gcd style={pgcdstyle},
  crep,
  crep style={seyes, opacityback=0.6},
}
}
```



```
\TabPGCD{420}{126}
```

```
\bigskip
```

```
Ainsi $\pgcd{420}{126}
=\DecompFP{\pgcd*{420}{126}}
=\pgcd*{420}{126}$
```

420 :	2	2	3		5	7
126 :	2		3	3		7

Ainsi PGCD (420;126) = $2 \times 3 \times 7 = 42$

Le tableau ainsi généré est construit l'aide d'une commande de la forme `\matrix[...]` (`rdarithm`) `{...}` ; intégrée dans un environnement `tikzpicture`.

Dès lors, de nombreuses clés, vues précédemment, restent parfaitement valides avec la commande `\TabPGCD` :

- `/rdarithm/line width` pour modifier l'épaisseur des lignes;
- `/rdarithm/level distance` pour modifier l'espacement entre les lignes;
- `/rdarithm/cell style` et `/rdarithm/cell append style` pour modifier le style commun à toutes les cellules du tableau;
- `/rdarithm/gcd style` pour modifier le style des cellules qui contiennent un facteur premier commun aux deux nombres;
- `/rdarithm/cell m-n` pour modifier le style d'une cellule particulière;
- `/rdarithm/picture style` et `/rdarithm/overlay` pour modifier l'apparence du graphique;
- `/rdarithm/crep`, `/rdarithm/crep width` et `/rdarithm/crep style` pour inclure des boîtes permettant à l'élève d'écrire sa réponse.

```
\TabPGCD[%
  forced width=12cm, line width=1pt, level distance=5mm,
  gcd style={fill=arithmcol3, fill opacity=0.25, text opacity=1},
  picture style={show background rectangle,
    background rectangle/.style={rounded corners=1ex,
      pattern=checkerboard, pattern color=black!30, opacity=0.5}},
  overlay={
    \node[red, node font=\ttfamily\scriptsize] at (cell 2-6) {vide} ;
  }
]{420}{126}
```

cell 1-1	cell 1-2	cell 1-3	cell 1-4	cell 1-5	cell 1-6	cell 1-7
420 :	2	2	3		5	7
126 :	2		3	3	vide	7
cell 2-1	cell 2-2	cell 2-3	cell 2-4	cell 2-5	cell 2-6	cell 2-7

Au déclenchement de la commande `\TabPGCD`, toutes les cellules du tableau se voient affecter le style `rdarithm default cell style D` défini ci-contre où `\rdarithm@leveldistance` contient la valeur fournie par la clé `/rdarithm/level distance`.

```
\tikzset{
  rdarithm default cell style D/.style={
    rectangle, minimum width=3em,
    inner sep=3pt, outer sep=0pt,
    align=center,
    minimum height=\rdarithm@leveldistance
  }
}
```

Le tableau construit par la commande `\TabPGCD` s'avère relativement « fragile » (ce n'est pas un tableau mais une matrice au sens TikZ). Il faut, en particulier, veiller à ce que la valeur attribuée à la clé `/rdarithm/level distance` soit plus grande que la hauteur maximale du contenu des cellules sans quoi le tableau généré prendra une forme totalement inappropriée.

```
\setcrep{correction=false}
\tikzset{%
  mygcdstyle/.cor style=%
    {fill=arithmcol3!50}{}}
}
\TabPGCD[%
  line width=1pt, level distance=1cm,
  gcd style={mygcdstyle}, crep,
  crep style={seyes, opacityback=0.6}
]{273}{105}
```

273 :				
105 :				

```
\setcrep{correction=true}
\tikzset{%
  mygcdstyle/.cor style=%
    {fill=arithmcol3!50}{}}
}
\TabPGCD[%
  line width=1pt, level distance=1cm,
  gcd style={mygcdstyle}, crep,
  crep style={seyes, opacityback=0.6}
]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

Des clés spécifiques à cette commande sont également disponibles.

/rdarithm/tab orientation=horizontal|vertical (par défaut horizontal, initialement horizontal)

Cette clé permet de modifier la disposition du tableau.

```
\TabPGCD[tab orientation=vertical]{273}{105}
```

273 :	105 :
3	3
	5
7	7
13	

/rdarithm/header style=<style TikZ> (initialement {fill=gray!50, node font=\bfseries})

Style attribué aux deux cellules qui contiennent l'entête du tableau.

/rdarithm/prefix header=<text> (initialement vide)

/rdarithm/suffix header=<text> (initialement ~:)

Il est possible d'ajouter des chaînes de caractères (avant et après) les nombres entiers affichés dans les cellules d'entête.

```
\TabPGCD[%
  level distance=1cm,
  header style={fill=green!25},
  prefix header=Décomposition de~,
  suffix header={~::~~}] {273}{105}
```

Décomposition de 273 :	3		7	13
Décomposition de 105 :	3	5	7	

/rdarithm/rules=none|all|horizontal|Horizontal|vertical|Vertical

par défaut all, initialement all

Cette clé permet de préciser la façon de tracer les lignes du tableau.

Lorsque :

- **/rdarithm/tab orientation** vaut **horizontal**, alors, avec **rules=Horizontal**, les lignes supérieures et inférieures du tableau voient leur épaisseur augmenter de 70 %;
- **/rdarithm/tab orientation** vaut **vertical**, alors, avec **rules=Vertical**, seule la ligne partageant le tableau en deux est tracée.


```
\TabPGCD[%
  rules=Horizontal]{273}{105}
\hspace{1em}et\hspace{1em}
\TabPGCD[%
  tab orientation=vertical,
  rules=Vertical]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

et

273 :	105 :
3	3
	5
7	7
13	

/rdarithm/rule color=⟨couleur⟩

(initialement black)

Couleur des lignes du tableau.

```
\TabPGCD[%
  rules=Horizontal,
  rule color=blue]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

Par défaut, le tableau produit par la commande `\TabPGCD` possède une largeur « naturelle », c'est-à-dire que la largeur de chaque cellule est essentiellement donnée par la valeur de la clé `/tikz/minimum width`. Quand la commande `\TabPGCD` est appelée, la valeur de cette clé est fixée à 3 em à travers le style `rdarithm default cell style D` et peut être modifiée en agissant, par exemple, sur la clé `/rdarithm/cell style`. Afin d'éviter des éventuels chevauchements de cellules, Il conviendra de veiller à ne pas affecter une valeur trop petite à la clé `/tikz/minimum width`.

Si l'utilisateur souhaite fixer la largeur totale du tableau, il peut recourir aux clés présentées ci-dessous.

/rdarithm/forced width=⟨longueur⟩

(initialement 0cm)

La valeur de cette clé (à condition qu'elle soit non nulle) indique la largeur totale du tableau.

Il faut, bien sûr, veiller à fournir une valeur suffisamment grande pour éviter le chevauchement des cellules.

```
\TabPGCD[%
  rules=Horizontal,
  forced width=\linewidth
]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

Dans ce cas là, la largeur des cellules est calculée par l'extension (utilisation automatique de la clé `/tikz/text width`) et il n'est ainsi plus tenu compte de la valeur fournie à la clé `/tikz/minimum width`.

De surcroît, l'éventuel espace horizontal (attribué à l'aide des clés `/tikz/inner sep` ou `/tikz/inner xsep`) ajouté autour du contenu des cellules n'est plus pris en compte mais est déterminé par la clé présentée ci-après.

/rdarithm/inner xsep=⟨longueur⟩

(initialement 3pt)

Quand la clé `/rdarithm/forced width` est utilisée avec une valeur non nulle, la largeur des cellules est calculée par l'extension. Ce calcul tient alors compte de la valeur attribuée à la clé `/rdarithm/inner xsep` qui désigne l'espacement horizontal ajouté avant et après le contenu de chaque cellule (cette clé n'a aucun effet sur l'espacement horizontal lorsque le tableau adopte une largeur naturelle).

```
\TabPGCD[%
  rules=Horizontal,
  forced width=\linewidth,
  inner xsep=15pt]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

/rdarithm/header width=⟨longueur⟩

(initialement 0cm)

Quand la clé `/rdarithm/forced width` est utilisée avec une valeur non nulle et lorsque le tableau est disposé horizontalement, la largeur de la colonne qui contient les cellules d'entête est automatiquement calculée par l'extension.

Fournir une valeur non nulle à cette clé, permet d'indiquer la valeur souhaitée pour la largeur de la colonne d'entête. Pour différentes raisons techniques (prise en compte de la largeur des bordures, de l'espacement horizontal intégré dans les cellules, ...), l'extension se contente d'approcher la valeur fournie par l'utilisateur.

```
\TabPGCD[%
  rules=Horizontal,
  forced width=\linewidth,
  header width=2cm]{273}{105}
```

273 :	3		7	13
105 :	3	5	7	

`\RowPGCD[<options>]{<nombre entier>}{<nombre entier>}`

Plutôt que d'opter pour la présentation en tableau proposée par la commande `\TabPGCD`, on peut préférer la mise en forme suivante :

```
\RowPGCD[cf style={inner sep=1.5pt},
  colors={arithmcol1!50,arithmcol2!50,
    arithmcol3!50}]{420}{126}
```

$$420 = 2 \times 2 \times 3 \times 5 \times 7$$

$$126 = 2 \times 3 \times 3 \times 7$$

Une telle mise en forme est produite à partir d'un environnement **aligned**. Les facteurs premiers communs aux deux nombres passés comme arguments à la commande `\RowPGCD` sont insérés dans des commandes du type `\tikzmarknode[<options>]{<...>}`. Cette dernière commande reçoit, en particulier, comme *<option>* le style défini par la clé `/rdarithm/cf style`.

`/rdarithm/link=none|line|bubble`

(par défaut bubble, initialement bubble)

Cette clé permet de préciser la façon dont sont reliés les facteurs premiers qui se correspondent.

```
\RowPGCD[cf style={
  inner sep=1.5pt,
  fill=green!30},
  link=none]{420}{126}
```

$$420 = 2 \times 2 \times 3 \times 5 \times 7$$

$$126 = 2 \times 3 \times 3 \times 7$$

```
\RowPGCD[cf style={
  inner sep=1.5pt,
  fill=arithmcol},
  link=line]{420}{126}
```

$$420 = 2 \times 2 \times 3 \times 5 \times 7$$

$$126 = 2 \times 3 \times 3 \times 7$$

```
\RowPGCD[%
  cf style={
    inner sep=1.5pt}
]{420}{126}
```

$$420 = 2 \times 2 \times 3 \times 5 \times 7$$

$$126 = 2 \times 3 \times 3 \times 7$$

Lorsque la valeur **bubble** est fournie à la clé `/rdarithm/link`, les nœuds créés par les commandes `\tikzmarknode` sous-jacentes se voient automatiquement attribuer le style `{circle, fill=arithmcol}` où **arithmcol** est une couleur particulière utilisable uniquement avec la commande `\RowPGCD`. La « valeur » de cette couleur est dynamique et change en fonction du couple de facteurs premiers considéré. Pour chaque couple de facteurs premiers, l'extension utilise successivement chaque couleur définie par la clé `/rdarithm/colors`. Ainsi, si l'on souhaite obtenir un remplissage identique pour toutes les bulles, il suffit de fournir, à la clé `/rdarithm/colors`, une liste de couleurs réduite à un seul élément.

```
\RowPGCD[cf style={inner sep=1.5pt},
  colors=yellow!50]{420}{126}
```

$$420 = 2 \times 2 \times 3 \times 5 \times 7$$

$$126 = 2 \times 3 \times 3 \times 7$$

En interne, les commandes `\tikzmarknode` et les « liens » entre les nœuds ne sont pas produits au même moment : il s'agit de plusieurs objets graphiques différents. Ainsi, il n'est pas recommandé de modifier l'opacité du remplissage (les chevauchements entre ces objets deviendraient visibles).

Lorsque la valeur **line** est fournie à la clé `/rdarithm/link`, les nœuds correspondants sont reliés entre eux par des segments dont l'épaisseur est donnée par la valeur de la clé `/rdarithm/line width`.

Comme le montre l'exemple ci-dessous (en zoomant sur les nœuds reliés en diagonale), des segments épais peuvent produire un rendu peu agréable à l'œil.

```
\RowPGCD[cf style={
  inner sep=1.5pt,
  fill=arithmcol},
link=line,
line width=2pt]{420}{126}
```

$$\begin{array}{l} 420 = 2 \times 2 \times 3 \times 5 \times 7 \\ 126 = 2 \times 3 \times 3 \times 7 \end{array}$$

Il est toutefois possible de remédier à cela en utilisant l'une et/ou l'autre des deux clés suivantes.

`/rdarithm/link style=<style TikZ>`

(initialement vide)

Lorsque la valeur `line` est fournie à la clé `/rdarithm/link`, cette clé permet de modifier le style utilisé pour tracer les segments.

```
\RowPGCD[cf style={
  inner sep=1.5pt, fill=arithmcol},
link=line,
line style={shorten >=-1.2pt,
  shorten <=-1.2pt},
line width=2pt]{420}{126}
```

$$\begin{array}{l} 420 = 2 \times 2 \times 3 \times 5 \times 7 \\ 126 = 2 \times 3 \times 3 \times 7 \end{array}$$

```
\RowPGCD[cf style={
  inner sep=1.5pt, fill=arithmcol},
link=line,
line style={arrows = {
  Latex[width=4pt, length=2pt]-
  Latex[width=4pt, length=2pt]}},
line width=1.5pt]{420}{126}
```

$$\begin{array}{l} 420 = 2 \times 2 \times 3 \times 5 \times 7 \\ 126 = 2 \times 3 \times 3 \times 7 \end{array}$$

`/rdarithm/to m=<style TikZ>`

(initialement vide)

Lorsque la valeur `line` est fournie à la clé `/rdarithm/link`, les nœuds sont reliés entre eux à l'aide d'une commande de la forme `\draw[...] (x) to[...] (y)` ;.

La clé `/rdarithm/to m` (où `m` désigne le numéro du couple de facteurs premiers, de la gauche vers la droite) permet d'affecter un style TikZ particulier directement au niveau de l'opérateur `to`.

```
\RowPGCD[cf style={
  inner sep=1.5pt, fill=arithmcol},
link=line, line width=2pt,
line style={shorten >=-1.2pt,
  shorten <=-1.2pt},
to 2/.style={out=-90, in=90},
to 3/.style={out=-90, in=90}}%
{420}{126}
```

$$\begin{array}{l} 420 = 2 \times 2 \times 3 \times 5 \times 7 \\ 126 = 2 \times 3 \times 3 \times 7 \end{array}$$

```
\RowPGCD[cf style={magnifying glass,
  draw=gray, line width=2pt,
  draw opacity=0.5,
  inner sep=1.5pt, fill=arithmcol},
link=line,
line style={arrows = {
  Latex[width=6pt, length=4pt, quick]-
  Latex[width=6pt, length=4pt, quick]}},
to 2/.style={out=-100, in=80},
to 3/.style={out=-100, in=80},
line width=1.5pt]{420}{126}
```

$$\begin{array}{l} 420 = 2 \times 2 \times 3 \times 5 \times 7 \\ 126 = 2 \times 3 \times 3 \times 7 \end{array}$$

Remarque : même si l'intérêt demeure discutable, les clés `/rdarithm/crep`, `/rdarithm/crep width` et `/rdarithm/crep style` restent utilisables avec la commande `\RowPGCD`.

De surcroît, en considérant que les facteurs premiers communs sont dans un « tableau » à deux lignes, la clé `/rdarithm/cell m-n` permet de modifier localement le style associé à l'un des facteurs communs.

```
% On définit préalablement les différents styles ci-dessous.
\setrdarithm{%
  style dynamique/.cor style={cf style={inner sep=1.5pt, fill=arithmcol}, link=line}%
  {cf style={inner sep=1.5pt}, link=none},
  mon style/.style={%
    cresp, cresp style={seyes, opacityback=0.6},
    line width=2pt, line style={shorten >=-1.2pt, shorten <=-1.2pt},
    colors={arithmcol1!50,arithmcol2!50,arithmcol3!50},
    cell 2-3/.cor style={fill=blue!30}{},
    style dynamique
  }
}
```

```
\setcresp{correction=false}
\RowPGCD[mon style]{420}{126}
```

$$420 = \frac{\quad}{\quad} \times \frac{\quad}{\quad} \times \frac{\quad}{\quad} \times \frac{\quad}{\quad} \times \frac{\quad}{\quad}$$

$$126 = \frac{\quad}{\quad} \times \frac{\quad}{\quad} \times \frac{\quad}{\quad} \times \frac{\quad}{\quad}$$

```
\setcresp{correction=true}
\RowPGCD[mon style]{420}{126}
```

$$420 = \boxed{2} \times \boxed{2} \times \boxed{3} \times \boxed{5} \times \boxed{7}$$

$$126 = \boxed{2} \times \boxed{3} \times \boxed{3} \times \boxed{7}$$

Fractions

`\fracirr{⟨nombre entier⟩}{⟨nombre entier⟩}`

`\dffracirr{⟨nombre entier⟩}{⟨nombre entier⟩}`

Ces commandes permettent d'écrire une fraction irréductible. La variante `\fracirr` utilise la commande `\frac` pour composer la fraction tandis que la variante `\dffracirr` utilise la commande `\dffrac`.

On a $\frac{15}{45} = \frac{15}{45}$
 mais aussi $\frac{63}{96} = \frac{63}{96}$

On a $\frac{15}{45} = \frac{1}{3}$ mais aussi $\frac{63}{96} = \frac{21}{32}$

`\DecompFrac[⟨options⟩]{⟨nombre entier⟩}{⟨nombre entier⟩}`

Cette commande décompose en produits de facteurs premiers le numérateur et le dénominateur d'une fraction (passés, respectivement, en premier et second arguments obligatoires à la commande). Les facteurs communs au numérateur et au dénominateur sont également mis en évidence.

`\DecompFrac{105}{273}`

$$\frac{3 \times 5 \times 7}{3 \times 7 \times 13}$$

La commande `\DecompFrac` s'apparente à une version légèrement modifiée de la commande `\RowPGCD`. Il est donc possible de se servir de la couleur particulière `arithmcol` afin de styliser les facteurs communs.

```
 $\frac{105}{273} =$ 
\DecompFrac[cf style={circle, inner sep=1.5pt,
  fill=arithmcol},
  colors={arithmcol1!50,arithmcol2!50,
  arithmcol3!50}]{105}{273}=
\dffracirr{105}{273}
```

$$\frac{105}{273} = \frac{\textcolor{red}{3} \times 5 \times \textcolor{teal}{7}}{\textcolor{red}{3} \times \textcolor{teal}{7} \times 13} = \frac{5}{13}$$

`/rdarithm/frac mode=frac|dfrac`

(par défaut dfrac, initialement dfrac)

Par défaut, avec `\DecompFrac`, la fraction est composée à l'aide d'une commande `\dfrac`. En utilisant la valeur `frac` avec la clé `/rdarithm/frac mode`, c'est la commande `\frac` qui sera utilisée par composer la fraction.

Pour plus de commodités, les deux styles ci-contre sont prédéfinis.

```
\setrdarithm{%  
  frac/.style={frac mode=frac},  
  dfrac/.style={frac mode=dfrac},  
}
```

On a $\frac{105}{273} = \frac{3 \times 5 \times 7}{3 \times 7 \times 13}$
et
 $\frac{105}{273} = \frac{3 \times 5 \times 7}{3 \times 7 \times 13}$

On a $\frac{105}{273} = \frac{3 \times 5 \times 7}{3 \times 7 \times 13}$
et $\frac{105}{273} = \frac{3 \times 5 \times 7}{3 \times 7 \times 13}$

- A -

`aligned` environnement, 3, 18
`arithmcol` couleur, 18, 20

- B -

`bottom` valeur, 3
`bubble` valeur, 18

- C -

`cell 1` clé, 6
`cell append style` clé, 6
`cell m-n` clé, 6
`cell style` clé, 6
`cf` clé, 13
`cf style` clé, 13
Clés
 `/rdarithm/`
 `cell 1`, 6
 `cell append style`, 6
 `cell m-n`, 6
 `cell style`, 6
 `cf`, 13
 `cf style`, 13
 `colors`, 3
 `column sep`, 8
 `crep`, 4
 `crep style`, 4
 `crep width`, 4
 `decorate`, 3
 `forced width`, 17
 `frac mode`, 21
 `gcd style`, 13
 `header style`, 16
 `header width`, 17
 `inner xsep`, 17
 `label`, 9
 `label extrasep`, 9
 `label style`, 9
 `level distance`, 5
 `line style`, 19
 `line width`, 5
 `link`, 18
 `node opacity`, 3
 `node style`, 3
 `overlay`, 7
 `picture style`, 5
 `powers`, 2
 `prefix header`, 16
 `prefix label`, 9
 `prime append style`, 6
 `prime style`, 6
 `rule color`, 17

`rules`, 16
 `separator`, 2
 `sibling distance`, 5
 `stop`, 3
 `suffix header`, 16
 `suffix label`, 9
 `tab orientation`, 16
 `to m`, 19

`colors` clé, 3
`column sep` clé, 8

Commandes

`\DecompFP`, 2
`\DecompFPA`, 2
`\DecompFPB`, 4
`\DecompFPC`, 8
`\DecompFrac`, 20
`\dfracirr`, 20
`\fracirr`, 20
`\ListeDiviseurs`, 12
`\ListeFP`, 2
`\LOIDiviseurs`, 12
`\LOIFP`, 2
`\PGCD`, 14
`\pgcd`, 14
`\pgcd*`, 14
`\PPCM`, 14
`\ppcm`, 14
`\ppcm*`, 14
`\RowPGCD`, 18
`\setrdarithm`, 1
`\TabDiviseurs`, 12
`\TabPGCD`, 14
`\tikzmarknode`, 18
`\xfpmod`, 1

Couleurs

`arithmcol`, 18, 20
`crep` clé, 4
`crep style` clé, 4
`crep width` clé, 4

- D -

`\DecompFP`, 2
`\DecompFPA`, 2
`\DecompFPB`, 4
`\DecompFPC`, 8
`\DecompFrac`, 20
`decorate` clé, 3
`\dfracirr`, 20

- E -

Environnements

`aligned`, 3, 18
 `tikzpicture`, 5, 7, 8, 10, 11, 15

- F -

`false` valeur, 3

forced width clé, 17
frac valeur, 21
frac mode clé, 21
\fracirr, 20

- G -

gcd style clé, 13

- H -

header style clé, 16
header width clé, 17
horizontal valeur, 16

- I -

inner xsep clé, 17

- L -

label clé, 9
label extrasep clé, 9
label style clé, 9
level distance clé, 5
line valeur, 18, 19
line style clé, 19
line width clé, 5
link clé, 18
\ListeDiviseurs, 12
\ListeFP, 2
\LOIDiviseurs, 12
\LOIFP, 2

- N -

node opacity clé, 3
node style clé, 3
none valeur, 2, 9

- O -

overlay clé, 7

- P -

\PGCD, 14
\pgcd, 14
\pgcd*, 14
picture style clé, 5
powers clé, 2
\PPCM, 14
\ppcm, 14
\ppcm*, 14
prefix header clé, 16
prefix label clé, 9
prime append style clé, 6
prime style clé, 6

- R -

right valeur, 3
\RowPGCD, 18
rule color clé, 17
rules clé, 16

- S -

separator clé, 2
\setrdarithm, 1
sibling distance clé, 5
stop clé, 3
suffix header clé, 16
suffix label clé, 9

- T -

tab orientation clé, 16
\TabDiviseurs, 12
\TabPGCD, 14
\tikzmarknode, 18
tikzpicture environnement, 5, 7, 8, 10, 11, 15
to m clé, 19

- V -

Valeurs

bottom, 3
bubble, 18
false, 3
frac, 21
horizontal, 16
line, 18, 19
none, 2, 9
right, 3
vertical, 16
vertical valeur, 16

- X -

\xfpmod, 1