Precedences

Gestion de précédences dans les listes de choix pour `Tty-Prompt#select`.

Deux modes d'utilisation sont possibles.

Le plus simple, par block :

```ruby
require 'precedences'

#
# Tty-prompt#select choices
#
choices = [
  {name:"Choix premier", value: :first},
  {name:"Deuxième choix", value: :second}
]

#
# Fichier où les précédences seront enregistrées
#
precfile = File.join(__dir__, '.precedences')

choix = precedencize(choices, precfile) do |q|
  q.question "Choisir la valeur :"
end

```

`choix` contiendra la valeur choisie, soit `:first` soit `:second`.

Le second mode d'utilisation fonctionne sans bloc, avec les valeurs par défaut :

```ruby
require 'precedences'

##
# On définit choices et prefile de la même façon
# puis…

choices = precedencize(choices, precfile)

choix = Q.select("Choisir parmi : ", choices)
#
```

```ruby
# Pour enregistrer cette précédence (ne pas l'oublier !)
#
set_precedence(choix)

# ...
```

On peut même considérer un troisième bloc qui utilise juste la méthode `set_precedence(choix, fichier)` quand le traitement du choix se fait ailleurs. Typiquement, on l'utilise quand la valeur peut être transmise directement au script, par exemple en ligne de commande.

```ruby
require 'precedences'

choix = ARGV[0] || demande_la_valeur_du_choix || return

set_precedence(choix, "<file/to/precedences_file>")
```

Valeurs possibles

Dans l'utilisation normale, l'attribut `:value` des choices doit obligatoirement être de type `String`, `Symbol` ou `Numeric`, mais avec l'option `precedences_per_index`, il est possible d'utiliser n'importe quelle valeur (note : l'ordre est alors mémorisé par index — ce qui signifie qu'il ne faut pas modifier la liste en cours de route).

Par exemple :

```ruby
require 'precedences'

#
# Des choix avec des valeurs spéciales
#
choices = [
  {name:"La classe Integer"   , value: Integer},
  {name:"La classe Array"     , value: Array},
  {name:"La classe Hash"      , value: Hash},
]

choix = precedencize(choices, file) do |q|
  q.question "Choisis une classe"
  q.precedences_per_index  # <=== option pour que ça passe
end
```

On peut aussi définir une autre clé que `:value` pour le tri, avec la propriété `per_other_key`

```ruby
require 'precedences'

#
# Des choix avec des valeurs spéciales
#
choices = [
  {name:"La classe Integer"   , pkey: :entier, value: Integer},
  {name:"La classe Array"     , pkey: :liste   ,value: Array},
  {name:"La classe Hash"      , pkey: :table , value: Hash},
]

choix = precedencize(choices, file) do |q|
  q.question "Choisis une classe"
  q.per_other_key :pkey  # <=== autre clé
end
```

Options possibles

En mode block, on peut définir plusieurs choses :

```ruby
require 'precedences'

choix = precedencize(choices, precfile) do |q|
  #
  # La question
  #
  q.question = "Ma question"
  # ou
  q.question "Ma question"

  #
  # Le nombre de menus affichés
  # (noter que par défaut, tous les menus sont affichés, contrairement
  #  à tty-prompt qui les limite toujours)
  #
  q.per_page 5
  # ou q.per_page = 5

  #
  # L'affichage ou non de l'aide (:never par défaut)
```

```ruby
  #
  q.show_help = :always
  # ou
  q.show_help :start

  #
  # Le message d'aide à afficher
  #
  q.help = "Mon message d'aide"
  # ou
  q.help "Message d'aide"

  #
  # La valeur sélectionnée par défaut (lorsqu'il faut passer
  # outre l'ordre de précédence par exemple)
  # Cette valeur peut être passée par…
  #
  # Par index 1-based explicite
  q.default = 4
  # ou q.default 4
  #
  # Par valeur du :value
  q.default = :second
  # ou q.default :second
  #
  # Par extrait du name (ou name exact)
  q.default = "premier"
  # ou q.default "premier"
  # => Sélectionnera le choix "Choix premier"

  # Ajout de menus à la fin, jamais classés
  q.add "Dernier menu", :last
  q.add_choice "Tout dernier", :very_last

  # Ajout de menus au début, jamais classés
  q.add_choice "Tout premier".bleu, :very_first, **{at_top:true}
  q.add("Premier", :first, {at_top: true})

end

```

Ajouter un menu “Renoncer”

```ruby
choix = precedencize(choices, file) do |q|
  q.add_choice_cancel(:up, {value: :cancel, name: "Renoncer"})
end
```

Si on doit utiliser les valeurs par défaut que sont :

- `:name` est “Cancel”
- `:value` est `nil`
- position est `:down` alors on peut faire simplement

```ruby
choix = precedencize(choices, file) do |q|
  q.add_choice_cancel
end
```

> 😀 *Noter qu’on peut en fait se servir de ce menu pour ajouter n’importe quel autre menu que “Renoncer”.*

Ajouter un menu quelconque

Utiliser les méthodes `#add` ou `#add_choice` (alias). Cf. ci-dessus Options possibles.