Année universitaire 2016-2017
Université de Caen Normandie

Rapport sur les nouveautés du menu édition

Théo Sarrazin
L2 Informatique

Table des matières

1 Selection de la ligne courante

Dans un premier temps, nous avons ajoutés une fonction permettant de selectionner la ligne où ce trouve le curseur. Pour cela, nous recuperons l'objet QTextCursor de notre class Editeur (héritant de QTextEdit) puis nous utilisons la méthode **select** de cet objet qui nous permet de selectionner du texte dans notre Editeur, cette méthode prend en paramètre une méthode de selection, QTextCursor.LineUnderCursor dans notre cas. La ligne ou ce trouve notre cuseur va donc être selectionner. Pour appliquer ces modifications, nous devons appliquer notre objet QTextCursor à notre Editeur, pour cela on utilise la méthode **setTextCursor** de l'objet Editeur et on lui passe en paramètre notre QTextCursor.

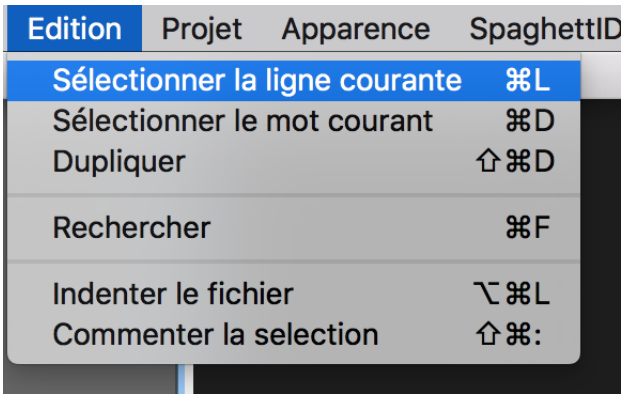

FIGURE 1 – Action du menu permettant la selection de la ligne courante

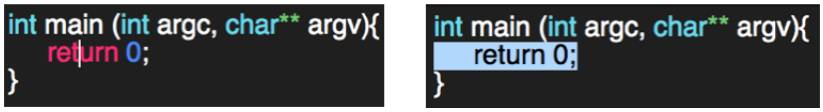

FIGURE 2 – Résultat de l'utilisation de la fonction selection de la ligne courante

2 Selection du mot courant

Pour l'ajout de la selection du mot courant, la démarche est exactement la même que pour la selection de la ligne courante, nous devons simplement changer la méthode de selection, passant de QTextCursor.LineUnderCursor à QTextCursor.WordUnderCursor, afin de ne plus sélectionner la ligne mais le mot présent au niveau du curseur.

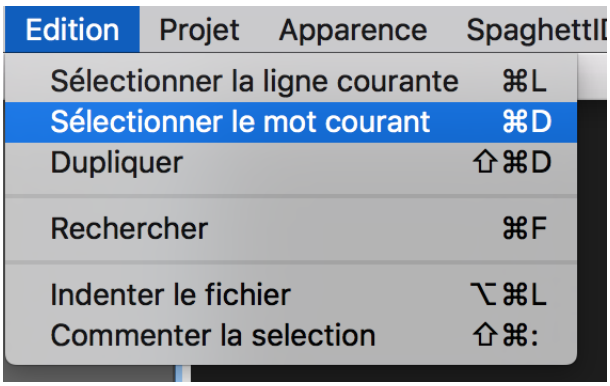

FIGURE 3 – Action du menu permettant la selection du mot courant

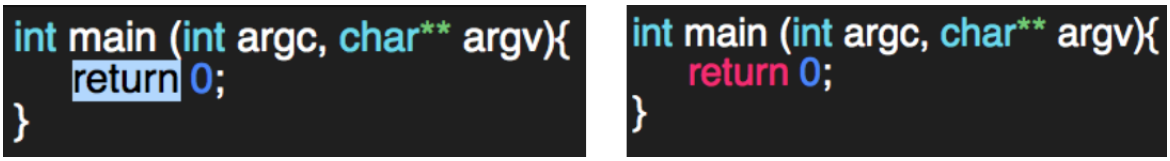

FIGURE 4 – Résultat de l'utilisation de la fonction selection du mot courant

3 Duplication

Pour l'ajout de la duplication du texte, nous avons choisi de différencier deux cas, le premier où rien n'est selectionné et le second où du texte est déjà selectionné. Dans le premier cas toutes la ligne est dupliquée et dans le second seulement la partie selectionnée est dupliquée.

Pour cela, nous recupperons une nouvelle fois le QTextCursor de notre Editeur, puis pour savoir dans quel cas nous sommes on utilise la méthode selectedText de l'objet QTextCursor. Ainsi si aucun text n'est selectionné nous selectionnons la ligne courante de le même façon que précédement de plus on assigne la valeur
n à la variable **return_** en effet si on duplique une ligne entière, on ajoute on retoure à la ligne entre la selection d'origine et la partie dupliquée. Puis on ajoute le texte dans notre objet Editeur grâce à la méthode **inserText** avec en paramètre la selection du QTextCursor (recuperrée grâce à la méthode **selectedText**) suivi de la variable **return_** elle même suivi de la selection du QTextCursor.

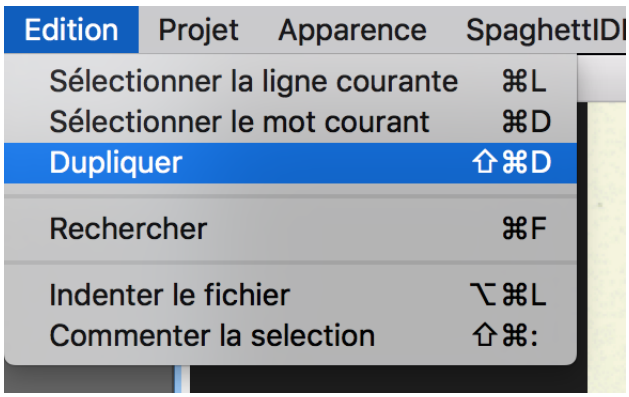

FIGURE 5 – Action du menu permettant de dupliquer

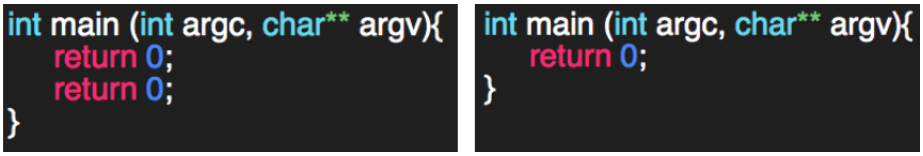

FIGURE 6 – Résultat de l'utilisation de la fonction permettant de dupliquer

4 Recherche

Pour la recherche dans le document, nous avons decidés d'ajouter une boite de dialogue permettant d'entrée le texte à rechercher. Pour cela nous avons créés une classe SearchDialog (héritant de QDialog), lors de l'affichage de cette boite de dialogue nous utilisons la méthode **exec**, qui rend impossible l'interaction avec la fenetre en arrière plan tant que la boite de dialogue est ouverte.

Cette boite de dialogue nous permer de taper le texte à rechercher, de choisir si on recherche en avant ou en arrière , mais aussi si on veut être sensible à la case.

FIGURE 7 – Boite de dialogue relative à la recherche

Pour la recherche on utilise la méthode **find** de notre objet Editeur. Cette méthode prend en paramètre le texte à rechercher, suivit de differents drapeaux. Dans notre cas nous utilisons le drapeau permettant d'executer la rechercher en arrière et le drapeau permettant de faire la recherche en étant sensible à la case (respectivement les drapeaux **QTextDocument.FindBackward** et **QTextDocument.FindCaseSensitively**)

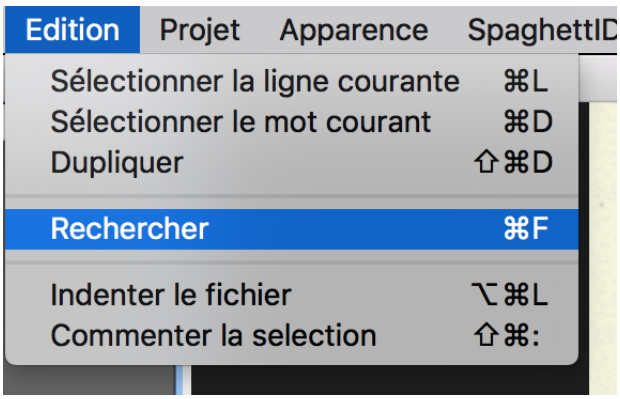

FIGURE 8 – Action du menu permettant de dupliquer

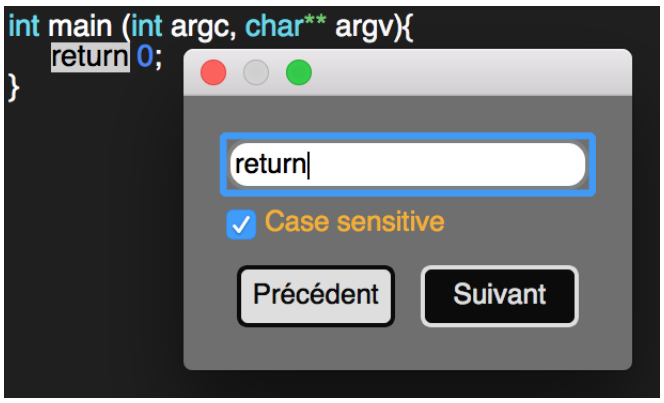

FIGURE 9 – Résultat de l'utilisation de la fonction permettant de dupliquer

5 Indentation du fichier

Pour l'indentation du document, nous allons changer son contenue (ajout/retrait de tabulation) nous devons donc stocker la position courante du curseur (grâce à la méthode **blockNumber** de l'objet QTextCursor). Par la suite on recupère le contenue du document grâce à la méthode **toPlainText** de l'objet Editeur. On créer une variable **indent_level**, qui contient le niveau courant d'indentation, puis on parcoure toutes les lignes de notre document, si le ligne contient le charactère "}", on retire 1 au niveau d'indentation puis on change la ligne pour ajouter au debut de cette dernière **indent_level** fois une tabulation, puis on ajoute 1 au niveau d'indentation si la ligne contient "{". Pour finir on defini le nouveau texte ainsi obtenu comme texte de notre document avec la méthode **setPlainText** et on replace le curseur au bon endroit.

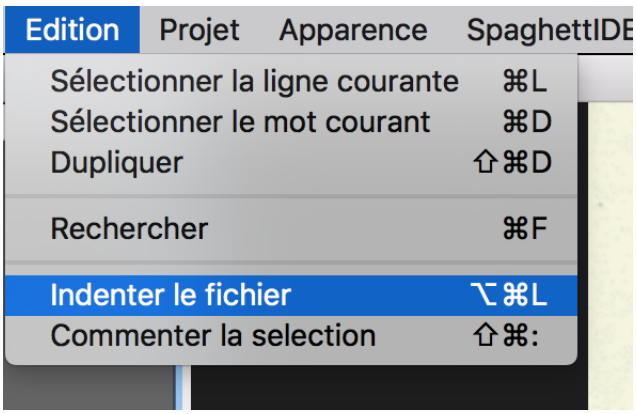

FIGURE 10 – Action du menu permettant d'indenter le fichier

FIGURE 11 – Résultat de l'utilisation de la fonction permettant d'indenter le fichier

6 Commenter le fichier

De la même façon que pour la duplication du texte, nous avons séparés cette action en deux cas, soit du texte est sélectionné soit rien n'est selectionné. Dans le cas ou du text est selectionné nous commenterons seulement à partir du debut de la selection. Si plusieurs lignes sont selectionnées elles seron evidement toutes commentées. Si il n'y a pas de texte selectionné, on commente la ligne courante.

Dans un premier temps, comme pour la duplication, si rien n'est selectionné on selectionne la ligne courante, puis on sauvegarde le text selectionné que l'on recupère grâce à la méthode **selectedText**, puis on supprime le texte selectionné avec la méthode **removeSelectedText**. Pour savoir si le texte est deja commenté, nous parcourons toutes les lignes, si une des lignes ne commence pas par "//" la selection est considérée comme non commenté. Par la suite nous parcourons chaque ligne du texte puis pour chaque ligne nous ajoutons/retirons les charactères "//" en fonction de si le texte et oui ou non déjà commenté.

Puis on ajoute le texte ainsi modifié à notre document en utilisant la méthode **insertText** (de l'objet QTextCursor).

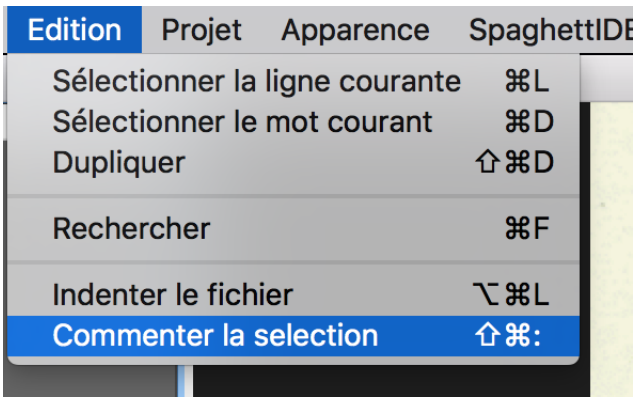

FIGURE 12 – Action du menu permettant de commenter le fichier

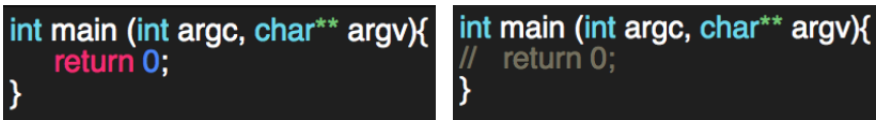

FIGURE 13 – Résultat de l'utilisation de la fonction permettant de commenter le fichier