Pourquoi un sous-programme :

En général, un programme est écrit pour résoudre un problème qui peut être très complexe. On commence donc habituellement par analyser le problème posé et on le divise en problèmes plus simples et donc plus faciles à résoudre.

Les sous-programmes permettent ainsi de décomposer un programme en plusieurs parties. Par ailleurs, un même sous-programme peut être utilisé à plusieurs reprises, ceci évitant de réécrire plusieurs fois le même code.

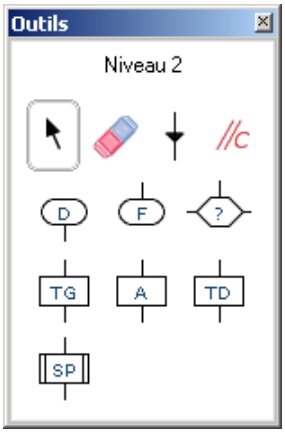
Nous avons vu dans les exercices précédents, faire le tour du terrain et revenir à la position initiale par exemple, que nous devions écrire plusieurs fois la même portion de code. Il est alors judicieux d'utiliser un sous-programme pour remplacer cette portion de code.

Aussi pour commencer nous allons réécrire le programme permettant au robot d'aller dans un coin en utilisant un sous-programme.

Créer un sous-programme :

Les sous-programmes sont disponibles à partir du niveau 2. Le niveau est affiché dans la palette d'outil. Si vous êtes au niveau 1, passez au niveau 2 en utilisant le menu **Configuration > Niveau**.
Avant de créer un sous-programme, vous devez avoir un nouveau programme vide. Si ce n'est pas le cas, fermez la fenêtre Programme et choisissez le menu **Fichier > Nouveau programme**.

Pour créer un sous-programme, choisissez le menu **Programmation > Nouveau sous-programme**. Le sous-programme est maintenant affiché dans la fenêtre programme à la place du programme principal.



Nom du sous-programme : chaque sous-programme est identifié par un nom que vous lui attribuez. Par défaut quand le sous-programme est crée il reçoit le nom **SousProgramme1**. Remplacer ce nom par **AvancerObstacle**

Utiliser un sous-programme :

Après avoir changé pour le niveau 2 (menu **Configuration > Niveau**), l'outil **SP** apparaît dans la fenêtre d'outils. Dans votre programme, vous pouvez utiliser le sous-programme comme une action élémentaire. Il suffit de saisir dans la boîte SP le nom du sous-programme que vous voulez utiliser.

Exercice

(1) Écrire un sous-programme AvancerObstacle qui fasse aller tout droit le robot jusqu'au mur.

(2) Réécrire le programme en utilisant le sous-programme.

Quand vous avez terminé votre organigramme, appelez le professeur pour le faire vérifier. Quand celui-ci est validé, lancez le programme (raccourcis « RobotProg réseau » sur le bureau) et saisissez l'organigramme de votre solution.

Si vous avez fini ...

On veut que le robot soit le plus efficace et minimise ses déplacements.

(3) Ecrire un sous-programme LongerMurADroite qui fasse aller tout droit le robot le long d'un mur à sa droite.

(4) Cherchez une autre solution pour le problème précédemment posé qui utilise le sous-programme LongerMurADroite.