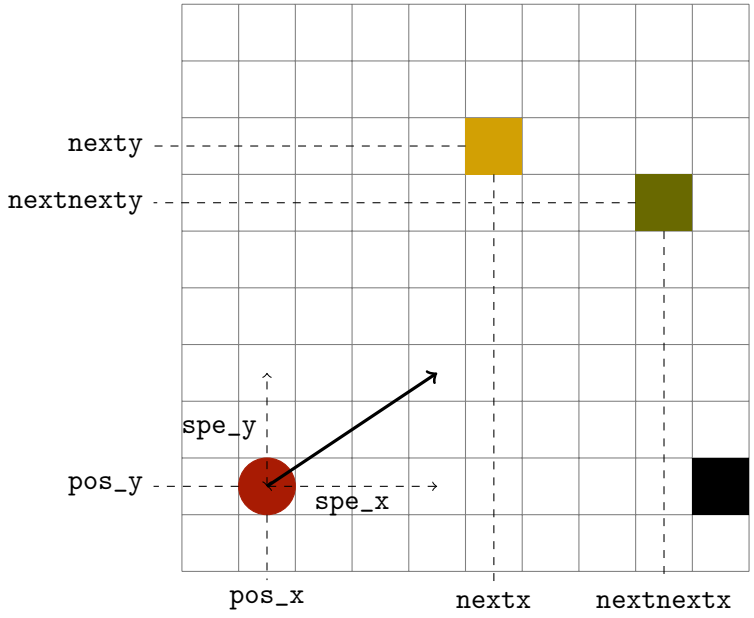
Le but de ce premier jeu de programmation est de programmer un robot qui contrôlera une voiture de course. La course se déroule dans un plateau carré de taille constante (`TAILLEPLATEAU`). La course est constituée d'une suite de portes (des "checkpoints") par lesquelles le robot doit faire passer la voiture. Ces portes, ainsi que la voiture sont repérées par leurs coordonnées entières dans le plateau. En plus de sa position sur le plateau, la voiture est aussi caractérisée par sa vitesse. Le robot ne peut contrôler directement ni la position de la voiture ni sa vitesse mais seulement son accélération (comme une vraie voiture). Finalement le robot n'est pas en mesure de voir toute la course mais seulement les deux prochaines portes à passer. Le comportement du robot dépend donc seulement :

- de la position de la voiture (deux coordonnées entières : `pos_x`, `pos_y`) ;
- de la vitesse de la voiture (deux coordonnées entières : `spe_x`, `spe_y`) ;
- de la position de la prochaine porte (deux coordonnées entières : `nextx`, `nexty`) ;
- de la position de la porte suivant la prochaine (deux coordonnées entières : `nextnextx`, `nextnexty`) ;

En fonction de ces 8 grandeurs, le robot en retourne 2 : `acc_x`, `acc_y` codant pour l'accélération de la voiture. La course a lieu au tour par tour de la manière suivante :

- Le robot reçoit les 8 grandeurs décrivant la voiture et retourne une accélération ;
- La vitesse de la voiture est mise à jour au moyen de l'accélération : `spe_x = spe_x + acc_x` et `spe_y = spe_y + acc_y` ;
- La position de la voiture est mise à jour au moyen de la nouvelle vitesse : `pos_x = pos_x + spe_x` et `pos_y = pos_y + spe_y` ;
- Si la position de la voiture est celle de la prochaine porte, l'objectif courant passe à la porte suivante.

Considérons l'exemple suivant donnant les grandeurs sur un schéma, le ● représente la position de la voiture, le ■ représente la position de la prochaine porte, ■ représente la position de la porte suivante et finalement ■ représente la position de la porte suivant la suivante.



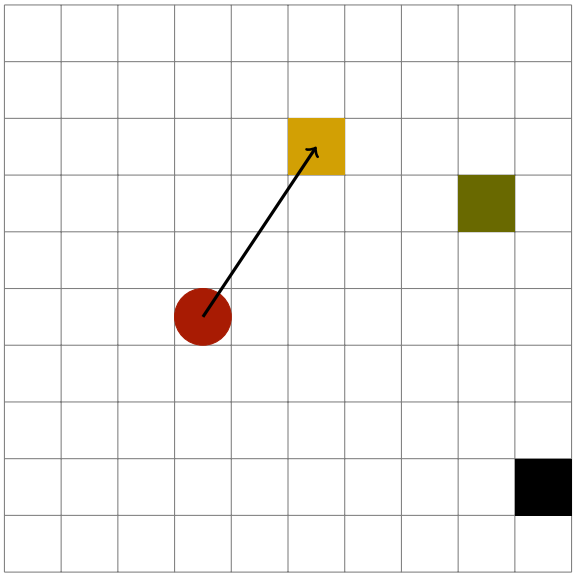
Les valeurs sont les suivantes :

- `pos_x` $= 1$, `pos_y` $= 1$;
- `spe_x` $= 3$, `spe_y` $= 2$;
- `nextx` $= 5$, `nexty` $= 7$;
- `nextnextx` $= 8$, `nextnexty` $= 6$;

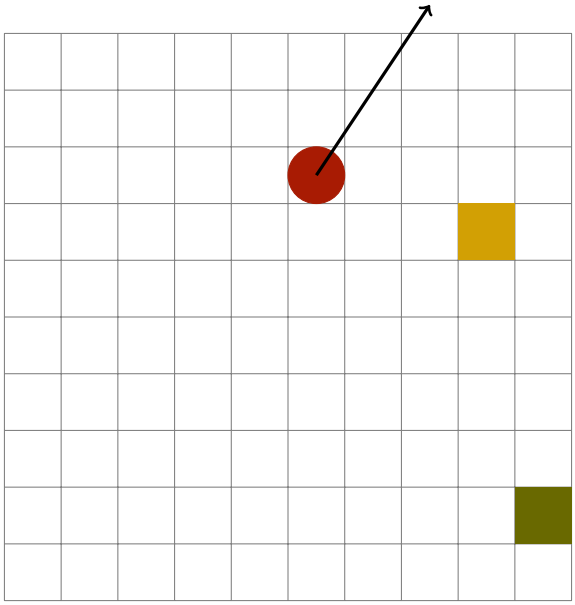
Admettons qu'après avoir donné ces valeurs au robot celui-ci décide que la voiture devrait accélérer de la manière suivante : $(-1, 1)$, alors les grandeurs sont mises à jour de la manière suivante :

- `spe_x` $= 2$ ($=3-1$), `spe_y` $= 3$ ($=2+1$) ;
- `pos_x` $= 3$ ($=1+2$), `pos_y` $= 4$ ($=1+3$) ;

Ainsi l'état de la course après une étape est la suivante :



À l'étape suivante le robot décide que la voiture a exactement la bonne vitesse et décide donc que la voiture ne devrait pas accélérer, la vitesse n'est donc pas changée mais la position de la voiture devient : `pos_x` $= 5$ ($=3+2$), `pos_y` $= 7$ ($=4+3$). Donnant alors l'état suivant :



Le robot doit maintenant exercer une accélération non nulle sur la voiture, au risque de sortir du plateau au prochain tour et d'être disqualifié.

Objectif du jeu. L'objectif est de définir un robot permettant de terminer la course le plus rapidement (le moins de tour) possible. La course n'est pas fixée à l'avance et le robot ne voit que les deux prochains objectifs.

Contraintes. La voiture pilotée par le robot n'est pas capable d'exercer une accélération aussi importante qu'elle le souhaite (une voiture ne peut pas passer de 0 à 1000km/h en 1ns). Ainsi à chaque tour l'accélération devra être bornée au sens suivant : si `acc_x`, `acc_y` code pour l'accélération voulue par le robot alors il faut que $|\texttt{acc\_x}|+|\texttt{acc\_y}| < \texttt{ACCMAX}$, où `ACCMAX` est une constante du programme, si le robot renvoie une grandeur pour l'accélération qui ne vérifie pas cette contrainte alors la voiture accélérera en utilisant l'accélération utilisée au dernier tour. La voiture *ne doit pas* sortir du plateau de jeu, ainsi si a un moment $(\texttt{pos\_x}, \texttt{pos\_y}) \notin [\![0, \texttt{TAILLEPLATEAU}[\![^2$ la voiture est retirée de la course.

Rendu. Vous trouverez une fichier `rendu_nom_prenom.c` sur la page web du cours. Ce fichier contient :

- les inclusions de librairie que vous pouvez utiliser ;
- des définitions de fonctions utiles pour le jeu ;
- des définitions des constantes du programme (dont la définition d'une course d'exemple) ;
- la définition d'une fonction `nom_prenom_comportement` que vous devez compléter avec le code de votre robot ;
- une fonction `main` fournie par l'enseignant qui s'occupe de simuler le comportement de votre robot sur une course d'exemple et qui vous affiche toutes les informations nécessaires au débuggage (pour changer la vitesse de simulation vous pouvez jouer avec la valeur de la constante `WAITTIME`).

Vous *devez* définir votre robot en complétant la fonction `nom_prenom_comportement`, vous avez le droit de définir des fonctions en plus mais ce doit être fait entre les deux commentaires limites. Chacune des fonctions que vous rajoutez entre les commentaires limites doit avoir un nom commençant par `[votre nom]_[votre prenom]_`. Vous *devez* changer le nom de la fonction `nom_prenom_comportement` en remplaçant `nom` par votre nom et `prenom` par votre prénom (il faut bien faire attention à le changer aussi dans la fonction `main`).

Évaluation des résultats. Quel robot de la classe termine la course en premier ?