Templates
Objectif 1 - Devant ou derrière ?

2. On va faire un template, car notre booléen est constant et on est égale à false donc on fait un template sur la fonction add_waypoint avec un booléen constant en paramètres et on appel add_waypoints avec en faisant add_waypoints<false> car le front était constant et égale à false.

Objectif 2 - Points génériques

1. On va faire une struct avec un template qui prend une taille et un type pour les points,
template<const int taille,typename type>, type sera entier, float et double.

3. On doit faire en sorte de mettre toutes les fonctions de Point2D et Point3D dans la classe pour que tout compile
en remplacant les Point2D et Point3D par Point que l'on a. Puis en dehors de la struct on va faire les 2 alias Point2D et Point3D
en les appelant avec leurs tailles et leurs type donc float. Sans oublier de faire les différents
constructeurs du Point2D Point3D.

4. Les erreurs se produisent que maintenant car on a des problèmes de types entre les différents
opérateurs maintenant après avoir tout implémenté.

5. On va faire plusieurs conditions dans les 2 constructeurs avec le static_assert et aussi dans
les fonctions y() et z(), le but est de verifier si la taille donnée au templates correspond bien
au nombre d'arguments. Donc on va vérifier pour le constructeur du Point2D si son Size est bien de 2,
et pour le constructeur de Point3D c'est pareil mais avec 3. Pour y() on doit vérifier si la taille est
plus grand ou égale à 2 car il faut vérifier si on est minimum en 2D et pour z() c'est pareil avec 3 pour
vérifier si on est en minimum 3D.

6. On va faire un template avec un "Args" et des ... pour avoir un variadic templates, qui veut dire qu'on peut
avoir plein d'arguments dans notre signature de fonctions ou constructeur sans préciser ce que l'on veut.
Donc on va ajouter avec l'arguments dans le template une référence universelle avec les &&, et mettres
dans notre array values tout les arguments données dans le constructeur car on veut y ajouter le x
le y le z etc.. en fonction de sa dimension. Mais on a un soucis de typage, les constructeurs des points
ne reconnaissent pas les valeurs et renvoit plein de problèmes de typages. Nous n'avons pas vu ca en cours,
j'ai fais mes petites recherches sur le net, demander au prof et j'ai pu trouver une solution.
Il faut changer le types de toutes les valeurs dans les constructeurs de Point en float, on ajoute le T first avec
T qui represente le typename qu'on ajoute en template et first le premier élement des arguments.
On peut maintenant vérifier si la taille des arguments coincide bien avec le Size en template, si oui pas d'assert
activé sinon si.