

C++ - 1 - J 2010

E 1 (, ,)

Écrire une classe C++ de nom `Rectangle` et qui permet :

- de créer un rectangle à partir de la donnée d'une largeur et d'une hauteur,
- d'obtenir son périmètre,
- d'obtenir sa surface,
- d'obtenir son affichage à l'écran.

On prendra soin de qualifier correctement les divers membres de la classe et d'être conforme au standard C++ pour l'affichage.

E 2 (, , ,)

1) Créez une classe `PaireQuelconque` permettant de stocker les adresses de deux objets.

2) Construisez en C++ (et en utilisant la construction précédente) de quoi permettre à un programmeur d'obtenir :

- une `PaireHeterogene` lorsqu'il tente de créer une paire avec deux objets de types différents
- une `PaireHomogene` lorsqu'il tente de créer une paire avec deux objets de types identiques
- que la création d'une paire avec un unique objet échoue.
- qu'il soit impossible de créer directement une `PaireQuelconque`, une `PaireHeterogene` ou une `PaireHomogene`

(L, , ,)

Puisque vous disposez de compétences reconnues, on vous charge de développer (en langage C++) un jeu (simple) dont le fonctionnement est le suivant :

- le jeu prend place sur une grille (NxN)
- sur cette grille sont placés :
 - le pion représentant le joueur
 - des pions représentant des cannibales gloutons
- au départ, les pions sont tous sur des cases différentes
- à chaque tour
 - les pions ne se déplacent que d'une case à la fois, dans l'une des huit cases voisines
 - le premier pion à se déplacer à chaque tour est celui du joueur
 - les pions gloutons se déplacent automatiquement en cherchant à se rapprocher le plus possible du pion du joueur
 - après les déplacements, le tour se termine par un règlement des conflits de position :
 - lorsque deux (ou plus) pions se trouvent sur une même case, ils disparaissent tous
 - les gloutons s'entredévorent et le nombre de goinfres disparus constitue le gain du score pour le joueur.
 - si le joueur fait partie des disparus la partie est terminée.

1. Modélisez le problème en faisant apparaître les concepts pertinents du problème, les relations entre les concepts et les abstractions qui vous semblent bonnes. La modélisation devra être dans le style UML,

2. Faites apparaître les attributs et actions pour chacune des classes séparément. Pensez à typer les divers éléments,
3. Implémentez votre modèle en C++, en respectant scrupuleusement le modèle. Dans ce premier temps, ne fournissez pas le code des méthodes (oubliez la partie algorithmique), mais qualifiez correctement les éléments de vos classes,
4. Fournissez le code des différentes méthodes (vous pouvez supposer l'existence d'une fonction de nom `dppd` prenant deux pions en paramètre et calculant la direction à prendre pour l'un des pions de sorte qu'