Nom de l'étudiant :
Durée du test : 60 à 75 minutes

Unity : Test n°6

Fondamentaux de l'informatique

Informations générales

Le test prend la forme d'une séance de travaux pratique noté. Vous pouvez utiliser le support de cours ainsi que les sites de référence sur le C++ (cppreference.com, cplusplus.com, reddit.com/r/cpp/, stackoverflow, ...).

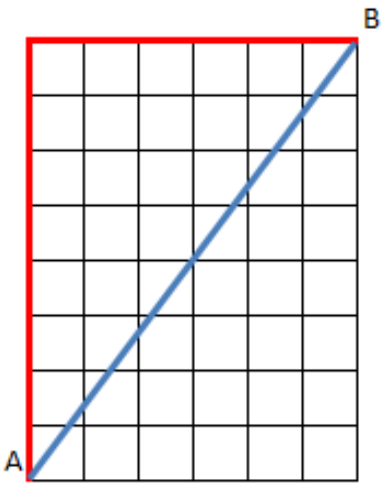
Plusieurs éléments sont déjà implémentés et prêts à l'emploi notamment une méthode `randomInt(int min, int max)` permettant de générer un nombre entier aléatoire et une structure `Point2D` disposant déjà des constructeurs et méthodes nécessaires. Les signatures des fonctions à implémenter sont déjà présentes (`mylib.hpp` et `mylib.cpp`). Vous êtes libres de créer de nouvelles fonctions au besoin.

Attention: Les fonctions dans `mylib.hpp` ne sont pas commentées ! Je serai attentif à la documentation de votre travail. (Inspirez-vous du fichier `Point2D.hpp`)

Pour récupérer le TP, je vous invite à *fork* le dépôt https://github.com/Nayboko/TP-GCC-2023 sur votre espace GitHub personnel. Au fur et à mesure de votre progression dans le TP, 1 Exercice = 1 Commit avec message au minimum.

Pour la notation, merci d'envoyer un lien d'invitation de votre dépôt à mon compte GitHub **@nayboko**.

En savoir plus



Distance Euclidienne (la diagonale) :
https://fr.wikipedia.org/wiki/Distance_entre_deux_points_sur_le_plan_cart%C3%A9sien

Distance de Manhattan (l'angle droit) :
https://fr.wikipedia.org/wiki/Distance_de_Manhattan

Nom de l'étudiant :
Durée du test : 60 à 75 minutes

Unity : Test n°6

Fondamentaux de l'informatique

Exercice 1 : Souvenir, souvenir...

Enoncé

Souvenez-vous de l'exercice Formes Pleines vu en TP. Et bien maintenant faisons les Formes Creuses.

Consigne

Créez la fonction automatisée permettant d'afficher dans le terminal les 2 formes creuses données. Puis, appelez la fonction afin d'afficher dans le terminal les deux formes creuses données en exemple.

Exemples

Forme pleine	Forme creuse	Forme creuse
* ** *** ****	********** * * **********	**** * * * * * * * * ****

Nom de l'étudiant :
Durée du test : 60 à 75 minutes

e·artsup
L'école de la passion créative

Unity : Test n°6

Fondamentaux de l'informatique

Exercice 2 : Une question de distance

Les formules

Distance Euclidienne

Dans le plan cartésien, les points sont définis à l'aide de leurs coordonnées cartésiennes.

Soient A et B deux points dans le plan cartésien, (x_A, y_A) les coordonnées du point A et (x_B, y_B) les coordonnées du point B. Alors la distance AB sur le plan vaut :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

Distance de Manhattan

Entre deux points *A* et *B*, de coordonnées respectives (X_A, Y_A) et (X_B, Y_B), la distance de Manhattan est définie par :

$$d(A, B) = |X_B - X_A| + |Y_B - Y_A|$$

Consignes

Soit un ensemble de points dont les positions sont générées aléatoirement dans un référentiel cartésien en deux dimensions. Chaque point peut avoir une position en x et en y comprise sur un intervalle [-15 ; 15].

- Implémenter la méthode `distanceEuclidienne` qui prend en paramètre `2 Point2D` ;
- Implémenter la méthode `distanceManhattan` qui prend en paramètre 2 `Point2D` ;
- Implémenter la méthode `plusProcheVoisin` qui prend en paramètre un tableau de `Point2D` non vide et un `Point2D` **P** choisi aléatoirement dans le tableau ainsi qu'un entier strictement positif **dist** correspondant à une distance maximale :
    - Utilisez la méthode `randomInt` pour générer aléatoirement des positions pour la liste de `Point2D` ;
    - Pour chaque `Point2D` contenu dans le tableau, si la distance euclidienne entre celui-ci et **P** est inférieure ou égale à la distance **dist**, ajouter le point courant dans un tableau `ppv` ;
    - Afficher la liste des plus proches voisins de **P** en fin de programme.

Nom de l'étudiant :
Durée du test : 60 à 75 minutes

e·artsup
L'école de la passion créative

Unity : Test n°6

Fondamentaux de l'informatique

Exercice 3 : Le jeu du plus ou du moins

Les règles

Le maître du jeu choisi :

- Un nombre entier compris entre 1 et 100 : c'est le nombre mystère. Il y a donc 100 nombres possibles ;
- Un nombre entier compris entre 5 et 15 : c'est le nombre de tentatives possibles donné au joueur.

Le joueur propose un nombre: si le nombre mystère est plus grand que la proposition, le maître du jeu répond : "plus", sinon, il répond "moins".

Le but du jeu est de deviner le nombre mystère en un minimum de coups sans dépasser le nombre de tentatives possibles.

Les points d'attention :

- Utilisez la méthode `randomInt`(int min, int max) disponible ;
- À chaque étape :
  - Inviter le joueur à entrer un nombre ;
  - Affichez le nombre de tentatives restantes.
- En fin de programme :
  - Affichez le nombre de tentatives une fois la condition de victoire remplie ;
  - Affichez un message de victoire ou de défaite en fonction de l'état du jeu.
- Bonus ;
  - Si le joueur rentre un nombre qui ne respecte pas les règles du jeu, l'inviter à nouveau à saisir un nombre sans compter la tentative ;
  - Permettre au joueur de relancer le jeu sans relancer le programme.

Consigne

Implémentez ce jeu en respectant les règles et les points d'attention.