robot.cpp

```cpp
//-------------------------------------------------------
//
// Fichier        : main.cpp
// Auteur(s)      : (C) Mariaux Ewan & Nicolas Sonnard
// Date           : 2020-12-2022
// But            :
//
// Modifications  :
// Remarque(s)    : Le soft a pas été terminé il se compile pas
//                  et il manque du code pour qu'il fonctionne
//
//-------------------------------------------------------

#include <random>
#include "robot.h"
#include "annex.h"


using namespace std;

//déclaration des deux variables static à 0
int Robot::nbreRobots = 0;
int Robot::prochainId = 0;



ostream& operator <<(ostream& os, const Robot& robot) {
    // affichage id (X,Y)
    return os << robot.id << "(" << robot.posX << "," << robot.posY << ")"<< endl;
}



Robot::Robot() : id(Robot::prochainId){
    //Incrément le prochain Id et le nombre de robots
    ++prochainId;
    ++nbreRobots;

    this->posX = 1;
    this->posY = 1;

}

Robot::Robot(Terrain& terrain) : id(Robot::prochainId){
    //Incrément le prochain Id et le nombre de robots
    ++prochainId;
    ++nbreRobots;

    //positionnement aléatoire du robot dans le terrain
    do{
        this->posX = nbrAleatoire(1, terrain.getLargeur() - 1);
        this->posY = nbrAleatoire(1, terrain.getHauteur() - 1);
        //vérifie que l'emplacement est libre
    }while(terrain.estLibre(posX, posY));

    //positione le robot dans le terrain
    terrain.setPositionRobot((char)this->id, this->posX, this->posY);

}

Robot::Robot(int id,int posX, int posY) : id(id){
    //Incrément le prochain Id et le nombre de robots
    ++prochainId;
    ++nbreRobots;

    this->posX = posX;
    this->posY = posY;
}
```

```cpp
//Constructeur par copie
Robot::Robot(const Robot& robot) : id(robot.getId()){
    //compie la position en x et y du robot
    this->posX = robot.posX;
    this->posY = robot.posY;
}

//Opérateur d'affectation
Robot& Robot::operator= (const Robot& robot){

    //vérifie que l'on copie pas le robot avec lui meme
    if (this != &robot) {
        //copie la position x et y du robot
        this->posX     = robot.posX;
        this->posY     = robot.posY;
    }

    return *this;
}


void Robot::deplacer(Game game){

    /*
    direction = 1 → la tondeuse monte
    direction = 2 → la tondeuse va vers la droite
    direction = 3 → la tondeuse descend
    direction = 4 → la tondeuse va vers la gauche
     */
    //constant pour le min et max du déplacement
    const int DEPLACEMENT_MIN = 1,
                DEPLACEMENT_MAX = 4;

    //choisi la direction aléatoirement
    int direction = nbrAleatoire(DEPLACEMENT_MIN,DEPLACEMENT_MAX);

    //stock la position du robot afin de faire le déplacement.
    int tmpPosY = this->posY;
    int tmpPosX = this->posX;
    do {
        //positionne le robot dans l'espace
        switch (direction) {
            case 1:
                --tmpPosY;
                break;
            case 2:
                ++tmpPosX;
                break;
            case 3:
                ++tmpPosY;
                break;
            case 4:
                --tmpPosX;
            default:
                return;
        }
        //check que le déplacement est dans les limites du terrain
    }while(tmpPosX > 0 and tmpPosX < game.getLargeur() and tmpPosY > 0 and tmpPosY < game.getHauteur());



    //affecte la nouvelle position au robot
    this->posX = tmpPosX;
    this->posY = tmpPosY;


}


int Robot::getId() const {
    //retourne l'id du robot
    return this->id;
```

```
}
int Robot::getPosX() const{
    //retourne la position X du robot
    return this->posX;
}
int Robot::getPosY() const{
    //retourne la position y du robot
    return this->posY;
}

void detruireRobot(std::vector<Robot>& vRobots, Robot& robot, int id, Terrain& terrain){

    //index de boucle
    size_t i = 0;
    for(vector<Robot>::iterator it = vRobots.begin(); it != vRobots.end(); ++it){
        //vérifie si c'est le robot avec l'id du robot à détruire
        if(vRobots[i].getId() == id){
            //Supprime le robot détruit de l'emplacement sur le terrain
            terrain.clearPosition(vRobots[i].posX,vRobots[i].posY);
            //Déconstruit le robot
            vRobots[i].~Robot();
            //supprime le robot dans le vecteur
            vRobots.erase(it);

            //Déplace le nouveau robot à l'emplacement du robot détruit.
            terrain.setPositionRobot((char)robot.id, robot.posX, robot.posY);


        }
        //incrément l'index de boucle
        ++i;
    }

}

Robot::Robot(int posX, int posY) : posX(posX), posY(posY)  { };

bool Robot::operator() (int posX, int posY){
    if(posX = this->posX and this->posY){
        return true;
    }
    return false;
}



Robot::~Robot(){

    //Décrémente le nombre de robot
    --nbreRobots;
}
```