IN2029 Programming in C++
Coursework

There is a single coursework in this module, counting for 30% of the overall module mark. This coursework is due at 5pm on Sunday 25th November. As with all modules, this deadline is hard, and extensions may only be requested via the standard Extenuating Circumstances procedure.

This coursework provides practice with classes and the C++ Standard Library. It does not require any features covered after session 6.

Overview

You are to implement two classes that form part of a simple game, with robots moving around in a 2-dimensional space.

I have prescribed a particular external structure and names for your classes: please follow these precisely, because I will be testing your solutions automatically. You may also add private members to your classes, and any other functions you find useful.

Description

You should implement and test the following classes.

class robot

representing a named robot moving around in a two-dimensional space.

Your class should have a single constructor

explicit robot(const string &n) set up a robot with name `n`, placed at the origin.

and the following public methods:

void move_north() moves the robot one step to the north.

void move_east() moves the robot one step to the east.

void move_south() moves the robot one step to the south.

void move_west() moves the robot one step to the west.

const string & name() const returns the name of the robot (as supplied to the constructor).

int north() const returns the current distance north of the robot. (This could be negative, if the robot has moved south more often than north.)

int east() const returns the current distance east of the robot. (As with the previous function, this value might also be negative.)

int travelled() const returns the total distance travelled by this robot since it was created.

You should also define an external function (not a member function):

int distance(const robot &r) returns the distance of robot `r` from the origin according the *Manhattan metric*: the least number of steps it would take to move the robot back to the origin. The standard function `abs(n)`, from the `<cstdlib>` standard header, may be useful here.

class game

A class holding many named robots.

Your class should have a default constructor and public methods

int num_robots() const returns the total number of robots in the game (initially none).

void move(const string &name, int dir) move the named robot one step in the specified direction (0 = north, 1 = east, 2 = south, 3 = west). If there is no robot of that name, one should be created at the origin and then moved as above.

int num_close() const returns the number of robots no more than 10 steps from the origin.

int max_distance() const returns the furthest distance of any robot from the origin.

string furthest() const returns the name of the robot that is the furthest distance from the origin.

vector<robot> robots_by_travelled() const returns a collection of all the robots in the system, arranged in increasing order of total distance travelled.

You should use library algorithms where appropriate.

In order to implement `move()` efficiently, your class should hold the robots in a map. You may use the following member functions of `map` not covered in the lectures:

count(k) returns 1 if the key `k` is in the map, and otherwise 0.

at(k) returns a reference to the value in the map corresponding to the key `k`. If the key is not present, it throws an exception (unlike `[k]`, which would create an entry).

emplace(k, v) creates a new entry in the map with key `k` and corresponding value `v`.

Marking

The classes will be worth 40% each.

The remaining 20% of the marks will be for clean programming style, including consistent layout, sensible identifier names, useful comments, avoidance of superfluous variables and data members, and general clarity of code. You may use language features not covered in the lectures, but your code must be standard C++. (If using Visual Studio, set the Disable Language Extensions property under C/C++ Language to Yes.)

Submission

Submit a ZIP file containing your source files only, to Moodle.