Международная цифровая олимпиада «Волга-IT`22»

Дисциплина «Прикладное программирование на C++»

Задание отборочного этапа

Два персонажа, Иван-царевич и Елена-прекрасная гуляли порознь в лесу и потерялись. Помогите им найти друг друга при том, что они не знают где находятся и как выглядит местность вокруг них.

Постановка задачи:

Карта местности имеет вид матрицы 10x10 где каждая клетка может быть либо проходимой, либо нет. Персонажи начинают свой путь с заранее неизвестной позиции и могут перемещаться влево, вправо, вверх, вниз или пропускать ход, оставшись на месте. Необходимо выяснить возможна ли встреча персонажей и если возможна, то привести их друг к другу. Под встречей подразумевается выполнение одного из двух условий:

- персонажи пришли в одну клетку матрицы одновременно;
- персонажи прошли мимо друг друга в противоположных направлениях.

Задание:

1. Определить возможна ли встреча персонажей за менее чем 1000 ходов и организовать их встречу если это возможно (500 баллов).
2. То же, но за менее чем 250 ходов (еще 250 баллов).

3. Построить карту местности на столько, на сколько это возможно за отведенное время (ходы), и вывести ее на экран в текстовом виде (250 баллов) где:
    a. Начальное положение персонажей обозначены знаками “@” и “&”, Ивана и Елены соответственно;
    b. Неизвестные клетки: “?”;
    c. Проходимые: “.” (точка);
    d. Непроходимые: “#”.

Если таких карт несколько, то вывести любую. Например,

```
@??????????
.#...#????
.#...#...?
.#.#.#.#.#
.#.#.#.#..
.#.#.#.##.
.#.#.#.#..
...#.#.#.#
???#.#.#..
???#...#.&
```

Структура проекта:

Для начала необходимо создать проект и подключить к нему два файла (fairy_tail.hpp и fairy_tail.cpp), написанные организаторами олимпиады. Файл input.txt необходимо положить рядом с исполняемым файлом проекта. Скачать их можно по ссылке: https://disk.yandex.ru/d/GKzL5MKjzeUSSg

Запрещается изменять файлы fairy_tail, как-любо получать доступ к закрытой (private) части этих файлов, а также программно читать исходные данные из input.txt.

Разрешается создавать экземпляр класса Fairyland и вызывать его методы go, canGo и getTurnCount.

Пример:

```cpp
#include "fairy_tail.hpp"
int main()
{
  // Создаем мир
  Fairyland world;
  // Определяем может ли Елена пойти на клетку вверх (up)
  std::cout << world.canGo(Character::Elena, Direction::Up) << std::endl;
  // Делаем шаг:
  // Иван (первый параметр) пропускает (pass) ход
  // Елена (второй параметр) идет вверх (up)
  // Функция возвращает true если пара встретилась и false иначе
  bool meet = world.go(Direction::Pass, Direction::Up);
  // В случае встречи (meet == true) можно получить количество затраченных ходов
  if (meet)
    std::cout << "Answer is found in " << world.getTurnCount() << std::endl;
  else
    std::cout << "Answer is not found" << std::endl;
  // Здесь мы завершаем наш пример, но можно продолжить вызывать функции go & canGo
  // до момента получения нужного нам ответа
  // Если встреча невозможна, то просто завершаем программу
  return 0;
}
```

Еще один пример доступен в файле main.cpp архива, доступного по ссылке выше.

Структура файла input.txt:

Для тестирования программы разрешается менять файл input.txt. Однако запрещается читать его программно (это сделает класс Fairyland за вас).

Кодировка файла: ASCII.

В 10 строках по 10 символов в каждой идет описание лабиринта в формате, описанному в основном задании, но без знаков вопроса.

Пример:

```
@#####....
.#...#....
.#...#...#
.#.#.#.#.#
.#.#.#.#..
.#.#.#.##.
.#.#.#.#..
...#.#.#.#
...#.#.#..
...#...#.&
```

Что хотим получить на проверку:

Решение принимается в виде ссылки на публичный репозиторий в GitHub.

Критерии оценки:

Оценивается полнота выполненного задания, качество кода, архитектура приложения, удобство сборки и установки приложения на жесткий диск, понятность кода, комментарии и документация в сложных местах.