Увод

1. Основата цел на играта е Том да стигне до Джери  с дрона и по пътя евентуално да разлее цветна боя на някои места. Като самата инфрмацията за стаята се взима от файл, който потребителя може да променя. След като пусне конзолата потребителя вижда кои са най-кратките пътища от Том до Джери, и избира един от всички, като след това му се показва колко колко е дълъг и завоя и боя има по него.
2. Структура на документацията
3. Проектиране
4. Реализация, тестване
5. Реализация на играта (включва важни моменти от реализацията на класовете и малки фрагменти от кода)
6. Тестови сценарии
7. Заключение

* Линк към репозиторито в Github:

https://github.com/ani4ka656/Tom-Jerry-SDP-Project

1. Проектиране.

Когато потребителя пусне програмата, той вижда вече дървото, съдържащо най-късите пътища, чийто върхове са инструкциите, които трябва да се въведат на дрона за стигне до Джери. Той трябва да избере един от тях като брой от ляво надясно;

Дронът на Том може да приема команди от следния тип

* **N(orth)** — лети на север
* **S(outh)** — лети на юг
* **E(ast)** — лети на изток
* **W(est)** — лети на запад
* **P(aint)** — разлей боя

Използвайки само тях, Том трябва да го “програмира”, за да реализира идеята си. Естествено има и някои ограничения:

* в стаята има мебели, над които е опасно да се лети
* батерията на дрона издържа за кратко и за това той не трябва да повтаря движенията си

За да завърши успешно мисията, котаракът трябва да вземе някои важни решения. Бързо осъзнал, че това е тежка задача, за която ще му трябва вашата помощ. Разполагате със следната информация, записана в текстов файл:

Тук е описано всичко относно структурата на самата стая, координатите, посоките, мебелите и боята.

<https://docs.google.com/document/d/1Ct_Iof1hUEWs6e4FBI57WIwbZFw-NMsAQ_JKWd2YkAk/edit>

Програмата трябва да разполага с подходящи методи, които:

* извеждат командите, които Том трябва да въведе в дрона си
* дават информация за
  + дължината на пътя
  + количеството разлята боя
  + броя завои

За да могат да се осъществят тези операции е създаден class Room, който се грижи за изпълнението им.

1. Реализация
2. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

* class :

Data Members (private):

class Room{

private:

vector<vector<int> >room;

pair<int,int> Jerry;

pair<int,int> Tom;

int furnitureCount, paintSpotsCount;

pair<int, int> paintSpots;

int n, m;

void loadGraphwithoutFurniture(vector<vector<pair<int, char> > > &);

void loadRoom(ifstream& fileName);

void findAllPaths(int, int, const vector<vector<pair<int, char> > > &, vector<bool> &, string=""); //dfs

int findMinimalPaths(int, int, const vector<vector<pair<int, char> > > &, vector<vector<char > >&);//bfs

void dfs(int, int, const vector<vector<pair<int, char> > > &);

void printPathTo(int v,vector<vector<char> >&parents );

public:

Room(string& fileName);

void start(); //Room& operator=(Room const& other);

void print();

};

1. Как работи програмата (тестови сценарии)
2. Заключение