Документация

Кратък анализ на задачата

Проекта се разделя на две основни задачи. Първата задача е намирането на всички пътища от Том до Джери, а втората задача е построяването на дърво от най-кратките намерени пътища. Картата на стаята, която се чете от файла, се записва в двумерен масив от тип char. На нея свободните места, по които може да лети дрона се отбелязват с „-“, местата, на които има мебел се отбелязват с „M“, местата подходящи за разливане на боя с „P“ и местата, на които се намират Том и Джери се отбелязват съответно с „T“ и „J“. За намиране на пътищата се използва рекурсивна функция, като за начало се вземат кориднатите на Том и крайната цел са кординатите на Джери, като се има предвид, че не може да се минава там където има мебели. Всички пътища се записват в списък от стрингове като последователност от символите „F“ (Front), „B“ (Back), „L“ (Left), „R“ (Right).

От тях се взимат най-късите пътища, с които се работи. За построяването на дърво от команди, които Том да въведе в дрона си, също се използва рекурсия. Взима се масив от стрингове от всичките най-къси пътища и се сравняват първите символи на всеки стринг. В зависимост от това колко различни символи има, тоест се тръгва в различни посоки, се създават толкова нови листа със съответните символи като стойности на тях. След това се като от стария масив от стрингове се премахне първия символ на всеки стринг се получава нов масив от стрингове и този нов масив се подава на същата функция и така докато масива остане празен. След това полученото дърво се записва в файл със специален синтаксис и се чете от програмата, която го визуализира – Graphviz.

Описание на класовете

Проектът се състои от два класа. Първият клас се нарича Map и в него се намира основната логика. Той съхранява картата на стаята, намира пътищата от Том до Джери и извежда исканата от Том информация. Вторият клас се нарича Tree и в него се намира имплементацията на дървото от команди, което се записва в файл и се извежда. Следва подровно описание на класовете и техните член-данни и член-функции.

Map.h

```
private:
Tree treeOfCommands; - Дървото в което, ще се запишат командите
char** map;  - Указател към двумерния масив, в който се пази пази картата на стаята
```

`int n, m;` - Размери на стаята (съответно височина и ширина)
`Point tom;` - Координати на Том
`Point jerry;` - Координати на Джери
`list<string> minPaths;` - Списък от стрингове, който съдържа минималните пътища
`list<string> allPaths;` - Списък от стрингове, който съдържа всички пътища
`char* path;` - Временна променлива, която се използва от findPath
`int position;` - Временна променлива, която се използва от findPath

`void loadFromFile(char* pathToFile);` - Зарежда файла в двумерен масив (map) и извиква findPath и minimizePaths()
`void savePath(char path[], int startPos, int endPos);` - Запазва нов път в списъка minPath
`void findPath(int row, int col, char direction);` - Рекурсивна функция за намиране на пътищата
`string removePaintFromPath(string path);`- Приема път като стринг и връща стринг, но без P-та
`int howMuchP(string path);` - Връща колко боя ще се разлее по пътя path
`string* listToStringArray(list<string> l);` - Връща масив от стрингове, които са записани в списък и добавя и "P"-paint към пътя
`int getListLength(list<string> l);` - Връща броя на елементите в списъка l
`void minPathsToTree();` - Записва дърво в treeOfCommands от списъка minPaths
`void minimizePaths();` - От всички пътища в списъка minPatsh оставя само най-късите

`public:`
`Map(char* pathToLoadFile);` - Конструктор, който взема пътя на файла, от които ще се чете стаята
`~Map();` - Декструктор. В него се трие динамично заделената памет
`int lengthMinPath();` - Връща дължината на най-късия път
`void whatCommands();` - Извежда какви команди да се въведат за всеки от пътищата
`void howMuchPaint();` - Извежда колко боя ще разлее на всеки от минималните пътища
`void howMuchTurns();` - Извежда броя на завоите на всеки от пътищата
`void printMinPaths();` - Извежда минималните пътища
`void showTreeOfCommands();` - Показва дървото от команди чрез Graphviz
`void printInfoAboutPath(int index);` - Принтира информация за път с даден индекс
`void maxPaintMinTurnsPath();` - Принтира пътя с най-много разляна боя и най-малко завои
`void twoDronesMaxPaint();` - Принтира два различни, равни по дължина, минимални пътя с максимално разлята боя

Структура Node

```
struct Node {
	char data; - Пази един от символите „F“, „B“, „R“, „L“ и „P“
	int number; - Номер на върха (уникален за всеки връх)
	// Указатели към други върхове
	Node* front;
	Node* right;
	Node* back;
	Node* left;
	Node* paint;
};
```

Tree.h

```
private:
int countNodes; - Брой на върховете
Node* root; - Указател към корена на дървото

int differentChars(string s); - Връща броя на различните символи в един низ
bool sameFirstChars(string arr[], int n); - Проверява дали първите букви на масив от стрингове са еднакви
int makeNewStringArray(string* &newArr, string oldArr[], int n, char data); - Подава се по референция нов масив, който се пълни с елементите на стария масив, които започват със символа data и функцията връща размера на новия масив
void writeLabels(ofstream& of, Node* root); - Използва се от saveTreeToFile(), записва етикетите с номерата на върховете
void writeConnections(ofstream& of, Node* parrent, Node* child); - Записва връзките между върхове

public:
Tree(); - Конструктор. Инициализира countNodes=1 и прави връх-корен със ст-ст S
~Tree();
Node* getRoot(); - Връща указател към корена на дървото
void minPathsToTree(string* arr, int n, Node* parrent); - Взима минималните пътища от масив от стрингове (в които има включени местата за боя) и прави дърво с корен parent
void saveTreeToFile(char* pathToFile); - Запазва дървото в файл по подходящ начин за визуализиране с Graphviz
```

Бъдещи подобрения

Вместо стаята да се представя с двумерен масив от тип char и да се обхожда с рекурсия, което при по-големи размери на масива става доста бавно, може да се представи чрез граф. И вече търсенето на най-късите пътища в графа ще става още по-бързо.