Том и Джери

Връзка към проекта: https://github.com/MLambeva/SDP-project
Изготвил: Моника Ламбева, ФН: 82017

1. Увод:

1.1 Идеята на проекта е успешно да се програмира дрон, чрез който Том да прави пакости и да обвинява своя приятел Джери. Анимационните герои се намират в стая, в която има мебели. Дронът може да извършва само определени движения и не може да лети над мебелите. Той трябва да бъде програмиран така, че успешно да стига до Джери, тръгвайки от Том.

1.2 Целта на разработката е по даден текстов файл да се пресъздаде стая, в която са разположени героите Том и Джери. В тази стая мебелите са с различна форма и над тях не трябва да минава дронът. Освен това батерията на дрона не издържа дълго, затова той не трябва да повтаря движенията си. Дронът може да изпълнява само пет команди – да ходи на север, на изток, на юг, на запад и да разлива боя. Разливане на боя е допустимо само на предварително зададени места. Освен да се програмира дрона, Том иска да му се визуализира дървото от команди с най-кратките пътища, като за тази цел съм използвала инструмента “GraphViz”. След визуализацията, по даден номер на листо (започвайки от 0), потребителят може да си избере за кой път иска да разбере повече информация. Програмата се реализира чрез главна функция, която демонстрира всички функционалности.

1.3 В документацията по-долу съм разяснила по-подробна информация относно:

- Как съм проектирала класовете си и какви структури от данни съм използвала в тях.
- Как съм реализирала методите в класовете.
- Как мисля, че съм се справила и мнението ми за бъдещо усъвършенстване.

2. Проектиране:

2.1 Проектът се състои от:

- четири файла с разшерение “.h”, в три от които са дефинирани отделните класове, а в четвъртия – структурата, определяща позиция спрямо координатна система;
- пет файла с разширение “.cpp”, в четири от които са имплементирани конструктурите и функциите на отделните класове и структурата, а петият е главната функция, в която се реализира проектът;

2.2 Основните файлове в проекта съдържат:

➢ **Структура “Position”**, която се представя чрез:

- две целочислени променливи, които представляват координати.

```
struct Position
{
    int x;
    int y;
```

В структурата са дефинирани функции за връщане на четирите посоки – север, изток, запад и юг, оператор за проверка на равенство, оператор за присвояване и функция за четене от текстов файл.

- **Клас “Room”**, който се представя чрез:
  - матрица от символи;
  - позиция на Джери;
  - позиция на Том;
  - вектор със всички възможни пътища за достигане от Том до Джери;

```
private:
std::vector<std::vector<char>> room;
Position Jerry;
Position Tom;
std::vector<std::string> possiblePaths;
```

В този клас са дефинирани функции, чрез които по даден текстов файл се създава матрица от символи, в която са разположени Том , отбелязан със символ ‘S’ и Джери – със символ ‘F’, мебелите, отбелязани със символ ‘1’ и местата, в които може да се разлива боя – със символ ‘P’.

- **Клас “Tree”**, който се характеризира с:
  - указател към структура, която е в самия клас, описваща възел със символ, пет указателя към други възли, които представляват позволените посоки за дрона и идентификатор, който отличава листата едно от друго;

```
private:
struct Node
{
    char data;
    Node* north;
    Node* east;
    Node* south;
    Node* west;
    Node* paint;
    int idLeaf = -1;
};
Node* start;
```

В този клас са дефинирани конструктор, деструктор и функции за създаване на дърво по дадени пътища, извеждане на път по дадено листо, визуализиране на всички пътища и други функции за работата на програмата.

- **Клас “DroneProgramming”**, който се характеризира със:
  - стая;
  - дърво, представящо всички пътища в стаята;
  - дърво, представящо най-кратките пътища в стаята;

В този клас са дефинирани функции, които имплементират основните и допълнителните функционалности на проекта. Чрез тях се извеждат най-кратките пътища в дадена стая, извежда се дължината, броя завои и количеството разлята боя по път, избран от потребителя чрез номер на листо.

```
private:
Room room;
Tree droneWithAllPaths;
Tree droneWithShortestPaths;
```

3. Реализация

3.1 Реализиране на функции:

- В **структурата "Position"** са имплементирани:
  - 4 функции, връщащи позициите, отговарящи на посоките север, изток, запад и юг
  - Оператор за проверка дали две позиции са равни.
  - Оператор за присвояване на стойност на дадена позиция.
  - Функция за четене от входен поток.

- В **клас "Room"** са имплементирани:
  - Множество функции, които са помощни и са поставени в "private" частта, понеже се използват само в областта на класа. Пример са функциите, чрез които четем от текстовия файл. Този файл започва с информация за размерите на стаята като изискването е да са две целочислени числа, разделени с интервал. След това има информация за позициите на Джери и Том, като там се използва функцията за четене на позиция. Следва информация за броя мебели, които има в стаята и броя на местата, подходящи за разливане на боя. Мебелите се описват чрез правоъгълник от символи от единици и празни места, като се задава за позиция на мебелта северозападният ъгъл, а краят на описанието е отбелязано с „===".
  - В този клас съм използвала структурата от данни „вектор".

- В **клас "Tree"** са имплементирани:
  - Конструктор и деструктор.
  - Структурите от данни, които са използвани в класа, са дърво и опашка.
  - Дървото е с пет съседа, които представляват командите, които могат да бъдат въведени в дрона – лети на север('N'), лети на юг('S'), лети на изток('E'), лети на запад('W') и разлей боя ('P'). То има листа, които имат идентификатор, номериран от 0. За създаването на идентификатор съм имплементирала рекурсивна функция.
  - В класа има метод, който има за аргумент номер на листо и чрез който се връща низ от валиден път от Том до Джери.
  - Чрез инструмента "GraphViz" създавам визуализация на дървото, която включва номерацията на листата и легенда.

- В **клас "DroneProgramming"**са имплементирани:
  - Функции, чрез които се принтират командите, които трябва да се въведат в дрона, след като потребителят си е избрал номер на листо. Въвеждането продължава до написване на валидно листо. По това листо се извеждат и дължината на избрания път, количеството разлята боя и броя на завоите.
  - В този клас са използвани структурите дърво и вектор.
  - Първата допълнителна функционалност е да се намери пътят с най-много разлята боя и най-малко завои от всички възможни пътища, като имплементацията е отделена в друга функция.

- Втората допълнителна функционалност също е отделена във функция и в нея се намира най-бързият път и междувременно с най-много разлята боя, като се допуска и по пътя да няма боя.
- Тези всичките функции, които показват работата на програмата, са обединени в друга функция, която се използва при стартиране на програмата.

```
int main()
{
    DroneProgramming drone;
    drone.start();

    return 0;
}
```

3.2 Примери:

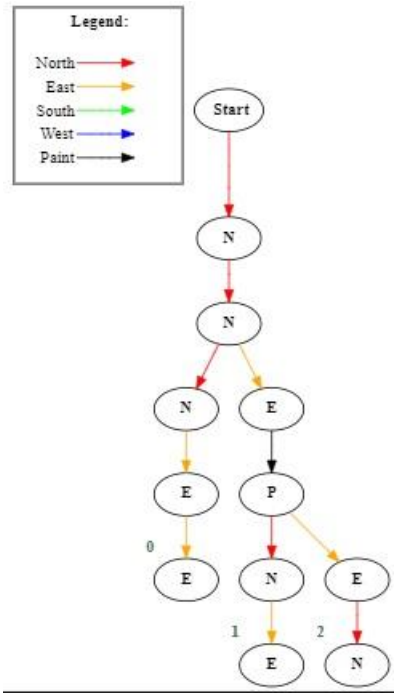
➢ Пример 1:

Потребителят въвежда име на файл, от който ще се чете информацията за стаята. При некоректно име, т.е. несъществуване на такъв файл, програмата започва отначало, т.е. се очаква ново задаване на име на файл.

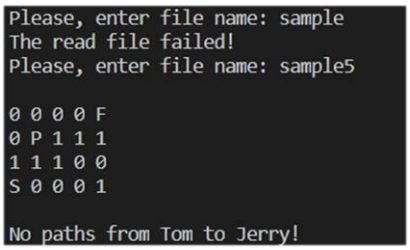
```
Please, enter file name: sample1

0 0 F 0
0 P 0 1
0 1 1 0
S 0 1 0

Please, type a file name to visualize tree with all possible paths in: firstSample
Please, type a file name to visualize tree with the shortest possible paths in: firstSample1
Please, enter number of leaf: 0

Commands which have to be entered in the drone: N->N->N->E->E
Length of path: 5
Amount of paint spilled: 0
Number of turns: 1

Paths with the most paint spill and the least turns: N->N->E->P->E->N
The shortest path with the most paint spill: N->N->E->P->N->E    N->N->E->P->E->N
```

Ако четенето е неуспешно, т.е. ако има невалидни данни - позициите на мебелите, анимационните герои или местата за разливане на боя са зададени извън стаята или върху места, на които не могат да бъдат разположени, програмата хвърля изключение и се очаква въвеждане на нов текстов файл.

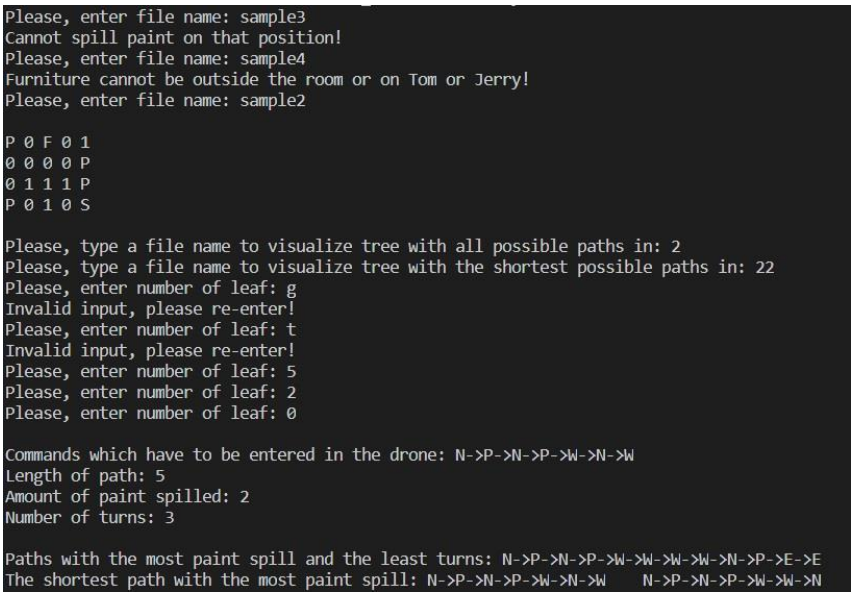
При успешно четене и коректни данни, се извежда стаята под формата на лабиринт и се очаква потребителят да напише две имена на файлове, в които да се запише визуалицацията на дървото с пътищата. След това потребителят може да види дървото с най-кратки пътища и да си избере номер на листо, като при некоректно въведено число, се дава възможност отново да се въведе. След това програмата показва на екрана изпълнението на основната функционалност, т.е. командите, които трябва да се въведат в дрона, дължината на пътя, количеството разлята боя и броя завои. Следва принтиране на пътищата с най-много разлята боя и най-малко завои, както и най-кратките пътища с междувременно най-много разлята боя.



- Пример 2:
  Първо съм въвела несъществуващ файл и ми дава възможност отново да въведа. При въвеждане на коректно име ми извежда стаята и както се забелязва, няма път от Том(‘S’) до Джери(‘F’) и програмата завършва.



- Пример 3:
  - В текстовия файл “sample3” на позиция за разливане на боя е зададена позицията на Том, което е некоректно, и съответно се хвърля изключение, което е хванато.
  - В текстовия файл “sample4” е направено опит мебелта да е по-дълга от размерите на стаята и отново се хвърля изключение.
  - В текстовия файл “sample2” данните са вече коректни и се принтира на екрана. След въвеждане на име на файл за записване на визуализацията, е показано как работи програмата при въвеждане на невалиден вход за номер на листо. След написано коректно листо, програмата работи по гореописания начин.



4. Заключение

Смятам, че успях да осъществя първоначалните си цели и идеи. В бъдеще, когато придобия повече познания, този проект може да се доусъвършенства, но със знанията ми до тук, съм способна само на това. 😊