# „Алиса в Страната на чудесата“

ПРОЕКТ

Велизар Зарков | ООП | 30. 05. 2021

# 1. Цели

1.1. **Описание:** Проектът представлява реализация на RPG (Role Play Game) игра. Алиса е главната героиня в играта, която ненадейно отново е попаднала в Страната на чудесата и трябва да се измъкне от нея. Задачата се изразява в това да се представи игрално поле - поредица от лабиринти, както и възможност за Single Player интерфейс, изразяващи сценарий от книгата на Луис Карол “Алиса в Страната на чудесата”. Целта на Алиса е да намери изходния портал. По пътя си може да срещне врагове, които да попречат на преминаването, или да намери оръжия, които да ѝ помогнат.

1.2. **Цел и задачи:** За целта трябва да се реализират: игрално поле, герои, предмети, свойства на героите и предметите, допълнителни помощни класове, като *string* и *container*, и др.. Също е необходима подходяща йерархия в която всички компоненти да се свържат и функции, които да изпълняват функционалната част на проекта. Също така е необходим потребителски интерфейс, чрез който потребителят може изпълнява команди чрез клавишите със стрелки от клавиатурата и съответно да получава обратна информация от програмата за текущото състояние в което се намира. Реализиране на подходящо форматиране и запаметяване на информацията за игралното поле, герои, предмети и т.н. във файлове, за да може при повторно зареждане на програмата, тя да зареди необходимата информация и да продължи нейното изпълнение. Допълнително са реализирани функции, чрез които могат лесно да се създават нови игрални полета, герои и предмети.

## Съдържание

# 2. Основни етапи в реализирането на проекта

2.1. **Основни дефиниции, концепции и алгоритми**: В проекта е реализиран алгоритъм за търсене на път в лабиринт.
operator<< и operator>> оператори предефинирани, така че всеки клас в който е упоменато, че са предефинирани, може да бъде въвеждан и извличан от потоци (файлов, вход от конзолата).
Голяма четворка и Голяма шестица – където е упоменато, че са реализирани да се разбират конструктор, copy конструктор, operator= и деструктор, а за Голяма шестица и move copy конструктор, както и move operator=.
Инициализиране – в началото на програмата се създават контейнери за картите, героите и предметите, също така се създава главния герой Алиса. След това се зареждат всички карти, герои и предмети от файловете в които са били записани и стават готови за употреба.

2.2. **Проблеми и сложност:** Едно от големите предизвикателства пред проекта беше създаването на функционалността за движение в лабиринта и изпълняване на всички видове проверки, както и събития (среща на герой и влизане в режим на битка, събиране на предмети, употреба на телепортиращи предмети). Също така входът от клавиатурата, чрез клавишите със стрелки (Използвана литература);

2.3. **Подходи за решаване на проблемите:** Планиране на цялостната архитектура, осмисляне на идеята и целите. Създаване на помощни средства (класове) с цел опростяване на по-сложните класове. Справяне с възникнали проблеми и отстраняването им.

# 3. Проектиране

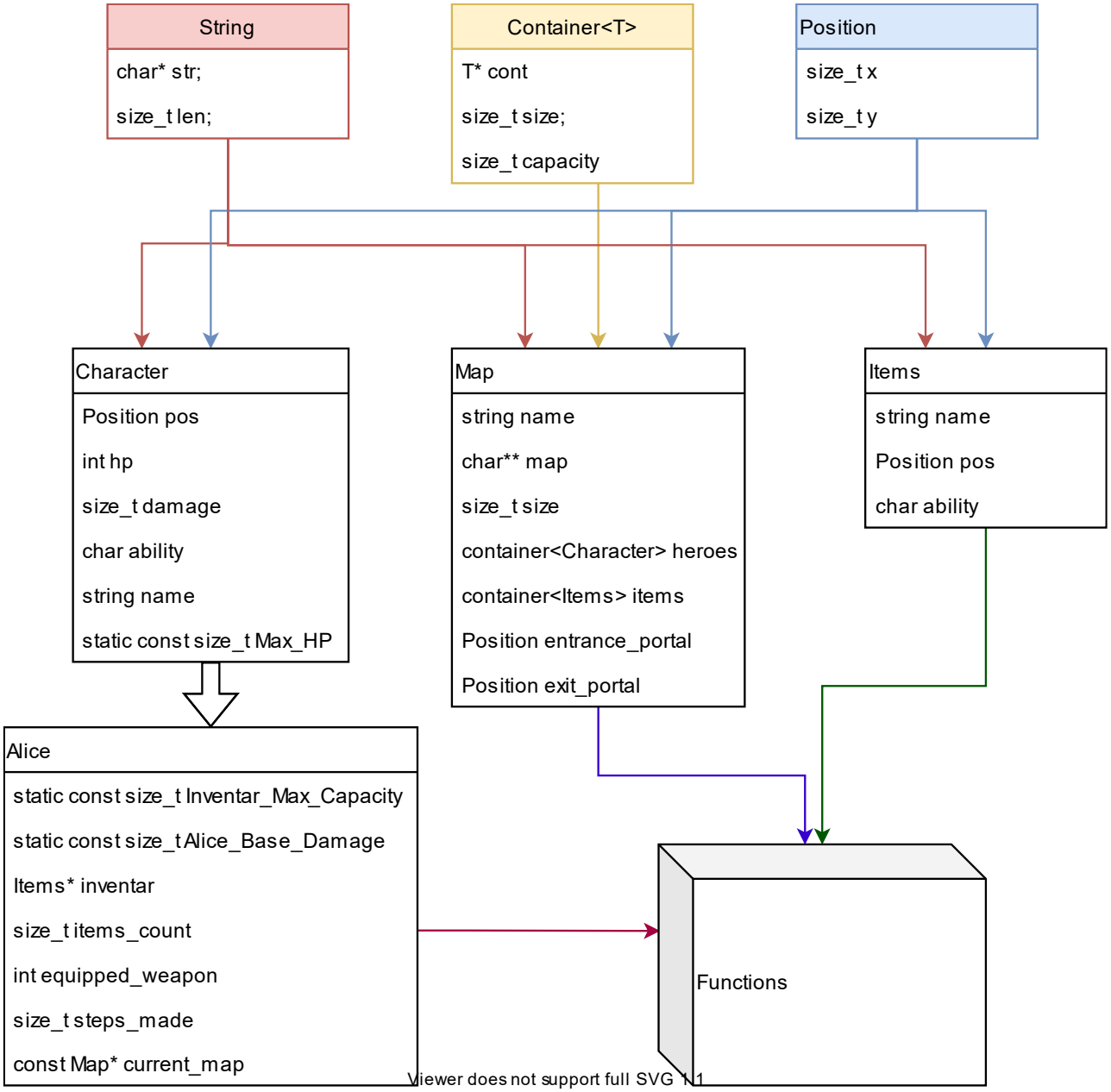
3.1. **Обща архитектура:** Проектът е разделен на няколко слоя: базови класове, помощни класове, поддържащи базовите класове и функционален слой. Архитектурата може да се види в диаграма 1.

3.1.1. **Поддържащи класове:** *String, Position* и *Container***.** Идеята на *String* е да осигурява работа с низове. *Position* е структура, която включва две координати. *Container* включва динамичен шаблонен масив.

3.1.2. **Базови класове:** *Character, Items, Map, Alice*. *Character* включва основна информация за героите в играта. Items включва основна информация за предметите. *Map* е игралното поле. *Alice* е вид *Character* с разширени възможности.

3.1.3. **Функционален слой:** Това е слоя, който обединява всички останали компоненти в съвместна работа. Включва: инициализация, създаване на карти, герои, предмети, взаимодействие между потребителя и програмата, както и функционалността на самата игра.

## 3.2.Диаграми:




диаграма 1.

## 4. Реализация

### 4.1. **Реализация на класовете:** Реализация на класовете по слоеве:

#### 4.1.1. **Помощни класове:** *String* съдържа динамичен масив от тип char и size_t за размера на низа. За да бъде функционален, класът е реализиран с три вида конструктори: като може да приема масив от символи, конструктор по подразбиране и по специфичен конструктор приемащ символ и дължина, като стрингът се запълва с подадения символ. Реализирана е Голямата шестица. Класът притежава оператори за сравнение, конкатенация и за достъп (operator[]), гетъри, функции за търсене на подстринг, функция запълване:

```
friend void fill(string& tb_fileld, const string& filler);
```

която запълва стринг с друг стринг пълнител, като запълването става от началото на стринга до запълването му до край или до свършване на стринга пълнител. Също така String притежава operator<< и operator>>.

*Position*: Структура с две променливи от тип `size_t` с два конструктора, един по подразбиране и един приемащ две координати, съдържа `operator==` за сравнение, `operator<<` и `operator>>` и move функции, които инкрементират дадена координата спрямо посоката в която искаме да се движим.

*Container*: е шаблонен клас съдържащ указател към шаблонния тип, `size_t` размер и капацитет на контейнера. Реализирани са: Голяма шестица, `operator<<` и `operator>>`, гетъри, функции за добавяне и премахване на обект, инициализираща функция, която запълва контейнера със стойности по подразбиране, за да бъдат разпознавани от операторите за сравнение, принтиране на съдържанието, оператор за достъп до елемент.

#### 4.1.2. **Базови класове:** *Character* съдържа string за името на героя, неговата позиция, здраве, щети, способности и статична променлива за максималното здраве на всеки герой. Реализирани са два конструктора: по подразбиране и един приемащ щети, способност и име. Няма реализирана Голяма четворка, поради добре дефинираните член-данни. Има гетъри и сетъри, `operator<<` и `operator>>`, оператори за сравнение (==, !=).

*Items* съдържа string за името на предмета, неговата позиция и способност. Реализирани са два конструктора: по подразбиране и един приемащ име и способност. Няма реализирана Голяма четворка, поради добре дефинираните член-данни. Има гетъри и сетъри, operator<< и operator>>, оператори за сравнение (==, !=). Както и функция activate, която се изпълнява при екипиране на предмет от Alice и представлява реализация на всички видове способности, които имат предметите и чрез `switch (ability)` се изпълнява способността притежавана от даден предмет.

*Map* е най-сложният клас съдържащ динамична матрица от тип char** която представлява игралното поле, като възможните символи за запълване на клетките са от клас `enum map_symbols { wall = 'w', free_space = 'f', entrance_portal = 'e', exit_portal = 'x'};`

Също така има алгоритъм за търсене на път в лабиринта, който се използва при създаването на карта, за нейното валидиране, както и за намиране на броя стъпки необходими до изходния портал, функции за проверка на валидно движение, дали клетката е в картата и дали клетката е проходима.

*Alice* наследява публично Character и добавя статичен масив inventory от тип Items, също така има променливи от тип `size_t` за броя на предметите, позиция на екипиран предмет от тип int, както и указател към картата на която се намира Алиса от тип `const Map*.` Не е осъществена Голяма четворка, член-данните са добре дефинирани, също така има функции за взимане на предмет, зареждане на карта (вкл. Приемане на указателя към картата и задаване позицията на входния портал), функции за движение в четирите посоки, проверка за наличност на предмет в инвентара и др.

## 4.2. Управление на паметта и реализация на алгоритмите. Оптимизации:

### 4.2.1. Алгоритъм за търсене на път в лабиринта:

`bool path_algorythm(Position pos, Map& map, size_t& steps)`

Алгоритъмът е рекурсивен и започва от подадената му позиция и проверява първо дали позицията е в картата, чрез функцията за проверка в класа Map, ако не е връща лъжа -> проверява дали настоящата позиция е изходния портал (дъно на рекурсията), връща истина -> проверява дали клетката е проходима, ако не е връща лъжа -> прави настоящата клетка непроходима -> извиква се рекурсивно функцията с нови позиции преместени съответно в четирите посоки -> ако е успешен се връща истина. Също така измерва броя стъпки необходими до края на лабиринта.

```
steps++;
   if (!pos_in_map(map, pos))
   {
        steps--;
        return false;
   }

   if (map.pos_is_exit_portal(pos))
        return true;

   if (check_cell_unwalkable(map, pos))
   {
        steps--;
        return false;
   }

   map[pos] = wall;
```

```
path_algorythm(pos.move_right(), map, steps);
path_algorythm(pos.move_down(), map, steps);
path_algorythm(pos.move_left(), map, steps);
path_algorythm(pos.move_up(), map, steps);

map[pos] = free_space;
return true;
```

4.2.2. **Оптимизация:** В класовете String, Container, Map са реализирани move конструктор и operator=.

4.3. **Планиране, описание, създаване на тестови сценарии:**
Създадени са примерни карти, герои и предмети, чрез функциите за създаване на такива, форматирани и запазени във файлове.

## 5. Заключение

5.1. **Обобщение:** Проектът притежава много функционалности и база за бъдещо развитие. Благодарение на ООП стила, реализираните класове дават добра абстракция, така че да могат да се създават неограничено много герои предмети и карти, които могат да преизползват част от функционалността, като например способностите на предметите и героите и т.н.

5.2. **Бъдещо развитие и препоръки:** За в бъдеще може да се разшири идеята за потребителския интерфейс като се добави нова функционалност Score показваща резултата на играча, като се калкулира спрямо брой направени стъпки, брой необходими стъпки за излизане от лабиринта, брой убити чудовища и т.н. в момента тази идея има достатъчна основа за да бъде реализирана безпроблемно и независимо.

## 6. Използвана литература:

За използване на _getch() функцията при въвеждане на команди чрез клавишите със стрелки. https://cboard.cprogramming.com/cplusplus-programming/1656-how-do-you-represent-arrow-keys-cplusplus.html
Използване на "\033[A\r" за връщане на редове нагоре в конзолата, за да могат да се изтриват https://stackoverflow.com/questions/1508490/erase-the-current-printed-console-line