Lösung für Aufgabe 6

a)

C1: 1

C2: n - 1

C3: n - 1

C4: $\sum_{i=1}^{n-1}(n-i+1)$

C5: $\sum_{i=1}^{n-1}(n-i)$

C6: $\sum_{i=1}^{n-1}(n-i)$

C7: n-1

b)

Lösung:

$O(n^2)$

Begründung:

Der Algorithmus beinhaltet zwei ineinander verschachtelte For-Schleifen. In der ersten For-Schleife wird jedes Element des Arrays betrachtet, wodurch sie eine Zeitkomplexität von O(n) aufweist. In der zweiten Schleife wird ein ausgewähltes Element mit jedem anderen Element des Arrays verglichen, wodurch sie ebenfalls eine Zeitkomplexität von O(n) aufweist.

Da die Schleifen ineinander verschachtelt sind, +läuft die innere Schleife n*n-Mal durch, wodurch die Zeitkomplexität des Algorithmus insgesamt O(n) * O(n) = $O(n^2)$ beträgt.

Informationen zu Aufgaben 1 - 5

Tastenbelegung:

WASD / Pfeiltasten	Spielerbewegung
I	Inventory
E	Aufheben von Items
ESC	Schließt das Spiel
G	Debug: Zeigt das fLocalGoal aller Tiles
H	Debug: Zeigt das fGlobalGoal aller Tiles

K	Berechnet Pfad und lässt die Spielfigur zum Ziel reisen
L	Debug: Zeigt den Pfad an, mithilfe dessen die Map generiert wurde
U (Im Inventar)	Debug / Für Aufgabe: Füllt das Inventar mit Gegenständen
Anmerkung:	Mit dem Inventar wird primär über die Maus interagiert (Item-Slots können angeklickt werden)

Aufgabe 1:

a)

Lösungsort:

MapManager.cpp, hauptsächlich die generateMap()-Funktion.

Im Ordner src/Tiles liegen die verschiedenen Tile-Klassen, die für diese Aufgabe verwendet wurden

Weitere Informationen:

Die MapManager-Klasse wurde von Anfang an so konzipiert, dass Maps automatisch generiert werden. Mithilfe der setTile()-Funktion könnten Maps aber auch manuell bearbeitet werden.

b)

Lösungsort:

MapManager.cpp in der generateMap()-Funktion, etwa bis zum Kommentar „Spawn chests“.

Weitere Informationen:

-/-

c)

Lösungsort:

MapManager.h in Form von std::vector<ItemBase> items (speichert alle Items einer map).

MapManager.cpp in der Draw()-Funktion unter dem Kommentar „Draw items“.

MapManager.cpp in generateMap() unter dem Kommentar „Spawn chests“.

Im Ordner src/Items liegt die erstellte ItemBase-Klasse, sowie alle davon abgeleiteten Kindklassen.

In src/Tiles liegt die erstellte TileChest-Klasse.

Weitere Informationen:

Ich habe die Aufgabe so verstanden, dass min. 5 Items in Form von Kisten auf der Map liegen sollen. Bewegt sich der Player auf eine Kiste, wird diese geöffnet und das Item wird direkt ins Inventar des

Spielers gelegt, sofern es nicht voll ist. Items können allerdings trotzdem offen auf der Map liegen, wenn sie z.B. aus dem Inventar des Spielers gedroppt werden.

Aufgabe 2

a)

Lösungsort:

Inventory.h Konstruktor, addItem(), setItem(), etc.

Die Equipment-Slots sind nach dem Schema „slotWeapon“, „slotNecklace“, etc. benannt.

ItemBase.cpp (Stärke-Attribut)

Weitere Informationen:

-/-

b)

Lösungsort:

In src/Characters befindet sich die Char-Klassenfamilie.

Movement wurde in Char::move() und Player::handleInput() umgesetzt.

Das Interface zum Ausrüsten und Ablegen von Items wird durch die Inventory-Klasse bereitgestellt.

Weitere Informationen:

Funktionen wie bspw. das Ausrüsten von Items können durch das Anklicken von Inventarslots genutzt werden.

Aufgabe 3

a)

Lösungsort:

Inventory.h in der sortInv()-Funktion.

Weitere Informationen:

-/-

b)

Lösungsort:

Inventory.h in der createDemoInv()-Funktion.

Inventory.h in Update() unter dem Kommentar „Sort buttons“.

In der Draw()-Funktion der Inventory-Klasse wird das Inventar visualisiert.

Weitere Informationen:

Die Sortierfunktionen können im Inventar-UI mit den Tasten links neben dem Schließen-Button verwendet werden. „W“ sortiert nach Gewicht, „$“ nach Preis und „N“ nach Name. Für das Sortieren wird bei jeder Aktivierung zwischen auf- und absteigend umgeschaltet, was durch die wechselnden Farben der Buttons gekennzeichnet wird. Wird ein blauer Button angeklickt, wird aufsteigend sortiert; wird ein grüner Button angeklickt, wird absteigend sortiert.

Ein Inventar zur Demonstration dieser Funktion kann mit der Taste „U“ generiert werden, wenn das Inventar geöffnet ist. Hierbei sollte beachtet werden, dass beim Benutzen dieser Funktion der alte Inhalt des Inventars überschrieben wird.

Aufgabe 4

Lösungsort:

In der MapManager-Klasse in den Funktionen, die in der Header-Datei mit dem Kommentar „For A* algorithm“ gekennzeichnet sind.

Weitere Informationen:

-/-

Aufgabe 5

a)

Lösungsort:

Für 1.: Geschieht im Konstruktor der MapManager-Klasse unter spawnPlayer().

Für 6.: Geschieht in MapManager::autoTraverse().

Für 7.: Geschieht in MapManager::checkWinCondition() (Generiert einen neuen Floor, wenn das Ziel erreicht wurde)

Für den Rest: Siehe die Lösungsorte der vorherigen Aufgaben

Weitere Informationen:

Die Taste zum Aktivieren der automatischen Traversierung ist „K“

Traversierung von Hand:

Lösungsort:

Für 1.: Geschieht in MapManager::Draw() unter dem Kommentar „Draw player“.

Für den Rest: Siehe die Lösungsorte der vorherigen Aufgaben

Weitere Informationen:

-/-

Traversierung mittels pathfinding Algorithmus

Lösungsort:

Für 1.: Geschieht in MapManager::autoTraverse() (drawPathIndicator wird auf wahr gesetzt und in Tile.cpp gezeichnet).

Für 2.: Geschieht in MapManager::checkForChests().

Für 3.: Geschieht in Inventory::addItem().

Für 4.: Geschieht in MapManager::checkWinCondition().

Weitere Informationen:

Die automatische Traversierung der Map wird durch die Spielfigur visualisiert, indem sie sich jeden Frame um ein Tile bewegt und dabei gerendert wird.