coding
contest
2018

KNAPP

AUFGABENSTELLUNG

ALLGEMEINE INFORMATIONEN

Die Aufgabenstellung ist als *Einzelarbeit* innerhalb von 2,5 Stunden zu lösen. Wie im realen Programmieralltag ist es erlaubt, das Internet zur Recherche zu verwenden. Bei Fragen zur Problemstellung, wende Dich bitte an einen der KNAPP-Betreuer.

EINLEITUNG

In einem Lager sind viele Produkte in großer Anzahl gelagert. Diese Produkte müssen dann für die Aufträge der Kunden aus dem Lager geholt und versandt werden. In einem klassischen Lager erfolgt die Lagerung in Regalen, zu denen der Mitarbeiter hingehen muss, um die einzelnen Produkte für die Aufträge der Kunden zu holen.

Um hier den Mitarbeiter zu entlasten hat KNAPP das Konzept Ware-zur-Person entwickelt und erst kürzlich eine neue Generation am Weltmarkt vorgestellt, das OSR Shuttle™ Evo, ein Shuttle-System das sich in zwei Dimensionen bewegen kann.

Dabei werden die Produkte in Behältern in einem automatisierten Regal gelagert und von automatischen Shuttles zu einer Arbeitsstation gebracht und von dort wieder zurück eingelagert. Dieses automatisierte Regal besteht aus mehreren Gassen, in denen die Behälter mit Produkten jeweils links und rechts zu maximal zweifach tief hintereinander gelagert sind. Da das Shuttle nur einen Behälter aufnehmen kann, kann nur der vorderste Behälter von links oder rechts entnommen werden. Das Shuttle kann sich innerhalb einer Gasse vor und zurück bewegen, sowie am vorderen Ende zwischen den Gassen wechseln oder zum Arbeitsplatz fahren. Diese Fahrt erfolgt jedoch langsamer als die Fahrt in den Gassen.

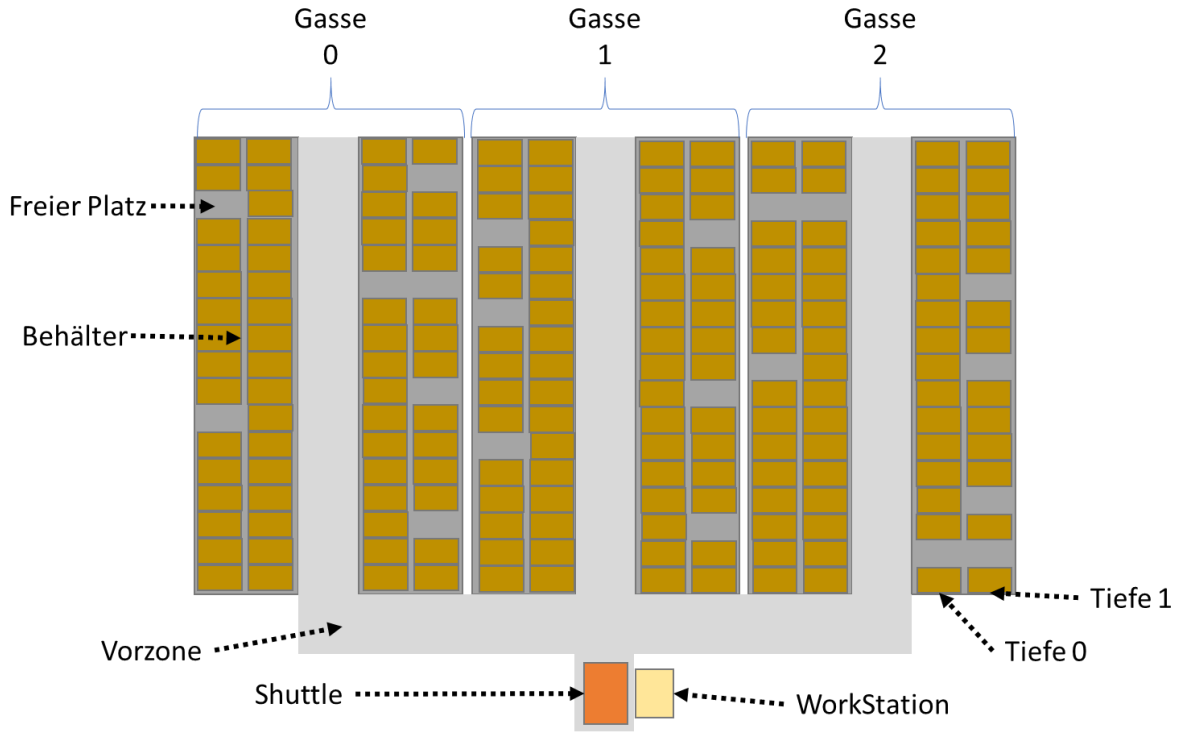

Abbildung 1 - Struktur OSR Shuttle™ Evo

AUFGABE

Deine Aufgabe ist es, ein Software-Modul zu entwickeln, dass die vorgegeben Aufträge abarbeitet. Die insgesamten Kosten zum Abarbeiten der Aufträge die durch Bewegung des Shuttles entstehen sollen dabei so gering wie möglich sein.

ERLÄUTERUNGEN

Produkt

Produkte (*Product*) sind Waren, die gehandelt werden und voneinander unterscheidbar sind. Ein rotes Hemd, Größe 42 ist ein Produkt, es unterscheidet sich von einem roten Hemd Größe 43.

Ein Produkt hat einen Code (*ProductCode*), der es eindeutig identifiziert.

Behälter

Ein Behälter (*Container*) dient zur Aufbewahrung und zum Transport von Produkten im Lager. In diesem Beispiel ist immer nur ein Produkt pro Behälter in einer bestimmten Stückzahl gelagert. Die Stückzahl in den Behältern, auch für dasselbe Produkt ist unterschiedlich.

Ein Behälter hat einen Code (*ContainerCode*), der ihn eindeutig identifiziert.

Lagerort

Auf einem Lagerort (*Location*) in einem Regal können ein oder zwei Behälter abgestellt werden. Das Verschieben der Behälter beim Abstellen oder Entnehmen der Behälter erfolgt in diesem Beispiel automatisch. Wird ein zweiter Behälter abgestellt, so wird der bereits dort stehende Behälter automatisch nach hinten verschoben. Wird der vordere Behälter wieder entnommen, wird der hintere Behälter automatisch wieder nach vorne verschoben.

Jeder Lagerort befindet sich an einer bestimmten *Position* im Regal, diese Position benötigt das Shuttle um den Lagerort anzufahren. (*Siehe auch Abschnitt Aufbau*)

Auftrag

Ein Auftrag (*Order*) ist – in diesem Beispiel – ein Produkt in einer bestimmten Stückzahl das aus dem Lager an Kunden ausgeliefert wird.

Shuttle

Ein Shuttle bewegt Behälter innerhalb des Regals oder aus dem Regal zum Arbeitsplatz. Dazu kann es sich innerhalb der Gasse vor und zurück bewegen, die Gassen wechseln, sowie Behälter von einem Lagerort aufnehmen und abgeben.

Das Shuttle kann aus einem Lagerort nur den vordersten Behälter entnehmen.

Abbildung 2 - OSR Shuttle™ Evo

Workstation

An der Arbeitsstation können aus dem Behälter auf einem dort befindlichen Shuttle Produkte für einen Auftrag entnommen werden.

ABLAUF

Die anstehenden Aufträge müssen der Reihenfolge nach abgearbeitet werden und die für den Auftrag benötigte Menge aus Behältern im Lager an der Arbeitsstation kommissioniert werden. Die Aufträge sollen dabei mit so wenig Kosten wie möglich abgearbeitet werden. Diese Kosten entstehen durch die Arbeiten des Shuttles.

Für jeden Auftrag müssen ein oder mehrere Behälter ausgewählt und nacheinander zur Arbeitsstation gebracht werden.

An der Arbeitsstation erfolgt die Entnahme. Wenn der Behälter nicht mehr benötigt wird, so muss dieser wieder eingelagert werden, auch wenn er leer ist.

Folgende Schritte sind (in der Regel) notwendig um Produkte aus einem Behälter zu kommissionieren:

1. Shuttle zum Lagerort bringen von dem der Behälter entnommen werden soll (Shuttle::*MoveToPosition*)
2. Behälter auf das Shuttle laden (*Shuttle::LoadFromLocation*)
3. Shuttle zum Arbeitsplatz bringen (*Shuttle::MoveToPosition*)
4. Produkte kommissionieren (*Workstation::Pick*)
5. Shuttle zum Lagerort bringen an dem der Behälter wieder gelagert werden soll (*Shuttle::MoveToPosition*)
6. Behälter auf Lagerort stellen (*Shuttle::StoreToLocation*)

Es kann vorkommen, dass ein Auftrag mehr Produkte benötigt als sich in einem einzelnen Behälter befinden. In diesem Fall müssen mehrere Behälter für den selben Auftrag nacheinander aus dem Regal geholt werden.

Es kann vorkommen, dass sich ein benötigter Behälter an der hinteren, nicht zugänglichen, Stelle an einem Lagerort befindet. In diesem Fall muss der Behälter, der sich an erster Stelle befindet, entnommen und an einem anderen Platz abgestellt werden, sodass der benötigte Behälter zugänglich wird.

Nach dem Abarbeiten des letzten Auftrages ist der letzte Behälter wieder einzulagern sodass das Shuttle am Ende leer ist.

Aufbau der Beispielanlage

Die Gasse besteht aus dem Fahrweg des Shuttles und den links und rechts davon angeordneten Lagerorten. Der Weg den das Shuttle dabei innerhalb der Gasse zurücklegt wird von der Vorzone der Gasse ausgemessen und als *DistanceIntoAisle* bezeichnet.

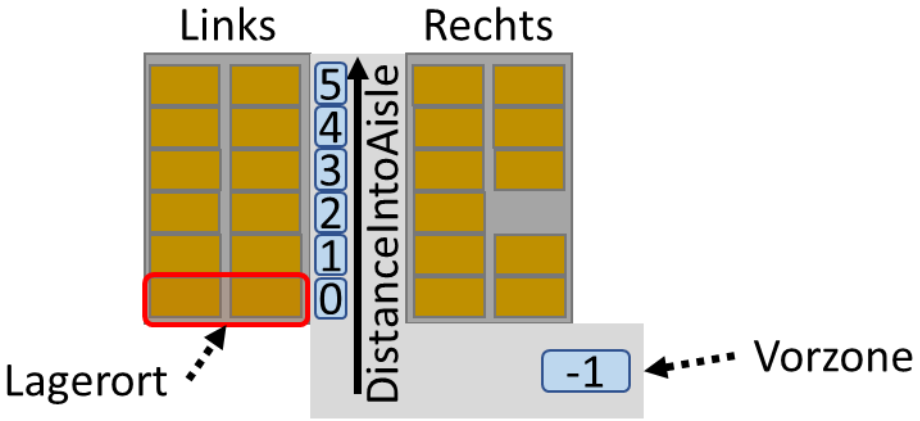

Abbildung 3 - Aufbau der Gasse

Die DistanceIntoAisle steigt dabei, beginnend mit 0 für die ersten Lagerorte an der Vorzone, pro Lagerort um eins an. Das Shuttle kann auf Positionen gesteuert werden, das ist die Kombination aus Gasse, DistanceIntoAisle und links oder rechts. Dabei kann das Shuttle immer die Lagerorte rechts und links vom Shuttle bedienen, unabhängig davon ob es auf die rechte oder linke Position bewegt wurde.

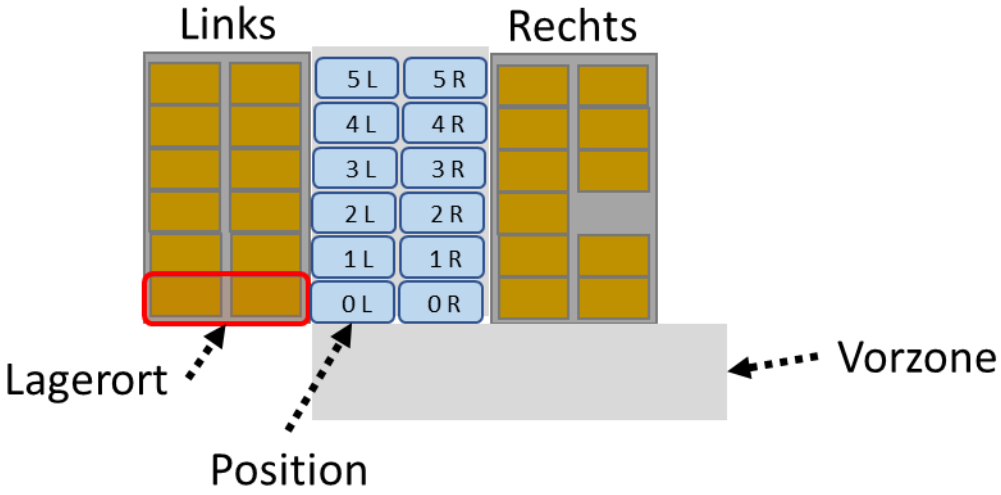

Abbildung 4 - Position in der Gasse

Die Beispielanlage in der Sandbox ist gegenüber der tatsächlichen Anlage stark vereinfacht und besteht aus diesen Komponenten:

Anzahl	Komponente	Anmerkung
1	Shuttle	
1	Arbeitsplatz	Vor Gasse Nummer 2
5	Gassen	Anzahl der Lagerorte pro Seite 300, daher max. DistanceIntoAisle von 299 und max. 1200 Behälter pro Gasse lagerbar

Kosten

Durch die Mechanik des Shuttles bedingt, sind die Kosten für Fahrten unterschiedlich. Folgende Kosten gelten für die einzelnen Bewegungen:

Bewegung	Kosten
Basiskosten wenn das Shuttle eine Distanz > 0 zurücklegt	1
Kosten für die Bewegung in der Gasse	1 * zurückgelegte Distanz (*DistanceIntoAisle*)
Kosten für das Wechseln der Gasse	50 * zurückgelegte Gassen
Laden oder Abstellen eines Behälters	10
Kosten für nicht bearbeiteten Auftrag	5000
Kosten für nicht leeres Shuttle am Ende	5000

Hinweis: Die Strecke zum Wechseln der Gassen befindet sich auf der Distanz -1.

Beispiel

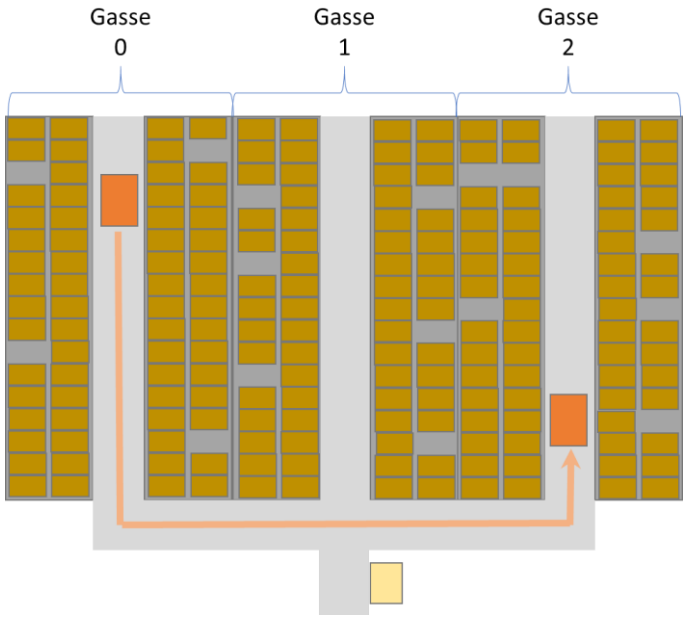

Abbildung 5 - Kosten am OSR Shuttle™ Evo

Das Shuttle fährt von Gasse 0, DistanceIntoAisle 14 in die Gasse 2, DistanceIntoAisle 4.

Teilbewegung	Kosten
Gasse 0/ DIA 14 → Vorzone	1+15
Gasse 0 → Gasse 2	1 + (2*50)
Vorzone → Gasse 2/ DIA 4	1+5
Gesamt	***123***

- Es werden nur auf den Bewertungsserver hochgeladene Ergebnisse gewertet.
- Die Resultate werden am Server mit den dort hinterlegten Algorithmen errechnet.
- Die Punkte errechnen sich aus
    - Den verursachten Kosten
        - Nicht bearbeitete Aufträge werden mit Zusatzkosten bewertet
        - Wenn das Shuttle am Ende nicht leer ist werden Zusatzkosten eingerechnet
    - Dem Abgabezeitpunkt

Die Gewichtung zwischen Abgabezeitpunkt und Kosten ist dabei so gewählt, dass in der Regel eine bessere Lösung auch bei späterer Abgabe ein besseres Ergebnis bedeutet.

Bei mehreren Abgaben (Uploads) zählt immer die beste Abgabe zu dem Zeitpunkt an dem diese getätigt wurde. Wenn Du nach dem Upload einer Lösung weiterarbeitest und durch Deine Änderungen das Ergebnis schlechter wird, bleibt die zweite Abgabe unberücksichtigt.

ABGABEMODUS

Zur Beurteilung des Ergebnisses muss die vom KNAPP-Code automatisch erzeugte Datei `upload2018.zip` über die Abgabeseite hochgeladen werden. In diese Datei wird Dein Source automatisch mitverpackt. Dieser Source muss das Ergebnis erzeugt haben. KNAPP behält sich hier Kontrollen vor.

Du kannst alle Code Teile modifizieren, die Ergebnisdatei (`warehouse-operations.csv`) wird von uns interpretiert und darf daher im Format nicht verändert werden.

UPLOAD WEBSITE

Unmittelbar nach dem Upload erhältst Du ein detailliertes Feedback zu deiner Abgabe.

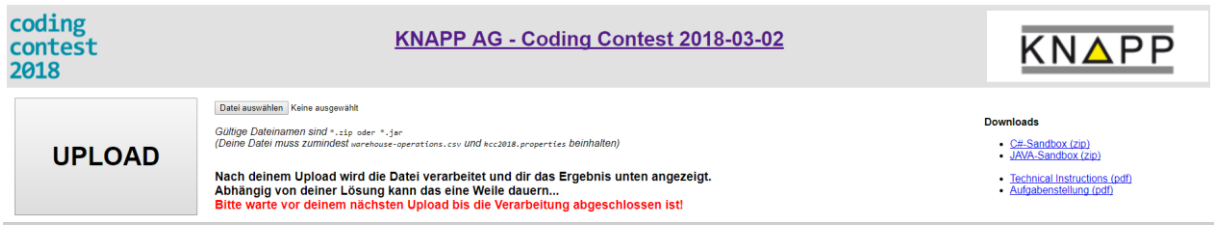

Abbildung 6: Upload-Server (Beispiel)

Im Feedback siehst Du die Probleme, die noch in deiner Abgabe vorhanden sind, und deinen Score für diesen Upload. Je *niedriger* dieser Wert ist, desto besser ist das Ergebnis.

TIPPS

- Versuche mit Deiner Software das Ergebnis zuerst mit einfachen Strategien zu verbessern und arbeite danach an weiteren Optimierungen.
- Schau Dir den von uns vorgegeben Code an und prüfe, ob Du Teile wiederverwenden oder erweitern kannst.
- Du kannst alle Teile des Source-Codes verändern. Die Abgabedatei muss jedoch dem vorgegebenen Format entsprechen.
- Lade öfters hoch – es wird nur die beste Abgabe bewertet.
- Die Lösung ist von Anfang an gültig und kann hochgeladen werden. Die Kosten – und damit Dein Score – werden mit jedem kommissionierten Behälter besser.

CODE-DETAILS

Folgende Daten stehen Dir von Beginn an zur Verfügung:

1. alle Aufträge (*orders*)
2. alle Behälter und deren Inhalte (*containers*)
3. alle Lagerorte (*locations*)

Dein Startpunkt ist die Methode **`RunWarehouseOperations()`** in der Klasse **`solution`**::**`Solution.`**

Die wichtigsten Methoden:

`Warehouse::GetOrders()` – liefert alle Aufträge in der Reihenfolge in der sie abgearbeitet werden müssen.

`Warehouse::GetAllContainers()` – liefert alle Behälter die im System vorhanden sind.

`Shuttle::MoveToPosition()` – bewegt das Shuttle zu einem Lagerort oder zum Arbeitsplatz.

`Shuttle::LoadFromLocation()` – Lädt den Behälter an der vordersten Position des Lagerortes auf das Shuttle. Wenn dahinter noch ein Behälter steht, so wird dieser automatisch an Position 0 vorgezogen.

`Shuttle::StoreToLocation()` – Stellt den Behälter vom Shuttle auf den Lagerort an Position 0. Wenn sich schon ein Behälter dort befindet, wird dieser auf Position 1 verschoben.

`WorkStation::PickOrder()` – Kommissioniert Produkte aus dem Behälter am Arbeitsplatz für den aktuellen Auftrag. Die Stückzahl die dabei kommissioniert wird, wird von unserem Code errechnet und ist entweder die im Auftrag offene Stückzahl oder die Anzahl der Produkte im Behälter, je nachdem was niedriger ist.

`Warehouse::CalculateMoveCosts()` – Berechnet die Kosten für eine Shuttlebewegung zwischen zwei Positionen.

`Warehouse::WarehouseCharacteristics` – Enthält Information über die Beschaffenheit des Lagers.(Java: `Warehouse::getWarehouseCharacteristics()`)

KLASSENDIAGRAMM

Im Klassendiagramm ist die Struktur der wichtigsten Klassen dargestellt. In den Klassen selbst sind die wichtigsten Methoden und Attribute dargestellt. Das Diagramm folgt dem Java-Source. In C# sind einige Methoden, dem Stil der Sprache folgend, durch *Properties* ersetzt. Ebenso sind in C# Methodennamen immer großgeschrieben.

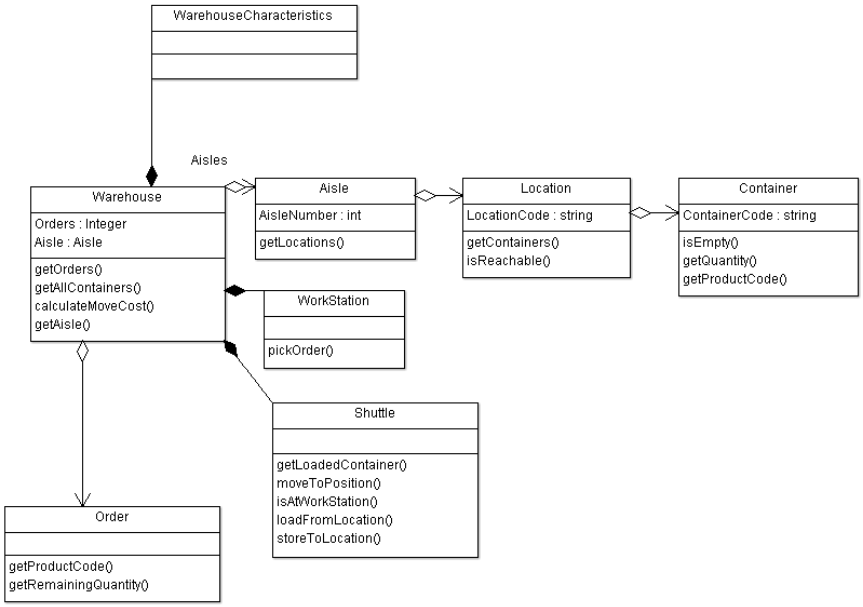

Abbildung 7 - Klassendiagramm