VL9: Von ASCII nach Postscript - Teil 2

(Noch nicht fertig)

Inhalt

1. Einleitung

Im bisherigen Verlauf der Lehrveranstaltung wurde in stark vereinfachter Form verdeutlicht, was unter einem Softwareprojekt im Rahmen des MDA-Ansatzes der OMG zu verstehen ist. Das allgemeine Konzept wurde am Beispiel einer konkreten Problemstellung konkretisiert und illustriert. In diesem Zusammenhang sind auch erste Grundelemente der Programmiersprache C++ eingeführt worden. Im Rahmen der Anwendungsbeispiele sollen nun weitere Elemente der Programmiersprache C++ erläutert werden.

2. Aufgaben

Für die weiteren Überlegungen nehmen wir an, dass eine einfache Aufgaben gelöst werden soll (vgl. Bild 1). Ein Programm begrüsst einen Anwender, fragt nach dem Namen einer ppmp2ps-Datei, liest diese Datei ein, analysiert den Inhalt, holt sich passende PS-Informationen aus einer Tabelle, baut dabei einen PS-text zusammen und speichert diesen dann ab. In diesem Beispiel wird nicht auf mögliche Fehler geachtet (siehe dazu VL 7).

Daraus ergeben sich folgende ***typische Aufgaben*** (die z.T. schon in VL 5+6 behandelt worden sind):

1. Anzeigen auf Bildschirm (siehe VL 5+6)
2. Eingabe über Bildschirm (siehe VL 5+6)
3. Lesen einer Datei (siehe VL 5+6)
4. Analyse eines Textes
5. Nachschlagen von Informationen
6. Zusammenbau eines neuen Textes
7. Schreiben einer Datei (siehe VL 5+6)

Zunächst sollen die Aufgaben 4-6 hier näher besprochen werden, da diese noch nicht behandelt worden sind.

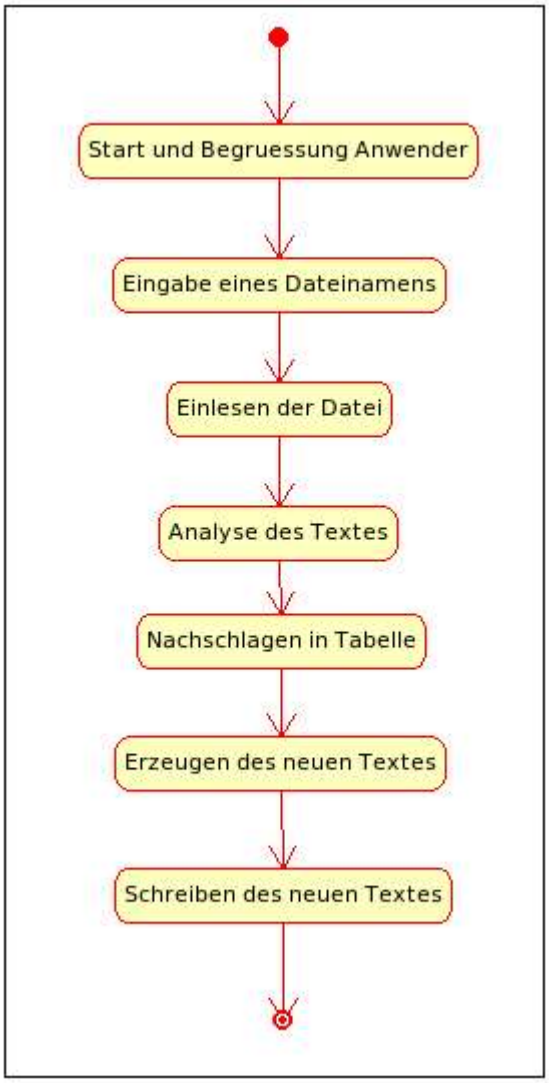

Bild 1: Übersicht über einfache Anwendung

3. Nachschlagen von Informationen

Allgemein soll ja ein ASCII-Text in ein PS-Programm übersetzt werden. Dies macht natürlich nur Sinn, wenn der ASCII-Text Sachverhalte beschreibt, die sich als PS-Programme darstellen lassen. Und da PS-Programme *Zeichnungen* darstellen, sind also nur solche Sachverhalte interessant, die sich *zeichnen* lassen. Dabei wurde schon in der letzten VL klar, dass man sich zur Beschreibung von Sachverhalten auf eine bestimmte *Beschreibungssprache* festlegen sollte.

3.1. Festlegung einer Beschreibungssprache

Für das Beispiel mit den Verkehrssituationen hatten wir folgende einfache Sprache diskutiert:

Auto Car1 auf Strasse S1 Richtung Osten bei Markierung M5

Auto Car2 auf Strasse S2 Richtung Norden zwischen Markierung M3 und M4

Dies könnte man in einer einfachen kleinen *Grammatik* wie folgt weiter präzisieren:

ObjTyp	***ObjID***	***befindet sich auf***	***Strasse***	***StrID***	***zwischen***	***Marke***	***und***	***Marke***	***in Richtung***
● Pkw ● Fahrrad ● Motorrad ● Fussgänger ● Lkw ● ...	Name oder Kürzel	befindet sich auf	Strasse	Name oder Kürzel	zwischen	Name oder Kürzel	und	Name oder Kürzel	● Norden ● Osten ● Süden ● Westen

Denkbar wären zusätzlich auch noch Operationen, durch die die Objekte ihre *Position verändern*. Dies wird hier noch nicht berücksichtigt.

Ausserdem sei angemerkt, dass man im allgemeinen an dieser Stelle eine *formale Grammatik* definiert, durch die eine Zielsprache definiert wird. Dies ist Gegenstand einer anderen Vorlesung. Hier werden im weiteren Verlauf nur *konkrete Sprachbeispiele* benutzt. Für die nächsten Beispiele wird angenommen, es gibt in der zu bschreibenden Welt nur Strassen *von Westen nach Osten*. Der *Name einer Strasse* steht immer direkt bei der obersten Begrenzung am westlichen Startpunkt. *Markierungen* verlaufen immer an der unteren Begrenzung, alle 20 Pixel. In der Welt gibt es *nur Pkws* die entweder nach Osten oder nach Westen fahren. Folgende Beschreibungen sind also möglich:

PKW HU-MU 495 befindet sich zwischen Markierung M3 und M3 in Richtung Westen.

PKW F-L 333 befindet sich zwischen Markierung M1 und M2 Richtung Osten.

3.2. Datenstrukturen für Informationen

Nachschlagen von Informationen bedeutet im vorliegenden Fall, dass zu einem identifizierten Element der Beschreibungssprache (z.B. PKW HU-MU 495, Markierung M3, ...) ein entsprechendes Element der *Bildsprache* (hier die PS-Sprache) nachgeschlagen werden kann. Es handelt sich also um die allgemeine Struktur:

Beschreibungssprache---> Bildsprache

PKW HU-MU 975 ---> PS-Text

Für eine solche Aufgabenstellung haben sich eine Reihe von Standard-Lösungen herauskristallisiert. Im Rahmen der STL von C++ (siehe: http://www.stlport.org/ sowie http://www.sgi.com/tech/stl/) können z.B. folgende genannt werden:

- Vektor (Element von *Sequence Container*)
- Liste (Element von *Sequence Container*)
- String
- Map (Element von *Associative Container*)
- Hash-Tabelle (Noch nicht Teil des STL-Standards, aber in neueren STL-Portierungen implementiert (siehe Lit.Liste der PPmP-Homepage)

Um diese Elemente aus der STL nutzen zu können, müssen allerdings erst einige technische C++-Details erklärt werden, damit man diese Elemente verstehen kann. Neben den Begriffen *Referenz*, *Konstruktor* und *Dekonstruktor* sind dies vor allem die Begriffe *Template* und *Iterator*.

3.3 Referenzen

Als wichtiges zusätzliches Hilfsmittel in C++ gegenüber C gilt das Sprachelement *Referenz* (siehe

http://www.fbmnd.fh-frankfurt.de/~doeben/I-PROGR2/I-PROGR2-TH/VL1/i-progr2-th-vl1.html#Referenzen). Die Referenz wird sehr oft kombiniert mit dem Sprachmittel *const,* so dass man den Ausdruck *...const TYP& var_name... findet.*

3.4 Konstruktoren und Destruktoren

Desweiteren kann man im Kontext von Klassen von den speziellen Operatoren *Konstruktor* und *Dekonstruktor* Gebrauch machen. Erklärungen mit Beispielen finden sich unter :

http://www.fbmnd.fh-frankfurt.de/~doeben/I-PROGR2/I-PROGR2-TH/VL5/i-progr2-th-vl5.html . Dort wird auch erklärt, was *Copy-Konstruktoren* und *Assignment-Konstruktoren* sind.

3.5 Templates (Schablonen)

Eine Erläuterung zum Begriff *Template* findet sich unter

http://www.fbmnd.fh-frankfurt.de/~doeben/I-PROGR2/I-PROGR2-TH/VL11/i-progr2-th-vl11.html .