

BlockTag 1: Aufgaben & Lösungen

(SEKS-Aufgabe, WS2009, 2009-12-06)

– TEIL 1 –

Drücken Sie im Listener von Factor die Taste F1. Es erscheint ein neues Browser-Fenster mit der Factor-Dokumentation. Unter „Getting started“ findet sich ein Tutorial „Your first program“. Arbeiten Sie alle vier Teile dieses Tutorials durch.

Stoßen Sie im Lösungstext auf z.B. ein „→ vocabularies“, dann erhalten Sie in der Factor-Dokumentation unter diesem Schlagwort mehr Informationen. Steht der Begriff in Anführungszeichen, wie z.B. bei → „USE“, dann bezieht sich der Verweis auf die Wortdefinition und nicht auf einen Artikel.

Your first program

Was ist ein Vokabular (*vocabulary*)? – Eine Sammlung oder auch eine Menge von Wörtern, → vocabularies.

Was macht das Scaffold-Tool? – Das englische Wort „scaffold“ heißt soviel wie „Gerüst“. Das Scaffold-Tool legt im Arbeitsverzeichnis „work“ in der Factor-Distribution für ein neues Vokabular ein namensgleiches Verzeichnis und eine .factor-Datei an, die minimal vorbereitet zur weiteren Arbeit ist → tools.scaffold. Weitere Dateien können je nach Konfiguration zum „Gerüst“ dazukommen. Typischerweise werden zu einem Vokabular noch zwei zusätzliche Dateien angelegt: eine Datei zur Dokumentation der Wörter eines Vokabulars, per Konvention <vocname>-docs.factor, und eine Datei zum automatisierten Test von Wörtern des Vokabulars, <vocname>-tests.factor. Beachten Sie, dass selbst die Dokumentation ein Factor-Programm ist!

Was machen die Wörter „USE:“ und „USING:“? Worin unterscheiden sie sich? – Mit „USE: vocabulary“ wird das angegebene Vokabular zum Suchpfad bei der Auflösung von Wörtern hinzugefügt (→ „USE“). Mit „USING: vocabularies ... ;“ geschieht dasselbe für die Folge der angegebenen Vokabulare. Ergänzender Hinweis: Wie sie schon wissen, sind Vokabulare in Verzeichnissen organisiert. Die .-Notation bei Vokabularen bildet sich auf verschachtelte Verzeichnisse ab. Siehe dazu → vocabs.loader.

In welchen Wurzel-Verzeichnissen (*root directories*) sucht Factor nach Vokabularen, um ein Wort aufzulösen? – Factor sucht standardgemäß in den Verzeichnissen „core“, „basis“, „extra“ und „work“. In „core“ finden sich elementare Vokabulare, die das Kernsystem ausmachen. Das Kernsystem wird um eine Reihe nützlicher Bibliotheken und Werkzeuge in „basis“ erweitert. Mit „extra“ stehen weitere Bibliotheken zur Verfügung. In „work“ finden sich in der Regel individuell erstellte Vokabulare, → vocabs.roots.

Wie lautet das Standard-Vokabular (*default vocabulary*) für die Definition neuer Wörter im Listener? – Im Listener ist „scratchpad“ das Standard-Vokabular, das neue Wortdefinitionen aufnimmt.

Was macht das Wort „IN:“? – Ein „IN: vocabulary“ gibt das Vokabular an, welches neue Wortdefinitionen aufnimmt. In .factor-Dateien ist ein „IN:“-Wort unvermeidbar, da es dort kein scratchpad-Vokabular gibt, → „IN:“.

Was macht das Wort „reload“ (Stackeffect und Definition)? – Das Wort „reload (name –)“ lädt erneut den Quellcode und die Dokumentation des angegebenen Vokabulars , → „reload“.

Mit welchem Wort ist vermutlich die Taste F2 assoziiert? – Mit der F2-Taste das Wort „refresh-all“ (→ vocabs.refresh) („Runtime code reloading“) assoziiert.

Was macht das Wort „:“? – Mit diesem Wort werden neue Wörter im aktuellen Vokabular definiert, → „:“.

Was ist ein Stack-Effekt? – Zur Definition von Wörtern mittels „:“ und „GENERIC:“ gehört die Angabe der Auswirkungen auf den Datenstack: wieviele Datenwörter entfernt das Wort vom Stack und wieviele Datenwörter legt es auf dem Stack ab, → effects.

Es gibt eine Konvention zu Wörtern, die mit einem Fragezeichen „?“ enden. Wann wird das Fragezeichen verwendet? – Ein Wort, dessen Name mit einem Fragezeichen endet, ist ein sogenanntes Prädikat: Es hinterlässt per Konvention auf dem Stack einen booleschen Wert.

Wenn Sie in einer Datei Wörter in einem Vokabular definieren, ohne ein „USE:“ oder „USING:“ zu verwenden, dann werden Sie vermutlich eine Fehlermeldung beim Laden des Vokabulars erhalten. Warum? – Wenn Factor eine Datei einliest, kennt es ausschließlich die Wörter des „syntax“-Vokabulars. Da Wörter in aller Regel durch eine Folge anderer Wörter definiert werden, müssen die in den Definitionen verwendeten Wörter über „USE:“ bzw. „USING:“ bereit gestellt werden, → word-search. Alles was Factor zu Beginn kann ist Parsen! Weitere Fähigkeiten müssen über Vokabulare bereit gestellt werden.

Wie finden Sie heraus, zu welchem Vokabular ein Wort gehört? – Der Browser von Factor gibt darüber Auskunft. Dort ist zu jedem Wort die Zuordnung zu einem Vokabular verzeichnet. Eine Möglichkeit ist, das Search-Formular im Browser zu nutzen; eine andere ist, im Listener das Wort einzutippen und C-h zu drücken.

Wenn das Wort „f“ für false steht, was ist dann der Wert für true? – Das f-Wort ist ein Parsing-Wort, dem ein f-Objekt entspricht. Es ist das einzige Objekt, das nicht true ist. Es gibt also keinen ausgezeichneten true-Wert in Factor. Allerdings gibt es mit „t“ einen kanonischen (als Richtschnur geltenden) true-Wert, → booleans.

Wie ist ein „unit-test“ (sprich das Wort „unit-test“) definiert? – Die Syntax ist definiert als „[output] [input] unit-test“. Für weitere Details, → „unit-test“.

Wie werden die Unit-Tests zu einem Vokabular aufgerufen? – Der Aufruf erfolgt mit „<vocname> test“.

– TEIL 2 –

Nachdem Sie Ihr erstes Factor-Programm erstellt haben, arbeiten Sie bitte die 10 Artikel unter „Factor cookbook“ durch.

Basic syntax cookbook

Factor unterscheidet Wörter von Literalen: Was ist ein Literal, was ein Wort? – Ein Literal ist eine Notation zur Darstellung von Datenwerten. Üblich und verbreitet sind Integer-Literale (wie z.B. 7, -13) und String-Literale (wie „Hello“). In Factor gibt es darüber hinaus eine Literal-Syntax unter anderem für Quotierungen „[...]“, für Arrays „{ ... }“, für Vektoren „V{ ... }“, einzelne Zeichen (*characters*), Hashtables „H{ ... }“ usw., → syntax-literals. Factor realisiert die Literal-Syntax über sogenannte Parsing-Wörter – dazu wird es eine eigene Übungseinheit geben. Die Literale werden in entsprechende Objekte umgewandelt. In der

Terminologie (Sprechweise) von Factor ist ein Wort das, was in anderen Sprachen einer Funktion oder einer Prozedur entspricht. Im Grunde ist alles, was kein Literal ist, potentiell ein Wort. Allerdings muss das Wort als Name in einem Vokabular definiert sein. Die Syntax zu einem Wort entspricht nicht seinem Datenwert, weshalb ein Wort kein Literal ist. Beachten Sie, dass wir bei der mathematischen Definition von Wörtern diese Unterscheidung nicht gemacht haben. Die Unterscheidung in Wörter und Literale ergibt sich vorrangig aus einer Parse-Sicht.

Was ist ein „primitives Wort“? – Ein „primitives Wort“ (*primitive*) ist ein Wort, das die Factor VM direkt zur Verfügung stellt und nicht aus anderen Wörter über eine Definition zusammengesetzt ist, → primitives.

Wie ist die Datenstruktur des Stacks definiert? (Das ist eine allgemeine Informatik-Frage.) – Erinnern Sie sich an GDI2, dort haben wir die Datenstruktur eines Stacks definiert!

Shuffle word and definition cookbook

Warum werden manche Wörter als *shuffle words* bezeichnet? – Die sogenannten *shuffle words* verändern das Arrangement der Daten auf dem Datenstack. Beispiele sind „drop“, „dup“, „over“ und „swap“.

Definieren Sie ein Wort „double“, das den Wert der obersten Zahl auf dem Stack verdoppelt! – „: double (n -- 2*n) dup + ;“

Control flow cookbook

Was ist eine Quotierung (*quotation*)? – In Factor ist eine Quotierung eine anonyme Funktion. Was damit gemeint ist wird klarer, wenn Sie sich vor Augen führen, dass jede Wortdefinition nichts anderes ist als eine Abbildung von einem Wortnamen zu einer Quotierung, die den Rumpf der Wortdefinition enthält. Dem Aufruf eines Wortes entspricht die Abarbeitung der Wörter in der Quotierung; sie definieren die Funktion des Wortes. Man spricht von einer solchen Quotierung auch als „named quotation“. Eine Quotierung außerhalb des Kontextes einer Wortdefinition ist eine „unnamed quotation“, eine anonyme Funktion, → quotations.

Erläutern Sie die Arbeitsweise des Wortes „sign-test“, das wie folgt definiert ist:

```
: sign-test ( n -- )
  dup 0 < [
    drop "negative"
  ] [
    zero? [ "zero" ] [ "positive" ] if
  ] if print ;
```

Der oberste Wert auf dem Stack wird dupliziert. Das Duplikat wird daraufhin getestet, ob es größer oder kleiner als Null ist; ein boolescher Wert, „t“ oder „f“, ist das Ergebnis. Dann folgen zwei Quotierungen und ein „if“. Ist das Duplikat kleiner als Null, wird die erste Quotierung, ansonsten die zweite Quotierung ausgeführt. Die erste Quotierung entfernt den Ausgangswert vom Stack („drop“) und legt auf dem Stack den String „negative“ ab. Die zweite Quotierung testet, ob der Ausgangswert gleich Null ist. Danach folgen zwei Quotierungen und ein „if“. Ist der Wert Null, wird der String „zero“, sonst „positive“ abgelegt. Zuguterletzt wird das Resultat („negative“, „zero“ bzw. „positive“) vom Stack genommen und über die Konsole ausgegeben.

Wofür steht „{ 1 2 3 }“? – Es handelt sich um ein Array mit den Werten 1, 2 und 3.

Was macht „map“? (Anders gefragt: Wie ist „map“ definiert? Als Antwort wird mehr als die Angabe des Stack-Effects erwartet.) – Das Wort „map (seq quot – newseq)“ wendet die Quotierung auf jedes einzelne Element der Sequenz an. Die Ergebniswerte bilden eine neue Sequenz.

Definieren Sie ein Wort „positive? (n – ?)“. Filtern Sie mit Hilfe dieses Wortes aus der Sequenz „{ -12 10 16 0 -1 -3 -9 }“ die positiven Zahlen heraus.

```
: positive? ( n -- ? ) 0 > ;
{ -12 10 16 0 -1 -3 -9 } [ positive? ] filter ==> { 10 16 }
```

Dynamic variables cookbook

Was ist ein Symbol? – Ein Symbol ist ein Wort, das sich selbst auf dem Stack ablegt, wenn es aufgerufen wird.

Was ist ein Namensraum (*namespace*)? – Ein Namensraum entspricht einer Assoziationen von Schlüssel (keys) zu Werten (*values*); man spricht auch von Schlüssel/Wert-Bindungen. Per Konvention werden in Factor Symbole als Schlüssel verwendet. Die Verschachtelung von Namensräumen (*scope nesting*) wird über einen Namensstapel (*namestack*), genauer: einen Vektor von Namespaces, abgebildet, → namespaces.

Was sind „dynamically-scoped variables“? – Variablen, deren aktueller Wert von der letzten Änderung zur Laufzeit abhängt (und nicht von einem lexikalischen Blockkontext) sind „dynamische Variablen“. Der aktuelle Wert wird in einem implizit mitgeführtem Namensraum (und eben nicht explizit auf dem Stack) vorgehalten.

Vocabularies cookbook

Inwiefern unterscheidet sich bezüglich verfügbarer Vokabulare die Arbeit im Listener von der Arbeit mit einer Programmdatei? – Im Listener sind standardgemäß bereits eine Reihe von Vokabularen geladen. In einer Programmdatei müssen die verwendeten Vokabulare explizit angegeben werden.

Was hat es mit dem Wort „DEFER:“ auf sich? – Wörter müssen definiert sein bevor sie im Programmcode verwendet werden können. Es gibt Ausnahmen, wie z.B. bei wechselseitig rekursiv definierten Wörtern, bei denen eine Wortdefinition vor der Verwendung des Worts nicht möglich ist. Hier hilft das Wort „DEFER:“ aus! Es erlaubt die nachträgliche Definition eines Wortes.

Application cookbook

Wofür ist das Wort „MAIN:“ gut? – Ein Vokabularname, der mit „run“ aufgerufen wird, beginnt die Ausführung mit dem durch „MAIN:“ ausgezeichneten Wort.

Wofür ist das „deplay-tool“ gut? – Es dient der Erstellung von eigenständigen Ausführungsdateien zu einer Anwendung. Damit kann eine Anwendung ohne den Factor-Listener und ohne unnötigen Ballast durch andere Vokabulare ausgeliefert werden.



Scripting cookbook

Sie können dieses Cookbook überspringen. Es betrifft die Erstellung von Programmen, die geeignet sind als Werkzeuge für die Kommandozeile in Linux/Unix und – in etwas begrenzterem Maße – für Windows.

Factor philosophy

Was ist ein Kombinator (*combinator*)? – Ein Kombinator ist ein Wort, das eine Quotierung auf dem Stack erwartet. Die Quotierung kommt dabei in ihrer Funktion als „verzögerter Code“ (anonyme Funktion) zum Einsatz.

Pitfalls to avoid

Angenommen, auf dem Stack liegt eine Zahl: Was genau macht das Wort „dup“? – Die Frage zielt darauf ab sich in Erinnerung zu rufen, dass auf dem Stack nur Referenzen auf Objekte liegen. Das Wort „dup“ dupliziert die Referenz, nicht das Objekt. Das ist im Kontrast zur unserer mathematischen Definition einer konkatenativen Sprache: Dort werden immutable (nicht veränderbare) Werte, nicht Referenzen, auf dem Stack manipuliert.

Next steps

Aus diesem Artikel bearbeiten Sie bitte das Vokabular „roman“. Hier ist gefordert, dass Sie jede einzelne Zeile in roman.factor verstehen, mit Ausnahme der Zeilen, die durch „<<“ und „>>“ markiert sind.