X2X (Programmiersprache)

(aus Wikipedia am 22. August 2014)

X2X [gesprochen ex-to-ex] ist eine anwendungsspezifische, üblicherweise interpretierte höhere Programmiersprache.[3] Ihre Entwurfsphilosophie betont die einfache Anwendung zum Parsen und Generieren von Dateien mit textuellem Format.

X2X	
Paradigmen:	multiparadigmatisch
Erscheinungsjahr:	2012
Entwickler:	Josef Hübl
Aktuelle Version:	1.0.0 (01. Mai 2012)
Typisierung:	stark, dynamisch
Wichtige Implementierungen:	x2xGen [1]
Beeinflusst von:	C, C++, Pascal,EBNF
Lizenz:	Triple-S GmbH[2]
www.x2x.ch (http://www.x2x.ch//)	

Ein X2X-Programm gliedert sich in TYPE-, PATTERN- und SCRIPT-Dateien. TYPEN sind sowohl beschreibend für Datenstrukturen als auch Vorgaben zum Parsen von Dateien. Ein PATTERN ist ein Muster bzw. ein Template für eine Ausgabedatei und wird immer als Makro aufgerufen. SCRIPTE steuern den Programmablauf und dienen entweder als Programmeinsprungspunkt (MAINScript) oder werden als Makros verwendet.

X2X unterstützt mehrere Programmierparadigmen. Größtenteils kommt funktionale Programmierung zum Einsatz. Objektorientierte Programmierung wird nur eingeschränkt auf die Verwendung von Namensräumen unterstützt. Andererseits gibt es z. B. keine PRINTAnweisung; stattdessen werden Ausgaben in Form von sogenannten PATTERN (dt. Muster) nach dem WYSIWYG-Prinzip spezifiziert. Als Besonderheit erlaubt X2X Datentypdefinitionen die implizit einen Parser definieren. Wie andere dynamische Sprachen wird X2X als Skriptsprache genutzt.

Der Name der Sprache ist durch die Handelsmarke **X2X** geschützt[4], deren Inhaberin die Triple-S Strukturen-Software-Systeme GmbH ist und die die Definition der Sprache in der Referenzumsetzung x2xGen pflegt.

X2X gilt als einfach zu erlernende Sprache, da sie über eine klare und übersichtliche Syntax, sowie vergleichbar zu VBA über einen reduzierten, anwendungsspezifischen Sprachumfang verfügt.

Inhaltsverzeichnis

Entwicklungsgeschichte

Die Sprache wurde ab dem Jahr 2005 von Josef Hübl im Rahmen von industriellen Softwareprojekten[5] entwickelt und war ursprünglich für das automatisierte Übertragen von Softwarearchitekturen in C-Sourcecode gedacht. Bis 2012 wurde die Sprache für den Einsatz zur modellgetriebenen Softwareentwicklung und allgemeine textuelle Daten- und Dateitransformationen erweitert. Mit Veröffentlichung der Sprache wurde sie als Version 1.0.0 eingefroren.

Namensherkunft

Der Name X2X bezieht sich auf das Umformen von einem x-beliebigen Dateiformat in ein anderes x-beliebiges Dateiformat.

Ziele und Besonderheiten

X2X wurde mit dem Ziel entworfen, vergleichbar zu einem AUTOSAR-RTE-Generator, aus den textuellen Beschreibungen der Komponenten und Interfaces einer Softwarearchitektur den C Programmcode zu generieren. Insbesondere sollte dabei das Dateiausgabeformat vom Anwender nach dem WYSIWYG-Prinzip editierbar sein. Dies hatte zur Folge, dass die X2X Anweisungen ähnlich zu Präprozessor-Anweisungen in den Ausgabetext eingebettet werden. Als spätere Anforderung kam hinzu, dass auch die Formate der Eingabedatei durch den Anwender veränderbar sein sollten, wobei der zum Einlesen der Eingabedatei notwendige Parser zur Laufzeit gebildet werden sollte, damit der Anwender sich nicht mit den üblichen Schwierigkeiten von Parser-Generatoren auseinandersetzen musste. Als Folge davon werden nichtelementare Datentypen in einer von EBNF abgeleiteten Notation definiert.

Datentypen und Strukturen

X2X besitzt die klassischen elementaren Datentypen wie z. B. ::INTEGER, ::FLOAT, ::STREAM (Zeichenkette), ::BOOL. Neben der einer Ganzzahl- und

Gleitkommaarithmetik unterstützt X2X die Verarbeitung von Zeichenketten, wobei insbesondere gesamte Dateiinhalte als Zeichenketten behandelt werden.

Das Besondere an X2X sind eine Vielzahl von *Strukturtypen*, die in erster Linie zum Parsen verwendet werden. Der Strukturtyp ::SEQUENCE definiert ein Tupel und ::LIST beschreibt eine Liste. ::CHOICE steht für eine Auswahl, ::OPTION definiert das wahlweise Vorhandensein eines Strukturelements, ::DISTANCE kennzeichnet eine Zeichenkette die durch das nachfolgende Strukturelement beendet wird, usw. Für Listen stehen allgemeine Methoden zum Bearbeiten (z. B. alphabethisches Sortieren) zur Verfügung. Auch kann über die Elemente einer oder mehrerer Listen gleichzeitig iteriert werden. (Dies ist einer der häufigsten Anwendungsfälle.)

Die Datentypen werden dynamisch verwaltet und eine statische Typprüfung (wie z. B. bei C++) ist gegeben.

Syntax

Alle X2X Anweisungen beginnen und enden mit dem Zeichen '$'. Alle Zeichenfolgen außerhalb einer X2X-Anweisung werden als statischer Text in die Ausgabe kopiert.

`$/* ... */$` klammert einen Kommentar in C/C++ ähnlicher Notation.

Anders als bei herkömmlichen Programmiersprachen folgt bei der Deklaration einer Variablen der Datentyp dem Bezeichner der Variablen, wobei der dem Bezeichner des Datentypen ein doppelter Doppelpunkt '::' vorangestellt wird. (Z. B. deklariert `$MYVAR::INTEGER$` eine Variable mit dem Namen 'MYVAR' mit Datentyp ::INTEGER.)

Der einfache Doppelpunkt ':' wird dagegen als Namensraum-Trennzeichen verwendet. (z. B.: NAME1:NAME2:NAME3) In der Regel ergeben sich solche zusammengesetzte Namen aus der hierarchischen Struktur von Datentypen, die zum Parsen der Eingangsdateien verwendet werden. In X2X ist es zulässig Namensabkürzungen (z. B.: NAME1:~:NAME3) zu verwenden um den Sourcecode übersichtlich zu halten, wobei der Interpreter im Falle einer Mehrdeutigkeit eine Fehlermeldung ausgibt.

Besteht eine X2X Anweisung nur aus dem Namen einer Variablen (z. B. `$MYVAR$`), kennzeichnet dies einen Platzhalter, der in der Ausgabedatei durch den Wert der Variablen ersetzt wird.

Jeder Übergabeparameter eines Makros bzw. einer Funktion ist entweder mit '&' als Referenz oder mit '#' als konstante Referenz zu kennzeichnen. Letztere können innerhalb des Makros nur gelesen, aber nicht verändert werden. (Ein X2X Interpreter erzeugt von einem Übergabeparameter grundsätzlich keine Kopie, da davon ausgegangen wird, dass es sich im Allgemeinen um ein ganzen (u.U. sehr großen) Dateiinhalt handelt.)

In X2X unterscheidet man zwischen einfachen und Blockanweisungen, wobei Blockanweisungen durch `$BEGIN ... $` und `$END ... $` geklammert sind und beide Anweisungen bis auf die Schlüsselwörter 'BEGIN' und 'END' identisch sein müssen. (Insbesondere bei großen Blöcken trägt dies wesentlich zur Lesbarkeit bei.)

Ein besonderer Schwerpunkt von X2X ist auf die Verarbeitung von Listen gelegt. `$BEGIN ALL MYLIST$ .... $END ALL MYLIST$` ist z. B. eine typische Blockanweisung bei der über alle Elemente einer Liste (hier 'MYLIST') iteriert werden kann, d.h. innerhalb des Blockes sowohl lesender als schreibender Zugriff auf die Elemente der Liste besteht. `$BEGIN IF condition$ .... $END IF condition$` kennzeichnet einen

Block der nur dann ausgeführt wird, wenn die Bedingung 'condition' erfüllt ist, wobei 'condition' z. B. der numerische Vergleich zweier Variablen sein kann. Die Vergleichsoperatoren sind dabei wie in C/C++ über '<', '<=', '==', '!=', '>=' und '>' gegeben.

`else` und `switch` Anweisungen, wie man sie aus anderen strukturierten Sprachen wie C, Perl oder Pascal kennt, gibt es in X2X nicht.

Jeglicher X2X Programmcode muss durch die Blockanweisungen `$BEGIN X2X VERSION="1.0.0"$ ... $END X2X VERSION="1.0.0"$` geklammert sein, wobei sich die Versionsangabe "1.0.0" auf die Version der Sprachdefinition von X2X bezieht.

Beispiel

```
$BEGIN X2X VERSION="1.0.0"$
$BEGIN SCRIPT MY_EXAMPLE()$
$/*****************************************************************************
Dieses X2X Skript lädt den anwenderspezifische Datentypen und das Ausgabemuster,
parst die Eingabedatei mit dem Datentypen und erzeugt die Ausgabedatei mit
dem Muster
*****************************************************************************/$
$DIR::X2X:DIRECTORY$
$DIR:SET_PATH(".")$
$/* Laden des anwendungsspezifischen Datentypen */$
$::MY_EXAMPLE_type::X2X:TYPE = DIR:FILE("MY_EXAMPLE_type.x")$
$/* Laden des Ausgabemusters */$
$MY_EXAMPLE_pattern()::X2X:PATTERN = DIR:FILE("x.MY_EXAMPLE_pattern.txt.2x")$
$/* Parsen der Eingabedatei */$
$MY_DATA::MY_EXAMPLE_type = DIR:FILE("MY_EXAMPLE_input.txt")$
$/* Erzeugen der Ausgabedatei */$
$DIR:FILE("MY_EXAMPLE_output.txt") = (::X2X:FILE) MY_EXAMPLE_pattern( MY_DATA )$
$END SCRIPT MY_EXAMPLE()$
$END X2X VERSION="1.0.0"$
```

Obiges X2X Skript verwendet folgenden anwendungsspezifischen Datentypen zum Parsen der Eingabedatei:

```
$BEGIN X2X VERSION="1.0.0"$
$/*****************************************************************
Dieser Typ wird zum Parsen der die Eingabedatei verwendet
*****************************************************************/$
$TYPE ::MY_EXAMPLE_type = ::SEQUENCE {
'Begin of file'
MY_LIST::LIST{
MY_ID::IDENTIFIER MY_INT::INTEGER
}
REMAINDER::STREAM{}
}
$
$END X2X VERSION="1.0.0"$
```

Dies ist der Inhalt der Eingabedatei.

```
Begin of file
AAA 111
BBB 222
End of file
```

Dies ist das verwendete Ausgabemuster.

```
$BEGIN X2X VERSION="1.0.0"$
$BEGIN PATTERN MY_EXAMPLE_pattern( #DATA::MY_EXAMPLE_type )$
$/************************************************************
Mit diesem Muster wird der Inhalt der Ausgabedatei erzeugt
************************************************************/$
$
This is the output!
$BEGIN ALL DATA:MY_LIST$
my identifier = $DATA:MY_LIST:MY_ID$ my integer = $DATA:MY_LIST:MY_INT$
$END ALL DATA:MY_LIST$
$DATA:REMAINDER$
$
$
$END PATTERN MY_EXAMPLE_pattern( #DATA::MY_EXAMPLE_type )$
$END X2X VERSION="1.0.0"$</code>
```

Obiges Muster / Ausgabemakro erzeugt folgende Ausgabe:

```
This is the output!
my identifier = AAA my integer = 111
my identifier = BBB my integer = 222
End of file
```

Standardbibliothek

Die X2X Standardbibliothek verfügt lediglich über Funktionen zur Bearbeitung von Listen und Zeichenketten.

Implementierungen

x2xGen (Referenzimplementierung)

x2xEdit (Entwicklungsumgebung)

Entwicklungsumgebung

Ergänzend zum Interpreter x2xGen gibt es die Entwicklungsumgebung x2xEdit, die unter anderem auch Syntaxhighlighting bietet. x2xGen kann auch als Plugin in andere Entwicklungsumgebungen und Werkzeuge eingebunden werden. Insbesondere steht für Notepad++ ein eigenes AddOn für Syntaxhighlighting zur Verfügung[6].

Verbreitung und Einsatz

Ein typisches Anwendungsgebiet für X2X ist die Implementierung von Parser-Generator-Paaren zur Realisierung von modellgetriebener Softwareentwicklung, das Erzeugen oder Parsen von HTML-Seiten, CSV-Dateien und Dokumentation im RTF-Format, die Generierung von Sourcecode aus Excel-Arbeitsmappen, Erstellen von AUTOSAR-Komponenten-Generatoren, sowie allgemein das toolgestützte Umsetzen von industriellen Softwareentwicklungsprozessen (Stichworte: SPICE / CMMI) zur Implementierung von Embedded Systemen.

Kritik

In der Version 1.0.0 gibt es keine Möglichkeit Programme anderer Sprachen als Modul einzubetten.

Numerische Operationen stehen derzeit nur als unäre Operationen zur Verfügung.

Einige in anderen Sprachen gebräuchliche Kontrollstrukturen, wie `do`/`while`, sind in X2X nicht vorgesehen und müssen auf andere Weise realisiert werden.

Literatur

Triple-S GmbH: *X2X Online Dokumentation.* (http://www.sss.de/x2x-online-dokumentation) Abruf 8. August 2014

Triple-S GmbH: *X2X Grundlagen* (http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_11/X2X_Grundlagen.pdf) (Zum Download)

Referenzen

Heise Verlag: *X2X,X2X_Download* (http://www.heise.de/download/x2x-1192432.html) (Zum Download)

Weblinks

Offizielle X2X Website (http://x2x.sss.de) (deutsch)

X2X Referenzimplementierung (http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_2014/X2X_2014_Setup_1335_3841.exe) (Zum Download)

X2X Beispielanwendungen (http://www.sss.de/x2x-downloads) (Zum Download)

X2X Einführungsvideos (https://www.youtube.com/watch?v=bukqdh-3Tw0&list=PLevFD66G1f-BRRLiX4xpmuElkznxg5C_i&feature=mh_lolz)

Einzelnachweise

1. *x2xGen.exe (http://www.heise.de/download/x2x-1192432.html)*. In: *X2X_Download*. Heise Verlag. Abgerufen am 8. August 2014.
2. *Triple-S Homepage (http://www.sss.de)*. Triple-S GmbH. Abgerufen 2. am 8. August 2014.
3. *X2X Einführung (http://www.sss.de/x2x-online-dokumentation)*. In: *X2X Online Dokumentation*.Triple-S GmbH. Abgerufen am 8. August 2014.

4. *X2X Trademark No. 011131679 (https://oami.europa.eu/eSearch/#basic/1+1+1+1/X2X)*. Office for Harmonization in the Internal Market. Abgerufen am 8. August 2014.
5. *X2X Einsatz bei SiemensVDO (http://www.sss.de/empfehlung)*. SiemensVDO (jetzt: Continental AG). Abgerufen am 8. August 2014.
6. *X2X in Notepad++ (http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_2014/X2X_Notepad_plus_plus.zip)*. Chris Covie, Sourceforge. Abgerufen am8. August 2014.