HuBoTUC (humanoid robots of technical university of clausthal)

Reengineering des originalen Maschinenprogramm

Zhao Dalong 326724

Zhang Wenyong 333852

Technische Universität Clausthal
Institut für Informatik

Inhalt

- Robonova-I und RoboBasic
- Interpreter
- Objektcodes erkennen
- Algorithmen
- Programm und Testbeispiel
- Probleme des Programms
- Zusammenfassung

Robonova-I und RoboBasic

- Robonova-I: Roboter mit 16 Gelenken von HiTEC

Robonova-I und RoboBasic

- RoboBasic: Programmierumgebung zur Steuerung von Robonova-I.

- Nachteile von RoboBasic:

  -Beschränkungen in Sprachestruktur

  Eine einfache Lösung wäre die Erweiterung mit den Strukturen wie While-Schleife und Switch.

  -Kompatibilitätsproblem

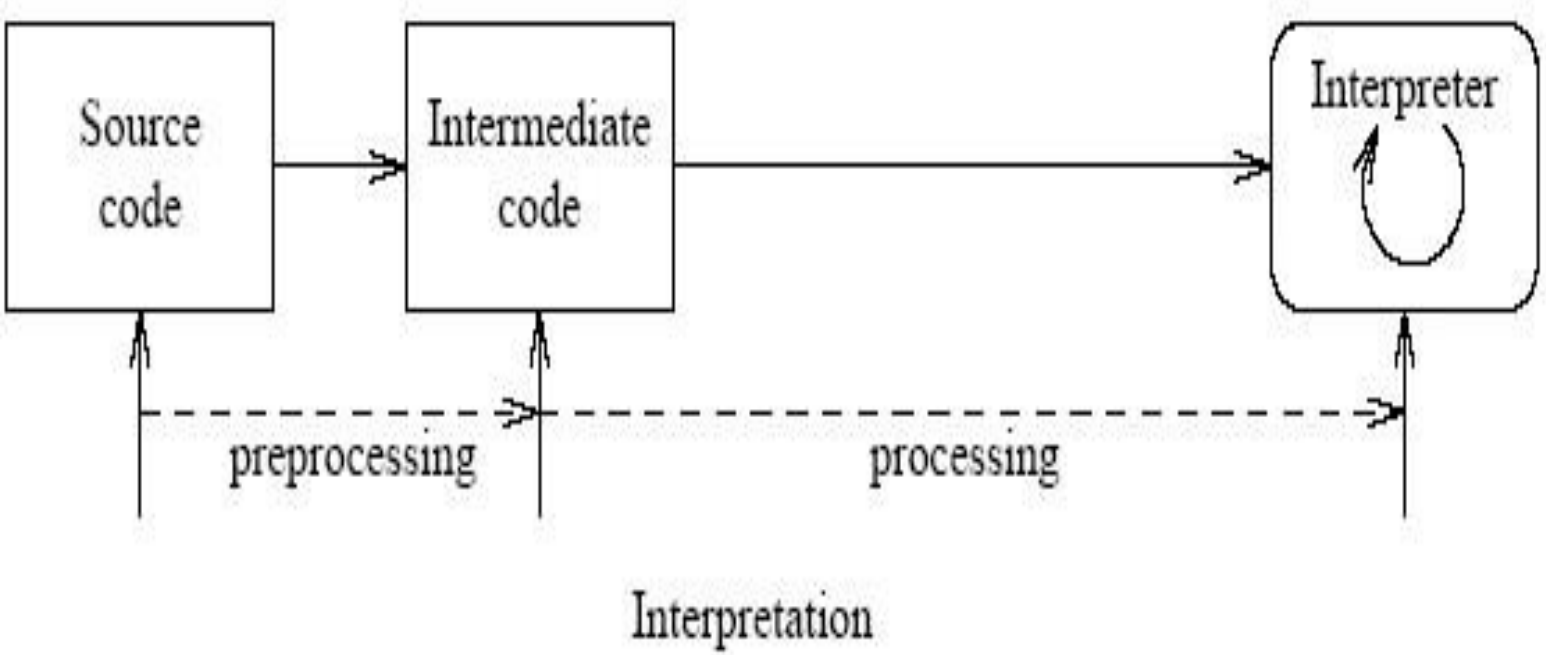
- Man braucht eine einfachere, erweiterte und verbesserte RoboBasic .

- Das Ziel unserer Arbeit: ein neuer Interpreter für RoboBasic.

Interpreter

- Ein Interpreter ist ein Programm, das die Quellecodes einliest, analysiert und interpretiert.
- Kostet relativ wenige Zeit mit der Analyse und Bearbeitung eines Programms.
- Der resultierende Code ist eine Art von Zwischencode.
- Der resultierende Code ist von einem anderen Programm interpretiert.
- Programmablauf ist langsam.

Interpreter



Objektcodes erkennen

- Ein Befehl von RoboBASIC wird von der Programmierumgebung eingelesen und bearbeitet.
- Nach der Bearbeitung erzeugt die Umgebung einen entsprechenden Objektcode.
- Dieser Code wird als eine OBJ-Datei gespeichert, die aus Hexadezimalzahl besteht.
- Die generierte OBJ-Datei wird von einem Hexeditor eingelesen. Mit Hilfe von Hexeditor wird der konkrete Inhalt als Hexadezimalzahl in einem Fenster dargestellt.
- Dort sind entsprechende Hexcodes und Kodierungsregeln für bestimmte RoboBASIC-Befehle herauszufinden.

Objektcodes erkennen

- Nachdem alle entsprechenden Hex-Codes gefunden worden sind, wird eine Datei als XML-Format mit alle von diesem Format erforderlichen Etiketten erstellt.

- In dieser Datei werden die Codes durch Nummerierung sortiert und als Datenbank gespeichert.

Objektcodes Erkennen

Beispiel: BREAK

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20	20	00	00	14
00	62	72	65	61	6B	2E	62	61	0A	03	09	0F	30	12	14
00	CB	C4	11	00											

Objektcodes Erkennen

- Viele zusätzliche Informationen der Quellcodedatei werden durch den Anfangscodeblock beschrieben.

Byte	Inhalt
k(1<=k<=111, hängt die Anzahl der Variablen, k=0 falls keine Variable vorhanden)	Speicherplatzreservierung für benutzerdefinierte Variablen, die später auftauchen können.
K+1 – k+8	Dateiname des Quelltextes
K+9 – k+11	Datum in der Form Jahr-Monat-Tag
K+12 – k+14	Uhrzeit in der Form Stunden-Minuten Sekunden
K+15- k+16	LOW- und HIGH-Anteil der Länge des Quelltextes (in Bytecodes)

Objektcodes Erkennen

- Der Endecodeblock hat ähnliche Funktionalität

Byte	Inhalt
1	C4
2	Anzahl der Codes, der von RoboBasic-Befehle erzeugt hat. Wenn keinen Code erzeugt, dann ist der Bytewert 10. Die Anzahl 1 enspricht Wert 11, usw.
3	00

Objektcodes Erkennen

- Manche zusätzlichen Bytecodes werden von RoboBASIC an einigen Stellen verwendet, um einen Bitwert oder 8 Bits ein Byte zu speichern.

- Viele unterschiedliche Befehle können durch die gleiche Logik mit einem Standardformat bearbeitet werden.

Typ	Flag
Bytewerte	12
Ganzzahlwerte	13
Bytevariablen	15
Ganzzahlvariablen	16

Objektcodes Erkennen

- Danach folgt die Speicheradresse oder Wert.

- Normalerweise wird die Speicheradresse der ersten Variable mit dem Code 40 kodiert, dann der zweiten 41 usw.

Typ	Kodierung
Bytewerte	Wert, 1 Byte
Ganzzahlwerte	LOW- und HIGH-Anteil des Wertes, 2 Bytes
Variablen	Speicheradresse der Variablen, 1 Byte

Objektcodes Erkennen

- Der Gruppencode zeigt, welcher Servomotor durch den Mikrokontroller gesteuert wird.

Gruppencode	Servomotoren	Kodierung
G6A	0-5	00 06
G6B	6-11	06 06
G6C	12-17	0C 06
G6D	18-23	12 06
G6E	24-29	18 06

Algorithmen

- Am Anfang werden die Quelldatei und XML-Datei eingelesen.
- Die Quellcodes von diesen 2 Dateien werden in einer Struktur-Array gespeichert und indexiert.
- Alle gespeicherten Quelltextwörter werden mit der Datenbank verglichen.
- Wenn eine Übereinstimmung in dem Datenbankarray gibt, überprüft das Programm üblichen Wörter in der gleichen Zeile, ob restlichen Ausdrücke mit der Grammatik von dem gerade gefundenen Wort übereinstimmt.
- Wenn nicht, dann wird eine Meldung von Syntaxfehler gezeigt. Dann fängt die weitere Überprüfung in der nächsten Zeile an.

Algorithmen

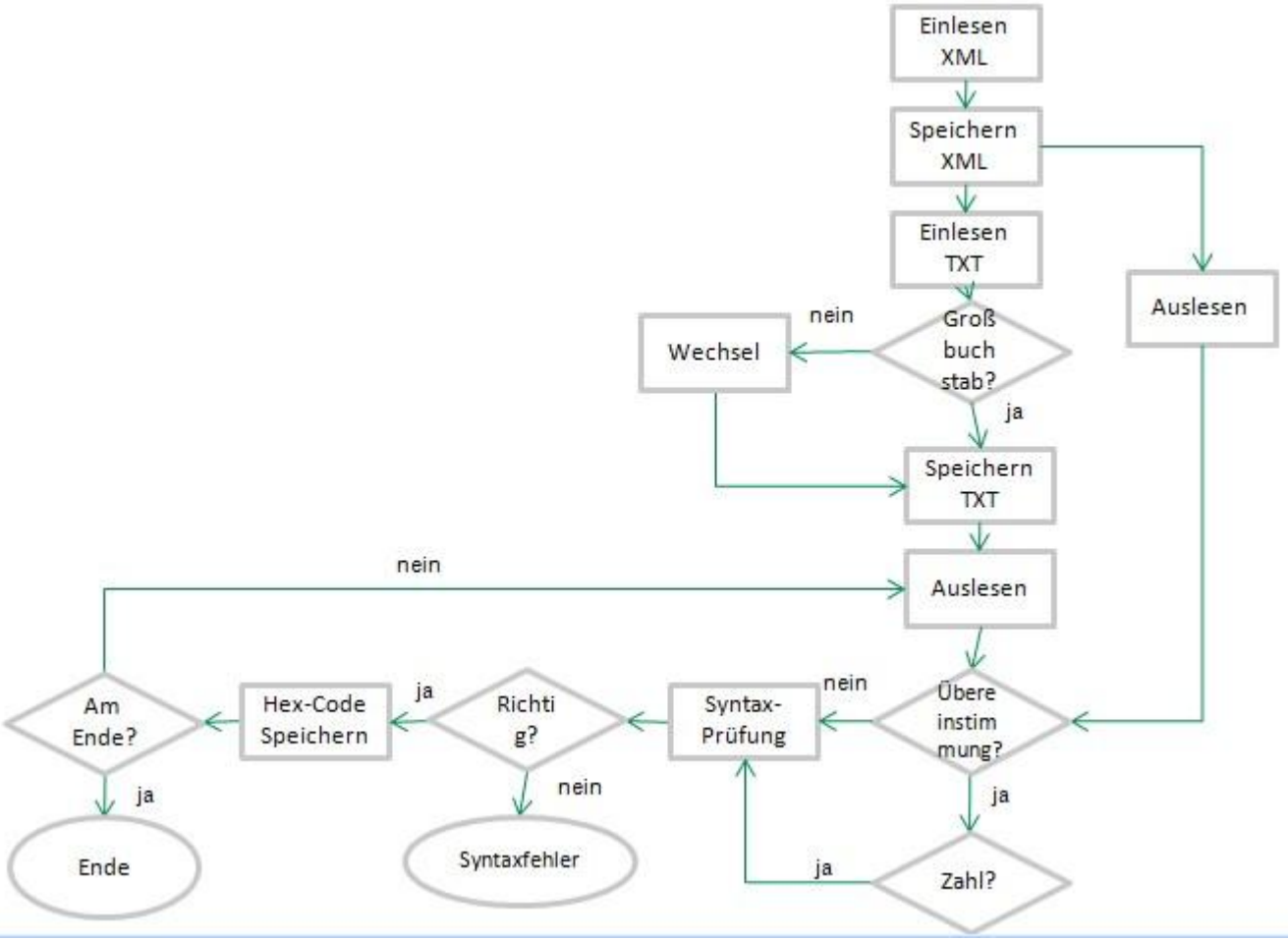
- Wenn es für ein Quelltextwort keine Übereinstimmung in dem Datenbankarray gibt, wird es nach folgenden Fällen unterschieden:

- Das gerade gelesene Wort ist eine Zahl. Dann wird diese Zahl direkt in eine Hexadezimalzahl konvertiert.

- Das gerade gelesene Wort ist falsch. Es handelt sich um einen Buchstabierfehler.

- Wenn keine Fehlermeldung auftaucht, gibt das Programm alle entsprechende Hex-Code in einer Textdatei aus.

Algorithmen



Programm und Beispiel

- Entwicklungsumgebung: Pelles C Version 6.00.4 for Windows 64 bit.

- Beispiel: „ MOVE G6A, 100,100,80,70,50,60“

```
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00 00 00 00  00 00 00 00  00 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 20 20 20
20 20 20 20  20 20 20 20  20 20 20 20  20 00 00 1C
00 6D 6F 76  65 31 2E 62  61 0A 02 18  0F 2B 24 1C
00 B0 00 06  64 64 50 46  32 3C C4 19  00 □  □  □
```

Programm und Beispiel

- Ein anderes Beispiel:

```
PTP SETON
PTP ALLON

'== motor diretion setting ======================
DIR G6A,1,0,0,1,0,0
DIR G6B,1,1,1,1,1,1
DIR G6C,0,0,0,0,0,0
DIR G6D,0,1,1,0,1,0

'== motor start position read ===================
GETMOTORSET G6A,1,1,1,1,1,0
GETMOTORSET G6B,1,1,1,0,0,0
GETMOTORSET G6C,1,1,1,0,0,0
GETMOTORSET G6D,1,1,1,1,1,0
SPEED 5

'== motor power on  =============================
MOTOR G24


        SPEED 10
        MOVE G6A,100, 151,  23, 140, 101, 100
        MOVE G6D,100, 151,  23, 140, 101, 100
        MOVE G6B,100,  30,  80, 100, 100, 100
        MOVE G6C,100,  30,  80, 100, 100, 100
        WAIT

        RETURN
'================================================


        MOVE G6A,100,  76, 145,  93, 100, 100
        MOVE G6D,100,  76, 145,  93, 100, 100
        MOVE G6B,100,  30,  80, 100, 100, 100
        MOVE G6C,100,  30,  80, 100, 100, 100
        WAIT

        RETURN
```

Probleme beim Programm

- Beim erzeugten Code ohne Zusatzinformationen(Dateiname, Datum)
- Es fehlt die Behandlung von Variablen.
- Behandlung von komplizierten Sprachstrukturen.

Zusammenfassung der Arbeit

- Nachteile von RoboBasic.
- Ideen des Entwurfs von dem neuen Interpreter.
- Verbesserte Eigenschaften des Interpreters.
- Potential für Weiterentwicklung

Literaturen

- Herbert Schildt
  C The Complete Reference
  Fourth Edition
  ISBN: 0072121246

- Installationsanleitung der RoboBasic-Software

- Handbuch für Befehlsreferenz RoboBasic

- XML [http://en.wikipedia.org/wiki/XML]

- Compiler und Interpreter [http://web.cs.wpi.edu/~gpollice/cs544-f05/CourseNotes/maps/Class1/Compilervs.Interpreter.html]

Ende

Vielen Dank für Ihre Aufmerksamkeit!