

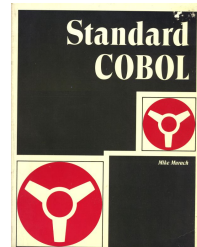
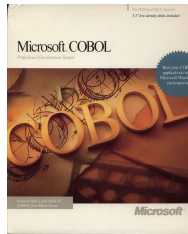
COBOL

Die erste Programmiersprache für Wirtschaftsanwendungen

Sebastian Deußner

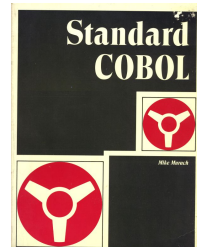
12. Februar 2014

Namensbedeutung



- COBOL steht für COmmon Business-Oriented Language.

Namensbedeutung



- COBOL steht für COmmon Business-Oriented Language.
- entwickelt für betriebswirtschaftliche Programme (im Gegensatz zum technisch-wissenschaftlichen Fokus anderer Sprachen).

- entwickelt durch Arbeitsgruppe in der 2. Hälfte von 1959

- entwickelt durch Arbeitsgruppe in der 2. Hälfte von 1959
- federführend war Grace Hopper, die Erfinderin des ersten Compilers (A-0) und Mitentwicklerin einiger früher Computer (zB Harvard Mark I+II, UNIVAC I)

- entwickelt durch Arbeitsgruppe in der 2. Hälfte von 1959
- federführend war Grace Hopper, die Erfinderin des ersten Compilers (A-0) und Mitentwicklerin einiger früher Computer (zB Harvard Mark I+II, UNIVAC I)
- basierte auf Hoppers FLOW-MATIC, IBMs COMTRAN ("Business-Version" von FORTRAN) und Honeywells FACT

- entwickelt durch Arbeitsgruppe in der 2. Hälfte von 1959
- federführend war Grace Hopper, die Erfinderin des ersten Compilers (A-0) und Mitentwicklerin einiger früher Computer (zB Harvard Mark I+II, UNIVAC I)
- basierte auf Hoppers FLOW-MATIC, IBMs COMTRAN ("Business-Version" von FORTRAN) und Honeywells FACT
- erster Standard (COBOL 60) festgelegt am 03. Januar 1960, danach individuelle (inkompatible) Erweiterungen von verschiedenen Firmen

Versionen des Standards

- ANS COBOL 1968: ANSI Standard um die verschiedenen Änderungen nach 1959 zu einer neuen Basis zusammenzufassen

Versionen des Standards

- ANS COBOL 1968: ANSI Standard um die verschiedenen Änderungen nach 1959 zu einer neuen Basis zusammenzufassen
- COBOL 1974: 2. ANSI Standard, neue Features wie Datei-Organisation, Report Modul; Einführung der Möglichkeit ein Programm in Funktionen zu unterteilen; Abschaffung von Features, deshalb inkompatibel zu vorherigen Standards

Versionen des Standards

- ANS COBOL 1968: ANSI Standard um die verschiedenen Änderungen nach 1959 zu einer neuen Basis zusammenzufassen
- COBOL 1974: 2. ANSI Standard, neue Features wie Datei-Organisation, Report Modul; Einführung der Möglichkeit ein Programm in Funktionen zu unterteilen; Abschaffung von Features, deshalb inkompatibel zu vorherigen Standards
- COBOL 1985: überarbeiteter ANSI Standard, Einführung Gültigkeitsbereich-Terminatoren (z.B. END-IF), neue Features geschachtelte Unterprogramme, neue Statements, die Operatoren \geq und \leq ; ersetzen von selbstmodifizierenden Code (ALTER) durch Referenz Modifikation

Entwicklung um und nach 2000

- viele Programm aus den 80ern und davor verwendeten in Records Datumsfelder mit fester 2-stelliger Jahreszahl

Entwicklung um und nach 2000

- viele Programm aus den 80ern und davor verwendeten in Records Datumsfelder mit fester 2-stelliger Jahreszahl
- => um 1999/2000 Massensterben unter COBOL Anwendungen wegen Y2K-Bug; Programme mussten entweder grundlegend überarbeitet oder ganz ersetzt werden

Entwicklung um und nach 2000

- viele Programm aus den 80ern und davor verwendeten in Records Datumsfelder mit fester 2-stelliger Jahreszahl
- => um 1999/2000 Massensterben unter COBOL Anwendungen wegen Y2K-Bug; Programme mussten entweder grundlegend überarbeitet oder ganz ersetzt werden
- COBOL 2002: neue zeitgemäße Features wie Locale/Unicode-Unterstützung, Objektorientierung zur Einbindung in Java und .NET-Frameworks, Fließkommaunterstützung

Entwicklung um und nach 2000

- viele Programm aus den 80ern und davor verwendeten in Records Datumsfelder mit fester 2-stelliger Jahreszahl
- => um 1999/2000 Massensterben unter COBOL Anwendungen wegen Y2K-Bug; Programme mussten entweder grundlegend überarbeitet oder ganz ersetzt werden
- COBOL 2002: neue zeitgemäße Features wie Locale/Unicode-Unterstützung, Objektorientierung zur Einbindung in Java und .NET-Frameworks, Fließkommaunterstützung
- Ausblick: COBOL 20XX Standard in Entwicklung (erwartet 2014-6), baut primär Objektorientierung aus und passt Datentypen an neue Standards an

Einsatzgebiete

- Haupteinsatzgebiet ist betriebswirtschaftliche Datenverarbeitung

- Haupteinsatzgebiet ist betriebswirtschaftliche Datenverarbeitung
- bei klassischer Aufteilung nach Benutzerschnittstelle, Verarbeitungsteil und Datenhaltungsteil stellt COBOL den Verarbeitungsteil eines EDV-Programms

- COBOL hatte wenig Einfluss auf spätere Programmiersprachen da Fokus auf relativ simple Algorithmen und hohes I/O-Volumen akademisch uninteressant (Ausnahme: SAPs Programmiersprache ABAP)

- COBOL hatte wenig Einfluss auf spätere Programmiersprachen da Fokus auf relativ simple Algorithmen und hohes I/O-Volumen akademisch uninteressant (Ausnahme: SAPs Programmiersprache ABAP)
- geschätzte 70-80% aller Geschäftstransaktion involvieren COBOL Programme, oft als Teil von jahrzehntelang gewachsenen Systemen

heutiger Einsatz

- COBOL hatte wenig Einfluss auf spätere Programmiersprachen da Fokus auf relativ simple Algorithmen und hohes I/O-Volumen akademisch uninteressant (Ausnahme: SAPs Programmiersprache ABAP)
- geschätzte 70-80% aller Geschäftstransaktion involvieren COBOL Programme, oft als Teil von jahrzehntelang gewachsenen Systemen
- komplette Neuentwicklungen nur noch selten in COBOL

- COBOL hatte wenig Einfluss auf spätere Programmiersprachen da Fokus auf relativ simple Algorithmen und hohes I/O-Volumen akademisch uninteressant (Ausnahme: SAPs Programmiersprache ABAP)
- geschätzte 70-80% aller Geschäftstransaktion involvieren COBOL Programme, oft als Teil von jahrzehntelang gewachsenen Systemen
- komplette Neuentwicklungen nur noch selten in COBOL
- 2009 geschätzt über 40 Milliarden Zeilen COBOL Code in Industrieprogrammen verwendet, Wachstumsrate 4 Milliarden Zeilen Code pro Jahr

Vorteile

- entworfen um Programme auf verschiedener Hardware laufen zu lassen ohne große Veränderung des Codes

Vorteile

- entworfen um Programme auf verschiedener Hardware laufen zu lassen ohne große Veränderung des Codes
- durch mehrstufiges Design der einzelnen Programmsprachen-Module lassen sich COBOL-Programme auf sehr eingeschränkter Hardware mit geringen Anpassungen ausführen

Vorteile

- entworfen um Programme auf verschiedener Hardware laufen zu lassen ohne große Veränderung des Codes
- durch mehrstufiges Design der einzelnen Programmiersprachen-Module lassen sich COBOL-Programme auf sehr eingeschränkter Hardware mit geringen Anpassungen ausführen
- der ANSI-Standard wird durchschnittlich alle 10-15 Jahre an neue Gegebenheiten angepasst, z.B. kam 2002 Unterstützung für Objektorientierung dazu

Vorteile

- entworfen um Programme auf verschiedener Hardware laufen zu lassen ohne große Veränderung des Codes
- durch mehrstufiges Design der einzelnen Programmsprachen-Module lassen sich COBOL-Programme auf sehr eingeschränkter Hardware mit geringen Anpassungen ausführen
- der ANSI-Standard wird durchschnittlich alle 10-15 Jahre an neue Gegebenheiten angepasst, z.B. kam 2002 Unterstützung für Objektorientierung dazu
- es gibt einen kostenlosen quelloffenen Compiler namens GNU COBOL (ehemals OpenCOBOL) für POSIX-kompatible Betriebssysteme

Nachteile

- bis COBOL 74 gab es keine Möglichkeit Programme zu strukturieren, zB waren alle Variablen global

Nachteile

- bis COBOL 74 gab es keine Möglichkeit Programme zu strukturieren, zB waren alle Variablen global
- aufgrund vieler Eigenentwicklungen von Firmen und anderen Gruppen gibt es viele verschiedene Standards von COBOL die teilweise inkompatibel zueinander sind

Nachteile

- bis COBOL 74 gab es keine Möglichkeit Programme zu strukturieren, zB waren alle Variablen global
- aufgrund vieler Eigenentwicklungen von Firmen und anderen Gruppen gibt es viele verschiedene Standards von COBOL die teilweise inkompatibel zueinander sind
- da die Syntax sich an geschriebenem Englisch orientiert werden COBOL Programme schnell recht langwierig zu schreiben

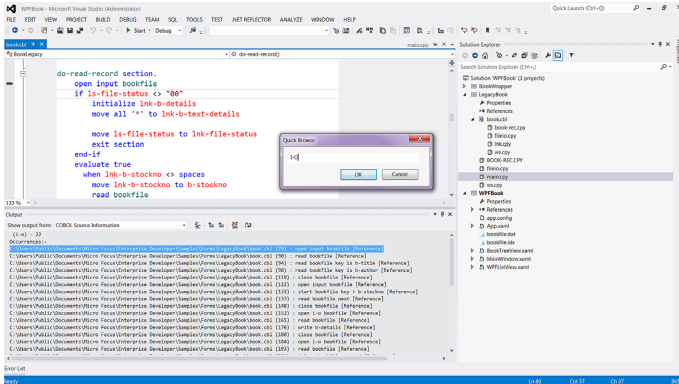
- viele Computer/Betriebssystemhersteller (zB IBM, Siemens, Unisys, HP) entwickelten (und vertreiben teilweise immer noch) eigene Compiler, oft mit eigenen Erweiterungen des Standards

- viele Computer/Betriebssystemhersteller (zB IBM, Siemens, Unisys, HP) entwickelten (und vertreiben teilweise immer noch) eigene Compiler, oft mit eigenen Erweiterungen des Standards
- es gibt verschiedene Codegeneratoren die COBOL-Programme oder Teile davon generieren um die Entwicklungsarbeit zu erleichtern

- viele Computer/Betriebssystemhersteller (z.B. IBM, Siemens, Unisys, HP) entwickelten (und vertreiben teilweise immer noch) eigene Compiler, oft mit eigenen Erweiterungen des Standards
- es gibt verschiedene Codegeneratoren die COBOL-Programme oder Teile davon generieren um die Entwicklungsarbeit zu erleichtern
- Plugins für viele gängige IDEs, z.B. cobolclipse für Eclipse

Entwicklungsumgebung (2)

- verbreitetste (kommerzielle) COBOL IDE ist Visual COBOL von Micro Focus; besitzt moderne Features wie Generierung von Java Byte-Code, .NET Unterstützung und Webservices



- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.
- Kontrollstrukturen:

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.
- Kontrollstrukturen:
 - IF...ELSE: funktioniert wie erwartet

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.
- Kontrollstrukturen:
 - IF...ELSE: funktioniert wie erwartet
 - EVALUATE: vergleichbar mit CASE/Switch (eingeführt mit COBOL 85)

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.
- Kontrollstrukturen:
 - IF...ELSE: funktioniert wie erwartet
 - EVALUATE: vergleichbar mit CASE/Switch (eingeführt mit COBOL 85)
 - PERFORM: vergleichbar mit for-Schleifen

- COBOL wurde so entworfen das der Code relativ lesbarem Englisch entspricht z.B. "ADD b TO c GIVING a" ($a = b + c$)
- importieren von Programmteilen als sgn. Copybooks mit COPY Befehl, vergleichbar mit `#include` in C u.Ä.
- Kontrollstrukturen:
 - IF...ELSE: funktioniert wie erwartet
 - EVALUATE: vergleichbar mit CASE/Switch (eingeführt mit COBOL 85)
 - PERFORM: vergleichbar mit for-Schleifen
 - GO TO: Standard GOTO, optional mit Bedingung

- Grunddatenstruktur ist ein sogenannter Record, ein multidimensionales heterogenes Array

- Grunddatenstruktur ist ein sogenannter Record, ein multidimensionales heterogenes Array
- Records können mit homogenen Arrays gemischt verwendet werden, ein Array kann also Inhalt eines Records sein und umgekehrt

- Grunddatenstruktur ist ein sogenannter Record, ein multidimensionales heterogenes Array
- Records können mit homogenen Arrays gemischt verwendet werden, ein Array kann also Inhalt eines Records sein und umgekehrt
- über die PICTURE Klausel kann man den Inhalt (inklusive Feldgrößen) von Records exakt definieren

- Grunddatenstruktur ist ein sogenannter Record, ein multidimensionales heterogenes Array
- Records können mit homogenen Arrays gemischt verwendet werden, ein Array kann also Inhalt eines Records sein und umgekehrt
- über die PICTURE Klausel kann man den Inhalt (inklusive Feldgrößen) von Records exakt definieren
- Variablen müssen bei der Deklaration typisiert werden, allerdings hat man Dank Records und der PICTURE Klausel vergleichsweise viel Freiheit dabei

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code
 - Algorithmen Beschreibungen und Daten Beschreibungen

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code
 - Algorithmen Beschreibungen und Daten Beschreibungen
- IDENTIFICATION: beginnt jedes Programm, benennt Programm und den Autor (Autor optional), sonstige Kommentare als Dokumentation

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code
 - Algorithmen Beschreibungen und Daten Beschreibungen
- IDENTIFICATION: beginnt jedes Programm, benennt Programm und den Autor (Autor optional), sonstige Kommentare als Dokumentation
- ENVIRONMENT: beinhaltet maschinenabhängige Programmspezifikationen wie zB Verbindungen zwischen dem Programm und externen Daten

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code
 - Algorithmen Beschreibungen und Daten Beschreibungen
- IDENTIFICATION: beginnt jedes Programm, benennt Programm und den Autor (Autor optional), sonstige Kommentare als Dokumentation
- ENVIRONMENT: beinhaltet maschinenabhängige Programmspezifikationen wie zB Verbindungen zwischen dem Programm und externen Daten
- PROCEDURE: beinhaltet die Algorithmen

Aufteilung eines Programms

- jedes Programm besteht aus 4 Teilen (DIVISION) zur Trennung von:
 - hardwareabhängigem und unabhängigem Code
 - Algorithmen Beschreibungen und Daten Beschreibungen
- IDENTIFICATION: beginnt jedes Programm, benennt Programm und den Autor (Autor optional), sonstige Kommentare als Dokumentation
- ENVIRONMENT: beinhaltet maschinenabhängige Programmspezifikationen wie zB Verbindungen zwischen dem Programm und externen Daten
- PROCEDURE: beinhaltet die Algorithmen
- DATA: beinhaltet die Daten Beschreibungen

Code-Beispiele

Hello World

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
PROCEDURE DIVISION.  
    DISPLAY "Hello, world."  
    STOP RUN.
```

Code-Beispiele

Hello World

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
PROCEDURE DIVISION.  
    DISPLAY "Hello, world."  
    STOP RUN.
```

Schleifen

```
PERFORM VARYING i FROM 0 BY 1  
    UNTIL i >= 10  
    ...  
END-PERFORM
```

Code-Beispiele

Hello World

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
PROCEDURE DIVISION.  
    DISPLAY "Hello, world."  
    STOP RUN.
```

Schleifen

```
PERFORM VARYING i FROM 0 BY 1  
    UNTIL i >= 10  
    ...  
END-PERFORM
```

EVALUATE

```
20 EVALUATE True  
    WHEN Nenner > 0  
        COMPUTE Zahl = Zaehler / Nenner  
22 WHEN Nenner < 0  
        COMPUTE Zahl = Zaehler / Nenner * -1  
24 WHEN Other  
        DISPLAY "Fehler"  
26 MOVE 0 TO Zaehler  
End-Evaluate
```

- ① <http://en.wikipedia.org/wiki/COBOL>
- ② <http://de.wikipedia.org/wiki/COBOL>
- ③ Terrence W. Pratt - Programming Languages: Design and Implementation (Prentice Hill 1975)
- ④ OpenCOBOL Programmers Guide