Introduction into COBOL

Cobol is probably one of the most unused programming Languages at this time. It is mainly used in companies, that want to change their old databases to newer ones because, Cobol is compatible to nothing :). A list of Compilers for the Mac and PC is in the appendix.

In Cobol there are 4 Divisions, where the user has to input some information about the Computer he uses, where variables get declared, etc.

1.) The Identification Division:
an example for an identification division is (the blue words are non-changable words):

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TEST.
```

The first line just names the Division. In the second line stands the program's name. In that case the name is "test". Note: In all divisions (and just in divisions), there's a "." after every command.

2.) The Environment Division:
an example for an environment division is (the blue words are non-changable words):

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
TERMINAL IS  T.
DECIMAL-POINT IS COMMA.
```

the Environment Division does subdivide into sections. For this time we just need the Configuration Section. Here we configure how our terminal is named, and what char you want to

use as a decimal point. You need the terminal name for in- and output (like "accept variable from T").

3.) The Data Division:
an example for a data division is (the blue words are non-changable words):

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 TEXT PIC X(7).
01 NUMBER1 PIC 99.
01 NUMBER2 PIC 9(15).
```

Here we define our variables. We don't write "int x" or "dim x as integer" in Cobol, like we do it in C and other programming languages. Here every variable has a number (in our example the number is 01). That number can be any number from 01 to 65. It‘s better to define every variable with 01, so we wont lose overview. After this number, there comes the variable-name. Next to the name, stands the word "pic". This doesn‘t mean picture, and for basic programming we must not know, what it means. Then we write even a "X" or a "9". A "X" means, that the variable is alphanumeric, a "9" means that it is numeric. Then comes the number of places we wan‘t to use for the variable. "XXXXXX“ means the same like "X(6)". Here we have three variables with the number 01. The first one is named TEXT and has seven alphanumeric places. The second one is named NUMBER1 and has two numeric places. The third one is named NUMBER2 and has 15 numeric places.
Note: Alphanumeric means, that the user can input letters and numbers. Numeric means, only to input numbers.

4.) The Procedure Division:
an example for a procedure division is (the blue words are non-changable words):

```
PROCEDURE DIVISION.
MAIN.
```

The word "main" is the name of our first sub-program.
Note: Often we don't need Sub-programs in Cobol. So the Sub-program "main" contains the whole source.

5.) Basic Commands:

5.1.) In- and Output.

For inputting a variable use: `Accept <variable name> from <Terminal name>`
For outputting a variable use: `Display <variable name> upon <Terminal name>`
For outputting a text use: `Display <"Text here"> upon <Terminal name>`
Note: We can't input variables with a default prompt.

5.2.) Mathematics Operations:

For adding two variables use: Add <variable1> to <variable2> giving <variable>.
For subtracting two variables use: `Subtract <variable1> from <variable2> giving <variable>`
For multiplying two variables use: `Multiply <variable1>  by <variable2> giving <variable>`
For dividing two variables use: `Divide <variable1> into <variable2> giving <variable>`
If you have to compute more than two variables use: `Compute <variable> = 15668*2+1`
Note: We can sure, also compute some numbers like: `Divide 17 into 13 giving x`. The "giving" option mustn‘t be used. If we just write `Add x to y` the result will be stored in the variable x.

5.3) Loops:

In Cobol there‘s just one kind of loop, we need for starting. It is:
```
Perform until <Condition>.
```

An example for a loop is:
```
Perform until x = 10
 Display x upon t
 x = x + 1
end-perform
```

Note: In Cobol we must tell the Compiler, when the loop is over. In a perform loop we are doing this with an "end-perfom".

5.4.) Query‘s:

There is only one query we need at this time. The correct syntax is:
```
If <Condition> then
 blah
 blah
else
 blah
 blah
end-if
```

Note: A "if" must also end with an "end-if".

6.) Now we know the Basic Cobol-command‘s. Let‘s write our first program: a calculator (what else ;):

```
       IDENTIFICATION DIVISION.
        PROGRAM-ID.  CALC.
        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        SPECIAL-NAMES.
            TERMINAL IS T.
```

```
        DATA DIVISION.
        WORKING-STORAGE SECTION.
        01 NUMBER1 PIC 9(4).
        01 NUMBER2 PIC 9(4).
        01 OPERATOR PIC X.
        01 END? PIC X.
        01 RESULT PIC 9(9).
        PROCEDURE DIVISION.
        MAIN.
            PERFORM UNTIL END? NOT= "Y" OR "y"
            DISPLAY "PLEASE INPUT THE FIRST NUMBER:" UPON T
            ACCEPT NUMBER1 ONE FROM T
            DISPLAY "PLEASE INPUT THE OPERATOR (*, -, +, /):"
            ACCEPT OPERATOR FROM T
            DISPLAY "PLEASE INPUT THE SECOND NUMBER" UPON T
            IF OPERATOR = "+" THEN
              ADD ZAHL1 TO ZAHL2 GIVING RESULT
            END IF
            IF OPERATOR = "-" THEN
             SUBTRACT NUMBER2 FROM NUMBER1 GIVING RESULT
            END IF
            IF OPERATOR = "*" THEN
             MULTIPLY NUMBER1 BY NUMBER2 GIVING RESULT
            END IF
            IF OPERATOR = "/" THEN
             DIVIDE NUMBER2 INTO NUMBER1 GIVING RESULT.
            END IF
            DISPLAY "THE RESULT OF" NUMBER1 " " OPERATOR " " NUMBER2 " = "
RESULT
            DISPLAY ""
            DISPLAY "AGAIN? (Y/N)"
            INPUT END?
           END-PERFORM
           STOP RUN.
```

7.) The Appendix:

Now we know some basic Cobol-commands. I don‘t know if I will write more about Cobol, cause my opinion is that you won't need it very often :). Anyway, if you wan't to tell me something (like doing a second section :)., mail to: bern@pantucek.vienna.at. btw: I even know some Contact addresses, where Cobol-compilers are viable.

7.1) Where can I get a COBOL compiler ?

7.1.1 for DOS, Windows or OS/2 ?

Acu, CA, Liant, MBP and Micro Focus all produce compilers for one or more of these environments. Microsoft used to re-badge the Micro Focus compiler, but not any more.

7.1.2 for Windows/NT ?

Micro Focus has a COBOL 32-bit SDK available for Windows/NT.

7.1.3 for UNIX ?

Acu, Liant and Micro Focus have products available across a large number of UNIX platforms. Some OEMs re-badge and/or reengineer these products for their own systems, too.

7.1.4 for the Macintosh ?

Micro Focus and Liant produce a COBOL development system for the Mac running A/UX.

Micro Focus have also announced that they will be releasing a product on MacOS. There are currently no dates for release.

7.1.5 for other environments ?

Most major vendors have their own COBOL implementation, or have someone else's ported to their platform(s). There are quite a few available for CP/M and MP/M, and one is even rumoured to have been available for the PERQ workstation.

7.2) Is there a free COBOL compiler ?

Just for the record, no COBOL tools are listed in the Catalog of compilers, interpreters, and other language tools posted to comp.compilers and comp.lang.misc.

However, two books in the book list come with a COBOL compiler. See section 10 for details.

7.2.1 for DOS, Windows or OS/2.

There is a freely available COBOL compiler for DOS. It can be found on many archive sites, named cobol650.zip. The sources are not available, however. Have a read through Section 8 before you start.

Also, it may be possible to run the freely available CP/M
compiler (see 4.5) under a freely available CP/M emulator.

7.2.2 for Windows/NT ?

The CP/M compiler/emulator combination should work here, too.

7.2.3 for UNIX ?

There are no well-documented examples of a freely available COBOL
compiler for UNIX.

COBOL 6.50 might run under a UNIX emulation of a DOS system,
however. (For example, Sun's WABI for the Sparc and Intel
platforms, or dosemu under Linux.)

The CP/M compiler (see 4.5) should run under a CP/M emulator
for UNIX in a similar fashion.

7.2.4 for the Macintosh ?

Not that I know of.

7.2.5 for other environments ?

There is a freely available CP/M COBOL compiler/interpreter
(NPS Micro COBOL). This is available via anonymous FTP
from oak.oakland.edu in /pub/cpm/cobol.

Article by: 4E71
Edited by: ZiffDuck