COBOL = "**CO**mmon **B**usiness-**O**riented **L**anguage"

Grace Hopper

(1906-1992)
New York City
Yale math (1934)
Vassar College
Navy Reserves (1943)
Helped to create UNIVAC (Universal Automatic Computer) (1952)
Invented first linker (1952)
Coined the word “bug”
Mother of COBOL

In 2009 it was estimated 220 billion lines of COBOL in existence, which was ~80% of the world's actively used code. (~5 Bln added annually). Today (2019) COBOL still runs in traditional banking, lots of large scale government systems, insurance and health care.

COBOL was designed in 1959 (60 years ago).
It is primarily used on **mainframe computers** in business, finance, and governments.
It handles large-scale batch and transaction processing jobs.
It run on various operating systems (z/OS, z/VSE, VME, Unix, OpenVMS and Windows).

Academic computer scientists were not interested in COBOL.
COBOL has been criticized for its verbosity, design process, and poor support for structured programming.

COBOL was designed in 1959 by **CODASYL** (Conference/Committee on Data Systems Languages).
It was partly based on work by **Grace Hopper**, commonly referred to as "**the (grand)mother of COBOL**".
It was created as part of a **US Department of Defense** effort to create a portable programming language.
It was standardized in 1968 and has since been revised four times (last time in 2014).

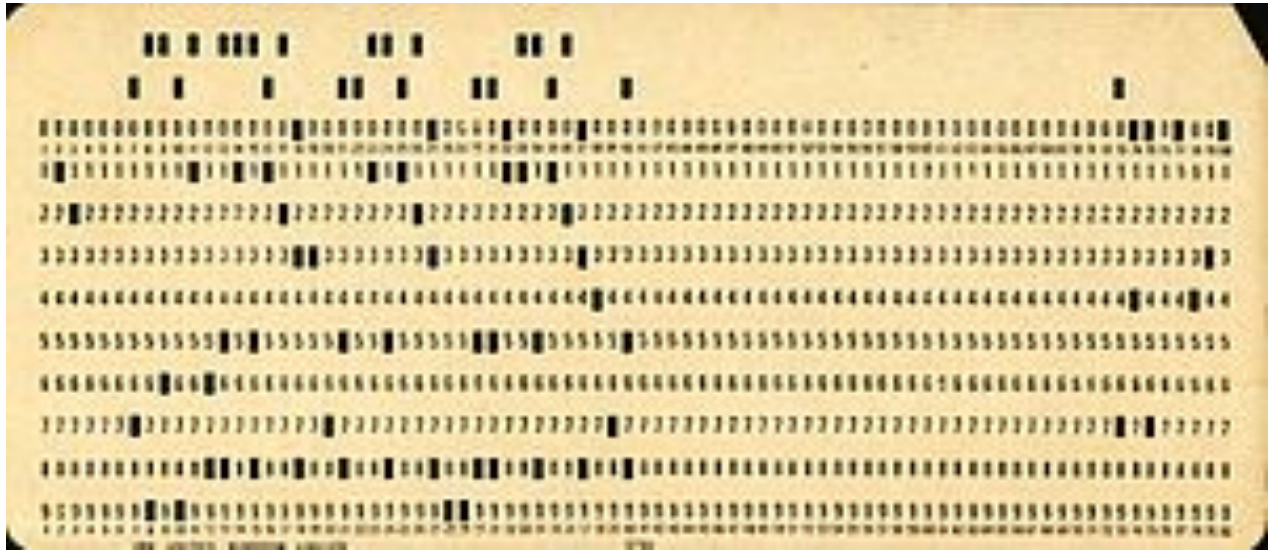
COBOL is compiled, imperative, procedural and, since 2002, object-oriented.
COBOL statements have an English-like syntax, which was designed to be readable and self-documenting.
However, it is too verbose, uses over 300 reserved words. COBOL code is split into four divisions (identification, environment, data and procedure) containing a rigid hierarchy of sections, paragraphs and sentences.
COBOL uses only global variables. Variable names may contain dashes in them.
COBOL code is written using 80-character lines (this comes historically from punched cards), and different columns in a line serve different purpose (for example, an asterisk in col.7 makes it a comment).
The COBOL standard specifies 43 statements, 87 functions, and just one class.

IBM = International Business Machines.
BMs (Business Machines) are not computers.
They are MECHANICAL MACHINES.
They store data on punched cards.
They can:
- mechanically take one card at a time
- read information from the card,
- sort/filter through many cards to find information
- do simple calculations, for example, calculate totals.

Idea: create standard way to encode numbers and characters on punched cards.

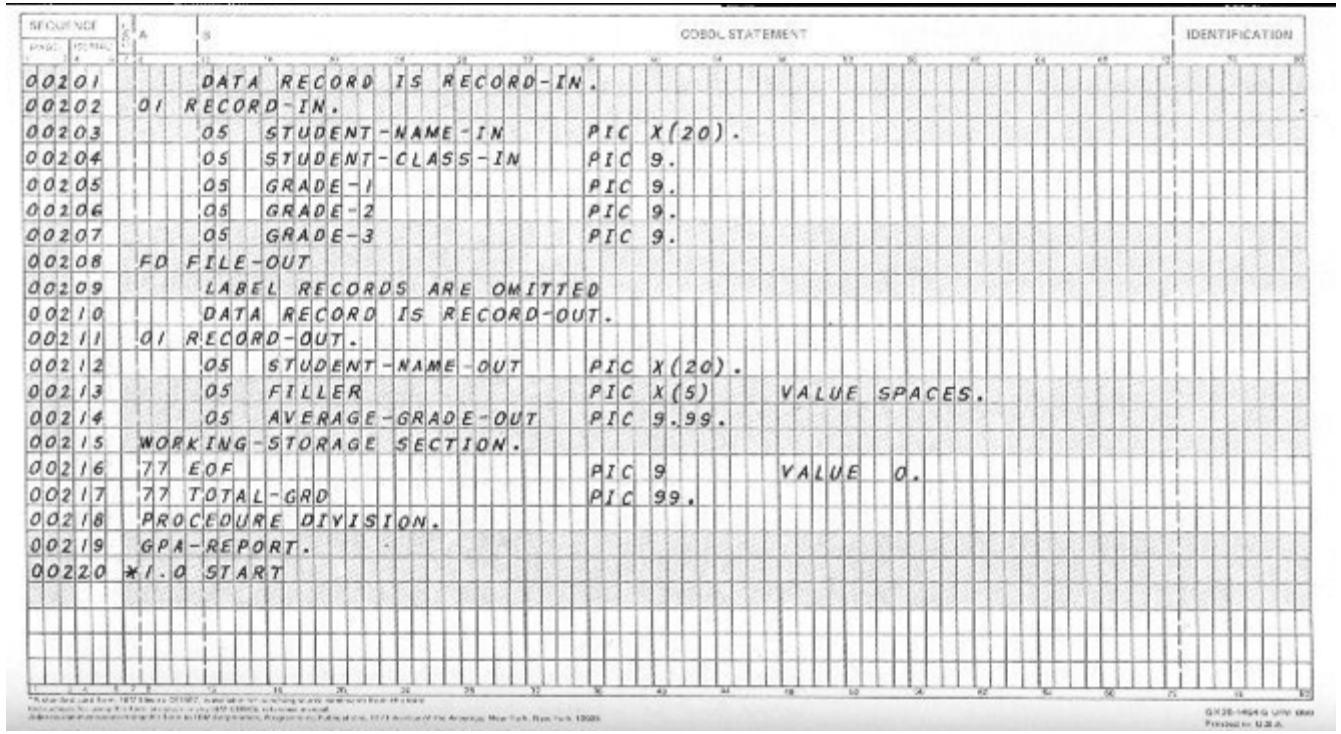
Idea: the logic of calculations can also be also stored on punch-cards.

Idea: create standard way to express compute-instructions (programming language).

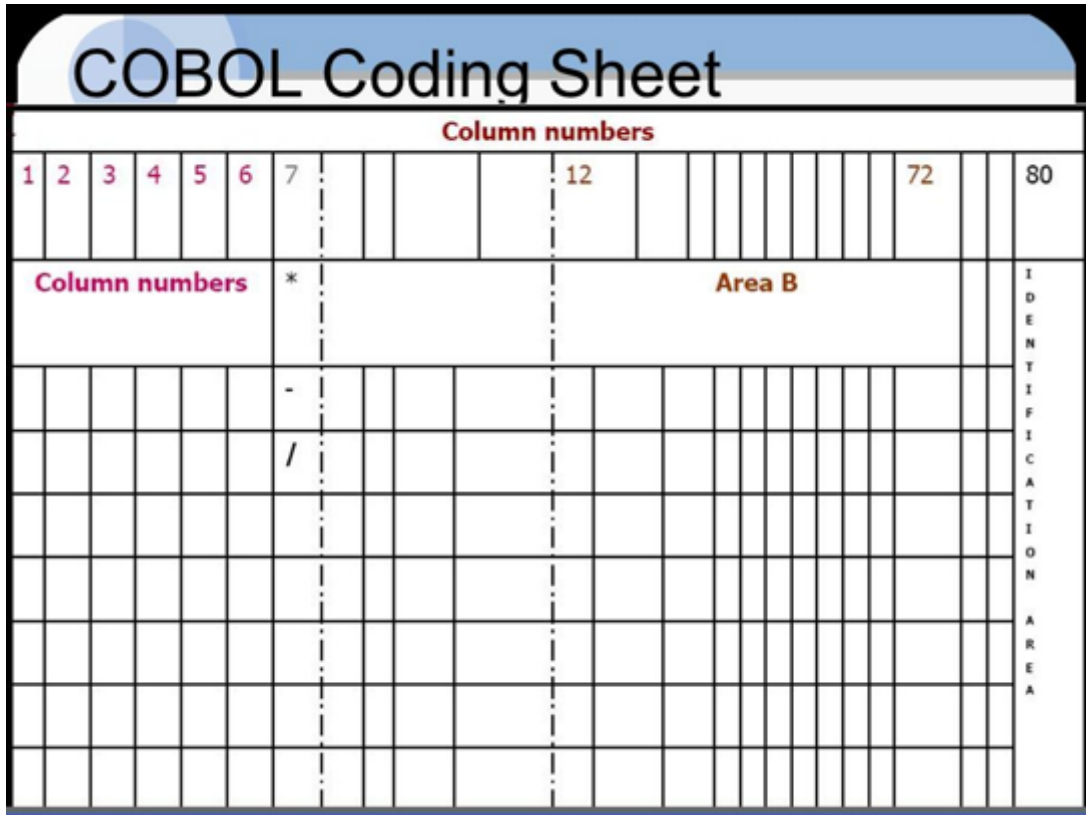


Punched Card

COBOL was designed by a committee : **CODASYL** (Conference/Committee on Data Systems Languages).
It was designed in time of punched cards.
The code was written on a “COBOL Coding Sheet”, which has 80 columns – like a standard punched card.
Each line of code can be transferred to a single punched card.
Note - different columns/areas on the card are used for different purposes
(for example, an asterisk in col.7 makes it a comment).



```
00201        DATA RECORD IS RECORD-IN.
00202    01  RECORD-IN.
00203        05  STUDENT-NAME-IN        PIC X(20).
00204        05  STUDENT-CLASS-IN       PIC 9.
00205        05  GRADE-1                PIC 9.
00206        05  GRADE-2                PIC 9.
00207        05  GRADE-3                PIC 9.
00208    FD  FILE-OUT
00209        LABEL RECORDS ARE OMITTED
00210        DATA RECORD IS RECORD-OUT.
00211    01  RECORD-OUT.
00212        05  STUDENT-NAME-OUT       PIC X(20).
00213        05  FILLER                 PIC X(5)      VALUE SPACES.
00214        05  AVERAGE-GRADE-OUT      PIC 9.99.
00215    WORKING-STORAGE SECTION.
00216    77  EOF                        PIC 9         VALUE 0.
00217    77  TOTAL-GRD                  PIC 99.
00218    PROCEDURE DIVISION.
00219    GPA-REPORT.
00220    *1.0 START
```



COBOL can be installed on regular laptop – Windows, MacOS, Linux.

Below are instructions for MacOS

http://macappstore.org/gnu-cobol/

brew install gnu-cobol

https://www.consulting-bolte.de/index.php/tech-blog/cobol/184-install-cobol-on-macos-x

https://www.tutorialspoint.com/compile_cobol_online.php

OpenCobolIDE

brew install pyqt5

pip3 install OpenCobolIDE --upgrade

run it by:
    openCobolIde

COBOL – “Hello World!” (example on Unix)

---- file “hello.cob” ----------

```
HELLO * HISTORIC EXAMPLE OF HELLO WORLD IN COBOL
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
           DISPLAY "HELLO, WORLD!".
           STOP RUN.
```

The compiler is **cobc**, which is executed as following:

```
$ cobc -o hello hello.cob
$ ./hello
Hello World!
```

```
Columns post-year-2002:

Column   Area
------------------------
1-6      Sequence Number
7        Indicator
8-255    Program text
```

JCL Script Example

```
//MYJOB   JOB  CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),REGION=256K
//S1      EXEC PGM=IEBGENER
//SYSUT1   DD  DATA,DLM=$$
/*************************************************************/
/*                                                           */
/*                      Hello World                          */
/*                                                           */
/*************************************************************/
$$
//SYSUT2   DD  SYSOUT=A
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  DUMMY
//
```

JCL = Job Control Language

Most JCL Statements start with // in column 1 and 2.
(There is also a delimiter statement /* or custom defined)

Everything to the right of column 72 is ignored

JCL MUST BE CODED IN UPPERCASE

Every job begins with a JOB statement

Every Job has at least 1 Step (= EXEC statement)

Every Step (usually) has at least one dataset (= DD statement)

Mainframe ("Big Iron", "Big Blue") – "big" throughput and reliability.

Mainframe – computers (hardware + software) were historically introduced as largest servers for commercial transaction processing.
They are famous for being able to do massive transactional processing of data, while maintaining 100% availability/reliability (built-in internal redundancy, hot-swappable modules for upgrades and repair).

Most common (dominant) today are IBM mainframe computers running 64-bit z/OS (introduced in 2000).

In 2017 IBM claimed that 92 of the world's 100 biggest banks used its mainframes, and that they handle 87% of global credit card transactions,

But IBM did NOT invent mainframe

1950 – UNIVAC (UNIVersal Automatic Computer) - first mainframe.
The company "Unisys" still exists and sells mainframes.

During this time IBM was selling IBM 700/7000 series of computers for electronic data processing – they were not mainframe yet.

1964 - The first IBM mainframe (IBM System/360).
Sabre – first Online Reservation – was using IBM TPF (Transaction Processing Facility)

Today there are still exist some non-IBM mainframes:
"Fujitsu-Siemens", NEC, Unisys, Bull.

Modern IBM z15 with 4 frames (19" cabinets).
Together up to:
- 190 configurable cores,
- 129 zIIPs (z-Integrated-Information-Processors)
- 40 TB memory
- 99.999% availability
- can run Linux and Z/OS.
- can host up to 2.4 million containers!

Note: today's cloud Virtual Servers are comparable, for Example, AWS Unix VMs: up to 128 vcpus, 24 TB RAM

IBM latest z15 microprocessors are comparable in performance with latest Intel Xeon or AMD processors. Although architecture is different.
12 cores, 14 nm, 5.2 GHz clock rate, 64bit, 256 MB L3 cache (21MB/core – compare with 2.5MB/core for Intel)

Processors in a Mainframe:
CP = Central Processor
SAP = System Assistance Processor - for I/O, error recovery, etc.
IFL = Integrated Facility for Linux
zAAP - to run Java
zIIP = z9® Integrated Information Processor - to handle database workloads.
ICF = Integrated Coupling Facility (internal scratch=pad).
Spare = can be used to replace failed PU (Processing units).

IBM Mainframe OS and Systems

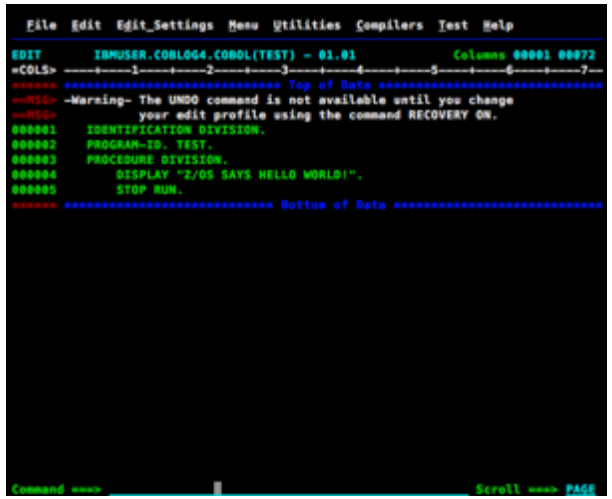
1964 – IBM System/360
1970 – IBM System/370 – Virtual Memory, Virtual Storage
1974 – IBM MVS = Multiple Virtual Storage – common OS under System/370
1995 – IBM System/390 – to simplify packaging and ordering
2000 – IBM z/OS – 64-bit OS (written in PL/X, HLASM, and C/C++)

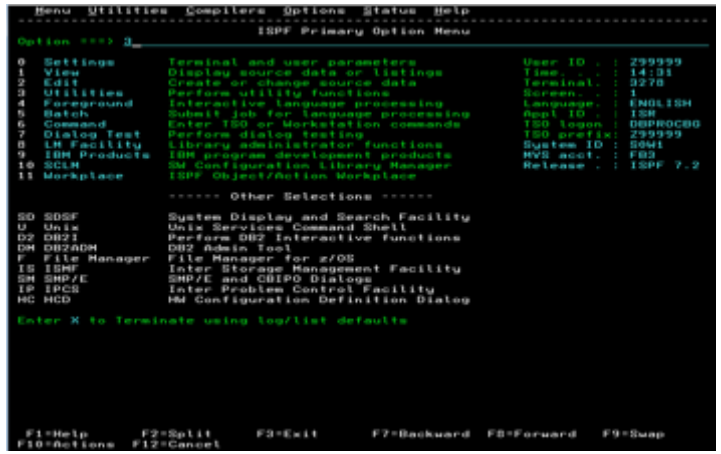
Common Systems:

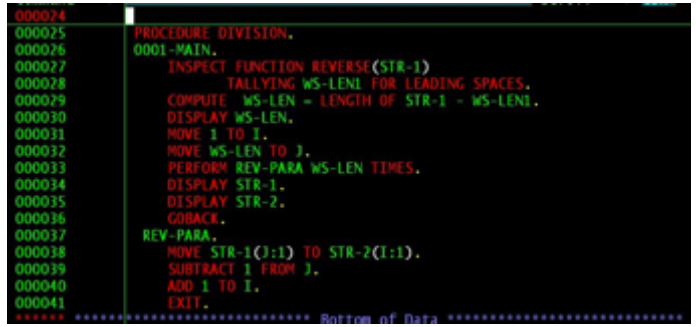
CICS - Customer Information Control System - high-volume online transaction processing
COBOL – Common Business-Oriented Language
PL/I = PL/1 = Programming Language One (since 1960), PL/X = vaguely PL/I-like language
HLASM = High Level Assembler
IMS - Information Management System - a hierarchical transactional database.
DB2 – Relational Database
RACF - Resource Access Control Facility (a security system)
SNA – Systems Network Architecture
IBM MQ – Messaging
VSAM - record-oriented data access methods (indexed, fast)
REXX - interpreted programming language, easy to read and use
CLIST - Command List - a procedural programming language (superseded by Rexx)
SMP/E - System Modification Program/Extended - a tool to manage software installations on z/OS system
JCL - Job Control Language – scripting languages to write script to run a batch job or start a subsystem
TSO/E - "Time Sharing Option/Extensions" – interactive system for sys-admins
ISPF - Interactive System Productivity Facility – terminal interface software (screen editor, etc.)
Unix-style hierarchical HFS[NB 2] and zFS file systems

1988 - AS/400 (AS = Application System - also called the "IBM iSeries") - a midrange server from IBM, very common.

IBM Terminal Interface







VSAM = Virtual Storage Access Method

VSAM was introduced by IBM in 1970's.
It is a collection of file storage/access methods used in OS/390,
MVS (Multiple Virtual Storage), and now in z/OS operating systems.

VSAM offers faster access than flat files because it uses an inverted index (B+tree)
based on embedded prime key field, that you define.

VSAM records can be of fixed or variable length.
IBM now promotes DB2, a relational database management system,
although VSAM linear datasets are still used to contain tablespaces
and index spaces within the system.

VSAM supports four data set organizations:
- Key Sequenced Data Set (KSDS)
- Relative Record Data Set (RRDS)
- Entry Sequenced Data Set (ESDS)
- Linear Data Set (LDS).

The KSDS, RRDS and ESDS organizations contain records,
while the LDS organization (added later to VSAM) simply contains a sequence of pages
with no intrinsic record structure, for use as a memory-mapped file.

Big-endian vs Little-endian

Big-endian – most significant value is stored first.
Little-endian - least significant value in the sequence is stored first.

Consider a two-byte number 4F52.

In big-endian computer it will be stored like this:

Addr	Value
1000	4F
1001	52

In little-endian computer it will be stored like this:

Addr	Value
1000	52
1001	4F

Big-Endian (network-order):
- IBM's 370 mainframes
- most RISC-based computers
- Motorola microprocessors
- TCP/IP

Little-endian:
- Intel processors (CPUs)
- DEC Alphas

Note about Mac computers:

~20 years ago Macs were running on big-endian Motorola 68K processors.
Then they switched to PowerPC which were bi-endian (could be placed in big-endian or little-endian mode).
~10 years ago MacOS switched to Intel chipset (little-endian).

Converting COBOL to other languages

Many companies worked on migrating legacy COBOL systems to other languages, typically Java or C#.

```
task:
  ANTLR 4 - https://www.antlr.org/
  google for corresponding grammas (on GitHub):
    https://github.com/antlr/grammars-v4/tree/master/cobol85
    https://github.com/antlr/grammars-v4/blob/master/cobol85/Cobol85.g4

  lex-ing / parsing , creating tree
  translate the tree into another language (python?)

Alternative approach (Renato) – self-made parser.
```

VSAM files => SQL DB => Graph DB

Indexed VSAM file is a file of records with primary key
Nowadays when people need indexed storage – they simply use RDBMS (Relational Database).

Relations become more and more important.
Recent trend is to move from RDBMS to GraphDB, where relationships become 1st class citizens.

New languages for Graph data – (Graph languages).