

**Common Organization Business-Oriented Language**

**Ricard Meyerhofer Parra**

**LP 2015-16 Q1**

**Professor: Albert Rubio**

**Índex**

**Introducció** 3

**Context històric** 3

**Evolució de COBOL** 3

**Popularitat de COBOL** 3

**Llenguatges similars** 4

**Paradigmes de programació** 4

**Què és un paradigma?** 4

**Paradigmes de COBOL** 4

**Imperatiu** 4

**Procedimental** 4

**Estructurat** 4

**Orientat a objectes** 5

**Compilat o interpretat?** 5

**Sistema de tipus de COBOL** 5

**Principals aplicacions** 6

**Altres característiques** 6

**Llenguatge autodocumentat i fàcil de mantenir** 6

**DIVISION** 7

**Exemples** 9

**Bibliografia** 13

**Introducció**

**Context històric**

A finals de la dècada de 1950, els usuaris i productors de software eren cada cop més conscients de l’augment del cost de la programació. L’any 1959 una inspecció va arribar al càlcul que en qualsevol procès d'instal·lació de dades, el cost rondava els US$800,000 i que el traduir aquests perquè funcionessin en un nou hardware en costava uns $600,000. Un altre exemple seria el departament de defensa dels EE.UU que es trobava amb haver d’ensenyar als seus auditors a treballar amb una gran varietat de llenguatges per a poder documentar progressos, costos, etc.

Així doncs en un moment en el qual hi havia un boom de nous llenguatges de programació, es veia una necessitat clara de la creació d’un llenguatge comú orientat al negoci que disminuís de forma significativa aquests costos (fos un llenguatge fàcil d’entendre i no fos dependent de la màquina) i temps de conversió.Així doncs l’any 1959, un comitè anomenat CODASYL (Conference On DAta SYstem Languages) va desenvolupar COBOL. Els membres del comitè procedien de tres sectors: contructores d’ordinadors, usuaris del govern (bàsicament del departament de defensa) de EE.UU i usuaris de grans empreses. La definició del llenguatge es va completar l’any 1960 sota el nom de COBOL-60 inspirant-se en Flow-Matic i COMTRAN (els creadors dels llenguatges Grace Hopper i Bob Bemer van participar en la definició de COBOL) tot i que majoritàriament COBOL està basat en Flow-Matic.

**Evolució de COBOL**

Posteriorment a la seva creació i gràcies als usuaris de COBOL va evolucionar ràpidament i va ser revisat el 1961 i 1965 per afegir-hi noves funcionalitats. El 1968 va ser estandarditzat amb la seva versió ANSI essent revisada posteriorment el 1974, 1985 va ser ampliat amb funcions matemàtiques, l’any 2002 incorpora l’orientació a objectes i la seva última revisió és del 2014.

**Popularitat de COBOL**

* L’any 1970 COBOL ja era el llenguatge més utilitzat del món.
* L’any 1997, Gartner Group va estimar un total de 200 bilions de linees de codi existent, les quals representaven un 80% dels programes de negoci.
* L’any 1999 el 50% de tots els programes de funcionalitats crítiques es feien amb COBOL.
* Enquestes fetes l’any 2006 i 2012 van trobar que més d’un 60% de les organitzacions usen COBOL i que la meitat d’aquests era a nivell de software intern i un 36% dels directius tenen pensat migrar de COBOL alguns d’ells però no ho fan pel preu.

Actualment COBOL està passat de moda, no s’ensenya en les universitats i el programador de COBOL té de mitjana 55 anys però… ha caigut en desús?

Certament en comparació amb el que havia sigut COBOL, ha baixat i molt però si mirem per exemple el TIOBE index d’aquest desembre, trobem COBOL a la posició 20, probablement pel fet que molts programes del sector de negoci encara es mantenen (tal com hem vist per les enquestes del 2012) i es segueixen fent amb COBOL per a no haver d’afrontar el cost de renovació a un altre llenguatge per altra part se segueixen fent revisions del llenguatge per tant no, COBOL no és mort.

**Llenguatges similars**

COBOL tal com hem dit anteriorment es va inspirar en Flow-Matic i COMTRAN doncs els seus creadors van participar en la seva creació. Per altra part COBOL ha inspirat a altres llenguatges com PL/I, ABAP/4, DIBOL i a part COBOL té un gran nombre de dialectes com Micro Focus COBOL, IBM Enterprise COBOL…

**Paradigmes de programació**

**Què és un paradigma?**

Entenem com a paradigma a una orientació o filosofia de programació que s’utilitza al moment d’implementar un programa. Hi ha certs paradigmes que solen ser millors que d’altres per a resoldre certs problemes i viceversa.

**Paradigmes de COBOL**

COBOL al llarg de la seva història ha anat incorporant paradigmes de programació, actualment disposa dels següents:

* Imperatiu.
* Procedimental.
* Estructurat.
* Orientat a objectes.

**Imperatiu**

El paradigma imperatiu on majoritàriament les sentències són assignacions, descriu com realitzar el càlcul en una sèrie de sentències, executades segons un control de flux explícit que modifiquen l’estat del programa el qual són cel·les de memòria que contenen dades/referències i representen l’estat del programa.

**Procedimental**

Indica un model d’organització de programes els quals s’organitzen com col·leccions de subrutines mitjançant invocacions entre elles. Es compleixen les propietats d’encapsulament, independència entre procediments…

**Estructurat**

És un subparadigma del paradigma imperatiu. El punt més notori d’aquest paradigma és que redueix la importància dels GOTO (els quals generen un model de codi espagueti) el qual COBOL va deixar enrere l’any 1985 amb el PERFORM juntament amb la incorporació de subprogrames a niuats...

**Orientat a objectes**

Les classes i les interfícies són presents a COBOL des del 2002. Les classes tenen objectes que contenen mètodes de classes i variables, instàncies d’objectes que contenen instàncies de mètodes i variables. L’herència i les interfícies proporcionen polimorfisme.

**Compilat o interpretat?**

COBOL es tracta d’un llenguatge compilat tot i que en podem trobar intèrprets. El fet de ser un llenguatge compilat i no un llenguatge interpretat ens beneficia doncs en general les implementacions dels llenguatges compilats són més ràpides, ja que tradueixen directament a codi específic per a la màquina i són més eficients, però també els llenguatges compilats solen ser menys portables que els interpretats (cal dir que tot això depèn molt de les implementacions i només ho podem dir en termes generals però en un llenguatge com COBOL que porta 56 anys a les seves espatlles amb optimitzacions de compiladors segur que és més ràpid que un intèrpret).

**Sistema de tipus de COBOL**

El tipat de COBOL es tracta d’un tipat feble i estàtic.

Tipat feble doncs podem trobar els següents errors de tipus:

* Alliberament explícit de memòria (deallocate/delete).
* Type cast doncs a COBOL una variable numèrica pot ser representada per exemple per una variable alfanumèrica.
* Aritmètica de punters no es dóna en COBOL.

Estàtic que implica que la comprovació de tipus es realitza en temps de compilació, la majoria d’errors de tipat es detecti via compilador i que sigui necessari declarar el tipus de les variables.

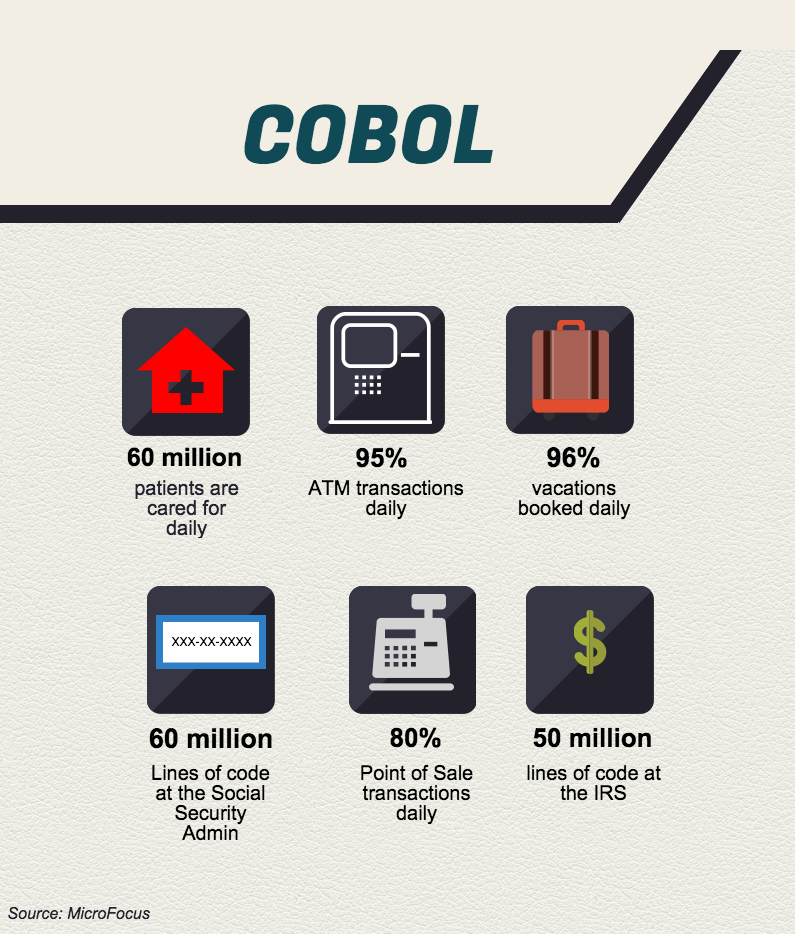
És un llenguatge de tipus segur, ja que no permet errors de tipus durant l’execució del programa, si hi ha d’haver un error de tipus, el notificarà en la compilació (és estàtic de fet, COBOL és un llenguatge ràpid).

Originàriament COBOL (COBOL-60) només disposava de 3 tipus de dades:

* Numeric.
* Alphabetic.
* Alphanumeric.

Posteriorment s’han afegit altres tipus com boolean o bit (COBOL 2002).

**Principals aplicacions**



Les principals aplicacions de COBOL tal com es pot veure una mica per la seva història (marginats de la comunitat de ciències de computació des dels seus inicis i el motiu de la seva creació), pel tipus de programes desenvolupats fins ara i pel mateix nom (Common Oriented-**Business** Language) doncs podem dir que la seva principal aplicació el desenvolupament de programes enfocats a negoci i món empresarial i no a l’aplicació científica o d’enginyeria (de fet COBOL tenia unes capacitats de càlcul inferiors a FORTRAN per exemple).

Així doncs podem trobar COBOL sobretot en sector financer, de negoci i administratius per a empreses i governs.

COBOL actualment segueix sent molt utilitzat en legacy applications implementades en computadores centrals majoritàriament programes de processament a gran escala per lots i feines de gestió de transaccions.

**Altres característiques**

**Llenguatge autodocumentat i fàcil de mantenir**

Un dels suposats avantatges de COBOL (doncs també és un dels motius pel qual és tan criticat) és el fet que COBOL és un llenguatge molt proper a l’anglès pensat perquè amb poca preparació fos entenedor (el qual implica que gent que no sap programar, pugui entendre’l). Això certament l’apropa al llenguatge natural i el fa més proper a nosaltres però, per altra part ens genera una sintaxi molt farragosa que resulta pesada i a part provoca que hi hagi moltes paraules reservades (més de 300) i redundància al moment d’expressar una mateixa cosa com per exemple:

|  |
| --- |
| x **IS GREATER THAN** y |
| x **GREATER** y |
| x **>** y |

El fet de ser fàcil de mantenir és quelcom que va de la mà amb el fet de ser un llenguatge entenedor doncs serà més fàcil de mantenir, veure errors i solucionar-los.

**DIVISION**

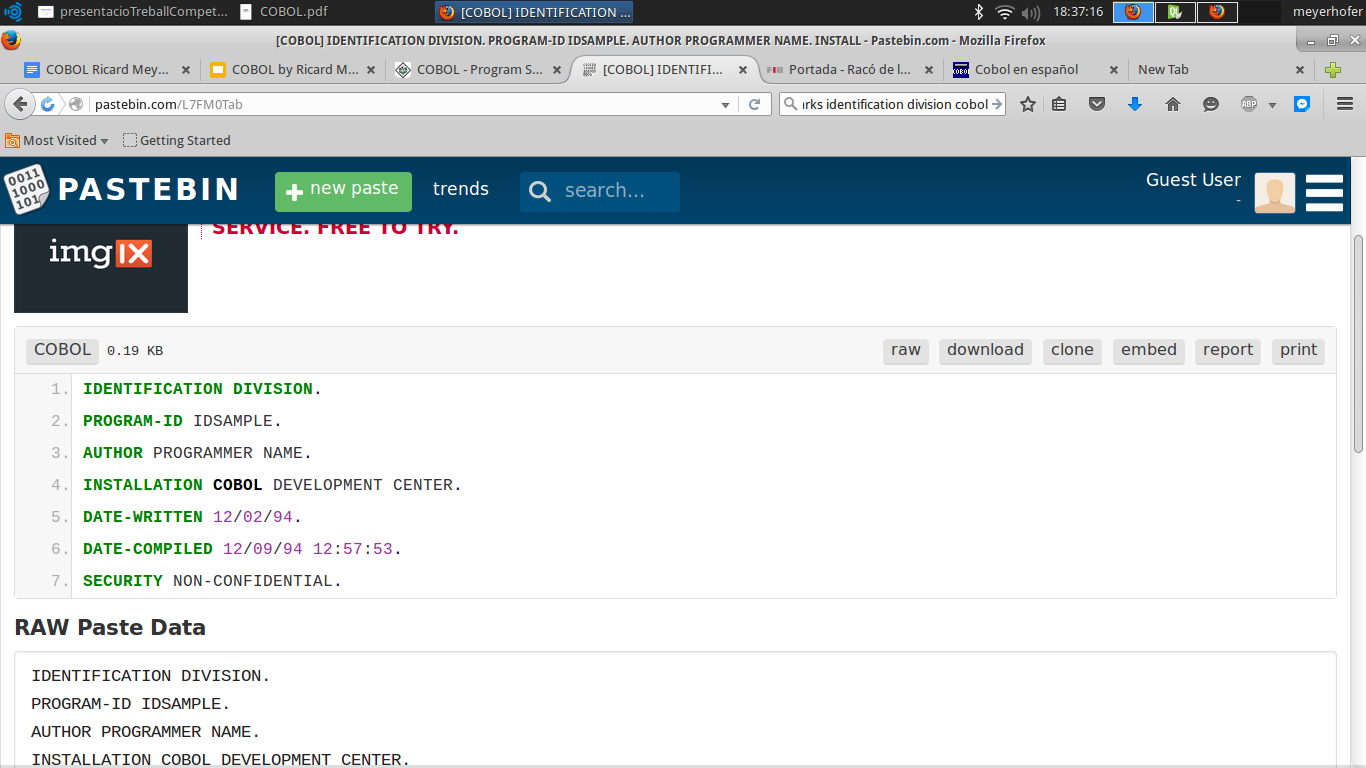
Tot programa en COBOL es divideix en 4 parts anomenades DIVISION. Cada una d’aquestes, realitza una funció dins del programa i té unes capçaleres, les quals poden ser obligatòries o opcionals.

### **Identification Division**

És la primera DIVISION de tot programa i l’única que obligatòriament ha d’aparèixer. Són utilitzades pel programador i pel compilador per a identificar el programa. En aquesta DIVISION, el paràgraf PROGRAM-ID (que especifica el nom del programa i té un màxim de 30 caràcters) ha d’aparèixer obligatòriament.

La resta de capçaleres són opcionals i proporcionen informació sobre el programa. Les quals són:

* AUTHOR
* INSTALLATION
* DATE-WRITTEN
* DATE-COMPILED
* SECURITY



### **Environment Division**

Aquesta divisió s’utilitza per a especificar els fitxers d’input/output del programa. Conté dues seccions:

**Configuration section**

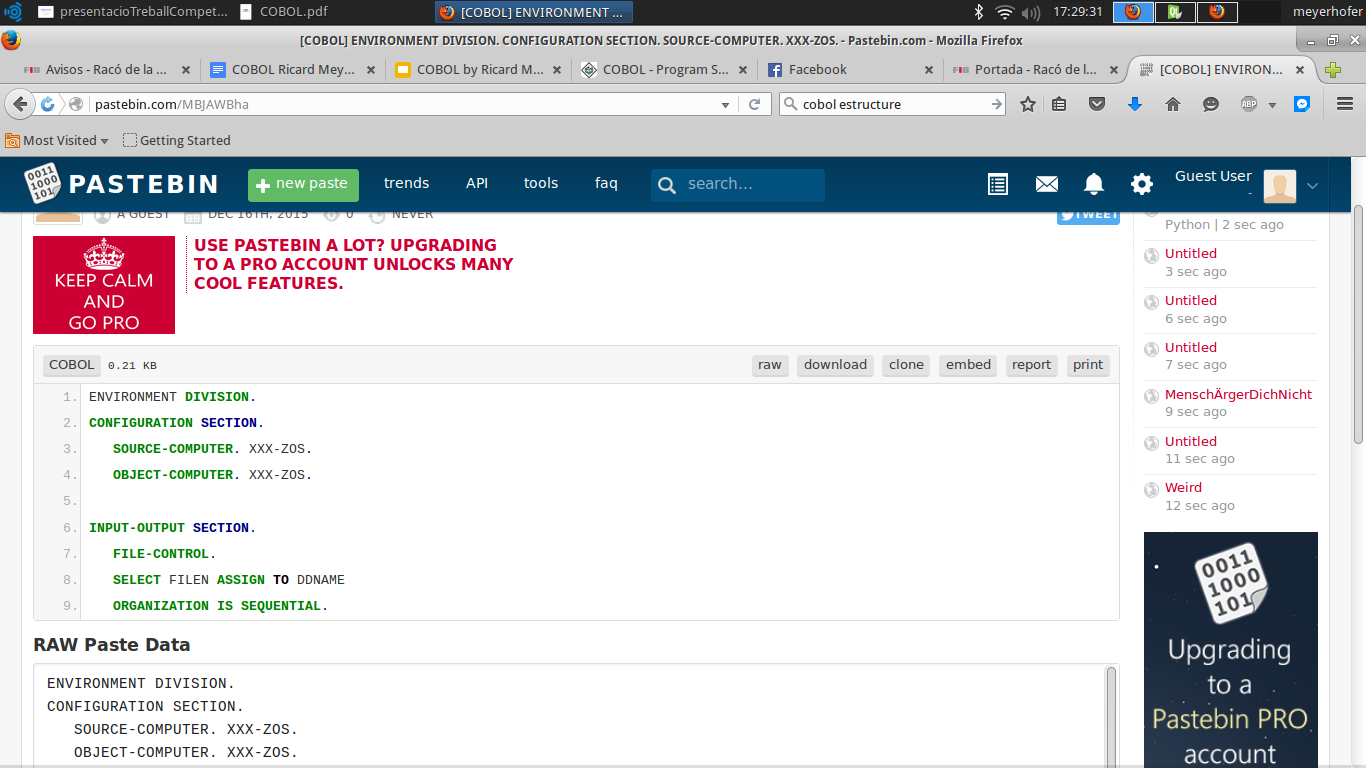
Proporciona informació sobre sistemes on el programa és escrit i executat. Està format per dos paràgrafs:

* Source computer: Emprat pel sistema per a compilar el programa.
* Object computer: Emprat pel sistema per a executar el programa.

**Input-Output section**

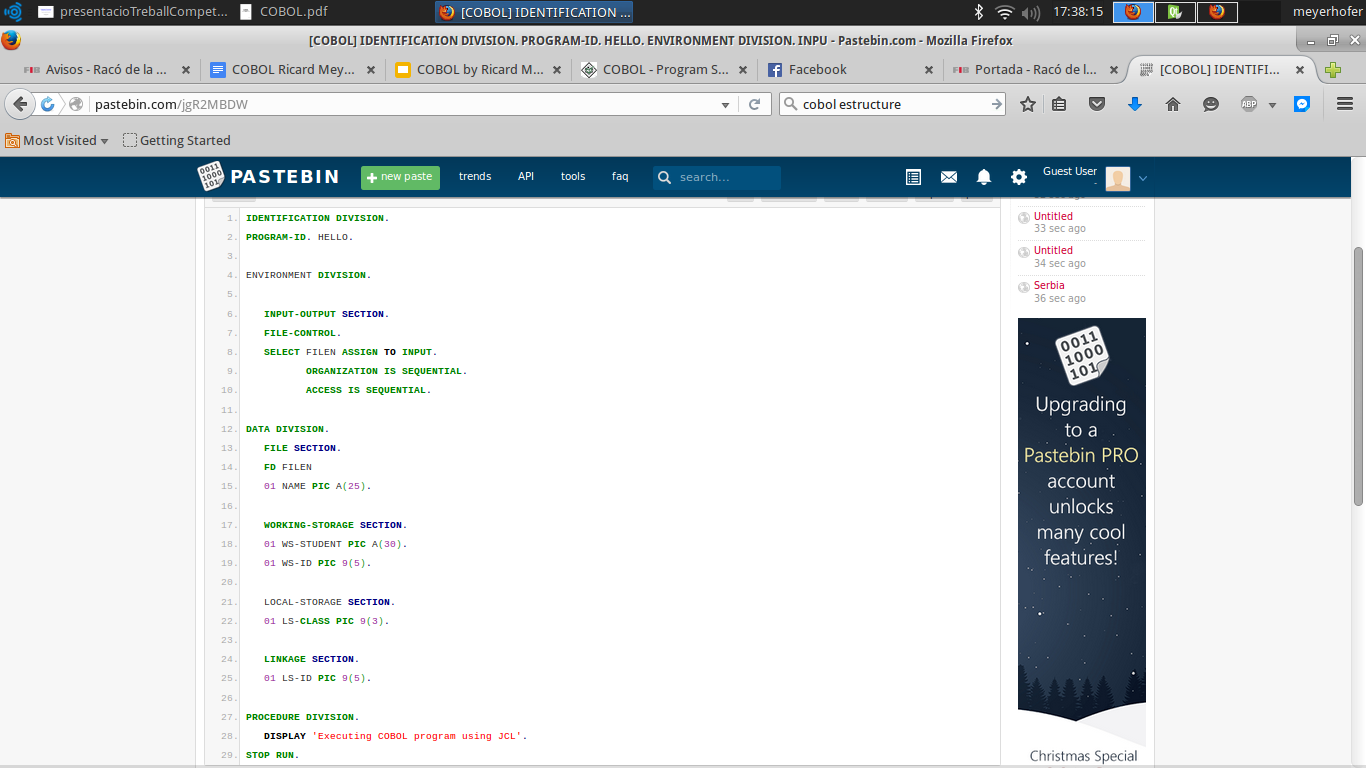
Proporciona informació sobre els fitxers que han de ser usats pel programa. Consisteix en dos paràgrafs:

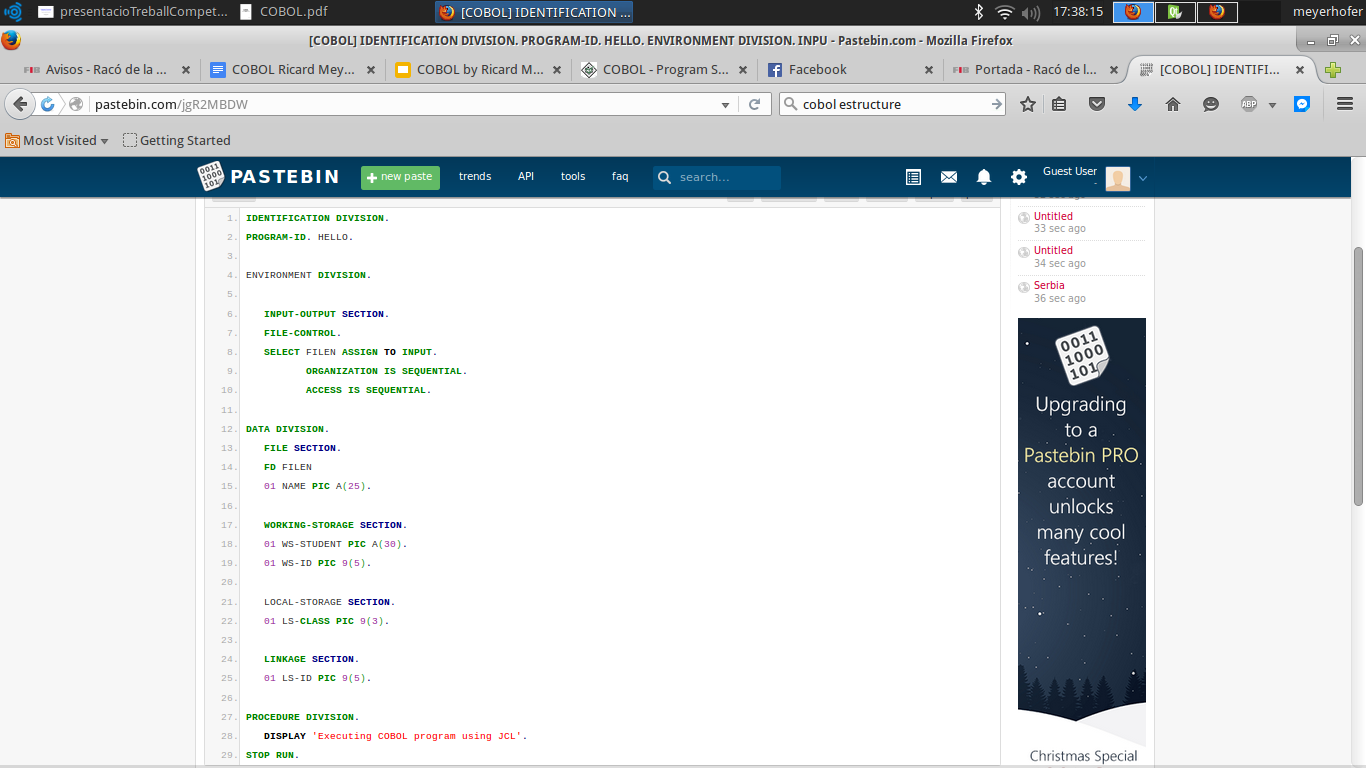
* File control: Proveeix informació externa de les dades usades en el programa.
* I-O control: Proporciona informació dels fitxers usats pel programa.



### **Data Division**

Aquesta divisió s’utilitza per a definir les variables usades, està formada per 4 parts:

* File section: S’utilitza per a definir l’estructura del fitxer.
* Working-Storage section: S’utilitza per a declarar variables temporals i estructures de fitxers que s’usen en el programa.
* Local-Storage section: Similar a l’anterior però es diferencia en el fet que les variables s’inicialitzaran cada cop que el programa comenci una execució.
* Linkage section: S’usa per a descriure els noms que es reben d’una crida externa.

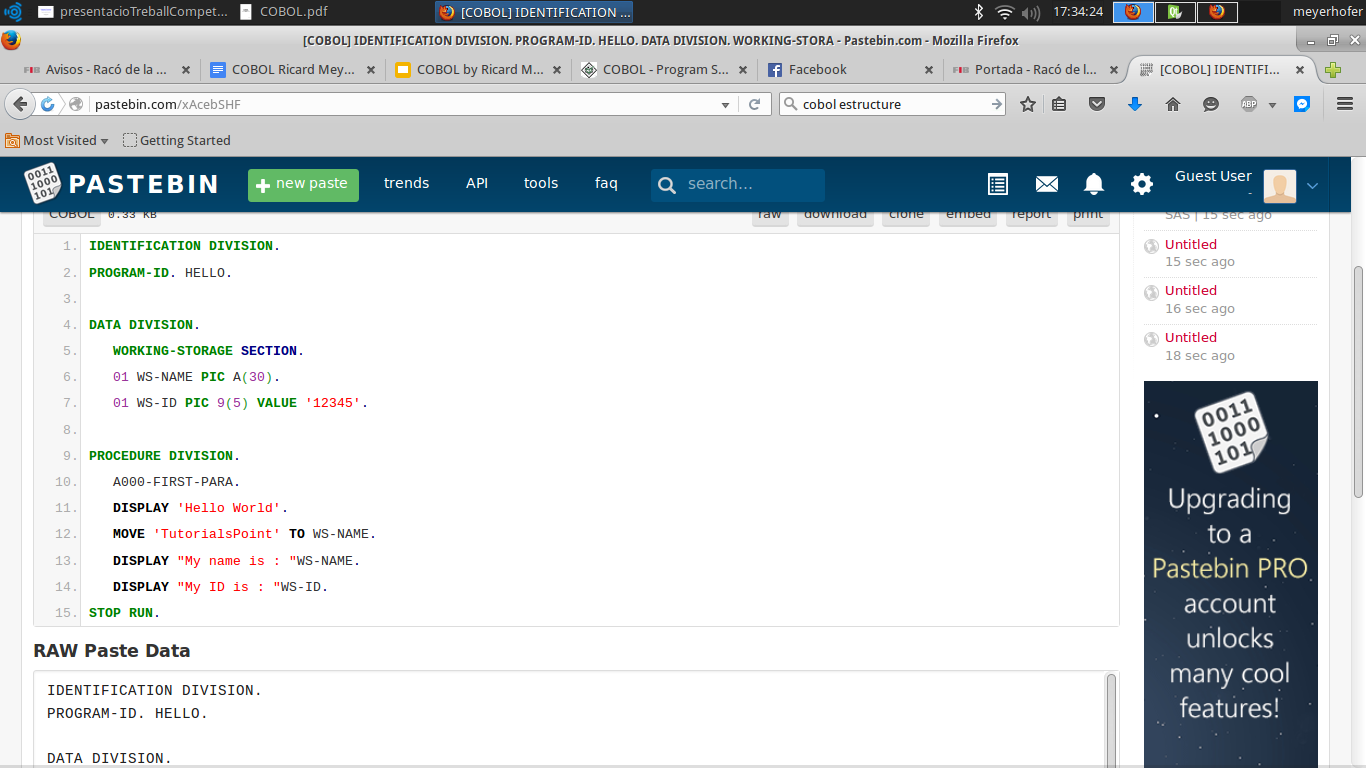


**Procedure Division**

És on hi ha la lògica del programa. Consisteix en un conjunt de statements que utilitzen variables definides a la data division. En aquesta divisió els paràgrafs i les seccions són definides per l’usuari.

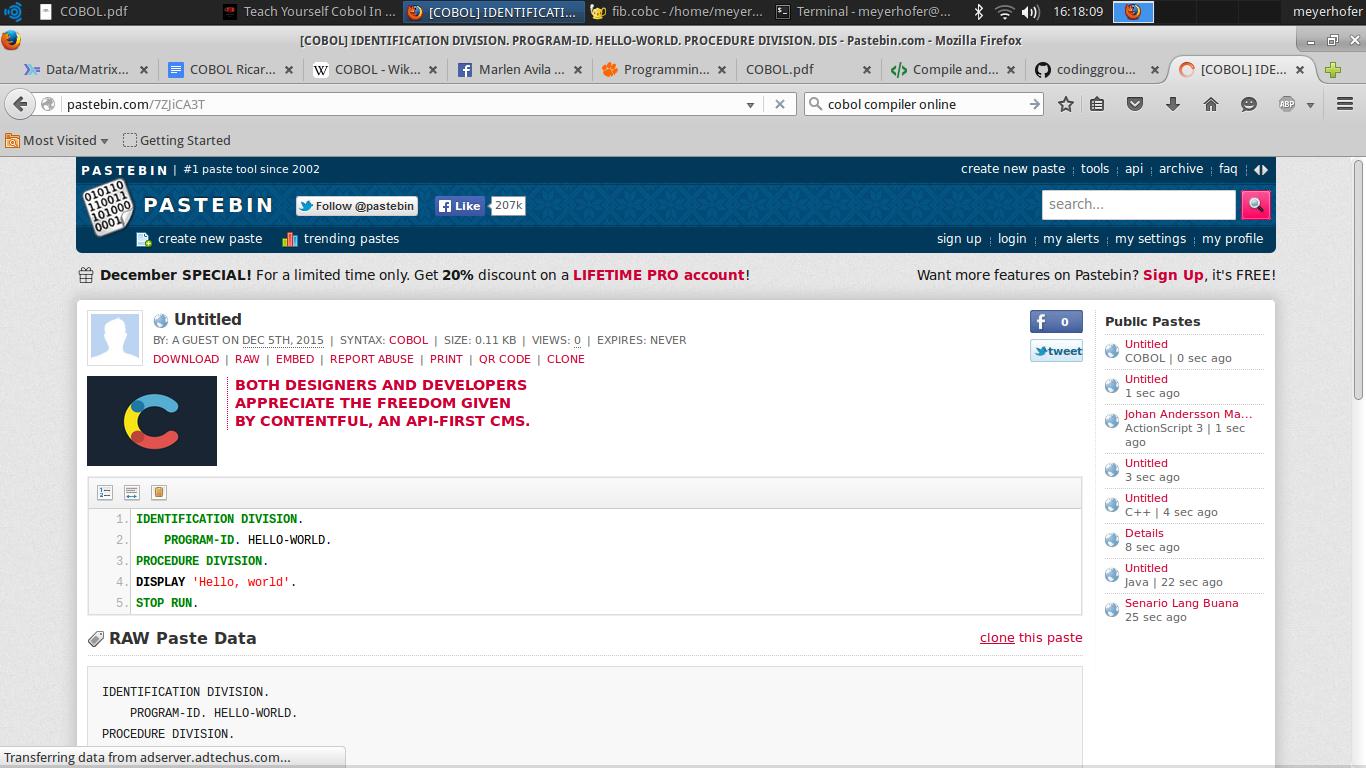
Requisits de la DIVISION:

* Almenys ha de contenir un statement
* L’últim statement ha de ser o un STOP RUN o un EXIT PROGRAM



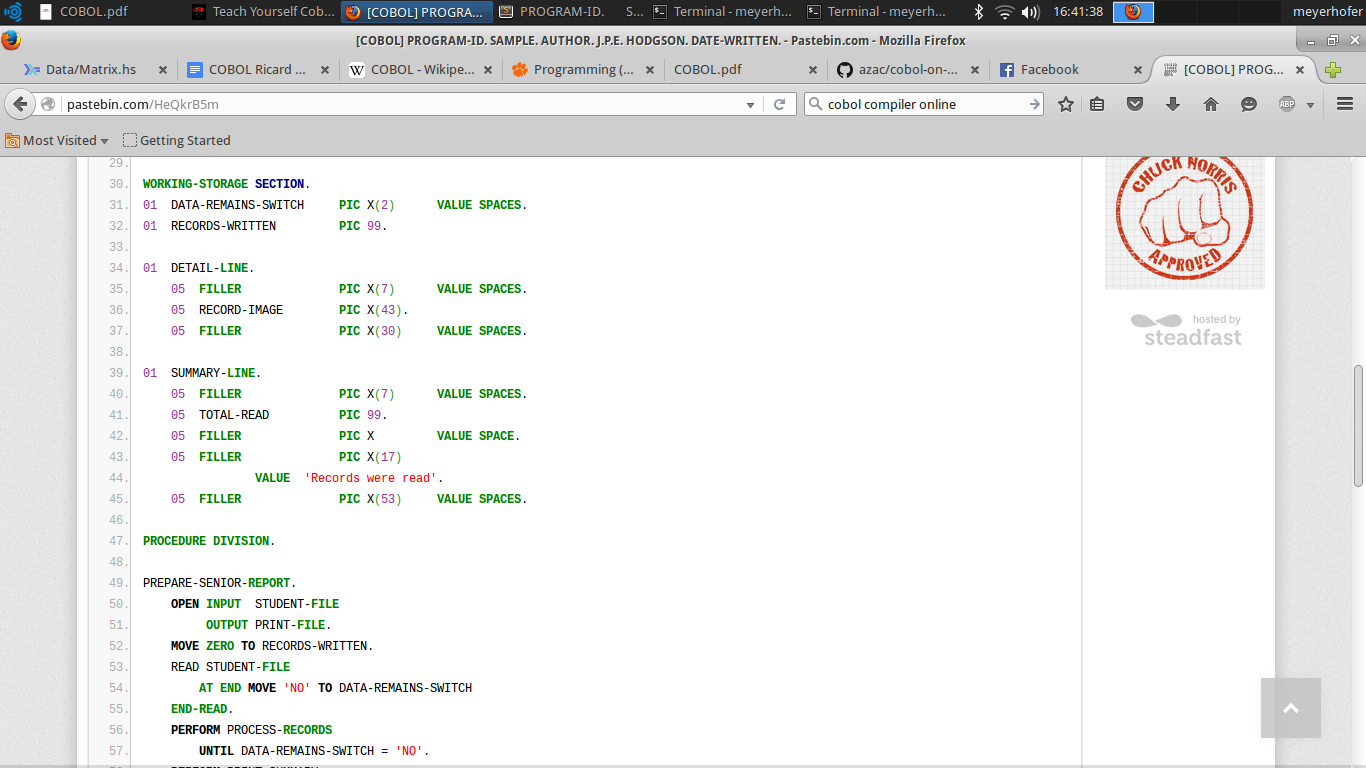
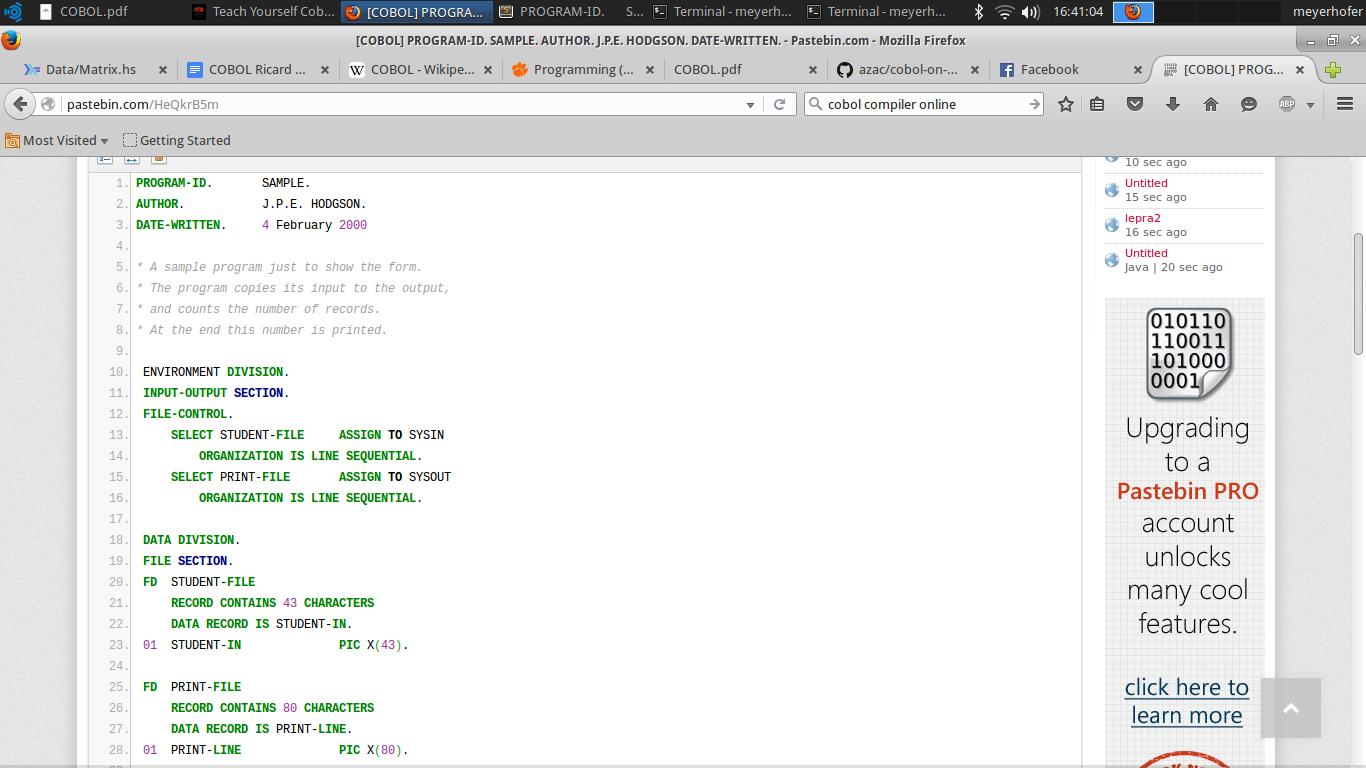
**Exemples**

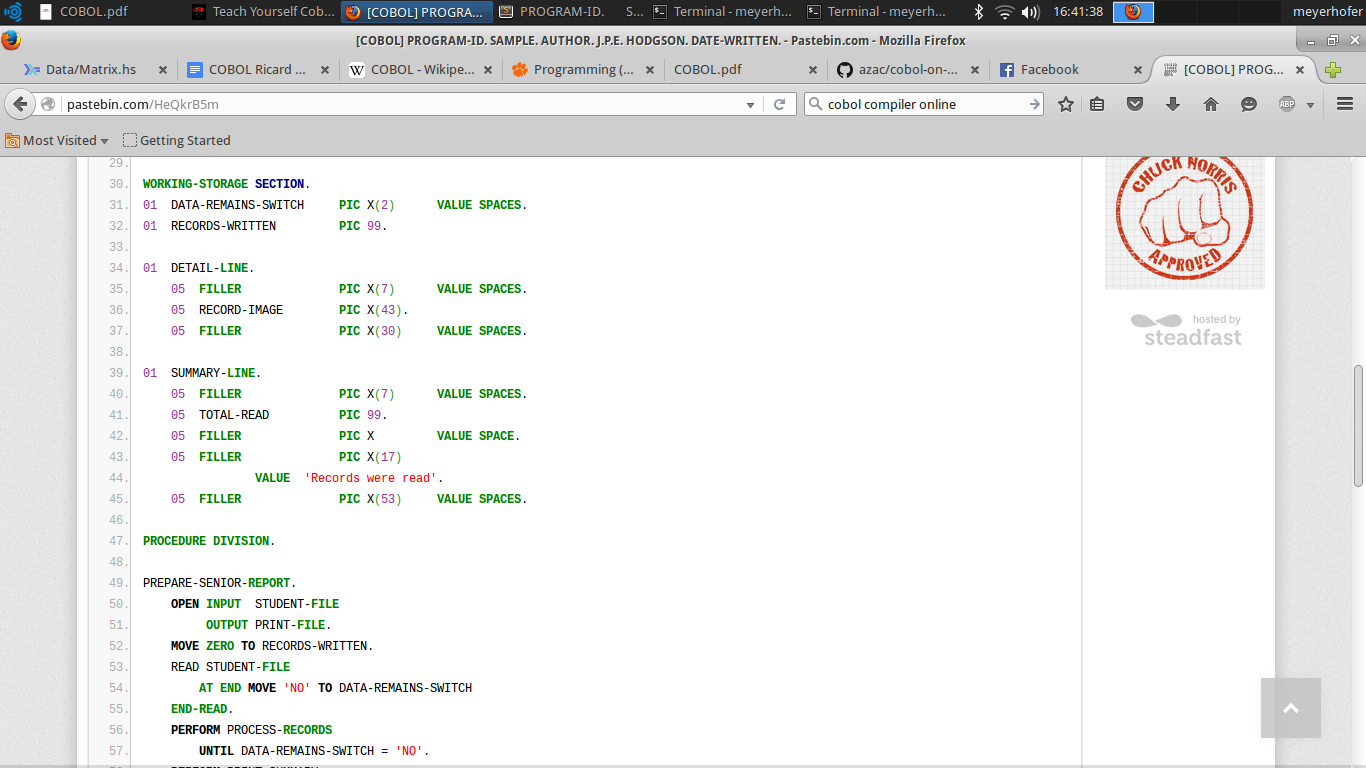
**Hello World:**

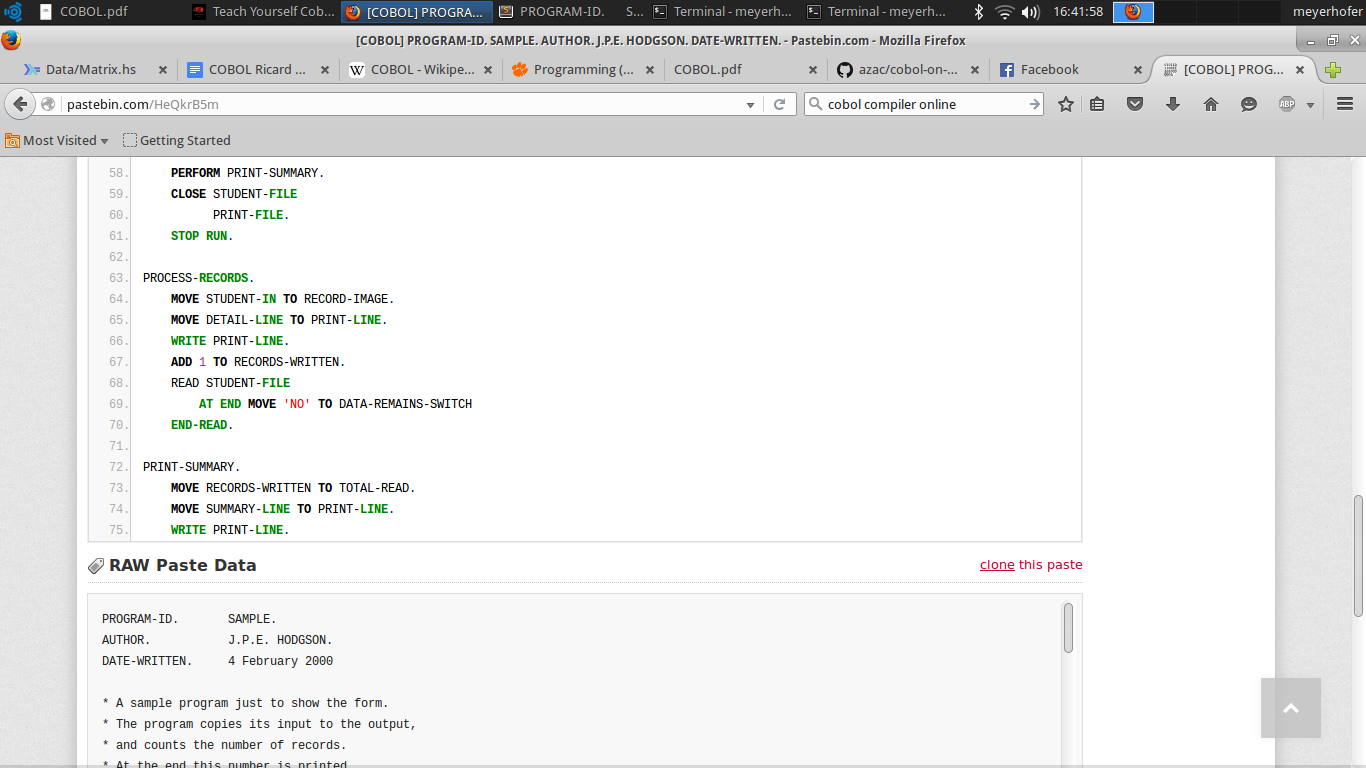


**Calcula els primers 90 termes de la successió de Fibonacci:**



**Copia l’input a l’output i en printeja el nombre al final:**





**Bibliografia**

**Introducció:**

<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

<https://en.wikipedia.org/wiki/COBOL>

<http://blog.hackerrank.com/the-inevitable-return-of-cobol/>

**Paradigmes:**

<http://people.cs.clemson.edu/~turner/courses/cs428/current/webct/content/pz/ch2/ch2_6.html>

**Compilat o interpretat?**

<http://www.cs.uky.edu/~paulp/CS216F12/CS216ProgNote1.htm>

**Sistema de tipus**

<http://www.csee.umbc.edu/courses/graduate/631/Fall2002/COBOL.pdf>

<http://www.infor.uva.es/~cvaca/asigs/docpar/imperativo.pdf>

<https://books.google.es/books?id=VZOcbYCxpQgC&pg=PA10&lpg=PA10&dq=cobol+strongly+typed&source=bl&ots=t7b_2GUUpm&sig=wZ27-a9WWWrAhz9PMo6YzXzsXEU&hl=es&sa=X&ved=0ahUKEwiUmPvWncXJAhXFvRoKHfPCDvIQ6AEINzAD#v=onepage&q=cobol%20strongly%20typed&f=false>

<http://pages.cs.wisc.edu/~raghavan/tacas05.pdf>

**1er exemple:**

Teach Yourself COBOL in 21 Days Autor [Mo Budlong](http://dl.acm.org/author_page.cfm?id=81100397278&CFID=574515666&CFTOKEN=74198723)

**Segon exemple:**

<https://github.com/biracruz/codingground/blob/15b6e5727b98f9ca6e779bdffbd3a793463bc76e/cobol/fibonacci.cobc>

**Tercer exemple:**

<http://www.csee.umbc.edu/courses/graduate/631/Fall2002/COBOL.pdf>

**COBOL report abril 1960**

<https://archive.org/stream/bitsavers_codasylCOB_6843924/COBOL_Report_Apr60#page/n117/mode/2up/search/compiled>

**COBOL 2002**

<http://www.ibm.com/developerworks/rational/cafe/attachments/3149-1-2179/S8204%20-%20Boston%20-%202002%20COBOL%20-%20Good,%20Bad,%20and%20the%20Ugly.pdf>

**Efecte 2000:**

http://www.elmundo.es/su-ordenador/99/SOR159/SOR159e2000.html

**Altres característiques:**

<http://www.tutorialspoint.com/cobol/cobol_program_structure.htm>

<http://www.escobol.com/modules.php?name=Sections&op=viewarticle&artid=12>

**Llibres usats en General juntament per corroborar i complementar la informació web que he anat trobant:**

COBOL for the 80's Autor: J. Wayne Spence

COBOL Autor: Javier Ceballos