



- [Home](#)
  - [About](#)
  - [Business Plan »](#)
  - [Communication »](#)
  - [Dieting](#)
  - [Sales](#)
  - [Sitemap](#)
  - [Videos »](#)
  - [Web Design »](#)
- 
- [Communication »](#)
  - [Diet Nutritional](#)
  - [Flash Tutorial](#)
  - [How To »](#)
  - [Investing](#)
  - [iPad »](#)
  - [Marketing »](#)
  - [Most Popular](#)
  - [Royalty Free Photos](#)
  - [Sales](#)
  - [Web Design »](#)

[share](#)

## Learn COBOL in One Video

Posted by [Derek Banas](#) on Apr 23, 2020 in [Web Design](#) | [0 comments](#)



COBOL runs the US Financial System, Social Security Administration, Department of Defense, Internal Revenue Service, the Majority of State Financial / Unemployment Systems and Numerous other Critical Systems. Currently there is a critical need to create more COBOL Programmers.

In this one video I did my best to cover what you'd learn about COBOL in a standard 500 page book. If there is a demand for more videos I will make them here for free.

### COBOL Tutorial : Learn COBOL in One Video



- ▶▶ Get my Python Programming Bootcamp Series for \$9.99 ( Expires April 28th ) : <https://bit.ly/SavePython8>
- ▶▶ Highest Rated Python Udemy Course + 23 Hrs + 102 Videos + New Videos Every Week

### COBOL CHEAT SHEET

#### COBOL CHEAT SHEET

```
2
3  INSTALL ON WINDOWS : https://www.it-cooking.com/projects/how-to-install-gnucobol-for-cygwin/
4
```

5 COBOL (Common Business Oriented Language) is the most rigidly structured programming language. It focuses on data oriented business  
6 [SWITCH TO CODE]  
7  
8 COBOL was created to match natural language. The COBOL program is made up of paragraphs, sections, sentences, verbs and other components.  
9  
10 >>SOURCE FORMAT FREE  
11  
12 The 1st 6 characters are reserved for sequence numbers used for organizing punch cards. The 7th is reserved for an \* which denotes a comment.  
13  
14 IDENTIFICATION DIVISION.  
15 PROGRAM-ID. hello.  
16 ENVIRONMENT DIVISION.  
17 DATA DIVISION.  
18 PROCEDURE DIVISION.  
19  
20 There are 4 divisions being identification, environment, data and procedure.  
21  
22 A program is structured into program, division, section, paragraph, sentence and statements. Programs contain divisions. Divisions contain sections.  
23  
24 IDENTIFICATION DIVISION.  
25 PROGRAM-ID. intro.  
26 AUTHOR. Derek Banas .  
27 DATE-WRITTEN. April 20th 2020  
28  
29 The identification division contains information about the program. Like the name that is used to call for this program's code to be executed.  
30  
31 ENVIRONMENT DIVISION.  
32 INPUT-OUTPUT SECTION.  
33 FILE-CONTROL.  
34  
35 Environment contains environment info being the computer it is running on, devices available, country specific information.  
36  
37  
38  
39  
40  
41  
42  
43  
44 DATA DIVISION.  
45 FILE SECTION.  
46 WORKING-STORAGE SECTION.  
47  
48 Data describes the data and has 4 sections being the file, working-storage, linkage and report. The File section describes data files.  
49  
50 hello.cob  
51  
52 >>SOURCE FORMAT FREE  
53 \*> Compile with

```

54 *> cobc -x hello.cob      cobc -x hello.cbl
55 *> ./hello
56 IDENTIFICATION DIVISION.
57 *> Define name used to call for this program
58 PROGRAM-ID. hello.
59 AUTHOR. Derek Banas .
60 DATE-WRITTEN. April 15th 2020
61 DATA DIVISION.
62
63 -----
64
65 Data
66
67 There are 3 types of data being Numeric, Alphanumeric and Alphabetic. While you can declare a variable has a certain type you can
68
69 3 Types of Data
70 字符串
71 Literals are constants that are either Alphanumeric (surrounded by quotes) or Numeric (+- Signed, floats or integers).
72
73 You can define a constant like this 01 PIValue CONSTANT AS 3.14.
74 常量
75 Figurative Constants
76 zeros
77 Figurative constants are named values that can be used to write a value to every character in a variable. MOVE ZEROS TO PayCheck
78
79 ZERO, ZEROES, ZEROS : Assigns zero
80 SPACE, SPACES : Assigns a space FF
81 HIGH-VALUE, HIGH-VALUES : Assigns largest value of defined type
82 LOW-VALUE, LOW-VALUES : Assigns smallest value of defined type 00
83
84 -----
85
86 WORKING-STORAGE SECTION. declare data type
87 *> Can hold a alphanumeric with max length
88 *> of 30 and starting value You
89 01 UserName PIC X(30) VALUE "You".
90
91 *> Declare a single digit integer between 0-9
92 *> with a starting value of 0
93 *> ZEROS is a constant equal to 0
94 01 Num1 PIC 9 VALUE ZEROS.
95 01 Num2 PIC 9 VALUE ZEROS.
96
97 *> Double digit int between 0-99 with starting
98 *> value of 0
99 01 Total PIC 99 VALUE 0.
100
101 *> Hierarchal variable
102 01 SSNum.

```

level fieldName  
 级 名字 类型 长度 值  
 01 UserName PIC X(30) "You"

① PIC = picture clause

② X : alphanumeric

③ 9 : 1位数字

④ 99. 3位数字

⑤ zeros == 0

```

103      02 SSArea    PIC 999.
104      02 SSGroup   PIC 99.
105      02 SSSerial  PIC 9999.
106
107  PROCEDURE DIVISION.
108  *> Displays the string and doesn't skip to a newline
109  DISPLAY "What is your name " WITH NO ADVANCING
110  *> Stores the value entered
111  ACCEPT UserName
112  DISPLAY "Hello " UserName
113
114  MOVE ZERO TO UserName
115  DISPLAY UserName
116
117  DISPLAY "Enter 2 values to sum "
118  ACCEPT Num1
119  ACCEPT Num2
120  *> Solves the problem and stores it in Total
121  COMPUTE Total = Num1 + Num2
122  DISPLAY Num1 " + " Num2 " = " Total
123  DISPLAY "Enter your social security number "
124  *> Receive and output part of SSNum
125  ACCEPT SSNum
126  DISPLAY "Area " SSArea
127
128  *> Ends the program
129  STOP RUN.
130
131  tutorial2.cob
132
133      >>SOURCE FORMAT FREE
134  IDENTIFICATION DIVISION.
135  PROGRAM-ID. tutorial2.
136  AUTHOR. Derek Banas .
137  DATE-WRITTEN. April 15th 2020
138  DATA DIVISION.
139
140  WORKING-STORAGE SECTION.
141  *> Defines an alphanumeric type 10 spaces long
142  *> with the default value of "Stuff" (System Defined Max Length)
143  *> The Picture Clause is where we define the
144  *> data type and size (COBOL isn't a typed language)
145  *> X means any type of character on your keyboard
146  01 SampleData PIC X(10) VALUE "Stuff".
147
148  *> Only letters and spaces are allowed
149  01 JustLetters PIC AAA VALUE "ABC".
150
151  *> 4 of numbers from 0 to 9 (31 max values)

```

Display " " with no advancing  
 Accept UserName  
 Display ↓

pic X (10) value " "

```

152 01 JustNums PIC 9(4) VALUE 1234.
153
154 *> Signed number
155 01 SignedInt PIC S9(4) VALUE -1234.
156
157 *> 4 digit decimal with 2 decimal places
158 01 PayCheck PIC 9(4)V99 VALUE ZEROS.
159
160 *> A Group Item is a collection of values
161 *> They are structured using level numbers
162 *> where the highest number is lowest
163 *> in the hierarchy (01 - 49)
164 01 Customer.
165     02 Ident PIC 9(3).
166     02 CustName PIC X(20).
167     02 DateOfBirth.
168         03 MOB PIC 99.
169         03 DOB PIC 99.
170         03 YOB PIC 9(4).
171
172 01 Num1 PIC 9 VALUE 5.
173 01 Num2 PIC 9 VALUE 4.
174 01 Num3 PIC 9 VALUE 3.
175 01 Ans PIC S99V99 VALUE 0. - 90.81
176 01 Rem PIC 9V99.
177
178 PROCEDURE DIVISION.
179 *> MOVE is used to assign values
180 MOVE "More Stuff" TO SampleData
181 MOVE "123" TO SampleData
182 *> You can assign numerics to alphanumerics
183 *> because typing isn't enforced
184 MOVE 123 TO SampleData
185 DISPLAY SampleData
186 DISPLAY PayCheck
187 *> Entering data this way requires additional
188 *> filled spaces
189 MOVE "123Bob Smith" 12211974" TO Customer
190 DISPLAY CustName
191 DISPLAY MOB "/" DOB "/" YOB
192
193 *> Figurative Constants
194 *> Zero in every space
195 MOVE ZERO TO SampleData
196 DISPLAY SampleData
197 *> Space in every space
198 MOVE SPACE TO SampleData
199 DISPLAY SampleData
200 *> Question mark in every space

```

PIC 9999 = PIC 9(4)

signed Num PIC S9(4) VALUE -1234.

01 paycheck PIC 9(4)V99 VALUE 12.34

123

ID + CustName (20)

12/21/1974

00 000 00000

```

201 MOVE HIGH-VALUE TO SampleData
202 DISPLAY SampleData
203 *> Space in every space
204 MOVE LOW-VALUE TO SampleData
205 DISPLAY SampleData
206 *> Quote in every space
207 MOVE QUOTE TO SampleData
208 DISPLAY SampleData
209 *> Defined value in every space
210 MOVE ALL "2" TO SampleData
211 DISPLAY SampleData
212
213 *> Math
214
215 *> Add Num1 to Num2 and store in Ans
216 ADD Num1 TO Num2 GIVING Ans
217 SUBTRACT Num1 FROM Num2 GIVING Ans
218 MULTIPLY Num1 BY Num2 GIVING Ans
219 DIVIDE Num1 INTO Num2 GIVING Ans
220 *> You can get quotient and remainder separately
221 DIVIDE Num2 INTO Num1 GIVING Ans REMAINDER Rem
222 DISPLAY "Remainder " Rem
223
224 *> Using more then 2 variables
225 ADD Num1, Num2 TO Num3 GIVING Ans
226 ADD Num1, Num2, Num3 GIVING Ans
227 DISPLAY Ans
228
229 *> Can also use COMPUTE
230 COMPUTE Ans = Num1 + Num2
231 COMPUTE Ans = Num1 - Num2
232 COMPUTE Ans = Num1 * Num2
233 COMPUTE Ans = Num1 / Num2
234 DISPLAY Ans
235 *> 5 to the power of 2
236 COMPUTE Ans = Num1 ** 2
237 DISPLAY Ans
238
239 *> Using parentheses
240 COMPUTE Ans = (3 + 5) * 5
241 DISPLAY Ans
242 COMPUTE Ans = 3 + 5 * 5
243 DISPLAY Ans
244
245 *> You can round output
246 COMPUTE Ans ROUNDED = 3.0 + 2.005
247 DISPLAY Ans
248
249 STOP RUN.

```

?.?.?.?.?.?.?.?.

2(10)

giving

} giving A remainder B

Compute Result = A + B

Rounded = .....

```

250
251 tutorial3.cob
252
253     >>SOURCE FORMAT FREE
254 IDENTIFICATION DIVISION.
255 PROGRAM-ID. tutorial3.
256 AUTHOR. Derek Banas .
257 DATE-WRITTEN. April 15th 2020
258
259 *> Define a custom data classification
260 ENVIRONMENT DIVISION.
261 CONFIGURATION SECTION.
262 SPECIAL-NAMES.
263     CLASS PassingScore IS "A" THRU "C", "D".
264
265 DATA DIVISION.
266 WORKING-STORAGE SECTION.
267     01 Age PIC 99 VALUE 0.
268     01 Grade PIC 99 VALUE 0.
269     01 Score PIC X(1) VALUE "B".
270
271     01 CanVoteFlag PIC 9 VALUE 0.
272         88 CanVote VALUE 1.
273         88 CantVote VALUE 0.
274
275 *> Used to demonstrate evaluate
276 01 TestNumber PIC X.
277     *> Level 88 designates multiple values
278     88 IsPrime      VALUE "1", "3", "5", "7".
279     88 IsOdd        VALUE "1", "3", "5", "7", "9".
280     88 IsEven       VALUE "2", "4", "6", "8".
281     88 LessThan5    VALUE "1" THRU "4".
282     88 ANumber      VALUE "0" THRU "9".
283
284 PROCEDURE DIVISION.
285
286 *> If Conditional
287 DISPLAY "Enter Age : " WITH NO ADVANCING
288 ACCEPT Age
289 IF Age > 18 THEN
290     DISPLAY "You can vote"
291 ELSE
292     DISPLAY "You can't vote"
293 END-IF
294
295 *> Logical and Conditional Operators
296 *> ELSE IF statements are to be avoided
297 *> < or LESS THAN
298 *> > or GREATER THAN

```

} Grade

} VoteFlag

} testNumber

} If  
Else  
End-If



```

299 *> = or EQUAL TO
300 *> NOT EQUAL TO
301 IF Age LESS THAN 5 THEN
302     DISPLAY "Stay Home"
303 END-IF
304 IF Age = 5 THEN
305     DISPLAY "Go to Kindergarten"
306 END-IF
307 *> You can also use OR instead of AND
308 IF Age > 5 AND Age < 18 THEN
309     COMPUTE Grade = Age - 5
310     DISPLAY "Go to Grade " Grade
311 END-IF
312 *> <= or LESS THAN OR EQUAL TO
313 IF Age GREATER THAN OR EQUAL TO 18
314     DISPLAY "Go to college"
315 END-IF
316
317 *> You can verify values fit a classification
318 IF Score IS PassingScore THEN
319     DISPLAY "You Passed"
320 ELSE
321     DISPLAY "You Failed"
322 END-IF
323
324 *> There are built in classifications
325 *> NUMERIC, ALPHABETIC, ALPHABETIC-LOWER
326 *> ALPHABETIC-UPPER
327 IF Score IS NOT NUMERIC THEN
328     DISPLAY "Not a Number"
329 END-IF
330
331 *> Use set to toggle values to true or false
332 IF Age > 18 THEN
333     SET CanVote TO true
334 ELSE
335     SET CanVote TO false
336 END-IF
337 DISPLAY "Vote " CanVoteFlag
338
339 *> Evaluate performs a certain action based on
340 *> which value is assigned to a variable
341 DISPLAY "Enter Single Number or X to Exit : "
342 ACCEPT TestNumber
343 *> Will execute until data that isn't a numeric is entered
344 *> Used for iteration which I'll cover next
345 PERFORM UNTIL NOT ANumber
346 *> Executes different displays based on condition met
347 *> Only one match can occur

```

} if A, display "

} if A and/or B then .....

} less/greater than /equal to

}

} If A, set B to true/false

```

348     EVALUATE TRUE
349         WHEN IsPrime     DISPLAY "Prime"
350         WHEN IsOdd       DISPLAY "Odd"
351         WHEN IsEven      DISPLAY "Even"
352         WHEN LessThan5   DISPLAY "Less than 5"
353         WHEN OTHER      DISPLAY "Default Action"
354     END-EVALUATE
355     ACCEPT TestNumber
356 END-PERFORM

```

} evaluate ...  
Accept

```

357
358 STOP RUN.
359
360 tutorial4.cob
361
362     >>SOURCE FORMAT FREE
363 IDENTIFICATION DIVISION.
364 PROGRAM-ID. tutorial4.
365
366 PROCEDURE DIVISION.
367 *> Gravity driven programming falls through the
368 *> code until a condition or goto redirects it
369 *> Open paragraphs are executed through gravity
370 *> while closed paragraphs are executed by name.
371 *> Open paragraphs are basically ways to name blocks
372 *> of code (tags).
373 *> Data created in a closed paragraph can't be
374 *> accessed outside of it. They are traditional functions.
375
376 *> This demonstrates the flow
377 SubOne.
378     DISPLAY "In Paragraph 1"
379     PERFORM SubTwo
380     DISPLAY "Returned to Paragraph 1"
381     *> Execute code multiple times
382     PERFORM 2 TIMES
383     DISPLAY "Repeat"
384     END-PERFORM
385     STOP RUN.
386
387 SubThree.
388     DISPLAY "In Paragraph 3".
389
390 SubTwo.
391     DISPLAY "In Paragraph 2"
392     PERFORM SubThree
393     DISPLAY "Returned to Paragraph 2".
394
395 Subroutines
396

```

open Paragraph

closed Para :

>>>

In Paragraph 1

In Paragraph 2

IP ... 3

Returned to para:-2

Returned ..

Repeat

Repeat .

```

397 You can compile a subroutine separately and then use its code in another program.
398
399 GETSUM.cob
400
401 COMPILE THIS WITH : cobc -m GETSUM.cob
402
403     >>SOURCE FORMAT FREE
404 IDENTIFICATION DIVISION.
405 PROGRAM-ID. GETSUM.
406 DATA DIVISION.
407 *> These variables will be assigned by the calling program
408     LINKAGE SECTION.
409     01 LNum1     PIC 9.
410     01 LNum2     PIC 9.
411     01 LSum      PIC 99.
412 *> Place the variables in the same order in which they are passed
413 PROCEDURE DIVISION USING LNum1, LNum2, LSum.
414 *> We can update the value of sum and when this ends it will update in the calling program
415     COMPUTE LSum = LNum1 + LNum2.
416
417 EXIT PROGRAM.
418
419 coboltut45.cob
420
421 COMPILE THIS WITH : cobc -x coboltut45.cob
422 EXECUTE ./coboltut45
423
424     >>SOURCE FORMAT FREE
425 IDENTIFICATION DIVISION.
426 PROGRAM-ID. coboltut.
427 DATA DIVISION.
428 WORKING-STORAGE SECTION.
429     01 Num1      PIC 9 VALUE 5.
430     01 Num2      PIC 9 VALUE 4.
431     01 Sum1      PIC 99.
432 PROCEDURE DIVISION.
433 *> Call the subroutine in the other file and display the result
434 CALL 'GETSUM' USING Num1, Num2, Sum1.
435 DISPLAY Num1 " + " Num2 " = " Sum1.
436
437 STOP RUN.
438
439 tutorial5.cob
440
441     >>SOURCE FORMAT FREE
442 IDENTIFICATION DIVISION.
443 PROGRAM-ID. tutorial5.
444
445 DATA DIVISION.

```

*Handwritten notes:*

- No Value here!
- 9 5 + 4 → Pass Value
- 5 + 4 = 9

```
446 WORKING-STORAGE SECTION.
447 01 Ind PIC 9(1) VALUE 0.
448
449 PROCEDURE DIVISION.
450 WhileLoop.
451 *> Works like while loop that executes while the index
452 *> is greater than 5
453 PERFORM OutputData WITH TEST AFTER UNTIL Ind > 5
454 *> Jumps to another paragraph
455 GO TO ForLoop.
456
457 OutputData.
458 DISPLAY Ind.
459 ADD 1 TO Ind.
460
461 *> Perform varying works like a for loop where Ind starts
462 *> with a value of 1 defined after FROM and increments by
463 *> 1 defined after BY until the condition is met
464 ForLoop.
465 PERFORM OutputData2 VARYING Ind FROM 1 BY 1 UNTIL Ind=5
466 STOP RUN.
467
468 OutputData2.
469 DISPLAY Ind.
470
471 tutorial20.cob
472
473 >>SOURCE FORMAT FREE
474 IDENTIFICATION DIVISION.
475 PROGRAM-ID. tutorial10.
476 *> We can format data as it is entered
477 *> using edited pictures
478 DATA DIVISION.
479 WORKING-STORAGE SECTION.
480 01 StartNum PIC 9(8)V99 VALUE 00001123.55.
481 *> Replace zeroes with space and add decimal
482 01 NoZero PIC ZZZZZZ9.99.
483 *> No zeroes and commas (Also use *)
484 01 NoZPlusC PIC ZZ,ZZZ,ZZ9.99.
485 *> Convert to dollars (Also use +, -)
486 01 Dollar PIC $,$$$,$$9.99.
487 01 BDay PIC 9(8) VALUE 12211974.
488 *> Insert / (Also use B)
489 01 ADate PIC 99/99/9999.
490
491 PROCEDURE DIVISION.
492 MOVE StartNum TO NoZero
493 DISPLAY NoZero
494 MOVE StartNum TO NoZPlusC
```

```
495 DISPLAY NoZPlusC
496 MOVE StartNum TO Dollar
497 DISPLAY Dollar
498 MOVE BDay TO ADate
499 DISPLAY ADate
500 STOP RUN.
501
502
503
504
505
506
507
508 tutorial21.cob
509
510     >>SOURCE FORMAT FREE
511 IDENTIFICATION DIVISION.
512 PROGRAM-ID. tutorial21.
513 DATA DIVISION.
514 WORKING-STORAGE SECTION.
515 *> Most programming languages use floating point
516 *> calculations which can introduce errors.
517 *> COBOL uses fixed point decimal arithmetic
518 *> and allows you to define how you will round.
519 01 Price PIC 9(4)V99.
520 01 TaxRate PIC V999 VALUE .075.
521 01 FullPrice PIC 9(4)V99.
522
523 PROCEDURE DIVISION.
524 DISPLAY "Enter the Price : " WITH NO ADVANCING
525 ACCEPT Price
526 COMPUTE FullPrice ROUNDED = Price + (Price * TaxRate)
527 DISPLAY "Price + Tax : " FullPrice.
528
529 STOP RUN.
530
531 tutorial22.cob
532
533     >>SOURCE FORMAT FREE
534 IDENTIFICATION DIVISION.
535 PROGRAM-ID. tutorial22.
536 *> COBOL provides many ways to work with strings
537
538 DATA DIVISION.
539 WORKING-STORAGE SECTION.
540 01 SampStr      PIC X(18) VALUE 'eerie beef sneezed'.
541 01 NumChars     PIC 99 VALUE 0.
542 01 NumEs        PIC 99 VALUE 0.
543 01 FName       PIC X(6) VALUE 'Martin'.
```

```
544 01 MName      PIC X(11) VALUE 'Luther King'.
545 01 LName      PIC X(4)  VALUE 'King'.
546 01 FLName     PIC X(11).
547 01 FMLName    PIC X(18).
548 01 SStr1      PIC X(7)  VALUE "The egg".
549 01 SStr2      PIC X(9)  VALUE "is #1 and".
550 01 Dest       PIC X(33) VALUE "is the big chicken".
551 01 Ptr        PIC 9     VALUE 1.
552 01 SStr3      PIC X(3).
553 01 SStr4      PIC X(3).
554
555 PROCEDURE DIVISION.
556 *> Takes string SampStr counts all characters and
557 *> stores the value in NumChars
558 INSPECT SampStr TALLYING NumChars FOR CHARACTERS.
559 DISPLAY "Number of Characters : " NumChars.
560
561 INSPECT SampStr TALLYING NumEs FOR ALL 'e'.
562 DISPLAY "Number of e's : " NumEs.
563 *> Convert to uppercase
564 DISPLAY FUNCTION UPPER-CASE(SampStr).
565 *> Convert to lowercase
566 DISPLAY FUNCTION LOWER-CASE(SampStr).
567
568 *> Join 2 strings with a space between them
569 *> delimited specifies the end of the string
570 *> being size (the whole string) or spaces
571 *> (up to the 1st space) or some other character
572 *> surrounded with quotes like "#" for example
573 STRING FName DELIMITED BY SIZE
574 SPACE
575 LName DELIMITED BY SIZE
576 INTO FLName.
577 DISPLAY FLName.
578
579 *> Get just the 1st word up to the space
580 *> delimited by size gets the whole string
581 *> and join all 3 into a new string
582 *> If the container isn't big enough for the
583 *> string it overflows.
584 STRING FLName DELIMITED BY SPACES
585 SPACE
586 MName DELIMITED BY SIZE
587 SPACE
588 LName DELIMITED BY SIZE
589 INTO FMLName
590 ON OVERFLOW DISPLAY 'Overflowed'.
591 DISPLAY FMLName.
592
```

```
593 *> Grab The egg
594 STRING SStr1 DELIMITED BY SIZE
595 SPACE
596 *> Grab is and the space up to #
597 SStr2 DELIMITED BY "#"
598 *> Insert the above starting at index 1 as defined
599 *> by pointer
600 INTO Dest
601 WITH POINTER Ptr
602 ON OVERFLOW DISPLAY 'Overflowed'.
603 DISPLAY Dest.
604
605 *> Replacing is used to replace strings or characters
606 INSPECT Dest REPLACING ALL 'egg' BY 'dog'.
607 DISPLAY Dest.
608
609 *> Unstring splits a string into multiple strings
610 *> based on a delimiter
611 UNSTRING SStr1 DELIMITED BY SPACE
612 INTO SStr3, SStr4
613 END-UNSTRING.
614 DISPLAY SStr4.
615
616 STOP RUN.
617
618 tutorial6.cob
619
620     >>SOURCE FORMAT FREE
621 IDENTIFICATION DIVISION.
622 PROGRAM-ID. tutorial6.
623
624 ENVIRONMENT DIVISION.
625 INPUT-OUTPUT SECTION.
626 *> Connect the name of the customer file name in this
627 *> code to a file. Records on separate lines
628 FILE-CONTROL.
629     SELECT CustomerFile ASSIGN TO "Customer.dat"
630     ORGANIZATION IS LINE SEQUENTIAL
631     ACCESS IS SEQUENTIAL.
632
633 DATA DIVISION.
634 *> File section describes data in files
635 FILE SECTION.
636 *> FD (File Description) describes the file layout
637 FD CustomerFile.
638 *> Design the customer record
639 01 CustomerData.
640     02 IDNum      PIC 9(8).
641     02 CustName.
```

```
642         03 FirstName    PIC X(15).
643         03 LastName     PIC X(15).
644
645 WORKING-STORAGE SECTION.
646 01 WSCustomer.
647     02 WSIDNum          PIC 9(5).
648     02 WSCustName.
649         03 WSFirstName  PIC X(15).
650         03 WSLastName   PIC X(15).
651
652 PROCEDURE DIVISION.
653 *> COBOL focuses on working with external files or
654 *> databases. Here we will work with sequential files
655 *> which are files you must work with in order. They
656 *> differ from direct access files in that direct access
657 *> files have keys associated with data.
658 *> Field : Individual piece of information (First Name)
659 *> Record : Collection of fields for an individual object
660 *> File : Collection of numerous Records
661
662 *> We process a file by loading one record into memory
663 *> This is called a Record Buffer
664
665 *> Open the file and if it doesn't exist create it
666 *> Add data to all fields, write them to the file
667 *> and close the file
668 OPEN OUTPUT CustomerFile.
669     MOVE 00001 TO IDNum.
670     MOVE 'Doug' TO FirstName.
671     MOVE 'Thomas' TO LastName.
672     WRITE CustomerData
673     END-WRITE.
674     CLOSE CustomerFile.
675 STOP RUN.
676
677 tutorial7.cob
678
679     >>SOURCE FORMAT FREE
680 IDENTIFICATION DIVISION.
681 PROGRAM-ID. tutorial7.
682
683 ENVIRONMENT DIVISION.
684 INPUT-OUTPUT SECTION.
685 FILE-CONTROL.
686     SELECT CustomerFile ASSIGN TO "Customer.dat"
687     ORGANIZATION IS LINE SEQUENTIAL
688     ACCESS IS SEQUENTIAL.
689
690 DATA DIVISION.
```



```
691 FILE SECTION.
692 FD CustomerFile.
693 01 CustomerData.
694     02 IDNum      PIC 9(8).
695     02 CustName.
696         03 FirstName PIC X(15).
697         03 LastName  PIC X(15).
698
699 WORKING-STORAGE SECTION.
700 01 WSCustomer.
701     02 WSIDNum      PIC 9(5).
702     02 WSCustName.
703         03 WSFirstName PIC X(15).
704         03 WSLastName  PIC X(15).
705
706 PROCEDURE DIVISION.
707 *> Extend adds new data to the end of the file
708 OPEN EXTEND CustomerFile.
709     DISPLAY "Customer ID " WITH NO ADVANCING
710     ACCEPT IDNum.
711     DISPLAY "Customer First Name " WITH NO ADVANCING
712     ACCEPT FirstName.
713     DISPLAY "Customer Last Name " WITH NO ADVANCING
714     ACCEPT LastName.
715     WRITE CustomerData
716     END-WRITE.
717     CLOSE CustomerFile.
718 *> Enter customers using ascending keys for later example
719 STOP RUN.
720
721 tutorial8.cob
722
723     >>SOURCE FORMAT FREE
724 IDENTIFICATION DIVISION.
725 PROGRAM-ID. tutorial8.
726
727 ENVIRONMENT DIVISION.
728 INPUT-OUTPUT SECTION.
729 FILE-CONTROL.
730     SELECT CustomerFile ASSIGN TO "Customer.dat"
731     ORGANIZATION IS LINE SEQUENTIAL
732     ACCESS IS SEQUENTIAL.
733
734 DATA DIVISION.
735 FILE SECTION.
736 FD CustomerFile.
737 01 CustomerData.
738     02 IDNum      PIC 9(8).
739     02 CustName.
```

```
740         03 FirstName    PIC X(15).
741         03 LastName     PIC X(15).
742
743 WORKING-STORAGE SECTION.
744 01 WSCustomer.
745     02 WSIDNum          PIC 9(5).
746     02 WSCustName.
747     03 WSFirstName     PIC X(15).
748     03 WSLastName      PIC X(15).
749 *> NEW : Used to react to end of file
750 01 WSEOF               PIC A(1).
751
752 PROCEDURE DIVISION.
753 *> Input is used to read from the file
754 OPEN INPUT CustomerFile.
755     PERFORM UNTIL WSEOF='Y'
756         READ CustomerFile INTO WSCustomer
757         AT END MOVE 'Y' TO WSEOF
758         NOT AT END DISPLAY WSCustomer
759     END-READ
760     END-PERFORM.
761     CLOSE CustomerFile.
762 STOP RUN.
763
764 tutorial9.cob
765
766     >>SOURCE FORMAT FREE
767 IDENTIFICATION DIVISION.
768 PROGRAM-ID. tutorial9.
769 *> Here we'll design and print a customer report
770
771 ENVIRONMENT DIVISION.
772 INPUT-OUTPUT SECTION.
773 FILE-CONTROL.
774     *> Define the file to save the report to
775     SELECT CustomerReport ASSIGN TO "CustReport.rpt"
776     ORGANIZATION IS LINE SEQUENTIAL.
777     *> The file that provides the data
778     SELECT CustomerFile ASSIGN TO "Customer.dat"
779     ORGANIZATION IS LINE SEQUENTIAL.
780
781 DATA DIVISION.
782 FILE SECTION.
783 *> Define FD and custom print line
784 FD CustomerReport.
785 01 PrintLine PIC X(44).
786
787 *> Info on customer data
788 FD CustomerFile.
```

```
789 01 CustomerData.
790     02 IDNum     PIC 9(8).
791     02 CustName.
792         03 FirstName PIC X(15).
793         03 LastName  PIC X(15).
794     88 WSEOF     VALUE HIGH-VALUE.
795
796 WORKING-STORAGE SECTION.
797 *> Break the report up into pieces
798 01 PageHeading.
799     02 FILLER PIC X(13) VALUE "Customer List".
800 01 PageFooting.
801     02 FILLER PIC X(15) VALUE SPACE.
802     02 FILLER PIC X(7) VALUE "Page : ".
803     02 PrnPageNum PIC Z9.
804 *> Column headings for data
805 01 Heads PIC X(36) VALUE "IDNum      FirstName      LastName".
806 *> Customer data to print with spaces defined
807 01 CustomerDetailLine.
808     02 FILLER PIC X VALUE SPACE.
809     02 PrnCustID PIC 9(8).
810     02 FILLER PIC X(4) VALUE SPACE.
811     02 PrnFirstName PIC X(15).
812     02 FILLER PIC XX VALUE SPACE.
813     02 PrnLastName PIC X(15).
814 *> Printed at end of report
815 01 ReportFooting PIC X(13) VALUE "END OF REPORT".
816 *> Tracks number of lines used, when to print footer
817 *> and new heading
818 01 LineCount PIC 99 VALUE ZERO.
819     88 NewPageRequired VALUE 40 THRU 99.
820 *> Track number of pages
821 01 PageCount PIC 99 VALUE ZERO.
822
823 PROCEDURE DIVISION.
824 PrintReport.
825 OPEN INPUT CustomerFile
826 OPEN OUTPUT CustomerReport
827 PERFORM PrintPageHeading
828 *> Read customer file until end
829 READ CustomerFile
830     AT END SET WSEOF TO TRUE
831 END-READ
832 PERFORM PrintReportBody UNTIL WSEOF
833 *> Advancing moves down defined number of lines
834 WRITE PrintLine FROM ReportFooting AFTER ADVANCING 5 LINES
835 CLOSE CustomerFile, CustomerReport
836 STOP RUN.
837
```

```
838 *> Prints heading and tracks page count
839 PrintPageHeading.
840 WRITE PrintLine FROM PageHeading AFTER ADVANCING Page
841 WRITE PrintLine FROM Heads AFTER ADVANCING 5 LINES
842 MOVE 3 TO LineCount
843 ADD 1 TO PageCount.
844
845 *> Handles creating new page logic and printing customer
846 *> data
847 PrintReportBody.
848 IF NewPageRequired
849     MOVE PageCount TO PrnPageNum
850     WRITE PrintLine FROM PageFooting AFTER ADVANCING 5 LINES
851     PERFORM PrintPageHeading
852 END-IF
853 *> Move data to be printed to report
854 MOVE IDNum TO PrnCustID
855 MOVE FirstName TO PrnFirstName
856 MOVE LastName TO PrnLastName
857 WRITE PrintLine FROM CustomerDetailLine AFTER ADVANCING 1 LINE
858 ADD 1 TO LineCount
859 READ CustomerFile
860     AT END SET WSEOF TO TRUE
861 END-READ.
862
863 tutorial10.cob
864
865     >>SOURCE FORMAT FREE
866 IDENTIFICATION DIVISION.
867 PROGRAM-ID. tutorial10.
868 *> This program has a menu system and allows you to
869 *> Add, Update, Delete and Display Customer Data
870 ENVIRONMENT DIVISION.
871 INPUT-OUTPUT SECTION.
872 FILE-CONTROL.
873 *> Select to use a file with keys (Indexed File)
874 *> We will randomly access data vs. sequential
875 *> Define the name associated with the key
876     SELECT CustomerFile ASSIGN TO "customers.txt"
877     ORGANIZATION IS INDEXED
878     ACCESS MODE IS RANDOM
879     RECORD KEY IS IDNum.
880
881 DATA DIVISION.
882 FILE SECTION.
883 *> Model customer data
884 FD CustomerFile.
885     01 CustomerData.
886     02 IDNum PIC 99.
```

```
887         02 FirstName PIC X(15).
888         02 LastName PIC X(15).
889
890 WORKING-STORAGE SECTION.
891     *> Customer menu choice
892     01 Choice PIC 9.
893     *> Tracks whether to exit
894     01 StayOpen PIC X VALUE 'Y'.
895     *> Tracks whether the customer exists
896     01 CustExists PIC X.
897
898 PROCEDURE DIVISION.
899 StartPara.
900     *> To access data randomly you must use I-O mode
901     OPEN I-O CustomerFile.
902     *> Continue execution until StayOpen is N which
903     *> happens if the user enters a number not 1 thru 4
904     PERFORM UNTIL StayOpen='N'
905         DISPLAY " "
906         DISPLAY "CUSTOMER RECORDS"
907         DISPLAY "1 : Add Customer"
908         DISPLAY "2 : Delete Customer"
909         DISPLAY "3 : Update Customer"
910         DISPLAY "4 : Get Customer"
911         DISPLAY "0 : Quit"
912         DISPLAY ": " WITH NO ADVANCING
913         ACCEPT Choice
914         *> Execute different paragraphs based on option
915         EVALUATE Choice
916             WHEN 1 PERFORM AddCust
917             WHEN 2 PERFORM DeleteCust
918             WHEN 3 PERFORM UpdateCust
919             WHEN 4 PERFORM GetCust
920             *> When N we jump out of the loop
921             WHEN OTHER move 'N' TO StayOpen
922         END-EVALUATE
923
924     END-PERFORM.
925     *> Close the file and stop execution
926     CLOSE CustomerFile
927     STOP RUN.
928
929 AddCust.
930     DISPLAY " ".
931     DISPLAY "Enter ID : " WITH NO ADVANCING.
932     ACCEPT IDNum.
933     DISPLAY "Enter First Name : " WITH NO ADVANCING.
934     ACCEPT FirstName.
935     DISPLAY "Enter Last Name : " WITH NO ADVANCING.
```

```
936     ACCEPT LastName.
937     DISPLAY " ".
938     *> Write customer data or display error if ID taken
939     WRITE CustomerData
940         INVALID KEY DISPLAY "ID Taken"
941     END-WRITE.
942
943
944 DeleteCust.
945     DISPLAY " ".
946     DISPLAY "Enter Customer ID to Delete : " WITH NO ADVANCING.
947     ACCEPT IDNum.
948     *> Delete customer based on ID
949     DELETE CustomerFile
950     INVALID KEY DISPLAY "Key Doesn't Exist"
951     END-DELETE.
952
953 UpdateCust.
954     MOVE 'Y' TO CustExists.
955     DISPLAY " ".
956     DISPLAY "Enter ID to Update : " WITH NO ADVANCING.
957     ACCEPT IDNum.
958     *> Read customer or mark N if doesn't exist
959     READ CustomerFile
960         INVALID KEY MOVE 'N' TO CustExists
961     END-READ.
962     *> Display error because ID doesn't exist
963     IF CustExists='N'
964         DISPLAY "Customer Doesn't Exist"
965     ELSE
966         DISPLAY "Enter the New First Name : " WITH NO ADVANCING
967         ACCEPT FirstName
968         DISPLAY "Enter the New Last Name : " WITH NO ADVANCING
969         ACCEPT LastName
970     END-IF.
971     *> Update record for matching ID
972     REWRITE CustomerData
973     INVALID KEY DISPLAY "Customer Not Updated"
974     END-REWRITE.
975
976
977 GetCust.
978     *> Assume customer exists
979     MOVE 'Y' TO CustExists.
980     DISPLAY " ".
981     DISPLAY "Enter Customer ID to Find : " WITH NO ADVANCING.
982     ACCEPT IDNum.
983     *> Mark N if customer ID doesn't exist
984     READ CustomerFile
```

```

985         INVALID KEY MOVE 'N' TO CustExists
986     END-READ.
987     *> Display error
988     IF CustExists='N'
989         DISPLAY "Customer Doesn't Exist"
990     ELSE
991         DISPLAY "ID : " IDNum
992         DISPLAY "First Name : " FirstName
993         DISPLAY "Last Name : " LastName
994     END-IF.
995
996 tutorial11.cob
997
998     >>SOURCE FORMAT FREE
999     *> Tables contain multiple data items like arrays
1000     *> Indexes are called subscripts in COBOL and start
1001     *> at subscript 1 instead of 0. You define the
1002     *> containing data with a record description.
1003     IDENTIFICATION DIVISION.
1004     PROGRAM-ID. tutorial11.
1005     DATA DIVISION.
1006
1007     WORKING-STORAGE SECTION.
1008     *> Declare a 1 dimensional table
1009     01 Table1.
1010         02 Friend PIC X(15) OCCURS 4 TIMES.
1011
1012     *> Declare a multidimensional table
1013     01 CustTable.
1014         02 CustName OCCURS 5 TIMES.
1015             03 FName PIC X(15).
1016             03 LName PIC X(15).
1017
1018     *> Declare a table with indexes
1019     01 OrderTable.
1020         02 Product OCCURS 2 TIMES INDEXED BY I.
1021             03 ProdName PIC X(10).
1022             03 ProdSize OCCURS 3 TIMES INDEXED BY J.
1023             04 SizeType PIC A.
1024
1025     PROCEDURE DIVISION.
1026     *> Fill 1D table with data and output
1027     MOVE 'Joy' TO Friend(1).
1028     MOVE 'Willow' TO Friend(2).
1029     MOVE 'Ivy' TO Friend(3).
1030     DISPLAY Friend(1).
1031     DISPLAY Table1.
1032
1033     *> Fill MD table with data and output

```

```
1034     MOVE 'Paul' TO FName(1).
1035     MOVE 'Smith' TO LName(1).
1036     MOVE 'Sally' TO FName(2).
1037     MOVE 'Smith' TO LName(2).
1038     DISPLAY CustName(1).
1039     DISPLAY CustTable.
1040
1041     *> Working with indexed tables
1042     *> Set index value with SET
1043     SET I J TO 1.
1044     MOVE 'Blue Shirt' TO Product(I).
1045     MOVE 'S' TO ProdSize(I,J).
1046     *> Increment with SET
1047     SET J UP BY 1
1048     MOVE 'M' TO ProdSize(I,J).
1049     *> Decrement with SET
1050     SET J DOWN BY 1
1051     *> Fill with product information
1052     MOVE 'Blue ShirtSMLRed Shirt SML' TO OrderTable.
1053     *> Increment I as we get products
1054     PERFORM GetProduct VARYING I FROM 1 BY 1 UNTIL I>2.
1055     GO TO LookUp.
1056
1057 GetProduct.
1058     DISPLAY Product(I).
1059     *> Get associated product sizes
1060     PERFORM GetSizes VARYING J FROM 1 BY 1 UNTIL J>3.
1061
1062 GetSizes.
1063     DISPLAY ProdSize(I,J).
1064
1065 LookUp.
1066     SET I TO 1.
1067     *> Search will look for supplied value or
1068     *> output Not Found
1069     SEARCH Product
1070         AT END DISPLAY 'Product Not Found'
1071         WHEN ProdName(I) = 'Red Shirt'
1072             DISPLAY 'Red Shirt Found'
1073     END-SEARCH.
1074
1075 STOP RUN.
1076
1077 tutorial12.cob
1078
1079     >>SOURCE FORMAT FREE
1080 IDENTIFICATION DIVISION.
1081 PROGRAM-ID. tutorial12.
1082 DATA DIVISION.
```



```
1083 *> Here I'll show you how to prefill tables
1084 *> with the redefines clause
1085 WORKING-STORAGE SECTION.
1086 01 ProdTable.
1087     02 ProdData.
1088         *> Because we don't need to identify labels
1089         *> for the data in this string we use filler
1090         03 FILLER PIC X(8) VALUE "Red SML".
1091         03 FILLER PIC X(8) VALUE "Blue SML".
1092         03 FILLER PIC X(8) VALUE "GreenSML".
1093     02 FILLER REDEFINES ProdData.
1094         03 Shirt OCCURS 3 TIMES.
1095             04 ProdName PIC X(5).
1096             04 ProdSizes PIC A OCCURS 3 TIMES.
1097 *> If data is stored as a string but you want to use it
1098 *> as a numeric use redefines to do so automatically
1099 01 ChangeMe.
1100     02 TextNum PIC X(6).
1101     02 FloatNum REDEFINES TextNum PIC 9(4)V99.
1102
1103 *> Accept a string, convert it into a useable float
1104 *> and an edited number
1105 01 StrNum PIC X(7).
1106 01 SplitNum.
1107     02 WNum PIC 9(4) VALUE ZERO.
1108     02 FNum PIC 99 VALUE ZERO.
1109 01 FNum REDEFINES SplitNum PIC 9999V99.
1110 01 DollarNum PIC $$,$$9.99.
1111
1112 PROCEDURE DIVISION.
1113 DISPLAY Shirt(1).
1114 MOVE '123456' TO TextNum.
1115 DISPLAY FloatNum.
1116
1117 *> Divide the string into 2 strings based on delimiter
1118 *> and then edit the output
1119 DISPLAY "Enter a Float : " WITH NO ADVANCING
1120 ACCEPT StrNum
1121 UNSTRING StrNum
1122     DELIMITED BY "." OR ALL SPACES
1123     INTO WNum, FNum
1124 MOVE FNum TO DollarNum
1125 DISPLAY DollarNum
1126
1127 STOP RUN.
1128
1129 tutorial13.cob
1130
1131 >>SOURCE FORMAT FREE
```

```
1132 IDENTIFICATION DIVISION.
1133 PROGRAM-ID. tutorial13.
1134 *> This program sorts a file by ID
1135 *> Sample file Data saved in student.dat
1136 *>5Derek
1137 *>4Paul
1138 *>3Sue
1139 *>2Sally
1140 ENVIRONMENT DIVISION.
1141 INPUT-OUTPUT SECTION.
1142 FILE-CONTROL.
1143 *> Line Sequential puts data on separate lines
1144     SELECT WorkFile ASSIGN TO 'work.tmp'.
1145     SELECT OrgFile ASSIGN TO 'student.dat'
1146         ORGANIZATION IS LINE SEQUENTIAL.
1147     SELECT SortedFile ASSIGN TO 'student2.dat'
1148         ORGANIZATION IS LINE SEQUENTIAL.
1149 DATA DIVISION.
1150 FILE SECTION.
1151 FD OrgFile.
1152 01 StudData.
1153     02 IDNum      PIC 9.
1154     02 StudName   PIC X(10).
1155 *> SD (Sort File Description) describes layout
1156 *> for sorted files
1157 SD WorkFile.
1158 01 WStudData.
1159     02 WIDNum     PIC 9.
1160     02 WStudName  PIC X(10).
1161 FD SortedFile.
1162 01 SStudData.
1163     02 SIDNum     PIC 9.
1164     02 SStudName  PIC X(10).
1165
1166 PROCEDURE DIVISION.
1167 SORT WorkFile ON ASCENDING KEY SIDNum
1168     USING OrgFile
1169     GIVING SortedFile.
1170
1171 STOP RUN.
1172
1173 tutorial14.cob
1174
1175     >>SOURCE FORMAT FREE
1176 IDENTIFICATION DIVISION.
1177 PROGRAM-ID. tutorial14.
1178 *> This merges files that contain data structured
1179 *> the same
1180 *> Sample file Data saved in student.dat
```

```
1181 *>5Derek
1182 *>4Paul
1183 *>3Sue
1184 *>2Sally
1185 *> Sample data from student3.dat
1186 *>1Sam
1187 *>6Mark
1188 ENVIRONMENT DIVISION.
1189 INPUT-OUTPUT SECTION.
1190 FILE-CONTROL.
1191 *> Line Sequential puts data on separate lines
1192     SELECT WorkFile ASSIGN TO 'work.tmp'.
1193     SELECT File1 ASSIGN TO 'student.dat'
1194         ORGANIZATION IS LINE SEQUENTIAL.
1195     SELECT File2 ASSIGN TO 'student3.dat'
1196         ORGANIZATION IS LINE SEQUENTIAL.
1197     SELECT NewFile ASSIGN TO 'student4.dat'
1198         ORGANIZATION IS LINE SEQUENTIAL.
1199 DATA DIVISION.
1200 FILE SECTION.
1201 FD File1.
1202 01 StudData.
1203     02 IDNum      PIC 9.
1204     02 StudName   PIC X(10).
1205 FD File2.
1206 01 StudData2.
1207     02 IDNum2     PIC 9.
1208     02 StudName2  PIC X(10).
1209 SD WorkFile.
1210 01 WStudData.
1211     02 WIDNum     PIC 9.
1212     02 WStudName PIC X(10).
1213 FD NewFile.
1214 01 NStudData.
1215     02 NIDNum     PIC 9.
1216     02 NStudName PIC X(10).
1217
1218 PROCEDURE DIVISION.
1219 MERGE WorkFile ON ASCENDING KEY NIDNum
1220     USING File1, File2
1221     GIVING NewFile.
1222
1223 STOP RUN.
1224
1225 Object Oriented COBOL
1226
1227     >>SOURCE FORMAT FREE
1228 IDENTIFICATION DIVISION.
1229 PROGRAM-ID. tutorial15.
```

```
1230 *> Object Oriented COBOL uses classes to describe
1231 *> data and methods to work with that data
1232
1233 REPOSITORY.
1234     CLASS ToDoCls AS "todolist".
1235
1236 DATA DIVISION.
1237 WORKING-STORAGE SECTION.
1238 01 WorkToDo USAGE OBJECT REFERENCE ToDoCls.
1239 01 ToDoToAdd PIC X(20).
1240     88 EndOfInput VALUE SPACES.
1241 01 DescToAdd PIC X(50).
1242
1243 PROCEDURE DIVISION.
1244 INVOKE ToDoCls "new" USING BY CONTENT "Work ToDo"
1245     RETURNING WorkToDo
1246
1247 PERFORM AddToToDoList WITH TEST AFTER UNTIL EndOfInput
1248 INVOKE WorkToDo "PrintToDos"
1249
1250 STOP RUN.
1251
1252 AddToToDoList.
1253 DISPLAY "Enter To Do : " WITH NO ADVANCING
1254 ACCEPT ToDoToAdd
1255 DISPLAY "Enter Description : " WITH NO ADVANCING
1256 ACCEPT DescToAdd
1257 INVOKE WorkToDo "AddItemToDo"
1258     USING BY CONTENT ToDoToAdd, DescToAdd.
```

## Leave a Reply

Your email address will not be published.

Comment

Comment

Name

Email

Website

Submit Comment

Search

Social Networks

Facebook

YouTube

Twitter

LinkedIn

Buy me a Cup of Coffee

"Donations help me to keep the site running. One dollar is greatly appreciated." - (Pay Pal Secured)

Donate



My Facebook Page

Like Share 6K people like this. [Sign Up](#) to see what your friends

Archives

- [June 2020](#)
- [May 2020](#)
- [April 2020](#)
- [March 2020](#)
- [February 2020](#)
- [January 2020](#)
- [December 2019](#)
- [November 2019](#)
- [October 2019](#)
- [August 2019](#)
- [July 2019](#)
- [June 2019](#)
- [May 2019](#)
- [April 2019](#)
- [March 2019](#)
- [February 2019](#)
- [January 2019](#)

- [December 2018](#)
- [October 2018](#)
- [September 2018](#)
- [August 2018](#)
- [July 2018](#)
- [June 2018](#)
- [May 2018](#)
- [April 2018](#)
- [March 2018](#)
- [February 2018](#)
- [January 2018](#)
- [December 2017](#)
- [November 2017](#)
- [October 2017](#)
- [September 2017](#)
- [August 2017](#)
- [July 2017](#)
- [June 2017](#)
- [May 2017](#)
- [April 2017](#)
- [March 2017](#)
- [February 2017](#)
- [January 2017](#)
- [December 2016](#)
- [November 2016](#)
- [October 2016](#)
- [September 2016](#)
- [August 2016](#)
- [July 2016](#)
- [June 2016](#)
- [May 2016](#)
- [April 2016](#)
- [March 2016](#)
- [February 2016](#)
- [January 2016](#)
- [December 2015](#)
- [November 2015](#)
- [October 2015](#)
- [September 2015](#)
- [August 2015](#)
- [July 2015](#)

- [June 2015](#)
- [May 2015](#)
- [April 2015](#)
- [March 2015](#)
- [February 2015](#)
- [January 2015](#)
- [December 2014](#)
- [November 2014](#)
- [October 2014](#)
- [September 2014](#)
- [August 2014](#)
- [July 2014](#)
- [June 2014](#)
- [May 2014](#)
- [April 2014](#)
- [March 2014](#)
- [February 2014](#)
- [January 2014](#)
- [December 2013](#)
- [November 2013](#)
- [October 2013](#)
- [September 2013](#)
- [August 2013](#)
- [July 2013](#)
- [June 2013](#)
- [May 2013](#)
- [April 2013](#)
- [March 2013](#)
- [February 2013](#)
- [January 2013](#)
- [December 2012](#)
- [November 2012](#)
- [October 2012](#)
- [September 2012](#)
- [August 2012](#)
- [July 2012](#)
- [June 2012](#)
- [May 2012](#)
- [April 2012](#)
- [March 2012](#)
- [February 2012](#)

- [January 2012](#)
- [December 2011](#)
- [November 2011](#)
- [October 2011](#)
- [September 2011](#)
- [August 2011](#)
- [July 2011](#)
- [June 2011](#)
- [May 2011](#)
- [April 2011](#)
- [March 2011](#)
- [February 2011](#)
- [January 2011](#)
- [December 2010](#)
- [November 2010](#)
- [October 2010](#)
- [September 2010](#)
- [August 2010](#)
- [July 2010](#)
- [June 2010](#)
- [May 2010](#)
- [April 2010](#)
- [March 2010](#)
- [February 2010](#)
- [January 2010](#)
- [December 2009](#)

Powered by [WordPress](#) | Designed by [Elegant Themes](#)  
[About the Author](#) [Google+](#)