

COBOL STANDARDS

The following is a list of requirements and standards to be followed in this course. They are the basic, or default standards, and, as such, are subject to specific requirements of each assignment. Not adhering to the submission, documentation, and programming requirements and standards are grounds for loss of points on the assignment.

A. REQUIREMENTS FOR SUBMITTING PROGRAMS AND DOCUMENTATION

1. All assignment packages will be submitted in a folder with both left and right inside pockets. **IMPORTANT:** The date/time the program package is received by the instructor or by the MIS Department staff will be the official date/time of submission, not the date of compilation or execution.

The folder will be arranged as follows:

- a. ON OUTER COVER: Name, course, section number, and assignment number.
- b. INSIDE LEFT POCKET: Required computer-generated output containing, at least, the program listing(s) (e.g., .LIS printouts from the compilation), and the program output (e.g., .RPT printouts) and your disk, when required, with the same data on the disk as on the outer cover.

The computer-generated output from Micro Focus Personal COBOL shall have the date and time on each page of the listing, and each page of the listing and the output shall be either initialed or signed by you.

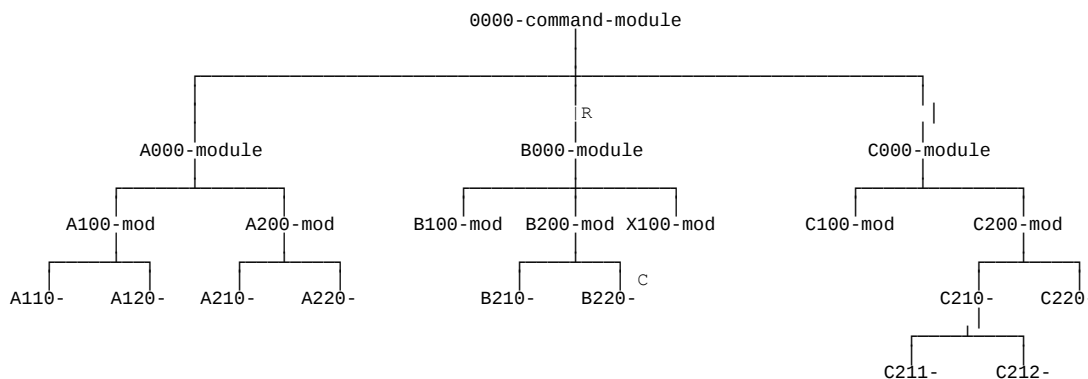
The computer-generated output shall, when possible, be on continuous sheets. Any separation of sheets will require that your signature or initials be on each sheet submitted.

- c. INSIDE RIGHT POCKET: External documentation which shall consist of hierarchy chart(s) and macro flow chart(s) stapled together, and other documentation as required by the assignment. Unless specifically excluded in the assignment or in class, all the documentation is required. Your name, MIS 102, section number and assignment number must be on each page.
2. The hierarchy and flow charts shall be on 8-1/2 x 11 inch paper (reduce if necessary). They **MUST** be produced by computer software. You may, however, use any software you wish. The forms of the hierarchy charts and of the flow charts are discussed below.

3. Other external documentation shall be typed or be computer-printed (e.g., from a word processor).
4. **NOTE:** A structured program can easily be depicted in a hierarchy chart. The purpose of the hierarchy chart is to show the organization of the program. It is not the detailed logic the diagram is concerned with, but the control and the approach to the solution that is addressed. Other graphical tools, such as flow charts and Warnier-Orr diagrams, are better suited to show the specifics of a program.
5. **RECOMMENDATIONS:** Aesthetically arrange the components of your chart. Plan the chart and create an initial layout before finalizing it. Use the connector symbols sparingly and for vertical continuations only. Keep the boxes uniform in size and the lines straight and continuous. Boxes in a hierarchy chart represent not only paragraphs or modules of the program; they also identify the major functions or forms of action.

B. CHARTS IN GENERAL

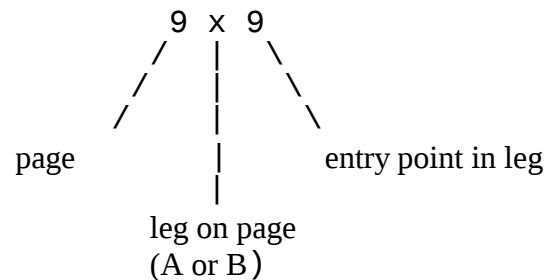
1. The following method of module identification will be used in your program and, therefore, your hierarchy and flowcharts as well. This scheme allows one to follow the flow of control and logic both forwards and backwards. Note that each subordinate level paragraph number identifies its calling module by letter and by number. If more levels are required, begin with a 00000-command-module (five zeros).



If more levels are required, begin with a 00000-command-module (five zeros).

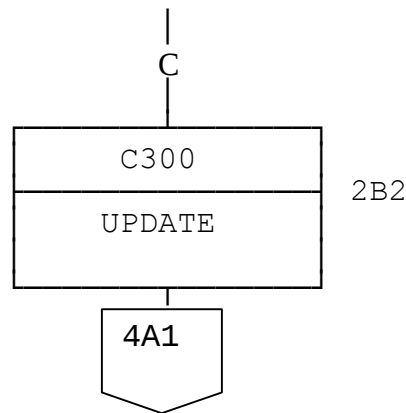
NOTE: Module identifiers and module names appear as shown in 8. below.

2. A paragraph called by more than one control module, e.g., a read-data-module, will be prefixed by an X for an input process (as in X100-READ-DEAN-FILE) or a Y for an output process (e.g., X200-TOP-OF-PAGE-ROUTINE). If there are more than one of either type, they will be prefixed by X200-input-module-2, X300-input-module-3, etc., or Y200-output-module-3, etc.
3. A module that is **Repeated** (e.g., PERFORM UNTIL...) is designated by an **R** in or by the connector of your hierarchy chart as shown above with the B000-module.
4. A module that is executed subject to a condition (e.g., an IF...THEN PERFORM module) has a **C** in or by the connector.
5. All charts will
 - a. be symmetrical
 - b. use standard page numbering (see 6 below)
 - c. be consecutively numbered, e.g., "1 of 6", then "2 of 6".
 - d. be contained on 8 1/2" x 11" bond paper.
 - e. have standard page labels as shown below.
6. Page references are in the following format.



Consequently, '2B1' is on (a) the second page, (b) leg B (right hand column), and the first entry point from the top.

7. For hierarchy charts which cannot be completed on one page, use connectors containing the page-leg-entry point presented in 6. above. (See example below.)
8. For module entries in hierarchy charts and flow charts which have more detailed, explanatory charts, the page-leg-entry point will be entered next to the module. (See example below.)



9. Hierarchy Charts, Macro Flow Charts, and the PROCEDURE DIVISION are related as follows:
- Paragraph names and numbers are consistent
 - Paragraphs are in the same ascending numerical sequence
 - Predefined Function symbols in a Macro Flow Chart match top-to-bottom, left-to-right sequence of the Hierarchy Chart. Both match the call (PERFORM) sequence of the PROCEDURE DIVISION.
10. A program, BILLING.CBL, which approximately represents the desired programming style for this class, is on my web page. This file may be copied to your account and/or downloaded to your PC. A hierarchy chart and a flow chart for that program will be provided to you in the next few weeks.

C. OUTPUT SPECIFICATIONS

- Multiple pages must be numbered.
- Page number and date on title line.
- Output is symmetrical vertically and horizontally.
- Start printing each separate report at beginning of new page.
- All numeric fields zero suppressed to units digit (e.g., PIC Z,ZZ9, PIC ZZZ,ZZ9.99). That is, the output of numeric variables will zero suppress up to the last digit to the left of the decimal point.
- Detail lines double spaced unless otherwise specified.

7. No more than 20 printed ***detail*** lines per page unless otherwise specified.
8. All reports restricted to 80 print positions.
9. Commas used in monetary fields.
10. Double column heading lines symmetrically aligned, e.g.,

ITEM
NUMBER

11. Both vertical and horizontal printing must be centered unless otherwise directed.
12. There must be three blank lines between the report title and report or column headings unless otherwise directed.
13. There must be one blank line between column headings and the first detail, data line.
14. There must be at least one blank line between the last detail line and summary total(s).
15. All computed statistics (e.g., means or percentages) will be printed out with one decimal place, unless otherwise specified.
16. Floating dollar signs are to be avoided unless otherwise directed.
17. Specify completely the fields of the edited, numeric output which is to be printed, e.g., PIC \$ZZZ,ZZ9.99 but not PIC \$Z(3),ZZ9.99. It's easier to see how your output will look. It's your judgment on how to deal with alpha-numeric fields.
18. It is most helpful and a real time-saver to use a printer layout sheet to design your printed output from which you can specify the appropriate DATA DIVISION entries.

D. PROGRAMMING STANDARDS

1. Program readability, clarity, maintainability, and structure are paramount and essential. Make liberal use of blank lines, strings of asterisks, and so on along with liberal use of comments.
2. In the PROCEDURE DIVISION, use the paragraph numbering scheme presented above.
3. All user-defined data names in the WORKING-STORAGE SECTION, will be prefixed by WSn where n is a sequence number of the working storage entries. Examples are

01 WS1-INPUT-RECORD-AREA.
01 WS2-REPORT-HEADER.

Subordinate data items, e.g., at the 05 level, will each be prefixed by the same prefix as that at their respective 01 level.

4. User-defined data-names in an FD (file definition) entry shall be prefixed by the initials of the user-defined file name (e.g., TF- for TRANSACTION-FILE).
5. The IDENTIFICATION DIVISION must include program remarks bracketed by asterisks. These remarks must include the following:
 - a. A short overview of the purpose of the program
 - b. Frequency of program operation (e.g., monthly), if applicable.
 - c. Course number and section number
 - d. Program or assignment number
 - e. Name of instructor
6. A "/" placed in position 7 of the line preceding the DATA DIVISION and the PROCEDURE DIVISION so that these divisions begin at the top of a new page of the listing.
7. Use the page eject symbol ("/" in column 7) to prevent group items in the DATA DIVISION and modules (paragraphs) in the PROCEDURE DIVISION from being separated by a page break.
8. All FD entries in the FILE SECTION will be separated by at least one blank line.
9. All 01 entries in the WORKING-STORAGE SECTION will be separated by at least one blank line.
10. Spacing between DIVISIONS, SECTIONS, LEVELS, and paragraphs.

11. PIC, VALUE, and USAGE clauses aligned.
12. Throughout the DATA DIVISION, use the 01 level number **primarily** for data group items while defining elementary data items at the subordinate levels (e.g. 05).
13. Data-name values are to be initialized in WORKING-STORAGE DIVISION rather than in the PROCEDURE DIVISION.
14. 88 levels used for ***all*** coded items where appropriate and for data validation (e.g., 88 VALID-VALUES). The use of "88" levels, with clear and representative condition names, is encouraged. The use of condition names generally improves the program readability.
15. 05 levels defined as ***much as possible*** in WORKING-STORAGE SECTION rather than FILE SECTION.
16. USAGE COMP will be used for all arithmetic fields that are used several times (e.g., accumulators, counters, running totals, and intermediate storage values). They will also be Signed (e.g., PIC S9(5)V99 USAGE COMP). Providing for these items to be plus or minus can assist in debugging and finding errors in that a negative value for a gross sales amount, for example, would suggest a serious problem. Further, in order to know if a company has had a loss, the accumulated value for PROFIT must provide for a minus value.
17. You must provide comprehensive internal documentation in IDENTIFICATION DIVISION and at the beginning of paragraphs.
18. It is recommended that the input area(s) and the output area(s) each be identified by a comments entry, e.g.,

```
*****  
* INPUT AREA *  
*****
```

19. It is recommended that subscript-data-names be suffixed with -SUB and index-data-names be suffixed with -IX or -IDX.
20. Switches and flags have unique functions. Consequently, each switch- or flag-data-name (NOT the associated condition-name) have the suffix -FLAG or -SW.
21. Align the TOs, particularly with multiple, successive MOVE statements.
22. For printed output, all WRITE statements will use the ADVANCING clause with few exceptions.

- 23. Limited use of COMPUTE statements.
- 24. **NO** GO TOs except as directed. If you **must** use a GO TO, a written justification stipulating why no other coding structure will work is required to avoid loss of points. The reasoning must be acceptable.
- 25. Limited use of arithmetic symbols (e.g., "=", "<").
- 26. Clauses and subordinate data and procedure entries be indented by four (4) columns, e.g.,

***SELECT internal-file-name
ASSIGN TO external-file-name.***

***READ internal-file-name
INTO input-work-area-data-name
AT END statement.***

- 27. A data-name must not be abbreviated to an extent that it obscures the meaning of the data which it represents. Use all 30 characters, if necessary, to arrive at a clear and meaningful data-name.

Acceptable abbreviations include EOF (end of file), IDX (index), MAX (maximum), MIN (minimum), TRANS (transaction), and SOC-SEC-NUM (social security number). Unless the name is approaching 30 characters, it should be spelled out. When in doubt, ask your me.

- 28. Independent but qualitatively similar data items, e.g., totals, counters, should each be grouped in its own group at the 01 level. For example,

01 WS22-TOTALS.
05 WS22-DEPARTMENT-TOTAL.
05 WS22-STORE-TOTAL.
05 WS22-REGION-TOTAL.
05 WS22-COMPANY-TOTAL.

This eliminates the need to use 77 levels, which is not allowed.

- 29. It is important to include internal documentation in the program for each module in the PROCEDURE DIVISION. In particular, if the module performs a complex routine, in-line documentation describing it should appear above the module heading surrounded by asterisks.

30. No paragraph should exceed a single page in length. The reader should not need to turn a page to continue with a procedure. It is highly desirable that this be applied to grouped (01 level) items in the DATA DIVISION.
31. If you are unsure of the output of your program, calculate the answer(s) by hand from a sample of a data file. In some assignments, you may be able to verify your answers against previously produced reports. Check for consistencies throughout the output.
32. **ABSOLUTELY NO** in-line PERFORMs unless specifically authorized by your instructor.
33. For input/output, always use **READ...INTO...** and **WRITE...FROM....**
34. All program statements will be written in capital letters.
35. All main-line (i.e., driver) paragraphs will consist only of PERFORM statements and a STOP RUN, **REGARDLESS** of the examples in any text. For example,

 PERFORM initial-process-para.
 PERFORM file-processing-para UNTIL condition.
 PERFORM summary-report-para.
 STOP RUN.
36. **NO** use of CORRESPONDING.
36. IF statements and complete nested IF statements shall terminate with a period or scope terminator. Such statements shall be written, when appropriate, with the ELSE phrase when appropriate) in the same column as its associated IF.

The IF statements shall take the indentation formats:

- | | | | |
|----|---|----|--|
| a. | IF condition
statement(s)-1
ELSE
statement(s)-2. | b. | IF condition-1
statement(s)1
IF condition-2
Statement(s)-2
ELSE
Statement(s)-3
ELSE
Statement(s)-4. |
|----|---|----|--|