Sample Code

Decision

Don't place decision to execute in the subroutine

```
perform get_start_end_date.

form get_start_end_date.
  check not ( s_workdt[] is initial ).
  ...more processing...
endform.
```

instead do:

```
If not ( s_workdt[] is initial ).
  perform get_start_end_date.
endif.
```

Case Statement

Use CASE statement instead of IF...ELSEIF when possible (It is only possible in equality tests)

```
if t_bsid-maber = c_alumni_debit.
  ...do something...
elseif t_bsid-maber = c_medical_debit.
  ...do something...
else.
  ...do something else...
endif.
```

instead do:

```
case t_bsid-maber.
  when c_alumni_debit.
    ...do something...
  when c_medical_debit.
    ...do something...
  when others.
  ...do something else...
endcase.
```

Nested If

Test most likely to fail first (specific to general)

```
if t_covp-vrgng = 'COIN'.         " a document created in FI
  if t_covp-objnr = 'PR00003078'. " a specific cost object (wbs element)
    ...do something...
  endif.
endif.
```

instead do:

```
if t_covp-objnr = 'PR00003078'. " a specific cost object (wbs element 6453100)
  if t_covp-vrgng = 'COIN'.     " a document created in FI
    ...do something...
  endif.
```

```
endif.
```

And

Test most likely to fail first (specific to general)

```
if t_covp-vrgng = 'COIN' and t_covp-objnr = 'PR00003078'. "FI doc and wbs 6453100
  ...do something...
endif.

instead do:

if t_covp-objnr = 'PR00003078' and t_covp-vrgng = 'COIN'. "6453100 and FI
  ...do something...
endif.
```

OR

Test most likely to succeed first (general to specific)

```
if t_covp-objnr = 'PR00003078' or t_covp-vrgng = 'COIN'. "6453100 or FI
  ...do something...
endif.

instead do:

if t_covp-vrgng = 'COIN' or t_covp-objnr = 'PR00003078'. "FI or 6453100
  ...do something...
endif.
```

Sort and READ TABLE

SORT and READ TABLE t_tab WITH KEY ... BINARY SEARCH when possible especially against non-buffered table (Data Dictionary -> Technical Info)

```
loop at t_bseg.
  select single name1 from kna1 into w_name
    where kunnr = t_bseg-kunnr.
endloop.

instead do:

data: begin of t_kna1 occurs 0,
  kunnr like kna1-kunnr,
  name1 like kna1-name1,
end of t_kna1.

select kunnr name1 from kna1 into table t_kna1.

sort t_kna1 by kunnr.

loop at t_bseg.
  read table t_kna1 with key kunnr = t_bseg-kunnr binary search.
  if sy-subrc = 0.
    w_name = t_kna1-name1.
  endif.
endloop.
```

Read TABLE and Modify

```
    read table t_vbap
      with key ps_psp_pnr = t_tab-pspnr binary search.
    t_vbap-kunnr = w_kunnr.
    modify table t_vbap index sy-tabix.
```

Modifing internal tables

When you need to modify an internal table use field symbols instead of reading table into header and modifying.

```
  loop at t_vbak.
    select single kunnr into t_vbak-payer
        from vbpa
        where vbeln = t_vbak-vbeln
        and posnr = c_head_data       " 000000 item indicates header
        and parvw = c_payer.          " RG is payer
    if sy-subrc <> 0.
      message a056 with 'VBPA' t_vbak-vbeln.
    endif.
    modify t_vbak.
  endloop.  " loop through contract headers and find payers
```

instead do:

```
field-symbols <f_vbak> like t_vbak.

  loop at t_vbak assigning <f_vbak>.
    select single kunnr into <f_vbak>-payer
        from vbpa
        where vbeln = <f_vbak>-vbeln
        and posnr = c_head_data       " 000000 item indicates header
        and parvw = c_payer.          " RG is payer
    if sy-subrc <> 0.
      message a056 with 'VBPA' <f_vbak>-vbeln.
    endif.
  endloop.  " loop through contract headers and find payers
```

COLLECT (building an internal table with unique key)

When you need to build a table of unique row entries and you aren't using select into COLLECT is a useful command. It will summarize, treating all character fields as the unique key for the internal table.

```
  if not ( iw_current is initial ).
    <f_vbap>-current = <f_vbap>-current + iw_current.
    read table t_vbak key vbeln = <f_vbap>-vbeln binary search.
*  by definition header must exist if line item exists! Hence no return code check
    t_expenses-prime_contract = t_vbak-bstnk.   " primary contract
    t_expenses-posid = <f_prps>-posid.          " billable wbs
    t_expenses-matnr = w_matnr.                 " material
    t_expenses-kbetr = iw_current.              " current expenses
    collect t_expenses.            " current by wbs/contract/material
  endif.      " are there dollars charged to this contract?
```

Sort by fields

SORT tables BY fields

```
sort t_kna1.
```

Instead

```
sort t_kna1 by kunnr.
```

Select Single

```
select name1 from kna1 into w_name up to 1 rows
  where kunnr = t_bseg-kunnr.
```

instead do:

```
select single name1 from kna1 into w_name
  where kunnr = t_bseg-kunnr.
```

or:

```
select single name1 ktokk from lfa1 into (w_vname1, w_ktokk)
  where kunnr = t_bseg-kunnr.
```

Select fields into an internal table

```
data: begin of t_kna1 occurs 0,
  kunnr like kna1-kunnr,
  name1 like kna1-name1,
end of t_kna1.

select kunnr name1 from kna1 into table t_kna1. "replaces contents of t_kna1
```

Appending lines to internal table

```
data: begin of t_proj occurs 0,
        pspnr like proj-pspnr,         " internal project def. number
        objnr like proj-objnr,         " PD + internal proj number
      end of t_proj.

data: begin of t_prps occurs 0,
        objnr like prps-objnr,         " WBS element object # PR+int rsn
        pspnr like prps-pspnr,         " WBS element internal rsn
        posid like prps-posid,         " WBS element external number
        stufe like prps-stufe,         " hierarchy level (1 is top)
        fakkz like prps-fakkz,         " billing indicator (X=yes)
        zbillt like prps-zbillt,       " billing type (01=cost reimburs)
        psphi like prps-psphi,         " number of project def. internal
        zend   like prps-zend,         " expiration date
        ztermcd like prps-ztermcd,     " term code
        up    like prhi-up,            " parent WBS element internal
        budget like rpsco-wtp00,       " auth. total on WBS element
        actual like rpsco-wtp00,       " total expenditure on WBS
        revenue like rpsco-wtp00,      " total revenue on WBS
        begbal like rpsco-wtp00,       " begin balance on WBS
      end of t_prps.

  refresh t_prps.
  clear t_prps.

  loop at t_proj.
    select objnr pspnr posid stufe fakkz zbillt psphi zend ztermcd
           from prps into t_prps          "place row into header of t_prps
           where psphi eq t_proj-pspnr    "prps project = project rsn
           and zbillt eq p_billt.         "billing type input param (01)
```

```
      append t_prps.                 "append header to body of table
    endselect.         "notice no keyword table which is why the endselect
    if sy-subrc <> 0.
       write t_proj-pspnr to w_posid.   "convert to wbs element ext #
       w_error = 'Project & with no WBS element'(002).
       replace '&' with w_posid(7) into w_error.
       write w_error.
    endif.
  endloop.    "loop through all the billable projects
```

For all entries faster than appending

Instead of appending (as in above example) do the following (but make sure unique key is in select clause):

```
  select objnr pspnr posid stufe fakkz zbillt psphi zend ztermcd "pspnr is unique
       from prps into table t_prps      "place row into header of t_prps
       for all entries in t_proj        "all billable projects in internal table
       where psphi eq t_proj-pspnr      "prps project = project rsn
       and zbillt eq p_billt.           "billing type input param (01)
```

If the select clause is not unique, it is the equivalent of using a "select distinct". In other words duplicates will not be selected.

Corresponding fields

```
data: begin of t_kna1 occurs 0,
        kunnr like kna1-kunnr,
        name1 like kna1-name1,
      end of t_kna1.

select name1 from kna1 into corresponding fields of table t_kna1.
```

"For all entries in" 3 pitfalls to avoid

```
  select shkzg wrbtr saknr "debit/credit indicator, amount, GL acct
    from bseg into table t_bseg
    for all entries in t_bkpf
    where belnr = t_bkpf-belnr and bukrs = 'CUR '.
```

1) This is the equivalent of saying "select distinct shkzg wrbtr saknr"

```
It will only pick up one line item if multiple line items appear
with the same debit/credit indicator, amount and GL Account.

If you want all occurrences of these you must have a select statement
that includes the table's unique key, also called primary key. In the
data dictionary (SE11) the fields belonging to the unique key are marked with
an "X" in the key column.
```

instead do:

```
select bukrs belnr gjahr buzei shkzg wrbtr saknr "bseg unique key + d/c ind, amt, GL acct
  from bseg into table t_bseg
  for all entries in t_bkpf
  where belnr = t_bkpf-belnr and bukrs = 'CUR '.
```

2) FOR ALL ENTRIES IN...acts like a range table, so that if the "one" table is empty, all rows in the "many" table are selected.

```
  if not t_proj[] is initial.
    select objnr pspnr posid stufe fakkz zbillt psphi zend ztermcd
           from prps into table t_prps
           for all entries in t_proj        "all billable projects
           where psphi eq t_proj-pspnr      "prps project = project rsn
           and zbillt eq p_billt.         "billing type input param (01)
  endif.                           "if there are any projects
```

3) If the parent table (t_proj) is very large there is performance degradation

Instead loop through parent table(t_proj), appending to child table (t_prps)

```
  loop at t_proj.
    select objnr pspnr posid stufe fakkz zbillt psphi zend ztermcd
           from prps into t_prps          "place row into header of t_prps
           where psphi eq t_proj-pspnr    "prps project = project rsn
           and zbillt eq p_billt.         "billing type input param (01)
      append t_prps.                 "append header to body of table
    endselect.        "notice no keyword table which is why the endselect
    if sy-subrc <> 0.
       write t_proj-pspnr to w_posid.   "convert to wbs element ext #
       w_error = 'Project & with no WBS element'(002).
       replace '&' with w_posid(7) into w_error.
       write w_error.
    endif.
  endloop.    "loop through all the billable projects
```

Where clause (rules based) example

```
      where objnr = c_offset "OR000009999900 dummy int order
        and lednr = '00' "standard ledger
        and versn = '000'. "General Planning
        and wrttp = c_actual "04 actuals value type
        and gjahr =< gjahr "fiscal year
        and kstar = c_overrun "0000422342 cost overruns
        and beknz <> c_settlement. "A means a settlement
```

Delete all rows from table

```
      Delete from zzsupradr client specified where mandt = sy-mandt.
```

Append one table to another when structures are the same

```
      data t_name           like is_name occurs 0 with header line.
      data t_prob           like is_name occurs 0 with header line.


299    loop at t_name where room = space.
300      write: / t_name-name, t_name-room, t_name-cnt.
301      move t_name-name to t_prob-name.
302      move t_name-room to t_prob-room.
303      move t_name-cnt  to t_prob-cnt.
304      append t_prob.
305    endloop.
```

instead do:

```
299    loop at t_name where room = space.
300      write: / t_name-name, t_name-room, t_name-cnt.
301      append t_name to t_prob.
305    endloop.
```

In source documentation

```
*
* internal table to rollup the following totals to billable wbs element:
* expenditure, budget, revenue, wip (work in process, i.e. not billed),
* overexpenditure & to be billed. unique key is psphi (project #),
* posid (billable wbs element), pspnr (billable internal wbs number),
* vbeln (sales doc # for sales order),
* posnr (line item # for sales order), zend (expiration date on posid),
* ztermcd (term code on posid), zspon (sponsor customer id), billreq
* (billing request sales doc #)
*
data: begin of t_tab occurs 0,
        psphi  like prps-psphi,        " project internal number
        posid  like prps-posid,        " rollup external WBS element
        pspnr like prps-pspnr,         " rollup WBS element internal rsn
        vbeln like vbap-vbeln,         " SD order number
        posnr like vbap-posnr,         " SD order item #
        zend   like prps-zend,         " expiration date for rollup wbs
        ztermcd like prps-ztermcd,     " term code for rollup wbs
        zspon like prps-zspon,         " customer ID
        billreq like vbap-vbeln,       " Billing Request Sales Doc #
        total  like rpsco-wtp00,       " total expenditure
        budget like rpsco-wtp00,       " authorized total
        revenue like rpsco-wtp00,      " total revenue
        wip like rpsco-wtp00,          " expense - revenue
        overex like rpsco-wtp00,       " overexpenditure
        tobill like rpsco-wtp00,       " wip - overex
        begbal like rpsco-wtp00,       " beginning balance
        billed like rpsco-wtp00,       " billing request doc total
      end of t_tab.
```

Beginning block of source code documentation

At the top of the program (pattern button -> other pattern -> DOCU_BLOCK):

```
************************************************************************
*  SYSTEM: CO
*  AUTHOR: Allan Davidson
*  DATE  : June 1999
*  TITLE : Change Supervisor/Addressee/Rooms/mail codes
* PURPOSE : This report changes Supervisor, Addressee,
*           Supervisor room, Addressee room, or mailcode
*           of WBS-elements, cost center or orders.
*           This is a modified copy of ZCOCHNG with two major changes:
*           1. MIT ID is used for searching instead of name.
*           2. An interactive list is created, rather than a word
*              document, for checking updates prior to making them.
*
*           There are two possible steps:
*           1. Search possible WBS-elements, cost center or orders,
*              which are like the inputfield and display result in an  *
*              interactive list.
*           2. Update the cost objects from the interactive list.
************************************************************************
*  AMENDMENTS:
*  VER    DATE      AUTHOR    DESCRIPTION OF CHANGE
*  ===  ========  ==========  ====================
************************************************************************
```

At the top of FORMs (Allan Davidson wrote ZZARD007 in SF2 to check this documentation for a program)

```
*&---------------------------------------------------------------------*
*&      Form  CALCULATE_ROLLUP_TOTALS
*&---------------------------------------------------------------------*
**********************************************************************
* Calculate wip (work in process), cost overruns, to be billed.      *
* If to be billed is zero delete from table.                         *
* Get the sales order numbers to which the WBS elements are linked   *
* and report on any projects with wip but no sales order.            *
**********************************************************************
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
```

Check statements versus where clause criteria

```
  loop at t_prps where tabid = w_posid.  "problem wbs elements
    select distinct meinb kstar into table t_coep   "collect unit meas.
           from coep                      "& cost elements from trans
           where lednr = '00'             "encourage index coep_1
             and objnr = t_prps-objnr     "wbs object #
             and wrttp in r_wrttp         "31, 03-12 actuals value types
             and versn = '000'            "encourage index coep_1
             and kstar in r_kstar_cosp.   "expense cost elements

    check sy-subrc = 0.  "expenses found for this wbs element continue otherwise get next wbs

    loop at t_coep.
      check t_coep-meinb = w_meinb.   "problem unit of measurement
      ...
    endloop.   "loop through cost elements and units of measurement
  endloop.    "loop through wbs elements with potential problem
```

instead do:

```
  loop at t_prps where tabid = w_posid.  "problem wbs elements
    select distinct kstar into table t_kstar   "collect potential
           from coep                      "problem expense cost elements
           where lednr = '00'             "encourage index coep_1
             and objnr = t_prps-objnr     "wbs object #
             and wrttp in r_wrttp         "31, 03-12 actuals value types
             and versn = '000'            "encourage index coep_1
             and kstar in r_kstar_cosp    "expense cost elements
             and meinb = w_meinb.   "problem unit of measurement

    check sy-subrc = 0.  "problem expenses found for this wbs element continue otherwise get next wbs

    ...
  endloop.   "loop through wbs elements with potential problem
```

BUT MAKE SURE THAT THIS CONTINUES TO USE THE INDEX!

Use text elements or constants

Use text elements for translatable text.
Use constants for non-translatable literals.
This sample code is a good example because it uses the text element (012) in conjunction with the hard coded text.This documents the text element and provides for the possibility of multi-language support.

```
constants:
  c_csks(4) value 'CSKS',             " Cost centers

    write: / c_csks, '-- # of rows need be updated:'(012), w_cnt.
```

SQL statistical functions

```
*---------------------------------------------------------------------*
*  CALCULATE AUTHORIZED TOTAL PER WBS ELEMENT
*---------------------------------------------------------------------*

    select wtp00 from rpsco into table t_rpsco    "summarized budget
      where objnr = t_prps-objnr        "PR+rsn wbs elem object
        and wrttp eq c_budget.          "41 budget value type
    loop at t_rpsco.
      add t_rpsco-wtp00 to t_prps-budget. "total budget
    endloop.
```

instead do:

```
  select sum( wtp00 ) from rpsco into t_prps-budget  "summarized budget
      where objnr = t_prps-objnr        "PR+rsn wbs elem object
        and wrttp eq c_budget.          "41 budget value type
```

FORM Parmeter

```
      perform status_change using c_status.

form status_change using value(p_c_status).
  ...
endform.
```

Test return codes or handle the possibility that the unexpected happens

Example 1:

```
loop at t_bseg.
      clear w_name.
      read table t_kna1 with key kunnr = t_bseg-kunnr binary search.
      if sy-subrc = 0.
        w_name = t_kna1-name1.
      endif.
endloop.
```

Example 2:

```
 loop at t_proj.
    select objnr pspnr posid stufe fakkz zbillt psphi zend ztermcd
           from prps into t_prps          "place row into header of t_prps
           where psphi eq t_proj-pspnr    "prps project = project rsn
           and zbillt eq p_billt.         "billing type input param (01)
      append t_prps.                "append header to body of table
    endselect.        "notice no keyword table which is why the endselect
    if sy-subrc <> 0.
       write t_proj-pspnr to w_posid.   "convert to wbs element ext #
       w_error = 'Project & with no WBS element'(002).
       replace '&' with w_posid(7) into w_error.
       write w_error.
    endif. "see below for explanation of why the return code is tested after endselect
  endloop.    "loop through all the billable projects
```

Paraphrased from "Teach Yourself ABAP/4 in 21 Days" by Ken Greenwood (page 80):

If you have coded a select ... endselect, the value of sy-subrc must be tested after the endselect. "Why? The answer lies in the fact that the code between the select and endselect is executed once for each row returned from the database. If zero rows are returned from the database, the code between select and endselect is never executed. Therefore, you must code the test for sy-subrc after the endselect."

Example 3:

```
  select vbeln                          "uses VBAK______A index
    from vbak into table t_vbak
    where vbeln in r_vbeln              "billing request doc range '0070000000' - '0074999999'
      and erdat = max_datum             "original run date
      and auart = c_zpsr                "billing request ZPSR
      and vkorg = c_vkorg          "sales org 1000 (sponsored programs)
      and ernam = t_zjobrun-uname.      "user who ran original job
  if sy-subrc <> 0.
    clear w_samedate.                   "no bills were created
    exit.
  endif.
```

Example 4:

```
    call transaction 'KB14'
      using t_bdctab mode 'N' messages into t_msgtab.
    if sy-subrc = 0.
      write: / t_doc-posid(8),
             'Overexpenditure posting'(038), t_doc-doc01,
             'reversed:'(039), sy-msgv1(10).
      commit work.
    else.
      reversal_trouble = 'X'.
      clear t_error.
      t_error-posid = t_doc-posid.
      t_error-vbeln = t_doc-doc01.
      perform process_errors.
      t_error-sortkey = c_error_reversed.
      append t_error.
      perform bdc_session_create using 'Reverse' 'KB14'.
    endif.                 "was the call transaction successful
```

Example 5:

```
(1) If a function module has no exceptions, sy-subrc will always be 0.
An example of such a function module is Z_R3_LIST_REPORT_SELECTIONS.
It is pointless to test a return code from such a function module and
the "pattern" button doesn't suggest it.

(2) If a function module always uses the construct, MESSAGE ...
RAISING ..., then it is handling exceptions with its own error
messages. Unless you want processing to continue even if an error is
encountered, you can drop the "EXCEPTIONS"  section when you call the
function module. For instance, below is an example of redundant code
created using the "pattern" button.

  call function 'SSF_FUNCTION_MODULE_NAME'
       exporting
            formname           = p_form
       importing
            fm_name            = w_fmname  " generated function module
       exceptions
            no_form            = 1
```

```
           no_function_module = 2
           others             = 3.
  if sy-subrc <> 0.
    message id sy-msgid type sy-msgty number sy-msgno
           with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  endif.
```

This code should be changed to:

```
 call function 'SSF_FUNCTION_MODULE_NAME'
       exporting
            formname           = p_form
       importing
            fm_name            = w_fmname.  " generated function module
* function module is handling bad return code with appropriate error messages
```

The comment is helpful for the reviewer but should not be mandatory.

(3) If a function module uses the construct, RAISE ...., then it short dumps when an exception is encountered! Regardless of whether you want to continue or stop processing, you need to handle bad return codes from such function modules to avoid a short dump. The following code would cause **a short dump** in the unlikely event that p_domain is invalid:

```
  call function 'DDIF_DOMA_GET'
       exporting
            name          = p_domain
            langu         = sy-langu
       tables
            dd07v_tab     = t_dd07v.
```

On the other hand the following code, produced by the "pattern" button, **won't work** because the system fields sy-msgid, sy-msgty, sy-msgno etc, are not filled when the construct, RAISE ...., is used:

```
 call function 'DDIF_DOMA_GET'
       exporting
            name          = p_domain
            langu         = sy-langu
       tables
            dd07v_tab     = t_dd07v
       exceptions
            illegal_input = 1
           others        = 2.
  if sy-subrc <> 0.
    message id sy-msgid type sy-msgty number sy-msgno
           with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  endif.
```

There are many valid ways to handle return code checking on a function module like this. Three reasonable examples are:

```
  call function 'DDIF_DOMA_GET'
       exporting
            name          = p_domain
            langu         = sy-langu
       tables
            dd07v_tab     = t_dd07v
       exceptions
           others        = 1.
  if sy-subrc <> 0.
    p_desc = p_code.
    exit.
  endif.

  call function 'DDIF_DOMA_GET'
```

```
       exporting
            name          = p_domain
            langu         = sy-langu
       tables
            dd07v_tab     = t_dd07v
       exceptions
           others        = 1.
* sy-subrc  will always be 0 because p_domain is always valid
```

The comment is helpful for the reviewer but should not be mandatory.

```
  call function 'JOB_CLOSE'
       exporting
            jobname              = my_job-jobname
            jobcount             = my_job-jobcount
            strtimmed            = c_yes
       exceptions
            cant_start_immediate = 1
            invalid_startdate    = 2
            jobname_missing      = 3
            job_close_failed     = 4
            job_nosteps          = 5
            job_notex            = 6
            lock_failed          = 7.
  if sy-subrc eq 0.
*                                   Job & number & started in background
    message i180(26) with my_job-jobname  my_job-jobcount.
  else.
    case sy-subrc.
      when 1.
        w_problem = 'Can not start immediate'(071).
      when 2.
        w_problem = 'Invalid start date'(070).
      when 3.
        w_problem = 'Jobname missing'(069).
      when 4.
        w_problem = 'Job close failed'(068).
      when 5.
        w_problem = 'Job no steps'(013).
      when 6.
        w_problem = 'Job notex'(010).
      when 7.
        w_problem = 'Lock failed'(007).
      when others.
        w_problem = sy-subrc.
    endcase.
* System error: Termination in routine JOB & &
    message e040(zz) with 'CLOSE'(014) w_problem.
  endif.
```

Example 6:

```
  if w_batch < 999.
    lw_action = 'insert'(041).
    zjobrun-mandt = sy-mandt.
    zjobrun-pgnam = w_report_name.
    w_batch = w_batch + 1.
    zjobrun-batch = w_batch.
    zjobrun-uname = sy-uname.
    insert into zjobrun values zjobrun.
  else.
    lw_action = 'update'(042).
    update zjobrun
       set datum = lw_date
           uzeit = lw_time
           uname = sy-uname
```

```
            reccount = 0
            errcount = 0
            cntrlrec = 0
            credit = 0
            debit = 0
            errfile = ' '
            review = ' '
            text = ' '
          where pgnam = w_report_name
            and batch = w_batch.
  endif.
  if sy-subrc = 0.
    commit work.
  else.
*   Required update of table ZJOBRUN failed
    message i032 with lw_action sy-subrc.
    stop.
  endif.
```

Example 7:

Watch out for MODIFY - should use update or insert instead

Example 8: (dataset)

```
  open dataset w_fname for input in text mode   " get input data
                             message lw_msg.    " any errors place here
  if sy-subrc <> 0.
    message i002 with lw_msg.
    stop.
  endif.

   read dataset w_fname into lw_row.
    if sy-subrc <> 0.                           " end of file reached
      exit.                                     " leave do loop
    endif.


  loop at lt_files.
    delete dataset lt_files-name.
    if sy-subrc = 0.
      t_body-line = lt_files-name.
      append t_body.
    endif.
  endloop.


form write_unix_file.
  open dataset myfile for output in text mode.
  if sy-subrc <> 0.
    write sy-subrc to w_subrc.
    concatenate 'Problem opening' myfile 'to UNIX.' w_subrc
                    into t_body-line separated by space.
    append t_body.
    concatenate p_maber 'Invoice Load problems encountered'
                        into w_subject.
    perform send_mail.
    message a000 with 'Problem opening UNIX.' sy-subrc." stop processing because transfer will short dump
  endif.                                "problem with open
  loop at t_invoice.
    transfer t_invoice to myfile.
  endloop.                              "loop through invoices
  close dataset myfile.
```

Handling Dates

Example 1: Dates being used in a BDC session or Call Transaction

```
data:
       w_bldat(10),                    "doc date - string conversion
       w_budat(10),                    "posting date - string conversion
       d_bldat   like sy-datum,        "doc date - today's day
       d_budat   like sy-datum.        "posting date - last day of month

    d_bldat = sy-datum.                "doc date
    call function 'RP_LAST_DAY_OF_MONTHS'
         exporting
              day_in            = d_bldat
         importing
              last_day_of_month = d_budat
         exceptions
              day_in_no_date    = 1.
    if sy-subrc = 0.
      write d_bldat to w_bldat.      "WRITE uses conversion exit to format date
      write d_budat to w_budat.      "allows for correct user defaults
    endif.

  perform dynpro using:
          'X' 'SAPMF05A' '0100',
          ' ' 'BKPF-BLDAT' w_bldat,    "07/09/1999 if user's defaults are mm/dd/yyyy
          ' ' 'BKPF-BUDAT' w_budat,    "07/31/1999
          ' ' 'BKPF-BLART' w_blart,    "doc type from zardiv
          ' ' 'BKPF-BUKRS' w_bukrs,    "company code CUR
          ' ' 'BKPF-WAERS' w_waers,    "currency USD
          ' ' 'BKPF-MONAT' w_monat,    "POST PERIOD
          ' ' 'RF05A-NEWBS' w_bschl_deb,    "customer debit posting key
          ' ' 'RF05A-NEWKO' cur_id.    "customer ID
```

Example 2: Default dates being set during INITIALIZATION

```
data      w_date         like sy-datum.        "used to build defaults
parameters: p_start like bsid-zfbdt default '',     "begin baseline date
            p_end   like bsid-zfbdt default ''.       "end baseline date

initialization.
  w_date = sy-datum.
  w_date+6(2) = '01'.
  w_date = w_date - 1.
  w_tdate = w_date.
  w_date+6(2) = '01'.
  w_fdate = w_date.
  if sy-batch eq ' '.
    p_end = w_tdate.    "do not WRITE w_tdate to p_end!
    p_start = w_fdate.  "19990601 actual 06/01/1999 displays 1//19/06/0 if WRITE had been used
  endif.
```

Example 3: Dates being used in the SUBMIT statment

```
parameters: p_date like sy-datum.

initialization.
  p_date = sy-datum.            "do not WRITE sy-datum to p_date!

start-of-selection.
  submit zzard036 with p_datum = p_date and return. "submits with value 19990709

REPORT ZZARD036.
```

```
parameters: p_datum like sy-datum.               "receives 19990709

start-of-selection.

  write: / 'called program:', p_datum.  "the proper formatting occurs here on write 07/09/1999
```

Typed Parameters

```
data: begin of t_vbrk occurs 0,
        vbeln like vbrk-vbeln,                  " bill doc number
        xblnr like vbrk-xblnr,                  " audit voucher number
        fkdat like vbrk-fkdat,                  " FI posting date
        zuonr like vbrk-zuonr,                  " prime contract number
        kunrg like vbrk-kunrg,                  " payer
        netwr like vbrk-netwr,                  " current billed amount
        contract like vbak-vbeln,               " SAP prime contract
        srv_from like vbrk-fkdat,               " begin service date
        cumulative like vbrk-netwr,             " cumulative billed
        budget like zmiprdetail-program_amount, " funded amount
      end of t_vbrk.

constants:
  c95(2) type c value '95',                     " 95 contract
  c1995(5) type c value 'F1995',                " 1995 prime contract
  c20(3) type c value 'F20'.                    " contracts in 21st cent

    perform find_prime_contract using <f_vbrk>-zuonr+7(2)
                              changing <f_vbrk>-contract.

form find_prime_contract using    p_year type c       " note that you can't specify length
                          changing p_contract like vbak-vbeln.

  if p_year  = c95.                                       " '95'
    p_contract = c1995.                                   " 'F1995'
  else.
    concatenate c20 p_year into p_contract.           " F2000, F2005...
  endif.
endform.                    " find_prime_contract
```

Sample code for a [complex select statement using subqueries] (both EXISTS and NOT EXISTS)