Workshop Abap

Atenção! Esta lista de exercícios faz parte do Workshop de Abap, não é um material de auto estudo. Deve ser feita sob a orientação de um instrutor.

Use o seu número de aluno para nomear os seus objetos, assim:

Zxx_yyyyyyyyyyyyyyyyyyyyy

Onde:

Z : obrigatório na maioria dos objetos criados pelo cliente SAP (Z ou Y, usaremos Z neste curso);

xx : o seu número de aluno;

_ (travessão) : não é obrigatório, mas vai ajudar na busca de programas do curso;

yyyyyyyy : livre, mas não pode conter espaços;

Exemplos: Z05_EXERCICIO_3_1, Z12_LISTA_CLIENTES

Observação: para nomes de programas e identificadores dentro do código (palavras reservadas e nomes de variáveis), o abap não diferencia minúsculas de maiúsculas.

Tópico 1 – Introdução, programas Report e telas de seleção

Observação: em geral os programas do tipo Report seguem a seguinte divisão:

- ✓ Declarações globais (tipos, variáveis e classes)
- ✓ Tela de seleção e seus eventos
- ✓ START-OF-SELECTION
- ✓ END-OF-SELECTION.
- ✓ Definições de rotinas (FORM)

1.1 - Tela de seleção, declaração de variáveis:

- Montar tela de seleção com 2 números inteiros (PARAMETERS);
- Declarar 4 variáveis inteiras (DATA). Cada uma armazenará o resultado de uma operação com os parâmetros de entrada: soma, diferença, produto e quociente (resultado da divisão do primeiro número pelo segundo número);
- Neste exercício vamos trabalhar somente com variáveis inteiras;

- Lembre que não existe divisão por zero. Caso ocorra este cenário, imprima o literal "INEXISTENTE". Será necessário uso do comando IF ou tratar a exceção com TRY...CATCH:

```
TRY.
   <expressão aritimética>
  CATCH cx_sy_zerodivide.
   <tratamento do erro>
ENDTRY.
```

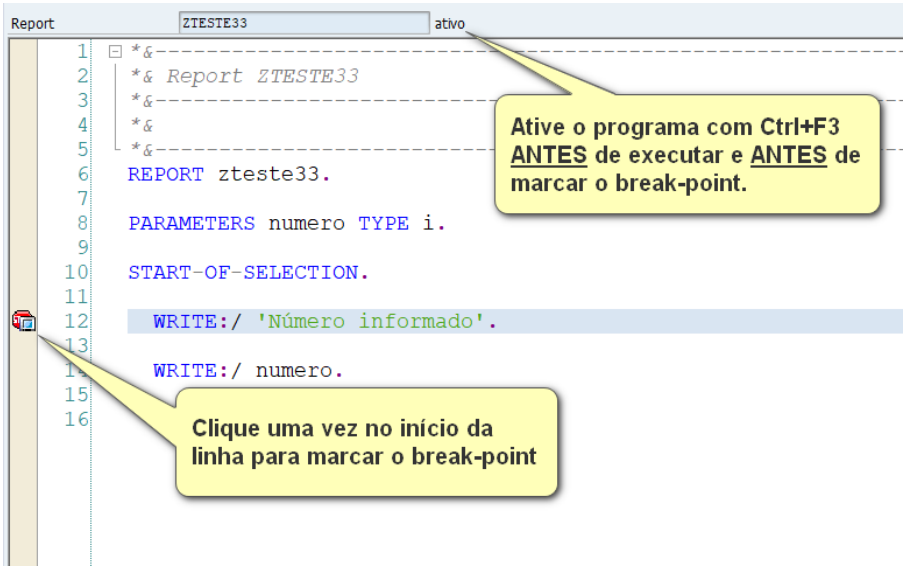
- Imprimir (WRITE) os dois números digitados e o resultado das operações, conforme o lay-out abaixo:

```
Número 1: 999
Número 2: 999
Soma: 999999
Diferença: 999999
Produto: 999999
Divisão: 999999
```

- **<u>Atenção para os procedimentos de testes abaixo. Ele será necessário em quase todos os exercícios deste curso:</u>**

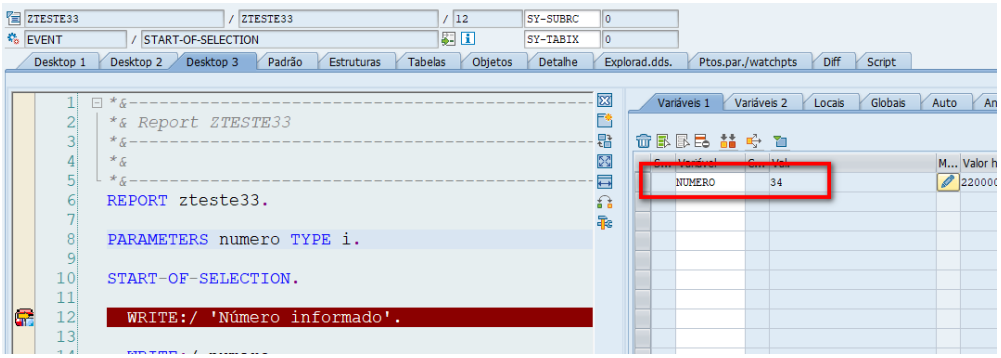
  <u>Procedimentos de testes:</u>

  a) Ativar o programa (Ctrl+F3) e elimine os erros de ativação. Não ative programas com erros!
  b) **<u>Antes de executá-lo</u>** (F8), coloque um break-point na primeira linha executável (primeira linha do evento START-OF-SELECTION), conforme o exemplo abaixo:



  c) Tecle F8 para executar o programa;
  d) Tecle F5 na janela de debug para executar cada linha do programa;

e) Mostre na janela da direita todos os valores das variáveis de tela e do programa.



Cenário de testes:

1º número	2º Número	Soma	Diferença	Produto	Quociente
2	3	5	-1	6	1
3	2	5	1	6	2
0	17	17	-17	0	0
1	3	4	-2	3	0
6	2	8	4	12	3
673	201	874	472	135.273	3
45	0	45	45	0	INEXISTENTE
-17	-3	-20	-14	51	6
-4	8	4	-12	-32	-1
2	-2	0	4	-4	-1

Obs.:

- apesar de não ser obrigatório, é uma boa prática colocar o nome do evento START-OF-SELECTION;
- o comando WRITE está em desuso para exibição de valores em tela para o usuário, mas como ferramenta de aprendizado é mais prático do que outros recursos de exibição (janelas popup e ALV, vistos mais adiante neste curso).

1.2 - Tela de seleção, loop, IF

- Tela de seleção: 2 números inteiros e obrigatórios;
- Fazer as verificações abaixo. Em caso de erro, encerrar o programa (comando STOP):

  - os dois números devem ser positivos e não podem ser zero; caso contrário → Erro;
  - o primeiro número deve ser menor do que o segundo número; caso contrário → Erro;

- a diferença entre o segundo número e o primeiro número deve ser pelo menos 5 (cinco). Se for menor do que 5 → Erro;

- Se não ocorrer nenhum erro na etapa anterior, use o comando WHILE … ENDWHILE para imprimir as duas listas abaixo:
  - Lista 1: imprimir uma sequência de números inteiros começando do primeiro número até o segundo número. Por exemplo, se o primeiro número for 5 e o segundo número for 13, imprimir: 5 6 7 8 9 10 11 12 13;
  - Lista 2: imprimir a mesma sequência anterior, mas em ordem inversa, do segundo número até o primeiro número. Por exemplo, se o primeiro número for 5 e o segundo número for 13, imprimir: 13 12 11 10 9 8 7 6 5;
- Calcular e imprimir a soma da lista de números;
- Calcular e imprimir a média aritmética simples da lista de números;
- Calcular e imprimir a multiplicação do terceiro item da lista 1 pelo quinto item da lista 2. Por exemplo, se os números informados forem 5 e 13, temos:

  - Lista 1: 5 6 7 8 9 10 11 12 13 → terceiro item é 7;
  - Lista 2: 13 12 11 10 9 8 7 6 5 → quinto item é 9;
  - Para este exemplo o resultado da multiplicação seria 63 (7 x 9).

- Não utilize variáveis estruturadas neste exercício: listas, vetores, matrizes, estruturas, arrays, tabelas internas.

Cenário de testes:

1º número	2º Número	Resultado
2	3	Erro: diferença menor do que 5
3	2	Erro: primeiro número maior do que o segundo
0	17	Erro: número inválido
5	13	Imprimir conforme exemplo do enunciado e produto = 63
45	0	Erro: número inválido
-17	-3	Erro: número inválido
-4	8	Erro: número inválido
2	-2	Erro: número inválido

Desafio: imprima novamente uma lista crescente, do menor número até o maior número, mas somente números pares. Dica: como obter o resto de uma divisão por 2?

1.3 - Tela de seleção, substrings, operações com datas

- Tela de seleção (2 campos - PARAMETERS)

  - Nome com 30 posições caracter (obrigatório);

- Campo no formato data (obrigatório);
- Use o evento INITIALIZATION para inicializar a data da tela com o dia de amanhã;

- Imprimir:

  - Data do dia anterior à data informada;
  - Data com o primeiro dia do mês seguinte. Exemplos:

    - se a data informada for 17.08.2026, imprimir 01.09.2026;
    - se a data informada for 08.12.2021, imprimir 01.01.2022;

  - Número de dias entre a data informada e a data atual (somente se a data informada for maior que a data atual);
  - Tamanho (em caracteres) do nome (função STRLEN);
  - 3 primeiros caracteres do nome informado na tela de seleção;
  - 2 últimos caracteres do nome informado na tela de seleção;
  - Imprimir a data informada por extenso, por exemplo, 25 de janeiro de 2025. Para encontrar o mês utilize o comando IF ou o comando CASE.

1.4 - Tela de seleção, evento AT SELECTION-SCREEN, variáveis de ambiente

- Fazer uma tela para cadastro de usuário, com 5 parâmetros:

  - Nome (char40) – obrigatório --> JOSÉ
  - Sobrenome (char30) – obrigatório --> ANTUNES: ANTUN0624
  - Data inicial – obrigatório – inicializar com a data atual;
  - Data final
  - Checkbox "Aplicar data no código", com o valor default ' ' (desmarcado).

- Consistir no evento AT SELECTION-SCREEN, com mensagens do tipo "E" em caso de erro:

  - a data inicial deve ser maior ou igual a data atual. Se não for, mostrar mensagem de erro “A data inicial deve ser maior ou igual a data atual”;
  - se a data final for informada, ela deve ser maior ou igual a data inicial. Caso contrário, mostrar a mensagem de erro: “Data final deve ser maior ou igual a data inicial”;
  - para as mensagens, criar uma classe de mensagens Zxx na transação SE91, onde "xx" é o seu número de usuário;

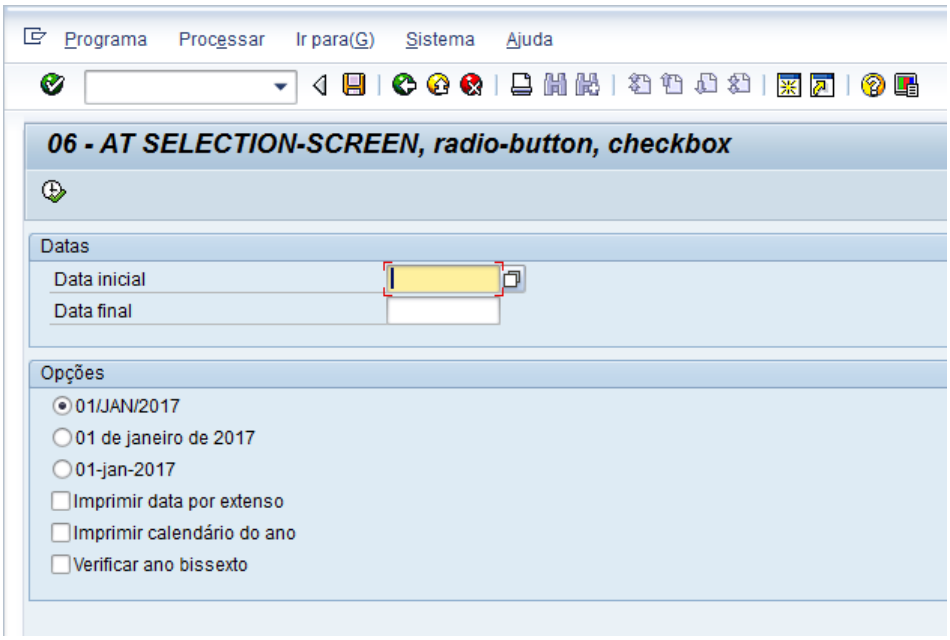
- Montar e imprimir um código de usuário:

  - Se o checkbox estiver marcado, o código de usuário deve ser formado pelos 5 primeiros caracteres do sobrenome + mês + ano da data inicial informada na tela (mês e ano com dois dígitos cada um). Não é necessário verificar o tamanho do sobrenome;

- Se o checkbox estiver desmarcado, o código de usuário deve ser formado pelo primeiro caracter do nome + o sobrenome;
- Em ambos os casos, eliminar os espaços em branco do código formado, usando o comando CONDENSE;

- Imprimir a quantidade de dias entre a data inicial e a data final (se a data final for informada);
- Imprimir o código do usuário (usuário de logon do sap - SY-UNAME) que está fazendo o cadastro, a data atual (SY-DATUM), a hora atual (SY-UZEIT) o nome do programa (SY-REPID) e a transação em execução (SY-TCODE).

1.5 - Tela de seleção, evento AT SELECTION-SCREEN, radio-button, checkbox



- A data inicial é obrigatória;
- A data final é opcional, mas se for informada:
  - deve ser menor ou igual à data atual (emitir mensagem de erro caso contrário);
  - deve ser maior que a data inicial (emitir mensagem de erro caso contrário);
- Dos três checkboxes da tela, pelo menos um dos dois primeiros DEVE estar marcado (emitir mensagem de erro caso contrário);
- Se o checkbox "Imprimir data por extenso" estiver marcado, imprimir as duas datas conforme o formato indicado no radio-button selecionado. Atenção: na primeira e terceira opções, o nome do mês deve vir com 3 letras;
- Se o checkbox "Imprimir calendário do ano" estiver marcado, imprimir todo o calendário do ano da data inicial (a partir de 01 de janeiro) um dia por linha, saltando uma linha a cada mês;

- Se o checkbox “Verificar ano bissexto” estiver marcado, verificar se o ano da data inicial é bissexto. Se sim, imprimir “XXXX é ano bissexto”, onde XXXX é o ano da data inicial.

2.1 - Recuperando dados de uma tabela – registro único

- Fazer uma tela de seleção com o ID do cliente (tabela SCUSTOM) – usar PARAMETERS
- Imprimir o nome, cidade e idioma do cliente selecionado
- Usar SELECT...SINGLE
- Observar rotina de conversão no idioma: consultar o registro na transação SE16 e ver o conteúdo do campo LANGU. Por que o valor listado pelo seu programa é diferente daquele consultado na tabela SCUSTOM (transação SE16)? Use agora a transação SE16N e faça a mesma consulta.

2.2 - Recuperando dados de várias tabelas:

- Fazer uma tela de seleção com os seguintes campos, todos eles não obrigatórios:
  - Código da companhia aérea (tabela SCARR - "Denominação breve da companhia aérea");
  - Número do voo (tabela SPFLI - "Nº da conexão de voo");
  - Data do voo (tabela SFLIGHT - "Data do voo");
  - Número da reserva (tabela SBOOK - "Nº de marcação");
- Os campos acima também são chaves primárias das tabelas citadas;
- Faça a leitura da tabela correspondente **SOMENTE** se todos os campos da chave primária da tabela forem informados (SELECT SINGLE);
- Se a leitura for feita e não encontrar registro, emitir mensagem (comando WRITE):
  - "Cia aérea não encontrada" (SCARR)
  - "Horário de voo não encontrado" (SPFLI)
  - "Voo não encontrado" (SFLIGHT)
  - "Reserva não encontrada" (SBOOK)
- Imprimir, se encontrar, para cada tabela:
  - SCARR: código da cia aérea, nome e site;
  - SPFLI: código da cia aérea, número do voo, cidade e país de origem, cidade e país de destino, duração do voo, nome do aeroporto de origem e nome do aeroporto de destino (tabela SAIRPORT - fazer 2 outros SELECT...SINGLE);
  - SFLIGHT: código da cia aérea, número do voo, data do voo, preço e moeda do voo, tipo de avião, velocidade de cruzeiro e fabricante do avião. Atenção: os dois últimos campos estão na tabela SAPLANE, fazer outro SELECT...SINGLE;
  - SBOOK: código da cia aérea, número do voo, data do voo, número da reserva, número do passageiro, peso da bagagem.

IMPORTANTE!! Verifique o conteúdo de cada tabela na transação SE16 ou SE16N, antes de fazer a codificação de cada leitura.

2.3 - Alterando dados de um registro

- Tela de seleção:
  - Código do passageiro (SCUSTOM-ID), obrigatório;
  - Novo e-mail do passageiro (SCUSTOM-EMAIL), obrigatório;
- No evento AT SELECTION-SCREEN:
  - Verificar a existência do código do passageiro. Se não existir, emitir a mensagem de erro "Passageiro não cadastrado".
  - Consistir o novo e-mail, usando a rotina VALIDA_EMAIL, logo abaixo;
- START-OF-SELECTION:
  - Atualizar (comando UPDATE ou MODIFY) o registro, alterando o e-mail.

- Observação importante: como regra geral, não atualizamos tabelas standard diretamente com comandos UPDATE, MODIFY, INSERT ou DELETE. Para este propósito, o SAP fornece outras ferramentas (vistas mais adiante).

```
*&---------------------------------------------------------------------*
*&      Para informações e tutorial de expressões regulares acesse:
*&      http://www.regular-expressions.info
*&---------------------------------------------------------------------*
FORM valida_email  USING p_email
             CHANGING email_ok.

  DATA: go_regex   TYPE REF TO cl_abap_regex,
        go_matcher TYPE REF TO cl_abap_matcher.

  CREATE OBJECT go_regex
    EXPORTING
      pattern     = '\w+(\.\w+)*@(\w+\.)+(\w{2,4})'
      ignore_case = abap_true.
  go_matcher = go_regex->create_matcher( text = p_email ).
  IF go_matcher->match( ) IS INITIAL.
    CLEAR email_ok.
  ELSE.
    email_ok = 'X'.
  ENDIF.

ENDFORM.                    "valida_email
```

2.4 - Inclusão, exclusão e alteração de registros

- Fazer um programa para manter a tabela SCARR. Campos: código, nome e site;
- Deixar somente o código obrigatório (SCARR-CARRID);
- Criar 3 radiobuttons: Criar, Alterar, Excluir. Colocar os radio-buttons dentro de uma moldura com o título "Operação". A opção "Alterar" deve ser o valor inicialmente marcado;
- Ao teclar F8 executar a ação selecionada no radiobutton;

- Observe que na alteração e exclusão é obrigatório que a cia aérea exista. Em caso de erro emitir a mensagem "Companhia aérea inexistente";
- Na inclusão a cia aérea não deve existir. Na tentativa de incluir uma cia aérea já existente, emitir a mensagem de erro "Companhia aérea já cadastrada";
- Apenas o código da cia e o nome são campos obrigatórios para inclusão e alteração. Na exclusão, somente o código é obrigatório;
- Use rotinas separadas (PERFORM) para cada operação,
- Observação importante: como regra geral, não atualizamos tabelas standard diretamente com comandos UPDATE, MODIFY, INSERT ou DELETE. Para este propósito, o SAP fornece outras ferramentas (vistas mais adiante).

3.1 - Recuperando dados em tabelas internas

- Tela de seleção: seleção múltipla do código do cliente (SELECT-OPTIONS);
- Fazer três rotinas para recuperar registros da tabela SCUSTOM em uma tabela interna. A primeira rotina usará tabela interna sem header line; a segunda usará com header line e a terceira rotina usará field-symbols.
- Em cada rotina imprimir o código, nome e cidade dos clientes a partir da tabela interna criada, usando uma work-area (primeira rotina), a linha de cabeçalho (segunda rotina) ou field-symbol (terceira rotina);
- Lembre: field-symbol é mais indicado. Tabela interna com linha de cabeçalho está obsoleta!

3.2 - Atualização de tabela transparente (do banco de dados) a partir de tabela interna

a) Consulte a tabela SFLIGHT na transação SE16 ou SE16N e anote em papel, bloco de notas, MS Word ou outro local:

- Selecione 3 ou mais registros com o valor de SEATSOCC superior a 60% do valor de SEATSMAX;
- Selecione 3 ou mais registros com o valor de SEATSOCC igual ou inferior a 60% do valor de SEATSMAX;

b) Tela de seleção:

- Código da companhia aérea, seleção múltipla, obrigatório;
- Data do voo, seleção múltipla, obrigatório. Usar o evento INITIALIZATION para inicializar a faixa de datas com o mês atual: coloque na data inicial o primeiro dia do mês e na data final o último dia do mês;

c) Aumentar em 15% o preço da passagem (campo SFLIGHT-PRICE) dos registros selecionados cuja ocupação na classe econômica seja inferior ou igual a 60%;

d) Sugestão:

- carregar para uma tabela interna os registros de acordo com a tela de seleção;

- eliminar dela os voos com ocupação superior a 60%. Usar os campos da classe econômica: SEATSMAX – número máximo de assentos; SEATSOCC – número de assentos ocupados;
- Em seguida, aplicar o aumento NA TABELA INTERNA;
- NÃO EXECUTAR MODIFY na tabela transparente!!!
- Teste o seu programa usando o debug, para verificar na tabela interna de resultado os registros levantados no item a). Somente avance para o item e) se sua tabela interna de resultado tiver SOMENTE as linhas que deveriam ser alteradas na tabela transparente SFLIGHT;

e) Coloque o MODIFY SFLIGHT from tabela_interna no seu programa, FORA do loop;

f) Após a execução do seu programa, consulte novamente na transação SE16/SE16N, os registros selecionados no item a);

- Observação importante: como regra geral, não atualizamos tabelas standard diretamente com comandos UPDATE, MODIFY, INSERT ou DELETE. Para este propósito, o SAP fornece outras ferramentas (vistas mais adiante).

3.3 – Programa: criar e preencher uma tabela interna com "hard code":

- Tela de seleção: não tem. Não use comandos PARAMETERS nem SELECT-OPTIONS neste programa;
- Use a transação SE11 para visualizar a estrutura BDCDATA do dicionário de dados. Observe o tipo e tamanho dos campos desta estrutura;
- Declarar uma tabela interna no programa com a linha do tipo BDCDATA;
- Fazer uma rotina ou método para inserir **UMA** linha nesta tabela interna. Esta rotina/método terá 5 (cinco) parâmetros de entrada (um para cada coluna da tabela interna) e nenhum parâmetro de saída ou retorno;
- Chamar a rotina/método anterior 10 vezes para preencher a tabela interna **exatamente** como mostrado abaixo. Estas 10 chamadas podem ser feitas dentro de uma outra rotina/método ou diretamente no evento START-OF-SELECTION:

PROGRAM	DYNPRO	DYNBEGIN	FNAM	FVAL
SAPMF02B	0100	X		
			BDC_OKCODE	/00
			BNKA-BANKS	BR
			BNKA-BANKL	104167824
SAPMF02B	0110	X		
			BDC_OKCODE	=UPDA
			BNKA-BANKA	CAIXA ECONÔMICA FEDERAL
			BNKA-STRAS	RUA DA BAHIA, 35
			BNKA-ORT01	CENTRO
			BNKA-BRNCH	CENTRAL

- Os valores acima serão informados diretamente no programa, com “hard-code”. Exemplo: 'BDC_OKCODE', '104167824', 'BNKA-BANKL', 'RUA DA BAHIA, 35', '0100', 'X', etc;
- Atenção para os valores, escreva **exatamente** como mostrado na figura acima:
  - no valor BDC_OKCODE foi usando um travessão (_);
  - no valor BNKA-BANKL foi usando um hífen (-);
- Não use o comando LOOP neste programa;
- Não é necessário imprimir nada;
- Observe que há valores vazios na tabela; não é necessário colocar nenhum valor na tabela interna para estes valores vazios;
- Use o debug para verificar se a tabela interna foi preenchida corretamente.

3.4 - Operações com tabela interna

- Este programa não tem tela de seleção;
- Declarar uma tabela interna de nome T_PASSAGEIRO com três campos, todos eles com referência à tabela SCUSTOM:
  - Número do passageiro;
  - Nome do passageiro;
  - E-mail do passageiro;
- Carregar para a tabela interna T_PASSAGEIRO todos os registros da tabela SCUSTOM;
- Para cada linha da tabela T_PASSAGEIRO, criar 2 (duas) linhas em outra tabela interna de nome T_RESERVA_PASSAGEIRO, com dados da tabela SBOOK. A tabela interna T_RESERVA_PASSAGEIRO deve ter 3 campos:
  - Número do passageiro;
  - Identificador, do tipo N com tamanho 2: este campo terá o valor 01 para a primeira linha do passageiro e valor 02 para a segunda linha;
  - Data do voo: na primeira linha (Identificador = 01), colocar a data do primeiro voo; na segunda linha (Identificador = 02), colocar a data do último voo. Havendo somente um voo para o passageiro, preencher a primeira linha com a data do voo e deixar em branco a data do voo na segunda linha;
- Não havendo reservas para um passageiro, não é necessário tomar nenhuma ação, ou seja, não precisa colocar nada na tabela T_RESERVA_PASSAGEIRO nem emitir mensagens;
- Como você vai encontrar o primeiro e último voo na tabela SBOOK? Sugestão: declarar uma tabela interna que será preenchida com todos os voos de UM passageiro. Faça um SELECT para ler os voos dentro do LOOP da tabela T_PASSAGEIRO. Em seguida classifique esta tabela e use READ TABLE para ler a primeira linha e depois a última linha;
- Este programa não tem tela de seleção nem saída visível. Para testá-lo, utilize o debug;
- **ATENÇÃO: não é recomendável usar comandos de banco dados dentro do comando LOOP, como foi feito na sugestão acima. Para este propósito temos outros recursos visto mais adiante no curso. Use o SELECT dentro do LOOP somente para este exercício;**

- Observação: parte desta atividade poderia ser feita com um SELECT e a cláusula JOIN. Não use JOIN neste exercício, a intenção é a prática na manipulação de tabelas internas.

3.5 – Cálculo da "taxa de embarque variável"

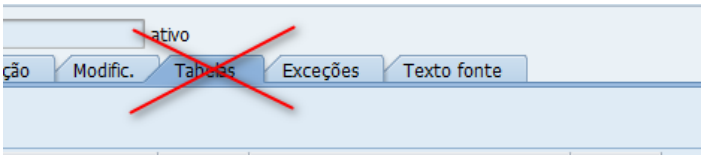
- Fazer um programa para calcular a "taxa de embarque variável" por companhia aérea;
- Tela de seleção:
  - Valor a ser rateado, obrigatório, do tipo s_price;
  - Período de datas, seleção múltipla, obrigatório;
- Regras:
  - O valor informado na tela será rateado para todas as companhias aéreas que tiverem reservas agendadas (tabela SBOOK) no período informado;
  - O peso de rateio de cada companhia será a soma do número de reservas;
  - Reservas em dezembro, janeiro e julho tem peso 100% maior que o normal, ou seja, cada reserva nestes meses deve ser considerada como 2 reservas;

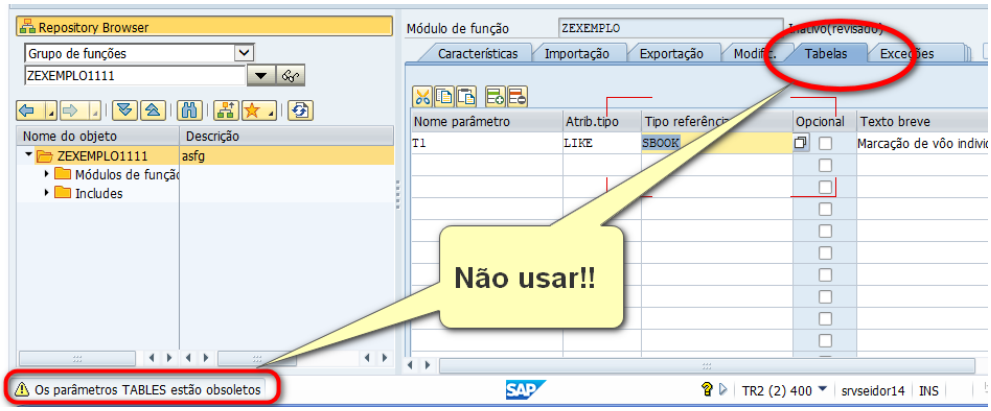
Imprimir o código da cia aérea, o peso e o valor rateado;
Dicas: a) carregar a tabela SBOOK; b) calcular o peso, acumulando em uma tabela interna (uma linha por companhia); c) calcular o rateio

4.1 – Módulo de função Zxx_VOO_4_1 (xx é seu número de usuário)

- Parâmetros de entrada obrigatórios: cidade de origem e aeroporto de origem;
- Parâmetros de entrada opcionais: cidade de destino, aeroporto de destino;
- Parâmetros de saída (exportação):
  - Tabela interna com o código da companhia, o número do voo, hora de partida e quantidade de voos programados. Os três primeiros campos estão na tabela SPFLI; a quantidade de voos programados é o total de ocorrências na tabela SFLIGHT para a cia aérea+número do voo;
  - Como a cidade de destino e aeroporto de destino são opcionais, o módulo de função deve estar preparado para os 4 casos abaixo. Se algum campo não for informado, não use ele no SELECT:
    - Caso 1: cidade de destino e aeroporto de destino não foram informados (IF xxxxx IS INITIAL...) – retornar todos os voos que partem da origem independente do destino;
    - Caso 2: cidade de destino foi informada e aeroporto de destino não foi informado – retornar todos os voos que partem da origem até a cidade de destino, independente do aeroporto de destino;
    - Caso 3: cidade de destino não foi informada e aeroporto de destino foi informado – retornar todos os voos que partem da origem até o aeroporto de destino, independente da cidade de destino;
    - Caso 4: cidade de destino e aeroporto de destino foram informados – retornar todos os voos que parte da origem até a cidade e aeroporto de destino;

  - Atenção! Não utilize a aba "Tables" do módulo de função, está obsoleto:





Utilize a aba "Exportação". Crie uma estrutura e tipo de tabela (SE11) para definição da tabela de retorno e coloque-o no "Tipo de referência" do parâmetro de exportação;

- Usar as tabelas SPFLI e SFLIGHT;
- Não use o comando WRITE dentro de módulos de função, ele é exclusivo para programas report.

4.2 - Estoque de material - faça o programa abaixo:

- Tela de seleção: nome do arquivo de entrada, seleção simples, obrigatório, tipo STRING (... type string);
- Utilizar o módulo de função GUI_UPLOAD (ou o método estático GUI_UPLOAD da classe CL_GUI_FRONTEND_SERVICES) para ler o arquivo texto informado na tela de seleção. Passe o valor X no parâmetro HAS_FIELD_SEPARATOR;
- Usar o arquivo Material-Centro-TR2.txt para testes. Para facilitar, copie-o para seu computador;
- Para carregar o arquivo texto, use uma tabela interna com três campos caracter com os tamanhos 18, 4 e 3;
- Para valores numéricos de código de material, o SAP armazena nas tabelas estes códigos com zeros à esquerda. Assim, TODOS os usos de código de material numérico dentro de um programa Abap deve ser feito com zeros à esquerda, veja:

Arquivo texto:

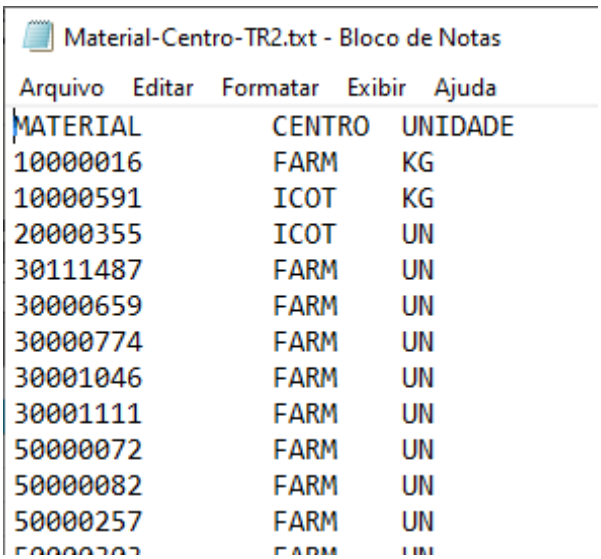
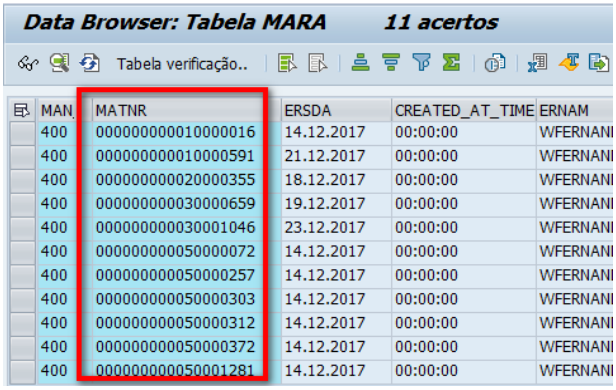


Tabela MARA:



- Como resolver? É necessário converter o código de material lido com o módulo de função CONVERSION_EXIT_MATN1_INPUT (rotina de conversão no domínio MATNR). Faça uma chamada deste módulo de função para cada material lido do arquivo texto;
- Para cada material/centro/unidade de medida lido do arquivo texto, encontre a quantidade de estoque disponível, utilizando o módulo de função BAPI_MATERIAL_AVAILABILITY. Passe os parâmetros PLANT (centro), MATERIAL (material) e UNIT (unidade de medida);
- O valor disponível em estoque está na primeira linha da tabela interna de retorno WMDVEX, campo COM_QTY. Apesar de não ser utilizado neste programa, a tabela de retorno WMDVSX é obrigatória;
- Observe que no arquivo texto pode ter algum material inexistente no SAP. Utilize o parâmetro RETURN: se ele for preechido na chamada da BAPI_MATERIAL_AVAILABILITY, indica que houve um erro na chamada do módulo de função. Neste caso, imprima o valor do campo MESSAGE;
- Listagem de saída: código do material, centro, unidade de medida, quantidade em estoque e mensagem (se houve erro);
- Atividade extra (A):

- inclua na listagem o tipo de material (MARA-MTART), grupo de mercadorias (MARA-MATKL), o número antigo do material (MARA-BISMT) e a descrição do material no idioma de logon (MAKT-MAKTX)
- Atividade extra (B):
  - inclua na listagem: código e nome do grupo de compradores (MARC-EKGRP e T024-EKNAM) e a quantidade de pedidos de compra existentes para o material (ver tabela EKPO, use o campo EKPO-MATNR para busca).

5.1 – Criação de tabela do usuário, SM30

- Criar a tabela ZXXUSUARIO_REDE, o diálogo de atualização e a transação para chamar a SM30:

Campo	Chave	Tipo	Tamanho	Dec	Descrição
MANDT	X	MANDT	3		Mandante
CODIGO	X	NUMC	5		Código do usuário
NOME		Caracter	40		Nome do usuário
SOBRENOME		Caracter	30		Sobrenome
DATA_INICIAL		Date	8		Início da atividade
DATA_FINAL		Date	8		Fim da atividade
USERNAME		Caracter	7		Nome do usuário na rede
DATA_CRIACAO		Date	8		Data de criação
USER_RESP		XUBNAME	12		Responsável pela criação
ATIVO		Caracter	1		Ativo (“X”) ou não “ “

- Criar um elemento de dados e domínio para cada campo, exceto para os campos MANDT (elemento de dados MANDT) e USER_RESP (elemento de dados XUBNAME);
- Para o campo ATIVO, criar também um domínio com os valores:
  - “X” – Ativo
  - “ “ - Desativado (um espaço em branco);
- Inserir 5 registros com a transação criada.

6.1 – Programa on-line com uma tela

- Criar tela 9000 com 4 campos:
  - Código da companhia aérea: campo de entrada obrigatório;
  - Sigla do aeroporto de origem: campo de entrada opcional;
  - Nome da companhia aérea: campo de saída (bloquear digitação)
  - Número de vôos: campo de saída (bloquear digitação)
- Criar botão na barra application: “Busca vôo” e associar a tecla Ctrl+F2. Atribuir também um ícone ao botão;
  Exibir o nome da companhia aérea ao processar o PAI;

- Ao acionar o botão, buscar a quantidade de vôos (tabela SPFLI) partindo do aeroporto de origem (SPFLI-AIRPFROM) e companhia aérea (SPFLI-CARRID) informados;
- Ativar o botão “BACK” e usar LEAVE PROGRAM;
- No PAI, buscar a quantidade de vôos e atribuir o valor à variável de tela;
- Criar transação ZQTDEVOO para chamar o programa;