Documentação Técnica: Script ADVPL LUFR001 (Relatório Auxiliar Clientes/Fornecedores)

Este documento detalha o script ADVPL `LUFR001.TXT`, uma customização desenvolvida para o ERP Protheus com a finalidade de gerar um relatório auxiliar de movimentações financeiras de Clientes ou Fornecedores (FINR510). O objetivo é fornecer uma referência completa sobre a estrutura, funcionalidade e parâmetros desta customização.

Visão Geral

O script `LUFR001` implementa a rotina de relatório financeiro `FINR510`, permitindo a emissão de um diário auxiliar com as movimentações de contas a receber ou a pagar dentro de um período e filtros específicos. Ele processa dados das tabelas de títulos e movimentações bancárias para apresentar um extrato detalhado ou resumido, conforme a configuração do usuário.

Componentes do Script

O script `LUFR001` é estruturado nas seguintes funções:

- **`lufr001()`**: A função de usuário principal que serve como ponto de entrada da customização. Ela gerencia a inclusão de arquivos de cabeçalho, a definição de variáveis de escopo privado, a chamada para uma possível rotina de log customizada (`U_LUFT_LOG`), a configuração dos parâmetros do relatório através da função `pergunte`, a configuração das opções de impressão (`SetPrint`, `SetDefault`), e a execução da rotina de impressão principal através de `RptStatus`.
-
- **`FA510Imp(lEnd, wnRel, cString)`**: Uma função estática crucial que contém a lógica de seleção e processamento dos dados. Ela define variáveis locais, trata a seleção de tipos de título com base em uma função externa (`FR510TIP`) e parâmetro (`mv_par15`), ajusta a tabela SX5 se necessário (`AjustaSx5`), processa a impressão de termos de abertura/encerramento, determina o modo de listagem (por filial ou empresa), configura os cabeçalhos do relatório, cria e gerencia um arquivo de trabalho temporário (`cNomeArq`), localiza e grava os títulos a receber (tabela SE1) e a pagar (tabela SE2) dentro dos parâmetros especificados, localiza e grava as movimentações bancárias (tabela SE5), e finalmente executa o loop de impressão dos dados contidos no arquivo temporário. Também é responsável pela limpeza dos arquivos temporários ao final.
-
- **`SubTot510(dData, nTotTit, nTotDeb, nTotCrd, aColu)`**: Uma função estática que imprime os subtotais de títulos, débito e crédito para cada data no relatório.
-

- **`GravaTrab(cTipo, dEmissao, dVencto, nValor, aDados, dDataConv, cAliasTmp)`**: Função estática responsável por gravar um registro no arquivo de trabalho temporário (`cNomeArq`). Ela recebe informações sobre o tipo de registro (título ou movimentação), datas, valor, dados do cliente/fornecedor e histórico, formatando-os para a estrutura do arquivo temporário e determinando o indicador de débito/crédito.
- **`Inside(cTp)`**: Função estática que verifica se um determinado tipo de título (`cTp`) está contido na lista de tipos selecionados pelo usuário (`cTipos`), considerando o parâmetro `mv_par15`.
- **`AjustaSx5()`**: Função estática que verifica a existência e, se necessário, inclui na tabela genérica SX5 (`05`) uma entrada para o tipo de moeda "R$" com as descrições correspondentes em português, espanhol e inglês.

Parâmetros de Entrada (MV_PARs)

O comportamento e os filtros aplicados na geração do relatório são controlados pelos seguintes parâmetros de sistema (`MV_PARs`), obtidos através da função `pergunte("FIN510",.F.)`:

- **`mv_par01`**: Data de emissão inicial para seleção dos títulos.
- **`mv_par02`**: Data de emissão final para seleção dos títulos.
- **`mv_par03`**: Tipo de entidade para o relatório: 1 para Clientes, 2 para Fornecedores.
- **`mv_par04`**: Código inicial do cliente ou fornecedor.
- **`mv_par05`**: Código final do cliente ou fornecedor.
- **`mv_par06`**: Página inicial para impressão.
- **`mv_par07`**: Página final para impressão.
- **`mv_par08`**: Opção de impressão de termo: 1 para Abertura, 2 para Encerramento, 3 para Nenhum.
- **`mv_par09`**: Indica se deve imprimir apenas o termo (1).
- **`mv_par10`**: Indica se imprime valores financeiros (1) ou apenas valores originais (2).
- **`mv_par11`**: Filtro por tipo de movimentação/título: 1 para Todos, 2 para Normais, 3 para Adiantamentos.

- `mv_par12`: Prefixo inicial dos títulos.
- `mv_par13`: Prefixo final dos títulos.
- `mv_par14`: Define a abrangência da listagem: 1 para Filial atual, 2 para Empresa (todas as filiais). O script força para Filial se houver apenas uma filial cadastrada.
- `mv_par15`: Indica se a seleção de tipos de título será personalizada (1) ou não.
- `mv_par16`: Natureza inicial para filtro.
- `mv_par17`: Natureza final para filtro.
- `mv_par18`: Define a data de conversão para valores em moeda estrangeira: 1 para Data de Emissão, 2 para Data Base do Sistema, 3 para Data de Vencimento Real, Outros para Data Final dos Parâmetros (`mv_par02`).
- `mv_par19`: Indica se lista baixas estornadas (1 para Sim, 2 para Não).

Tabelas Acessadas

O script realiza operações de leitura e busca nas seguintes tabelas do Protheus:

- **SE1**: Cadastro de Títulos a Receber. Utilizada para selecionar títulos a receber dentro dos critérios informados.
- **SE2**: Cadastro de Títulos a Pagar. Utilizada para selecionar títulos a pagar dentro dos critérios informados.
- **SE5**: Movimentações Bancárias. Utilizada para selecionar as baixas de títulos e outras movimentações financeiras dentro dos critérios informados.
- **SM0**: Cadastro de Empresas/Filiais. Utilizada para verificar o número de filiais.
- **SX1**: Cadastro de Perguntas (Gets). Utilizada para obter as descrições das variáveis de parâmetro (`MV_PARs`).
- **SX5**: Tabelas Genéricas. Utilizada para verificar e incluir o tipo de moeda "R$".
- **SA1**: Cadastro de Clientes. Acessada para obter o nome do cliente ao listar títulos a receber.
- **SA2**: Cadastro de Fornecedores. Acessada para obter o nome do fornecedor ao listar títulos a pagar.

- **SA6**: Cadastro de Bancos. Utilizada para obter a moeda do banco nas movimentações bancárias em ambientes que não sejam o Brasil.
-

Lógica de Seleção e Processamento

A seleção de registros na função `FA510Imp` ocorre primariamente através da aplicação de filtros nas tabelas SE1, SE2 e SE5, utilizando índices e condições baseadas nos `MV_PARs`.

Para Títulos a Receber (SE1) e Títulos a Pagar (SE2), são aplicados filtros por data de emissão, código do cliente/fornecedor, prefixo, natureza, e tipo de título. Títulos específicos, como aqueles originados de rotinas de transferência ou faturamento específico, são excluídos.

Para Movimentações Bancárias (SE5), os filtros incluem data de digitação, código do cliente/fornecedor, prefixo, natureza, tipo de documento e situação. Baixas por motivo de faturamento e transferências de devedor são explicitamente filtradas. O script também verifica e filtra movimentações de adiantamento e estornos com base nos parâmetros de tipo de movimentação (`mv_par11`) e lista de baixas estornadas (`mv_par19`).

Os registros que atendem aos critérios de seleção são então gravados no arquivo de trabalho temporário (`cNomeArq`) pela função `GravaTrab`. Esta função também trata a conversão de valores em moeda estrangeira com base no `mv_par18` e determina se o lançamento é a débito ou crédito.

Estrutura do Arquivo de Trabalho Temporário

O arquivo de trabalho temporário (`cNomeArq`), criado pela função `CriaTrab`, possui a seguinte estrutura de campos:

- `CODIGO`: Caractere, armazena o código do cliente ou fornecedor.
- `LOJA`: Caractere, armazena o código da loja.
- `DATAX`: Data, armazena a data chave para ordenação e quebra no relatório (geralmente data de emissão ou digitação).
- `NUMERO`: Caractere, armazena o número completo do título (Prefixo+Número+Parcela).
- `EMISSAO`: Data, armazena a data de emissão do título.
- `VENCREA`: Data, armazena a data de vencimento real do título.
- `VALOR`: Numérico, armazena o valor do título ou movimentação, possivelmente convertido.
- `HISTOR`: Caractere, armazena o histórico do lançamento.
- `DC`: Caractere, indica se o lançamento é 'D' (Débito) ou 'C' (Crédito).

O arquivo temporário é indexado por `Dtos(DATAX)+Numero+Codigo+DC` para ordenação do relatório.

Lógica de Impressão

A impressão do relatório é controlada na função `FA510Imp` após a população do arquivo de trabalho. O script itera sobre os registros no arquivo temporário (`cNomeArq`), gerando o cabeçalho, imprimindo cada registro e calculando os subtotais por data através da chamada à função `SubTot510`. Ao final, imprime os totais gerais e o rodapé. A formatação da impressão, incluindo o número de linhas por página e a compressão, é definida por parâmetros e variáveis internas.

Perguntas Comuns e Respostas

Aqui estão algumas perguntas frequentes que usuários podem ter sobre o relatório auxiliar gerado pelo script LUFR001 (FINR510), com respostas baseadas na análise do código:

- **Não consigo visualizar títulos com data de emissão fora do período que selecionei. É um erro?** Não é um erro. O relatório filtra os títulos a receber (SE1) e a pagar (SE2) estritamente pelas datas informadas nos parâmetros "A partir de" (`mv_par01`) e "Até a data" (`mv_par02`) para o campo de data de emissão (`E1_EMISSAO` e `E2_EMIS1`).
- **Por que algumas baixas não aparecem no relatório, mesmo estando dentro do período de digitação?** O relatório de movimentação bancária (SE5) aplica diversos filtros. Além da data de digitação (`E5_DTDIGIT`) estar dentro do período, são verificados o código do cliente/fornecedor, prefixo, natureza, tipo de documento (`E5_TIPODOC`) e situação (`E5_SITUACA` diferente de "C" - cancelado). Baixas com motivo "FAT" (faturamento) e transferências de devedor com motivo "TRF" são explicitamente excluídas.
- **Selecionei para imprimir apenas valores originais, mas ainda vejo valores diferentes do principal do título. Por quê?** Mesmo selecionando para imprimir apenas valores originais (`mv_par10 == 2`), o script ainda considera o valor líquido (`nValliq`) da movimentação bancária (`E5_VALOR`) e subtrai os valores de juros, multa, correção monetária e desconto. Em cenários de moeda estrangeira, a conversão do valor líquido também é aplicada.
- **Gerei o relatório para Fornecedores, mas alguns títulos a receber aparecem. Por que isso acontece?** Quando o relatório é gerado para Fornecedores (`mv_par03 == 2`), o script seleciona registros da tabela SE2 (Títulos a Pagar) e da tabela SE5 (Movimentações Bancárias). Na tabela SE5, ele inclui movimentações de recebimento (`E5_RECPAG == "R"`) que se referem a adiantamentos de fornecedores ou estornos de baixas de adiantamentos de fornecedores, que, na lógica do relatório, são tratados de forma invertida em relação à carteira principal.
- **Selecionei "Adiantamentos" no filtro de tipo de título, mas o relatório parece trazer outros tipos. O que pode estar acontecendo?** O filtro por tipo de título (`mv_par11`) interage com o parâmetro `mv_par15` ("Seleciona Tipos?") e a lista de tipos obtida pela função `finRTipos` e armazenada na variável `cTipos`. Se o `mv_par15` não for 1 ou a variável `cTipos` estiver vazia, o filtro por tipo de título pode não funcionar como esperado, pois a função `Inside` retornará `.T.`

(verdadeiro) para qualquer tipo. Além disso, a lógica de inclusão de adiantamentos (`mv_par11 == 3`) e normais (`mv_par11 == 2`) na tabela SE5 considera as variáveis `MVRECANT` e `MVPAGANT`.

- **O relatório está listando por filial, mas eu selecionei para listar por empresa. Como corrigir?** O script possui uma regra que força a listagem por filial (`mv_par14 = 1`) caso o sistema possua apenas uma filial cadastrada na tabela SM0 (`SM0->(Reccount())==1`), independentemente da opção selecionada pelo usuário. Verifique o cadastro de filiais no seu ambiente.
- **O que significa a opção "Converte por"?** Esta opção (`mv_par18`) define a data que será utilizada como referência para a conversão de valores de títulos ou movimentações que estejam em moeda diferente da primeira moeda do sistema. As opções incluem Data de Emissão, Data Base do Sistema, Data de Vencimento Real ou a Data Final informada nos parâmetros do relatório.

Script do relatório

Parte 1

```
// Programa: LUFR001
// Autor: Rafael Bonoldi
// Data: 22.03.05
// Descrição: Relatório Diário Auxiliar de Clientes / Fornecedores
// Este programa gera um relatório auxiliar detalhando movimentações financeiras de clientes ou fornecedores dentro de um período especificado.
// Permite visualizar valores financeiros ou apenas valores originais.
// A customização LUFT_LOG é chamada para registrar a utilização do programa.

#INCLUDE "FINR510.CH" // Arquivo de definição para rotinas financeiras
#Include "PROTHEUS.Ch" // Arquivo de definição geral do Protheus

User Function lufr001() // Ponto de entrada principal para a execução da função

    // Definição e inicialização de variáveis locais e privadas
    // Estas variáveis armazenam textos descritivos, configurações de tamanho, nomes de arquivos e parâmetros de controle.
    LOCAL cDesc1 := OemToAnsi(STR0001)  // Descrição 1: Propósito do programa (Diário Auxiliar Clientes/Fornecedores)
    LOCAL cDesc2 := OemToAnsi(STR0002)  // Descrição 2: Opções de impressão (valores financeiros/originais)
    LOCAL cDesc3 := OemToAnsi(STR0003)  // Descrição 3: Complemento da descrição 2
    LOCAL wnrel                         // Nome do arquivo de relatório em disco
    LOCAL aTam          := TAMSX3("E1_CLIENTE") // Tamanho do campo E1_CLIENTE na tabela SX3
    // LOCAL limite := IIF(aTam[1] > 6 ,220,132) // Definição de limite de linha baseado no tamanho do campo cliente (comentado)
```

```
   LOCAL Tamanho:=IIF(aTam[1] > 6 ,"G","M") // Tamanho da impressão (Grande/Médio) baseado no tamanho do campo cliente
   LOCAL cString:="SE1" // String de controle para a rotina SETPRINT

   // Declaração de variáveis privadas
   PRIVATE titulo := OemToAnsi(STR0004)  // Título do relatório (Diário Auxiliar de Clientes/Fornecedores)
   PRIVATE cabec1 // Cabeçalho da primeira linha do relatório
   PRIVATE cabec2 // Cabeçalho da segunda linha do relatório
   PRIVATE aReturn := { OemToAnsi(STR0005), 1,OemToAnsi(STR0006), 2, 2, 1, "",1 }  // Array de retorno para configuração de impressão (Zebrado, Administracao, etc.)
   PRIVATE nomeprog:= "FINR510" // Nome do programa associado
   PRIVATE aLinha  := { },nLastKey := 0 // Array de linhas e última tecla pressionada
   PRIVATE cPerg       := "FIN510" // Código da rotina de perguntas no SX1
   PRIVATE cTipos  := "" // Tipos de títulos selecionados para o relatório

   // Verifica e executa a função de log customizada, se existir
   // Esta seção é responsável por registrar a utilização desta customização para fins de rastreamento.
   IF FindFunction("U_LUFT_LOG")
      U_LUFT_LOG() // Chama a função U_LUFT_LOG para gerar o log de utilização
   Endif

   // Carrega as perguntas e parâmetros da rotina FIN510
   // A rotina 'pergunte' exibe a tela de parâmetros para o usuário.
   pergunte("FIN510",.F.)

   // Variáveis de sistema (MV_PARxx) utilizadas como parâmetros
   // A rotina 'pergunte' armazena as respostas do usuário nessas variáveis.
   // mv_par01: Data de início do período
   // mv_par02: Data de fim do período
   // mv_par03: Tipo (1 - Cliente, 2 - Fornecedor)
   // mv_par04: Código inicial do cliente/fornecedor
   // mv_par05: Código final do cliente/fornecedor
   // mv_par06: Página inicial para impressão
   // mv_par07: Página final para impressão
   // mv_par08: Impressão de termos (Abertura, Encerramento, Nenhum)
   // mv_par09: Imprime apenas os termos
   // mv_par10: Imprime valores financeiros (Sim/Não)
   // mv_par11: Tipos de títulos a imprimir (Todos, Normais, Adiantamentos)
   // mv_par12: Prefixo inicial do título
   // mv_par13: Prefixo final do título
   // mv_par14: Listar por (1 - Filial, 2 - Empresa)
   // mv_par15: Seleciona tipos específicos de título
   // mv_par16: Natureza inicial
   // mv_par17: Natureza final
   // mv_par18: Data para conversão de moeda
   // mv_par19: Lista baixas estornadas
```

```
    // Configurações iniciais para a impressão do relatório
    // A rotina 'SetPrint' configura o ambiente de impressão com base nos parâmetros.
    wnrel := "FINR510"            // Nome padrão do arquivo de relatório em disco
    wnrel :=
SetPrint(cString,wnrel,cPerg,@titulo,cDesc1,cDesc2,cDesc3,.F.,"",.T.,Tamanho,"",.F.)
    If nLastKey == 27 // Verifica se o usuário cancelou a operação na tela de parâmetros
        Return // Sai da função
    Endif

    // Define configurações padrão para a impressão
    // A rotina 'SetDefault' aplica configurações como zebrado, administração, etc.
    SetDefault(aReturn,cString)
    If nLastKey == 27 // Verifica se o usuário cancelou a operação na tela de configuração de
impressão
        Return // Sai da função
    Endif

    // Executa a rotina de impressão do relatório
    // A rotina 'RptStatus' exibe uma barra de status durante a execução do relatório.
    RptStatus({|lEnd| Fa510Imp(@lEnd,wnRel,cString)},Titulo) // Chama a função FA510Imp
para gerar o relatório

Return // Fim da função lufr001
```

Parte 2

```
Static Function FA510Imp(lEnd,wnRel,cString)
    // Descrição: Função principal de impressão do relatório Diário Auxiliar de Clientes /
Fornecedores.
    // Esta função coordena a seleção de dados, gravação em arquivo temporário e a
impressão do relatório.
    // Parâmetros:
    // lEnd: Variável lógica para indicar se o relatório foi cancelado pelo usuário.
    // wnRel: Nome do arquivo de relatório em disco.
    // cString: String de controle.

    LOCAL CbCont,CbTxt // Variáveis para controle de cabeçalho e rodapé
    LOCAL cNome,cIdent,aCampos:={},cNomeArq,nCompress,nC,nQuebra:=0 // Variáveis
para cabeçalhos, campos, nome do arquivo temporário, compressão, controle de quebra
    LOCAL nTotTit:=0,nTotDeb:=0,nTotCrd:=0 // Totais por título: Total de Títulos, Total Débito,
Total Crédito
    LOCAL nGerTit:=0,nGerDeb:=0,nGerCrd:=0 // Totais gerais: Total de Títulos, Total Débito,
Total Crédito
    // LOCAL dEmissao,dVencto,dDtDigit // Variáveis de data (comentadas)
    LOCAL nValliq // Valor líquido
    LOCAL nIndex // Índice
```

```
   LOCAL aInd:={} // Array de índices temporários
   LOCAL cCondE1:=cCondE2:=cCondE5:="" // Condições de filtro para as tabelas SE1,
SE2, SE5
   LOCAL cIndE1 :=cIndE2 :=cIndE5 :="" // Nomes de índice para as tabelas SE1, SE2, SE5
   LOCAL nTamNro:= TamSx3("E1_NUM")[1] // Tamanho do campo E1_NUM da tabela SX3
   // LOCAL nRegSe1Atu := SE1->(RecNo()) // Posição atual do registro na tabela SE1
(comentado) [cite: 8]
   // LOCAL nOrdSe1Atu := SE1->(IndexOrd()) // Ordem do índice atual da tabela SE1
(comentado) [cite: 8]
   // LOCAL nRegSe2Atu := SE2->(RecNo()) // Posição atual do registro na tabela SE2
(comentado) [cite: 8]
   // LOCAL nOrdSe2Atu := SE2->(IndexOrd()) // Ordem do índice atual da tabela SE2
(comentado) [cite: 8]
   // LOCAL cChaveSe1 // Chave da tabela SE1 (comentado) [cite: 8]
   // LOCAL cChaveSe2 // Chave da tabela SE2 (comentado) [cite: 8]
   Local aStru := SE1->(dbStruct()), ni // Estrutura da tabela SE1 e variável de iteração [cite:
8]
   Local aVariaveis := {}, aSavSet := {} // Arrays para variáveis e configurações de ambiente
   LOCAL aTam          := TAMSX3("E1_CLIENTE") // Tamanho do campo E1_CLIENTE da
tabela SX3
   LOCAL limite := IIF(aTam[1] > 6 ,220,132) // Limite de linha para impressão
   LOCAL Tamanho:= IIF(aTam[1] > 6 ,"G","M") // Tamanho da impressão (Grande/Médio)
   Local aColu  := {} // Array para posições das colunas na impressão
   // Local nMoedaBco          :=        1 // Moeda do banco (comentado)
   Local nMoedaT               :=        1 // Moeda do título
   Local cAliasTmp // Alias temporário para tabelas
   Local aDados := Array(9) // Array para armazenar dados do registro a ser gravado no
arquivo temporário
   Local nVlFin := 0 // Valor financeiro
   Local dDataConv // Data para conversão de moeda
   Local nX // Variável de iteração
   Local cSimb          :=        "" // Símbolos das moedas
   Local cSimbolo       :=        "" // Símbolo da moeda atual
   Local lTemBxCan := .F. // Indica se há baixa cancelada [cite: 9]
   Local i,n // Variáveis de iteração [cite: 9]
   Private nDecs     := GetMv("MV_CENT") // Número de casas decimais para valores
monetários

   // Verifica a existência e executa um bloco de código customizado para tipos de título
(FR510TIP)
   If ExistBlock("FR510TIP")
      cTipos   := ExecBlock("FR510TIP") // Executa o bloco e armazena os tipos selecionados
   Endif

   // Obtém os símbolos das moedas utilizadas no sistema, exceto para o Brasil
   // Estes símbolos serão usados para filtrar tipos de título que representam moedas.
   If cPaisLoc != "BRA"
      For nX := 1 to MoedFin() // Itera sobre as moedas configuradas
```

```
            cSimbolo := GetMv("MV_SIMB"+Str(nX,1)) // Obtém o símbolo da moeda
            If !Empty(cSimbolo) // Verifica se o símbolo não está vazio
               cSimb    += PadR(cSimbolo,3)+"," // Adiciona o símbolo à string de símbolos
            EndIf
            cSimbolo    := "" // Limpa a variável para a próxima iteração
         Next nX
         cSimb  := Substr(cSimb,1,len(cSimb)-1)  // Remove a última vírgula da string de
símbolos
      EndIf // cSimb agora contém os símbolos das moedas separadas por vírgula (ex:
"R$,US$,€")

      // Definição de variáveis para controle de linhas e colunas na impressão
      // cbtxt: Espaço em branco para controle de tabulação.
      // cbcont: Contador de linhas para o rodapé.
      // li: Linha atual de impressão.
      cbtxt          := SPACE(10) [cite: 10]
      cbcont         := 0 [cite: 10]
      li             := 80 [cite: 10]

      // Impressão dos termos (Abertura / Encerramento)
      // Verifica se a opção de imprimir termos (mv_par08) não é 'Nenhum' (3).
      If mv_par08 != 3
         lImprimiu := .F. // Flag para indicar se algum termo foi impresso
         cArqAbert:=SubStr(cArqRel,1,7)+"A.TRM" // Nome do arquivo para o termo de abertura
         cArqEncer:=SubStr(cArqRel,1,7)+"E.TRM" // Nome do arquivo para o termo de
encerramento
         aVariaveis := {} // Reinicia o array de variáveis para os termos

         // Coleta variáveis do cabeçalho da empresa (SM0)
         dbSelectArea("SM0") // Seleciona a tabela SM0 (Empresas)
         For i:=1 to FCount() // Itera sobre os campos da tabela SM0
            If FieldName(i)=="M0_CGC" // Se o campo for o CNPJ/CPF
               AADD(aVariaveis,{FieldName(i),Transform(FieldGet(i),"@R
99.999.999/9999-99")}) // Adiciona o campo formatado
            Else
               If FieldName(i)=="M0_NOME" // Se o campo for o nome da empresa, ignora (já é
tratado no título)
                  Loop // Pula para a próxima iteração
               Endif
               AADD(aVariaveis,{FieldName(i),FieldGet(i)}) // Adiciona outros campos
            Endif
         Next // Fim da iteração sobre os campos da SM0

         // Coleta variáveis das perguntas do SX1 para os termos
         SX1->(dbSeek("FIN51001")) // Posiciona na tabela SX1 (Perguntas)
         While SX1->X1_GRUPO=="FIN510" // Itera enquanto o grupo da pergunta for FIN510
            If !empty(SX1->X1_VAR01) // Verifica se o campo de variável não está vazio
```

```
AADD(aVariaveis,{Rtrim(Upper(SX1->X1_VAR01)),&("MV_PAR"+SX1->X1_ORDEM)}) //
Adiciona a variável do SX1 e seu valor correspondente do MV_PAR
            Endif
            SX1->(dbSkip()) // Avança para o próximo registro no SX1
        EndDo // Fim da iteração sobre as perguntas do SX1

        // Impressão do Termo de Abertura ou Encerramento
        If mv_par08 == 1// Opção: Termo de Abertura
            aSavSet:=__SetSets() // Salva as configurações de ambiente atuais
            cArqAbert:=CFGX024(cArqAbert,STR0004) // Prepara o arquivo de termo de
abertura [cite: 10]
            __SetSets(aSavSet) // Restaura as configurações de ambiente
            Set(24,Set(24),.t.) // Configura o SET PRINT para imprimir no arquivo
            lImprimiu := ImpTerm(cArqAbert,aVariaveis,AvalImp(Limite)) // Imprime o termo de
abertura
        ElseIf mv_par08 == 2     // Opção: Termo de Encerramento
            aSavSet:=__SetSets() // Salva as configurações de ambiente atuais
            cArqEncer:=CFGX024(cArqEncer,STR0004)  // Prepara o arquivo de termo de
encerramento [cite: 10]
            __SetSets(aSavSet) // Restaura as configurações de ambiente
            Set(24,Set(24),.t.) // Configura o SET PRINT para imprimir no arquivo
            lImprimiu := ImpTerm(cArqEncer,aVariaveis,AvalImp(Limite)) // Imprime o termo de
encerramento
        Endif

        // Ejeta a página se um termo foi impresso
        If lImprimiu
            Eject // Avança para a próxima página na impressora
        Endif

        // Verifica se a opção é imprimir apenas os termos
        If mv_par09 == 1// Opção: Somente o termo
            Set Device To Screen // Restaura a saída do dispositivo para a tela
            If lImprimiu // Se um termo foi impresso
                If aReturn[5] = 1 // Verifica a configuração de spool
                    Set Printer To // Restaura a impressora padrão
                    Commit // Confirma as operações
                    ourspool(wnrel) // Envia o arquivo para o spool de impressão
                    MS_FLUSH() // Libera o buffer de memória
                Endif
            Endif
            Return // Sai da função
        Endif
    Endif // Fim da seção de impressão de termos

    // Ajusta a tabela SX5 e seleciona tipos de título, se aplicável
    // Chama a função AjustaSx5 para incluir o tipo de moeda "R$".
```

```
   // Se o parâmetro mv_par15 estiver marcado, chama a rotina finRTipos para selecionar os
tipos de título.
   AjustaSx5() // Garante que o tipo de moeda "R$" exista na tabela SX5
   If mv_par15 == 1 // Verifica se a seleção de tipos está ativa
      finRTipos() // Chama a rotina para seleção de tipos de título
   Endif

   // Força a listagem por filial se houver apenas uma filial configurada
   // Se a tabela SM0 (Empresas) contiver apenas um registro, o relatório será sempre por
filial. [cite: 11]
   mv_par14 := Iif(SM0->(Reccount())==1,1,mv_par14) // Ajusta mv_par14 (Listar por) se
houver apenas uma filial [cite: 11]

   // Definição dos cabeçalhos do relatório
   // O título principal e os cabeçalhos das colunas são definidos aqui.
   titulo:= OemToAnsi(STR0007)
+IIF(mv_par03==1,OemToAnsi(STR0008),OemToAnsi(STR0009))  // Define o título com
base no tipo (Clientes/Fornecedores) [cite: 12]

   m_pag := mv_par06 // Inicializa o contador de páginas com o valor do parâmetro
mv_par06

   IF mv_par03==1 // Se for Cliente
      cNome :=OemToAnsi(STR0010)  //"NOME DO CLIENTE             " [cite: 12]
      cIdent:=OemToAnsi(STR0011)  //"CLIENTE" [cite: 12]
   Else // Se for Fornecedor
      cNome :=OemToAnsi(STR0012)  //"NOME DO FORNECEDOR                    " [cite: 12]
      cIdent:=OemToAnsi(STR0013)  //"FORNEC " [cite: 12]
   Endif

   nCompress:=aReturn[4] // Nível de compressão da impressão
   cabec1 :=
OemToAnsi(STR0014)+IIf(aTam[1]>6,Space(14),"")+cNome+OemToAnsi(STR0015)   //
Define a primeira linha do cabeçalho [cite: 12]
   cabec2 := Space(32)+cIdent+IIf(aTam[1]>6,Space(47),Space(33))+OemToAnsi(STR0032)
// Define a segunda linha do cabeçalho [cite: 12]

   // Criação do arquivo de trabalho temporário
   // Um arquivo temporário é criado para armazenar os dados a serem impressos no
relatório.
   aTam:=TamSX3("E1_CLIENTE") // Obtém o tamanho do campo E1_CLIENTE [cite: 12]
   AADD(aCampos,{"CODIGO"  ,"C",aTam[1],aTam[2]}) // Adiciona campo CODIGO
   aTam:=TamSX3("E1_LOJA") // Obtém o tamanho do campo E1_LOJA [cite: 12]
   AADD(aCampos,{"LOJA"    ,"C",aTam[1],aTam[2]}) // Adiciona campo LOJA
   aTam:=TamSX3("E1_EMISSAO") // Obtém o tamanho do campo E1_EMISSAO [cite: 12]
   AADD(aCampos,{"DATAX"    ,"D",aTam[1],aTam[2]}) // Adiciona campo DATAX (Data de
referência)
```

```
   AADD(aCampos,{"NUMERO"  ,"C",16,0}) // Adiciona campo NUMERO
(Prefixo+Número+Parcela)
   aTam:=TamSX3("E1_EMISSAO") // Obtém o tamanho do campo E1_EMISSAO [cite: 12]
   AADD(aCampos,{"EMISSAO" ,"D",aTam[1],aTam[2]}) // Adiciona campo EMISSAO (Data
de emissão)
   aTam:=TamSX3("E1_VENCREA") // Obtém o tamanho do campo E1_VENCREA [cite: 12]
   AADD(aCampos,{"VENCREA","D",aTam[1],aTam[2]}) // Adiciona campo VENCREA (Data
de vencimento real)
   aTam:=TamSX3("E1_VLCRUZ") // Obtém o tamanho do campo E1_VLCRUZ [cite: 12]
   AADD(aCampos,{"VALOR"   ,"N",aTam[1],aTam[2]}) // Adiciona campo VALOR
   AADD(aCampos,{"HISTOR"  ,"C",40,0}) // Adiciona campo HISTOR (Histórico)
   AADD(aCampos,{"DC"      ,"C",1,0}) // Adiciona campo DC (Débito/Crédito)
   cNomeArq:=CriaTrab(aCampos) // Cria o arquivo de trabalho com a estrutura definida
   dbUseArea( .T.,, cNomeArq, "cNomeArq", if(.F. .OR. .F., !.F., NIL), .F. ) // Abre o arquivo de
trabalho [cite: 13]

IndRegua("cNomeArq",cNomeArq,"Dtos(DataX)+Numero+Codigo+DC",,,OemToAnsi(STR00
16))  // Cria índice para o arquivo de trabalho [cite: 13]

   // Localiza e grava títulos a receber (SE1) no arquivo de trabalho, dentro dos parâmetros
   // Esta seção processa os títulos da tabela SE1 (Contas a Receber).
   If mv_par03 == 1 // Verifica se o relatório é para Clientes
      dbSelectArea("SE1") // Seleciona a tabela SE1
      dbSetOrder(6) // Define a ordem do índice (provavelmente por data de emissão)
      #IFDEF TOP // Se estiver usando o banco TOPCONN
         If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400
            cCondE1:=".T." // Condição inicial para filtro na query [cite: 14]
            cQuery := "SELECT * FROM " + RetSqlName("SE1") + " WHERE" // Início da
query SQL [cite: 14]
            cIndE1       :=IndexKey() // Obtém a chave do índice atual [cite: 14]

            If mv_par14 = 1 // Se listar por Filial
               cQuery += " E1_FILIAL = '" + xFilial("SE1") + "' AND " // Adiciona filtro por filial
[cite: 14]
            else // Se listar por Empresa
               cQuery += " E1_FILIAL BETWEEN '  ' AND 'ZZ' AND" // Filtro por todas as filiais
[cite: 14]
               cIndE1 :=Right(cIndE1,Len(cIndE1)-10) // Ajusta a chave do índice para
desconsiderar a filial [cite: 14]
            Endif
            cIndE1 := SqlOrder(cIndE1) // Converte a chave do índice para a sintaxe SQL
[cite: 14]

            dbSelectArea("SE1") // Seleciona a tabela SE1 [cite: 14]
            dbCloseArea() // Fecha a área da tabela SE1 [cite: 14]
            dbSelectArea("SA1") // Seleciona a tabela SA1 (Clientes) [cite: 14]

            // Adiciona as condições de filtro à query SQL
```

```
            cQuery += " E1_EMISSAO BETWEEN '" + DTOS(mv_par01) + "' AND '" +
DTOS(mv_par02) + "'" [cite: 14]
            cQuery += " AND E1_CLIENTE BETWEEN '" + mv_par04 + "' AND '" + mv_par05
+ "'" [cite: 14]
            cQuery += " AND E1_EMISSAO <= '" + DTOS(dDataBase) + "'" [cite: 14]
            cQuery += " AND E1_TIPO NOT LIKE 'PR%'" [cite: 14]
            If cPaisLoc <> "BRA" // Filtra tipos de moeda se não for Brasil [cite: 14]
               cQuery += " AND E1_TIPO NOT IN ('"+cSimb+"')" [cite: 14]
            Endif
            cQuery += " AND E1_PREFIXO BETWEEN '" + mv_par12 + "' AND '" + mv_par13
+ "'" [cite: 15]
            cQuery += " AND E1_NATUREZ BETWEEN '" + mv_par16 + "' AND '" + mv_par17
+ "'" [cite: 15]
            cQuery += " AND E1_ORIGEM NOT IN ('FINA280','TMSA490','LUFA05') " // Exclui
títulos gerados por rotinas específicas [cite: 15]
            cQuery += " AND E1_PARCELA = ' ' " // Exclui títulos de transferência [cite: 15]
            cQuery += " AND D_E_L_E_T_ <> '*'" // Considera apenas registros não
deletados [cite: 15]
            cQuery += " ORDER BY " + cIndE1 // Ordena os resultados pela chave do índice
[cite: 15]

            cQuery := ChangeQuery(cQuery) // Permite customização da query
            dbUseArea(.T., "TOPCONN", TCGenQry(,,cQuery), 'SE1', .F., .T.) // Executa a
query e abre a área da tabela SE1 [cite: 15]

            // Ajusta a estrutura da tabela no ambiente TOPCONN
            For ni := 1 to Len(aStru) // Itera sobre a estrutura original da tabela SE1 [cite: 15]
               If aStru[ni,2] != 'C' // Se o tipo do campo não for caractere [cite: 15]
                  TCSetField('SE1', aStru[ni,1], aStru[ni,2],aStru[ni,3],aStru[ni,4]) // Configura o
campo no TOPCONN [cite: 15]
               Endif
            Next // Fim da iteração sobre a estrutura da tabela SE1
         else // Se o tipo de servidor for AS/400
      #ENDIF // Fim da compilação condicional para TOPCONN
         If mv_par14 = 1 // Se listar por Filial
            dbSeek(xFilial()+Dtos(mv_par01),.t.) // Posiciona no primeiro registro dentro da
filial e data inicial [cite: 16]
            cCondE1:="xFilial()==E1_FILIAL .and. E1_EMISSAO>=mv_par01 .and.
E1_EMISSAO<=mv_par02" // Condição de filtro [cite: 16]
         Else // Se listar por Empresa
            cArqTrab :=CriaTrab(NIL,.F.) // Cria arquivo temporário para índice [cite: 16]
            AADD(aInd,cArqTrab) // Adiciona o nome do arquivo temporário ao array de
índices [cite: 16]
            cIndE1       :=IndexKey() // Obtém a chave do índice atual [cite: 16]
            cIndE1       :=Right(cIndE1,Len(cIndE1)-10) // Ajusta a chave do índice [cite: 16]
            IndRegua("SE1",cArqTrab,cIndE1,,cCondE1,OemToAnsi(STR0016))  // Cria índice
temporário com a condição [cite: 16]
```

```
            cCondE1:="E1_EMISSAO>=mv_par01 .and. E1_EMISSAO<=mv_par02" //
Condição de filtro principal [cite: 16]
            dbCommit() // Confirma as operações [cite: 16]
            nIndex:=RetIndex("SE1") // Obtém o número do índice atual [cite: 16]
            dbSelectArea("SE1") // Seleciona a tabela SE1 [cite: 16]
            #IFNDEF TOP // Se não estiver usando TOPCONN
               dbSetIndex(cArqTrab+OrdBagExt()) // Define o índice temporário [cite: 16]
            #ENDIF // Fim da compilação condicional
            dbSetOrder(nIndex+1) // Define a ordem de acesso usando o índice temporário
[cite: 16]
            dbSeek(Dtos(mv_par01),.t.) // Posiciona no primeiro registro com data de emissão
maior ou igual à data inicial [cite: 16]
         Endif
         #IFDEF TOP // Se estiver usando TOPCONN
         Endif
         #ENDIF // Fim da compilação condicional para TOPCONN

      dbSelectArea("SE1") // Garante que a área da SE1 está selecionada

      // Loop principal para processar os registros da tabela SE1
      While !Eof() .and. &(cCondE1) // Itera enquanto não for o fim do arquivo e a condição
for verdadeira

         // Verifica se o título foi transferido de Devedor
         // Esta subseção verifica na tabela SE6 (Transferência de Devedor) se o título atual
foi transferido. [cite: 17]
         // Joao Mattos - 03/03/2006. [cite: 17]
         aArea := GetArea() // Salva a área de trabalho atual [cite: 17]
         lTransferido := .f. // Flag para indicar se foi transferido [cite: 17]
         cQuery := " SELECT * FROM " + RetSqlName("SE6") + " WHERE" // Início da query
SQL para SE6 [cite: 17]
         cQuery += " E6_FILIAL  = '" + xFilial("SE6")  + "' AND " // Filtra por filial [cite: 17]
         cQuery += " E6_PREFIXO = '" + SE1->E1_PREFIXO + "' AND " // Filtra por prefixo
[cite: 17]
         cQuery += " E6_NUM     = '" + SE1->E1_NUM     + "' AND " // Filtra por número [cite:
17]
         cQuery += " E6_CLIENTE = '" + SE1->E1_CLIENTE + "' AND " // Filtra por cliente
[cite: 17]
         cQuery += " E6_LOJA    = '" + SE1->E1_LOJA    + "' AND " // Filtra por loja [cite: 17]
         cQuery += " E6_SITSOL  = '2' AND " // Filtra por situação 'Confirmada' [cite: 17]
         cQuery += " E6_DATSOL  <= '" + Dtos(SE1->E1_EMISSAO) + "' AND " // Filtra pela
data da solicitação [cite: 17]
         cQuery += " D_E_L_E_T_ <> '*'" // Considera apenas registros não deletados [cite:
17]
         cQuery := ChangeQuery(cQuery) // Permite customização da query [cite: 17]
         dbUseArea(.T., "TOPCONN", TCGenQry(,,cQuery), 'TRABA', .F., .T.) // Executa a
query e abre área temporária 'TRABA' [cite: 17]
         DbSelectArea("TRABA") // Seleciona a área temporária [cite: 17]
```

```
        DbGoTop() // Posiciona no primeiro registro [cite: 17]
        lTransferido := !Eof() // Se encontrou registros, marca como transferido [cite: 17]
        DbSelectArea("TRABA") // Seleciona a área temporária novamente [cite: 17]
        DbCloseArea() // Fecha a área temporária [cite: 17]
        RestArea(aArea) // Restaura a área de trabalho anterior [cite: 17]
        If lTransferido // Se o título foi transferido
           dbskip() // Pula para o próximo registro na SE1
           Loop // Reinicia o loop
        Endif
        // Fim do filtro de títulos por transferência de devedor

        // Filtros adicionais para títulos a receber (SE1)
        // Verifica se o cliente, data de emissão, tipo ou prefixo estão fora dos parâmetros.
        IF SE1->E1_CLIENTE < mv_par04 .or. SE1->E1_CLIENTE > mv_par05 .or.
SE1->E1_EMISSAO > dDataBase [cite: 18]
           dbskip() // Pula o registro se fora do intervalo de cliente ou data base [cite: 18]
           Loop // Reinicia o loop [cite: 18]
        Endif
        IF SE1->E1_TIPO $ MVPROVIS // Verifica se é um título provisório
           dbskip() // Pula se for provisório
           Loop // Reinicia o loop
        Endif
        If mv_par11 == 2 .and. E1_TIPO $ MVRECANT // Se a opção é 'Normais' e o tipo é
adiantamento [cite: 19]
           dbskip() // Pula se for adiantamento [cite: 19]
           Loop // Reinicia o loop [cite: 19]
        Endif

        If mv_par11 == 3 .and. !(E1_TIPO $ MVRECANT) // Se a opção é 'Adiantamentos' e
o tipo não é adiantamento [cite: 19]
           dbskip() // Pula se não for adiantamento [cite: 19]
           Loop // Reinicia o loop [cite: 19]
        Endif
        If cPaisLoc <> "BRA" .And. IsMoney(E1_TIPO) // Se não for Brasil e o tipo for moeda
[cite: 20]
           dbskip() // Pula se for tipo moeda [cite: 20]
           Loop // Reinicia o loop [cite: 20]
        Endif
        If !Empty( mv_par12 ) // Se o prefixo inicial foi informado
           If SE1->E1_PREFIXO < mv_par12 // Verifica se o prefixo é menor que o inicial
              dbskip() // Pula se menor
              Loop // Reinicia o loop
           Endif
        Endif

        If !Empty( mv_par13 ) // Se o prefixo final foi informado
           If SE1->E1_PREFIXO > mv_par13 // Verifica se o prefixo é maior que o final
              dbskip() // Pula se maior
```

```
            Loop // Reinicia o loop
         Endif
      Endif

      If ( SE1->E1_NATUREZ < MV_PAR16 .Or. SE1->E1_NATUREZ > MV_PAR17 ) //
Verifica se a natureza está fora do intervalo [cite: 21]
         dbskip() // Pula se estiver fora do intervalo de natureza [cite: 21]
         Loop // Reinicia o loop [cite: 21]
      Endif

      If !Inside(SE1->E1_TIPO) // Verifica se o tipo do título está entre os tipos
selecionados [cite: 21]
         dbskip() // Pula se o tipo não estiver selecionado [cite: 21]
         Loop // Reinicia o loop [cite: 21]
      Endif

      // Define a data para conversão de moeda
      If mv_par18 == 1 // Converte pela data de emissão
         dDataConv := SE1->E1_EMISSAO
      Elseif mv_par18 == 2 // Converte pela data base
         dDataConv := dDataBase
      Elseif mv_par18 == 3 // Converte pela data de vencimento real
         dDataConv := SE1->E1_VENCREA
      Else // Converte pela data final do período
         dDataConv := mv_par02
      Endif

      // Grava débito no arquivo de trabalho temporário - Emissão
      // Preenche um registro no arquivo temporário com os dados do título a receber.
      Reclock("cNomeArq",.T.) // Bloqueia o registro no arquivo de trabalho [cite: 22]
      Replace CODIGO     With SE1->E1_CLIENTE // Código do cliente [cite: 22]
      Replace LOJA       With SE1->E1_LOJA // Loja do cliente [cite: 22]
      Replace DATAX            With SE1->E1_EMISSAO // Data de referência
(emissão) [cite: 22]
      Replace NUMERO     With
SE1->E1_PREFIXO+SE1->E1_NUM+SE1->E1_PARCELA // Número completo do título
[cite: 22]
      Replace EMISSAO    With SE1->E1_EMISSAO // Data de emissão [cite: 22]
      Replace VALOR            With
If(MV_PAR18==1,SE1->E1_VLCRUZ,xMoeda(SE1->E1_VALOR,SE1->E1_MOEDA,1,dData
Conv,,If(cPaisLoc=="BRA",SE1->E1_TXMOEDA,0))) // Valor convertido [cite: 22]
      Replace HISTOR     With Iif(Empty(SE1->E1_HIST),STR0031,SE1->E1_HIST) //
Histórico [cite: 22]
      Replace VENCREA    With SE1->E1_VENCREA // Data de vencimento real [cite: 22]

      IF !SE1->E1_TIPO $ MVABATIM .and. ! ( SE1->E1_TIPO $
MVRECANT+"/"+MV_CRNEG ) // Verifica se não é abatimento, adiantamento ou crédito
negativo [cite: 22]
```

```
            Replace DC                  With "D" // Se não for, marca como Débito [cite: 22]
         Else
            Replace DC                  With "C"    // Se for abatimento, RA ou NCC, marca
como Crédito [cite: 22]
         Endif
         MsUnlock() // Libera o bloqueio do registro [cite: 22]
         dbSelectArea("SE1") // Retorna para a área da SE1
         dbSkip() // Avança para o próximo registro na SE1
      Enddo // Fim do loop de processamento da tabela SE1

      #IFDEF TOP // Se estiver usando TOPCONN
         If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400
            DBSelectArea("SE1") // Seleciona a área da SE1 [cite: 22]
            DbCloseArea() // Fecha a área da SE1 [cite: 22]
            ChkFile("SE1") // Verifica a integridade do arquivo [cite: 22]
         Endif
      #ENDIF // Fim da compilação condicional para TOPCONN
   Endif // Fim da seção de processamento de títulos a receber (SE1)

   // Localiza e grava títulos a pagar (SE2) no arquivo de trabalho, dentro dos parâmetros
   // Esta seção processa os títulos da tabela SE2 (Contas a Pagar).
   if mv_par03 == 2 // Verifica se o relatório é para Fornecedores
      aStru := SE2->(dbStruct()) // Obtém a estrutura da tabela SE2 [cite: 23]
      dbSelectArea("SE2") // Seleciona a tabela SE2 [cite: 23]
      #IFDEF TOP // Se estiver usando TOPCONN [cite: 23]
         If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400 [cite: 23]
            cCondE2:=".T." // Condição inicial para filtro na query [cite: 23]
            cQuery := "SELECT * FROM " + RetSqlName("SE2") + " WHERE" // Início da
query SQL [cite: 23]
            cIndE2       :=IndexKey() // Obtém a chave do índice atual [cite: 23]
            If mv_par14 = 1 // Se listar por Filial [cite: 23]
               cQuery += " E2_FILIAL = '" + xFilial("SE2") + "' AND " // Adiciona filtro por filial
[cite: 23]
            else // Se listar por Empresa [cite: 23]
               cQuery += " E2_FILIAL BETWEEN '  ' AND 'ZZ' AND" // Filtro por todas as filiais
[cite: 23]
               cIndE2 :=Right(cIndE2,Len(cIndE2)-10) // Ajusta a chave do índice [cite: 23]
            Endif
            cIndE2 := SqlOrder(cIndE2) // Converte a chave do índice para a sintaxe SQL
[cite: 23]

            dbSelectArea("SE2") // Seleciona a tabela SE2 [cite: 23]
            dbCloseArea() // Fecha a área da tabela SE2 [cite: 23]
            dbSelectArea("SA1") // Seleciona a tabela SA1 (Clientes) - Possível erro, deveria
ser SA2 (Fornecedores) [cite: 23]

            // Adiciona as condições de filtro à query SQL
```

```
            cQuery += " E2_EMIS1 BETWEEN '" + DTOS(mv_par01) + "' AND '" +
DTOS(mv_par02) + "'" [cite: 23]
            cQuery += " AND E2_FORNECE BETWEEN '" + mv_par04 + "' AND '" +
mv_par05 + "'" [cite: 23]
            cQuery += " AND E2_EMIS1 <= '" + DTOS(dDataBase) + "'" [cite: 23]
            cQuery += " AND E2_PREFIXO BETWEEN '" + mv_par12 + "' AND '" + mv_par13
+ "'" [cite: 24]
            cQuery += " AND E2_NATUREZ BETWEEN '" + mv_par16 + "' AND '" + mv_par17
+ "'" [cite: 24]
            cQuery += " AND D_E_L_E_T_ <> '*'" [cite: 24]
            cQuery += " ORDER BY " + cIndE2 [cite: 24]

            cQuery := ChangeQuery(cQuery) // Permite customização da query [cite: 24]
            dbUseArea(.T., "TOPCONN", TCGenQry(,,cQuery), 'SE2', .F., .T.) // Executa a
query e abre a área da tabela SE2 [cite: 24]
            Memowrite("C:\LUFR001.txt",cQuery) // Grava a query em um arquivo (para
debug) [cite: 24]

            // Ajusta a estrutura da tabela no ambiente TOPCONN
            For ni := 1 to Len(aStru) // Itera sobre a estrutura original da tabela SE2 [cite: 24]
               If aStru[ni,2] != 'C' // Se o tipo do campo não for caractere [cite: 24]
                  TCSetField('SE2', aStru[ni,1], aStru[ni,2],aStru[ni,3],aStru[ni,4]) // Configura o
campo no TOPCONN [cite: 24]
               Endif
            Next // Fim da iteração sobre a estrutura da tabela SE2
         Else // Se o tipo de servidor for AS/400 [cite: 24]
      #ENDIF // Fim da compilação condicional para TOPCONN [cite: 24]
         If mv_par14 = 1 // Se listar por Filial
            dbSetOrder(7) // Define a ordem do índice (provavelmente por filial+data de
emissão) [cite: 25]
            dbSeek(xFilial()+Dtos(mv_par01),.T.) // Posiciona no primeiro registro dentro da
filial e data inicial [cite: 25]
            cCondE2:="xFilial()==E2_FILIAL .and. E2_EMIS1>=mv_par01 .and.
E2_EMIS1<=mv_par02" // Condição de filtro [cite: 25]
         Else // Se listar por Empresa
            cArqTrab:=CriaTrab(NIL,.F.) // Cria arquivo temporário para índice [cite: 25]
            AADD(aInd,cArqTrab) // Adiciona o nome do arquivo temporário ao array de
índices [cite: 25]

cIndE2:="Dtos(E2_EMIS1)+E2_PREFIXO+E2_NUM+E2_PARCELA+E2_TIPO+E2_FORNE
CE" // Chave do índice temporário [cite: 25]
            IndRegua("SE2",cArqTrab,cIndE2,,,OemToAnsi(STR0016)) // Cria índice
temporário com a condição [cite: 25]
            cCondE2:="E2_EMIS1>=mv_par01 .and. E2_EMIS1<=mv_par02" // Condição de
filtro principal [cite: 26]
            dbCommit() // Confirma as operações [cite: 26]
            nIndex:=RetIndex("SE2") // Obtém o número do índice atual [cite: 26]
            dbSelectArea("SE2") // Seleciona a tabela SE2 [cite: 26]
```

```
            #IFNDEF TOP // Se não estiver usando TOPCONN [cite: 26]
               dbSetIndex(cArqTrab+OrdBagExt()) // Define o índice temporário [cite: 26]
            #ENDIF // Fim da compilação condicional [cite: 26]
            dbSetOrder(nIndex+1) // Define a ordem de acesso usando o índice temporário
[cite: 26]
            dbSeek(Dtos(mv_par01),.T.) // Posiciona no primeiro registro com data de
emissão maior ou igual à data inicial [cite: 26]
         Endif
         #IFDEF TOP // Se estiver usando TOPCONN [cite: 26]
         Endif
         #ENDIF // Fim da compilação condicional para TOPCONN [cite: 26]

      // Loop principal para processar os registros da tabela SE2
      While !Eof() .and. &(cCondE2) // Itera enquanto não for o fim do arquivo e a condição
for verdadeira [cite: 26]
         // Alterado Rafael Bonoldi
         // Esta condição não permite que sejam impressos títulos tipo fatura gerados pela
rotina FINA290.
         If "FINA290" == Alltrim(SE2->E2_ORIGEM) // Verifica se a origem é FINA290
            dbskip() // Pula o registro
            Loop // Reinicia o loop
         Endif

         // Filtros adicionais para títulos a pagar (SE2)
         // Verifica se o fornecedor, data de emissão, tipo ou prefixo estão fora dos
parâmetros.
         IF SE2->E2_FORNECE < mv_par04 .or. SE2->E2_FORNECE > mv_par05 .or.
SE2->E2_EMIS1 > dDataBase // Verifica intervalo de fornecedor e data base
            dbskip() // Pula o registro
            Loop // Reinicia o loop
         Endif

         If mv_par03 == 1 .or. SE2->E2_TIPO $ MVPROVIS // Se o relatório for de cliente ou
o tipo for provisório
            dbskip() // Pula o registro
            Loop // Reinicia o loop
         Endif

         If mv_par11 == 2 .and. (E2_TIPO $ MVPAGANT) // Se a opção é 'Normais' e o tipo é
adiantamento
            dbskip() // Pula se for adiantamento
            Loop // Reinicia o loop
         Endif

         If mv_par11 == 3 .and. ! ( E2_TIPO $ MVPAGANT) // Se a opção é 'Adiantamentos' e
o tipo não é adiantamento
            dbskip() // Pula se não for adiantamento
            Loop // Reinicia o loop
```

```
        Endif

        If !Empty( mv_par12 ) // Se o prefixo inicial foi informado
           If SE2->E2_PREFIXO < mv_par12 // Verifica se o prefixo é menor que o inicial
              dbskip() // Pula se menor
              Loop // Reinicia o loop
           Endif
        Endif

        If !Empty( mv_par13 ) // Se o prefixo final foi informado
           If SE2->E2_PREFIXO > mv_par13 // Verifica se o prefixo é maior que o final
              dbskip() // Pula se maior
              Loop // Reinicia o loop
           Endif
        Endif

        If ( SE2->E2_NATUREZ < MV_PAR16 .Or. SE2->E2_NATUREZ > MV_PAR17 ) //
Verifica se a natureza está fora do intervalo [cite: 27]
           dbskip() // Pula se estiver fora do intervalo de natureza [cite: 27]
           Loop // Reinicia o loop [cite: 27]
        Endif

        If !Inside(SE2->E2_TIPO) // Verifica se o tipo do título está entre os tipos
selecionados
           dbskip() // Pula se o tipo não estiver selecionado
           Loop // Reinicia o loop
        Endif

        // Define a data para conversão de moeda
        If mv_par18 == 1 // Converte pela data de emissão
           dDataConv := SE2->E2_EMIS1                    // emissao da contabilizacao
        Elseif mv_par18 == 2 // Converte pela data base
           dDataConv := dDataBase
        Elseif mv_par18 == 3 // Converte pela data de vencimento real
           dDataConv := SE2->E2_VENCREA
        Else // Converte pela data final do período
           dDataConv := mv_par02
        Endif

        // Grava crédito no arquivo de trabalho temporário - Emissão
        // Preenche um registro no arquivo temporário com os dados do título a pagar.
        Reclock("cNomeArq",.T.) // Bloqueia o registro no arquivo de trabalho [cite: 28]
        Replace CODIGO  With SE2->E2_FORNECE // Código do fornecedor [cite: 28]
        Replace LOJA With SE2->E2_LOJA // Loja do fornecedor [cite: 28]
        Replace DATAX         With SE2->E2_EMIS1 // Data de referência (emissão) [cite:
28]
        Replace NUMERO  With SE2->E2_PREFIXO+SE2->E2_NUM+SE2->E2_PARCELA
// Número completo do título [cite: 28]
```

```
        Replace EMISSAO With SE2->E2_EMISSAO // Data de emissão [cite: 28]
        Replace VALOR         With
If(MV_PAR18==1,SE2->E2_VLCRUZ,xMoeda(SE2->E2_VALOR,SE2->E2_MOEDA,1,dData
Conv,,If(cPaisLoc=="BRA",SE2->E2_TXMOEDA,0))) // Valor convertido [cite: 28]
        Replace HISTOR  With Iif(Empty(SE2->E2_HIST),STR0031,SE2->E2_HIST) //
Histórico [cite: 28]
        Replace VENCREA With SE2->E2_VENCREA // Data de vencimento real [cite: 28]

        If !(SE2->E2_TIPO $ MVABATIM) .and. ! ( SE2->E2_TIPO $
MVPAGANT+"/"+MV_CPNEG) // Verifica se não é abatimento, adiantamento ou crédito
negativo [cite: 28]
           Replace DC                With "C" // Se não for, marca como Crédito [cite: 28]
        Else
           Replace DC                With "D"    // Se for abatimento, PA ou CN, marca como
Débito [cite: 28]
        Endif
        MsUnlock() // Libera o bloqueio do registro [cite: 28]

        // Processa abatimentos que fizeram parte da fatura
        IF SE2->E2_TIPO $ MVABATIM .and. !Empty(SE2->E2_FATURA) .and. ; // Verifica
se é abatimento e tem número de fatura [cite: 29]
           SE2->E2_FATURA != "NOTFAT" .and. SE2->E2_DTFATUR <= mv_par02 //
Verifica se não é "NOTFAT" e a data da fatura está dentro do período [cite: 29]

           Reclock("cNomeArq",.T.) // Bloqueia o registro no arquivo de trabalho [cite: 29]
           Replace CODIGO  With SE2->E2_FORNECE // Código do fornecedor [cite: 29]
           Replace LOJA          With SE2->E2_LOJA // Loja do fornecedor [cite: 29]
           Replace DATAX         With SE2->E2_DTFATUR // Data de referência (data
da fatura) [cite: 29]
           Replace NUMERO        With
SE2->E2_PREFIXO+SE2->E2_NUM+SE2->E2_PARCELA // Número completo do título
[cite: 29]
           Replace EMISSAO With SE2->E2_DTFATUR // Data de emissão (data da fatura)
[cite: 29]
           Replace VALOR         With
If(MV_PAR18==1,SE2->E2_VLCRUZ,xMoeda(SE2->E2_VALOR,SE2->E2_MOEDA,1,dData
Conv)) // Valor convertido [cite: 29]
           Replace HISTOR   With STR0035 + SE2->E2_FATURA  //"BX EMIS FAT " +
número da fatura [cite: 29]
           Replace VENCREA       With SE2->E2_VENCREA // Data de vencimento real
[cite: 29]
           Replace DC            With "C" // Marca como Crédito [cite: 29]
           MsUnlock() // Libera o bloqueio do registro [cite: 29]
        Endif // Fim do processamento de abatimentos

        dbSelectArea("SE2") // Retorna para a área da SE2
        dbSkip() // Avança para o próximo registro na SE2
     Enddo // Fim do loop de processamento da tabela SE2
```

```
      #IFDEF TOP // Se estiver usando TOPCONN [cite: 29]
         If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400 [cite: 29]
            DBSelectArea("SE2") // Seleciona a área da SE2 [cite: 29]
            DbCloseArea() // Fecha a área da SE2 [cite: 29]
            ChkFile("SE2") // Verifica a integridade do arquivo [cite: 29]
         Endif
      #ENDIF // Fim da compilação condicional para TOPCONN [cite: 29]
   Endif // Fim da seção de processamento de títulos a pagar (SE2)

   // Localiza e grava movimentações bancárias (SE5) no arquivo de trabalho, dentro dos
parâmetros
   // Esta seção processa as movimentações da tabela SE5 (Movimentação Bancária).
   dbSelectArea("SE5") // Seleciona a tabela SE5
   dbSetOrder(4) // Define a ordem do índice (provavelmente por data de digitação)

   #IFDEF TOP // Se estiver usando o banco TOPCONN
      If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400
         cAliasTmp := "NEWSE5" // Define um alias temporário
         aStru := SE5->(dbStruct()) // Obtém a estrutura da tabela SE5 [cite: 30]
         cCondE5:=".T." // Condição inicial para filtro na query [cite: 30]
         cQuery := "SELECT * FROM " + RetSqlName("SE5") + " WHERE" // Início da query
SQL [cite: 30]
         cIndE5:=IndexKey() // Obtém a chave do índice atual [cite: 30]
         If mv_par14 = 1 // Se listar por Filial [cite: 30]
            cQuery += " E5_FILIAL = '" + xFilial("SE5") + "' AND " // Adiciona filtro por filial
[cite: 30]
         else // Se listar por Empresa [cite: 30]
            cQuery += " E5_FILIAL BETWEEN '  ' AND 'ZZ' AND" // Filtro por todas as filiais
[cite: 30]
            cIndE5 :=Right(cIndE5,Len(cIndE5)-10) // Ajusta a chave do índice [cite: 30]
         Endif
         cIndE5 := SqlOrder(cIndE5) // Converte a chave do índice para a sintaxe SQL [cite:
30]
         dbSelectArea("SA1") // Seleciona a tabela SA1 (Clientes) - Possível erro, deveria ser
SA2 (Fornecedores) em alguns casos [cite: 30]

         // Adiciona as condições de filtro à query SQL
         cQuery += " E5_DTDIGIT BETWEEN '" + DTOS(mv_par01) + "' AND '" +
DTOS(mv_par02) + "'" [cite: 30]
         cQuery += " AND E5_NUMERO <> '" +space(TamSX3("E5_NUMERO")[1])+"'" [cite:
30]
         cQuery += " AND E5_SITUACA <> 'C'" [cite: 30]
         cQuery += " AND E5_DTDIGIT <= '"+DTOS(dDataBase)+ "'" [cite: 30]
         cQuery += " AND E5_CLIFOR BETWEEN '" + mv_par04 + "' AND '" + mv_par05 + "'"
[cite: 30]
         cQuery += " AND E5_PREFIXO BETWEEN '" + mv_par12 + "' AND '" + mv_par13 +
"'" [cite: 30]
```

```
        cQuery += " AND E5_NATUREZ BETWEEN '" + mv_par16 + "' AND '" + mv_par17 +
"'" [cite: 30]
        cQuery += " AND E5_TIPODOC IN ('VL','VM','BA','CP','LJ','V2','ES','PA','RA')" // Filtra
por tipos de documento específicos [cite: 30]
        cQuery += " AND E5_MOTBX NOT IN ('TRF')" // Exclui movimentações com motivo
de baixa 'TRF' (Transferência de Devedor) - JOAO MATTOS 03/03/06 [cite: 31]
        cQuery += " AND D_E_L_E_T_ <> '*'" [cite: 31]
        If cPaisLoc <> "BRA" // Filtra tipos de moeda se não for Brasil [cite: 31]
           cQuery += " AND (E5_TIPO NOT IN ('"+cSimb+"') AND E5_TIPODOC <> 'LJ' ) "
[cite: 31]
        Endif
        If mv_par03 == 1 // Se for relatório de Clientes [cite: 31]
           cQuery += " AND ((E5_RECPAG = 'R')" // Inclui recebimentos [cite: 31]
           cQuery += " OR (E5_TIPODOC = 'ES' AND E5_RECPAG = 'P')" // Inclui estornos
de pagamentos [cite: 31]
           cQuery += " OR (E5_TIPO IN ('"+MV_CRNEG+"','"+MVRECANT+"') AND
E5_RECPAG = 'P'))" [cite: 31] // Inclui crédito negativo e adiantamento com Rec/Pag = P
[cite: 31]
        Endif
        If mv_par03 == 2 // Se for relatório de Fornecedores [cite: 32]
           cQuery += " AND ((E5_RECPAG = 'P')" // Inclui pagamentos [cite: 32]
           cQuery += " OR  (E5_TIPODOC = 'ES' AND E5_RECPAG = 'R')" // Inclui estornos
de recebimentos [cite: 32]
           cQuery += " OR  (E5_TIPO IN ('"+MV_CPNEG+"','"+MVPAGANT+"') AND
E5_RECPAG = 'R'))" [cite: 32] // Inclui débito negativo e adiantamento com Rec/Pag = R
[cite: 32]
        Endif

        cQuery += " ORDER BY " + cIndE5 [cite: 32] // Ordena os resultados pela chave do
índice [cite: 32]

        cQuery := ChangeQuery(cQuery) // Permite customização da query [cite: 32]
        dbUseArea(.T., "TOPCONN", TCGenQry(,,cQuery), cAliasTmp, .F., .T.) // Executa a
query e abre a área da tabela SE5 com alias temporário [cite: 32]

        // Ajusta a estrutura da tabela no ambiente TOPCONN
        For ni := 1 to Len(aStru) // Itera sobre a estrutura original da tabela SE5 [cite: 32]
           If aStru[ni,2] != 'C' // Se o tipo do campo não for caractere [cite: 32]
              TCSetField(cAliasTmp, aStru[ni,1], aStru[ni,2],aStru[ni,3],aStru[ni,4]) //
Configura o campo no TOPCONN [cite: 32]
           Endif
        Next // Fim da iteração sobre a estrutura da tabela SE5
     Else // Se o tipo de servidor for AS/400 [cite: 32]
  #ENDIF // Fim da compilação condicional para TOPCONN [cite: 32]
     cAliasTmp := "SE5" // Define o alias temporário como "SE5"
     If mv_par14 = 1 // Se listar por Filial
        dbSeek(xFilial(),.T.) // Posiciona no primeiro registro da filial
        cCondE5:="xFilial()==E5_FILIAL" // Condição de filtro por filial
```

```
      Else // Se listar por Empresa
         cArqTrab :=CriaTrab(NIL,.F.) // Cria arquivo temporário para índice [cite: 32]
         AADD(aInd,cArqTrab) // Adiciona o nome do arquivo temporário ao array de índices
[cite: 32]
         cIndE5:=IndexKey() // Obtém a chave do índice atual [cite: 32]
         cIndE5:=Right(cIndE5,Len(cIndE5)-10) // Ajusta a chave do índice [cite: 32]
         IndRegua("SE5",cArqTrab,cIndE5,,,OemToAnsi(STR0016))  // Cria índice temporário
com a condição [cite: 32]
         cCondE5:=".T." // Condição de filtro inicial [cite: 32]
         dbCommit() // Confirma as operações [cite: 32]
         nIndex:=RetIndex("SE5") // Obtém o número do índice atual [cite: 32]
         dbSelectArea("SE5") // Seleciona a tabela SE5 [cite: 32]
         #IFNDEF TOP // Se não estiver usando TOPCONN [cite: 32]
            dbSetIndex(cArqTrab+ordBagExt()) // Define o índice temporário [cite: 32]
         #ENDIF // Fim da compilação condicional [cite: 32]
         dbSetOrder(nIndex+1) // Define a ordem de acesso usando o índice temporário [cite:
32]
         dbGoTop() // Posiciona no início do arquivo [cite: 32]
      Endif

      #IFDEF TOP // Se estiver usando TOPCONN [cite: 32]
      Endif
      #ENDIF // Fim da compilação condicional para TOPCONN [cite: 32]

   // Cria índices temporários adicionais se listar por Empresa (mv_par14 == 2)
   If mv_par14 == 2
      If mv_par03 == 1 // Se for relatório de Clientes (SE1)
         // SE1
         dbSelectArea("SE1") // Seleciona a tabela SE1
         dbSetOrder(1) // Define a ordem do índice [cite: 32]
         cArqTrab :=CriaTrab(NIL,.F.) // Cria arquivo temporário para índice [cite: 32]
         AADD(aInd,cArqTrab) // Adiciona o nome do arquivo temporário ao array de índices
[cite: 32]
         cIndE1:=IndexKey() // Obtém a chave do índice atual [cite: 32]
         cIndE1:=Right(cIndE1,Len(cIndE1)-10) // Ajusta a chave do índice [cite: 32]
         IndRegua("SE1",cArqTrab,cIndE1,,,OemToAnsi(STR0016)) // Cria índice temporário
[cite: 32]
         dbCommit() // Confirma as operações [cite: 32]
         nIndex:=RetIndex("SE1") // Obtém o número do índice atual [cite: 32]
         dbSelectArea("SE1") // Seleciona a tabela SE1 [cite: 32]
         #IFNDEF TOP // Se não estiver usando TOPCONN [cite: 32]
            dbSetIndex(cArqTrab+OrdBagExT()) // Define o índice temporário [cite: 32]
         #ENDIF // Fim da compilação condicional [cite: 32]
         dbSetOrder(nIndex+1) // Define a ordem de acesso usando o índice temporário [cite:
32]
         dbGotop() // Posiciona no início do arquivo [cite: 32]
      Endif
      If mv_par03 == 2 // Se for relatório de Fornecedores (SE2)
```

```
            // SE2
            dbSelectArea("SE2") // Seleciona a tabela SE2
            dbSetOrder(1) // Define a ordem do índice [cite: 32]
            cArqTrab :=CriaTrab(NIL,.F.) // Cria arquivo temporário para índice [cite: 32]
            AADD(aInd,cArqTrab) // Adiciona o nome do arquivo temporário ao array de índices
[cite: 32]
            cIndE2:=IndexKey() // Obtém a chave do índice atual [cite: 32]
            cIndE2:=Right(cIndE2,Len(cIndE2)-10) // Ajusta a chave do índice [cite: 32]
            IndRegua("SE2",cArqTrab,cIndE2,,,OemToAnsi(STR0016) )  // Cria índice
temporário [cite: 32]
            dbCommit() // Confirma as operações [cite: 32]
            nIndex:=RetIndex("SE2") // Obtém o número do índice atual [cite: 32]
            dbSelectArea("SE2") // Seleciona a tabela SE2 [cite: 32]
            #IFNDEF TOP // Se não estiver usando TOPCONN [cite: 32]
               dbSetIndex(cArqTrab+OrdBagExt()) // Define o índice temporário [cite: 32]
            #ENDIF // Fim da compilação condicional [cite: 32]
            dbSetOrder(nIndex+1) // Define a ordem de acesso usando o índice temporário [cite:
32]
            dbGoTop() // Posiciona no início do arquivo [cite: 32]
         Endif
      Endif

      dbSelectArea(cAliasTmp) // Seleciona a área da tabela SE5 (ou alias temporário) [cite:
33]

      // Loop principal para processar os registros da tabela SE5
      While !Eof() .and. &(cCondE5) // Itera enquanto não for o fim do arquivo e a condição for
verdadeira [cite: 33]

         // Filtros iniciais para registros da SE5
         IF (Empty(E5_NUMERO)) .or. (E5_SITUACA = "C") .or. (E5_DTDIGIT > dDataBase) //
Verifica se o número está vazio, situação cancelada ou data de digitação maior que a data
base [cite: 33]
            dbskip() // Pula o registro
            Loop // Reinicia o loop
         Endif
         //Alterado Rafael Bonoldi
         //Foi inclusa condição para que o relatório não imprima os títulos baixados por motivo
de emissão de faturas. [cite: 34]
         IF Alltrim(E5_Motbx) $ "FAT" // Verifica se o motivo de baixa está em "FAT" [cite: 34]
            dbskip() // Pula o registro [cite: 34]
            Loop // Reinicia o loop [cite: 34]
         Endif

         If !(cAliasTmp)->E5_TIPODOC $ "VL/VM/BA/CP/V2/LJ/ES/PA/RA" // Verifica se o tipo
de documento está na lista permitida
            (cAliasTmp)->(dbSkip()) // Pula o registro
            Loop // Reinicia o loop
```

```
	Endif

	// Ignorar baixas do Loja de títulos à vista para países diferentes do Brasil
	If cPaisLoc <> "BRA" .And. E5_RECPAG=='R' .And. IsMoney((cAliasTmp)->E5_TIPO)
.And. E5_TIPODOC == "LJ" [cite: 35]
		(cAliasTmp)->(dbSkip()) // Pula o registro [cite: 35]
		Loop // Reinicia o loop [cite: 35]
	Endif

	// Verifica se existe estorno para esta baixa
	// Chama a função TemBxCanc para verificar se a baixa atual possui um estorno
associado.
	lTemBxCan := .F. // Inicializa a flag de baixa cancelada como falsa [cite: 36]
	dbSelectArea(cAliasTmp) // Seleciona a área da tabela SE5 [cite: 36]
	If TemBxCanc(
(cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TIPO+E5_CLIFOR+E5_LO
JA+E5_SEQ)) // Chama a função TemBxCanc [cite: 36]
		If mv_par19 == 2 // Se a opção for não listar baixas estornadas
			(cAliasTmp)->( dbSkip()) // Pula o registro [cite: 36]
			Loop // Reinicia o loop [cite: 36]
		Else // Se a opção for listar baixas estornadas
			lTemBxCan := .T. // Marca a flag de baixa cancelada como verdadeira [cite: 36]
		Endif
	Endif // Fim da verificação de estorno

	// Ignora movimentações de adiantamentos (PA/RA)
	If  (E5_TIPO $ MVRECANT .and. E5_TIPODOC == "RA") .or. (E5_TIPO $ MVPAGANT
.and. E5_TIPODOC == "PA") [cite: 37]
		dbskip() // Pula o registro [cite: 37]
		Loop // Reinicia o loop [cite: 37]
	Endif

	// Verifica a existência do título original em SE1 para estornos de adiantamento/crédito
negativo em clientes
	If  ((cAliasTmp)->E5_TIPO $ MVRECANT) .and. (cAliasTmp)->E5_TIPODOC == "ES"
.and. mv_par03 == 1 [cite: 38]
		dbSelectArea("SE1") // Seleciona a tabela SE1 [cite: 38]
		If mv_par14 = 1 // Se listar por Filial [cite: 38]
			SE1->(dbSetOrder(1)) // Define a ordem do índice [cite: 38]

SE1->(dbSeek(xFilial()+(cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TI
PO))) // Busca o título em SE1 [cite: 38]
		Else // Se listar por Empresa [cite: 38]

SE1->(dbSeek((cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TIPO))) //
Busca o título em SE1 [cite: 38]
		Endif
		If !(Found()) // Se não encontrou o título em SE1 [cite: 38]
```

```
            dbSelectArea(cAliasTmp) // Retorna para a área da SE5 [cite: 38]
            (cAliasTmp)->(dbSkip()) // Pula o registro na SE5 [cite: 38]
            Loop // Reinicia o loop [cite: 38]
         Endif
      Endif

      // Verifica a existência do título original em SE2 para estornos de adiantamento/débito
negativo em fornecedores
      If  ((cAliasTmp)->E5_TIPO $ MVPAGANT) .and. (cAliasTmp)->E5_TIPODOC == "ES"
.and. mv_par03 == 2 [cite: 39]
         dbSelectArea("SE2") // Seleciona a tabela SE2 [cite: 39]
         If mv_par14 = 1 // Se listar por Filial [cite: 39]
            SE2->(dbSetOrder(1)) // Define a ordem do índice [cite: 39]

SE2->(dbSeek(xFilial()+(cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TI
PO+E5_CLIFOR+E5_LOJA))) // Busca o título em SE2 [cite: 39]
         Else // Se listar por Empresa [cite: 39]

SE2->(dbSeek((cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TIPO+E5_
CLIFOR+E5_LOJA))) // Busca o título em SE2 [cite: 39]
         Endif
         If !(Found()) // Se não encontrou o título em SE2 [cite: 39]
            dbSelectArea(cAliasTmp) // Retorna para a área da SE5 [cite: 39]
            (cAliasTmp)->(dbSkip()) // Pula o registro na SE5 [cite: 39]
            Loop // Reinicia o loop [cite: 39]
         Endif
      Endif
      dbSelectArea(cAliasTmp) // Retorna para a área da SE5

      // Filtros adicionais para registros da SE5
      // Verifica se a data de digitação, cliente/fornecedor, prefixo ou natureza estão fora dos
parâmetros.
      IF E5_DTDIGIT < mv_par01 .or. E5_DTDIGIT > mv_par02 // Verifica se a data de
digitação está fora do período
         dbskip() // Pula o registro
         Loop // Reinicia o loop
      Endif

      IF E5_CLIFOR < mv_par04  .or. E5_CLIFOR > mv_par05 // Verifica se o código do
cliente/fornecedor está fora do intervalo [cite: 40]
         dbskip() // Pula o registro [cite: 40]
         Loop // Reinicia o loop [cite: 40]
      Endif

      If !Empty( mv_par12 ) // Se o prefixo inicial foi informado [cite: 40]
         If (cAliasTmp)->E5_PREFIXO < mv_par12 // Verifica se o prefixo é menor que o
inicial [cite: 40]
            dbskip() // Pula se menor [cite: 40]
```

```
            Loop // Reinicia o loop [cite: 40]
        Endif
    Endif

    If !Empty( mv_par13 ) // Se o prefixo final foi informado [cite: 40]
        If (cAliasTmp)->E5_PREFIXO > mv_par13 // Verifica se o prefixo é maior que o final
[cite: 40]
            dbskip() // Pula se maior [cite: 40]
            Loop // Reinicia o loop [cite: 40]
        Endif
    Endif
    If ( (cAliasTmp)->E5_NATUREZ < MV_PAR16 .Or. (cAliasTmp)->E5_NATUREZ >
MV_PAR17 ) // Verifica se a natureza está fora do intervalo [cite: 40]
        dbskip() // Pula se estiver fora do intervalo de natureza [cite: 40]
        Loop // Reinicia o loop [cite: 40]
    Endif

    // Filtros baseados no tipo de registro (Recebimento/Pagamento) e opções de
impressão (mv_par11)
    If (cAliasTmp)->E5_RECPAG == "R" .and. mv_par03 == 1 // Se for Recebimento e
relatório de Clientes [cite: 41]
        // Verifica quais registros de recebimento serão impressos (Normais /
Adiantamentos) [cite: 41]
        If mv_par11 == 2 .and. (E5_TIPO $ MVRECANT) // Se a opção é 'Normais' e o tipo é
adiantamento [cite: 42]
            dbskip() // Pula se for adiantamento [cite: 42]
            Loop // Reinicia o loop [cite: 42]
        Endif

        If mv_par11 == 3 .and. !(E5_TIPO $ MVRECANT)  // Se a opção é 'Adiantamentos' e
o tipo não é adiantamento [cite: 42]
            dbskip() // Pula se não for adiantamento [cite: 42]
            Loop // Reinicia o loop [cite: 42]
        Endif

    ElseIf (cAliasTmp)->E5_RECPAG == "P" .and. mv_par03 == 2 // Se for Pagamento e
relatório de Fornecedores [cite: 43]
        // Verifica quais registros de pagamento serão impressos (Normais / Adiantamentos)
[cite: 43]
        If mv_par11 == 2 .and. (E5_TIPO $ MVPAGANT) // Se a opção é 'Normais' e o tipo é
adiantamento [cite: 44]
            dbskip() // Pula se for adiantamento [cite: 44]
            Loop // Reinicia o loop [cite: 44]
        Endif
        If mv_par11 == 3 .and. !(E5_TIPO $ MVPAGANT) // Se a opção é 'Adiantamentos' e
o tipo não é adiantamento [cite: 44]
            dbskip() // Pula se não for adiantamento [cite: 44]
            Loop // Reinicia o loop [cite: 44]
```

```
            Endif
            // Ignora PA's (Pagamento de Adiantamento) pagos com Junta de Cheque [cite: 44]
            If (E5_TIPO $ MVPAGANT) .and. E5_TIPODOC == "BA" .and.
!empty(E5_NUMCHEQ) [cite: 45]
               dbskip() // Pula o registro [cite: 45]
               Loop // Reinicia o loop [cite: 45]
            Endif
         Endif


         // Filtros específicos para tipos de documento e Recebimento/Pagamento
         If mv_par03 == 1 .and. (cAliasTmp)->E5_RECPAG == "R" .and.
(cAliasTmp)->E5_TIPODOC =="ES" .and. !((cAliasTmp)->E5_TIPO $
MVRECANT+"/"+MV_CRNEG) [cite: 46]
            dbskip() // Pula o registro [cite: 46]
            loop // Reinicia o loop [cite: 46]
         Elseif mv_par03 == 2 .and. (cAliasTmp)->E5_RECPAG == "P" .and.
(cAliasTmp)->E5_TIPODOC =="ES" .and. !((cAliasTmp)->E5_TIPO $
MVPAGANT+"/"+MV_CPNEG) [cite: 47]
            dbskip() // Pula o registro [cite: 47]
            loop // Reinicia o loop [cite: 47]
         Endif

         // Mais filtros baseados em Recebimento/Pagamento, tipo e tipo de documento
         IF mv_par03 == 1 .and. (cAliasTmp)->E5_RECPAG != "R" [cite: 48]
            If (!((cAliasTmp)->E5_TIPO $ MVRECANT+"/"+MV_CRNEG ) .AND.
(cAliasTmp)->E5_TIPODOC !="ES") .or. ; // Baixa de RA [cite: 48]
               ((cAliasTmp)->E5_TIPO $ MVPAGANT+"/"+MV_CPNEG .AND.
(cAliasTmp)->E5_TIPODOC =="ES") .or. ; // Estorno da Baixa de PA [cite: 49]
               (( (cAliasTmp)->E5_TIPO $ MVRECANT) .AND. mv_par11 == 2).or. ; // [cite: 50]
               (!((cAliasTmp)->E5_TIPO $ MVRECANT) .AND. mv_par11 == 3) [cite: 50]

               (cAliasTmp)->(dbSkip()) [cite: 48]
               Loop [cite: 48]
            Endif
         Endif
         IF mv_par03 == 2 .and. (cAliasTmp)->E5_RECPAG != "P" [cite: 51]
            If (!( (cAliasTmp)->E5_TIPO $ MVPAGANT+"/"+MV_CPNEG ) .AND.
(cAliasTmp)->E5_TIPODOC !="ES") .or.; // Baixa de PA [cite: 52]
               ((cAliasTmp)->E5_TIPO $ MVRECANT+"/"+MV_CRNEG .AND.
(cAliasTmp)->E5_TIPODOC =="ES") .or. ; // Estorno da Baixa de RA [cite: 53]
               (( (cAliasTmp)->E5_TIPO $ MVPAGANT) .AND. mv_par11 == 2).or. ; // [cite: 54]
               (!((cAliasTmp)->E5_TIPO $ MVPAGANT) .AND. mv_par11 == 3) [cite: 54]

               (cAliasTmp)->(dbSkip()) [cite: 51]
               Loop [cite: 51]
            Endif
         Endif
```

```
// Filtros para baixas de adiantamentos com tipos de documento específicos
If mv_par03 == 1                    // Se for baixa de adiantamentos (Clientes) [cite: 55]
   If (cAliasTmp)->E5_RECPAG == "R" .and. E5_TIPO $ MVPAGANT+"/"+MV_CPNEG
.AND. (cAliasTmp)->E5_TIPODOC $ "VL/BA/DC/D2/MT/JR/J2/M2/CM/C2/CX" [cite: 55]
      dbskip() // Pula o registro [cite: 55]
      LOOP // Reinicia o loop [cite: 55]
   Endif
Endif

If mv_par03 == 2                    // Se for baixa de adiantamentos (Fornecedores) [cite:
56]
   If (cAliasTmp)->E5_RECPAG == "P" .and. E5_TIPO $ MVRECANT+"/"+MV_CRNEG
.AND. (cAliasTmp)->E5_TIPODOC $ "VL/BA/DC/D2/MT/JR/J2/M2/CM/C2/CX" [cite: 56]
      dbskip() // Pula o registro [cite: 56]
      LOOP // Reinicia o loop [cite: 56]
   Endif
Endif

// Filtro específico para Brasil e tipos de moeda "EF" ou "TF" com ordem de
recebimento preenchida
If cPaisLoc<>"BRA" .and. E5_TIPO=="EF |TF" .and. !Empty(E5_ORDREC)
   dbskip() // Pula o registro
   Loop // Reinicia o loop
Endif

// Verifica se o tipo da movimentação está entre os tipos selecionados
If ! Inside((cAliasTmp)->E5_TIPO)
   dbskip() // Pula o registro
   Loop // Reinicia o loop
Endif

// Filtro final para registros da SE5 (redundante, mas presente no código original)
IF Empty(E5_NUMERO) .or. E5_SITUACA = "C" [cite: 57]
   dbSkip() // Pula o registro [cite: 57]
   Loop // Reinicia o loop [cite: 57]
Endif

// Localiza o cliente ou fornecedor associado à movimentação da SE5
// Determina se a movimentação se refere a um cliente (Contas a Receber) ou
fornecedor (Contas a Pagar).
cCarteira := E5_RECPAG // Inicializa a carteira com o tipo de recebimento/pagamento
[cite: 58]
If E5_RECPAG == "R" // Se for Recebimento [cite: 58]
   If ( E5_TIPO$ MVPAGANT+"/"+MV_CPNEG).and.E5_TIPODOC $ "BAüVL" ; // Se
for adiantamento/débito negativo e tipo de documento for BA ou VL [cite: 58]
      .OR. E5_TIPODOC =="ES" // Ou tipo de documento for ES (Estorno) [cite: 58]
```

```
            cCarteira := "P"        // Considera como Pagamento (Baixa de adiantamento
inverte a carteira) [cite: 58]
         Endif
         IF       (E5_TIPO$ MVRECANT+"/"+MV_CRNEG) // Se for adiantamento/crédito
negativo [cite: 58]
            cCarteira := "R"                 // Considera como Recebimento (Cancelamento da
Baixa Adiantamento mantém a carteira) [cite: 58]
         Endif
      Endif

      If E5_RECPAG == "P" // Se for Pagamento [cite: 59]
         If (E5_TIPO$MVRECANT+"/"+MV_CRNEG).and.E5_TIPODOC $ "BAüVL" ; // Se for
adiantamento/crédito negativo e tipo de documento for BA ou VL [cite: 59]
            .OR. E5_TIPODOC =="ES" // Ou tipo de documento for ES (Estorno) [cite: 59]
            cCarteira := "R"        // Considera como Recebimento (Baixa de adiantamento
inverte a carteira) [cite: 59]
         Endif
         IF       (E5_TIPO$ MVPAGANT+"/"+MV_CPNEG) // Se for adiantamento/débito
negativo [cite: 59]
            cCarteira := "P"                 // Considera como Pagamento (Cancelamento da
Baixa Adiantamento mantém a carteira) [cite: 59]
         Endif
      Endif

      dbSelectArea(cAliasTmp) // Retorna para a área da tabela SE5
      IF cCarteira == "R" // Se a movimentação for de Recebimento (Cliente)
         If mv_par14 = 1 // Se listar por Filial
            SE1->(dbSetOrder(1)) // Define a ordem do índice na SE1

SE1->(dbSeek(xFilial()+(cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TI
PO))) // Busca o título em SE1
         Else // Se listar por Empresa

SE1->(dbSeek((cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TIPO))) //
Busca o título em SE1
         Endif
         If SE1->(!Found()) // Se não encontrou o título em SE1
            dbSelectArea(cAliasTmp) // Retorna para a área da SE5
            DbSkip() // Pula o registro na SE5
            Loop // Reinicia o loop
         Endif
      Else // Se a movimentação for de Pagamento (Fornecedor)
         If mv_par14 = 1 // Se listar por Filial
            SE2->(dbSetOrder(1)) // Define a ordem do índice na SE2

SE2->(dbSeek(xFilial()+(cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TI
PO+E5_CLIFOR+E5_LOJA))) // Busca o título em SE2
         Else // Se listar por Empresa
```

```
SE2->(dbSeek((cAliasTmp)->(E5_PREFIXO+E5_NUMERO+E5_PARCELA+E5_TIPO+E5_
CLIFOR+E5_LOJA))) // Busca o título em SE2
            Endif
            If SE2->(!Found()) // Se não encontrou o título em SE2
               dbSelectArea(cAliasTmp) // Retorna para a área da SE5
               DbSkip() // Pula o registro na SE5
               Loop // Reinicia o loop
            Endif
         Endif

         // Obtém informações do título associado (data de emissão, vencimento, moeda)
         IF cCarteira == "R" // Se for Recebimento (Cliente)
            dEmissao:=SE1->E1_EMISSAO // Data de emissão do título em SE1
            dVencto :=SE1->E1_VENCREA // Data de vencimento real do título em SE1
            nMoedaT       :=       SE1->E1_MOEDA // Moeda do título em SE1
            // Define a data para conversão de moeda
            If mv_par18 == 1 // Converte pela data de emissão
               dDataConv := SE1->E1_EMISSAO
            Elseif mv_par18 == 2 // Converte pela data base
               dDataConv := dDataBase
            Elseif mv_par18 == 3 // Converte pela data de vencimento real
               dDataConv := SE1->E1_VENCREA
            Else // Converte pela data final do período
               dDataConv := mv_par02
            Endif
         Else // Se for Pagamento (Fornecedor)
            dEmissao:=SE2->E2_EMISSAO // Data de emissão do título em SE2
            dVencto :=SE2->E2_VENCREA // Data de vencimento real do título em SE2
            nMoedaT       :=       SE2->E2_MOEDA // Moeda do título em SE2
            // Define a data para conversão de moeda
            If mv_par18 == 1 // Converte pela data de emissão
               dDataConv := SE2->E2_EMIS1
            Elseif mv_par18 == 2 // Converte pela data base
               dDataConv := dDataBase
            Elseif mv_par18 == 3 // Converte pela data de vencimento real
               dDataConv := SE2->E2_VENCREA
            Else // Converte pela data final do período
               dDataConv := mv_par02
            Endif
         Endif

         // Preenche o array aDados com informações da movimentação da SE5
         aDados[1] := E5_CLIFOR // Código do cliente/fornecedor
         aDados[2] := E5_LOJA // Loja do cliente/fornecedor
         aDados[3] := E5_DTDIGIT // Data de digitação da movimentação
         aDados[4] := E5_PREFIXO+E5_NUMERO+E5_PARCELA // Número completo da
movimentação
```

```
        aDados[5] := E5_DATA // Data da movimentação
        aDados[6] := E5_HISTOR // Histórico da movimentação
        aDados[7] := E5_RECPAG // Tipo de recebimento/pagamento
        aDados[8] := E5_TIPO // Tipo da movimentação
        aDados[9] := E5_TIPODOC // Tipo de documento da movimentação

        // Calcula o valor líquido da movimentação, considerando conversão de moeda e
opções de impressão
        nValliq :=
IIf(cPaisLoc=="BRA",E5_VALOR,xMoeda(E5_VLMOED2,nMoedaT,1,IIf(MV_PAR18!=1,dDat
aConv,E5_DATA))) // Calcula o valor líquido com ou sem conversão [cite: 60]
        If mv_par10 == 2  // Se a opção for não imprimir valor financeiro [cite: 60]
           If cPaisLoc == "BRA" // Se for Brasil
              nValliq -=    (E5_VLJUROS + E5_VLMULTA + E5_VLCORRE - E5_VLDESCO) //
Subtrai os componentes financeiros [cite: 60]
           Else // Se não for Brasil
              nVlFin := (E5_VLJUROS + E5_VLMULTA + E5_VLCORRE - E5_VLDESCO) //
Soma os componentes financeiros [cite: 60]

              // Converte os componentes financeiros para a moeda forte na data base do
relatório
              If MV_PAR18!=1 // Se a data de conversão não for a data de emissão [cite: 60]
                 // Converte o valor para a moeda 1 (geralmente dólar) na data da baixa [cite:
60]
                 nVlFin := Round(NoRound(xMoeda(nVlFin,1,nMoedaT,E5_DATA,3),3),2) [cite:
60]

                 // Converte o valor da moeda 1 para a moeda do título na data de conversão
[cite: 60]
                 nVlFin := Round(NoRound(xMoeda(nVlFin,nMoedaT,1,dDataConv,3),3),2) [cite:
60]
              Endif // Fim da conversão

              nValliq -=    nVlFin // Subtrai o valor financeiro convertido do valor líquido [cite: 60]
           Endif
        Endif

        // Grava o registro principal da movimentação no arquivo de trabalho temporário
        If E5_TIPODOC == "VM" // Se o tipo de documento for VM (Variação Monetária)
           GravaTrab("C",dEmissao,dVencto,nValliq,aDados,dDataConv,cAliasTmp)  // Grava
como Crédito (Registro principal)
        Else
           GravaTrab("B",dEmissao,dVencto,nValliq,aDados,dDataConv,cAliasTmp)  // Grava
como Baixa (Registro principal)
        Endif

        // Grava os componentes financeiros (Juros, Multa, Desconto, Correção Monetária) se
a opção for imprimir valores financeiros e não houver baixa cancelada
```

```
If mv_par10 == 1 .and. !lTemBxCan  // Se a opção for imprimir valor financeiro e não houver baixa cancelada [cite: 61]
    For n := 1 to 4 // Itera sobre os 4 tipos de componentes financeiros [cite: 61]
        DO CASE // Seleciona o campo e tipo do componente financeiro
            CASE n == 1 // Juros [cite: 61]
                cCampo := "E5_VLJUROS" [cite: 61]
                cTipo  := "J" [cite: 61]
                aDados[6] := STR0036 // Histórico: "Juros s/ Baixa" [cite: 61]
            CASE n == 2 // Multa [cite: 61]
                cCampo := "E5_VLMULTA" [cite: 61]
                cTipo  := "M" [cite: 61]
                aDados[6] := STR0037 // Histórico: "Multa s/ Baixa" [cite: 61]
            CASE n == 3 // Desconto [cite: 61]
                cCampo := "E5_VLDESCO" [cite: 61]
                cTipo  := "D" [cite: 61]
                aDados[6] := STR0038 // Histórico: "Desconto s/ Baixa" [cite: 61]
            CASE n == 4 // Correção Monetária [cite: 61]
                cCampo := "E5_VLCORRE" [cite: 61]
                cTipo  := "C" [cite: 61]
                aDados[6] := STR0039 // Histórico: "C.Monetaria s/ Baixa" [cite: 61]
        ENDCASE // Fim da seleção do componente financeiro

        If &(cCampo) != 0 .and. cPaisLoc == "BRA" // Se o valor do componente for diferente de zero e o país for Brasil [cite: 62]

GravaTrab(cTipo,dEmissao,dVencto,&(cCampo),aDados,dDataConv,cAliasTmp)  // Grava o componente no arquivo de trabalho [cite: 62]

        ElseIf &(cCampo) != 0 .and. cPaisLoc != "BRA" // Se o valor do componente for diferente de zero e o país não for Brasil [cite: 63]
            nVlFin := &(cCampo) // Obtém o valor do componente [cite: 63]

            // Converte os componentes financeiros para a moeda forte na data base do relatório [cite: 63]
            If MV_PAR18!=1 // Se a data de conversão não for a data de emissão [cite: 63]
                // Converte o valor para a moeda 1 (geralmente dólar) na data da baixa [cite: 63]
                nVlFin := Round(NoRound(xMoeda(nVlFin,1,nMoedaT,E5_DATA,3),3),2) [cite: 63]

                // Converte o valor da moeda 1 para a moeda do título na data de conversão [cite: 63]
                nVlFin := Round(NoRound(xMoeda(nVlFin,nMoedaT,1,dDataConv,3),3),2) [cite: 63]
            Endif // Fim da conversão dos componentes [cite: 63]
        Endif
    Next // Fim da iteração sobre os componentes financeiros
Endif // Fim da seção de gravação de componentes financeiros
```

```
        dbSelectArea(cAliasTmp) // Retorna para a área da tabela SE5
        dbSkip() // Avança para o próximo registro na SE5
    Enddo // Fim do loop de processamento da tabela SE5

    #IFDEF TOP // Se estiver usando TOPCONN [cite: 63]
        If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400 [cite: 63]
            DBSelectArea(cAliasTmp) // Seleciona a área da SE5 [cite: 63]
            DbCloseArea() // Fecha a área da SE5 [cite: 63]
            ChkFile("SE5") // Verifica a integridade do arquivo [cite: 63]
        Endif
    #ENDIF // Fim da compilação condicional para TOPCONN [cite: 63]

    // Inicia rotina de impressão do relatório a partir do arquivo de trabalho temporário
    // A impressão é realizada iterando sobre os registros do arquivo temporário (cNomeArq).
    dbSelectArea("cNomeArq") // Seleciona a área do arquivo de trabalho
    dbGoTop() // Posiciona no início do arquivo

    SetRegua(RecCount()) // Configura a barra de status com o total de registros a serem
impressos

    // Define as posições das colunas na impressão com base no tamanho do campo cliente
    aTam          := TAMSX3("E1_CLIENTE") // Obtém o tamanho do campo E1_CLIENTE
    aColu := IIF (aTam[1] > 6,{053,073,084,095,117,132},{039,059,070,081,103,118}) // Define
as posições das colunas

    // Loop principal para imprimir cada registro do arquivo de trabalho
    While !Eof() // Itera enquanto não for o fim do arquivo

        IF lEnd // Verifica se o usuário cancelou a impressão
            @PROW()+1,0 PSAY OemToAnsi(STR0017)  //"Cancelado pelo Operador" [cite: 64]
            Exit // Sai do loop de impressão
        Endif

        // Verifica se é necessário imprimir um novo cabeçalho de página
        IF li > 58 // Se a linha atual for maior que 58 (próximo da margem inferior)
            IF m_pag == mv_par07 // Se a página atual for a página final especificada
                m_pag := 2 // Reinicia o contador de páginas para continuar imprimindo após a
página final (comportamento incomum)
            Endif
            cabec(titulo,cabec1,cabec2,nomeprog,tamanho,IIF(nCompress==1,15,18)) //
Imprime o cabeçalho da página
        Endif

        // Inicia uma nova quebra por data, se necessário
        IF nQuebra == 0 // Se não houver quebra ativa
            dData := DATAX // Armazena a data do primeiro registro do grupo [cite: 64]
            nTotTit := 0 // Zera o total de títulos por grupo [cite: 64]
```

```
            nTotDeb := 0 // Zera o total de débito por grupo [cite: 64]
            nTotCrd := 0 // Zera o total de crédito por grupo [cite: 64]
            nQuebra := 1 // Marca que uma quebra está ativa [cite: 64]
        Endif

        // Verifica se a data do registro atual é diferente da data do grupo atual
        IF dData != DATAX // Se a data mudou [cite: 64]
            SubTot510(dData,nTotTit,nTotDeb,nTotCrd,aColu) // Imprime o subtotal do grupo
anterior [cite: 64]
            nGerTit += nTotTit // Adiciona os totais do grupo ao total geral [cite: 64]
            nGerDeb += nTotDeb // Adiciona os totais do grupo ao total geral [cite: 64]
            nGerCrd += nTotCrd // Adiciona os totais do grupo ao total geral [cite: 64]
            nQuebra := 0 // Reseta a flag de quebra [cite: 64]
            li ++; li ++ // Avança duas linhas (espaço entre grupos) [cite: 64]
            Loop // Reinicia o loop para processar o novo grupo [cite: 64]
        Endif

        IncRegua() // Incrementa a barra de status

        // Imprime os dados do registro atual no relatório
        @li,         0 PSAY DATAX // Data [cite: 64]
        @li, 11 PSAY Substr(NUMERO,1,3)+"-"+Substr(NUMERO,4,nTamNro) // Prefixo e
Número [cite: 64]
        @li, 28 PSAY Substr(NUMERO,nTamNro+4,3) // Parcela [cite: 64]
        @li, 32 PSAY CODIGO // Código do cliente/fornecedor [cite: 64]

        // Localiza o nome do cliente ou fornecedor nas tabelas SA1 ou SA2
        IF mv_par03 == 1 // Se for Cliente
            dbSelectArea("SA1") // Seleciona a tabela SA1
        Else // Se for Fornecedor
            dbSelectArea("SA2") // Seleciona a tabela SA2
        Endif
        dbSeek(xFilial()+cNomeArq->CODIGO+cNomeArq->LOJA) // Busca o
cliente/fornecedor pela filial, código e loja
        @li,aColu[1] PSAY
IIF(mv_par03==1,Substr(SA1->A1_NOME,1,19),Substr(SA2->A2_NOME,1,19)) // Imprime o
nome (truncado em 19 caracteres)
        Select cNomeArq // Retorna para a área do arquivo de trabalho

        @li,aColu[2] PSAY EMISSAO // Data de emissão
        @li,aColu[3] PSAY VENCREA // Data de vencimento real
        @li,aColu[4] PSAY SubStr(HISTOR, 1, 22) // Histórico (primeiros 22 caracteres)
        nC := IIF( DC = "D",aColu[5],aColu[6]) // Define a coluna de impressão (Débito ou
Crédito)
        @li, nC PSAY VALOR    Picture tm(VALOR,14,nDecs) // Imprime o valor formatado
        If Len(AllTrim(HISTOR)) > 22 // Se o histórico for maior que 22 caracteres
            li++ // Avança para a próxima linha
```

```
        @li,aColu[4] PSAY SubStr(HISTOR,23,17) // Imprime a continuação do histórico
(próximos 17 caracteres)
      Endif

      // Atualiza os totais por grupo e gerais
      nTotTit ++ // Incrementa o total de títulos do grupo
      IF DC == "D" // Se for Débito
         nTotDeb += VALOR // Adiciona ao total de débito do grupo
      Else // Se for Crédito
         nTotCrd += VALOR // Adiciona ao total de crédito do grupo
      Endif

      li ++ // Avança para a próxima linha de impressão
      dbSkip() // Avança para o próximo registro no arquivo de trabalho
   Enddo // Fim do loop de impressão

   // Imprime o subtotal do último grupo e o total geral, se houver registros no arquivo de
trabalho
   IF li != 80 // Verifica se houve alguma impressão (li é 80 inicialmente)
      IF li > 55 // Se a linha atual estiver próxima da margem inferior
         cabec(titulo,cabec1,cabec2,nomeprog,tamanho,IIF(nCompress==1,15,18)) //
Imprime o cabeçalho em uma nova página
      Endif
      SubTot510(dData,nTotTit,nTotDeb,nTotCrd,aColu) // Imprime o subtotal do último grupo
      nGerTit += nTotTit // Adiciona os totais do último grupo ao total geral
      nGerDeb += nTotDeb // Adiciona os totais do último grupo ao total geral
      nGerCrd += nTotCrd // Adiciona os totais do último grupo ao total geral
      li += 2 // Avança duas linhas
      @li,000 PSAY OemToAnsi(STR0018)  //"T O T A L  G E R A L ->" [cite: 64]
      @li,030 PSAY nGerTit             Picture "9999" // Imprime o total geral de títulos [cite:
64]
      @li,035 PSAY OemToAnsi(STR0019)  //"Movimentacoes" [cite: 64]
      @li,aColu[5] PSAY nGerDeb      Picture tm(nGerDeb,14,nDecs) // Imprime o total geral
de débito formatado [cite: 64]
      @li,aColu[6] PSAY nGerCrd      Picture tm(nGerCrd,14,nDecs) // Imprime o total geral
de crédito formatado [cite: 64]
      roda(cbcont,cbtxt,Tamanho) // Imprime o rodapé
   Endif
```

Parte 3

```
// Apaga o arquivo e índice temporário
   // Remove os arquivos temporários criados durante a execução do relatório para liberar
espaço.
   dbSelectArea("cNomeArq") // Seleciona a área do arquivo de trabalho
   dbCloseArea() // Fecha o arquivo de trabalho
   Ferase(cNomeArq+GetDBExtension()) // Apaga o arquivo de dados temporário
```

```
   Ferase(cNomeArq+OrdBagExt()) // Apaga o arquivo de índice temporário
   Set Device To Screen // Restaura a saída do dispositivo para a tela

   #IFDEF TOP // Se estiver usando o banco TOPCONN
      If TcSrvType() != "AS/400" // Se o tipo de servidor não for AS/400
         // Restaura as áreas e índices das tabelas SE1, SE2 e SE5 no ambiente TOPCONN
         dbSelectArea("SE1") // Seleciona a área da SE1
         dbCloseArea() // Fecha a área da SE1
         ChKFile("SE1") // Verifica a integridade do arquivo
         dbSelectArea("SE1") // Seleciona a área da SE1 novamente
         dbSetOrder(1) // Define a ordem do índice para 1

         dbSelectArea("SE2") // Seleciona a área da SE2
         dbCloseArea() // Fecha a área da SE2
         ChKFile("SE2") // Verifica a integridade do arquivo
         dbSelectArea("SE2") // Seleciona a área da SE2 novamente
         dbSetOrder(1) // Define a ordem do índice para 1

         dbSelectArea("SE5") // Seleciona a área da SE5
         dbSetOrder(1) // Define a ordem do índice para 1
      Else // Se o tipo de servidor for AS/400
   #ENDIF // Fim da compilação condicional para TOPCONN
         // Restaura as áreas e índices das tabelas SE1, SE2 e SE5 em ambientes que não
usam TOPCONN (ou AS/400)
         dbSelectArea("SE1") // Seleciona a área da SE1
         RetIndex("SE1") // Restaura o índice original da SE1
         dbSetOrder(1) // Define a ordem do índice para 1
         Set Filter To // Remove qualquer filtro aplicado à tabela SE1

         dbSelectArea("SE2") // Seleciona a área da SE2
         RetIndex("SE2") // Restaura o índice original da SE2
         dbSetOrder(1) // Define a ordem do índice para 1
         Set Filter To // Remove qualquer filtro aplicado à tabela SE2

         dbSelectArea("SE5") // Seleciona a área da SE5
         RetIndex("SE5") // Restaura o índice original da SE5
         dbSetOrder(1) // Define a ordem do índice para 1
         Set Filter To // Remove qualquer filtro aplicado à tabela SE5
#IFDEF TOP // Se estiver usando TOPCONN
   Endif
#ENDIF // Fim da compilação condicional para TOPCONN

   // Apaga arquivos de índice temporários adicionais, se existirem
   For nI:=1 to Len(aInd) // Itera sobre o array de nomes de arquivos de índice temporários
      If File(aInd[nI]+OrdBagExt()) // Verifica se o arquivo de índice existe
         Ferase(aInd[nI]+OrdBagExt()) // Apaga o arquivo de índice
      Endif
   Next // Fim da iteração
```

```
    // Finaliza a impressão e envia para o spool, se configurado
    If aReturn[5] = 1 // Verifica a configuração de spool (aReturn[5] = 1 indica spool)
        Set Printer TO // Restaura a impressora padrão (fecha o arquivo de impressão)
        dbCommitall() // Confirma todas as operações de banco de dados pendentes
        ourspool(wnrel) // Envia o arquivo de relatório para o spool de impressão
    End
    MS_FLUSH() // Libera o buffer de memória

Return // Fim da função FA510Imp

/*/
// Função: SubTot510
// Autor: Rafael Bonoldi
// Data: 22.03.05
// Descrição: Imprime o subtotal por data no relatório Diário Auxiliar.
// Esta função é chamada para imprimir os totais de títulos, débito e crédito para cada
agrupamento por data.
// Parâmetros:
// dData: Data do agrupamento.
// nTotTit: Total de títulos no agrupamento.
// nTotDeb: Total de débito no agrupamento.
// nTotCrd: Total de crédito no agrupamento.
// aColu: Array com as posições das colunas de impressão.
// Utilização: Chamada pela função FA510Imp durante a iteração dos registros impressos.
/*/
Static Function SubTot510(dData,nTotTit,nTotDeb,nTotCrd,aColu)

    li++ // Avança uma linha para a impressão do subtotal
    @li,  0 PSAY OemToAnsi(STR0027)+Dtoc(dData)  //"TOTAL DO DIA " + Data formatada
    @li,aColu[5] PSAY nTotDeb          Picture tm(nTotDeb,14,nDecs) // Imprime o total de
débito do grupo formatado
    @li,aColu[6] PSAY nTotCrd          Picture tm(nTotCrd,14,nDecs) // Imprime o total de
crédito do grupo formatado
Return (.t.) // Retorna verdadeiro (indica que a impressão do subtotal ocorreu)

/*/
// Função: GravaTrab
// Autor: Wagner Xavier
// Data: 25.11.92
// Substituído pelo assistente de conversão do AP6 IDE em 22/03/05
// Descrição: Grava um registro no arquivo de trabalho temporário.
// Esta função é responsável por preencher os campos do arquivo temporário (cNomeArq)
com os dados dos títulos ou movimentações.
// Parâmetros:
// cTipo: Tipo do registro a ser gravado (B: Baixa, C: Crédito, D: Débito, J: Juros, M: Multa,
D: Desconto, C: Correção).
// dEmissao: Data de emissão do título associado.
```

```
// dVencto: Data de vencimento real do título associado.
// nValor: Valor a ser gravado.
// aDados: Array contendo dados da movimentação (E5_CLIFOR, E5_LOJA, E5_DTDIGIT,
E5_NUMERO, E5_DATA, E5_HISTOR, E5_RECPAG, E5_TIPO, E5_TIPODOC).
// dDataConv: Data para conversão de moeda.
// cAliasTmp: Alias temporário da tabela de origem (SE5).
// Utilização: Chamada pela função FA510Imp para popular o arquivo de trabalho antes da
impressão.
/*/
// Substituido pelo assistente de conversao do AP6 IDE em 22/03/05 ==> Static Function
GravaTrab(cTipo,dEmissao,dVencto,nValor,aDados,dDataConv,cAliasTmp)
Static Function GravaTrab(cTipo,dEmissao,dVencto,nValor,aDados,dDataConv,cAliasTmp)
   LOCAL cDCR,cDCP,cAlias:=Alias() // Variáveis para definir Débito/Crédito no arquivo
temporário e salvar o alias atual
   Local nMoedaBco  :=        1 // Moeda do banco (inicializada como 1)

   // Conversão de moeda para países diferentes do Brasil
   If cPaisLoc  #"BRA"
      If
!Empty((cAliasTmp)->E5_BANCO+(cAliasTmp)->E5_AGENCIA+(cAliasTmp)->E5_CONTA)
// Verifica se os dados do banco/agência/conta estão preenchidos na SE5
         SA6->(DbSetOrder(1)) // Define a ordem do índice para a tabela SA6 (Bancos)

SA6->(DbSeek(xFilial()+(cAliasTmp)->E5_BANCO+(cAliasTmp)->E5_AGENCIA+(cAliasTm
p)->E5_CONTA)) // Busca o banco na SA6
         nMoedaBco    :=        Max(SA6->A6_MOEDA,1) // Obtém a moeda do banco (pelo
menos 1)
      EnDif // Fim da verificação de banco
      // Converte o valor para a moeda 1 (geralmente dólar) se nValor for Nil ou realiza a
conversão do valor
      nValor     :=        Iif(
nValor=Nil,Round(xMoeda((cAliasTmp)->E5_VALOR,nMoedaBco,1,IIf(MV_PAR18!=1,dData
Conv,(cAliasTmp)->E5_DATA),nDecs+1),nDecs),nValor )
   Endif // Fim da conversão de moeda

   // Define se o registro será Débito (D) ou Crédito (C) no arquivo de trabalho
   If cTipo $ "B#D" // Se o tipo for Baixa ou Débito
      cDCR="C" // Crédito na contrapartida
      cDCP="D" // Débito no arquivo de trabalho
   ElseIf cTipo $ "C" // Se o tipo for Crédito (Variação Monetária)
      If nValor < 0   // Verifica se o valor é negativo (Variação monetária negativa)
         cDCR="C" // Crédito na contrapartida
         cDCP="D" // Débito no arquivo de trabalho
         nValor := ABS(nValor) // Usa o valor absoluto
      Else
         cDCR="D" // Débito na contrapartida
         cDCP="C" // Crédito no arquivo de trabalho
      Endif
```

```
   Else // Para outros tipos (Juros, Multa, Desconto, Correção)
      cDCR="D" // Débito na contrapartida
      cDCP="C" // Crédito no arquivo de trabalho
   EndIf

   // Grava o registro no arquivo de trabalho temporário
   Reclock("cNomeArq",.t.) // Bloqueia o registro no arquivo de trabalho
   Replace CODIGO  With aDados[1]          // Código do cliente/fornecedor
(SE5->E5_CLIFOR)
   Replace LOJA        With aDados[2]      // Loja do cliente/fornecedor (SE5->E5_LOJA)
   Replace DATAX       With aDados[3]      // Data de referência (SE5->E5_DTDIGIT)
   Replace NUMERO  With aDados[4]          // Número completo da movimentação
(SE5->E5_PREFIXO+SE5->E5_NUMERO+SE5->E5_PARCELA)
   Replace VALOR       With nValor // Valor a ser gravado
   Replace EMISSAO With dEmissao // Data de emissão do título associado
   Replace HISTOR  With Iif(Empty(aDados[6]) .Or.; // Verifica se o histórico está vazio ou é
o histórico padrão de recebimento
                        Upper(aDados[6]) = Upper(OemToAnsi(STR0033)),;  // "Valor recebido s/
Titulo"
                        OemtoAnsi(STR0034), aDados[6]) // Se vazio/padrão, usa "Baixa de
Titulo", senão usa o histórico da movimentação
   Replace VENCREA With dVencto // Data de vencimento real do título associado

   // Define o campo DC (Débito/Crédito) no arquivo de trabalho com base no tipo de
recebimento/pagamento e tipo/documento da movimentação
   IF aDados[7] = "R" // Se for Recebimento (SE5->E5_RECPAG)
      Replace DC With Iif((!(aDados[8] $ MVRECANT+"/"+MV_CRNEG) .or.;      // Se não for
Adiantamento/Crédito Negativo OU
         aDados[9]=="ES" ) .and. (!(aDados[8] $ MVPAGANT+"/"+MV_CPNEG .and.
(cAliasTmp)->E5_MOTBX = "CMP" .and. aDados[9]=="ES")) ;  // Tipo de documento for
Estorno E (não for Adiantamento Pagamento/Débito Negativo E motivo de baixa for CMP E
tipo de documento for Estorno)
         ,cDCR,cDCP)      // Usa cDCR (Crédito) se a condição for verdadeira, senão usa
cDCP (Débito) - Lógica complexa de Débito/Crédito
   Else // Se for Pagamento
      Replace DC With Iif((!( aDados[8] $ MVPAGANT+"/"+MV_CPNEG) .or. ; // Se não for
Adiantamento Pagamento/Débito Negativo OU
         aDados[9]=="ES" ) .and. (!( aDados[8] $ MVRECANT+"/"+MV_CRNEG .and.
(cAliasTmp)->E5_MOTBX = "CMP" .and. aDados[9]=="ES")) ;   // Tipo de documento for
Estorno E (não for Adiantamento Recebimento/Crédito Negativo E motivo de baixa for CMP
E tipo de documento for Estorno)
         ,cDCP,cDCR) // Usa cDCP (Débito) se a condição for verdadeira, senão usa cDCR
(Crédito) - Lógica complexa de Débito/Crédito
   Endif
   MsUnlock() // Libera o bloqueio do registro
   dbSelectArea(cAlias) // Retorna para a área de trabalho anterior
Return Nil // Retorna Nil
```

```
/*/
// Função: Inside
// Autor: Desconhecido
// Data: Desconhecida
// Substituído pelo assistente de conversão do AP6 IDE em 22/03/05
// Descrição: Verifica se um tipo de título/movimentação está na lista de tipos selecionados.
// Esta função é utilizada para filtrar os registros com base nos tipos de título/movimentação que o usuário selecionou nos parâmetros (mv_par15 e cTipos).
// Parâmetros:
// cTp: O tipo de título ou movimentação a ser verificado.
// Utilização: Chamada pelas funções FA510Imp para filtrar registros das tabelas SE1, SE2 e SE5.
/*/
// Substituido pelo assistente de conversao do AP6 IDE em 22/03/05 ==> Static Function InSide(cTp)
Static Function InSide(cTp)
   // Verifica se a seleção de tipos não está ativa (mv_par15 != 1) E a string de tipos selecionados está vazia.
        IF mv_par15 != 1 .And. Empty(cTipos)
                Return .t. // Se a condição acima for verdadeira, retorna verdadeiro (todos os tipos são considerados "dentro")
        Else
      // Se a seleção de tipos está ativa OU a string de tipos selecionados não está vazia, verifica se o tipo (cTp) está presente na string de tipos selecionados (cTipos).
                Return (cTp$cTipos) // Retorna verdadeiro se o tipo estiver na string, falso caso contrário.
        Endif
Return // Retorna o resultado da condição (redundante após o Endif)

/*/
// Função: AjustaSx5
// Autor: Rafael Bonoldi
// Data: 22/03/05
// Descrição: Ajusta a tabela SX5 (Tabela Genérica) para incluir o tipo de moeda "R$".
// Esta função garante que o código "R$" exista na tabela SX5, que é usada para armazenar tipos de moeda.
// Utilização: Chamada pela função FA510Imp no início do processo de impressão para preparar o ambiente.
/*/
Static Function AjustaSx5()

   Local aArea := GetArea() // Salva a área de trabalho atual

   dbSelectArea("SX5") // Seleciona a tabela SX5
   If !(dbSeek(xFilial("SX5")+"05"+"R$ ")) // Tenta buscar o registro com filial atual, tabela "05" e chave "R$"
      RecLock("SX5",.T.) // Se não encontrou, bloqueia a tabela para inclusão
      X5_FILIAL        :=      xFilial("SX5") // Preenche o campo de filial
```

```
        X5_TABELA       :=      "05" // Preenche o campo de tabela
        X5_CHAVE                :=      "R$ " // Preenche o campo de chave
        X5_DESCRI       :=      "DINHEIRO" // Preenche a descrição em Português
        X5_DESCSPA      :=      "DINERO" // Preenche a descrição em Espanhol
        X5_DESCENG      :=      "CASH" // Preenche a descrição em Inglês
        MsUnlock() // Libera o bloqueio do registro
    Endif // Fim da verificação e inclusão

    RestArea(aArea) // Restaura a área de trabalho anterior

Return .T. // Retorna verdadeiro (indica que a função foi executada)
```