Logica Procedural

Sumário

Logica Procedural

Introdução

A melhor analogia seriam os bloquinhos Lego, montando e remontando partes do código, no caso mainframe usando o Ctrl-C & Ctrl-V, em conjunto com os slots de memória do TSO, A seguir apresento conceitos importantes para fixar e entender este tipo de paradigma que surgiu nos anos 70, para substituir e ordenar a velha programação monolítica, numa das crises do software. Com duas décadas até então, os administradores de CPD, descobriram a duras penas, que algo não ia bem, programas muito grandes, de difícil analise, desvios via GO TO e mil e uma função dentro do mesmo programa.

Pensando nisso, criaram novas regras, novos conceitos e maneiras de fazer as coisas melhor, pensando sempre no pobre Analista de Sustentação, que seria acionado em plena madrugada, quando um processo Batch abendasse.

Paradigma Procedural

Entre essas regras, refinaram os seguintes conceitos:

Procedimentos

Como mencionado anteriormente, os procedimentos são a espinha dorsal da programação procedural. Eles são criados para realizar tarefas específicas e podem receber argumentos e retornar valores. A capacidade de reutilizar procedimentos em diferentes partes de um programa é uma das vantagens significativas desse paradigma, lembrando cada procedimento o ideal é que tenha apenas uma entrada e uma saída. Podem ser internos parágrafos / seções ou externos SubProgramas, Programas Aninhados chamados via CALL.

Estruturação do Código

Na programação procedural, o código é frequentemente estruturado em procedimentos chamados em uma ordem específica para executar uma tarefa maior. Isso ajuda a dividir um programa complexo em partes gerenciáveis, facilitando o desenvolvimento e a manutenção. Evitando de cair na tentação de criar grandes monolitos e código com complexidade muito elevado.

Variáveis Locais e Globais

As linguagens procedurais geralmente suportam variáveis locais e globais. Variáveis locais são acessíveis apenas no procedimento em que foram declaradas, enquanto variáveis globais podem ser acessadas em todo o programa. No caso de um programa COBOL, utilizamos pseudo-globais, pois devido a característica da Linguagem, as variáveis são criadas na Data Division e estão disponíveis para uso no programa inteiro, porém o programador, pode segmentar seu uso de acordo com a tarefa: arquivos, banco de dados, sub-rotinas etc.

Controle de Fluxo

O controle de fluxo é uma parte crucial da programação procedural. Isso envolve estruturas de controle, como condicionais (if-else, evaluate) e loops (perform), que permitem a tomada de decisões e a repetição de tarefas.

Reutilização de Código

Uma das principais vantagens das linguagens procedurais é a capacidade de reutilização de código. Os procedimentos podem ser chamados várias vezes em um programa, economizando tempo e esforço na programação. Ademais no COBOL com o conceito de COPY, podemos utilizar COPÝ PROCEDURES para compartilhar código entre vários programas e ao mesmo tempo ter apenas um único ponto de atualização.

Utilizando sub-programas aquelas partes comuns a diversos processos também podem ser recicladas e reutilizadas em N programas, lembrando que a partir da versão 6.3 do COBOL o programador pode criar funções intrínsecas próprias e compartilhar entre sistemas.

Operações Logicas

Como foi visto até o momento, um programa sem operações logicas fica muito limitado, sendo apenas um executor de script, seguindo um fluxo sem muita variação e sem muito pensar, apenas executando tarefas repetitivas.

Não existe certo nem errado, tudo depende da necessidade imperativa ao codificar, porém ao colocarmos inteligência, criando controles de fluxo, desvios, repetições e validações nosso código ficará muito mais poderoso.

Para isso em perguntas sempre teremos duas respostas: Sim ou Não, Verdadeiro ou Falso, Maior ou Menor, Maior/igual ou Menor, Maior ou Menor/Igual esta dualidade é muito importante em nosso processamento de dados.

E para dar mais poder e aumentar a quantidade de respostas, podemos incluir mais alguns operadores: NOT, AND e OR, criar expressões aritméticas e expressões comparativa, a seguir iremos estudar cada um deles.

Operadores e Expressões

Expressões executam ações específicas, baseadas em um operador com um ou dois operandos. Um operando pode ser uma constante, uma variável ou um resultado de função.

Os operadores são aritméticos, lógicos e relacionais.

Operadores Aritméticos (+, -, *, /, **, %)

Os operadores aritméticos executam operações matemáticas, como adição e subtração com operandos. Há dois tipos de operadores matemáticos: unário e binário. Os operadores unários executam uma ação com um único operando. Operadores binários executam ações com dois operandos. Em uma expressão complexa, (dois ou mais operandos), a ordem de avaliação depende de regras de precedência.

Operadores Aritméticos Unários

Operadores unários são operadores aritméticos que desempenham uma ação em um único operando. A linguagem de script reconhece o negativo do operador unário (-).

O operador unário negativo inverte o sinal de uma expressão de positivo para negativo ou vice-versa. O efeito geral é de multiplicar o número por -1.

Conheça NOT, AND e OR

NOT

Converte a pergunta na negativa, realizando a instrução somente se for FALSE a resposta.

Exemplo:

```
IF NOT IDADE > 18
```

```
....
END-IF
```

Neste exemplo somente irá executar as instruções se a IDADE for menor que 18

AND

Nesta situação, os comandos somente serão executado se ambas as respostas forem TRUE.

Exemplo:

```
IF  IDADE > 18 AND SEXO = “MASCULINO”
....
END-IF
```

Neste exemplo somente irá executar as instruções se a IDADE for maior que 18 e SEXO igual a masculino

Resultado de Uso de Operadores Lógicos

A seguir apresento resultados de expressões logicas no COBOL, lembrando que podemos unir uma ou mais expressões, utilizando de Parênteses para unificar condições complexas.

AND

Expressão 1	Expressão 2	Expressão 1 AND Expressão 2
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

OR

Expressão 1	Expressão 2	Expressão 1 OR Expressão 2
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Operadores Aritméticos Binários

Símbolo	Operação	Exemplo	Descrição
+	Adição	a + b	Soma os dois operandos
-	Subtração	a - b	Subtrai o segundo operando do primeiro
*	Multiplicação	a * b	Multiplica os dois operandos
/	Divisão	a / b	Divide o primeiro operando pelo segundo
**	Potência	a ** b	Eleva o primeiro operando à potência do segundo

Operadores Relacionais

Símbolo	Operação	Exemplo	Descrição
<	Menor que	a < b	Verdadeira se a for menor que b.
>	Maior que	a GT b	Verdadeira se a for maior que b.
=	Igual a	a = b	Verdadeira se a for igual a b
NOT =	Diferente de	a NOT = b	Verdadeira se a for diferente de b.
<=	Menor ou Igual a	a <= b	Verdadeiro se a for menor ou igual a b.
>=	Maior ou Igual a	a GE b	Verdadeiro se a for maior ou igual a b.

Para Fixar

• Programa fonte:

É o conjunto de palavras ou símbolos escritos de forma ordenada, único fonte ou um conjunto de fontes usando copy books e copy procedures, contendo instruções em uma das linguagens de programação existentes, de maneira lógica.

• Programa objeto:

No caso do COBOL, os comandos estão em linguagem Natural e ainda utilizam-se de Macros Db2 e CICS. Necessitam que são explodidas, gerando um único código fonte. Lembrando que existem linguagem compiladas e as que são interpretadas (JCL). As linguagens compiladas, ou sejam, são linkeditadas e após ser compilado o código fonte, transformam-se em software, ou seja, programas executáveis.

• Compilador:

É usado principalmente para os programas que traduzem o código de fonte de uma linguagem de programação de alto nível (Natural) para uma linguagem de programação de baixo nível (Assembler).

• Linguagem de Alto Nível:

É como se chama, na Ciência da Computação de linguagens de programação, uma linguagem com um nível de abstração relativamente elevado, longe do código de máquina e mais próximo da linguagem humana.

• Linguagem de Baixo Nível:

trata-se de uma linguagem de programação que compreende as características da arquitetura do computador. Assim, utiliza somente instruções do processador, para isso é necessário conhecer os registradores da máquina.