01 Prefacio

Revisão: 13/07/2002

Abrangência

Versão 5.07	Versão 5.08	Versão 6.09	Versão 7.10	Versões Anteriores

Existe um ditado chinês que diz: “O Homem não tropeça em montanhas, tropeça em pedregulhos, areia, pequenos buracos, mas nunca em uma montanha”.

Isso nos remete a pensar que onde erramos é exatamente no simples, naquele detalhe quase imperceptível e que tem um valor muito grande para o todo. Avaliemos do ponto de vista humano; será tão difícil cumprimentar a todos, sermos mais amigos, mais serenos nas decisões e companheiros uns dos outros e trabalharmos em equipe? Por que muitas vezes não o fazemos? Por que insistimos no individualismo e no mal-humor? Não seria mais fácil, até mesmo óbvio, estarmos mais bem-humorados e dispostos a trabalhar em equipe, trocarmos conhecimento e discernimento nas decisões, pensarmos mais no todo porém se importando com as partes que o compõe?

Seria mais interessante se ao caminharmos por um parque, prestássemos mais atenção nas árvores, no caminho, nas flores, no canto dos passarinhos sem se esquecer do objetivo do passeio, sem perder a noção de tempo e distância, mas curtindo muito a paisagem, o detalhe.

Agora vamos traçar um paralelo com o nosso dia a dia. Não seria melhor ao reservarmos um fonte, verificarmos com mais atenção:

As condicionais? Afinal muitas vezes não testamos um ELSE.

Os filtros? Geralmente esquecemos de tentar otimizar a performance no SQL.

As mensagens? Afinal é tão comum nos depararmos com textos completamente sem sentido.

Os helps? Damos pouca atenção a eles e nos esquecemos que é a primeira coisa que o usuário tenta.

Imaginem algumas ligações menos por causa de uma simples documentação a mais! Aquele ponto de entrada que criamos e não pensamos nos supostos parâmetros que nosso pessoal em campo pode querer, ou mesmo no retorno mais adequado para aquela função.

Lembrem-se também da documentação do novo campo; Ela realmente é necessária? Se a chave de índice é imprescindível, por que não crio uma query? Ao responder um BOPS, não seria melhor que fosse sua última argumentação para o problema? Se isto ficar claro e bem resolvido não teremos mais aquela ocorrência ou dúvida. Se tivermos que explicar um processo para alguém, que o façamos de tal forma a não gerarmos incógnitas.

Por que ao invés de focarmos nossos esforços para “matarmos” o BOPS, não avaliamos o fonte para evitarmos NOVOS BOPS? Ao resolver uma ocorrência lembre-se de todos os pontos de implicação da sua atividade. O que isso irá impactar no serviço do outro? Sem falar em documentar no Quark!

Vamos trazer o comportamento do parque para o nosso trabalho também. Ao programar vamos nos ater aos detalhes, sermos mais críticos, pensarmos que aquela instrução a mais, significa muito para o sistema e que lá na frente, se tratado com descuido, pode causar problemas.

Tenha convicção que, se agirmos de maneira mais focada aos nossos propósitos, o passeio ou melhor a programação, será muito mais entusiasmada, produtiva e com uma margem de erro

bem menor. Com esse comportamento quem ganha somos nós; Microsiga!. Só assim teremos mais tempo de irmos ao parque no final de semana.

Lembre-se que não adianta decidirmos passear no parque do Ibirapuera no domingo, e não estarmos com a cabeça voltada para o passeio, ao invés disso pensarmos no trabalho, na DLLl que não comunica, no BOPS que não foi baixado, pois se assim for, estaremos tão voltados para outros fins que não curtiremos o passeio. Pense que para passear, ou melhor, programar, a regra também é valida, não adianta nem ao menos tentarmos se não estivermos concentrados para isso.

Enfim, quer uma prova de trabalho em equipe com um alto nível de qualidade e detalhes; este manual, que foi constituído em apenas 2 dias, com a colaboração de mais de 20 pessoas, focadas em seus objetivos, se atentando cada um com o seu tema. O resultado? Um trabalho excelente, um documento para nos ajudar a sermos melhores e não errarmos no fácil!

O Que é Fazer um Programa com Inteligência

Precisamos entender, antes de mais nada, o que é inteligência.

Segundo o dicionário Michaelis, inteligência significa:

faculdade de entender, pensar, raciocinar e interpretar;

Compreensão, conhecimento profundo.

De acordo com essa definição, se pretendemos utilizar nosso bem mais precioso em nosso trabalho, vamos precisar desenvolver alguns hábitos:

Devemos estudar o programa antes de começar a desenvolver. Imagine prestar um concurso ou fazer uma prova sem estudar. Vai ganhar um zero na certa! No programa não será diferente!

Fazer um levantamento dos programas que sofrerão as conseqüências das alterações realizadas. Todos esses programas deverão ser testados juntamente com o programa alterado.

Antes de criar uma função, consulte o Help Microsiga ou os colegas de trabalho, pois esta função já pode ter sido criada.

Ao criar uma função, certifique-se de que no cabeçalho conste algumas informações básicas como: descrição da função, sintaxe, definição dos parâmetros e autor. É comum ao desenvolver uma função, utilizarmos outra já pronta como exemplo, e neste momento o “copiar/colar” nos faz esquecer de alterar estas informações.

Imagine se alguém desenvolver uma função inconsistente e esquecer de trocar o seu nome no cabeçalho. Devemos assumir a responsabilidade de nossos atos.

Ao fazer a documentação das alterações realizadas, certifique-se de que as informações estão claras, não só para o seu entendimento mas para que os colegas não percam tempo tentando entender-las.

Ao realizar os testes, defina critérios. Antes de começar defina onde quer chegar. Não basta consistir suas alterações. O fato de suas alterações estarem funcionando como previstas não garante a não existência de erros.

Não limite-se a testar sua alteração na base que você utilizou durante o desenvolvimento, pois você criou o ambiente perfeito para que o programa funcione.

Pode parecer um pouco trabalhoso passar por estes processos no decorrer do desenvolvimento do sistema, mas se medidas como estas não forem tomadas, o que era extremamente simples se tornará extremamente trabalhoso.

Programando Simples, mas Certo

Qual profissional da área de informática ainda não se deparou com um código fonte que parecia estar escrito em outro dialeto mesmo com todo conhecimento adquirido naquela linguagem, este fato geralmente ocorre pela má utilização de sintaxes complexas que nem sempre significam um bom funcionamento do sistema.

Um profissional da área de informática não possui nenhum modelo padrão para desenvolver os seus algoritmos, porém é necessária a aplicação da ética profissional para que se possa desenvolver algoritmos de maneira simples e correta, este conceito se baseia nos seguintes aspectos :

Entender qual o objetivo do processo em questão

Analisar a melhor forma de desenvolver um algoritmo que seja de fácil manutenção.

Utilizar comandos e sintaxes que utilizem o máximo de simplicidade e clareza possível.

Erros que Podem ser Evitados

Existem alguns erros que com um pouco de atenção, podem ser evitados, tais como:

Verifique se a variável está declarada antes do uso;

Ao declarar uma variável, verifique qual a necessidade de ter essa variável e qual o tipo e a sua classe;

Classifiquem as funções e os procedimentos conforme a necessidade, como por exemplo, na declaração de um array, defina o seu tamanho e no uso verifique se o elemento existe;

Salve a ordem e a área e o registro do arquivo que será utilizadopara que no final do processo se recupere estes valores;

Evite retornar da função antes do seu final, ou seja, crie preferencialmente um único retorno;

Valide sempre o retorno do ponto de entrada;

Quando for gravar um arquivo que utiliza campos de outros arquivos, posicione todos os arquivos e registros antes de iniciar a gravação, e descreva o alias do campo;

Utilize de arquivo CH nas strings para localização;

Quando possível utilize a linguagem SQL, pois minimiza o tempo de execução em muitos processos.

A Importância de Programas Documentados

Todos sabemos o quanto é difícil elaborar e manter uma documentação técnica atualizada, ainda mais aqui na Microsiga, cuja dinâmica dos acontecimentos muitas vezes impede que isso seja

viabilizado. Diante desse cenário, o que nos resta? Obviamente que pelo menos os programas sejam documentados, bem documentados.

Documentar bem, não significa que tenhamos que escrever dezenas de linhas de comentários a cada linha de código. Significa que os comentários têm passar alguma informação relevante. Vemos comentários assim: “compara A com B” e só. Isso é óbvio, a leitura do código já nos diz isso. A documentação deve se ater a conceitos, por exemplo: “Se A for maior que B, o arquivo de saldos será atualizado, caso contrário o registro será rejeitado para que o saldo não fique negativo.”. Isto sim transmite alguma informação.

Também se pode utilizar desse recurso para fazer lembretes a fatos importantes que, se forem deixados de lado, podem comprometer o funcionamento das rotinas.

Por exemplo: “Ao acionar esta função, o arquivo XXX DEVE estar posicionado no índice 1”.

E os cabeçalhos? Quantos programas são “aproveitados” e nem sequer o nome do autor é trocado? Se o analista X tivesse escrito todos programas que aparece como autor ele deveria ter começado na época do Charles Babage. O cabeçalho das funções de conter o nome na dita cuja, autor, data de criação, uma descrição sumária de sua funcionalidade, a sintaxe e por último, mas não menos importante, a descrição dos argumentos de entrada e saída. A respeito desse último item deve-se ter especial atenção nas manutenções, pois novos argumentos são criados e nem sempre são declarados nessa seção da documentação do cabeçalho, isso é muito grave.

No IDE do PROTHEUS existem opções bastante interessantes para nos auxiliar nessa tarefa. Experimente as opções Inserir, Documentação de cabeçalho e Inserir, Documentação de Explicação.

Existe ainda um tipo de documentação que nem sempre é observada, é aquela inerente ao próprio código. Programas cujas variáveis são declaradas como nX, cVAR1, dAUX, nNUM, etc., são extremamente difíceis de entender e pior, manter. É conveniente que os nomes das variáveis retratem seu uso ou destino. Por exemplo: dDataDeS ou dDataDeE. Segundo as convenções da Microsiga, variáveis do tipo DATA devem ser iniciadas pela letra “d”. Assim “Data”, não acrescenta nada ao entendimento do que a variável representa. Nos sobrou o “dES” e o “dEE” para informar para que diados serve a bendita variável. Será saída, solução, saldo? Entrada, Estorno, Estoque? Que tal isso: dSeguro e dEntrega?

Enfim, como foi dito, não é preciso escrever um livro a cada programa, basta ser objetivo e se colocar na posição de quem não conhece o programa tão pouco o assunto. Algum dia você mesmo poderá estar nessa posição.

Cabeçalho de Programa / Função

O cabeçalho do programa é utilizado para identificar informações gerais sobre a rotina, seu autor, data, entre outras informações. É importante que esteja preenchida de forma correta e atualizada. Lembre-se de que nada adianta um cabeçalho que não informe nada ou pior ainda, com informações errôneas.

Lembre-se que um bom livro começa com um bom prefácio, e um bom programa começa com um cabeçalho útil e legível.

A manutenção/atualização do cabeçalho é de responsabilidade da última pessoa que alterou o fonte. O cabeçalho de programa padrão da Microsiga contém: rotina, autor, data do desenvolvimento, comentário sintético e sintaxe.

Grupos Relacionados

Principal / Guias de Referência / Como programar Advpl no ERP

Topo da Página